Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 928 – 950

doi:10.55730/1300-0632.4027

CCCD: Corner detection and curve reconstruction for improved 3D surface reconstruction from 2D medical images

Mriganka SARMAH, Arambam NEELIMA*
Department of Computer Science and Engineering, National Institute of Technology, Nagaland, India

Received: 30.06.2023 • **Accepted/Published Online:** 30.09.2023 • **Final Version:** 27.10.2023

Abstract: The conventional approach to creating 3D surfaces from 2D medical images is the marching cube algorithm, but it often results in rough surfaces. On the other hand, B-spline curves and nonuniform rational B-splines (NURBSs) offer a smoother alternative for 3D surface reconstruction. However, NURBSs use control points (CTPs) to define the object shape and corners play an important role in defining the boundary shape as well. Thus, in order to fill the research gap in applying corner detection (CD) methods to generate the most favorable CTPs, in this paper corner points are identified to predict organ shape. However, CTPs must be in ordered coordinate pairs. This ordering problem is resolved using curve reconstruction (CR) or chain code (CC) algorithms. Existing CR methods lead to issues like holes, while some chain codes have junction-induced errors that need preprocessing. To address the above issues, a new graph neural network (GNN)-based approach named curvature and chain code-based corner detection (CCCD) is introduced that not only orders the CTPs but also removes junction errors. The goal is to improve accuracy and reliability in generating smooth surfaces. The paper fuses well-known CD methods with a curve generation technique and compares these alternative fused methods with CCCD. CCCD is also compared against other curve reconstruction techniques to establish its superiority. For validation, CCCD's accuracy in predicting boundaries is compared with deep learning models like Polar U-Net, KiU-Net 3D, and HdenseUnet, achieving an impressive Dice score of 98.49%, even with only 39.13% boundary points.

Key words: 3D surface reconstruction, chain codes, corner detection, spline surface, graph neural network

1. Introduction

Three-dimensional (3D) reconstruction of human organs is done using multiple volumetric medical images obtained with methods such as computed tomography (CT) or magnetic resonance imaging (MRI). These images are read by the marching cube (MC) algorithm [1] as three-dimensional data, after which a 3D mesh is generated. Generally, the 3D data are point clouds stored as multidimensional arrays; however, efficient space partitioning based on a kd-tree is also possible [2]. Other point cloud processing algorithms include the ball pivoting algorithm [3], alpha shape reconstruction [4], and Poisson surface reconstruction [5]. However, these algorithms additionally require face normal and vertex normal direction, due to which the MC algorithm is preferred.

The limitation associated with the MC algorithm is that, in order to generate a surface, it requires a detailed segmentation of the region of interest (ROI). It also uses a large number of images, typically a few

*Correspondence: neelimaarambam@yahoo.co.in

hundred. The dimension of a single image used determines the size of the entire volume. Typically, a volume of 100 gray-scale images of dimensions of 256 × 256 requires 64 MB. In the LiTS dataset, the average size of one image is nearly 35 KB. The higher the accuracy demanded in reconstruction, the higher the size of the image volume, which leads to memory utilization in the order of hundreds of MB. Not only memory but also 3D mesh generation from such image data becomes difficult on a low-end computing platform [6]. Moreover, once the 3D mesh is rendered, the outcome of the MC algorithm is still a rough textured mesh. To remove that roughness, external smoothing algorithms such as BiNF [7], AGConv [8], and DNF-Net [9] are required, thereby also increasing the overall computation time. In [10], the authors proposed the generation of a fine wireframe from a point cloud, which reduces the redundant edges and noise. However, the generated wireframe possesses intersections that are not feasible for spline-based reconstructions.

The proposed method in this paper serves two purposes. First, it will use fewer corner points instead of an entire volume of images to render a 3D surface. For an image of dimensions of 512 × 512, a single corner point can be accessed by $[i]\,[j]$ indices. Thus, each corner is represented as a two-dimensional index that would require at most 64 bits (32 bits each to represent i,j index as integers). Then, for 100 corner points within an image, the memory required is 0.8 KB, and for a volume of 200 such images the maximum memory consumption is 160 KB or 0.15625 MB, which is significantly less compared to the whole image volume as used in the MC algorithm and state-of-the-art segmentation methods such as Polar U-Net [11], KiU-Net 3D [12], and HdenseUnet-liver [13]. Second, the applied methodology is such that no external smoothing algorithm is required.

Previous approaches that were used to generate curves from points include Crust [14], Connect-2d [15], Crawl, Peel, Gathan, nn-Crust [16], and SIGDT [17]. These are known as curve reconstruction (CR) methods. However, these methods are efficient only for dense and uniform point distributions and they generate holes and disjoint segments if random image corners are used.

Alternately, instead of using these CR methods, a twofold approach is applied. First, the corner points (CPs) are detected using common techniques such as Harris [18], Shi–Tomasi [19], SUSAN [20], and FAST [21]. Second, using a B-spline [22], these points are connected. An advantage of using B-splines over CR methods is that they are independent of the distribution of points and work well for both densely and sparsely distributed points.

However, a limitation of B-splines is that they can only be generated from an ordered set of points. While there are infinite ways to order CPs, there are only a few ways in which properly connected CPs can reasonably determine the object boundary.

The objective of this study is to use a graph neural network (GNN) [23] to align the detected CPs. Thereafter, B-splines are used to generate a boundary curve where the CPs are treated as control points (CTPs). Finally, using non-uniform rational B-splines (NURBSs), a 3D mesh is generated.

This paper is structured into 9 further sections, namely a literature review (Section 2), background study (Section 3), materials and methods (Section 4), the proposed algorithm (Section 5), results and discussion (Section 6), a summary (Section 7), limitations (Section 8), and conclusions (Section 9).

The contributions of this paper are to propose a novel curvature and chain code-based corner detection (CCCD) algorithm that does the following:

(a) Identification of a new problem in chain code junctions called isolated corners.

(b) Proposing the use of a supervised GNN to eliminate isolated corners from boundary curves and produce correct chain codes.

(c) Identification of CPs from corrected chain codes using local curvature.

(d) The use of B-splines and CPs to approximate the target boundary in 2D.

(e) Stacking of 2D B-splines using NURBSs for 3D surface reconstruction.

2. Literature review

Curve-based reconstructions are called spline curvature reconstruction or spline-based segmentation [24–29]. Splines are broadly categorized as Bezier curves [22], B-splines [22], and T-splines [30] and are created using a set of CTPs and a knot vector. The knot vector determines the impact of CTPs on a curve span. Basic Bezier curves lack the ability of local refinement by adding or removing CTPs. In contrast, B-splines allow for local refinement by allowing adjustments to the control points in specific regions. Thus, Bezier curves are not preferred for spline-based reconstructions. However, T-splines offer an improvement over B-splines by allowing the existence of T-junctions in the splines. Even though T-spline surfaces can reduce the model complexity, it is assumed that a 3D mesh is already formed and constructed with polygons. As the objective in this paper is to reconstruct an unknown 3D mesh from 2D medical images, T-splines are not suitable for this particular scenario. Jover et al. [31] proposed B-spline reconstruction from sparse and noisy points. They assumed that CTPs are ordered, but the order of boundary points is not guaranteed unless the segmentation itself is based on the predefined spline. Zhang et al. [25] proposed a spline-based segmentation by defining the CTPs for their segmentation problem manually. Gagan et al. [26] initialized two concentric circle templates for optical disc segmentation. Their method is specific to optic disk segmentation and is not generalizable. On the other hand, a proposed T-spline surface fitting algorithm [32] performs segmentation over a region, but the method is suited to 3D meshes only. Rahali et al. [29] implemented Chan–Vese [33] contour modeling using B-spline level-set representation. Their method is based on an elliptical shape seed and a given target boundary. A common trend in spline-based segmentation is the use of a shape prior. These shape priors are made from existing knowledge of the segmentation region.

In the existing literature, corner detection (CD)-based spline reconstruction stands out as an underexplored area. Although several CD methods such as Harris, SUSAN, Shi–Tomasi, and features from the accelerated segmented test (FAST) were compared in a previous study [34], their application to B-spline surfaces was not addressed in any recent work.

Existing CR algorithms such as Crust [14], Connect-2d [15], Crawl, Peel, Gathan, nn-Crust [16], and SIGDT [17] do not perform well on sparse data. These methods are based on proximity and continuity measures. A common technique for fitting a curve to a random point is the moving least squares (MLS) algorithm. Another technique followed in CR methods is using a ball of α radius and a predetermined sample size within the ball to determine connectivity with previous landmarked points. Another paradigm to generate ordered curves involves the use of directional chain codes (CCs) [28]. CCs are also used for CD. Some such CD methods from CCs are the Medioni–Yasumoto [35], Beus–Tiu [36], weighted-K-curvature [28], Rosenfeld–Johnson [37], and Cheng–Hsu [38] corner detectors. One limitation associated with CCs is that sparsely distributed and far-away points produce errors. It is observed that during the conversion of CCs to B-splines, some points remain outliers, particularly at junctions, resulting in disjoint curves and isolated corners. Hence, CCs require some

preprocessing. If such isolated corners are eliminated, then CCs have the potential to be highly beneficial in spline surface reconstruction from corners.

SimKGC [39] applies text-based graph-linking to improve its performance as compared to embedding-based graph-linking. By introducing negative learning, text-based graph learning is improved over embedding-based learning. Relphormer [40] represents every node in a network using the graph topology (t) and textual descriptions (d) uniquely identified by a (t,d) pair. This method performs well in question-and-answer (QA) tests. In [41], graph structure features were used as heuristics and the heuristic was a target structure. A GNN was trained to learn the heuristics. In [42], individual node features were learned from sequence and structure data, and then, using a graph convolution network (GCN) and pretrained features, new features were synthesized. The process reduced the overall training time of existing methods. GNN-based medical image segmentation was also implemented for retinal vessel segmentation [43] using a full-scale UNet.

For liver segmentation on the LiTS17 dataset, the state-of-the-art (SOTA) approach is UNet 3+ [44], which has a DSC of 0.9675, where the authors implemented deep supervision on different scales. Some other SOTA methods in liver segmentation are Polar-UNet [11], KiU-Net 3D [12], and H-DenseUNet [13], with DSCs of 0.9302, 0.9423, and 0.9650, respectively. For whole brain segmentation on the Brats2020 dataset, the SOTA is nnunet [45], which achieved a DSC of 0.8895. Similarly, for lung segmentation, Gite et al. [46] achieved a DSC for 0.9630 and a mean IOU of 0.95.

3. Background study

Definition 1 (Chain code) *A chain code (CC) is a manner of lossless compression and a representation of image contours.*

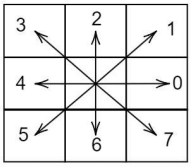

Figure 1. A basic 3×3 neighborhood of an 8-directional chain code.

Figure 1 shows how pixels are numbered from 0 to 7 using 8-directional CCs [28]. The CCs not only identify boundary points but also determine the order of the pixels, thereby generating a chain of points connecting the next pixel to a current point. Defining an order for points is an important task for any B-spline-based curve generation. An example of a chain-coded liver boundary is shown in Figure 2.

Definition 2 (Graph neural network) *A graph neural network (GNN) is a neural network architecture that represents nodes as vertices and synapses as edges. Directly connected nodes are called adjacent nodes. All adjacent nodes are neighbors of the activation node.*

GNNs [23] closely resemble Markov chain processes and recurrent neural networks. Most GNN tasks are categorized as graph-level tasks or node-level tasks and operate directly on graph-like data. Neural networks working on images apply either convolution or dense fully connected perceptron-like operations over an $n \times n$ kernel whereas GNNs perform aggregation and convolution only for the neighboring nodes as desired. Figure

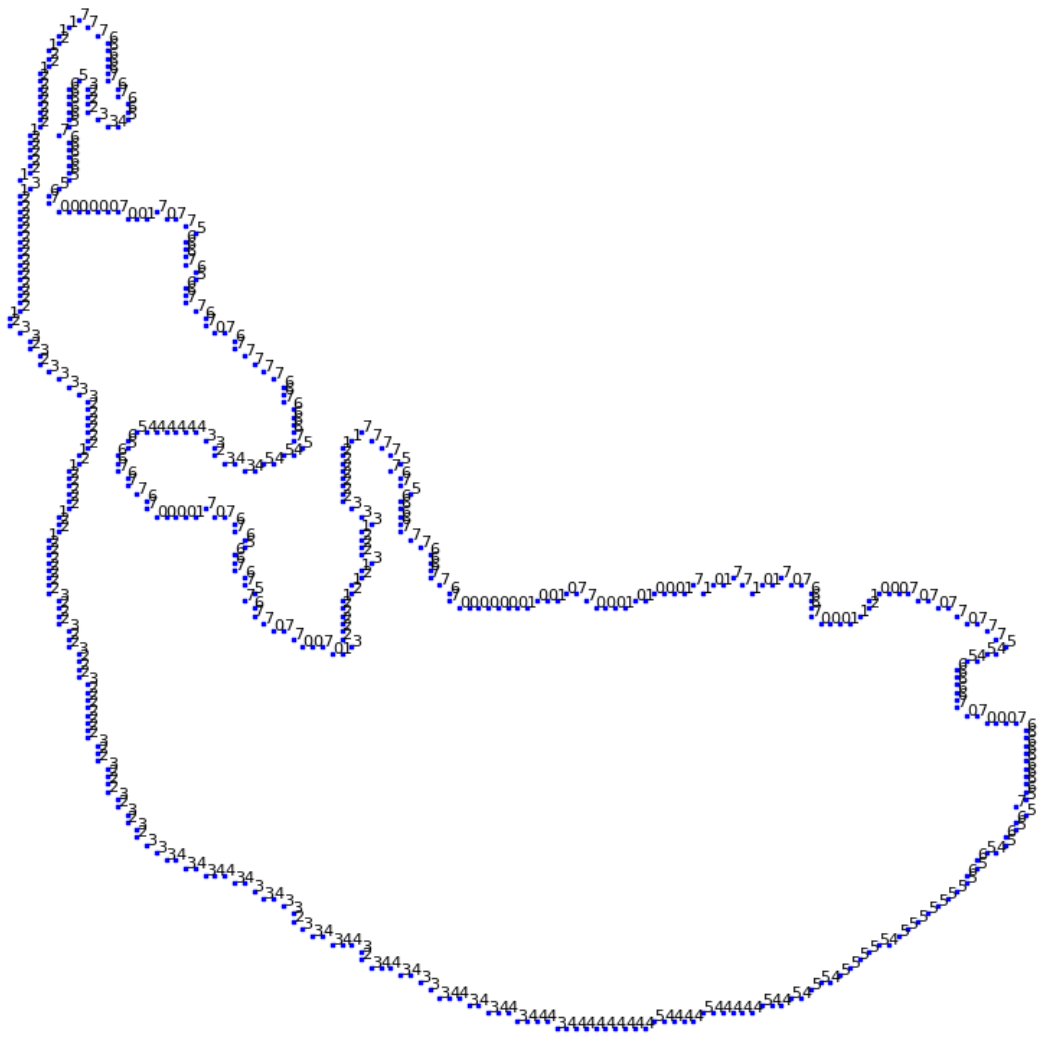

Figure 2. Sample chain-coded liver boundary.

3 shows 25 nodes and 72 links for an image of size 5×5. This architecture can be extended to represent any $n \times n$ image.

Let x, o, l, l_N be the tensor describing the states, outputs, labels, and target labels of individual nodes. The GNN can then be written as in Eq. (1), where F_w and G_w are respectively the global transition function and global output function:

$$\begin{aligned} x &= F_w\left(x(t), l\right) \\ o &= G_w\left(x, l_N\right) \end{aligned} \tag{1}$$

In [23], Scarselli et al. defined an iterative scheme for updating the states and output, as shown in Eq. (2):

$$\begin{aligned} x(t+1) &= F_w\left(x, l\right) \\ o(t+1) &= G_w\left(x(t), l_N\right) \end{aligned} \tag{2}$$

Learning in a GNN is measured according to the loss function ς shown in Eq. (3) [23], where G_i denotes a

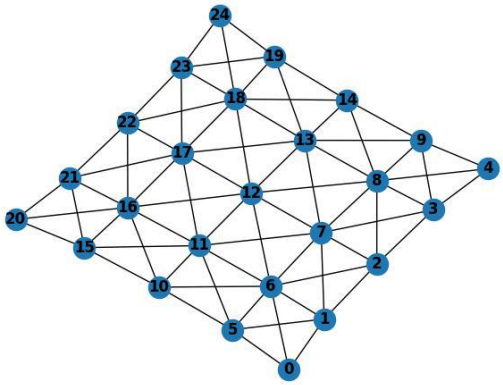

Figure 3. Sample 5×5 grid of nodes and synaptic links in a GNN. Every node is linked to its neighbor only. Thus, neural activation and operations such as convolution happen between these neighbors only.

subgraph of G. N_i and E_i denote a set of nodes and edges. i, j denotes grid position, and $t_{i,j}$ denotes the target label at node i, j. Pixels that are to be included in the CC are labeled '+1' and otherwise '–1'. Similarly, the nodes have binary input features $+1$ or -1. p is the number of nodes lengthwise and q_i is the number of neighbors of node i.

$$\varsigma = \{(G_i, n_{i,j}, t_{i,j}) \mid, G_i = (N_i, E_i) \in G; n_{i,j} \in N_i; t_{i,j} \in \Re^m, 1 \leq i \leq p, 1 \leq j \leq q_i\} \tag{3}$$

Definition 3 (Corner point from curvature) *A corner point (CP) is either a maximum or minimum curvature point within a window of k neighboring pixels. In order to evaluate a corner, the contour must have CCs and there should be no gaps.*

Based on [37], a CP is identified by performing smoothing on the chain across a window of width k as in Eq. (4):

$$\begin{aligned} a_{ik} &= (x_i - x_{i+k}, y_i - y_{i+k}) \\ b_{ik} &= (x_i - x_{i-k}, y_i - y_{i-k}) \end{aligned} \tag{4}$$

The k cosine vector is then computed as in Eq. (5). The best value for k is determined from $c_{i1}, c_{i2}, \cdots, c_{im}$ when $c_{im} < c_{im-1} \cdots < c_{ik} \not< c_{ik-1}$, i.e., when the values tend to decrease the value of k is found. Point i is a corner when c_{ik} is a local maximum or minimum such that $|i - j| \leq k/2$.

$$c_{ik} = \frac{a_{ik} \cdot b_{ik}}{|a_{ik}|\,|b_{ik}|} \tag{5}$$

Definition 4 (B-spline curves and surfaces) *A B-spline is a curve with segments, fitted to a set of CTPs such that a change in the position of any point has a local effect on a selected set of segments only.*

$$P(t) = \sum_{i=1}^{n+1} N_{i,k}(t) P_i \qquad , \; t_{min} \leq t \leq \mathrm{t_{max}} \tag{6}$$

$$N_{i,1}(t) = \begin{cases} 1, & t_i \le t_{i+1} \\ 0, & otherwise \end{cases} \tag{7}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \tag{8}$$

B-spline curves are defined with a basis function as shown in Eq. (6) [54] for n+1 CTPs, $P_1, P_2, \cdots, P_{n+1}$, where $t_i \le t_{i+1}$. The basis function definition is $N_{i,k}$. t is the knot vector and k is the degree of the curve. Uniform B-spline basis functions are as shown in Eq. (9):

$$\begin{aligned} N_{i,k}(t) &= \tfrac{1}{6}(2+t)^3 && -2 < t \le -1 \\ N_{i,k}(t) &= \tfrac{1}{6}\left(4 - 6t^2 - 3t^3\right) && -1 < t \le 0 \\ N_{i,k}(t) &= \tfrac{1}{6}\left(4 - 6t^2 + 3t^3\right) && 0 < t \le 1 \\ N_{i,k}(t) &= \tfrac{1}{6}(2-t)^3 && 1 < t < 2 \\ N_{i,k}(t) &= 0 && otherwise \end{aligned} \tag{9}$$

B-spline surfaces are extended B-splines in three-dimensional space. The inner summation of Eq. (10) describes a curve using points in a horizontal plane. The outer summation is applicable after stacking multiple splines on top of one another and describing a set of CTPs such that some vertical curves can be approximated along the z-axis.

$$P(s,t) = \sum_{i=1}^{m+1} \sum_{j=1}^{n+1} P_{i,j} N_{i,k}(s) N_{j,l}(t), s_{min} \le s \le s_{max},\ t_{min} \le t \le t_{max} \tag{10}$$

4. Materials and methods

4.1. Isolated corners

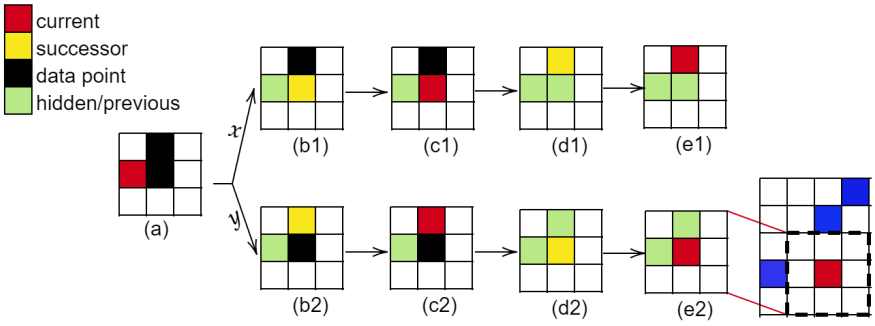

Figure 4. Sample 3×3 neighborhood generating isolated corners. From current state 'a' there are two paths showing two possible directions of traversal through boundary pixels.

An isolated corner is a point in a curve that cannot be linked to a new successor. Figure 4 explains the steps through which isolated corners are created. The visualization includes different colored pixels to represent

specific states: red for current pixels, black for neighborhood pixels, green for previously visited pixels, and yellow for next or successor pixels. In stage a, there are two possible paths, x and y, to choose a successor. Suppose path y is chosen. Then, at stage b2, the (yellow) pixel is the successor and the current pixel is marked as visited (green). At stage c2, the new successor becomes the current pixel (red). The current (red) pixel can only see the adjacent neighbor (black) within the neighborhood as the previous (green) pixels remain hidden to avoid recursion. Thus, the only option from hereon is to choose the black pixel as the next successor. Once selected, the successor is shown in stage d2 (yellow) and the current pixel becomes hidden (green). At stage e2, the current (red) pixel is seen to be isolated without any adjacent pixel causing the CC allocation to generate errors. Thus, in path y, both adjacent points are already visited (stages b2 and d2). On the other hand, by choosing path x, no isolated point errors are generated. However, backtracking on the occurrence of isolated corners incurs additional cost and hence a more reliable method is sought through the use of a GNN.

4.2. Removing isolated corners

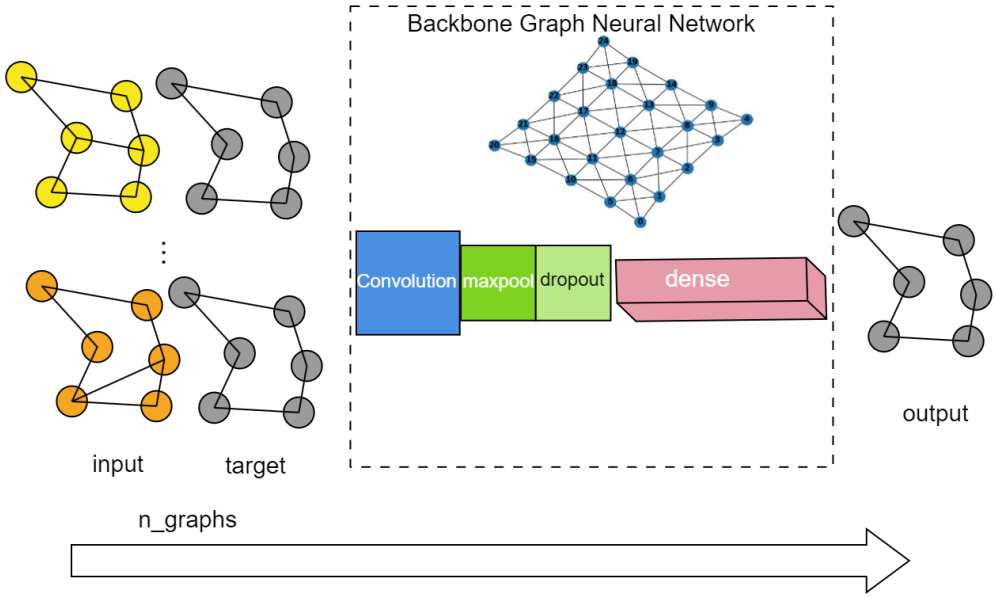

Figure 5. Proposed GNN-based model to autocorrect chain codes.

Figure 5 demonstrates the application of GNN convolution to aggregate the neighboring synapses as shown in Eq. (11) [53] where BN is batch normalization, W is the weight matrix, b is bias, $Dropout = 0.5$, Act is an activation function such as ReLU, and Agg is an aggregate function. The problem is to identify correct links (at most two adjacent neighbors) and remove unwanted links (more than two neighbors). Thus, the feature vector contains a binary link label (+1 for true links and –1 otherwise). Hence, the problem is treated as a link prediction problem.

$$x_i^{'} = Agg\left(\left\{Act\left(Dropout\left(BN\left(x_j W + b\right)\right)\right), j \in N\left(i\right)\right\}\right) \tag{11}$$

4.3. Dataset and platform description

Three datasets are used, namely LiTS17 [47], RIDER Lung PET-CT [48], and UPENN-GBM [49] (Figure 6). As the human brain has complex curvatures compared to some other larger organs, the UPENN-GBM dataset is used to validate the reconstruction ability of the proposed model. From each dataset, one CT volume is extracted, and 198 brain CT scans are used from the UPENN dataset. The data preparation step starts with CT image dilation [50] with an XOR operation on a 5×5 kernel. Thereafter, the dilated image is Gaussian-blurred and, using an edge detection method [51], the original boundary is identified.

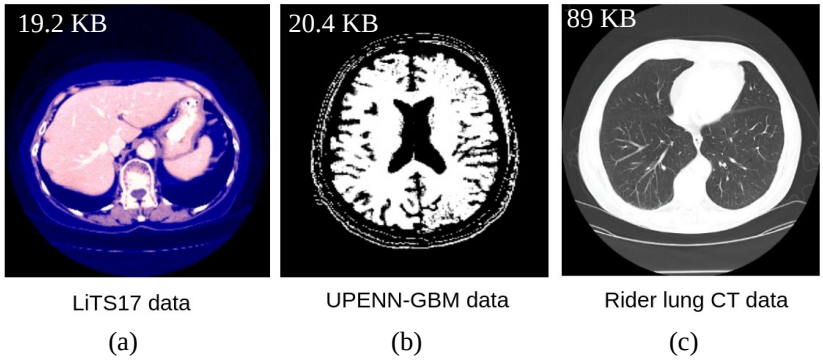

Figure 6. Raw images from the used datasets.

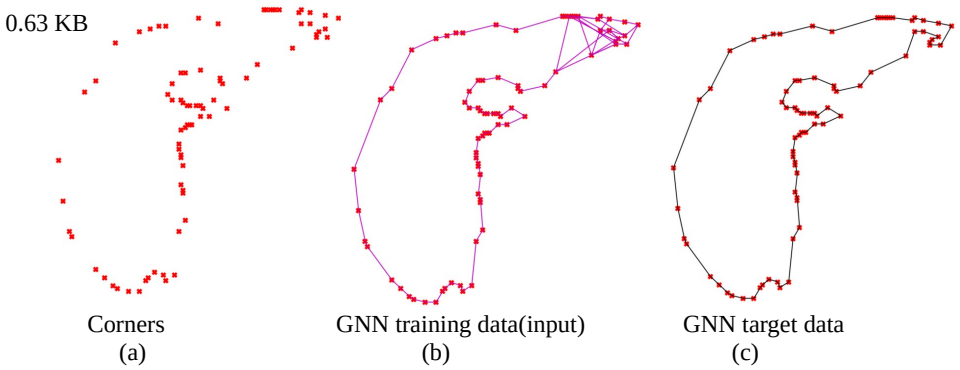

Figure 7. Examples of used training datasets.

The proposed model uses corners only (Figure 7a), while to train the GNN, random corner edges are generated for training (Figure 7b). In total, 290 graphs are trained, comprising different liver contours, out of which 200 are used for training, 40 (5%) for validation, and 50 for testing. The dataset [52] provides training and original data. A total of 1917 different corner points are identified and an adjacency matrix of 1917×1917 is registered for every graph. The total primary memory occupied by 209 training and target adjacency matrices is 6.4 GB.

To implement the GNN, the Spektral 1.0 package is used on a Tesla T4 GPU with 16 GB memory and 1.59 GHz clock rate. For other computational tasks, the Python programming language is used. Python opencv

version 4.7.0 is installed. The curve reconstruction methods are implemented on the Ubuntu 20 operating system using publicly available source codes from the respective authors.

4.4. Evaluation metrics

True positive (TP) is the number of foreground pixels shared by the ground truth and predicted segmentation. True negative (TN) is the number of background pixels shared by the ground truth and predicted segmentation. False positive (FP) is the number of incorrect pixels predicted as foreground by the predicted segmentation. False negative (FN) is the number of incorrect pixels predicted as background by the predicted segmentation. The evaluation metrics used in this paper are TP rate (Eq. (12)), FP rate (Eq. (13)), TN rate (Eq. (14)), FN rate (Eq. (15)), accuracy (Eq. (16)), misclassification (Eq. (17)), precision (Eq. (18)), specificity (Eq. (19)), Dice score (Eq. (20)), and Jaccard index (Eq. (21)).

$$TPrate = \frac{TP}{TP + FP + TN + FN} \tag{12}$$

$$FPrate = \frac{FP}{TP + FP + TN + FN} \tag{13}$$

$$TNrate = \frac{TN}{TP + FP + TN + FN} \tag{14}$$

$$FNrate = \frac{FN}{TP + FP + TN + FN} \tag{15}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

$$Misclassification = \frac{FP + FN}{TP + FP + TN + FN} \tag{17}$$

$$Precision = \frac{TP}{TP + FN} \tag{18}$$

$$Specificity = \frac{TN}{TN + FP} \tag{19}$$

$$Dice = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{20}$$

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{21}$$

5. Proposed algorithm

A step-by-step visualization of the proposed model is shown as a block diagram in Figure 8. The description is given in Algorithm 1. The process starts with a raw CT volume as input, preferably converted to gray scale. For any image of dimensions $n \times n$, the first task is performing Gaussian blur ($O\left(n^2\right)$) and dilatation ($O\left(n\right)$) operations (see Section 4.3). After that, the CC ($O\left(m\right)$) is generated from m boundary points (see

Algorithm 1: Proposed CCCD algorithm to generate 3D surfaces from 2D CT images.

```
i ← OriginalCTimage[N];
j ← None;
CC ← None;
corner ← None;
spline[] ← None;
surface ← None;
for i do
    if i notpreprocessed then
        i ← Dilation(GaussianBlur(i));
        j ← Boundary(i);
    else
        if ChainCode(j) ← Null then
            CC ← ChainCode(j);
            if CCcontains(isolatedCorners(CC)) then
                CC ← GNNbasedFilter(CC);
                corner ← identifyCorners(CC);
                spline[i] ← createSpline(corner);
            end
        end
    end
end
surface ← createSurface(spline);
```

Definition 1) and examined for the presence of junctions (see Section 4.1). If junctions are present, the GNN-based module ($O\left(m^2\right)$) is initiated and the CC is reevaluated for the newly predicted boundary such that all the junctions are removed (see Section 4.2). Once done, the CC curvature-based corner method ($O\left(m^2\right)$) is invoked and respective corners are calculated (see Definition 3). Using this ordered set of corner points, a B-spline is generated (see Definition 4) ($O\left(m^2\right)$). All images are then stacked on top of one another to evaluate a new B-spline in the z-axis. Thus, at the end of the algorithm, a 3D surface is produced. The complete time complexity for the proposed algorithm is ($O\left(n^2\right) + O\left(m^2\right)$), where $n \times n$ represents the image dimensions, m is the number of corners, and O is worst-case time complexity.

6. Results and discussion

This section is divided into several subsections. Section 6.1 analyzes alternative FUSE models, Section 6.2 compares the proposed model with these alternative FUSE methods, Section 6.2.1 analyzes the effects of parameters on the proposed model, Section 6.3 compares segmentation methods with the predicted boundary from the proposed model, Section 6.4 analyzes existing algorithms that use B-splines to generate boundaries from corners, and Section 6.5 compares the existing methods in curve reconstruction using medical images. One of the important implications from the results in Section 6.1 is that common CD methods detect false corners. For this reason, true boundary points are insufficient in enclosing the organ boundary (Figure 9).

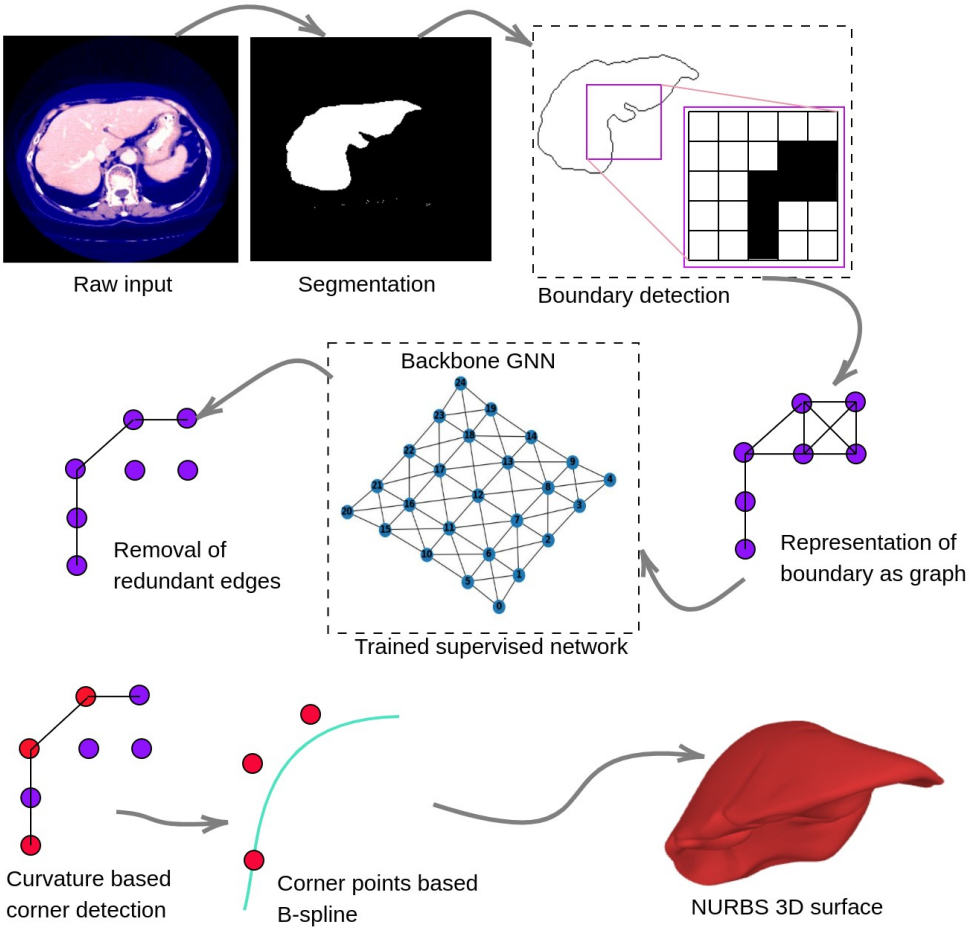

Figure 8. Block diagram of proposed model.

For comparative analysis, three CD methods, namely Harris, Shi–Tomasi, and FAST, are used. These CD methods are inherently not capable of generating a curve. Hence, they are combined with B-splines to create three fused methods, namely FUSE1 (Harris corner detection + B-spline), FUSE2 (Shi–Tomasi corner detection + B-spline), and FUSE3 (FAST corner detection + B-spline). These methods can be thought of as alternative approaches.

6.1. Result of FUSE1, FUSE2, and FUSE3

The Harris corner method exhibits false positives (red dots) close to the ground truth (blue dots) (Figure 9a), and 101 CPs are detected on the boundary, with the corresponding generated B-spline shown as a black curve. Thus, the FUSE1 Dice score is 87.54%. Moreover, unwanted self-intersections are also prominently seen (Figure 9a). Similarly, in Figure 9b, it is seen that the Shi–Tomasi corner method also detects false positives (red dots) in close proximity to the ground truth (blue dots). The actual number of CPs on the boundary is 11; thus, the FUSE2 Dice score achieved is 79.75%. Finally, Figure 9c shows the performance of the FAST algorithm, where

the number of CPs detected is comparatively higher than the previous two alternatives. For the test image used, 212 CPs are detected on the boundary. Thus, FUSE3 achieved a Dice score of 93.15%.

Comparing false positive rates shows that the Shi–Tomasi method has the highest percentage of false positives in detecting corners. From the original 53 corners, only 11 lie on the actual boundary (Table 1), which in turn reduces the overall Dice score.

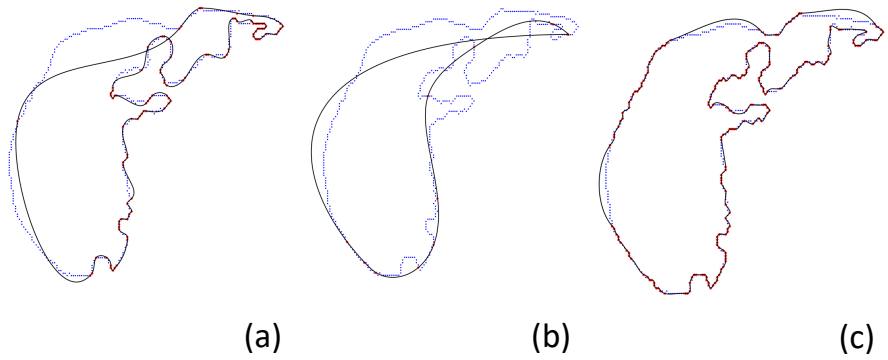

Figure 9. Opencv-based performance of (a) FUSE1: block size = 3, Sobel aperture = 7, free parameter k = 0. 04; (b) FUSE2: maximum control points set to 400, quality = 0.05, minimum distance set to 1; (c) FUSE3: blue dots are original boundary, red dots are corners, black curve is predicted boundary.

Table 1. Percentage drop from identified corner points to actual corner points on liver boundary.

Algorithm	No. of original points	No. of points on boundary	% false corners
Harris	716	232	67.5%
Shi-Tomasi	53	11	79.24%
Fast	212	212	**0%**

The Python opencv module with a predefined Harris corner detection function is used. The function accepts three parameters, namely block size, aperture, and free parameter k. To reduce the false positives, several combinations of these parameters are tried. As the CP count decreases, the CPs are spaced far apart and the spline self-intersects. Finding the right choice of parameter is a manual task and tedious. For the Shi–Tomasi corner method, the opencv *goodFeaturesToTrack* module is called and the three parameters given are maximum number of points, quality, and minimum distance.

6.2. Results of the proposed model (CCCD) compared to FUSE1, FUSE2, and FUSE3

As the proposed model initializes parameter k to detect corners, the optimum value differs for different images. Considering the liver boundary, four different k values are used (see Definition 3). For k = 5 where 121 CPs are generated (Figure 10a), where the predicted corners are red dots, the actual boundary is blue dots, and the generated boundary curve is in black, the Dice score is 95.83%. Figure 10b shows the result of the CCCD with k = 20, where 67 CPs are generated and the Dice score is 88.75%. In another case, Figure 10c shows the result

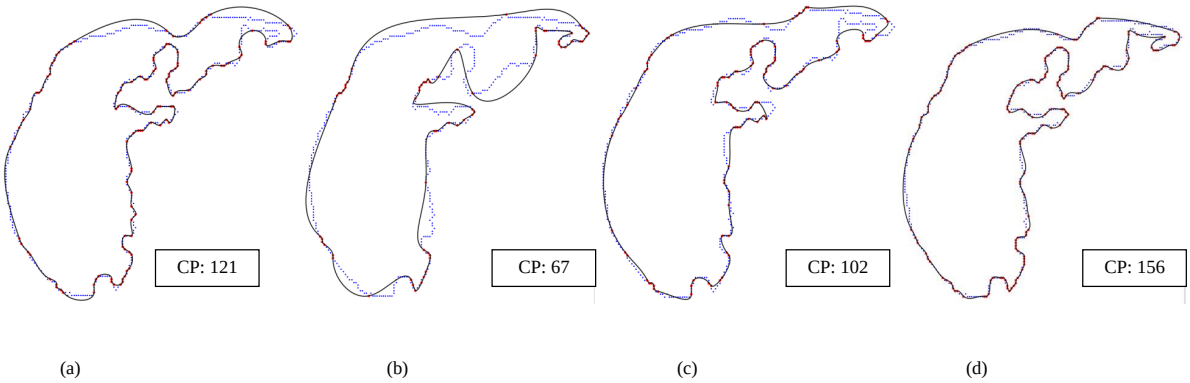

Figure 10. Performance of proposed CCCD: (a) reconstructed B-splines from 105 corner points; (b) reconstructed B-splines from 67 corner points; (c) reconstructed B-splines from 102 corner points; (d) reconstructed B-splines from 156 corner points. Blue dots are original boundaries, red dots are corners, black curves are predicted boundaries.

of the proposed CCCD with k = 7, where 102 CPs are generated and the Dice score is 96.00%. Finally, with k = 3 where 156 CPs are generated (Figure 10d), the Dice score is 98.49%.

Table 2. Comparison of proposed CCCD model versus FUSE1, FUSE2, and FUSE3.

Algorithm	Parameters	DSC	CP
FUSE1 (Figure 9a)	block size = 2, Sobel aperture = 3, k = 0.04	95.4%	232
FUSE2 (Figure 9b)	max. CTP = 400, quality = 0.1, min. dis. = 5	73%	11
FUSE3 (Figure 9c)	-	88.4%	212
Proposed CCCD (Figure 10c)	$k = 7$	96.00%	102
Proposed CCCD (Figure 10d)	$k = 3$	**98.49%**	156

A comparative analysis is presented in Table 2 showing the percentage Dice scores and the numbers of boundary CPs used. The proposed model achieved a Dice score of 98.49%. Comparing the proposed model with the FUSE methods shows that a higher Dice score was achieved even with fewer CPs (Table 2).

Comparing the CCCD with the FUSE methods over various metrics shows that the CCCD has the highest TP rate and lowest MC score (Figures 11a–11d). Among four different k values, the CCCD ($k = 3$) (Figure 11a) has the highest Dice score of 98.49%, and its respective FP and FN rates of 0.0029 and 0.00348 are lowest compared to FUSE1, FUSE2, and FUSE3. Subsequent k values are represented in Figure 11b ($k = 5$), Figure 11c ($k = 7$), and Figure 11d ($k = 20$). When $k = 20$, the FP rate of the CCCD increases, resulting in a low Dice score. The choice of parameter k is not predetermined, but any value below 20 gives the best results.

6.2.1. Performance of CCCD with varying parameter k

Figure 12 shows that for the same k applied on two different datasets, different results are produced. Figure 12a is a brain boundary that obtained a Dice score of 98.27%, whereas for the same k, the predicted liver boundary (Figure 12b) achieved a Dice score of 87.29%. Hence, the change in k affects the final outcome, as well.

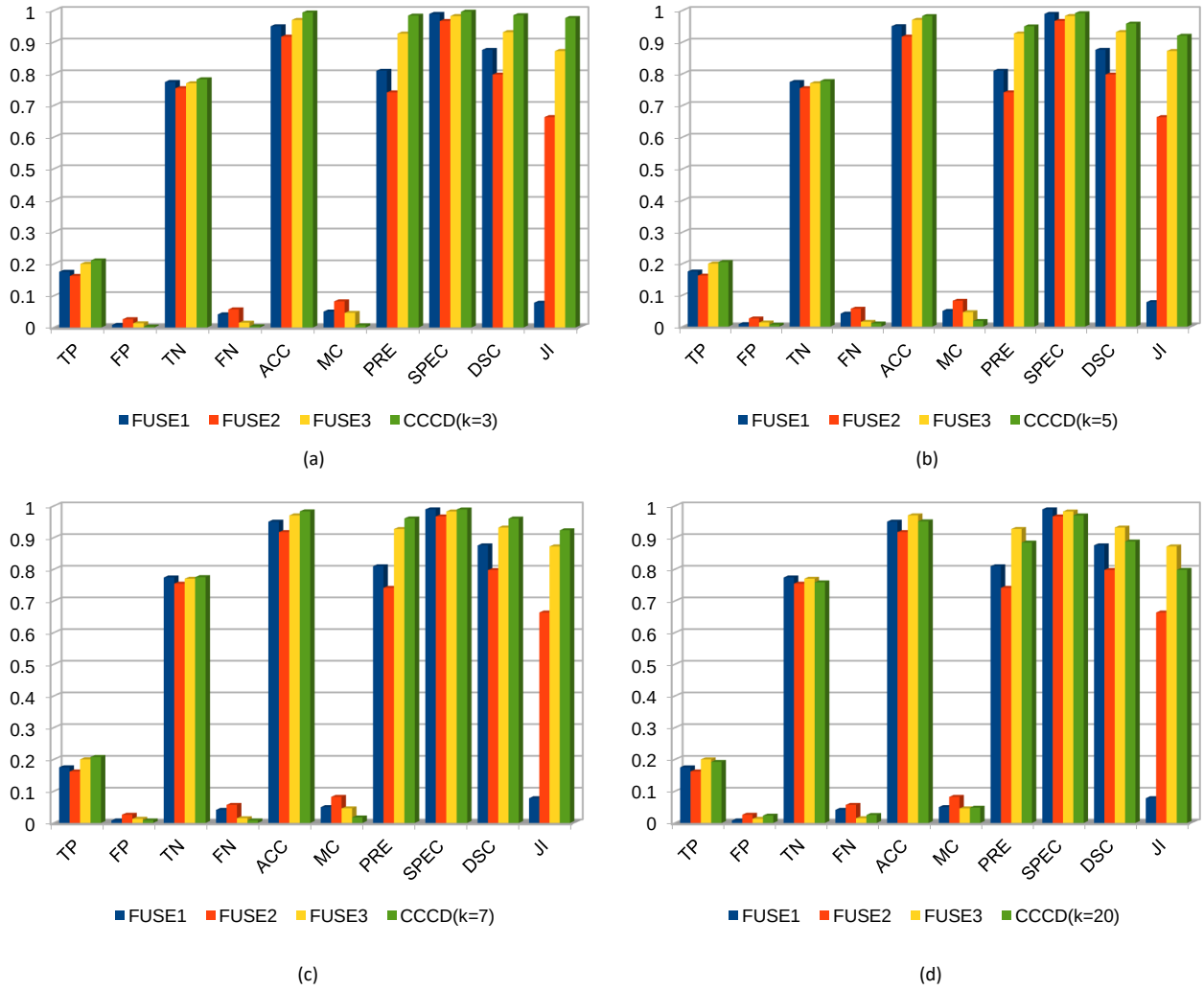

Figure 11. Performance comparison of FUSE1 (Harris + B-spline), FUSE2 (Shi–Tomasi + B-spline), and FUSE3 (FAST + B-spline) with the proposed CCCD ($k \in \{3, 5, 7, 20\}$). The comparison metrics are true positive (TP), false positive (FP), true negative (TN), false negative (FN), accuracy (ACC), misclassification (MC), precision (PRE), specificity (SPEC), Dice score (DSC), and Jaccard index (JI).

Figure 13 compares the differences in various metrics for the LiTS17 and UPENN-GBM datasets when parameter k is changed. These comparison metrics are accuracy (ACC) (Figure 13a), Dice score (DSC) (Figure 13b), and Jaccard index (JI) (Figure 13c), respectively. In these figures, it is observed that as the value of k increases, ACC, DSC, and JI decrease. For the LiTS data, there is a sudden drop in the accuracy rate as k approaches 20, and this can be attributed to the choice of the minimum local curvature as the criterion for identifying corners. Figure 13d shows that the sudden changes in ACC rate, Dice score, and Jaccard index are corrected by applying the local maximum as a criterion for identifying corners. When $k = 3$, the Dice score is 0.9849 on the LiTS and 0.9842 on the UPENN-GBM dataset.

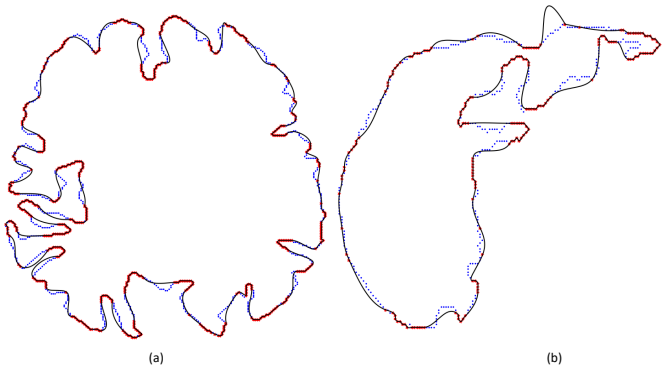

Figure 12. Performance of CCCD on two different datasets for parameter $k = 20$.

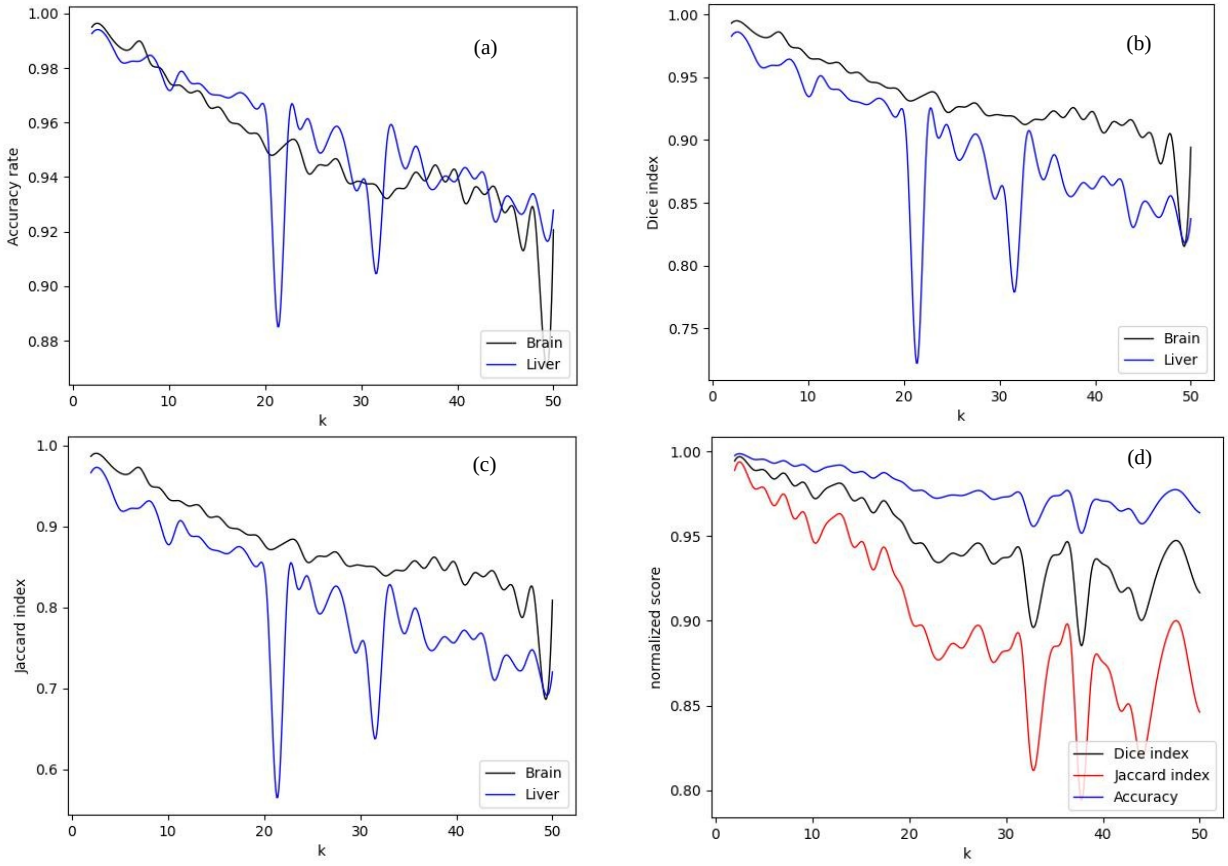

Figure 13. Performance comparison of CCCD with itself at different values of k when tested on LiTS17 and UPENN-GBM data using minimum local curvature for (a) accuracy (ACC), (b) Dice score (DSC), and (c) Jaccard index (JI); (d) DSC, JI, and ACC rate using maximum curvature on LiTS data.

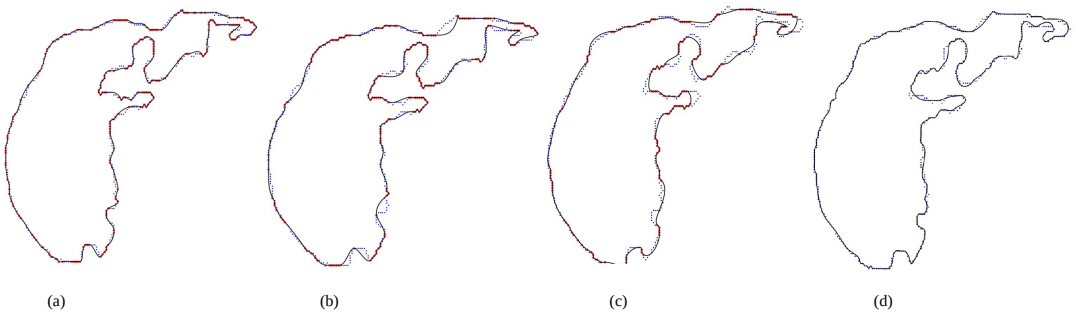

Figure 14. Performance in segmentation task of (a) proposed CCCD (k=3), (b) proposed CCCD (k=7), (c) proposed CCCD (k=12), and (d) HdenseUnet-Liver [11]. Blue dots are original boundaries, red dots are corners, black curves are predicted boundaries.

Table 3. Comparison of SOTA with CCCD using maximum curvature. Bold font shows the best performance.

Number of boundary pixels consumed	k	Model	Year	%Dice	Memory consumed by a single image
483	-	Polar U-Net [11]	2021	93.02	19.2 KB
483	-	KiU-Net 3D [12]	2020	94.23	19.2 KB
483 (Figure 14d)	-	HdenseUnet-liver [13]	2017	96.5	19.2 KB
133 (Figure 14c)	12	Proposed (CCCD)	-	**96.37**	**1.03 KB**
217 (Figure 14b)	7	Proposed (CCCD)	-	**96.00**	**1.69 KB**
189 (Figure 14a)	3	Proposed (CCCD)	-	**98.49**	**1.47 KB**

6.3. Performance comparison of liver segmentation with state-of-the-art (SOTA) methods

From Table 3, it can be seen that the proposed model is comparable to the SOTA deep learning methods in the segmentation task. The proposed CCCD with $k = 3$ achieved an impressive 98.49% Dice score. A significant reduction in memory is also seen when using the CCCD as it requires only corner information instead of the entire segmented image. Three cases are taken as examples. Figure 14a shows the predicted boundary by CCCD when $k = 3$, Figure 14b shows the predicted boundary by CCCD when $k = 7$, Figure 14c shows the predicted boundary by CCCD when $k = 12$, and Figure 14d shows the boundary predicted by H-DenseUNet. Thus, the number of boundary pixels consumed by Polar U-Net [11], KiU-Net 3D [12], and H-DenseUNet-liver [13] is 483, which is significantly higher compared to the proposed CCCD method. The proposed model successfully achieves 96.37% DSC from 133 boundary points only, reducing the meta data content by 72.46%. The initial image of 19.2 KB contains redundant information; however, the CCCD can efficiently perform well with 1.03 KB of input data.

6.4. Performance of SOTA methods in curve prediction from corner points

Figure 15 shows the predicted boundary as described by Marsousi et al. [55] using a seed spline and some corner points. The algorithm has no knowledge of the actual boundary. Here, the CPs used are from the Shi–Tomasi

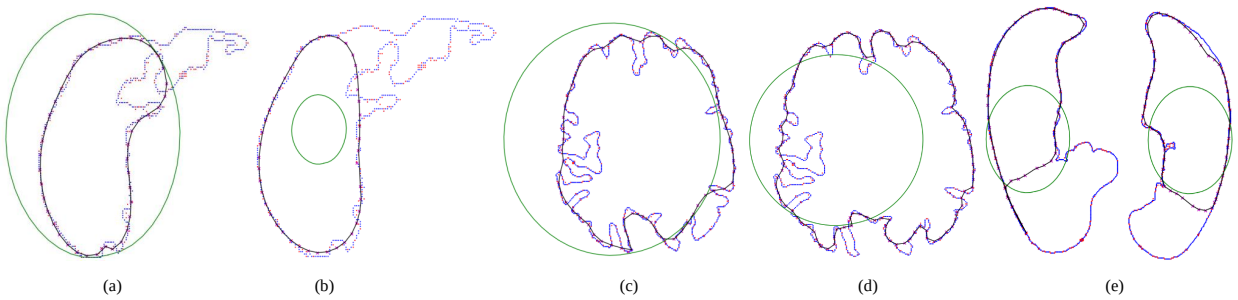

Figure 15. Wrapping of segmented boundaries (blue dots) by an initial seed B-spline (green). The initial seed includes the boundary corners (red dots) from the Shi–Tomasi corner detection method. The final B-spline boundary curve is shown in black. a, b, c, d, and e were produced using the work of Marsousi et al. [55]. × in magenta color represents predicted spline control points.

CD method, which are set as target CTPs to guide the seed towards the CPs. It is observed that the position and radius of the seed are crucial for the correct identification of the boundary. Thus, with changes in the dimensions and location of the radius, the predicted boundary is affected. As shown in Figures 15a and 15b, two different seed radii are used for the liver boundary, and the predicted boundary (black) is not improved with a decrease in seed radius. However, in Figures 15c and 15d, a slight improvement in the predicted boundary (black) is observed with a decrease in seed radius, but the DSC achieved is 79.78%. Figure 15e shows the predicted lung boundary, where a major section of the lungs is still not enclosed.

6.5. Performance comparison of curve reconstruction with SOTA methods

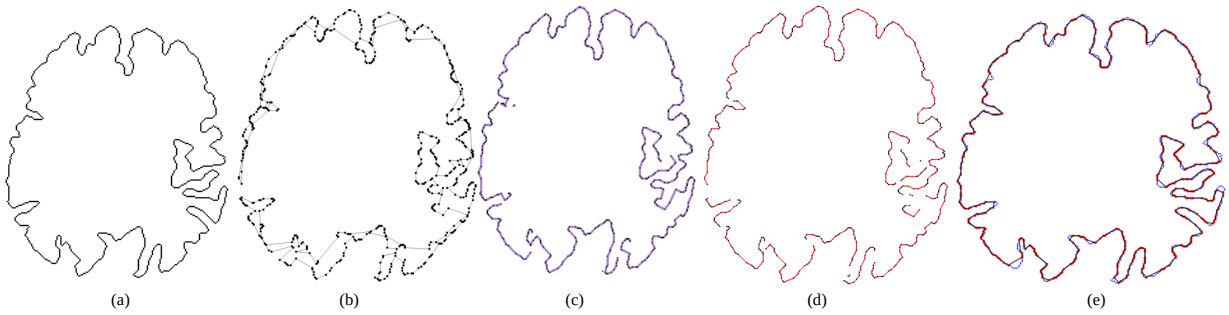

Figure 16. Performance of existing 2D curve reconstruction methods for given Harris corner points: (a) ground truth, (b) Connect2d, (c) Crawl, (d) nn-Crust, (e) proposed CCCD construction.

In this section three existing CR algorithms, namely Connect2d, Crawl, and nn-Crust, are applied to reconstruct a curve using the Harris CP approach. The boundaries produced by these CR methods are compared with the proposed CCCD. The compared methods produce disjointed segments that do not match the ground truth (Figures 16b–16d). Figure 16a portrays the ground truth. Figure 16b is the output of Connect2d, where multiple segments are seen that deviate from the original boundary. Figure 16c is the output from the Crawl

algorithm, where some sections still show disjoints that are absent in the ground truth. Finally, Figure 16d shows the result of the nn-Crust algorithm, which also produces unwanted disjoints and holes. The proposed model outperforms the existing methods without producing any disjointed segments (Figure 16e).

Table 4. Comparison of different curve reconstruction methods in connecting corner points.

Model	No. of connected segments	No. of disconnected segments
Ground truth	**1**	**0**
Connect2d	2	2
Crawl	4	4
nn-Crust	12	12
Proposed CCCD	**1**	**0**

Table 5. Time comparison in generating a curve from corner points.

Algorithm	Corner detection time (s)	Curve reconstruction time (s)
Harris	1.885	-
Shi–Tomasi	1.304	-
FAST	**0.93**	-
Connect2d	-	3.44
Crawl	-	6.87
nn-Crust	-	5.34
Proposed CCCD	2.889	**0.395**

Table 4 provides a comparative study of the number of connected and disconnected segments in the ground truth and boundaries generated by the existing CR methods. Connect2d predicts a boundary bearing two holes, Crawl predicts four holes in the boundary, and nn-Crust predicts 12 holes in the boundary, whereas the ground truth has zero holes. The CCCD generates a single connected segment without any disjoints, thereby connecting all boundary points successfully as compared to the other methods.

B-spline generation from corner points is a two-phase task where first the corners are detected and then they are connected as a boundary. Table 5 compares running times for converting corners into a fully closed boundary curve. The proposed CCCD method not only excels in corner detection but also demonstrates comparable running time. Even though FAST detects corners in 0.93 s, the CCCD detects corners in 2.889 s and also generates an enclosing boundary in an additional 0.395 s. Thus, the entire process is completed in 3.284 s. This feature is a benefit of the CCCD compared to other methods.

6.6. 3D surface reconstruction

The ultimate goal is to transform the ordered CPs into a 3D surface. Figure 17 shows a 3D liver surface generated by stacking the ordered points from multiple boundaries using the proposed method. This is done using the Python Visualization Toolkit (VTK). The time required for reconstruction is 4.287 s. Figure 17a is the surface reconstruction using the marching cube algorithm, where the roughness of the surface is visible in

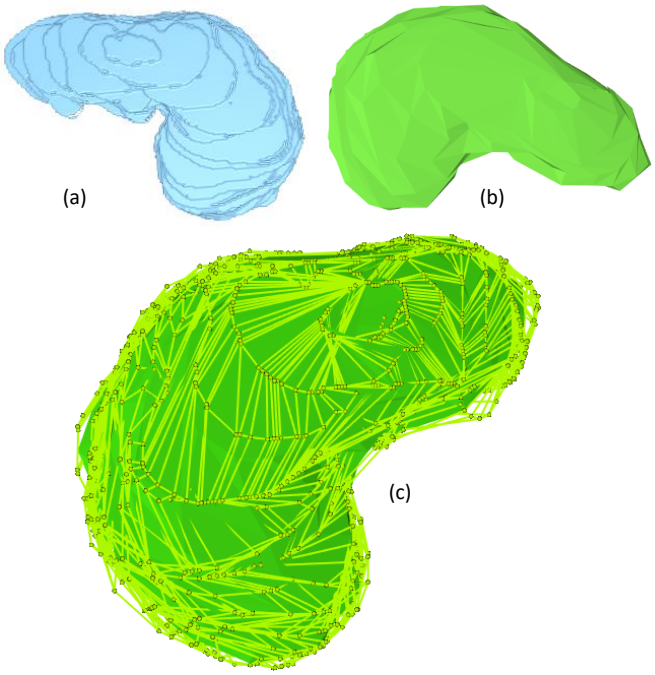

Figure 17. Reconstructed 3D surface of a human liver from the LiTS dataset.

the form of a grid-like structure. Figure 17b is the surface generated by the CCCD, where grid-like structures are not present. The 3D surface is depicted in dark green while the control points and grid form the light green curves that enclose it (Figure 17c).

7. Summary

As segmentation is an important preprocessing task in 3D reconstruction from medical images, SOTA segmentation methods use high-resolution images where each image is 19.2 KB. The proposed CCCD model has shown considerable improvement in DSC over the SOTA medical image segmentation methods even when the input data are reduced below 2 KB, containing only corner points. It has also been seen that training the GNN with minimum image information in the form of corner points reduces the cost of memory by 92.34%.

8. Limitations

Some of the limitations of the proposed model are as follows:

(a) As the optimum number of corner points depends on the organ's shape and size, the choice of the number of corner points is not automated.

(b) As the choice of optimum corner points is not automated, manual intervention is needed.

(c) As the model employs deep neural networks such as the GNN, the performance of the proposed model is based on the quality of the training samples used.

(d) The primary memory required for storing the adjacency matrix for 209 graphs exceeds 6 GB. For large graphs and better performance, higher system requirements are needed.

9. Conclusion

The paper has explored a method to significantly decrease the amount of information in medical images by focusing on the corners of a specific organ. It has evaluated three corner detection techniques, namely Harris, Shi–Tomasi, and FAST, and revealed that these methods generate false positive corners. The proposed model, however, achieves higher accuracy in identifying true positive corners along the image boundary. Despite using fewer ground truth boundary pixels, the model attains a better Dice score. This implies that the proposed technique could potentially reduce the necessary CT volume information when using corner points as metadata.

The method makes use of CC advantages in identifying curves and arranging curve points. Additionally, it employs a GNN model to identify CC errors and aid in the conversion of pixels to splines. This approach effectively reconstructs a 3D mesh from image boundaries and outperforms existing curve reconstruction methods such as Connect2d, Crawl, and nn-Crust, as these methods tend to create gaps with sparse point distributions.

In the future, the optimal number of corner points required for a 3D mesh can be automated irrespective of the image dataset. Intersecting B-splines can also be studied to replicate tissues and organs of complex 3D shapes such as the characters 'A,' 'H,' or 'X.'

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

[1] Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphics 1987; 21 (4): 163-169.

[2] Sayar A, Eken S, Öztürk O. Kd-tree and quad-tree decompositions for declustering of 2D range queries over uncertain space. Frontiers of Information Technology & Electronic Engineering 2015; 16: 98-108.

[3] Bernardini F, Mittleman J, Rushmeier H, Silva C, Taubin G. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics 1999; 5 (4): 349-359.

[4] Bernardini F, Bajaj CL. Sampling and reconstructing manifolds using alpha-shapes. In: Canadian Conference on Computational Geometry; Kingston, ON, Canada; 1997.

[5] Kazhdan M, Bolitho M, Hoppe H. Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing; Cagliari, Italy; 2006.

[6] Sarmah M, Neelima A. Quantization and surface smoothing of 3D meshes with truncated vertex coordinates: a central probability measure and Bayesian averaging approach. Signal, Image and Video Processing 2023 (in press).

[7] Wei M, Yu J, Pang WM, Wang J, Qin J et al. Bi-normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics 2014; 21 (1): 43-55.

[8] Wei M, Wei Z, Zhou H, Hu F, Si H et al. Agconv: Adaptive graph convolution on 3D point clouds. IEEE Transactions on Pattern Analysis and Machine Intelligence 2023 (in press).

[9] Li X, Li R, Zhu L, Fu CW, Heng PA. DNF-Net: A deep normal filtering network for mesh denoising. IEEE Transactions on Visualization and Computer Graphics 2020; 27 (10): 4060-4072.

[10] Tan X, Zhang D, Tian L, Wu Y, Chen Y. Coarse-to-fine pipeline for 3D wireframe reconstruction from point cloud. Computers & Graphics 2022; 106: 288-298.

[11] Benčević M, Galić I, Habijan M, Babin D. Training on polar image transformations improves biomedical image segmentation. IEEE Access 2021; 9: 133365-133375.

[12] Valanarasu J, Sindagi V, Hacihaliloglu I, Patel V. KiU-net: Overcomplete convolutional architectures for biomedical image and volumetric segmentation. IEEE Transactions On Medical Imaging 2021; 41: 965-976.

[13] Li X, Chen H, Qi X, Dou Q, Fu C et al. H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes. IEEE Transactions on Medical Imaging 2018; 37: 2663-2674.

[14] Amenta N, Bern M, Eppstein D. The crust and the beta-skeleton: combinatorial curve reconstruction. Graphical Models and Image Processing 1998; 60: 125-135.

[15] Ohrhallinger S, Mudur S. An efficient algorithm for determining an aesthetic shape connecting unorganized 2D points. Computer Graphics Forum 2013; 32: 72-88.

[16] Ohrhallinger S, Peethambaran J, Parakkat A, Dey T, Muthuganapathy R. 2D points curve reconstruction survey and benchmark. Computer Graphics Forum 2021; 40: 611-632.

[17] Marin D, Ohrhallinger S, Wimmer M. SIGDT: 2D curve reconstruction. Computer Graphics Forum 2022; 41: 25-36.

[18] Derpanis KG. The Harris Corner Detector. Toronto, Canada: York University, 2004.

[19] Shi J. Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; Seattle, WA, USA; 1994. pp. 593-600.

[20] Smith SM, Brady JM. SUSAN—a new approach to low level image processing. International Journal of Computer Vision 1997; 23 (1): 45-78.

[21] Trajković M, Hedley M. Fast corner detection. Image and Vision Computing 1998; 16 (2): 75-87.

[22] Harn D, Baker M. Computer Graphics (2nd Edition). Delhi, India: Prentice Hall of India, 1999.

[23] Scarselli F, Gori M, Tsoi A, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Transactions on Neural Networks 2008; 20: 61-80.

[24] Zhang A. NeuronSegBACH: Automated neuron segmentation using B-spline based active contour and hyperelastic regularization. Communications in Computational Physics 2020; 28 (3): 1219-1244.

[25] Gagan J, Shirsat H, Kamath Y, Kuzhuppilly N, Kumar J. Automated optic disc segmentation using basis splines-based active contour. IEEE Access 2022; 10: 88152-88163.

[26] Pawar A, Zhang Y, Anitescu C, Rabczuk T. Joint image segmentation and registration based on a dynamic level set approach using truncated hierarchical B-splines. Computers & Mathematics with Applications 2019; 78: 3250-3267.

[27] Gao J, Wang Z, Xuan J, Fidler S. Beyond fixed grid: learning geometric image representation with a deformable grid. In: Computer Vision-ECCV 2020 16th European Conference; Glasgow, UK; 2020. pp. 108-125.

[28] Liu H, Srinath M. Corner detection from chain-code. Pattern Recognition 1990; 23: 51-68.

[29] Rahali R, Dridi N, Salem Y, Descombes X, Debreuve E et al. Biological image segmentation using region-scalable fitting energy with B-spline level set implementation and watershed. IRBM 2022; 43: 640-657.

[30] Deng J, Chen F, Li X, Hu C, Tong W et al. Polynomial splines over hierarchical T-meshes. Graphical Models 2008; 70: 76-86.

[31] Jover I, Debarre T, Aziznejad S, Unser M. Coupled splines for sparse curve fitting. IEEE Transactions on Image Processing 2022; 31: 4707-4718.

[32] Lu Z, Jiang X, Huo G, Ye D, Wang B et al. A fast T-spline fitting method based on efficient region segmentation. Computational and Applied Mathematics 2020; 39: 55.

[33] Chan T, Vese L. Active contours without edges. IEEE Transactions on Image Processing 2001; 10: 266-277.

[34] Chen J, Zou L, Zhang J, Dou L. The comparison and application of corner detection algorithms. Journal of Multimedia 2009; 4 (6): 435-441.

[35] Medioni G, Yasumoto Y. Corner detection and curve representation using cubic B-splines. Computer Vision, Graphics, and Image Processing 1987; 39: 267-278.

[36] Beus H, Tiu S. An improved corner detection algorithm based on chain-coded plane curves. Pattern Recognition 1987; 20: 291-296.

[37] Rosenfeld A, Johnston E. Angle detection on digital curves. IEEE Transactions on Computers 1973; 100: 875-878.

[38] Cheng F, Hsu W. Parallel algorithm for corner finding on digital curves. Pattern Recognition Letters 1988; 8: 47-53.

[39] Wang L, Zhao W, Wei Z, Liu J. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. arXiv preprint, 2022. arXiv: 2203.02167.

[40] Bi Z, Cheng S, Zhang N, Liang X, Xiong F et al. Relphormer: Relational graph transformer for knowledge graph representation. arXiv preprint, 2022. arXiv: 2205.10852.

[41] Zhang M, Chen Y. Link prediction based on graph neural networks. In: 32nd Conference on Neural Information Processing Systems; Montreal, Canada; 2018. pp. 1-17.

[42] Long Y, Wu M, Liu Y, Fang Y, Kwoh CK et al. Pre-training graph neural networks for link prediction in biomedical networks. Bioinformatics 2022; 38 (8): 2254-2262.

[43] Li Y, Zhang Y, Cui W, Lei B, Kuang X et al. Dual encoder-based dynamic-channel graph convolutional network with edge enhancement for retinal vessel segmentation. IEEE Transactions on Medical Imaging 2022; 41 (8): 1975-1989.

[44] Huang H, Lin L, Tong R, Hu H, Zhang Q et al. UNet 3+: A full-scale connected UNet for medical image segmentation. In: ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing; Barcelona, Spain; 2022. pp. 1055-1059.

[45] Isensee F, Jäger PF, Full PM, Vollmuth P, Maier-Hein KH. nnU-Net for brain tumor segmentation. In: Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 6th International Workshop; Lima, Peru; 2021. pp. 118-132.

[46] Gite S, Mishra A, Kotecha K. Enhanced lung image segmentation using deep learning. Neural Computing and Applications 2023; 35: 22839-22853.

[47] Bilic P, Christ P, Vorontsov E, Chlebus G, Chen H et al. The Liver Tumor Segmentation Benchmark (LiTS). arXiv preprint, 2019. arXiv:1901.04056.

[48] Muzi P, Wanner M, Kinahan P. RIDER Lung PET-CT. The Cancer Imaging Archive. Bethesda, MD, USA; United States National Cancer Institute, 2015. https://doi.org/10.7937/k9/tcia.2015.ofip7tvm

[49] Bakas S, Sako C, Akbari H, Bilello M, Sotiras A et al. The University of Pennsylvania Glioblastoma (UPenn-GBM) Cohort: Advanced MRI, clinical, genomics, & radiomics. Scientific Data 2022: 9: 453.

[50] Soille P. Erosion and dilation. In: Soille P (editor). Morphological Image Analysis: Principles And Applications. Berlin, Germany: Springer, 2004. pp. 63-103.

[51] Sun R, Lei T, Chen Q, Wang Z, Du X et al. Survey of image edge detection. Frontiers in Signal Processing 2022; 2: 826967.

[52] https://github.com/mrigankaresearch/MGS_TJEECS.git

[53] You J, Ying Z, Leskovec J. Design space for graph neural networks. Advances in Neural Information Processing Systems 2020; 33: 17009-17021.

[54] Gordon W, Riesenfeld R. B-spline curves and surfaces. Computer Aided Geometric Design 1974: 95-126.

[55] Marsousi M, Eftekhari A, Kocharian A, Alirezaie J. Endocardial boundary extraction in left ventricular echocardiographic images using fast and adaptive B-spline snake algorithm. International Journal of Computer Assisted Radiology and Surgery 2010; 5: 501-513.