**1 Introduction**

Ultrasound (US) has evolved to be one of the major medical imaging systems, being responsible for one of five medical images used for medical diagnosis, and used in nearly all hospitals and clinics [Jensen]. The use of medical US has mayor advantages over other imaging modalities (magnetic resonance and computer tomography) such as: minimal invasion, low cost, ease to use, the ability to obtain images in real time from different perspectives, and no ionizing radiation [Halliwell]. Although these advantages, the use of conventional 2D US may have some drawbacks; the 2D US images represent thin planes of the patients, so choosing the visualization plane of the lesion is difficult since the user must mentally integrate many 2D images to form an impression of the 3D anatomy and pathology; finding the same location of the visualization plane is difficult since the US probe is controlled manually. This problems may be corrected by using 3D US [Fenster].

Three-dimensional ultrasonic imaging is becoming a widespread practice in clinical environments due to the potential applications based on 3D representation. It provides some interesting benefits: the spatial relationships among 2D planes are preserved, allowing an offline examination of scans previously recorded; renderization and visualization of planes that cannot be acquired because of geometrical constrains imposed by the US probe and the patient anatomy [Estepar]. High quality and instantaneous 3D imaging remains a long term goal of medical US research [Rohlling]. Several 3D US techniques have been reported in the past few years, but these can be coarsely classified as derived from 3D probes and those that obtain a 3D data from 2D B-scans acquired in rapid succession [Estepar]. Using 3D probes, where 3D volume is imaged directly from a single probe position can simplify the reconstruction of the 3D data set since the geometry of the acquired data is known, making real-time 3D reconstruction possible, but this approach does not allow for fine control of the location of the 2D planes, the reconstructed volume geometry and size is constrained by the probe, and the probes are bulkier and more expensive compared with conventional 2D US probes [Honggang]. Because of this, researchers have developed approaches to convert a conventional 2D US transducer into one that is capable of 3D imaging [Fenster 2]. In freehand acquisition, the operator holds the probe and manipulates it in the usual manner over the anatomy to be view and images in arbitrary positions and orientations are acquired. This technique offers special advantages because the operator can select optimal views and accommodate complex patient surfaces and is not limited to the transducer geometry and size [Fenster 3]. However the reconstruction step is still an acute problem with regards to computation time and reconstruction quality because of the sparsity of data [Pierrick].

Mainly there are two different approaches for volume reconstruction. The first kind of methods, reverse approach or Voxel Based Methods (VBM), take the position of the voxels within the volume and find the appropriate pixels within the US images. The second approach, are methods based on solving the forward approach, as tacking the position of each pixel within the B-mode images and finding the appropriate voxel positions within the volume, this methods are called Pixel Based Methods (PBM) [Schiepers]. The last are often more accurate by assigning the pixel values to the nearest voxel and allow several pixels to contribute to the values of each voxel, which is an improvement over the VBM algorithms, and with a fast enough implementation a PBM could be constructed as an iterative method, building the volume along with image acquisition and be made into a near real-time reconstruction. The most common PBM algorithm is the Pixel Nearest Neighborhood (PNN) and consists of two steps, bin-filling step (BFS) and hole-filling step (HFS); while the BFS depends on the number of images and in its simpler version is fast, the HFS depends on the number of empty voxels left by the BFS and usually is the slower step of the reconstruction [Solberg]. The development of fast or real-time methods for 3D ultrasound reconstructions with high resolution has received significant attention, since it has been demonstrated that this imaging technique has applicability in image guided surgery and interventions like neurosurgery, biopsy and radiation therapy [Gobbi]. Medical image processing in graphics processing units (GPU) has become quite popular recently because it makes it possible to apply more advanced algorithms and perform computational demanding tasks fast enough for a clinical context [Eklund]. Some reconstruction algorithms have been implemented in GPU to accelerate the reconstruction process: Moon et al. reported a PNN method implemented in GPU that can reconstruct volumes of 161x104x232 voxels in 26.97 seconds, and implemented a new Bayesian reconstruction method in GPU that is 46.39 times faster than the CPU version of their method and 2.86 times faster than the CPU version of the PNN algorithm [Moon GPU]; Dai et al. optimize the bin filling process in GPU for real time visualization of he reconstruction while acquiring the 2D images, using incremental reconstruction [Ohbuchi], after all images are acquired the hole filling step is done also in GPU, however no optimization is made in this step except for parallel computation [Dai].

Samet has provided a useful overview of the octree technique to image processing applications [Samet]. Hierarchical octrees (or quadtrees in two dimensions) offers an efficient method for the spatial discretization of sparse data by recursive decomposition [Saalehi]. K-D trees and octrees are widely adopted to accelerate neighbor search in three-dimensional data for large-scale datasets [Behley]. It is not of our knowledge that octrees have been used in order to optimize the computational time of ultrasound volume reconstruction; in this paper we propose the use of octrees in order to optimize the PNN method in both steps to obtain a fast full reconstruction with high resolution. This paper is organized as follows: Section 2 describes the background of pixel based reconstruction methods, focusing on the PNN method, and the background of octrees and naïve neighborhood search; Section 3 describes the PNN reconstruction method implemented using octrees and Morton keys; Section 4 describes the experiments and present the results of the octree based reconstruction; Section 5 shows the concluding remarks and opens a discussion for future work and possible upgrades.

**2 Background**

*Pixel Based Volume Reconstruction*

Usually the PBM consists of two steps: The distribution or bin-filling step, where the input pixels are traversed and the pixel value is applied to one or several voxels; and the Hole-Filling step, where the voxels are traversed and empty voxels (voxels that have not been filled in the BFS) are being filled [Solberg].

The most common PBM is the Pixel Nearest Neighbor. In the BFS, the intensity of each pixel in the 2D ultrasound images is assigned to the nearest voxel in the 3D ultrasound volume; the contribution of multiple pixels to a single voxel usually are averaged [Doud], but other variants are possible, like keeping the max value, the most recent value or the first value [Solberg] (FIGURA); in order to find the nearest voxel to the pixel the 2D US image coordinates must be converted to global coordinates [Moon], in the freehand approach a transformation matrix () from the B-scan coordinate system () to the reconstruction volume coordinate system () has to be predetermined by a spatial calibration [Dai]. If the images are acquire with sufficient separation (less than one voxel for the PNN) a HFS will not be needed, but in the freehand approach this is complex to achieve [Gobbi]. In the HFS the algorithm loops over the target volume and fill the gaps; for every empty voxel, non-empty neighboring voxels in a certain radius around the empty one are averaged into the resulting value, usually a growing radius starting at 1 voxel is used, if non-empty voxels are found the region will grow until non empty pixels are found or a maximum radius is reached [Miller], this may be computationally expensive, so a fixed radius may be used, the fixed radius may be the maximum radius used in the last approach or one that is function of the point spread function of the ultrasound [Estepar 2]. (FIGURA). In order to avoid the visible boundaries between the voxels assigned directly by the 2D ultrasound images during the BFS and the interpolated voxels computed during the HFS [Doud], a weighted distance can be used in the HFS in order to avoid far away pixels to contribute on the voxel and have smoother transitions, where the weight is the inverse distance between the center voxel and the non-empty voxel [Chen].

*Octrees and Morton Key*

The octree is a hierarchical representation of a array of unit cubes or voxels and it can be used as the primary data structure of 3D applications in image processing, such as object representation. An octree is a tree where the root node depicts the universe which encloses the entire object in a cubic space, then the space is recursively divided into octants whose nodes are either leaves or have eight children (FIGURA) [Rambally]. Usually the octree is stored as a record containing ten fields: eight fields contain pointers to the children; one contains a pointer to the parent; and the last describes the content of the field (color of the voxel) [Rambally]. Another popular way to encode the octrees is a linear octree encoding depending on Morton set [Morton], where no pointers are allocated to keep a spatial order of its nodes; instead, each leave is assigned a unique locational code formed by a sequence of octal digits that traces a path from the root to the node, where the digits identify the set of branches that must be traversed [Song]. This function maps each object to a unique key (Morton Key), where each key corresponds to some composite data describing the physical data inside the reconstruction space [Warren].

For naive radius neighbor search in an octree, the tree has to be traversed from its root recursively and investigate octants overlapping the search circle defined by an arbitrary query point ϵ and a radius . When an overlapping leaf octant is reached, all points *p* inside the octant are checked if . With this approach only points inside the overlapping leaf octants are compared and large subsets of points that are irrelevant for the query are discard [Behly] (FIGURA).

One of the main limitations of the used of Octrees is the amount of required memory. The size of the octree grows exponentially as the deepness (resolution) increases and the memory usage increases in order to storage the complete data structure [Bedorf]. One variation of Octrees, called Sparse Octrees, allows to create only the nodes that contain information, leaving the rest without definition or memory usage [Laine].

**3. Proposed Method: Octree Based Volume Reconstruction**

In this work we show an implementation of the PNN method using a Sparse Octree (SO) in order to reduce the computational time. As explained before, the PNN is composed of two steps: The bin-filling step (BFS) and the hole-filling step (HFS); in this implementation the BFS is used to create the Sparse Octree using the pixel information, while the HFS step creates the interpolated nodes of the Octree using the prior information obtained in the BFS.

*Initialization*

In this work we used particles as input data for the SO, these particles are the pixels of the acquired ultrasound images; one pixel is consider one particle. Initially each image is loaded in memory and processed in order to find the spatial coordinates of each pixel in each image using the estimated calibration parameters found in the calibration step, obtaining a point cloud that compose the initial volume. The resolution of the SO is measure in nodes and the size of the nodes depends of the dimensions of the point cloud that fills it, since the Octree is constructed as the minimum box that can contain all the particles that are included in the initial volume; from this bounding box the Octree is divided in octants and the node dimensions depend of the octree used levels.

*Morton Keys implementation*

The particles does not have a defined volume, since they are points in the space defined by their spatial coordinates (, , ) and the gray-level of the pixel. Since we try to arrange the particles and relate them directly with the data structure, we choose to represent the spatial data using Morton keys. Each particle spatial coordinates indicates its position in the space, to convert the spatial coordinates to its corresponding Morton key we took the binary representation of each coordinate and decompose it in triplets containing interleaved bits of each spatial coordinate; figure show an example of a Morton key (FIGURA). Each node of the SO also uses a Morton Key as spatial identifier; this allows arranging the nodes and finding a relation between the node and its particles, using the center of the node as reference point. Since the size of the volume is , where is the resolution of the Octree, the center of the nodes are always power of two, making the identification of the Morton Keys faster without reading the whole key. Using Morton Keys for both, particles and nodes, gives a direct relation between them and allows us to use a bit mask (Morton Mask) in order to determine the node to which a particle belongs at any level of the SO.

*Reconstruction Implementation*

Once the point cloud is constructed, each particle is used to fill the SO in the BFS. First the Morton Key of each particle is computed and introduce into the data structure; then the Morton Key of each node is also computed using the center of the node as reference point. The Morton Mask is used to process all the particles in parallel to determine to which node corresponds each particle. Once the belonging node is identified, the gray-level of the particle is averaged with the gray-level of the node.

The HFS is done after all particles have been assigned to a node. The Morton Keys allow a faster and efficient way to fill the empty voxels in the volume. The first step in the HFS is to find all the empty voxels in the volume. This empty voxels will only be created as nodes of the Octree when non-empty voxels are found inside a neighborhood of fixed or growing radius as explained before. Each neighborhood is computed using the Morton key of the central node, decodifying the Morton key in its three dimensions () and modifying it to match the neighborhood nodes adding or subtracting in each coordinate. These new coordinates will create a new Morton key that corresponds to a neighborhood node. Each node inside the radius will be visited in order to verify if the node should be created or not, founding two cases: in case all voxels in the neighborhood are empty, the central node will not be created since it is an isolated node; in case at least one neighborhood is non-empty, the gray-level of the central node is computed as a distance weighted interpolation of the gray-level values of all the non-empty voxels inside the neighborhood. When an empty node is created, this node is not taken into account to fill other empty-nodes, since it is an artificial node.