Spine Labeling in Axial Magnetic Resonance Imaging via Integral Kernels

Brandon Miles[a], Ismail Ben Ayed[b,g], Seyed-Parsa Hojjat[c], Michael H. Wang[d], Shuo Li[e], Aaron Fenster[e], Gregory J. Garvin[f]

[a]*Simon Fraiser University, Burnaby, BC, Canada*
[b]*Ecole de Technologie Superieure (ETS), Montreal, QC, Canada*
[c]*University of Toronto, Toronto, ON, Canada*
[d]*McGill University, Montreal, QC, Canada*
[e]*Western University, London, ON, Canada*
[f]*London Health Sciences Centre, London, ON, Canada*
[g]*Corresponding author*

Abstract

This study investigates a fast integral-kernel algorithm for classifying (labeling) the vertebra and disc structures in axial magnetic resonance images (MRI). The method is based on a hierarchy of feature levels, where pixel classifications via non-linear probability product kernels (PPK) are followed by classifications of 2D slices, individual 3D structures and groups of 3D structures. The algorithm further embeds geometric priors based on anatomical measurements of the spine. Our classifier requires evaluations of computationally expensive integrals at each pixel, and direct evaluations of such integrals would be prohibitively time consuming. We propose an efficient computation of kernel density estimates and PPK evaluations for large images and arbitrary local window sizes via *integral kernels*. Our method requires a single user click for a whole 3D MRI volume, runs nearly in real-time, and does not require an intensive external training. Comprehensive evaluations over T1-weighted axial lumbar spine data sets from 32 patients demonstrate a competitive structure classification accuracy of 99%, along with a 2D slice classification accuracy of 88%. To the best of our knowledge, such a structure classification accuracy has not been reached by the existing spine labeling algorithms. Furthermore, we believe our work is the first to use integral kernels in the context of medical images.

Keywords: Spine labeling, magnetic resonance imaging, integral kernels, geometric priors, probability product kernels.

1. Introduction

Radiologic assessment is an essential step in managing patients with spinal diseases or disorders. Magnetic resonance imaging (MRI) is commonly utilized for evaluating the inter-vertebral discs and other soft tissue structures [1], while cortical bone is better seen on Computed Tomography (CT). Nonetheless, the focus of lumbar spine CT is more often on the discs than on the bone. Accurate detection and labeling of different spinal structures is vital, as many interventions require precise anatomic information [2, 3, 4, 5, 6, 7, 8, 9]. Surgical mishaps could occur if the spinal level is not reported accurately. For instance, in MRI[1], benchmarking the axial-view slices facilitates the quantification and level-based reporting of common inter-vertebral disc displacements such as protrusion, extrusion and bulging [1], while labeling sagittal-view slices aids in defining a patient specific coordinate system. Furthermore, such detection and annotation algorithms provide inputs that facilitate significantly other difficult spine image processing tasks such as segmentation [11, 12], registration and fusion [13]. For instance, several recent spine segmentation algorithms assume that a labeling is given [11], or perform a labeling process along with the segmentation [12].

Generating these labels in a manual fashion is tedious, subjective, and time-consuming. Therefore, automating the process is desired and has recently sparked an impressive research effort [12, 2, 3, 4, 5, 6, 7, 8, 9]. Automated labeling of such images is, however, a challenging problem as the field of view (i.e. the number of visible vertebral levels), the distributions of image intensities, and the sizes, shapes, as well as orientations of different spinal structures are highly variable among different patients [2, 3, 9].

[1]MRI is the primary modality to assess disc disorders [1]. Unlike CT, MRI scans depict soft-tissue structures, thereby allowing to characterize/quantify disc displacements [1] and degenerations [10].

There are two major limitations with current automated spine labeling algorithms:

(1) Most of the current algorithms address the labeling problem through intensive training from a manually-labeled data set [2, 3, 7, 8, 9]. Such a training stage aims at learning the shapes, textures and appearances of different spinal structures. This knowledge is then used within a classification or regression algorithm, e.g., support vector machine [7], random forest regression [8] or graphical models [3, 5], to subsequently label different spinal structures in the test image. Such algorithms work very well on data sets that closely match the training data, but would require adjustment/retraining for different data sets or if the imaging modality and/or acquisition protocol are altered (e.g., an algorithm that is trained and built for CT images may not perform well on MRI data [3, 5, 6, 9]). This might impede the use of these algorithms in routine clinical practices, where a particular disorder might be analyzed radiologically using several different imaging modalities/protocols with widely variable imaging parameters (resulting in extremely high variation in image data).

(2) To the best of our knowledge, all of the current spine labeling algorithms focus on the sagittal view [2, 3, 4, 5, 6, 7, 8, 9]. However, the quantification and level-based reporting of common inter-vertebral disc displacements such as protrusion, extrusion and bulging require the radiologist to thoroughly inspect all individual axial slices [1], while visually cross-referencing such axial slices to their corresponding position in the sagittal view. Furthermore, in some cases, only the axial view is available for the patient while, in other cases, the two scans (i.e., axial and sagittal) might be acquired at different time points. In such cases, localizing the spinal structures in different views becomes a challenging task, even for an experienced radiologist, which motivates a *standalone* axial spine detection/labeling algorithm. Such a system would facilitate generating precise radiologic reports.

In this work, we present a robust, near real-time axial MRI labeling algorithm based on a hierarchy of feature levels, where pixel classifications via non-linear probability product kernels (PPK) are followed by classifications of

(i) 2D slices, (ii) individual 3D structures and (iii) groups of 3D structures. The method embeds robust geometric priors based on anatomical measurements that are well known in the clinical literature of the spine [14, 15]. Our classifier requires evaluations of integrals at each pixel. However, direct evaluations of such integrals would be prohibitively time consuming. We propose to use an efficient computation of kernel density estimates and PPKs for large volumes via *integral kernels*. The algorithm is $\mathcal{O}(nz)$, where n is the number of pixels in the image and z is the number of density bins. It can achieve near real-time results with a graphics processing unit (GPU) implementation. Furthermore, it does not require intensive external training. We report evaluations over 32 data sets of T1-weighted 3D MRIs of the lumbar spine, which show a structure classification accuracy of 99%, and a slice classification accuracy of 88%. We believe our structure classification accuracy has not been reached by the existing spine labeling algorithms. It is worth noting that integral histograms/kernels have been used recently in computer vision, in the context of template matching in photographs [16, 17]. We believe, however, that our work is the first to use integral kernels in the context of medical images.

2. Formulation

Our algorithm is based on a hierarchy of feature levels, with the features from the current level used as inputs to the next level. It requires a single user-selected point in one 2D slice of a given spine series. Based on this point, pixels are classified followed by (i) 2D slices, (ii) 3D single vertebra and (iii) 3D multiple vertebrae. The system further embeds robust geometric priors based on spine measurements that are well known in the clinical literature [14, 15], e.g., vertebra height and axial area.

2.1. Efficient Pixel-level Classifications via Integral Kernels

Pixelwise Probability Kernel Matching: We propose a non-linear classifier, which determines whether the neighborhood of each pixel $\mathbf{p}$ matches a target distribution denoted P^L. To provide the initial training, the user selects a single point $\mathbf{p}_o = (x_o, y_o)$ within the vertebral region in a single 2D axial slice in the series; see the example in Fig. 2. Then, prior distribution P^L is learned from a window of size $w \times h$ centered at p_o. Such neighborhood distributions contain contextual information, which provides much richer inputs to the classifier than individual-pixel intensities.

Let $\mathcal{D}_j : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$, $j \in [1 \dots N]$, be a set of input images, which correspond to the axial slices of a given spine series. Ω is the image domain and N is the number of slices in the series. For each $\mathcal{D} \in \{\mathcal{D}_j, j = 1 \dots N\}$ and each pixel $\mathbf{p} : (x, y) \in \Omega$, we seek to create a *non-linear* kernel based classifier by evaluating the following criterion:

$$\text{sign}\left(\phi\left(P_{\mathcal{W}(\mathbf{p}),\mathcal{D}} || P^L\right) - \rho\right) \tag{1}$$

where P^L is an *a priori* learned distribution, ρ is a constant and $P_{\mathcal{W}(\mathbf{p}),\mathcal{D}}$ is the kernel density estimate (KDE) of the distribution of image data $\mathcal{D}$ within a window $\mathcal{W}(\mathbf{p})$ centered at pixel $\mathbf{p} = (x, y) \in \Omega$:

$$P_{\mathcal{W}(\mathbf{p}),\mathcal{D}}(z) = \frac{\sum_{\mathbf{q} \in \mathcal{W}(\mathbf{p})} k_z^{\mathcal{D}}(\mathbf{q})}{C} \quad \forall z \in \mathcal{Z} \tag{2}$$

C is a normalization constant corresponding to the number of pixels within window $\mathcal{W}(\mathbf{p})$ and $\mathcal{Z}$ is a finite set of bins encoding the space of image variables.

$\phi(.||.)$ is a *probability product kernel* [18, 19], which measures the degree of similarity between two distributions:

$$\phi\left(P_{\mathcal{W}(\mathbf{p}),\mathcal{D}} || P^L\right) = \sum_{z \in \mathcal{Z}} \left[P_{\mathcal{W}(\mathbf{p}),\mathcal{D}}(z) P^L(z)\right]^{\gamma}, \quad \gamma \in [0, 1] \tag{3}$$

The higher $\phi(.||.)$, the better the similarity between the distributions. For instance, $\gamma = 0.5$ corresponds to the well-known Bhattacharyya coefficient [19]. The latter is always in the range of $[0, 1]$, with 1 indicating a perfect match between the distributions and 0 corresponding to a total mismatch.

The choice of kernel function $k_z^{\mathcal{D}}$ controls the degree of smoothness of image density estimates (2) within the current 2D slice. One choice is to use a bin counter, which yields the *normalized histogram* of image data within window $\mathcal{W}(\mathbf{p})$ of the current 2D slice: $k_z^{\mathcal{D}}(\mathbf{q}) = 1$ if $\mathcal{D}(\mathbf{q}) = z$ and 0 otherwise. Alternatively, one can use a Gaussian kernel $k_z^{\mathcal{D}}(\mathbf{q}) = \exp \frac{\|D(\mathbf{q})-z\|^2}{\sigma}$, where σ is a fixed parameter that controls how smooth the density estimates are. Experimentally, we did not observe a difference between Gaussian-kernel density and normalized histogram. Therefore, we opted for the latter as it yields a faster implementation. Notice that, at this pixel-level classification stage, the densities estimated at the current 2D slice are not affected by information from adjacent slices. However, in the 3D single-vertebra classification step (section 2.3), we will define a convolution kernel on the slice-level features, thereby combining contributions from several adjacent slices.

Efficient Computation of PPK via Integral Kernels: To embed rich contextual information about the vertebrae/discs, we need to use large-size windows in our classifiers. For large windows, the computation of (3) for each pixel in $\mathcal{D}$ is very expensive computationally if performed by direct evaluation. In the following, we describe an efficient computation of kernel density estimates and PPK evaluations for large images and arbitrary window sizes via integral kernels. Such integral-kernel method can be viewed as an extension of the integral-image method of Viola and Jones [20]. Introduced for the purpose of human face detection, the Viola-Jones method is well-known in computer vision. First, let us recall the integral-image method.

Integral images: Given an image $\mathcal{D}$, the corresponding integral image $\mathcal{I}^{\mathcal{D}}$ is defined as the sum of all pixel intensities to the left and above the current pixel:

$$\mathcal{I}^{\mathcal{D}}(x,y) = \sum_{u \leq x} \sum_{v \leq y} \mathcal{D}(u,v) \tag{4}$$

The sum of intensities of all pixels within an arbitrary rectangular region can be computed from $\mathcal{I}^{\mathcal{D}}$ using only the corners of the rectangle:

$$\sum_{u=x_1}^{x_2}\sum_{v=y_1}^{y_2}\mathcal{D}(u,v)=\mathcal{I}^{\mathcal{D}}(x_1,y_1)+\mathcal{I}^{\mathcal{D}}(x_2,y_2)-\mathcal{I}^{\mathcal{D}}(x_1,y_2)-\mathcal{I}^{\mathcal{D}}(x_2,y_1) \tag{5}$$

where (x_1, y_1) are the coordinates of the upper left corner of the rectangle and (x_2, y_2) are those of the lower right corner. Coordinates (x_2, y_1) correspond to the upper right corner, and (x_1, y_2) to the lower left corner. Since $\mathcal{I}^{\mathcal{D}}$ can be computed efficiently for the entire image and (5) can be computed very efficiently for a given rectangle, this method is very efficient when multiple windows need to be computed from the same image.

Integral kernels: To extend the idea of integral images to integral kernels, and to efficiently compute the PPKs in (3), we build for each slice $\mathcal{D}$ a set of separate kernel images defined over Ω: $k_1^{\mathcal{D}}, k_2^{\mathcal{D}}, \ldots k_z^{\mathcal{D}}, z \in \mathcal{Z}$, with $k_z^{\mathcal{D}}$ the Dirac kernel defined earlier. Then, we compute an integral kernel image based on each $k_z^{\mathcal{D}}$ (see the illustration in Fig. 1):

$$\mathcal{I}_z^{\mathcal{D}}(x,y)=\sum_{u\le x}\sum_{v\le y}k_z^{\mathcal{D}}(u,v) \tag{6}$$

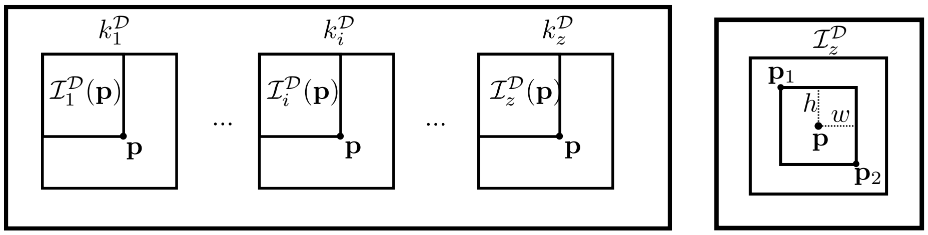
Now, we can easily show that $P_{\mathcal{W}(\mathbf{p}),\mathcal{D}}$ can be computed from the integral kernel images using five simple operations for each $\mathbf{p} = (x, y) \in \Omega$:

$$P_{\mathcal{W}(\mathbf{p}),\mathcal{D}}(z)=\frac{\mathcal{I}_z^{\mathcal{D}}(x_1,y_1)+\mathcal{I}_z^{\mathcal{D}}(x_2,y_2)-\mathcal{I}_z^{\mathcal{D}}(x_1,y_2)-\mathcal{I}_z^{\mathcal{D}}(x_2,y_1)}{(x_2-x_1+1)(y_2-y_1+1)} \tag{7}$$

where $x_1 = x - \frac{w}{2}$, $x_2 = x + \frac{w}{2}$, $y_1 = y - \frac{h}{2}$ and $y_2 = y + \frac{h}{2}$, with w and h being the width and height of $\mathcal{W}(\mathbf{p})$; see the illustration in Fig. 1.

This leads to a very efficient evaluation of classifier (1) for every $\mathbf{p} = (x, y) \in \Omega$, with a computational complexity that (i) is *linear* in the number of pixels in Ω and in the cardinality of $\mathcal{Z}$, and (ii) is independent of the window's size. This method is also highly suited to modern graphics cards because it is amenable to parallel implementations.

Examples of the obtained pixel-level classifications are shown in Fig. 2, where the results are depicted for both vertebrae and disc slices, based on the simple training input from a different slice. To avoid processing the entire slice, only pixels within a region-of-interest R_s around the input point are considered in the pixel-level classification process. The pixel-level classifications we obtained from (1) will be used to further generate slice-level classifications.



$k_1^{\mathcal{D}}$
$\mathcal{I}_1^{\mathcal{D}}(\mathbf{p})$
$\mathbf{p}$
$k_i^{\mathcal{D}}$
$\mathcal{I}_i^{\mathcal{D}}(\mathbf{p})$
$\mathbf{p}$
$k_z^{\mathcal{D}}$
$\mathcal{I}_z^{\mathcal{D}}(\mathbf{p})$
$\mathbf{p}$
...
...
$\mathcal{I}_z^{\mathcal{D}}$
$\mathbf{p}_1$
h
w
$\mathbf{p}$
$\mathbf{p}_2$

Figure 1: Illustration of integral kernels. Left: Computation of integral kernel images $1 \dots z$ for point $\mathbf{p} = (x, y)$; Right: A diagram of the window centered at $\mathbf{p} = (x, y)$ and defined by $\mathbf{p}_1 = (x_1, y_1)$ and $\mathbf{p}_2 = (x_2, y_2)$.

2.2. 2D Slice-level Features

We derive the second level of features from the area of pixels classified as vertebrae in a given 2D slice and from geometric priors. First, we group vertebra pixels into sets of 4-connected regions: S^i, $i = 1, 2, \dots$. These regions are then filtered, building a set $\mathcal{S}$ as follows:

$$\mathcal{S} = \left\{ S^i | \text{area}(S^i) > A_{\min} \text{ and } \|c_i, p_0\| < d_{\max} \right\} \tag{8}$$

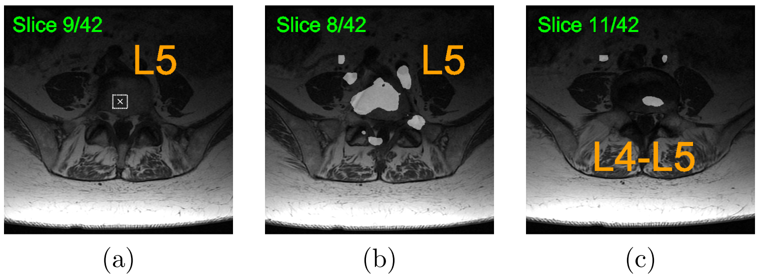


Figure 2: Illustration of pixel-level classifications from axial-view images of the spine. Left: a simple user input on a single slice used for training; Middle: pixel classifications for a vertebra in a new slice different from the training slice; Right: pixel classifications for an inter-vertebral disc on a new slice different from the training slice.

where $A_{\min}$ and $d_{\max}$ are pre-specified geometric priors, which we will define so as to reflect human spine measurements that are well known in the clinical literature [14, 15]. $\|.\|$ denotes the Euclidean distance. If $\mathcal{S} \neq \emptyset$, we use the area of the largest region in $\mathcal{S}$ as a 2D slice-level feature for the next step. Otherwise, we assign value 0 to this feature. We denote this feature A_k for slice $\mathcal{D}_k$.

2.3. 3D Single-Vertebra Classifications

The next level of classification is identifying individual vertebrae in 3D. We start with an input set of adjacent slices $\mathcal{D}_k, k \in [1, \ldots, N]$, in the neighborhood of a vertebra. These slices are all the slices within a geometric prior height H_s, either centered on the initial (user-provided) point or starting at the uppermost (or lowermost) slice of a previously identified vertebra. We start by classifying these slices as vertebra or not. As input, we use the 2D slice-level feature computed at the previous step (A_k). We apply a one-dimensional smoothing filter to the features of adjacent slices: $A_k^s = A_k * K$, where A_k^s is the smoothed data and K is a one-dimensional convolution kernel. Then, a slice is classified as vertebra if $A_k^s > t_{\text{area}}$, where t_{area} is a threshold given by $t_{\text{area}} = c_a \mu_{\text{area}}$, with c_a a user defined factor and μ_{area} the average of areas A_k^s. If the set of adjacent slices classified as vertebrae results in a vertebral height larger than a geometric prior $H_{\min}$, then we classify the 3D set of adjacent slices as vertebra. We define geometric priors H_s and $H_{\min}$ using well-known anatomical measurements of the spine [14, 15].

2.4. Multiple 3D Vertebra Classifications

To improve classification accuracy, we further employ an iterative model update. By using the location of the previously found vertebra, we update distribution P^L (using the center of the previous vertebra) and the search region required for finding the next vertebra. Then, the classification proceeds in both vertical directions of the spine. For the first vertebra, the initial search height, which we denote H_s^0, is defined to be twice the height of a vertebrae (which we fix using prior spine measurements that are well documented in the clinical literature [14, 15]), centered at the input point. For finding subsequent vertebrae, the search range H_s begins at the boundary of the previous vertebrae and extends for the height of a vertebra plus two inter-vertebral disc spaces (also defined with *a priori* known spine measurements [14, 15]). A summary of the procedure is given in Algorithm 1.

```
Algorithm 1: Vertebrae Classification Algorithm

• Given an initial input $\mathbf{p} = \mathbf{p}_0$ and vertebra $V_n = V_0 \in [V_{\min}, V_{\max}]$
    1) Learn the target probability distribution $P^L$.
    2) Set the search height $H_s = H_s^0$.
    3) For each slice $\mathcal{D}_j$ in the set of slices within search hight $H_s$:
        a) Use sign $\left(\phi\left(P_{\mathcal{W}(\mathbf{p}),\mathcal{D}_j}||P^L\right) - \rho\right)$ to classify each pixel $\mathbf{p}$ via integral kernels.
        b) Compute 2D slice-level feature $A_j$.
    4) Compute smoothed features $A_j^s$ for each $\mathcal{D}_j$ within search hight $H_s$.
    5) Using sign $\left(A_j^s - t_{\text{area}}\right)$, find the uppermost/lowermost slices for the current vertebra.
    6) Update the vertical search region $H_s$ and target distribution $P^L$ using $V_n$.
• If $V_n \leq V_{\max}$:
    7) While $V_n < V_{\max}$,
        a) Let $n = n + 1$
        b) repeat steps 3-6
• Else:
    8) Set $V_n = V_0$. Update the vertical search $H_s$ and target distribution $P^L$ based on $V_0$.
    9) While $V_n \geq V_{\min}$
        a) Let $n = n - 1$
        b) Repeat steps 3-6.
```

3. Data description

This retrospective study was approved by the Human Subjects Ethics Board of Western University, with the requirement for informed consent being waived. A total of 32 subjects were included in this study. The series of each subject contains a set of axial T1-weighted MRI slices of the lumbar spine. A total of 102 vertebrae, each corresponding to several 2D slices, were detected/annotated automatically. Furthermore, the algorithm annotated each 2D slice as either vertebra or inter-vertebral disc (The data included 749 slices in total). The slice thickness ranged from 4 to 5 mm and the in-plane voxel spacing ranged from 4.4 to 10 mm. The number of visible vertebrae within each series varied from one subject to another, which makes the problem challenging. These numbers are reported in Table 1.

Table 1: The number of vertebrae visible at each level for the lumbar spine data sets acquired from 32 subjects.

Vertebral Level	L5	L4	L3	L2	L1	T12
Number of Visible Vertebrae	32	32	21	11	5	1

The initial user click was placed on a single axial slice of the L5 vertebra of the lumbar spine. The algorithm labelled the axial slices as either vertebra or inter-vertebral disc; Fig. 3 depicts typical examples.

4. Choice of the parameters and input selection

The geometric parameters were fixed based on spine measurements that are well known in the clinical literature [14, 15]. We defined such geometric parameters in millimeters, so as to ensure independence of voxel spacing. These are the size of search windows $\mathcal{W}(\mathbf{p})$, the minimum classification area $A_{\min}$, the maximum distance $d_{\max}$, the initial search height H_s^0, the subsequent search height H_s, the minimum vertebrae height $H_{\min}$ and the region of interest R_s. Table 2 reports vertebra and disc measurements (height and width) that are known in the literature [14]. Furthermore, based on data for the lumbar spine [15], the major axis length of the vertebrae was found to be about 1.5 times the minor axis length (width of the vertebrae). Based on these values, the cross-sectional area of a vertebra can be overestimated by a rectangle of $1.5w^2_{\text{vertebrae}}$ and underestimated by an oval of size $0.375\pi w^2_{\text{vertebrae}}$, with $w_{\text{vertebrae}}$ denoting vertebra width. Our experimental heights, search ranges and area measurements correspond to these measurements, which can be found in Table 2. The classifier-related parameters were tuned experimentally. Variable γ was set equal to 0.5, which corresponds to the Bhattacharya distance between distributions. Pixel-level classification threshold ρ was set equal to 0.75. The number of bins was experimentally set to be 100. The 1D convolution parameter K was set to [0.3 1 0.3], while the area threshold factor c_a was fixed equal to 0.75.

Table 2: Vertebra and disc measurements as well as overestimates/underestimates for the lumbar vertebrae [14, 15].

Measurement type	Value
Vertebra Height (mm)	27.3 ± 1.2
Disc Height (mm)	8.8 ± 0.9
Vertebrae Width (mm)	34.3 ± 1.8
Vertebrae Area Overestimate (mm^2)	360
Vertebrae Area Underestimate (mm^2)	224

Table 3: Parameter selection.

Parameter	Symbol	value
Pixel Threshold	ρ	0.75
Number of Bins	$\mathcal{Z}$	100
Search Window ($mm \times mm$)	$\mathcal{W}$	12 x 12
Region of Interest ($mm \times mm$)	R_s	80 x 80
Minimum Area (mm^2)	$A_{\min}$	400
Max Distance (mm)	$d_{\max}$	40
Minimum Vertebrae Height (mm)	$H_{\min}$	12.5
Area Threshold Factor	c_a	0.75
Initial Search Height (mm)	H_s^0	50
Subsequent Search Height (mm)	H_s	45 0.83 ± 0.46

5. Validation method

The performance of the algorithm was validated based on the correct classification of vertebrae, the classification of individual slices and the distance of the vertebral uppermost and lowermost slices from the ground truth. The ground-truth annotations were manually generated from the axial images by a medical resident, and were reviewed/validated by a senior radiologist with over 20 years of experience in musculoskeletal radiology. Each slice was classified as either vertebra or disc based on the percentage of the vertebral column cross sectional area containing vertebra/disc in that slice. If more than 50% of the vertebral column consisted of a single vertebra, then that slice was labeled as belonging to that vertebra; otherwise, it was labeled as disc. This resulted in an uppermost and lowermost slice for each vertebra (e.g., L3 could be manually labeled to extend from axial slice 24 to slice 30).

To validate the correct classification of vertebrae, a vertebra was considered to be *correctly labeled* if: (1) there was at least one correctly labeled vertebral slice for that vertebra and (2) no slices were incorrectly labeled as another vertebra. If only condition (2) was met, the vertebra was considered to be *unlabeled*, since it was not given any label. The vertebra was considered to be *incorrectly labeled* if any of the vertebra's slices were incorrectly labeled as a different vertebra. Ideally, all vertebrae will be correctly labeled. An *incorrectly labeled* vertebra is a concern, since this can lead to incorrect diagnosis or treatment, whereas an *unlabeled* vertebra is merely inconvenient. To validate the correct classification of individual slices, slices were considered to be correctly labeled if they matched the ground truth and incorrectly labeled otherwise. To validate the vertebral uppermost and lowermost slice boundaries, a comparison was made with the manually identified ground truths. For each vertebra, the distance, in number of slices between the labeled uppermost slice of the vertebra and the ground truth uppermost slice, was calculated. The same principle was applied for the lowermost slices. Comparisons of these distances were then made for each vertebra over the set of subjects by calculating both the mean distances and the maximum distances in number of slices.

6. Results

6.1. Classification Accuracy

A total of 102 vertebrae were classified. A representative sample labeling for these images can be seen in Fig. 3. Of the 102 vertebrae, 101 were correctly identified and only 1 was incorrectly identified for a 99% structure-classification accuracy. The per slice classification accuracy was found to be 88%. These results are summarized in Table 4. The error in identifying the uppermost and lowermost vertebrae slice boundaries was found to be 0.83 ± 0.46 slices, with the average maximum distance from the classified uppermost and lowermost slice boundaries to the ground truth boundaries (over the 32 patients) being 1.44 ± 0.91 slices. It should be noted that, for the one vertebra that was defined as wrong, only one slice was incorrectly classified, with the rest of the vertebra being correctly classified. This error could be easily identified by a clinician. Additionally, for the majority of vertebrae, the boundaries are within one slice of the manually identified boundaries. These results confirm the clinical usefulness of our algorithm.

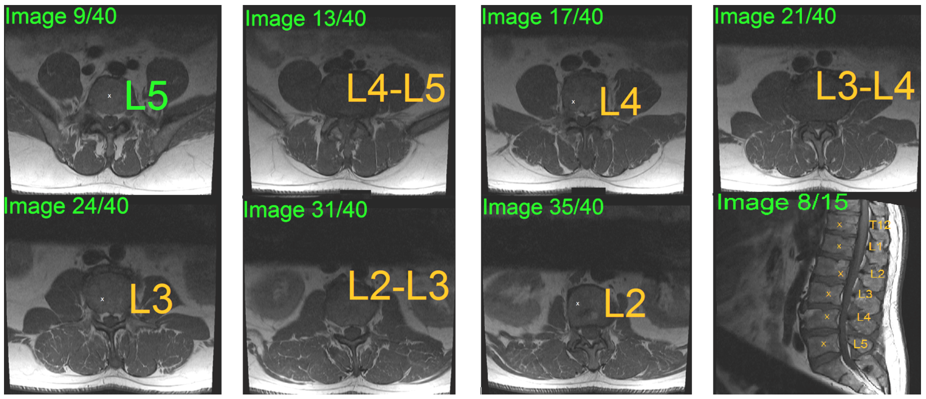


Figure 3: Representative output of the lumbar spine detection algorithm displaying axial slices from each analyzed level with the initial user input chosen at L5, with a labeled sagittal view provided for illustrative purposes.

No. of vertebral structures	Structure accuracy	No. of slices	Slice accuracy
102	99%	749	88%

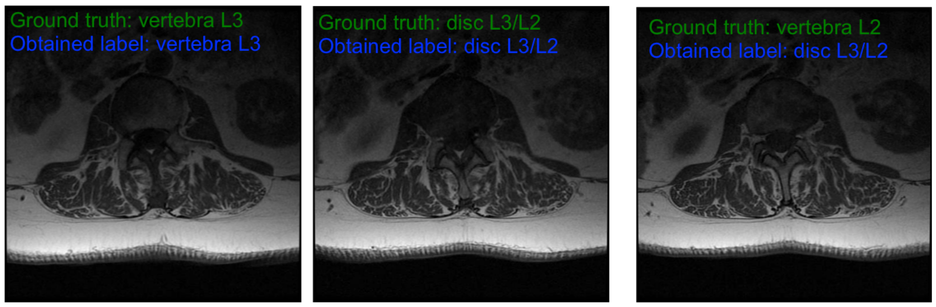
Table 4: Accuracy over 32 subjects.

Boundary Dist. (Slices)	Max Boundary Dist. (Slices)
0.83 ± 0.46	1.44 ± 0.91

Table 5: Boundary distance accuracy over 32 subjects.

6.2. A pathological case with a restricted disc space

Our data set included pathological cases where the inter-vertebral disc space is reduced. Fig. 4 depicts an example, in which the $L3/L2$ inter-vertebral structure corresponds to a *single* slice (Slice 25), with the top of vertebra $L3$ corresponding to the adjacent slice below (Slice 24) and the beginning of vertebra $L2$ corresponding to the adjacent slice above (Slice 26). For this example, the $L3/L2$ inter-vertebral disc slice was correctly annotated, and the algorithm yielded correct classifications for all the 3D vertebral structures surrounding this restricted inter-vertebral disc space ($L3$ and $L2$), i.e., for each structure we had the two conditions of correct classification verified: (1) there was at least one correctly labeled vertebral slice and (2) no slices were incorrectly labeled as another vertebra. Notice, however, that slice 26 was not correctly annotated as the start of the $L2$ vertebra.


Ground truth: vertebra L3
Obtained label: vertebra L3
Ground truth: disc L3/L2
Obtained label: disc L3/L2
Ground truth: vertebra L2
Obtained label: disc L3/L2

(a) Slice 24 (Top of L3) (b) Slice 25 (c) Slice 26 (Beginning of L2)

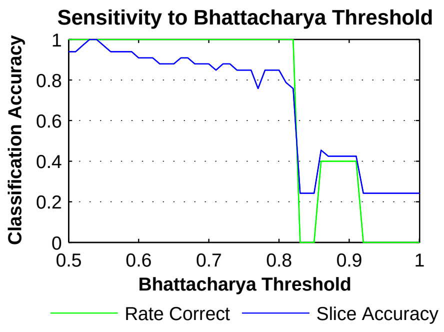
Figure 4: A pathological case, where the inter-vertebral disc space is restricted and corresponds to a single slice.

6.3. Sensitivity to parameters

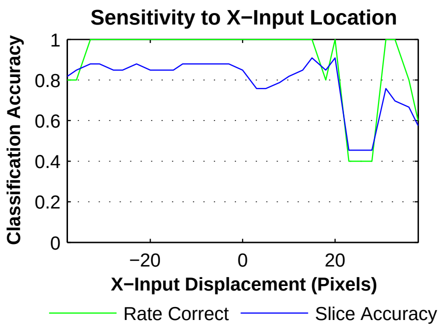
We performed an extensive sensitivity analysis in order to evaluate the robustness of the algorithm w.r.t the parameters. We varied each parameter in a range centered around the corresponding value in Table 3, and computed the classification accuracies for each range of values. The parameters that were analyzed for sensitivity included: the Bhattacharya threshold ρ, the x and y locations of the input point, the search window size, the number of bins and the area threshold. Fig. 5 depicts the results of varying the parameters. Varying the Bhattacharya threshold (Fig. 5a) produced very similar results for values in the range of $[0.5, 0.8]$, with a large drop in performance for values above 0.8. This is expected since, as the threshold gets higher, the classified pixels must match the target distribution more closely. This makes the algorithm less robust to variations away from the target distribution, excluding many pixels that are actually part of the vertebrae. For the x and y inputs, Fig. 5b and Fig. 5c show constant performance until about 20 pixels from the origin, giving a wide range of areas for selecting the initial point. The method was also very robust to window sizes (Fig. 5d): any window size greater than $9mm \times 9mm$ produced excellent results. The larger the input area for classification, the better the performance. Surprisingly, the method was not sensitive to the number of bins (Fig. 5e), neither to the area threshold (Fig. 5f). This demonstrated that these were not important parameters in the algorithm, and a possible speed up could be realized by reducing the number of bins.

6.4. Run times

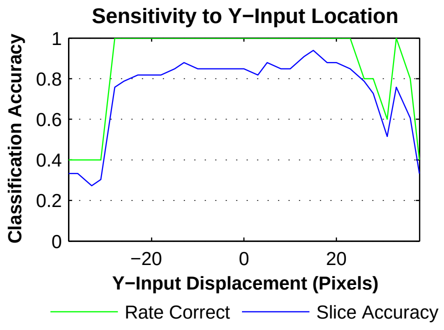
Table 6 reports the run times along with the computational complexity of our algorithm using a conventional calculation, the integral kernels and a GPU (Graphics Processing Unit) implementation of the integral kernels. The CPU code was written in Matlab (the Mathworks Nattick MA, USA), and the GPU code was written in CUDA. The experiments were run on a computer with a Xeon quad core processor (Intel, Santa Clara, CA, USA), 2Gb of RAM and an NVidia GeForce GTX 680 graphics card (Nvidia, Santa Clara, CA, USA). The



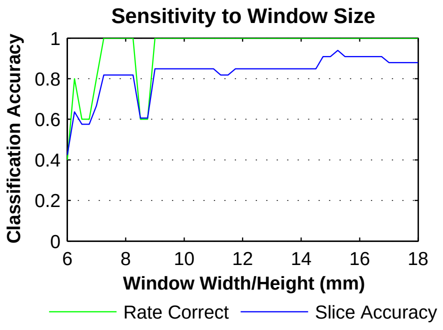
(a)



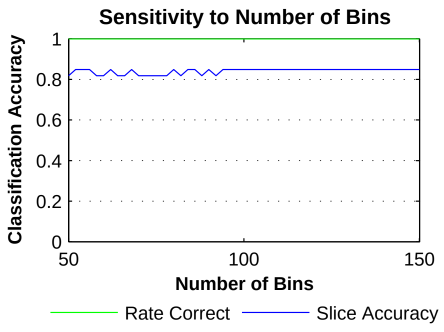
(b)



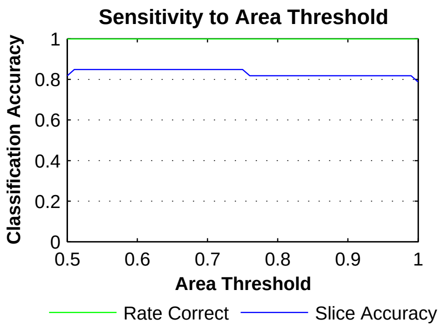
(c)



(d)



(e)



(f)

Figure 5: Analysis of the sensitivity of the algorithm to changes in various parameters: (a) Bhattacharyya threshold ρ, (b) X-input location, (c) Y-input location, (d) Window size, (e) Number of histogram bins and (f) Area threshold.

Table 6: Run time for 42 axial slices. The local window ($w \times h$) is 50×50 pixels. n is the number of pixels in the image, and z is the number of kernel features. The computational order of the integral image method is independent of the window size.

	CPU-conventional	CPU-Integral Images	GPU-Integral Images
Run time (s)	N/A	49.2	2.95
Order	$w \times h \times n \times z$	$n \times z$	$n \times z$

GPU implementation required 2.95 seconds for the whole volume, whereas the CPU one required 49.2 seconds. This running time was based on a T1 lumbar spine 3D image with 42 axial slices.

7. Conclusion

We investigated an efficient integral-kernel algorithm for classifying (labeling) the vertebra and disc structures in axial MR images. Based on an extension of integral images to kernel densities, our method built several feature levels, where pixel classifications via non-linear probability product kernels were followed by classifications of 2D slices, 3D single structures and 3D multiple structures. Furthermore, we embedded geometric priors based on known anatomical measurements of the spine. The ensuing algorithm runs in near real-time, when implemented on the GPU. Our experimental evaluations over 32 MR T1-weighted axial lumbar spine data sets (obtained from 32 patients) showed a structure classification accuracy of 99% and a slice classification accuracy of 88%. Our purpose was to design a system that works for the majority of patient data sets that are collected from routine MRI images of the spine: Our experiments included spine patients with typical disorders such as degenerative disc disease (DDD) and herniation. In the future, it will be interesting to extend our algorithm to:

- Images of instrumented (fusion) spines.
- Other parts of the spine, e.g., the cervical spine. In this case, we expect the problem to be more challenging due to larger displacements of the structures form one slice to another.

- Other MR sequences, such as proton density, as well as computed tomography (CT) images. We expect excellent results, as our learning part is based solely on a data-driven distributions, which is not limited to any specific modality.

It is worth noting that, as our method can be implemented in near real-time, it can be extended to accommodate user interventions and partial manual corrections when the method fails in labeling some parts of the spine, particularly in cases of atypical images such as those of instrumented (fusion) spines.

8. Acknowledgments

This work was done when Brandon Miles was a PhD student at Western, where he was supported by the Graduate Program in BioMedical Engineering and the Computer Assisted Medical Intervention (CAMI) training program, which is funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] D. F. Fardon, P. C. Milette, Nomenclature and classification of lumbar disc pathology: Recommendations of the combined task forces of the north american spine society, american society of spine radiology, and american society of neuroradiology, Spine 26 (5) (2001) E93–E113.

[2] R. S. Alomari, J. J. Corso, V. Chaudhary, Labeling of lumbar discs using both pixel- and object-level features with a two-level probabilistic model, IEEE Transactions on Medical Imaging 30 (1) (2011) 1–10.

[3] B. Glocker, J. Feulner, A. Criminisi, D. R. Haynor, E. Konukoglu, Automatic localization and identification of vertebrae in arbitrary field-of-view ct scans, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Vol. LNCS 7512, 2012, pp. 590–598.

[4] S. Huang, Y. Chu, S. Lai, C. Novak, Learning-based vertebra detection and iterative normalized-cut segmentation for spinal mri, IEEE Transactions on Medical Imaging 28 (10) (2009) 1595–1605.

[5] T. Klinder, J. Ostermann, M. Ehm, A. Franz, R. Kneser, C. Lorenz, Automated model-based vertebra detection, identification, and segmentation in ct images, Medical Image Analysis 13 (3) (2009) 471–482.

[6] J. Ma, L. Lu, Y. Zhan, X. S. Zhou, M. Salganicoff, A. Krishnan, Hierarchical segmentation and identification of thoracic vertebra using learning-based edge detection and coarse-to-fine deformable model, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Vol. LNCS 6361, 2010, pp. 19–27.

[7] A. B. Oktay, Y. S. Akgul, Localization of the lumbar discs using machine learning and exact probabilistic inference, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Vol. LNCS 6893, 2011, pp. 158–165.

[8] M. G. Roberts, T. F. Cootes, J. E. Adams, Automatic location of vertebrae on dxa images using random forest regression, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Vol. LNCS 7512, 2012, pp. 361–368.

[9] Y. Zhan, M. Dewan, M. Harder, X. S. Zhou, Robust mr spine detection using hierarchical learning and local articulated model, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Vol. LNCS 7510, 2012, pp. 141–148.

[10] S. Michopoulou, L. Costaridou, E. Panagiotopoulos, R. Speller, G. Panayiotakis, A. Todd-Pokropek, Atlas-based segmentation of degenerated lumbar intervertebral discs from mr images of the spine, IEEE Transactions on Biomedical Engineering 56 (9) (2009) 2225–2231.

[11] Z. Wang, X. Zhen, K. Tay, S. Osman, W. Romano, S. Li, Regression segmentation for m^3 spinal images, IEEE Transactions on Medical Imaging 34 (8) (2015) 1640–1648.

[12] B. D. Leener, J. Cohen-Adad, S. Kadoury, Automatic segmentation of the spinal cord and spinal canal coupled with vertebral labeling, IEEE Transactions on Medical Imaging 34 (8) (2015) 1705–1718.

[13] B. Miles, I. Ben Ayed, M. W. K. Law, G. J. Garvin, A. Fenster, S. Li, Spine image fusion via graph cuts, IEEE Transactions on Biomedical Engineering 60 (7) (2013) 1841–1850.

[14] I. Gilad, M. Nissan, A study of vertebra and disc geometric relations of the human cervical and lumbar spine, Spine 11 (2) (1986) 154–157.

[15] J. L. Berry, J. M. Moran, W. S. Berg, A. D. Steffee, A morphometric study of human lumbar and selected thoracic vertebrae, Spine 12 (4) (1987) 362–367.

[16] H.-W. Chang, H.-T. Chen, A square-root sampling approach to fast histogram-based search, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3043–3049.

[17] Y. Wei, L. Tao, Efficient histogram-based sliding window, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3003–3010.

[18] T. Jebara, R. I. Kondor, A. Howard, Probability product kernels, Journal of Machine Learning Research 5 (2004) 819–844.

[19] I. Ben Ayed, K. Punithakumar, S. Li, Distribution matching with the bhattacharyya similarity: A bound optimization framework, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (9) (2015) 1777–1791.

[20] P. A. Viola, M. J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.