Sparse Parallel Electronic Bowel Cleansing in CT Colonography

Richard Boyes, Xujiong Ye, Gareth Beddoe and Greg Slabaugh
Medicsight PLC
Kensington Centre, 66 Hammersmith Road, London W14 8UD
{richard.boyes,xujiong.ye,gareth.beddoe,greg.slabaugh}@medicsight.com

Abstract

We present a technique for storing the sparse data that often occurs when processing three dimensional medical images. The technique uses raster scan order to store the one dimensional volume indexes of each pixel location, and stores an inverted copy of these indexes for fast lookup. The inverted index is stored as a Judy array which is shown to be highly efficient in lookup times while using very little memory compared to hash tables. We demonstrate the efficiency of the data structure by performing partial volume segmentation and digital removal of oral contrast agent within CT Colonography (CTC). The method is demonstrated to be efficient in terms of speed and memory usage, and can parallelise efficiently.

1. Introduction

Modern medical imaging has dramatically increased the volume of data acquired in recent years. In the United States, there were about three million computed tomography (CT) scans performed in 1980, compared to an estimated 62 million in 2006 [3]. As requirements for more detailed imaging increase due to the superior diagnoses possible [6], the volume of data will increase exponentially, dramatically increasing the amount of storage and memory needed to process these scans.

Medical image processing is also becoming more complex: new, more sophisticated segmentation, registration and filtering techniques can require large amounts of memory and processing time. Many of these algorithms can take many minutes to execute which can be frustrating for the algorithm developer, end user and ultimately the patient. Medical image processing can also be constrained in other ways: for the radiologist to be able to review the scan interactively, the entire volume is loaded into memory, while copies may be needed for volume rendering or other processing.

CT colonography (CTC) is a promising screening technique for the detection of colonic polyps that can be a precursor to bowel cancer. CTC is an alternative to the current gold standard of optical colonoscopy, and is regarded as having higher patient acceptance [7]. The procedure involves the patient ingesting laxatives and contrast agent, which clear the bowel of any faecal matter and identify any residual material, respectively. The large intestine is then inflated using a CO_2 insufflator, and the subject scanned in both the prone and supine position to obtain three dimensional images. The colonic surface can be reconstructed to obtain a virtual flythrough of the colon by the reviewing radiologist, so colonic polyps which protrude from the colon wall can be identified. While the laxative procedure should clear the bowel of faecal material, usually some fluid remains, identifiable by the contrast agent. Digital removal of this contrast agent (commonly known as electronic bowel cleansing (EBC), an example is shown in Figure 1) is a useful technique for the radiologist as the fluid can occlude polyps (commonly this is why both a prone and supine scan is taken).

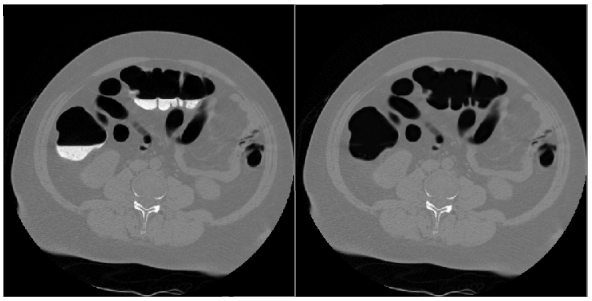

Figure 1. Axial view of a supine CT colonography scan. On the left is the scan with radio opaque tagging of leftover fluid in the colon, and on the right it has been digitally removed.

Further, computer aided detection (CAD) procedures are gaining traction in CTC in order to assist the radiologist and improve the accuracy of diagnosis. EBC can help in increasing the sensitivity of CAD algorithms as the algorithm would no longer need to account for different cases of air/fluid/tissue delineated boundaries.

Currently CTC examinations have large data require-

ments. An examination requires a prone and supine scan of the patient, with each scan typically requiring $450\ 512\times512$ image slices, a resolution of $(0.7 \times 0.7 \times 1.0)$ mm, and two bytes per pixel. These examinations are also increasing in size (due to the desire to make voxel sizes more isotropic). EBC can be implemented using simple thresholding techniques but these perform poorly due to partial volume effects at each voxel. Current state of the art techniques [11] [12] [4] usually approximate this partial voluming in some manner, which results in a smooth, natural boundary that looks 'correct' to the reviewer. However these techniques are computationally intensive. The method presented in [11] used partial volume segmentation to model the fluid that needed to be digitally subtracted from the scan, requiring buffer storage for different CT voxel classes of air, contrast enhanced fluid and tissue and its constitutive proportions. This approach can require storage of multiples of the original CT volume, which in a clinical environment may strain computational resources and negatively impact on workflow. However, in CTC (and hence EBC) only the large intestine needs to be examined and processed. Although CTC examinations can generate approximately 100 million data points for each scan, we have found that the colon and its contents only constitute approximately 5-10% of the volume of the scan.

In this paper, we introduce a sparse indexing method that allows us to store only those voxels relevant to the CTC examination, reducing the memory and computational requirements to process the scan. We demonstrate its efficacy by applying it to the problem of electronic bowel cleansing of the colon, using it with a similar partial volume segmentation technique [8] to obtain accurate estimates of the volume of tagged fluid at each voxel, and we then replace these partial volumes with an approximation to the intensity of air. The appearance of the electronically cleansed scan is shown to be visually accurate, while the method shows a high degree of performance, flexibility, scalability and low memory usage.

2. Methods

2.1. Sparse Volume Indexing

Sparse data processing is a common method used in matrix processing, whereby the matrix is populated primarily with zeros which are ignored. Large sparse matrices are found in science when solving partial differential equations, as the solution to unknown variables is usually dependent on a small number of neighbouring items. Similarly in medical image computing, regions outside of the structure of interest can be ignored. In CTC, this would be the distended colon and surrounding tissue. We wished to develop a data structure and method that can index a sparse three dimensional volume. A huge array of techniques exist for the spatial indexing of images [10]. We wished to use a data structure concordant with the following guidelines:

- Random access of any voxel, including neighbours, in constant, i.e., O(1) time.
- Extraction of the live voxels in any row or slice contiguously, i.e., given a slice number we can efficiently compute the memory offsets of the relevant voxels in constant time, with no lookup misses. This allows data to be processed in a slice by slice fashion, and is consistent with how medical images are stored on disk.
- Suitability for multi-threading.
- Low memory usage.

In order to store the sparse indexes, we require a mapping from the three dimensional spatial indexes to a one dimensional index, which can be done by using a space filling curve (see [10]).

We chose to implement the algorithm using raster scan order, based on its simplicity and ability to process voxels in a slice oriented fashion. Each spatial (in our case, slice, row and column) component can be indexed by a single 4 byte integer. As the image dimensions in the through plane are powers of two (as they are for most medical images), the raster scan order was created using bit contentation. Although this limits the technique to images that are powers of 2, we can use simple multiplication and modulus arithmetic (which are less efficient) to perform the same operations, or simply pad the images to the nearest power of 2, which needn't increase storage as only the offsets change.

Given the slice `(z)`, row `(y)` and column `(x)` indexes, the volume index `(vidx)` is calculated by:

```
vidx = (z << zshift) | (y << yshift) | x;
```

Where `zshift` and `yshift` represent the powers of two that offset the coordinates by the image dimensions, e.g., for a set of 512×512 images, the `zshift` would be 18 ($2^{18} = 512 \times 512$) and yshift would be 9 ($2^9 = 512$). The `z`, `y`, and `x` indexes of each voxel can be obtained by applying appropriate bitmasks and shifting the resultant bits, i.e.,

```
z = (vidx & zmask) >> zshift;
y = (vidx & ymask) >> yshift;
x = (vidx & xmask);
```

Masks and bit shifts can be computed when the scan is loaded thus:

```
zshift = 0;
yshift = xmask = 1;
while((1 << yshift) < imagewidth)
{
```

```
  xmask = xmask | (xmask << 1);
  yshift++;
}
zshift = yshift + 1;
ymask = (1 << yshift);
while((1 << zshift) < (imagewidth*imageheight)
{
  ymask = ymask | (ymask << 1);
  zshift++;
}
zmask = (0xffffffff << zshift);
```

We can store all the `vidx` components within the structure of interest in an array; any voxel properties (such as scan intensity) located at the same index can be stored in an array of the same dimension. Given this `VolumeIdx` array we also need to have an inverted copy, that is, given any one dimensional volume index (or voxel coordinate), a method of looking up the properties of that voxel is required.

This `InvertedVolumeIdx` can used for random access, looking up voxel neighbours, and efficient computation of the memory offsets of a given slice. Effectively the `InvertedVolumeIdx` represents an `int-int` mapping using an associative array, which can be implemented using a wide variety of techniques such as a hash table or binary tree. Figure 2 demonstrates the usage of the `InvertedVolumeIdx` and `VolumeIdx` indexes to obtain the neighbouring intensities of a voxel.

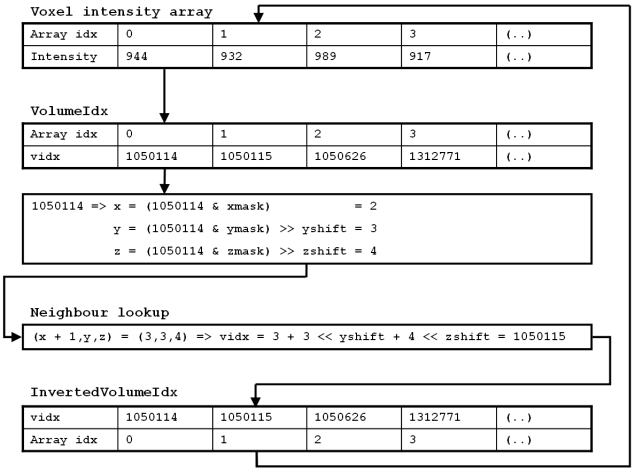

Figure 2. Flow diagram for a sparse array of voxel intensites, and how the the two sparse indexing data structures were used to lookup neighbouring intensities, in this case the $x + 1$ neighbour of the index $(x, y, z) = (2, 3, 4)$.

Initially we used a hash table to implement the `InvertedVolumeIdx`, due to its O(1) performance in lookup and insertion. However the memory footprint was excessive due to the need for the table to store both the key and value pairs, and also for the size of the table to be greater than the number of the stored entries.

We thus investigated a data structure called a Judy array [2]. Judy arrays are complex associative array data structures that use many optimisations to improve memory performance, particularly with regard to cache.

Judy arrays are implemented as a 256-ary trie (for retrieval), or one byte per level, allowing the expanse of 2^{32} (4 byte integers) to have only 4 levels. The choice of 256-ary by the designers of the Judy array was done to minimise cache-line fills (fetching of data from memory to cache), which penalise performance. It is claimed that in the worst case scenario there would only be four cache line fills, equal to the number of levels in the tree. Storing a 256-ary trie with four levels naively would use an excessive amount of space. Judy uses complex internal data structures to compress and manage redundant data in the tree.

Judy arrays are not particularly well covered within the scientific literature, excepting [5], which compared their implementation within an in memory database to other comparable data structures. A comparison of Judy arrays and hash tables is available at [1], which found Judy arrays had slightly superior performance for larger volumes ($> 1 \times 10^6$) of sequential data (which applies here due to the raster scan order) for both hit (item is in table) and miss (item is not in table) lookups, and used far less memory. We were less concerned with insertion and deletion times; once the sparse index is built we would predominantly be doing table lookups. The Judy array can also traverse its data contiguously in order (i.e., no lookup misses); this is very useful when accessing ranges of the data, such as processing the data in a slice or row oriented fashion where the volume index is strictly increasing, which is the case when using raster scan order. Other useful features include the ability to directly lookup the nth index efficiently, and count the number of items within a given range of keys. We performed some simple experiments using our own data to verify the Judy array's speed and low memory attributes.

Thus, given an image and mask identifying the voxels in the structure of interest, we construct the `VolumeIdx` and `InvertedVolumeIdx` by looping through the image, and any live voxels in the mask have their one dimensional volume index pushed onto the `VolumeIdx`. An array of voxel intensities (and any other properties) is updated in the same way. Once this process is complete, the `InvertedVolumeIdx` is computed by looping over the `VolumeIdx` and putting the associated entries into the Judy array. Any associated voxel properties can be stored in an array of the same size and layout as the `VolumeIdx`, while looking up voxel properties given a three dimensional index can be done by converting the three dimensional indexes to the one dimensional volume index, and then looking up the entry using the `InvertedVolumeIdx`. Neighbours of a voxel can be looked up in a similar manner, by decomposing any one dimensional volume index into

its constitutive three dimensional indexes and incrementing them in each respective direction, and then recomposing them into a one dimensional volume index, which we can use to lookup neighbouring voxel properties using the `InvertedVolumeIdx`. In this way looking up neighbours is an O(1) process. This process is demonstrated in Figure 2.

2.2. Partial Volume Segmentation in CT Colonography

We wished to apply the sparse indexing given above in a practical context where large amounts of storage are required for auxiliary variables. Performing a partial volume segmentation of the three materials that make up the contents of the insufflated colon (namely air, contrast agent and the colonic tissue) provides such a task, as floating point storage for up to three different materials at each voxel is required. We wished to perform a partial volume segmentation and subsequent reconstruction, replacing contrast agent with air, to electronically clean the large bowel in CT colonography, which will aid in the peformance of examiners and computer aided detection methods.

The partial volume segmentation technique used has been described previously [8]. We use an iterative expectation maximisation approach to the maximum a posteriori (MAP) solution of segmenting material mixtures. Supposing each voxel I_i could contain three types of material (human tissue, air and radio opaque tagging). Let material k contribute x_{ik} to the observation at voxel I_i, i.e., $I_i = \sum_k x_{ik}$, and let ρ_{ik} denote the fraction of material k within voxel i, with $\sum_k \rho_{ik} = 1$ and $0 \leq \rho_{ik} \leq 1$. The distribution of each material can be described by its mean and variance μ_k and σ_k^2, while the probability of sampling x_{ik} given these parameters can be given by:

$$Pr\left(X \mid \rho, \mu, \sigma^2\right) = \prod_{i,k} \frac{1}{\sqrt{2\pi\rho_{ik}\sigma_k^2}} \times$$
$$exp\left(-\frac{\left(x_{ik} - \rho_{ik}\mu_k\right)^2}{2\rho_{ik}\sigma_k^2}\right)\frac{1}{Z}exp\left(-\frac{1}{2}\beta U\left(\rho_{ik}\right)\right) \quad (1)$$

where Z is a normalisation constant and U is an *a priori* penalty used as a spatial constraint within the Markov random field (MRF) framework to ensure like materials are contiguous, and β is a parameter controlling the degree of the spatial penalty on the material mixtures. $U\left(\rho_{ik}\right)$ is defined as

$$U\left(\rho_{ik}\right) = \sum_{j \in nbor(i)} w_{ij}\left(\rho_{ik} - \rho_{jk}\right)^2 \quad (2)$$

where w_{ij} represents the weights between each pair of neighbours i, j. A weighting function for a group of neighbours is needed as the voxels are non-isotropic, and we require neighbouring voxels to have different contributions proportional to the inverted distance between neighbours. We calculate w_{ij} to be

$$w_{ij} = \frac{min\left(d\left(i,n\right)\right), n \in nbor\left(i\right)}{d\left(i,j\right)} \quad (3)$$

where $d\left(i,n\right)$ represents distance between two pairs of voxels i, n. In our implementation, we take into account a six neighbour stencil.

Using a Gaussian distribution for x_i and I_i the conditional expectation or E step of the log *a posteriori* distribution in the EM algorithm given the observed data I_i and an estimate of $\rho_{ik}, \mu_k, \sigma_k^2$ is:

$$E\left(ln\left(\Pr\left(X \mid \rho, \mu, \sigma^2\right) \mid Y, \rho, \mu, \sigma^2\right)\right) =$$
$$-\frac{1}{2}\sum_{i,k}\ln\left(2\pi\right) + \ln\left(\rho_{ik}\sigma_k^2\right) +$$
$$\frac{1}{\rho_{ik}\sigma_k^2}\left(x_{ik}^2 - 2\rho_{ik}\mu_k x_{ik} + \rho_{ik}^2\mu_k^2\right) + \beta U\left(\rho_{ik}\right) \quad (4)$$

The conditional means for x_{ik} and x_{ik}^2 at each iteration are given by

$$x_{ik} = \mathrm{E}\left(x_{ik} \mid I_i, \rho, \mu, \sigma^2\right) = \mathrm{E}\left(x_{ik}\right) +$$
$$\mathrm{Cov}\left(x_{ik}, I_i\right)\mathrm{Cov}\left(I_i\right)^{-1}\left(I_i - \mathrm{E}\left(I_i\right)\right)$$
$$= \rho_{ik}\mu_k + \frac{\rho_{ik}\sigma_k^2}{\sum_l \rho_{il}\sigma_l^2}\left(I_i - \sum_l \rho_{il}\mu_l\right) \quad (5)$$

and

$$x_{ik}^2 = \mathrm{E}\left(x_{ik}^2 \mid I_i, \rho, \mu, \sigma^2\right) = \mathrm{E}^2\left(x_{ik} \mid I_i, \rho, \mu, \sigma^2\right) +$$
$$\mathrm{Var}\left(x_{ik} \mid I_i, \rho, \mu, \sigma^2\right) = \left(x_{ik}\right)^2 + \rho_{ik}\sigma_k^2\frac{\sum_{l \neq k}\rho_{il}\sigma_l^2}{\sum_l \rho_{il}\sigma_l^2} \quad (6)$$

The maximization or M step is then used to calculate updated values of μ and σ^2 for each material class. Differentiating Eqn(4) with respect to μ_k and setting it to zero results in:

$$\mu_k = \frac{\sum_{i=1}^N x_{ik}}{\sum_{i=1}^N \rho_{ik}} \quad (7)$$

Likewise for σ_k^2:

$$\sigma_k^2 = \frac{1}{N}\sum_{i=1}^N \frac{x_{ik}^2 - 2\rho_{ik}\mu_k x_{ik} + \rho_{ik}^2\mu_k^2}{\rho_{ik}} \quad (8)$$

Differentiating Eqn(4) with respect to each ρ_{ik} results in a linear set of equations which needs to be solved for each voxel. We can also use the fact that $\sum_k \rho_{ik} = 1$ which reduces the number of equations by one. In the case of two material mixtures, we can update the partial volumes as:

$$\rho_{i1} = \frac{x_{i1}\sigma_{i2}^2\mu_1 + \mu_2^2\sigma_{i1}^2 - x_{i2}\sigma_{i1}^2\mu_2 + 2\beta\sigma_{i1}^2\sigma_{i2}^2\sum_j w_{ij}\rho_{i1}}{\sigma_{i2}^2\mu_1^2 + \sigma_{i1}^2\mu_2^2 + 2\beta\sigma_{i1}^2\sigma_{i2}^2\sum_j w_{ij}} \quad (9)$$

and $\rho_{i2} = 1 - \rho_{i1}$, while for three materials,

$$\begin{aligned}
\rho_{i1}\left(\frac{\mu_1^2}{\sigma_1^2} + 2\beta\sum_j w_{ij} + \frac{\mu_3^2}{\sigma_3^2}\right) + \rho_{i2}\left(\beta\sum_j w_{ij} + \frac{\mu_3^2}{\sigma_3^2}\right) &= \\
\frac{x_{i1}\mu_1}{\sigma_1^2} + 2\beta\sum_j w_{ij}\rho_{j1} - \frac{x_{i3}\mu_3}{\sigma_3^2} + \frac{\mu_3^2}{\sigma_3^2} + \beta\sum_j w_{ij}\rho_{j2} & \\
\rho_{i1}\left(\beta\sum_j w_{ij} + \frac{\mu_3^2}{\sigma_3^2}\right) + \rho_{i2}\left(\frac{\mu_2^2}{\sigma_2^2} + 2\beta\sum_j w_{ij} + \frac{\mu_3^2}{\sigma_3^2}\right) &= \\
\frac{x_{i2}\mu_2}{\sigma_2^2} + 2\beta\sum_j w_{ij}\rho_{j2} - \frac{x_{i3}\mu_3}{\sigma_3^2} + \frac{\mu_3^2}{\sigma_3^2} + \beta\sum_j w_{ij}\rho_{j1} & \quad (10)
\end{aligned}$$

After solving for ρ_{i1} and ρ_{i2}, $\rho_{i3} = 1 - \rho_{i1} - \rho_{i2}$. Note that $\sigma_{ik}^2 \approx \rho_{ik}\sigma_k^2$, where ρ_{ik} was stored from the previous iteration.

Given the description of the equations above, it is clear that the amount of storage data required for an entire partial volume description of the CT scan would be excessive. We used the sparse volume structure outlined in Section 2.1 to reduce the volume used by only considering voxels within the vicinity of the colon, as indicated in Figure 3. We further reduced the amount of data used by considering that most voxels will contain only one material type. We stored an index of the number of voxel classes at each voxel i as an array z_i and then stored an offset array as $o_{ik} = \sum z_i + k$ which was used to index floating point arrays of the fractional voxel properties ρ_{ik} and x_{ik}.

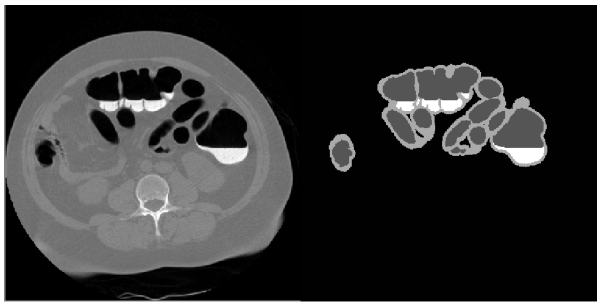

Figure 3. An example of a CT colonography scan and the initial labelling used before executing the partial volume segmentation and subsequent cleansing. The `VolumeIdx` and `InvertedVolumeIdx` data structures were populated using the labelled image on a slice by slice basis.

Given an initial colon segmentation of the original scan, which we obtained using simple thresholding and morphological operations, we identify the air and tagging labels using the K-means clustering algorithm. The segmentation is then dilated several times to give a region of tissue with a different label. This dilated, labelled mask (see Figure 3) is then used to initialise the sparse data structure and the partial volume arrays, assuming that any mixture of voxels within a 26-nbor kernel will imply that the central voxel will be made up of fractional parts of the labels within that neighbourhood. The fractional parts ρ_{ik} were initialised as $1/K$ for each class where K is the number of different labels within the neighbourhood. The global mean μ_k and variance σ_k^2 of each class were initialised using the mean and variance of voxels within each respective class label.

Given this initialisation, the algorithm for the partial volume segmentation is:

1: **for** $n = 0$ to MaxIterations **do**
2: Estimate class intensities at each voxel $x_{ik}(n)$ (Equation 5)
3: Update class means $\mu_k(n+1)$ (Equation 7)
4: Update class variances $\sigma_k^2(n+1)$ (Equation 8)
5: Update class volumes $\rho_{ik}(n+1)$ (Equation 9 and 10)
6: Check for convergence by calculating $\sum_k \left|\frac{\mu_k(n+1)}{\mu_k(n)}\right| \leq 1 + \varepsilon$
7: **end for**

2.3. Digital Removal of Contrast Agent

Given the partial volume segmentation of each material class, we can replace the contrast agent with an estimate of air:

$$I_i^{new} = I_i + (\mu_{air} - \mu_{tag})\,\rho_{itag} - \rho_{itag}\epsilon_i \qquad (11)$$

Where ϵ_i represents the error in the estimation I_i, approximately $I_i - \sum_k \rho_{ik}\mu_k$. We then apply a geometry preserving filter [9] to the cleansed voxels to smooth any remaining noise. This non-linear filter is calculated as an iterative diffusion equation,

$$I^{t+1} = I^t + \Delta t\left(\nabla \cdot \left(\frac{\nabla I}{\parallel \nabla I \parallel}\right) \parallel \nabla I \parallel\right) \qquad (12)$$

which we apply for 5 iterations and a step size of $\Delta t = 0.1$.

2.4. Parallelism

Each of the steps in the segmentation and the smoothing process is highly parallelisable, as it involves a simple loop over all of the voxels, and in the case of the updates to μ_k and σ_k^2 a reduction variable can be used. We parallelise each step in the partial volume segmentation and geometric diffusion using the OpenMP compiler directives.

2.5. Experiments

We computed the partial volume segmentation for 11 CT Colonography scans from four different hospitals, each with their own patient preparation and scanning workflow. Each scan had a significant amount of residual contrast agent. We removed the contrast agent from each scan and assessed the

outcome using visual inspection, and computed the time taken to perform the entire cleansing operation (including image I/O and the initial indexing of the scan), the memory used by the sparse volume index (including the Judy array), and noted the number of live voxels. For the partial volume segmentation, we choose the maximal number of iterations to be 12, β to be 5 (which we found empirically), and the convergence tolerance ε to be 0.0001.

We also calculated the time taken to apply the geometric diffusion filter for different numbers of threads to demonstrate that the sparse indexing method can parallelise well when working on trivial parallel problems. Using the OpenMP framework, we apply $1-4$ threads on a quad-core computer and computed the time taken to perform the geometric diffusion of Eqn(12) for each scan and plotted the results. We chose to highlight the parallel performance of the geometric diffusion process as although it is trivially parallelisable, it requires extensive lookup of neighbour voxels to calculate first and second-order derivative terms, which required a considerable amount of work to be done by the Judy array.

Electronic Bowel Cleansing Results					
Scan id	#Scan voxels '000s	#Colon voxels '000s	Sparse Index size (MB)	Total memory (MB)	Run time(s)
1	110886	10992	100	296	72
2	120586	8208	77	222	77
3	117703	6157	58	169	57
4	113508	8914	82	245	70
5	122946	8045	75	218	65
6	113246	7501	71	205	63
7	103547	5380	52	148	49
8	111149	2529	25	65	40
9	103285	4795	46	130	46
10	125043	5142	50	140	53
11	113508	4040	39	108	44

Table 1. Memory and time taken for bowel cleansing. Note the approximate 10 bytes/voxel storage requirements for the sparse volume index. The run time includes the disk I/O for loading the image and calculating the sparse index, and the output of the corrected voxels to a copy of the original scan.

3. Results and Discussion

Assessing the results visually, we found a smooth, natural appearance where the tagged material had been removed, apparent in Figures 1 and 4. The results presented in Table 1 demonstrate the effectiveness of the sparse volume index in reducing the memory used and the processing time. We estimate that to perform the experiments above using the whole scan in memory would have required over 1GB of memory and taken several minutes to execute. Although difficult to compare directly due to the more modern hardware used here, it took a similar method proposed in [11] approximately 20 minutes to complete, more than an order of magnitude slower.

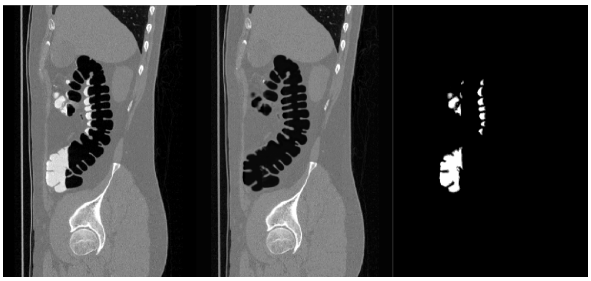

Figure 4. Pre and post images of a cleansed scan, with the associated partial tagging volume, reformatted in sagittal orientation.

It is evident that the sparse index uses approximately 10 bytes per voxel of memory from Table 1, which could be used as a rough rule of thumb for the memory requirements of the index. Noting that this is made up of the integer volume indexes (which constitute four bytes per voxel) and the inverted volume index, it implies that the Judy array required only six bytes per voxel storage, a significant saving when compared to other storage mechanisms (e.g., an open addressed hash table would require 8 bytes per voxel plus additional storage to ensure the load factor was reasonably small, say $\frac{1}{0.7} \approx 1.43$ times the actual entries - chained hash tables scale better with higher load factors but need extra storage for the linked list implementation at each bucket). Figure 5 clearly demonstrates the parallel scalability of the sparse index when applied to the simple diffusion process.

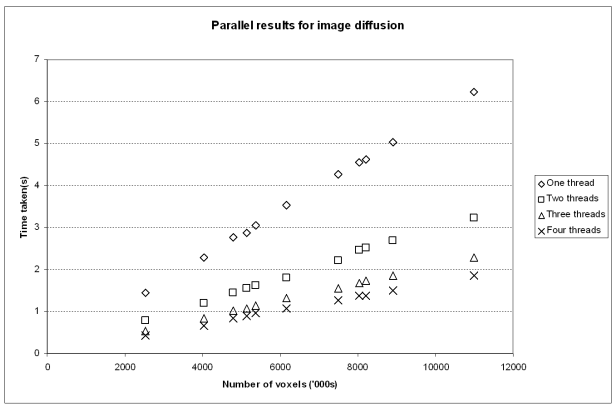

Figure 5. Parallel acceleration for each scan, demonstrating linear speedup per thread and across a range of voxel numbers, indicating a high level of scalability for the sparse indexing system. Similar scalability was observed for the parallel parts of the EBC pipeline.

4. Conclusions

We have presented a method for sparse volume indexing of a three dimensional medical image that compactly stores voxel properties and demonstrated its usefulness by implementing a state of the art electronic bowel cleansing technique and applying it to tagged images in CT colonography. The method is scalable and parallelises well, indicating it can be applied to a wide range of medical image processing tasks where the data processed is significantly less than the size of the scan. In future work, we hope to be able to implement the partial volume segmentation using a smaller representation of the colon as we only need to remove the tagged material, allowing us to discard a significant number of the air voxels from processing. We also wish to implement the algorithm to take into account a higher number of material classes, as separate 'pools' of tagging can have different intensity properties, due to the tagging agent not being diluted or dispersed uniformly. This would require a higher number of material classes but the maximum number of material mixtures still remaining as three.

The sparse representation presented in this paper is simple, efficient and flexible, and suitable for a wide range of medical imaging problems with large data requirements.

References

[1] S. Barrett. http://www.nothings.org/computer/judy. 3

[2] D. Baskins and A. Silverstein. http://judy.sourceforge.net. 3

[3] D. J. Brenner and E. J. Hall. Computed tomography - an increasing source of radiation exposure. *New England Journal of Medicine*, 357(22):2277–2284, 2007. 1

[4] W. Cai, M. Zalis, and H. Yoshida. Mosaic decomposition method for detection and removal of inhomogeneously tagged regions in electronic cleansing for ct colonography. In *Proceedings of SPIE*, volume 6915, February 2008. 2

[5] S. L. Fritchie. A study of erlang ets table implementations and performance. In *Proceedings of ACM SIGPLAN Workshop on Erlang*, pages 43–55, August 2003. 3

[6] S. Y. Jeong, M. J. Chung, C. S., S. Y. M, and L. K. S. 1024 matrix image reconstruction: Usefulness in high resolution chest ct. *J Korean Radiol Soc*, 55(6):565–569, 2006. 1

[7] C. D. Johnson and A. H. Dachman. Ct colonography: The next colon screening examination? *Radiology*, 216:311–319, 2000. 1

[8] Z. Liang and S. Wang. An em approach to map solution of segmenting tissue mixtures: A numerical analysis. *IEEE Transactions on Medical Imaging*, 28(2):297–310, 2009. 2, 4

[9] S. Manay and A. Yezzi. Anti-geometric diffusion for adaptive thresholding and fast segmentation. *IEEE Transactions on Image Processing*, 12(11):1310–1324, 2003. 5

[10] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, 2006. 2

[11] Z. Wang, Z. Liang, X. li, L. Li, B. Li, D. Eremina, and H. Lu. An improved electronic colon cleansing method for detection of colonic polyps by virtual colonoscopy. *IEEE Transactions on Biomedical Engineering*, 53(8):1635–1647, 2006. 2, 6

[12] M. E. Zalis, J. Perumpillichira, and P. F. Hahn. Digital subtraction bowel cleansing for ct colonography using morphological and linear filtration methods. *IEEE Transactions on Medical Imaging*, 23(11):1335–1343, 2004. 2