done recursively to produce a labeling in which points that are near each other have similar labels. This is similar to a separator-based technique for graph compression through relabeling[19] except that it occurs before any edges have been added to the mesh.

If not all vertices are known before the algorithm begins, our algorithm can assign a sparse labeling to the initial vertices. When a new vertex is added, it is assigned a label that is close to the labels of its neighbors. It would be inefficient to allocate storage for every possible label; instead, our algorithm uses an extra level of indirection to map vertex labels to memory blocks.

6. Implementation

6.1. *2D Triangulation*

Our 2-dimensional compressed data structure is implemented as follows.

For difference encoding our structure uses the *nibble code*, a code of our own devising that stores integers using 4-bit "nibbles". Each nibble contains three bits of data and one "continue" bit. The continue bit is set to **0** if the nibble is the last one in the representation of a integer, and **1** otherwise. Blandford, Blelloch, and Kash[42] found that this code is a good compromise between speed and space-efficiency.

It is sometimes necessary to store an extra bit b with a value v. This is accomplished with a shift operation: $v' \leftarrow 2v+b$. In particular, if any value might be negative, our difference coder stores its absolute value plus a sign bit: $v' \leftarrow 2|v|+\mathtt{sign}(v)$.

A vertex is represented with a nibble code for the degree of the vertex, followed by nibble codes for the differences to each of the vertex's neighbors. Our implementation stores two additional "special-case" bits with each neighbor to provide information about the triangle that precedes it in the link. One bit is set to indicate a gap in the link set: it indicates that there is no triangle preceding that neighbor in the mesh. The other bit is set when data is associated with the triangle preceding that neighbor. In this case, the code for that neighbor is followed with a nibble code representation of the data.

As an optimization, note that for many vertices none of the special-case bits will be set. Our implementation stores a bit with the degree of each vertex to indicate if none of its special-case bits are set; if this is so, those bits are omitted in the encoding of that vertex.

Our implementation stores the nibble codes for each vertex in an array containing one seven-byte block per vertex. If a block overflows (that is, if the storage needed is greater than seven bytes), additional space is allocated from a separate pool of seven-byte blocks. The last byte of the block stores a pointer to the next block in the sequence. Our implementation uses a hashing technique to ensure that the pointer never needs to be larger than one byte. This requires a hash function that maps (address, i) pairs to addresses in the spare memory pool. Our implementation tests values of i in the range 0 to 127 until the result of the hash is an unused block. It then uses that value of i as the pointer to the block. Under certain