

FIG. 1

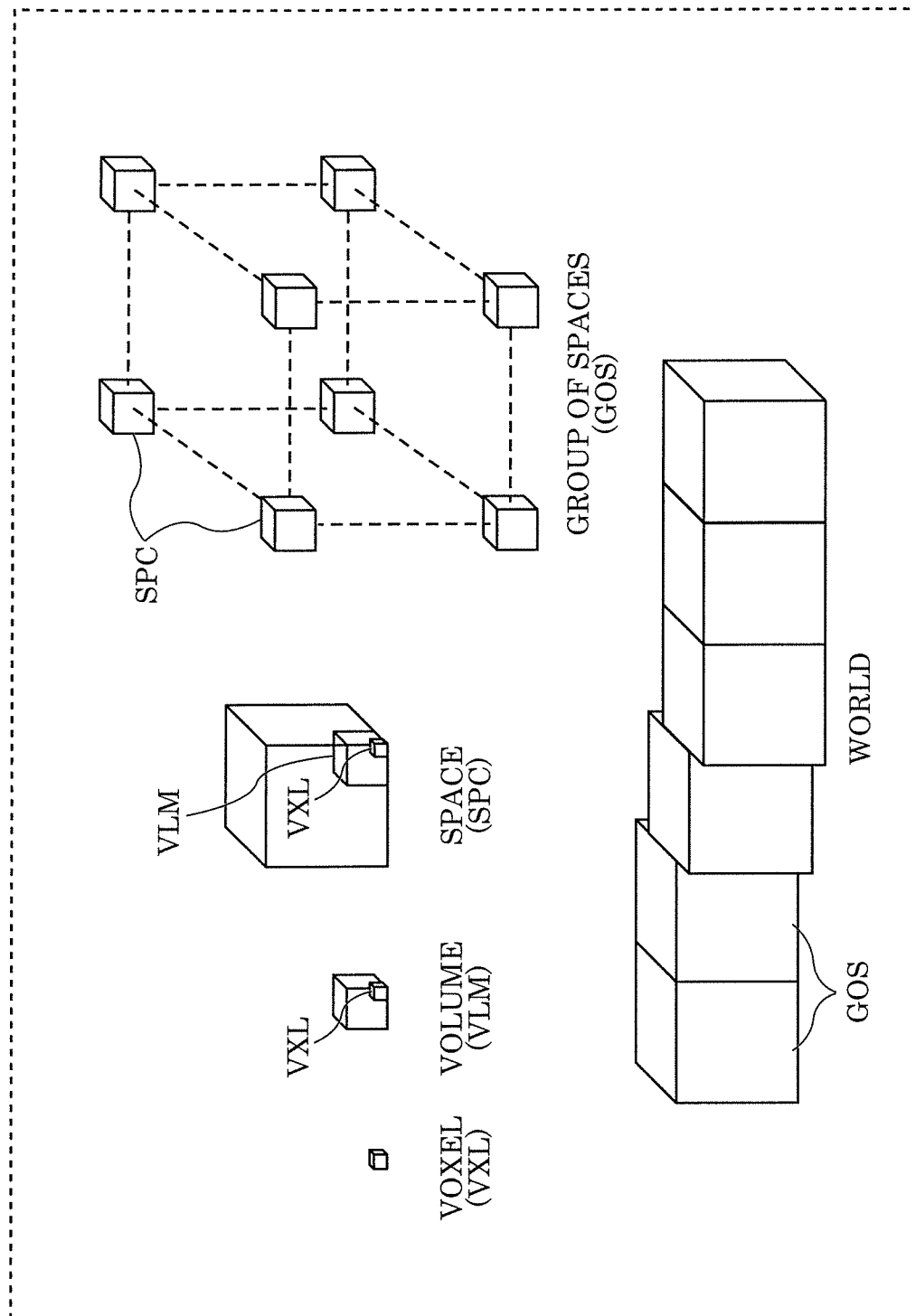


FIG. 2

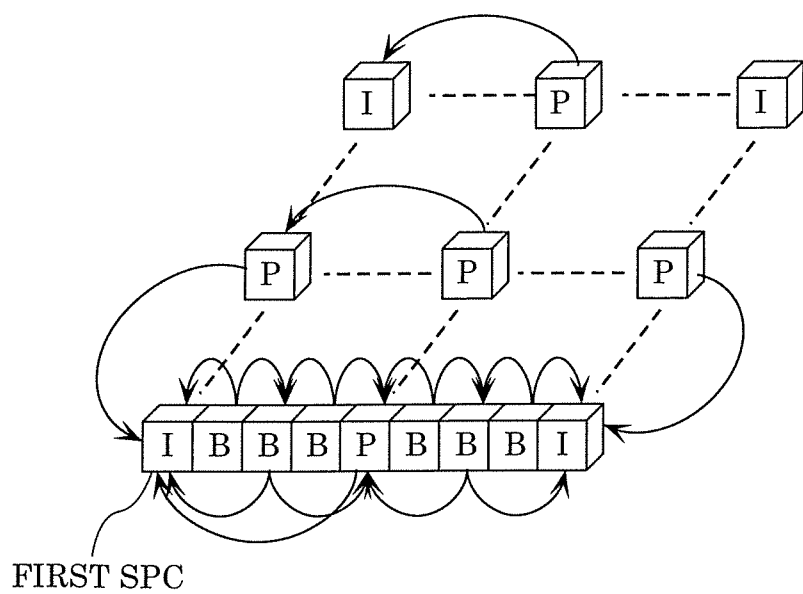


FIG. 3

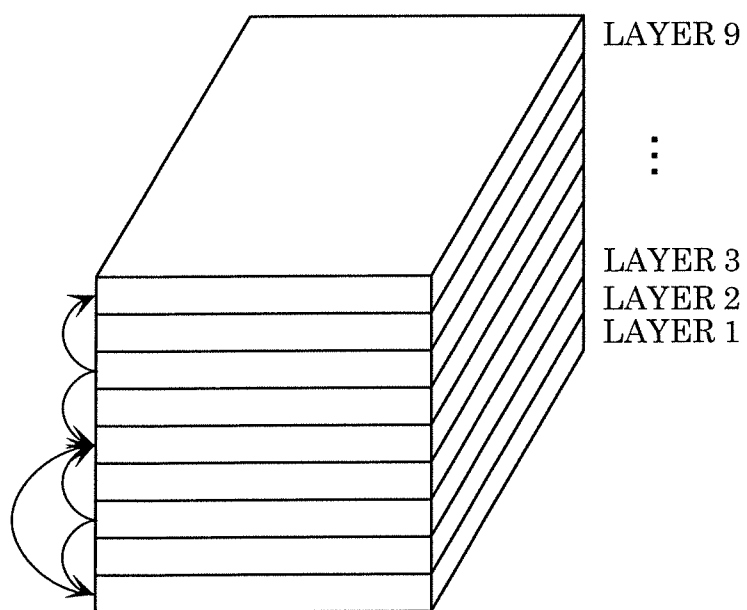


FIG. 4

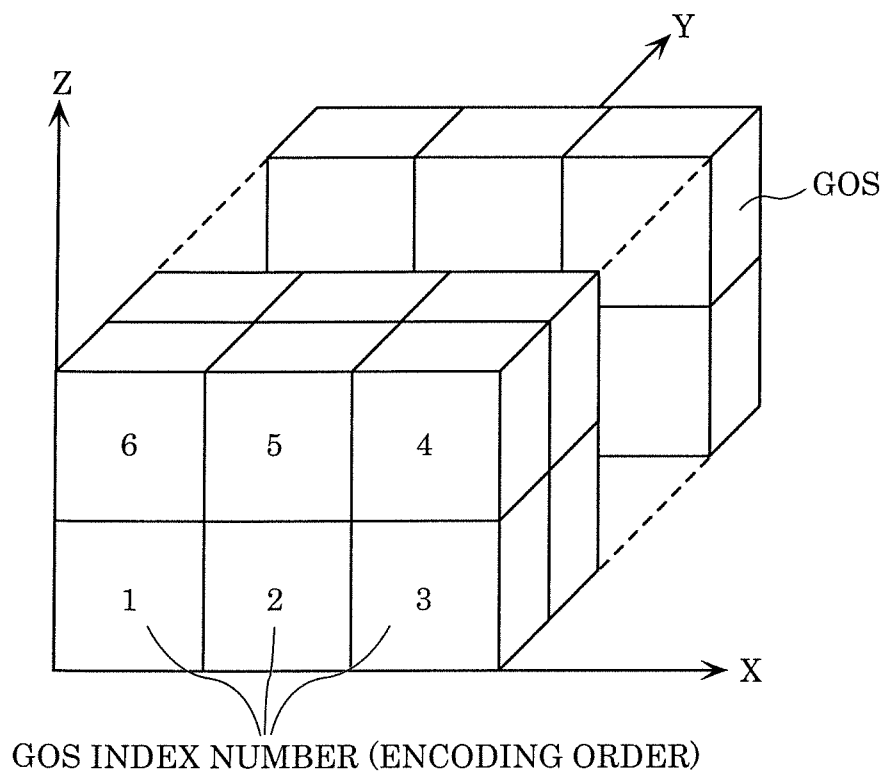


FIG. 5

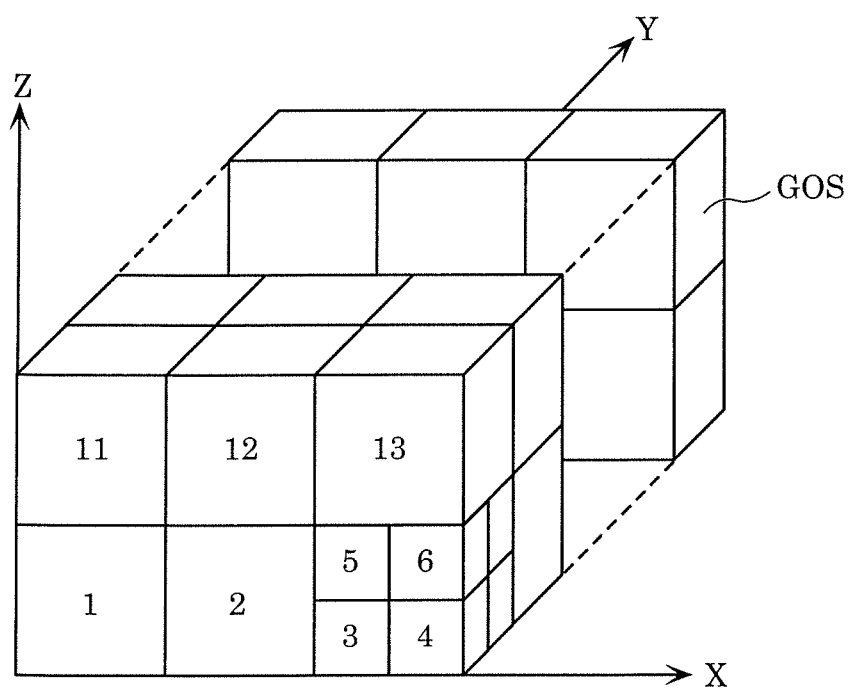


FIG. 6

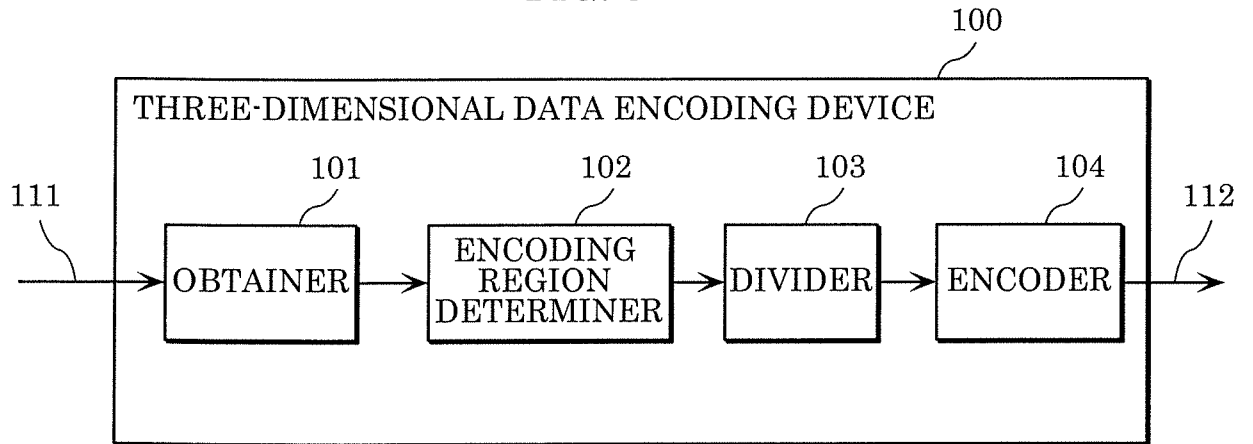


FIG. 7

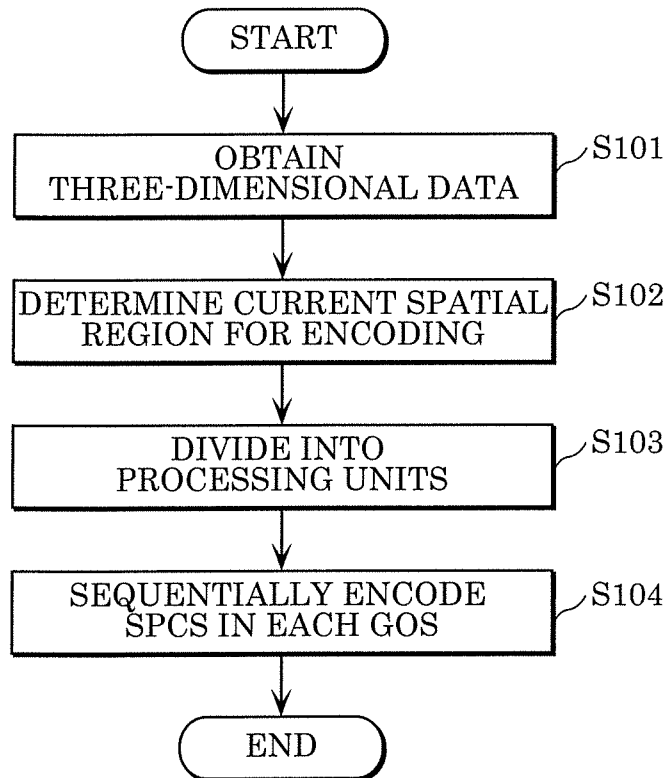


FIG. 8

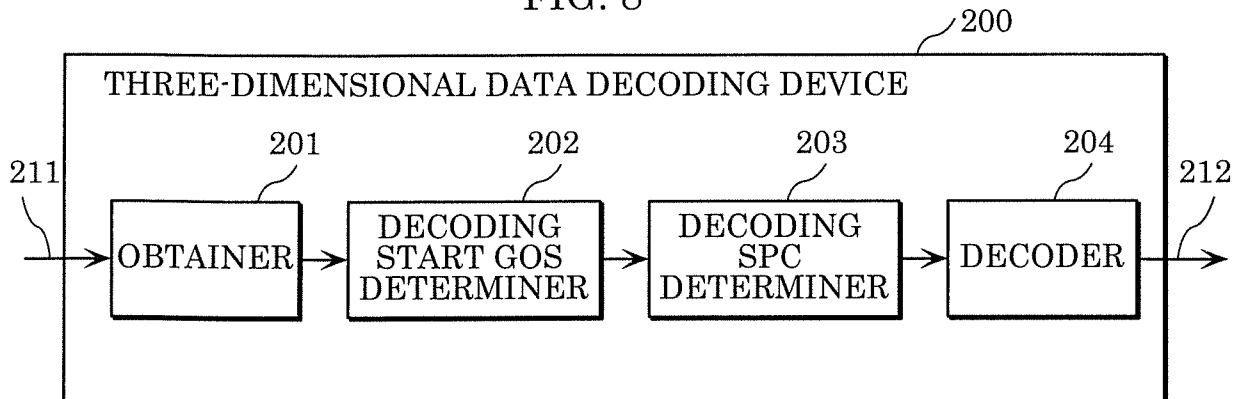


FIG. 9

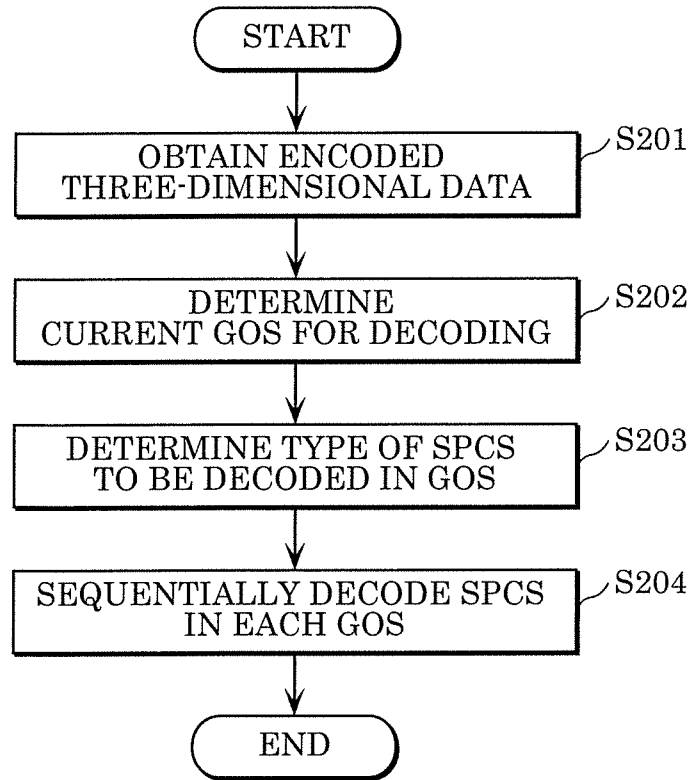


FIG. 10

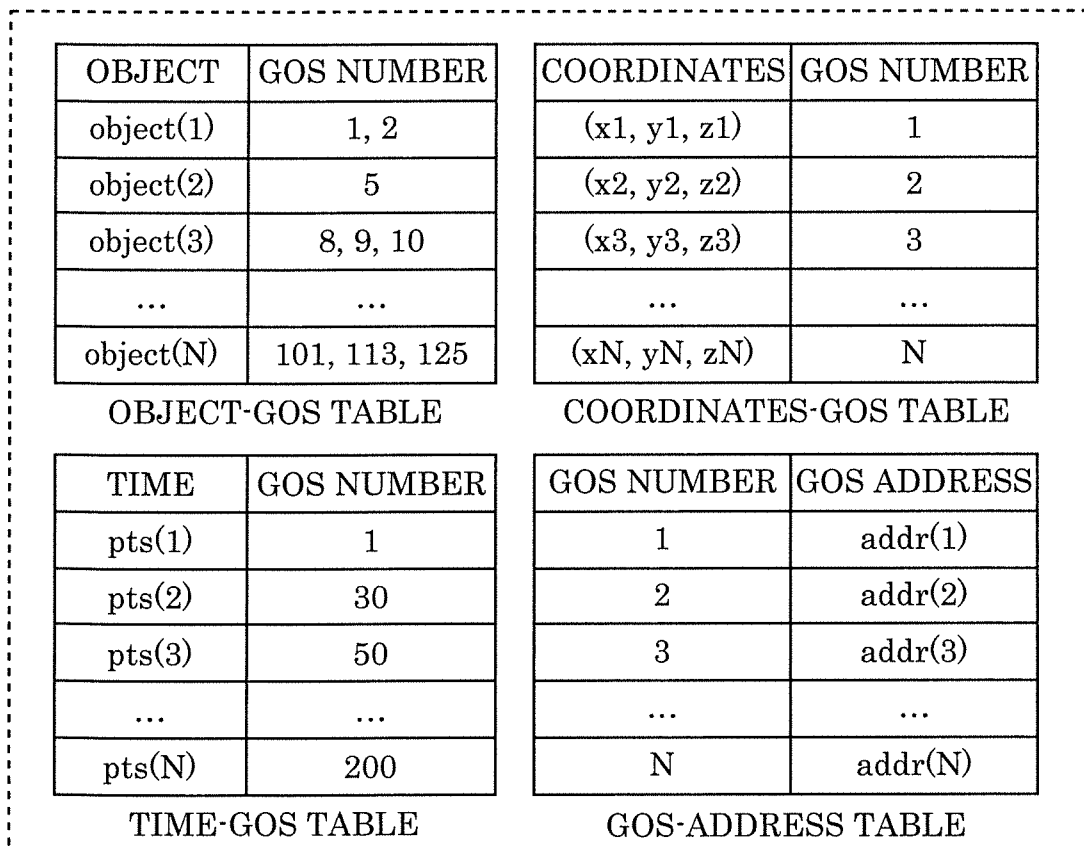


FIG. 11

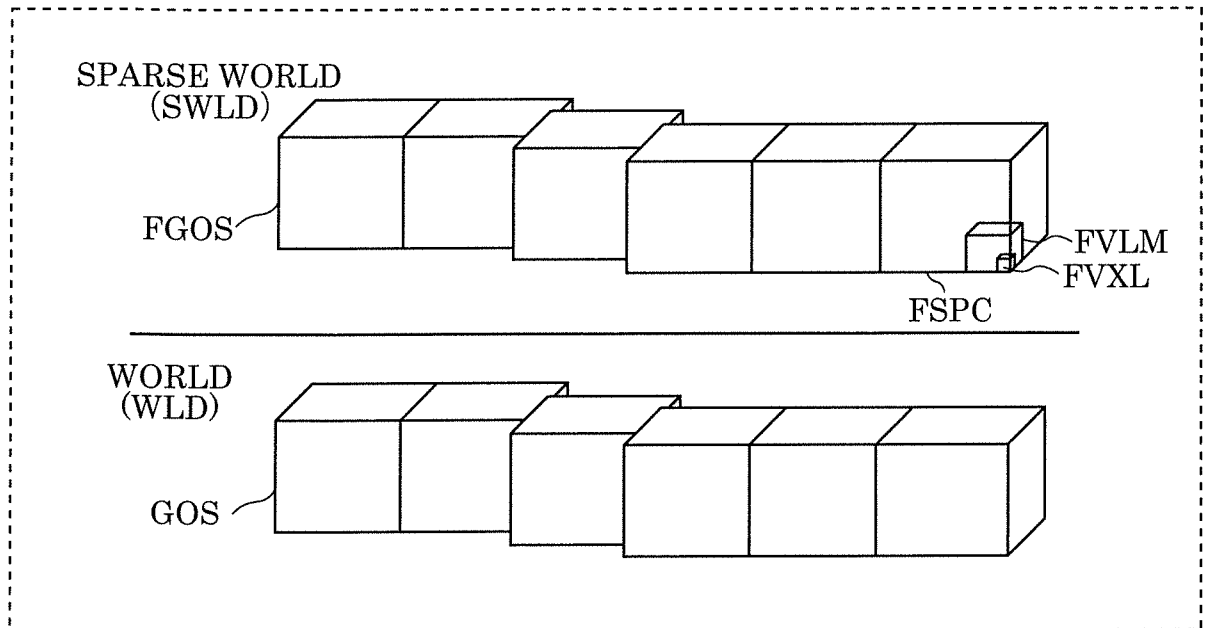


FIG. 12

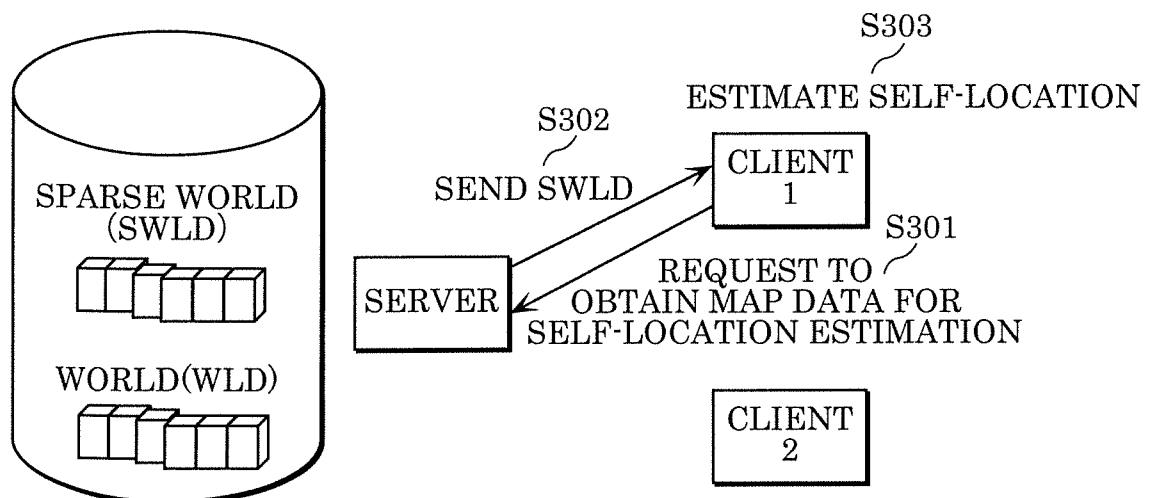


FIG. 13

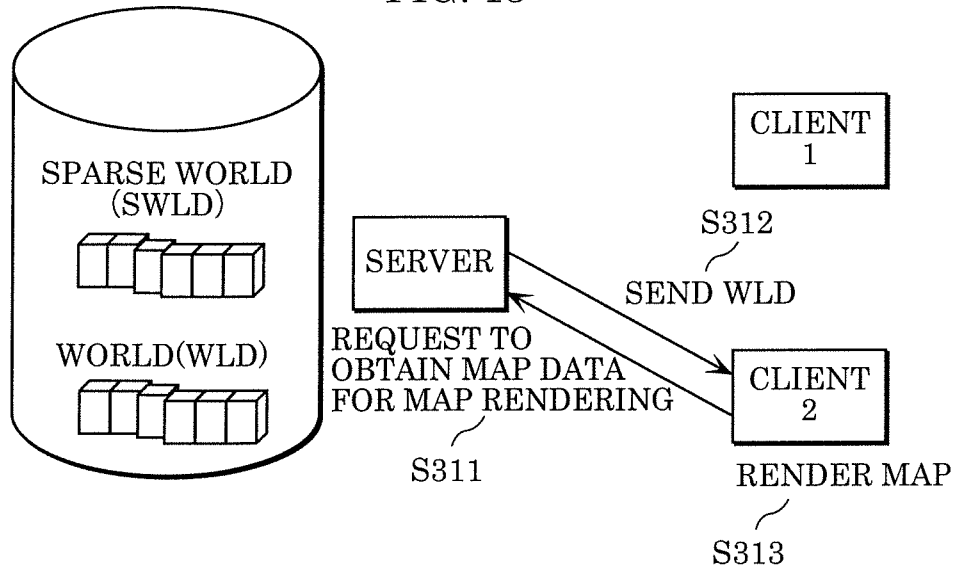


FIG. 14

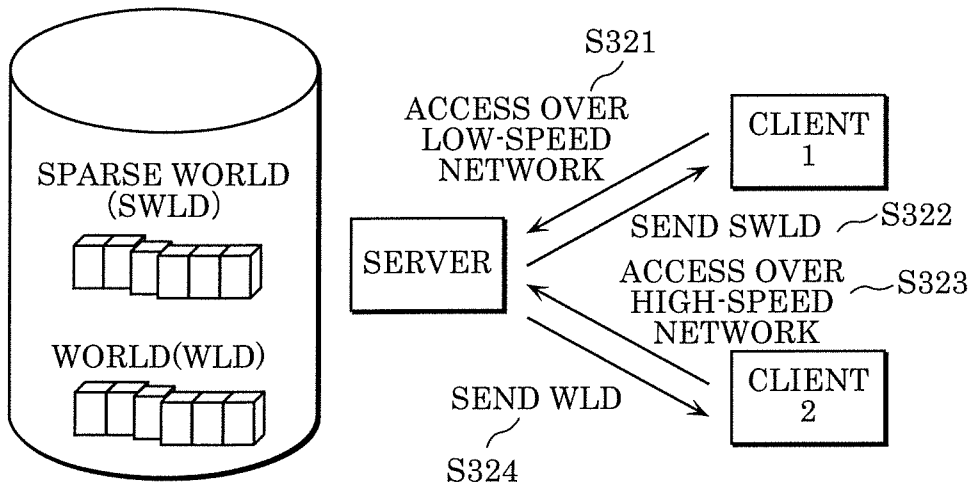


FIG. 15

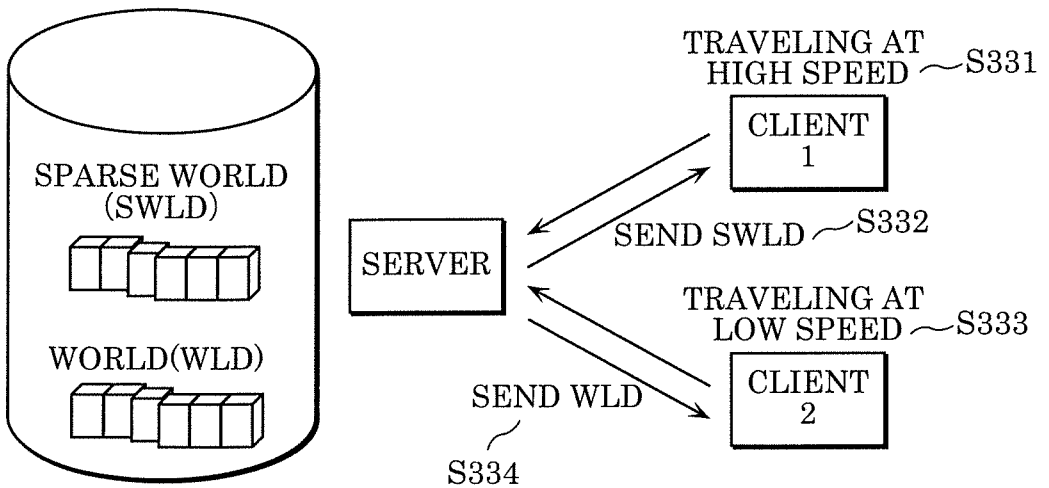


FIG. 16

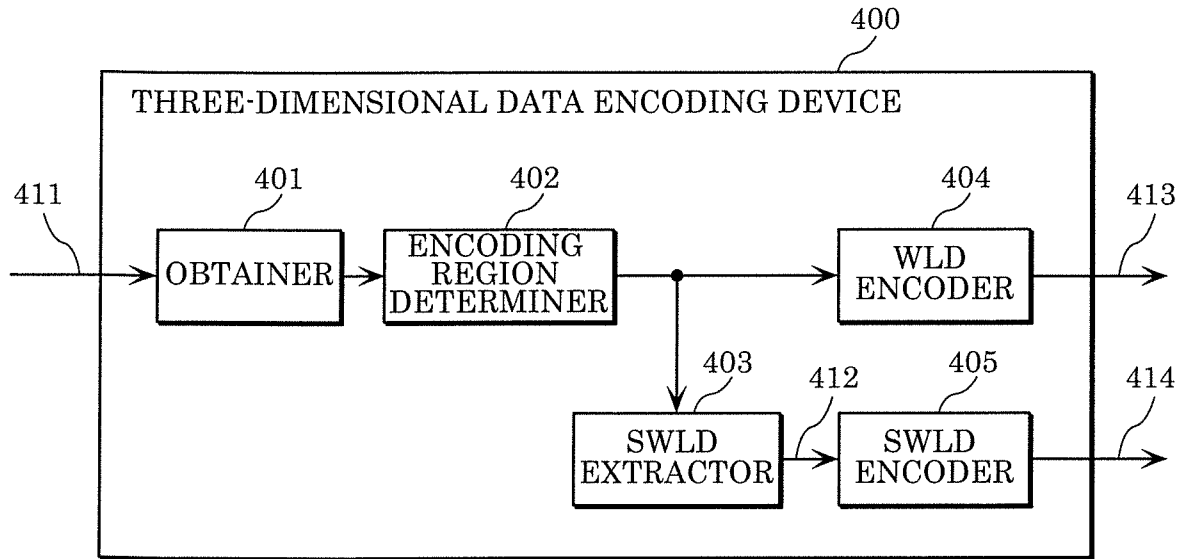


FIG. 17

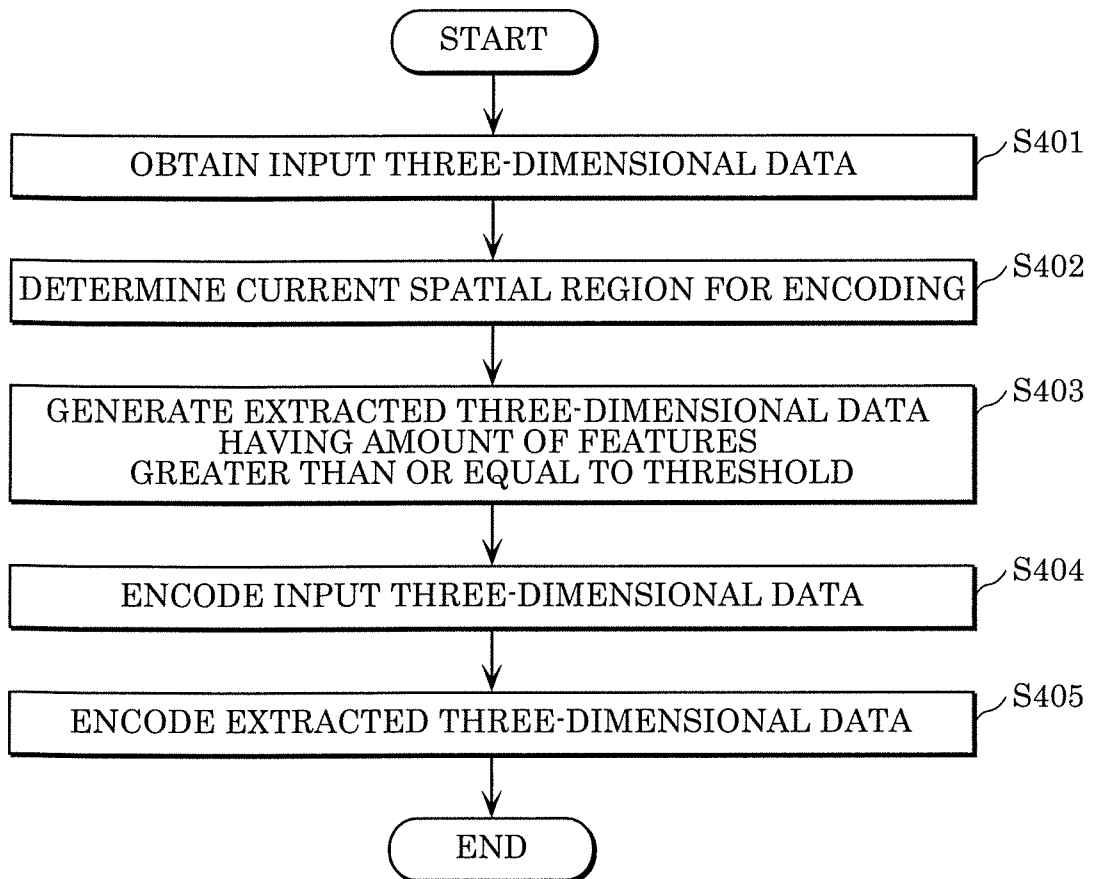


FIG. 18

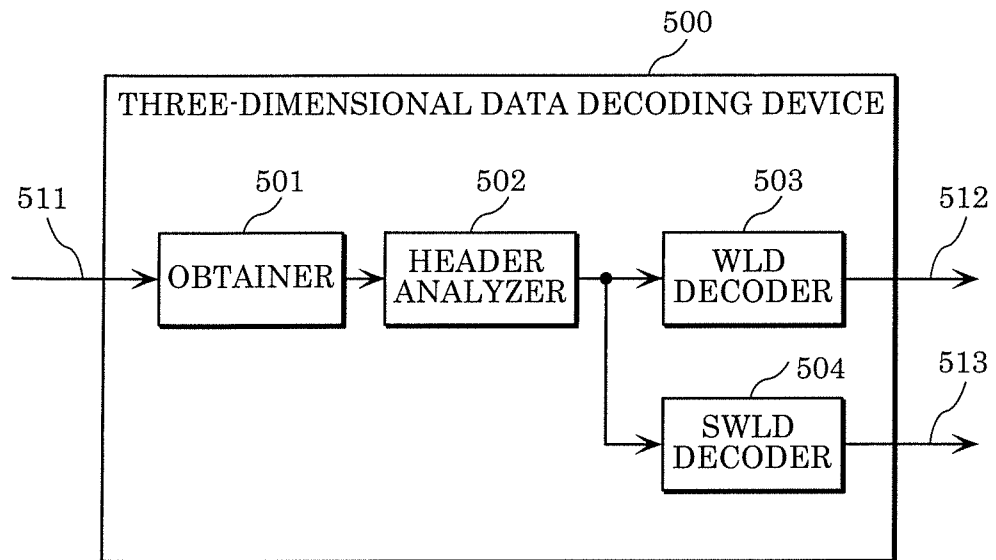


FIG. 19

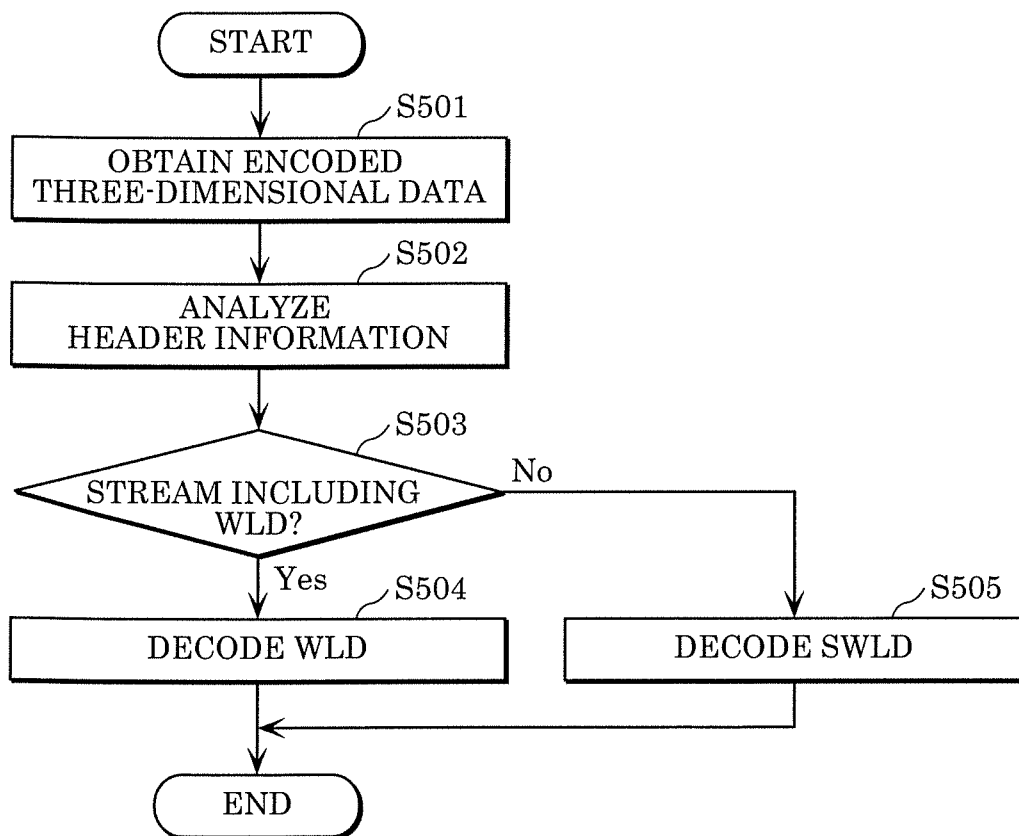


FIG. 20

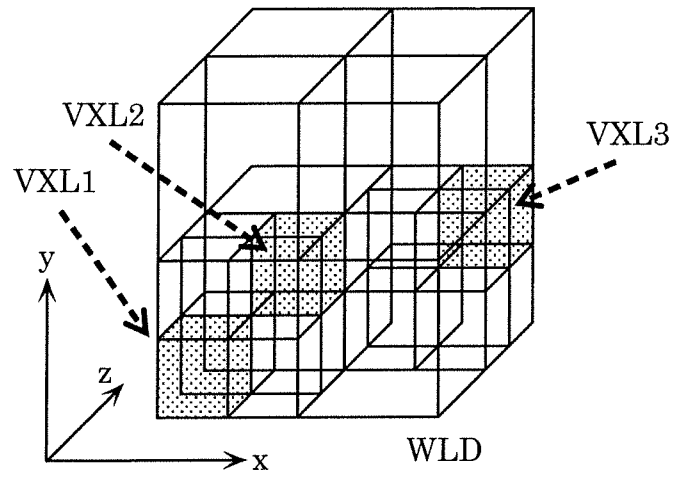


FIG. 21

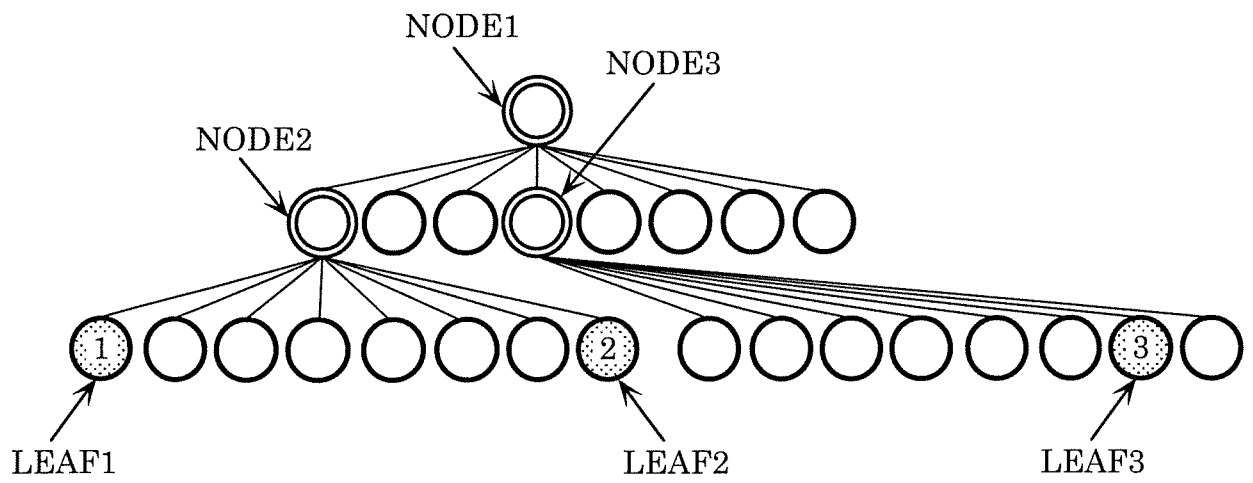


FIG. 22

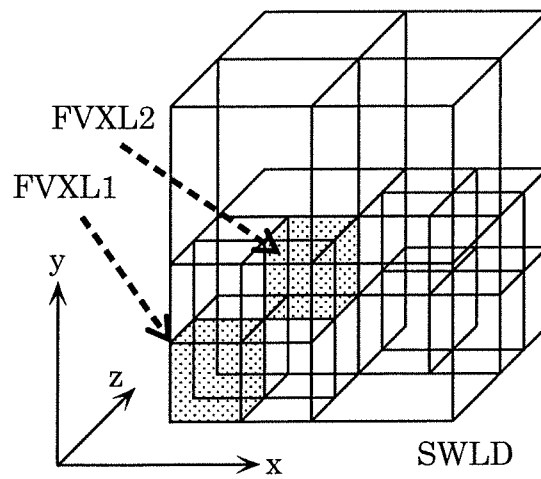


FIG. 23

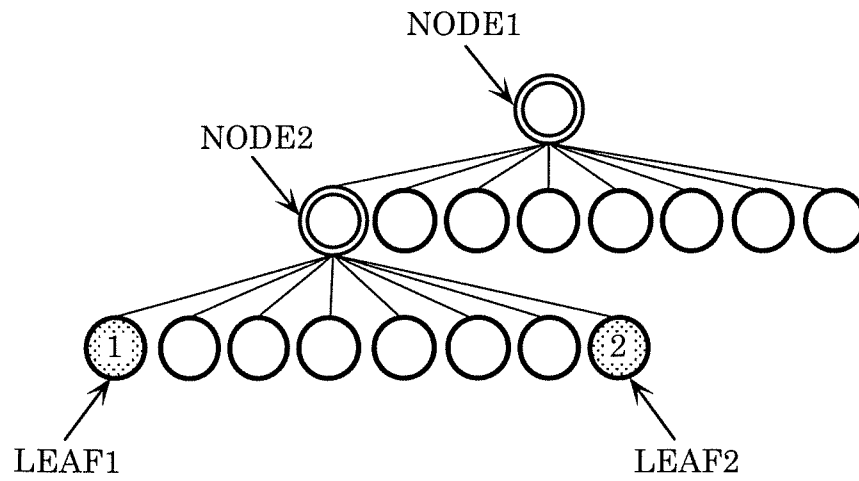


FIG. 24

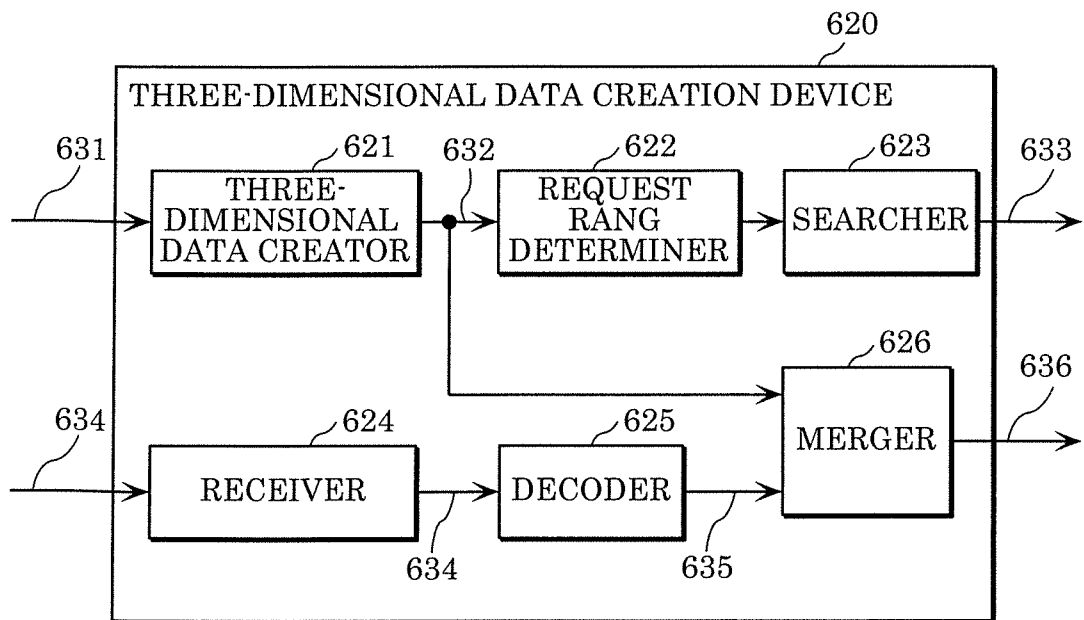


FIG. 25

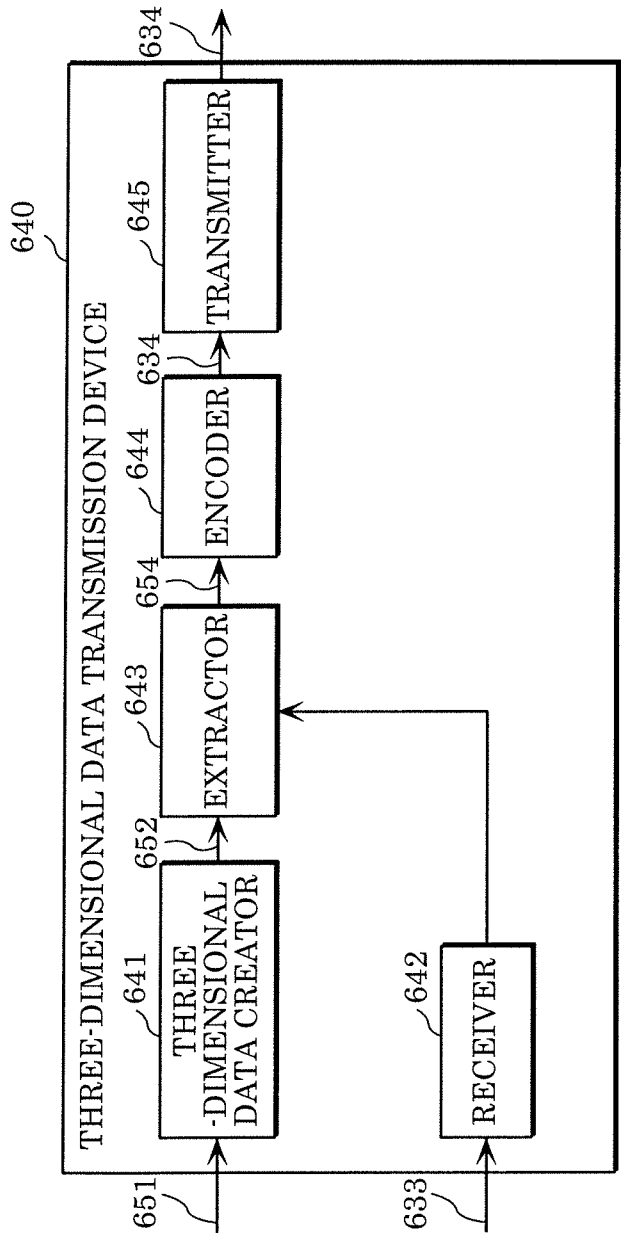


FIG. 26

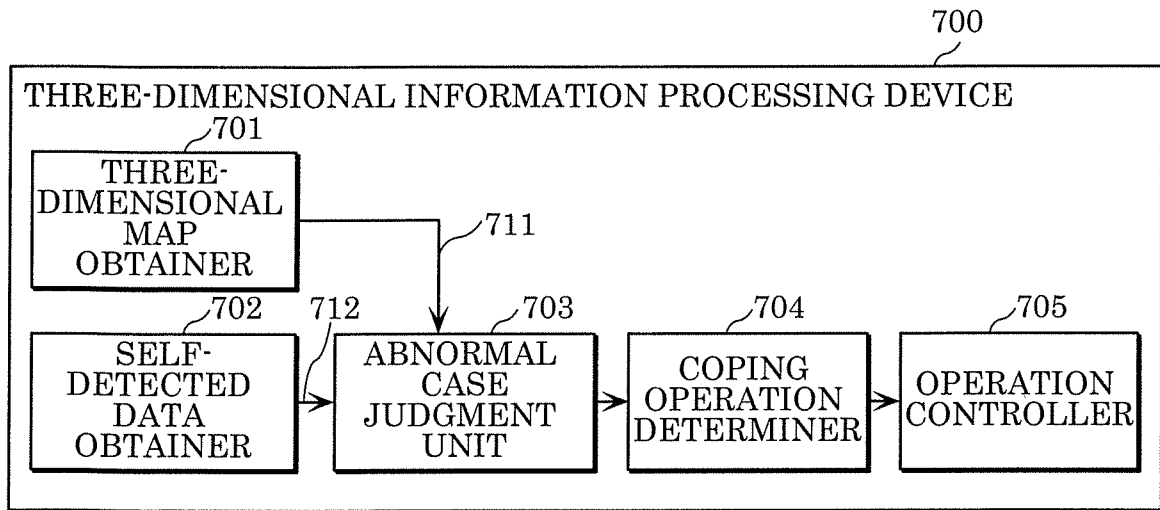


FIG. 27

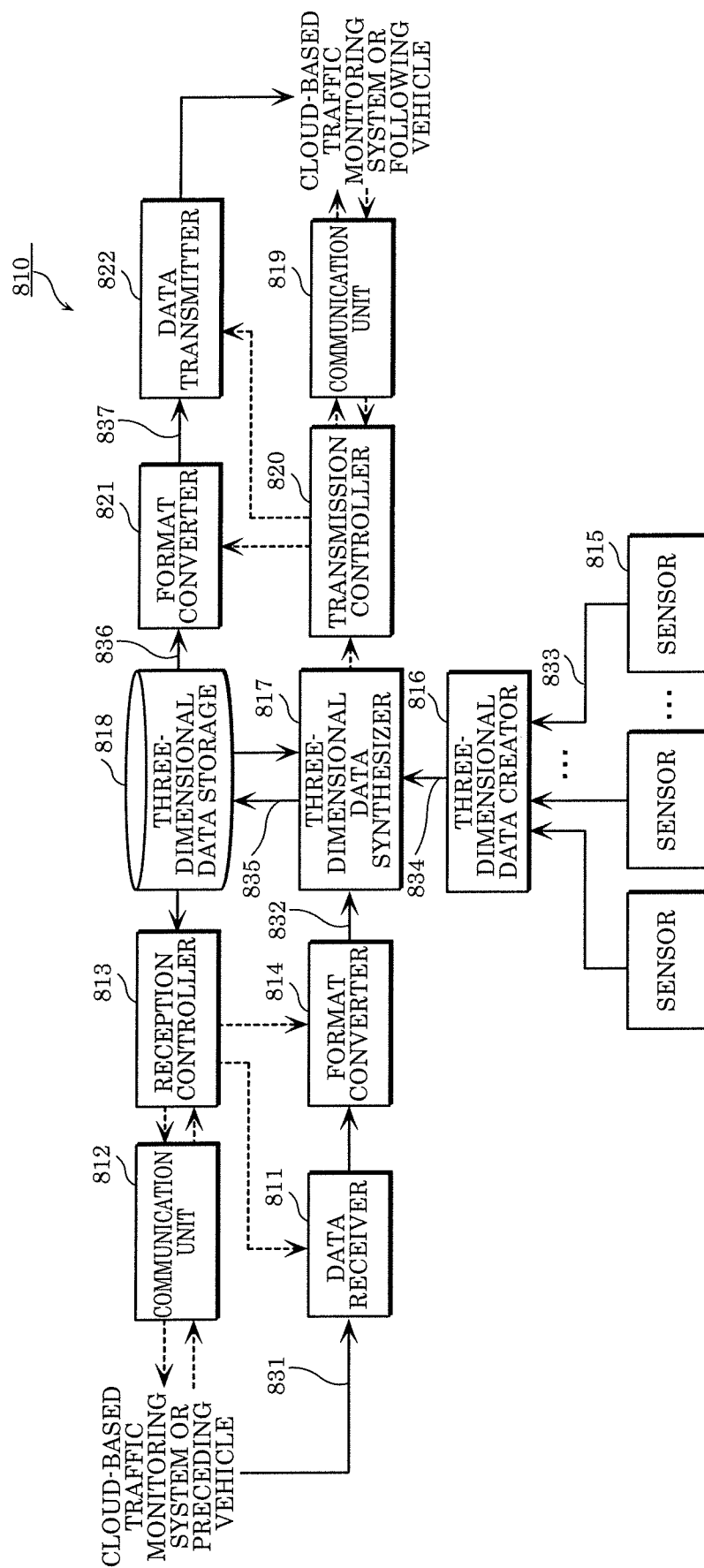


FIG. 28

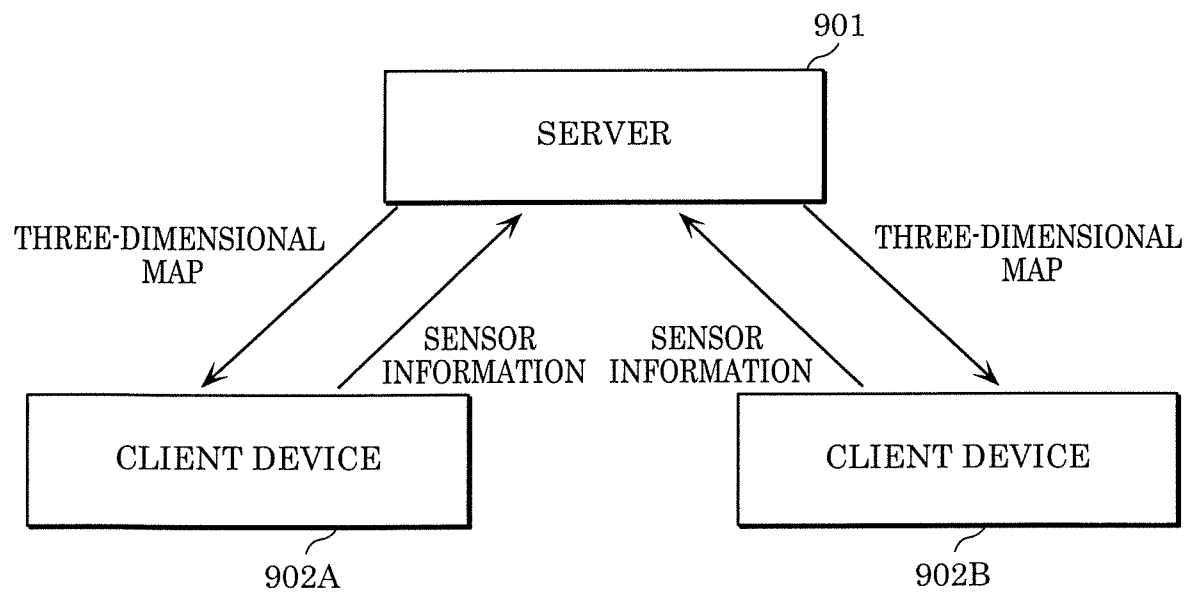


FIG. 29

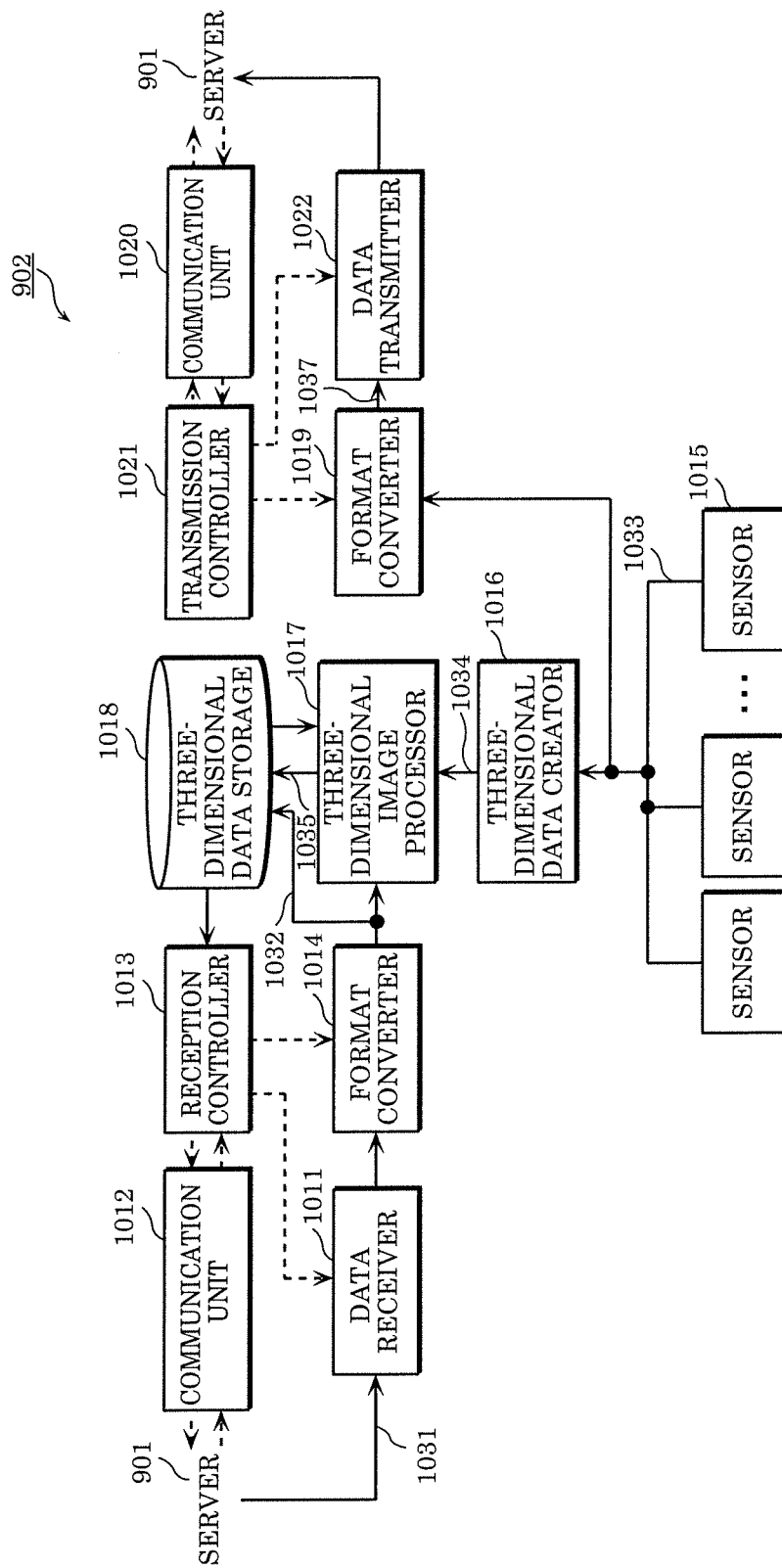


FIG. 30

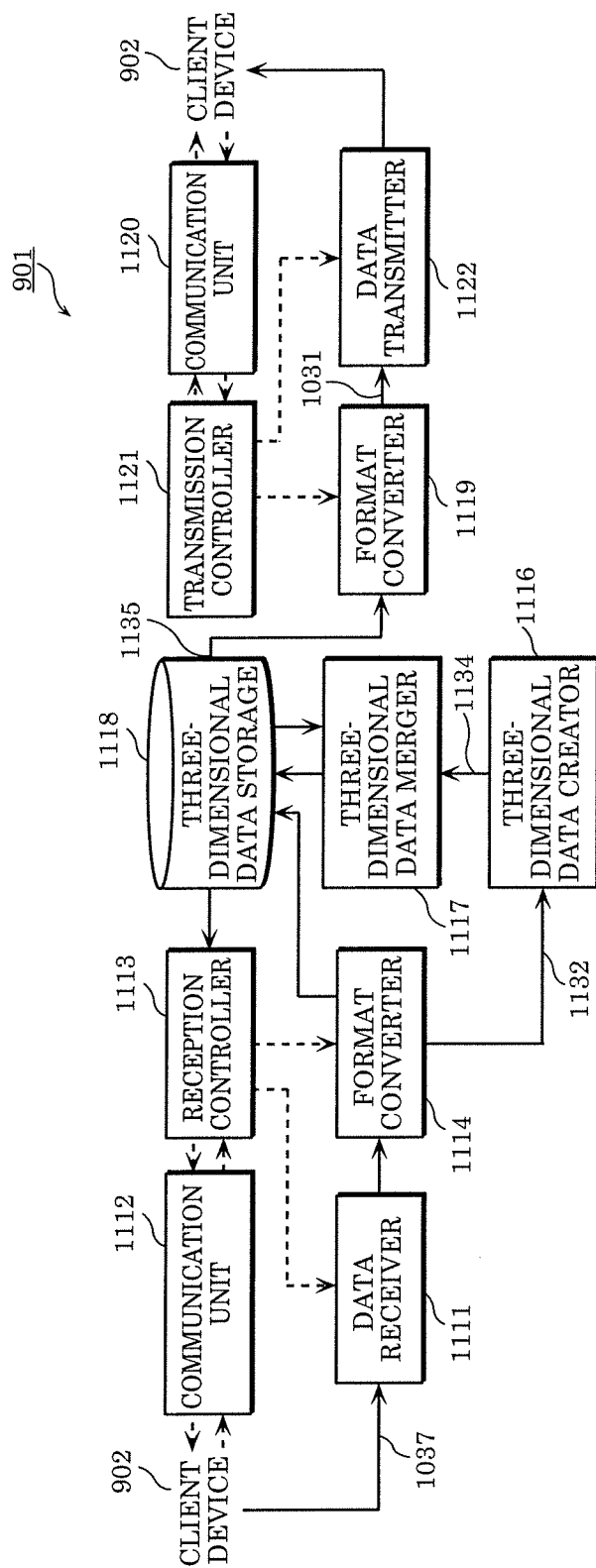


FIG. 31

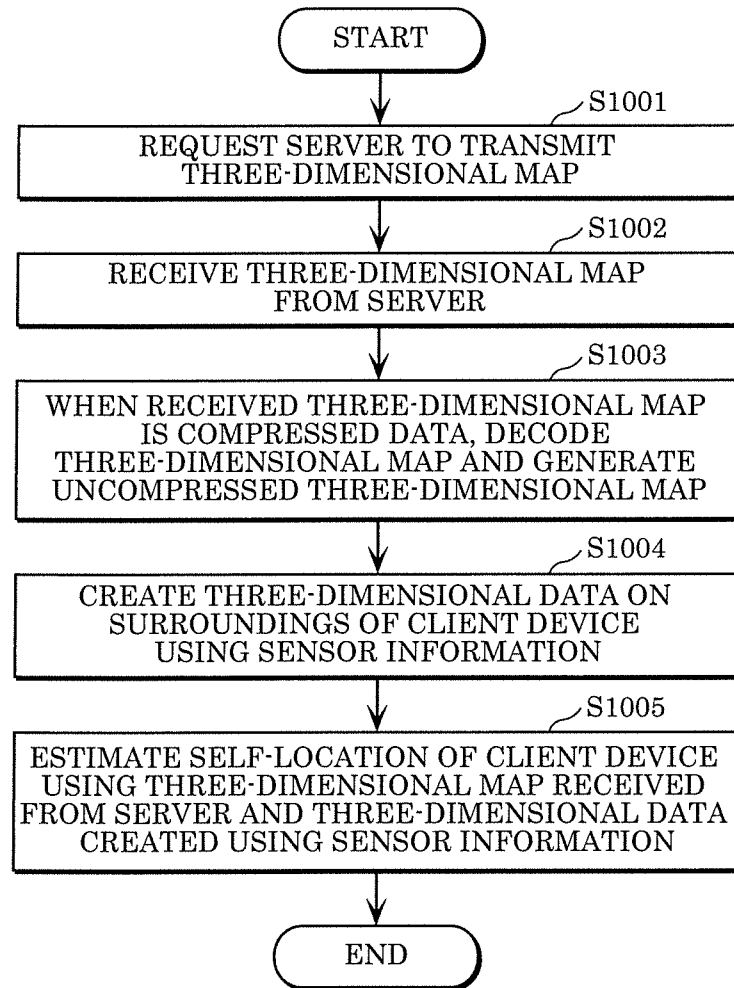


FIG. 32

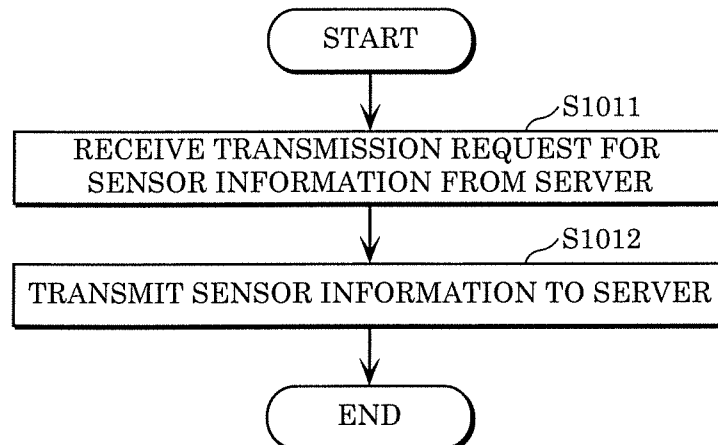


FIG. 33

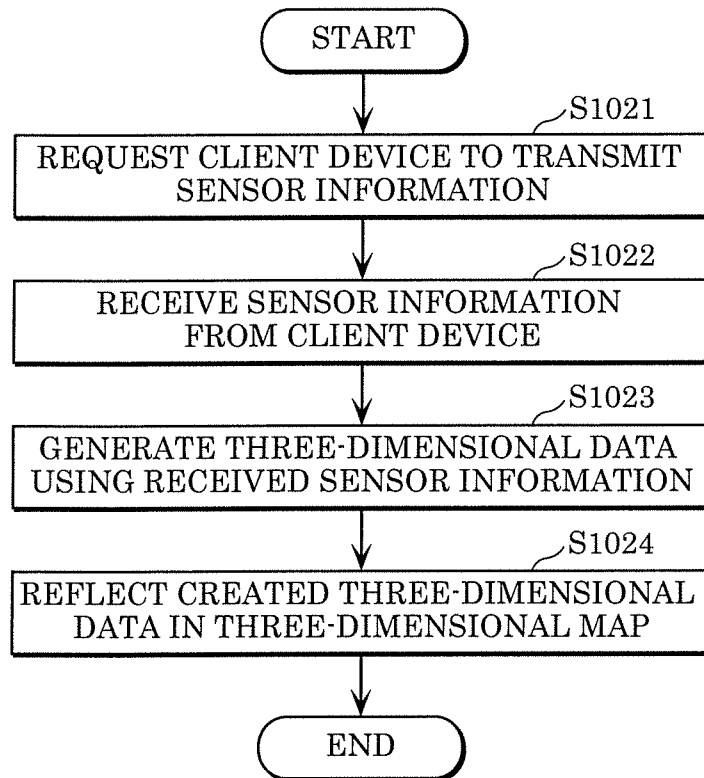


FIG. 34

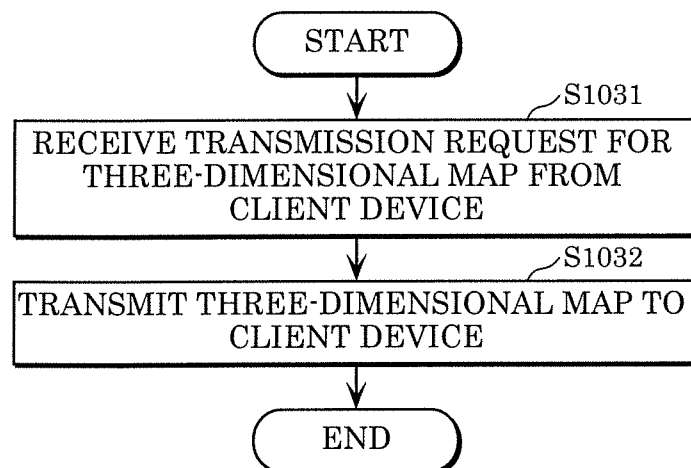


FIG. 35

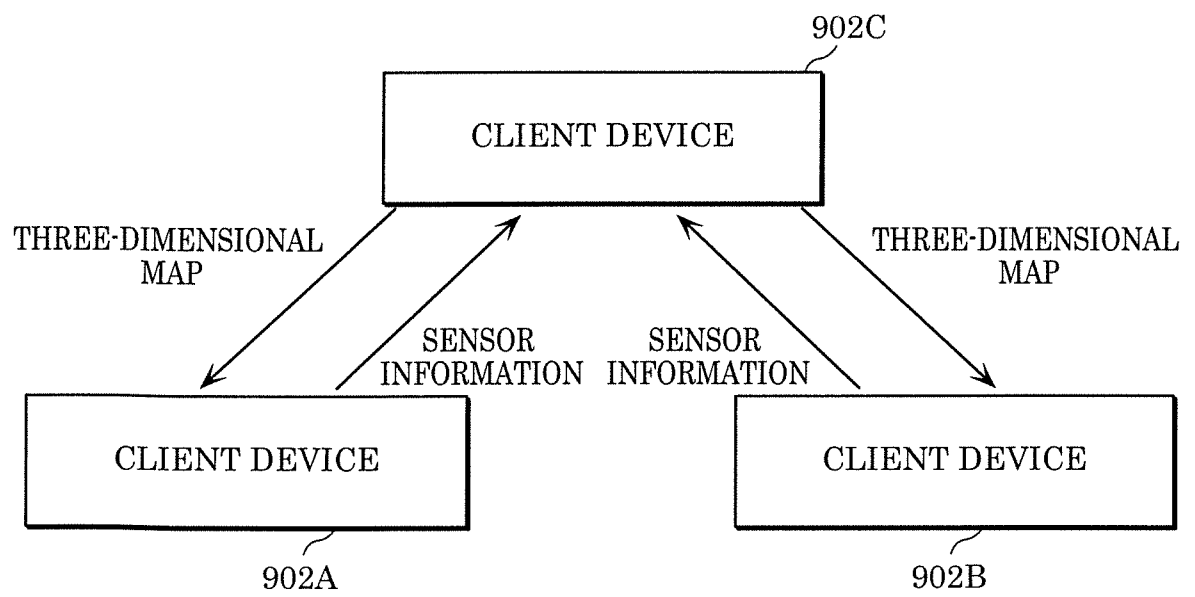


FIG. 36

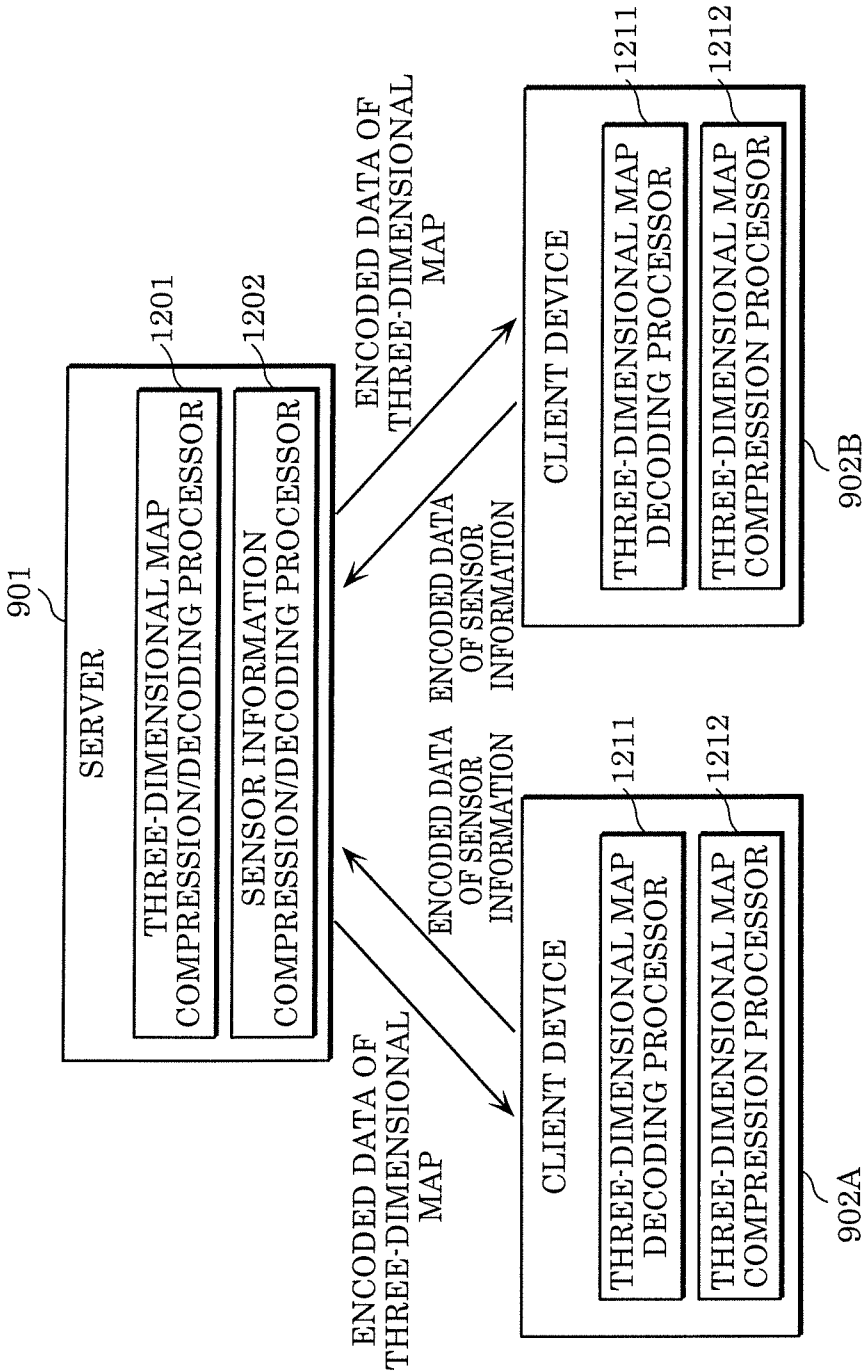


FIG. 37

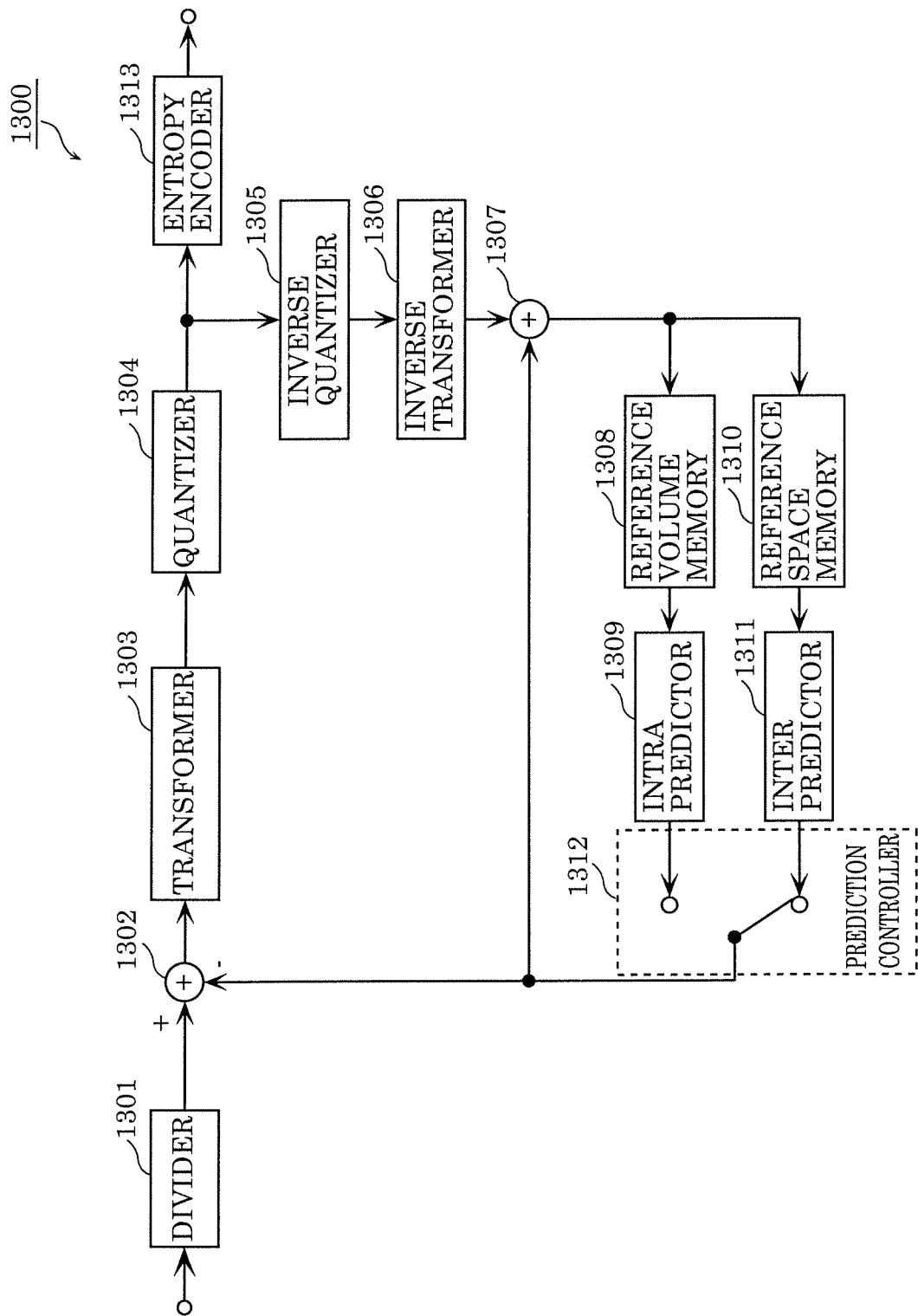


FIG. 38

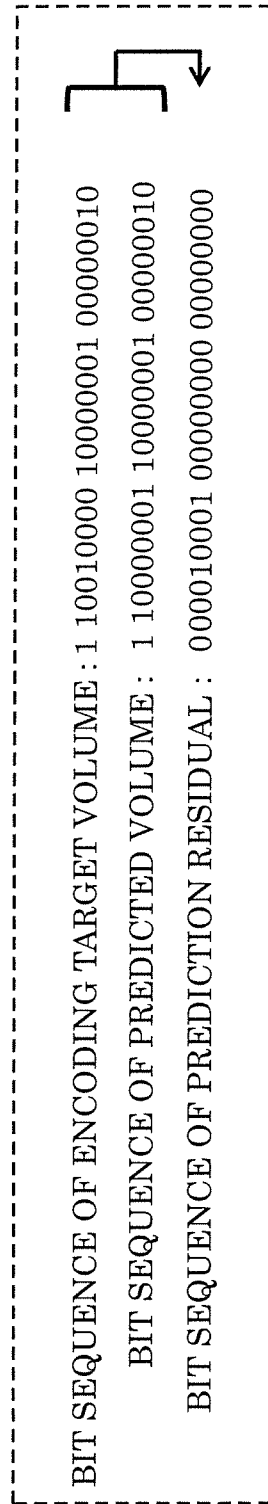


FIG. 39

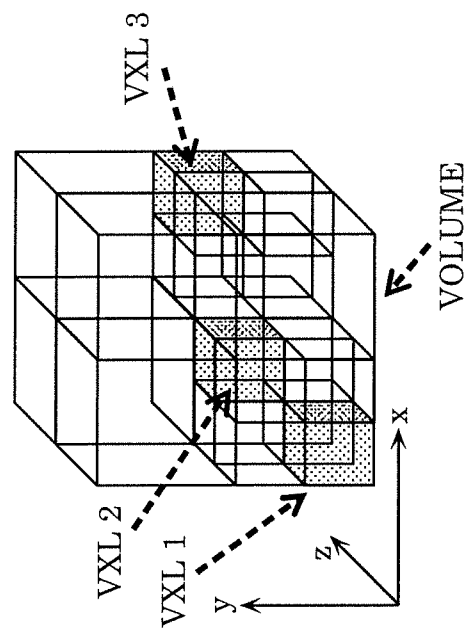


FIG. 40

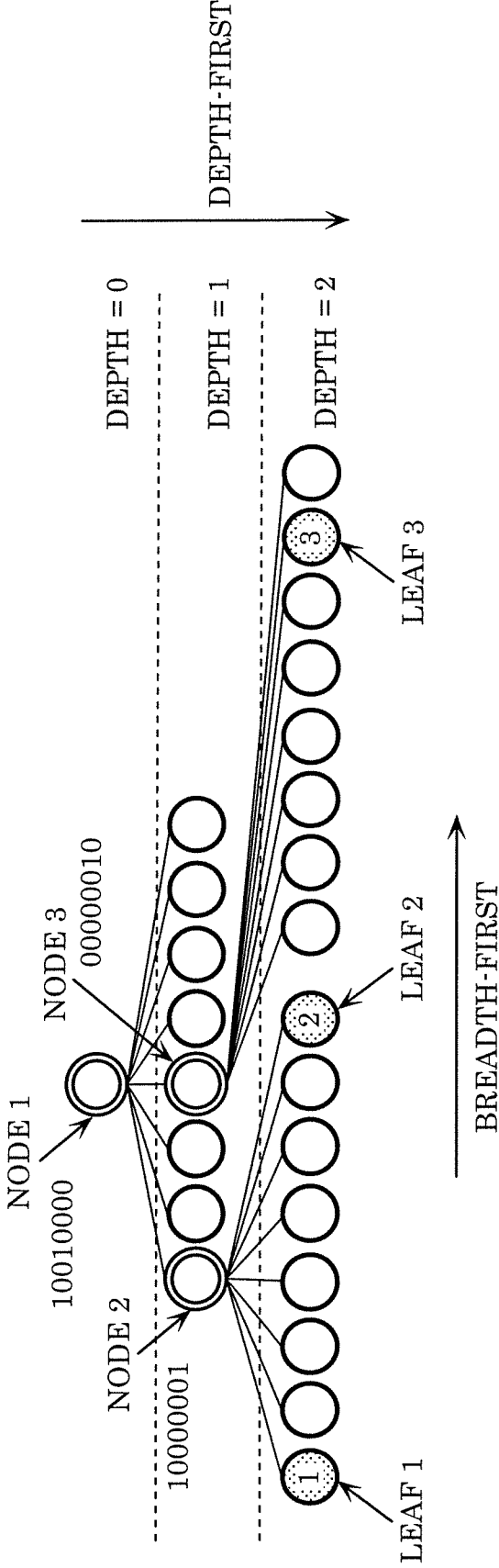


FIG. 41

A		BIT SEQUENCE WHEN SCANNING NODES AND LEAVES BREADTH-FIRST	: 10010000 10000001 00000010 leaf1 leaf2 leaf3
B		BIT SEQUENCE WHEN SCANNING NODES AND LEAVES DEPTH-FIRST	: 10010000 10000001 leaf1 leaf2 00000010 leaf3

FIG. 42

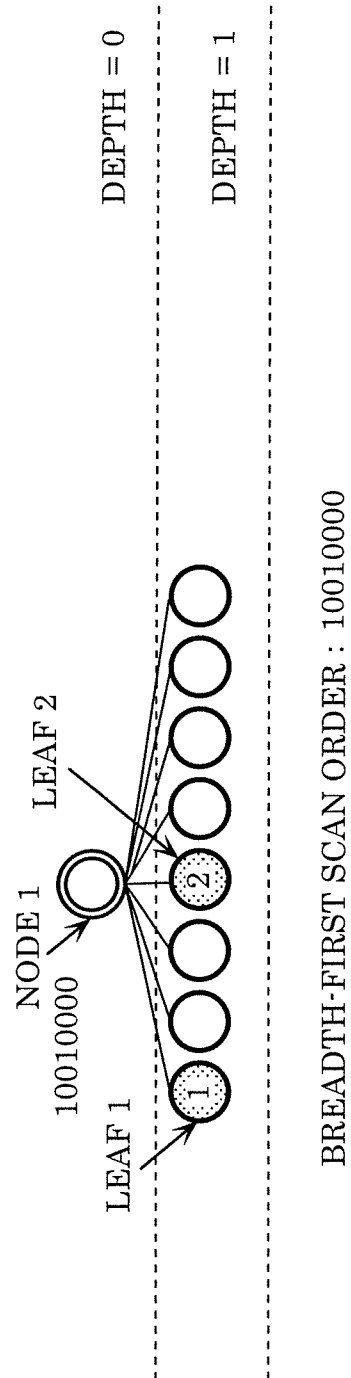


FIG. 43

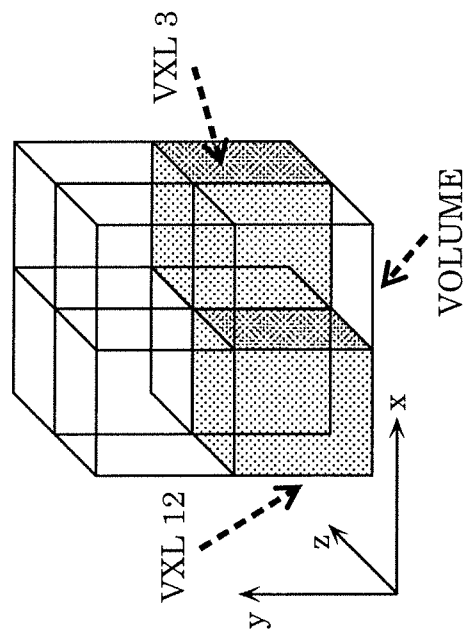


FIG. 44

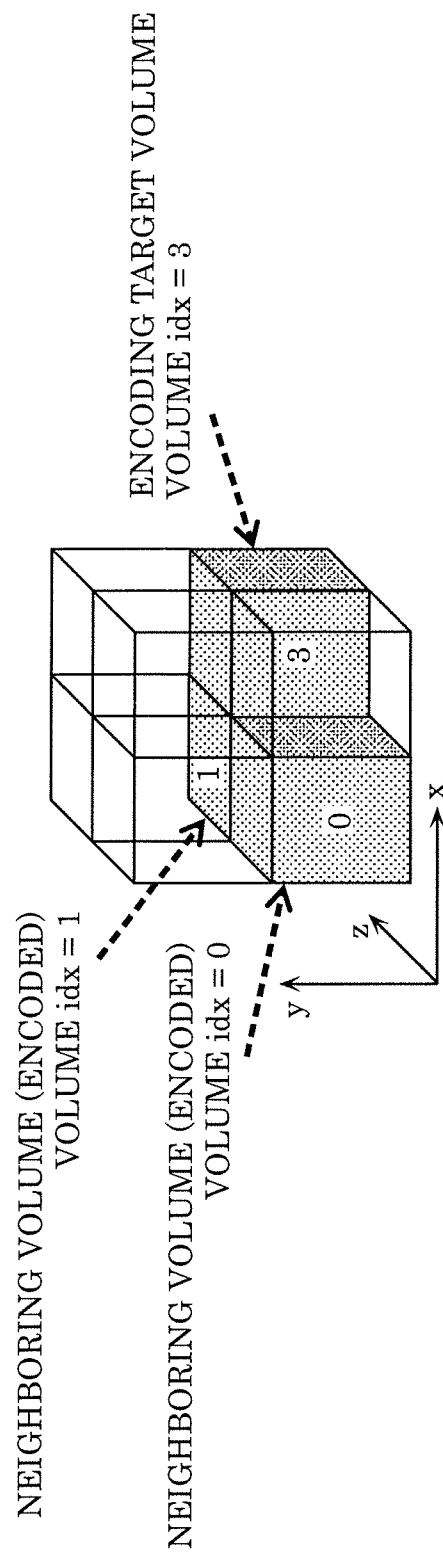


FIG. 45

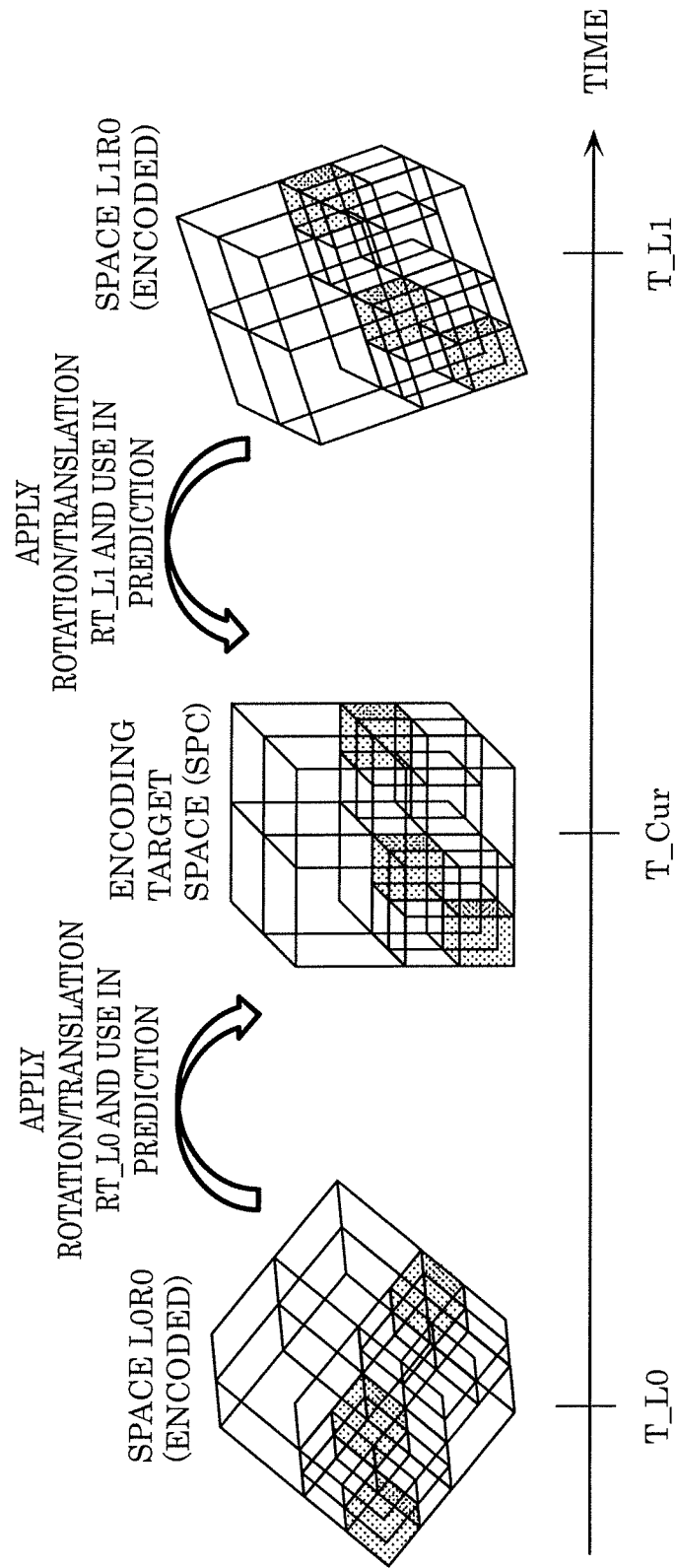


FIG. 46

```
space_header0 {  
    ...  
    for (i=0; i<MaxRefSpc_l0;i++) {  
        RT_flag_l0[i]  
        if (RT_flag_l0[i]) {  
            R_l0[i]  
            T_l0[i]  
        }  
    }  
    ...  
    for (i=0; i<MaxRefSpc_l1;i++) {  
        RT_flag_l1[i]  
        if (RT_flag_l1[i]) {  
            R_l1[i]  
            T_l1[i]  
        }  
    }  
    ...  
}
```

FIG. 47

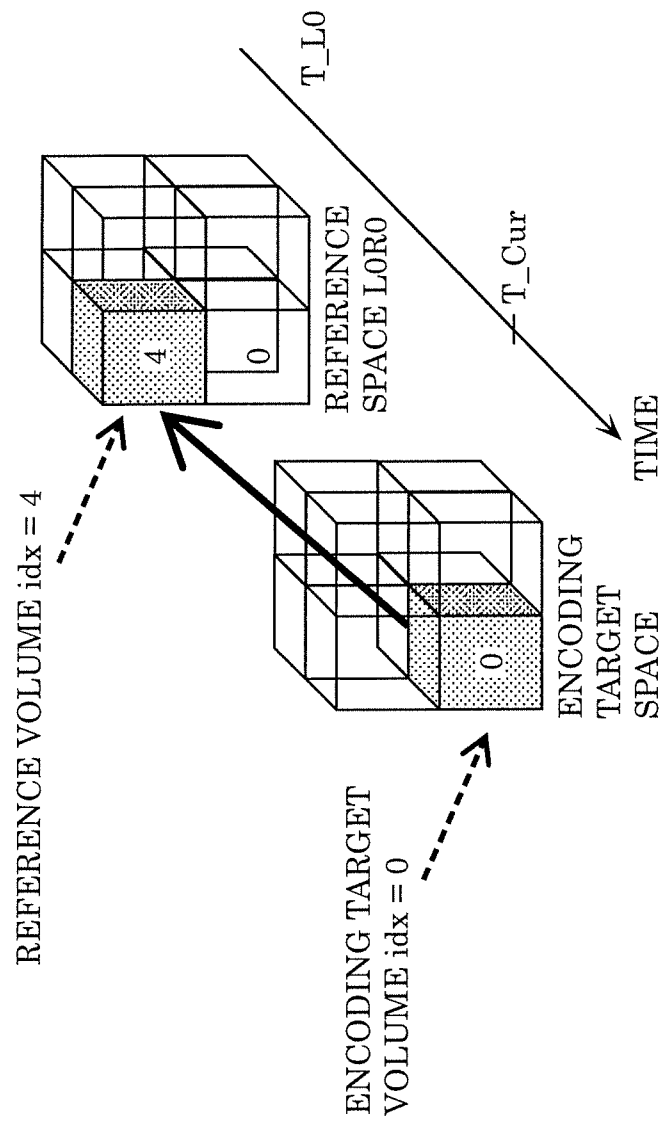


FIG. 48

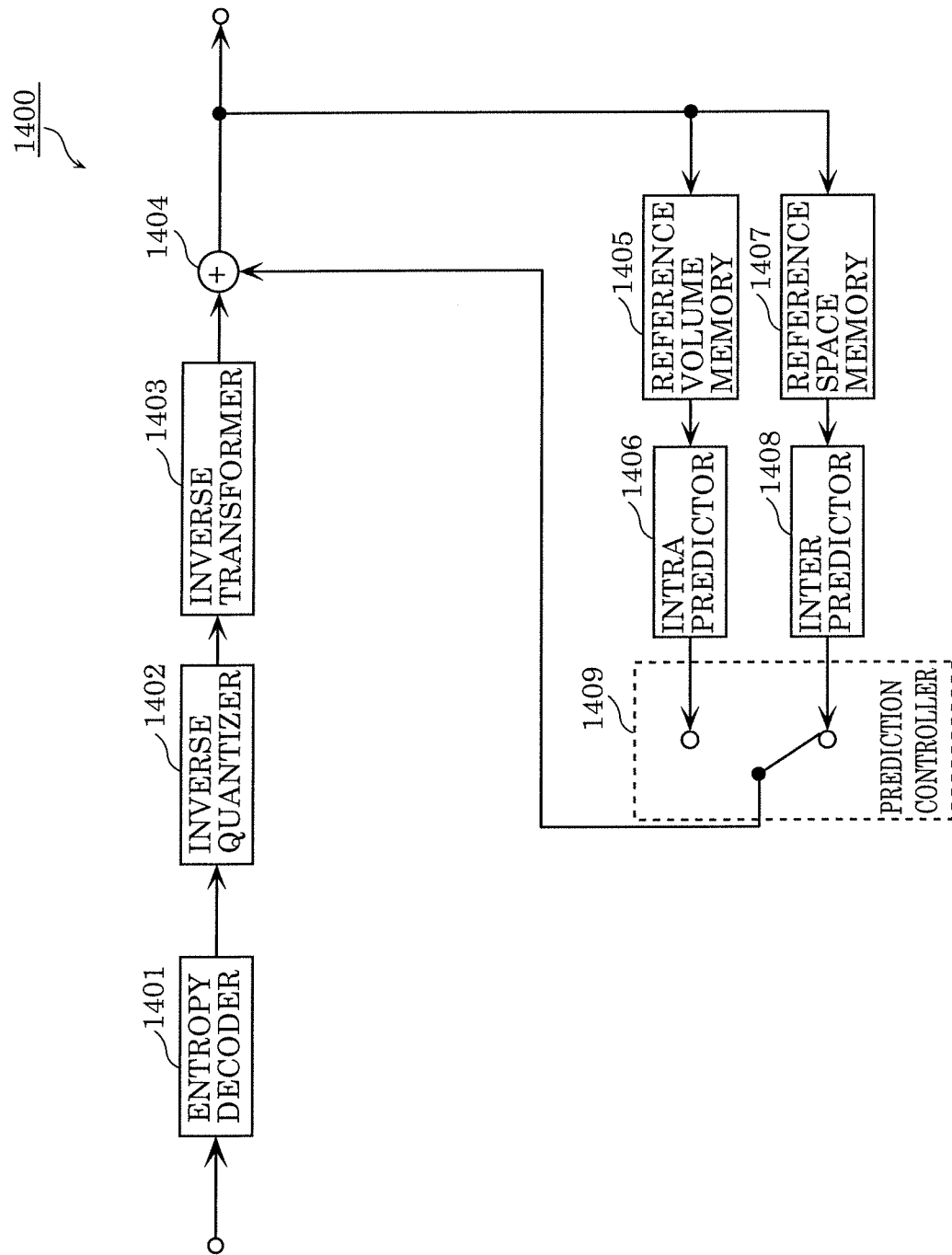


FIG. 49

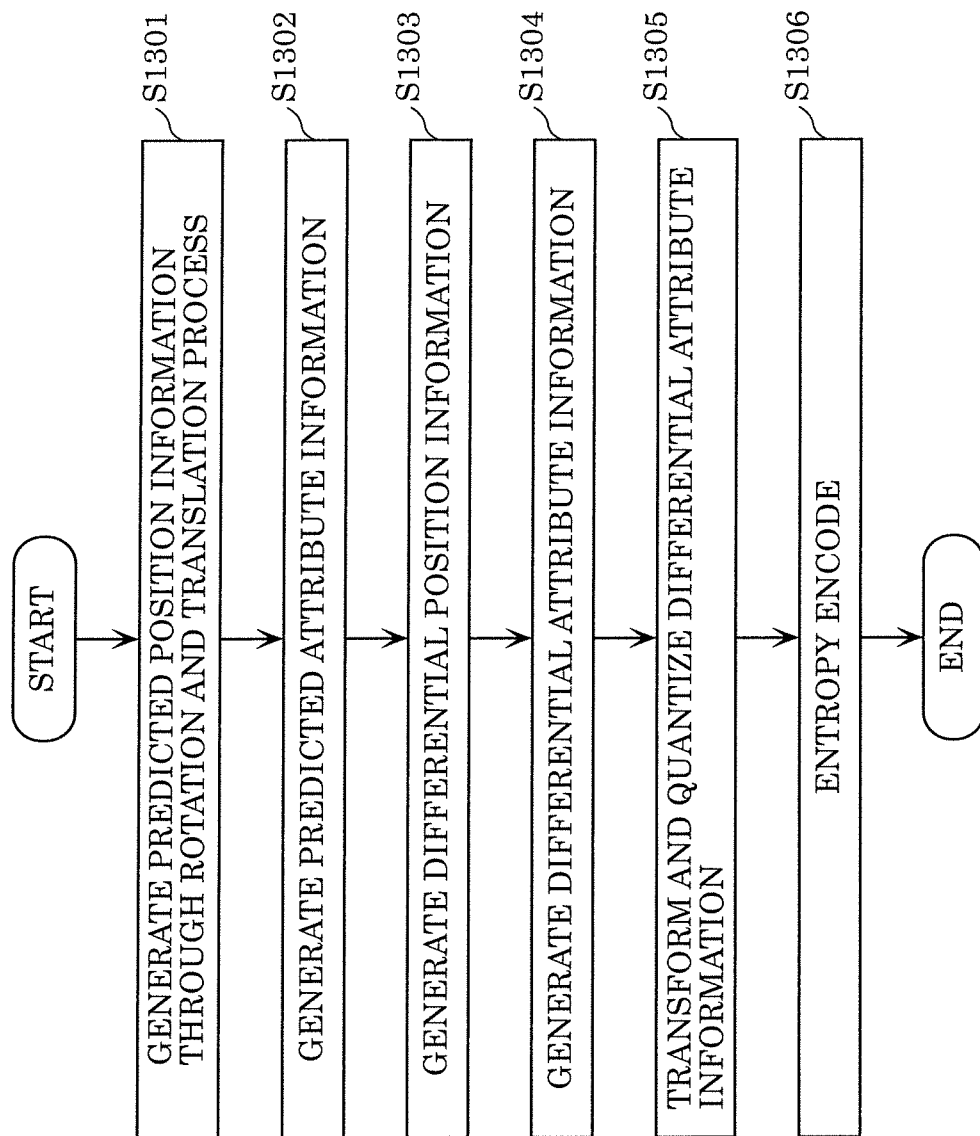


FIG. 50

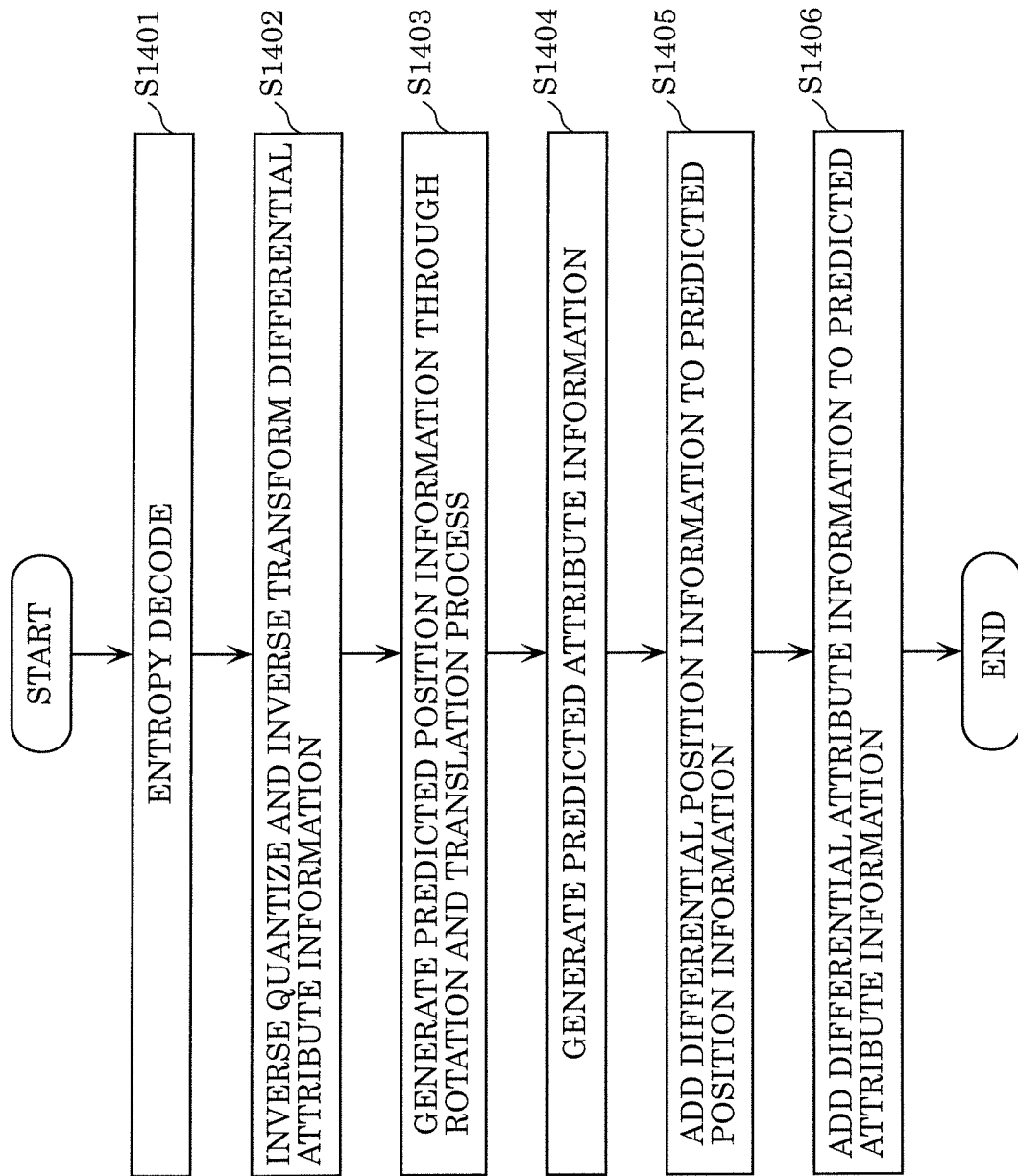


FIG. 51

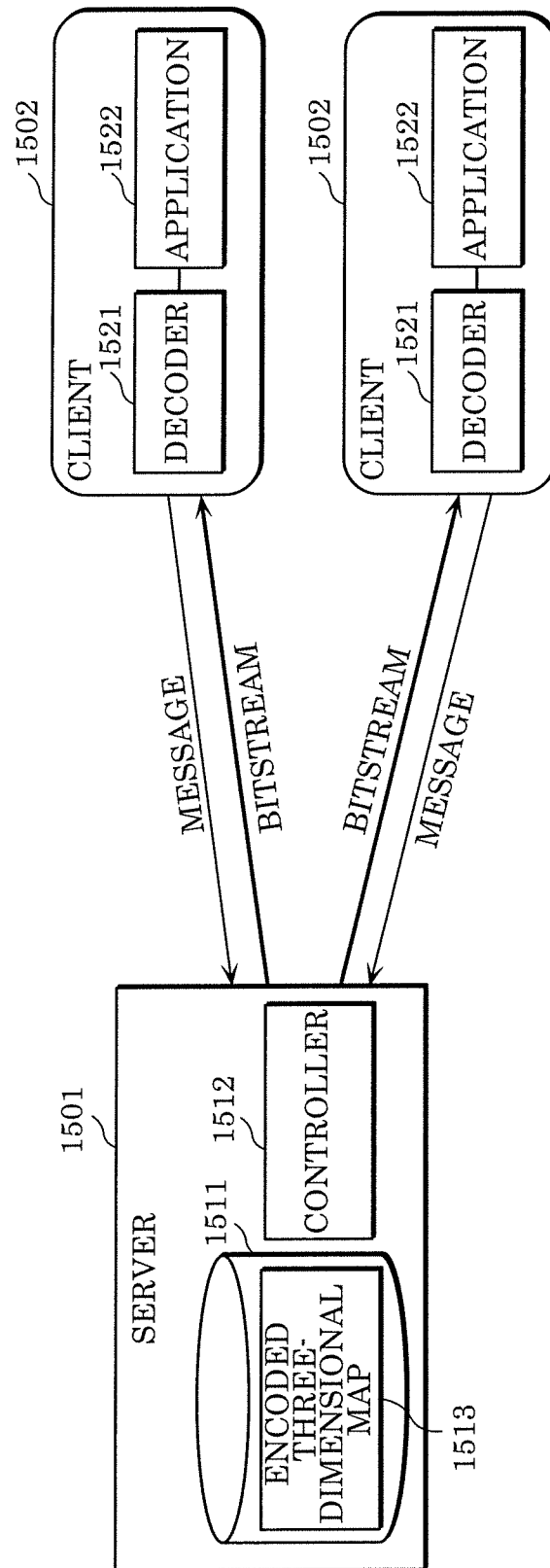


FIG. 52

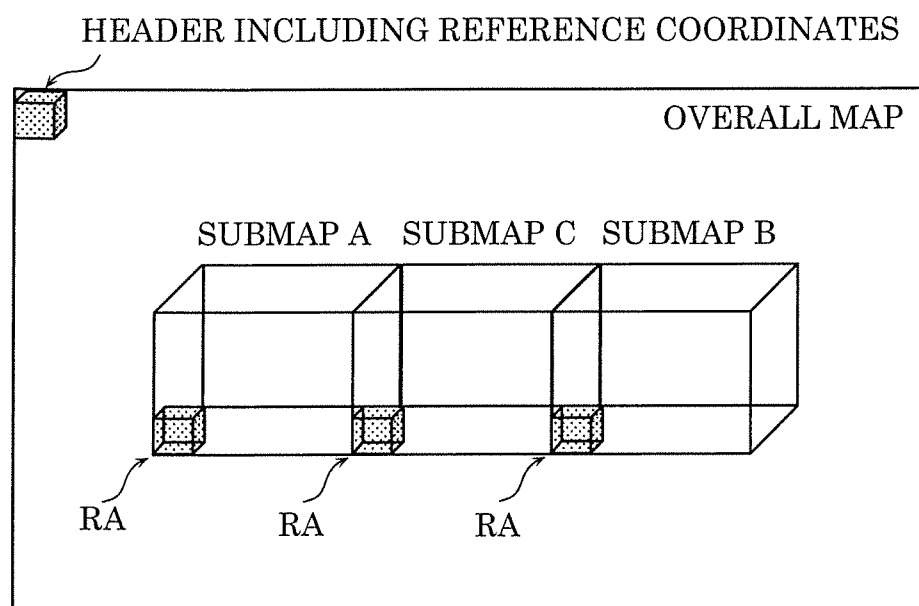


FIG. 53

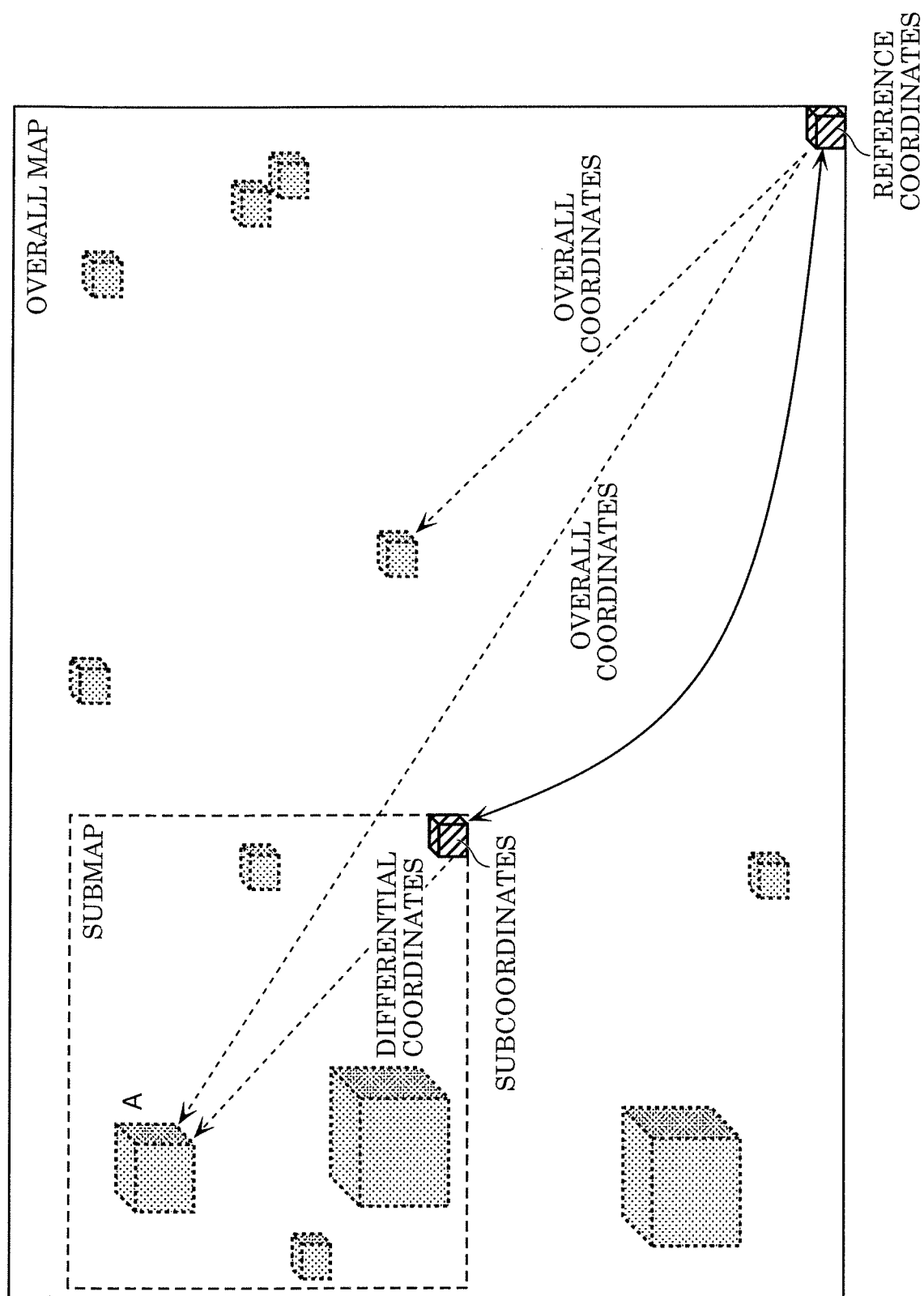


FIG. 54

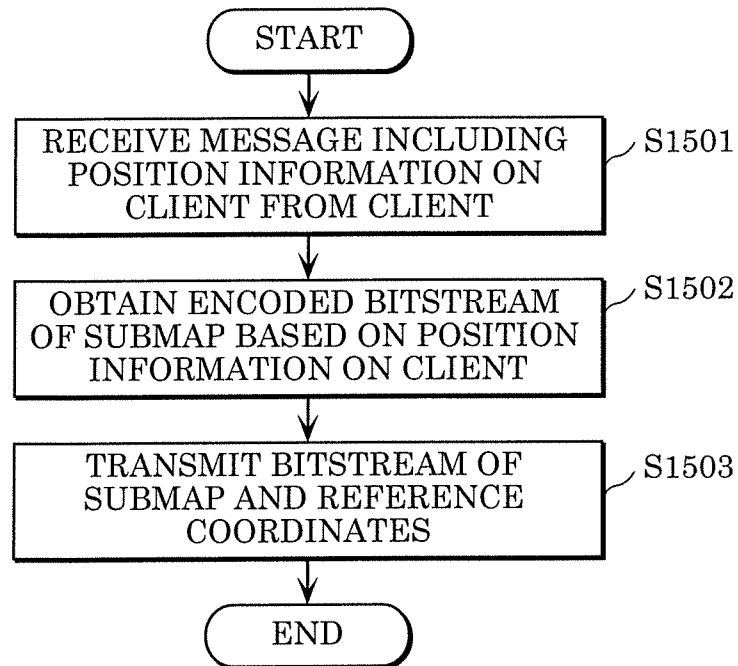


FIG. 55

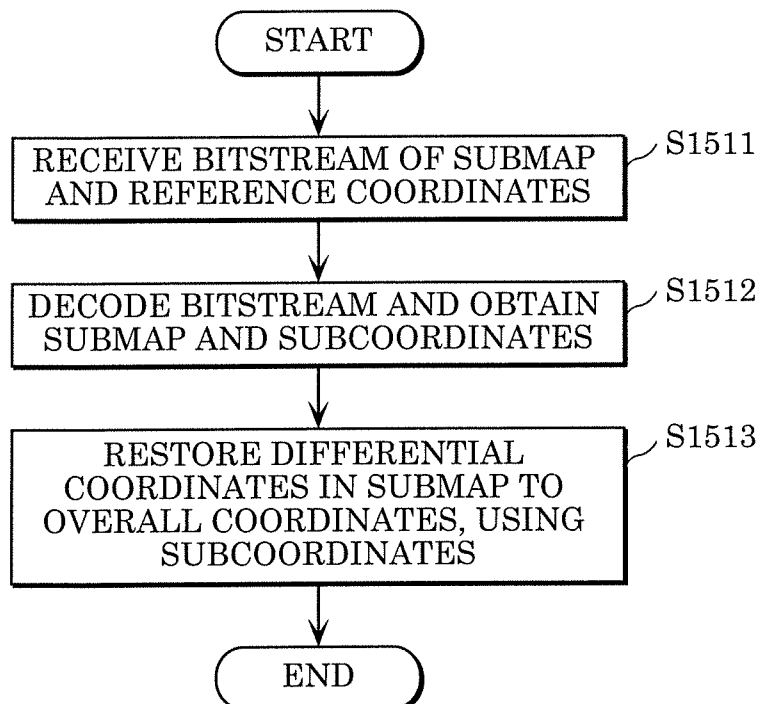


FIG. 56

```
sub-map_header() {  
    . . .  
    NumOfPoint  
    sub_coordinate_x  
    sub_coordinate_y  
    sub_coordinate_z  
    . . .  
    sub_map_data()  
    . . .  
}  
  
sub_map_data() {  
    . . .  
    for (i=0; i<NumOfPoint; i++) {  
        diff_x[i]  
        diff_y[i]  
        diff_z[i]  
    }  
    . . .  
}
```

FIG. 57

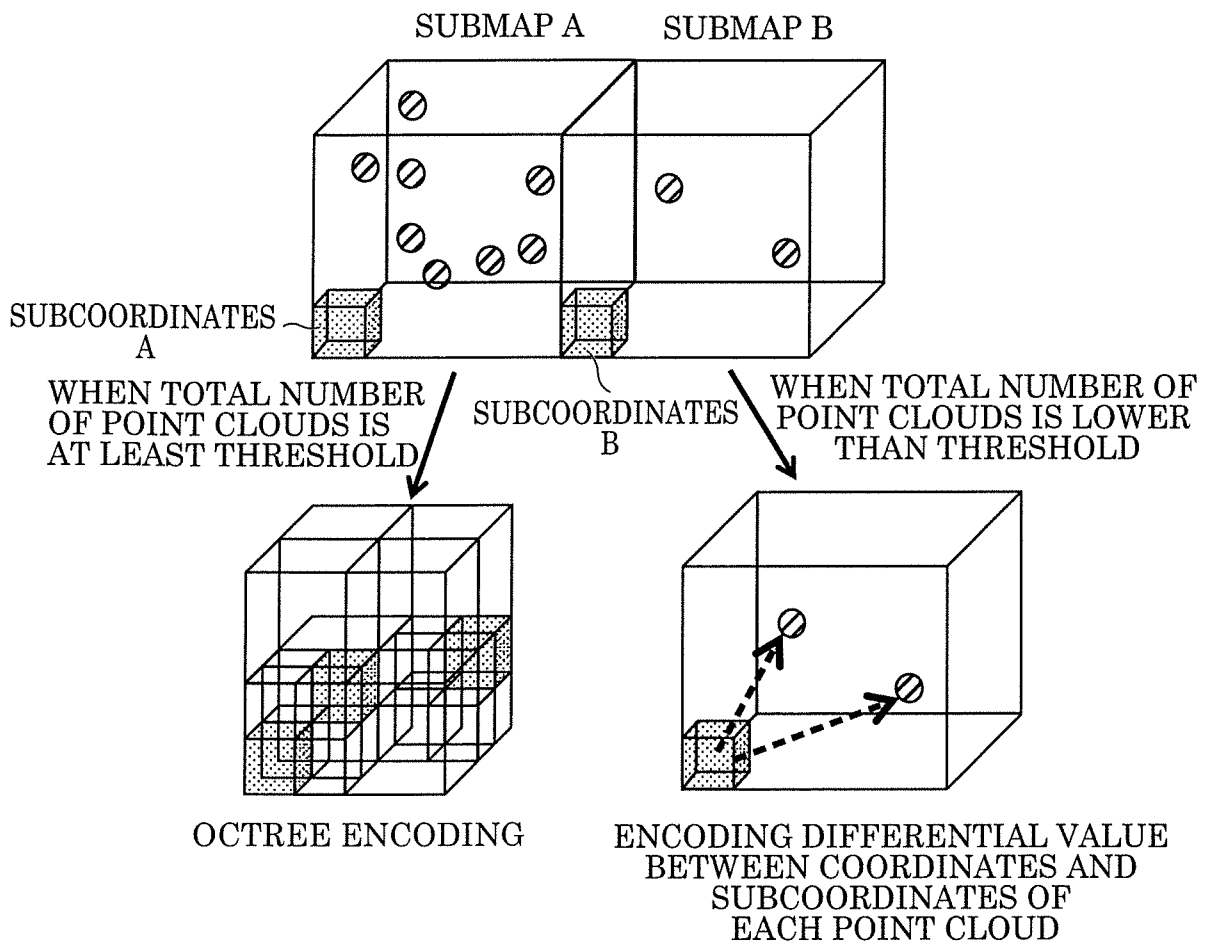


FIG. 58

```
sub_map_header() {  
    . . .  
    coding_type  
    If (coding_type == non_octree)  
    {  
        NumOfPoint  
        sub_coordinate_x  
        sub_coordinate_y  
        sub_coordinate_z  
    }  
    else if (coding_type==octree) {  
    {  
        octree_info()  
    }  
    . . .  
    sub_map_data()  
    . . .  
}  
  
sub_map_data() {  
    . . .  
    if (coding_type==non_octree)  
    {  
        for (i=0; i<NumOfPoint; i++) {  
            diff_x[i]  
            diff_y[i]  
            diff_z[i]  
        }  
    }  
    else (coding_type==octree) {  
        octree_data()  
    }  
    . . .  
}
```

FIG. 59

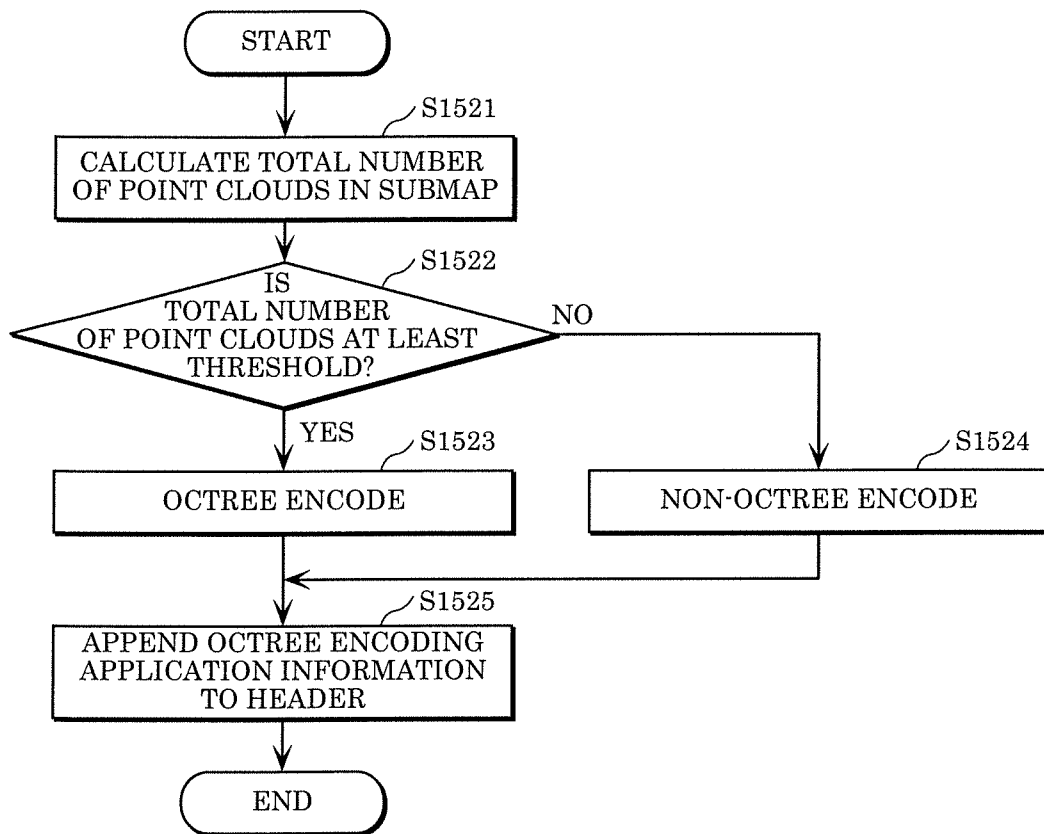


FIG. 60

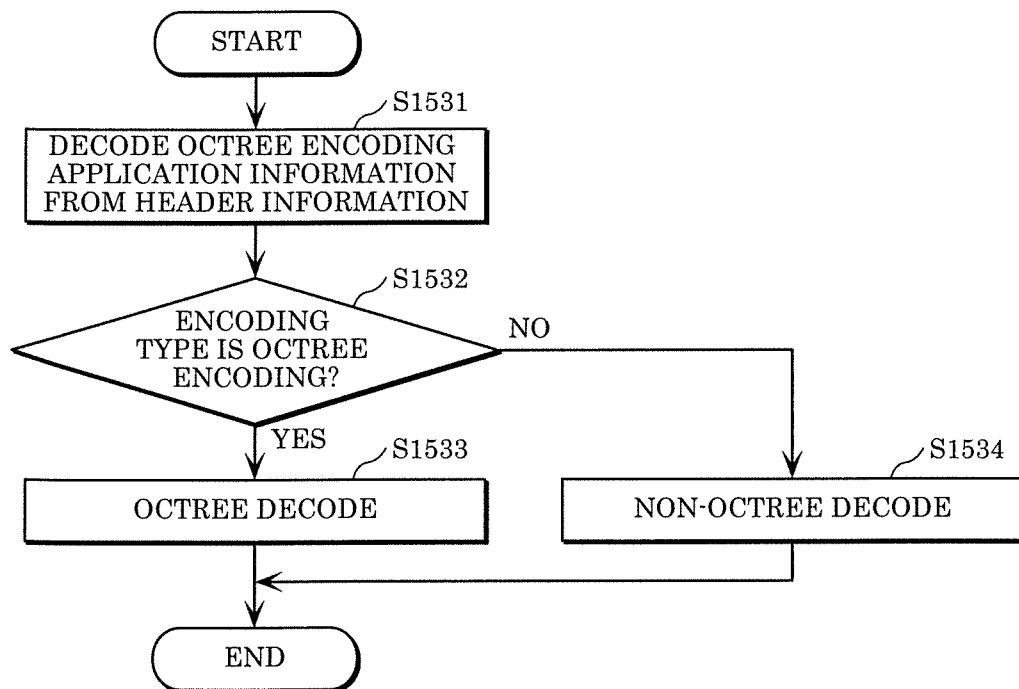


FIG. 61

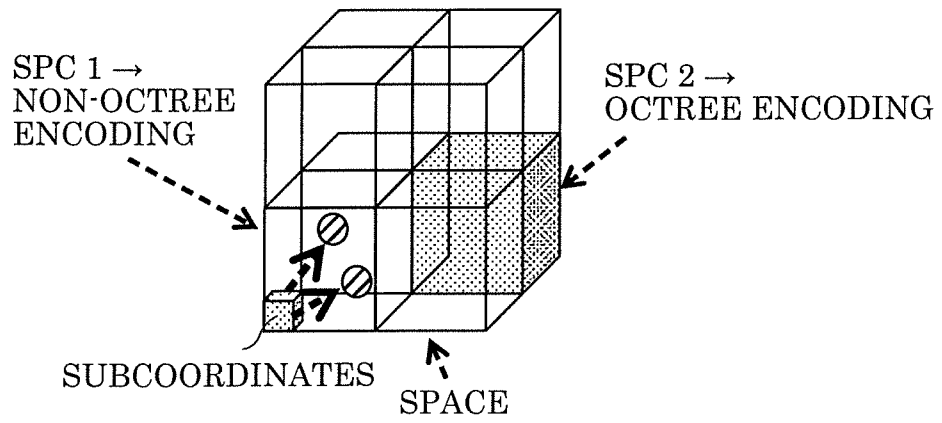


FIG. 62

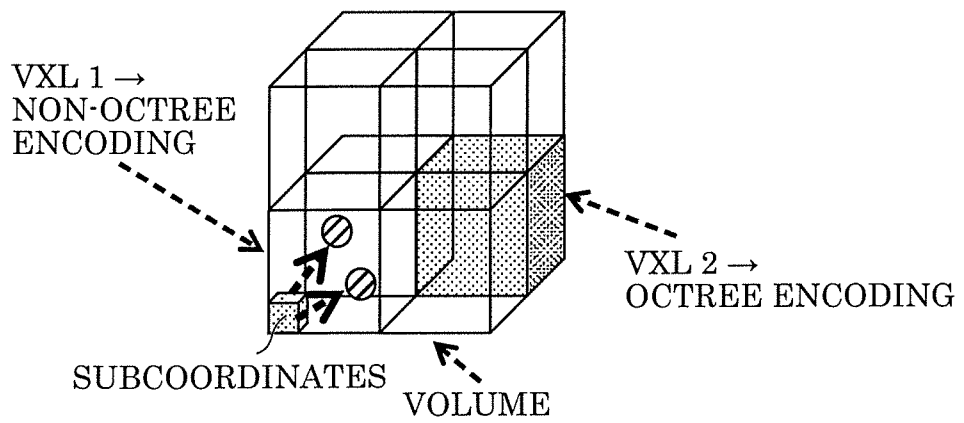


FIG. 63

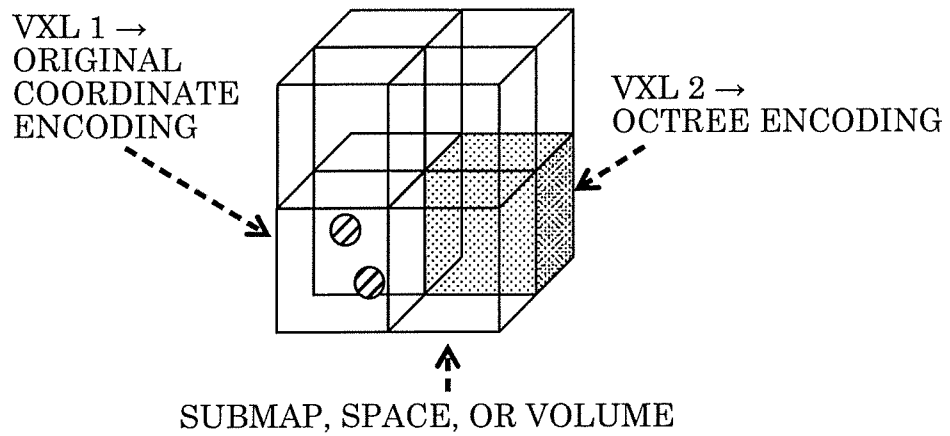


FIG. 64

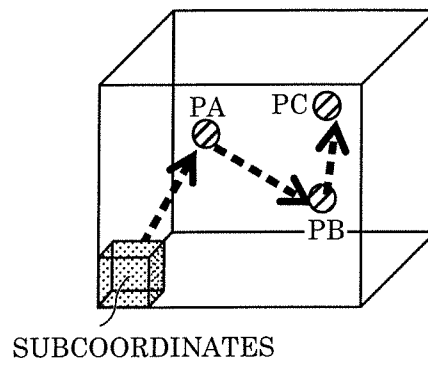


FIG. 65

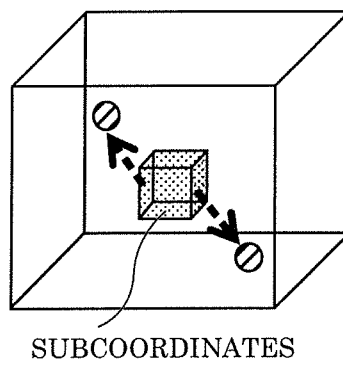


FIG. 66

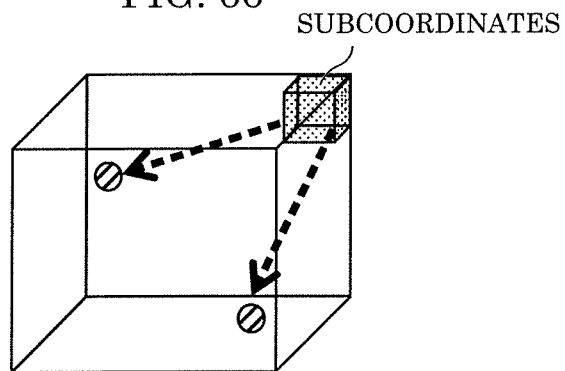


FIG. 67

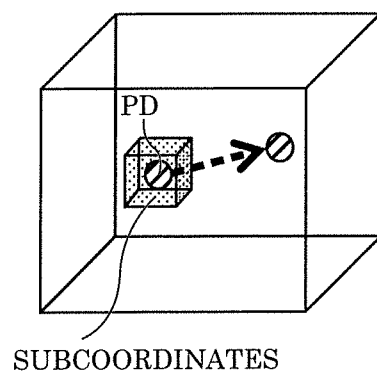


FIG. 68

```
volume_header() {  
    . . .  
    coding_type  
    If (coding_type == non_octree)  
    {  
        NumOfPoint  
        sub_coordinate_x  
        sub_coordinate_y  
        sub_coordinate_z  
    }  
    else if (coding_type==octree) {  
    {  
        octree_info()  
    }  
    . . .  
    volume_data()  
    . . .  
}  
  
volume_data() {  
    . . .  
    if (coding_type==non_octree)  
    {  
        for (i=0; i<NumOfPoint; i++) {  
            diff_x[i]  
            diff_y[i]  
            diff_z[i]  
        }  
    }  
    else (coding_type==octree) {  
        octree_data()  
    }  
    . . .  
}
```

FIG. 69

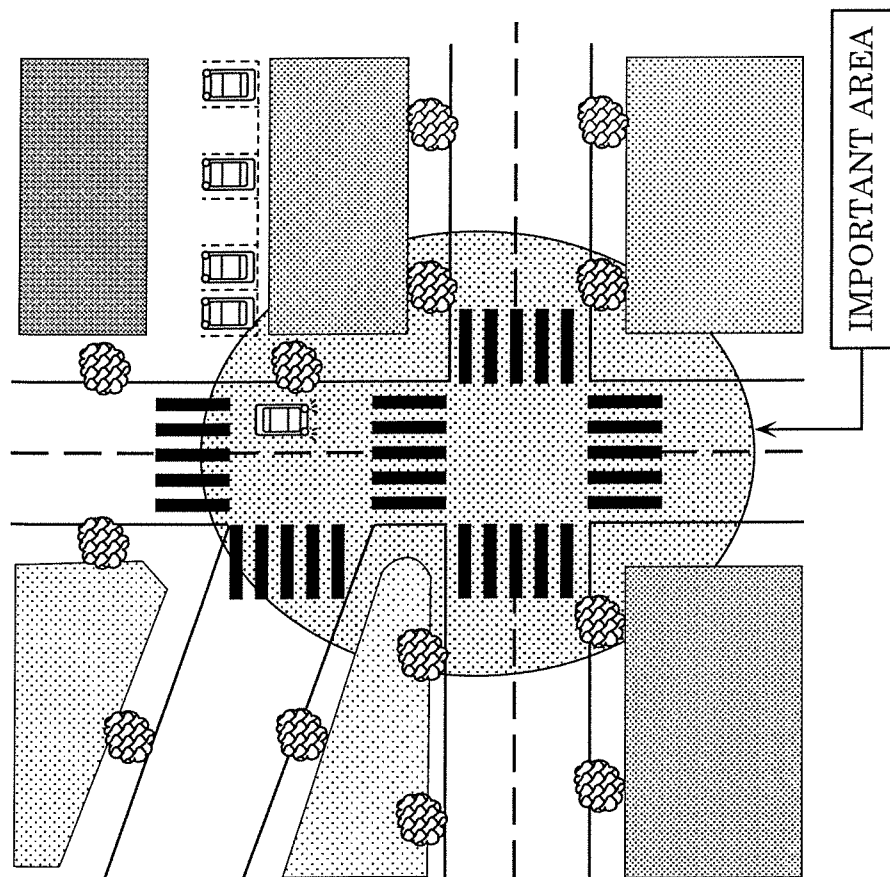


FIG. 70

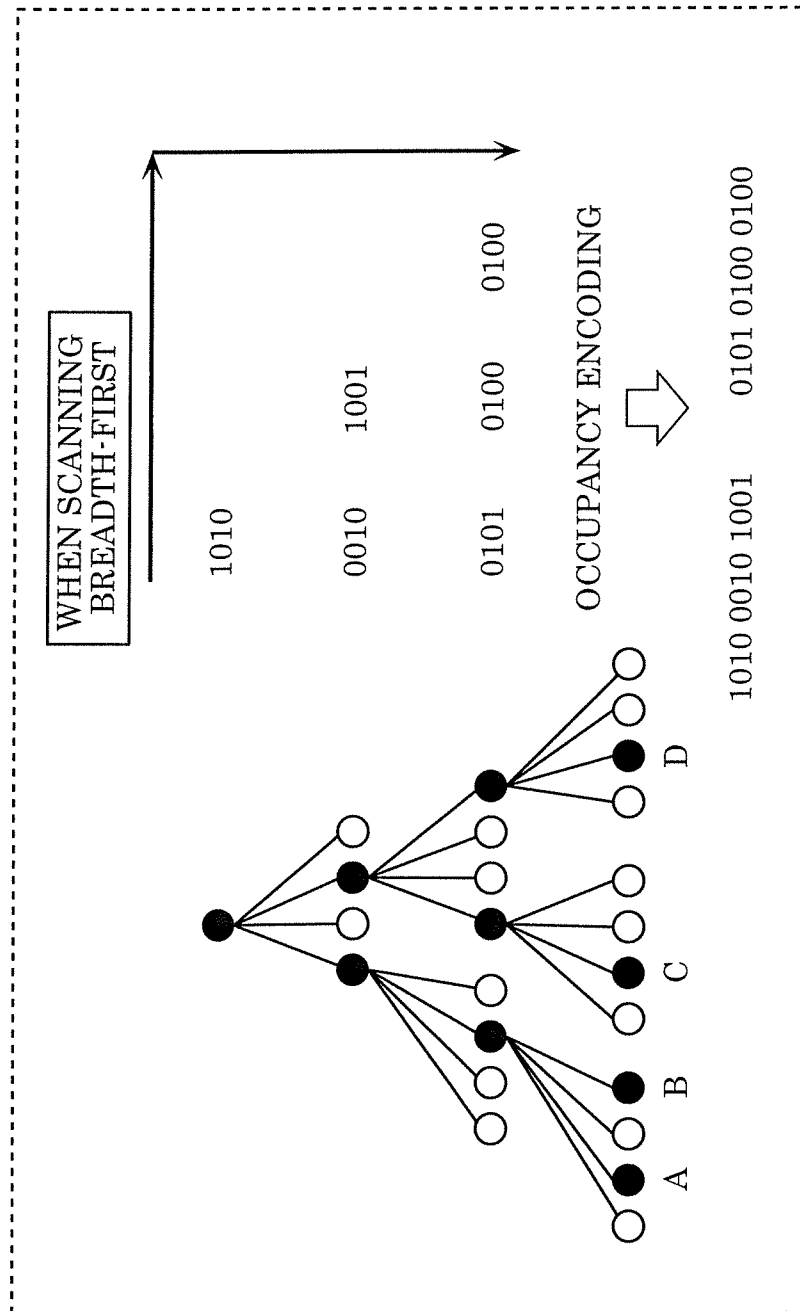


FIG. 71

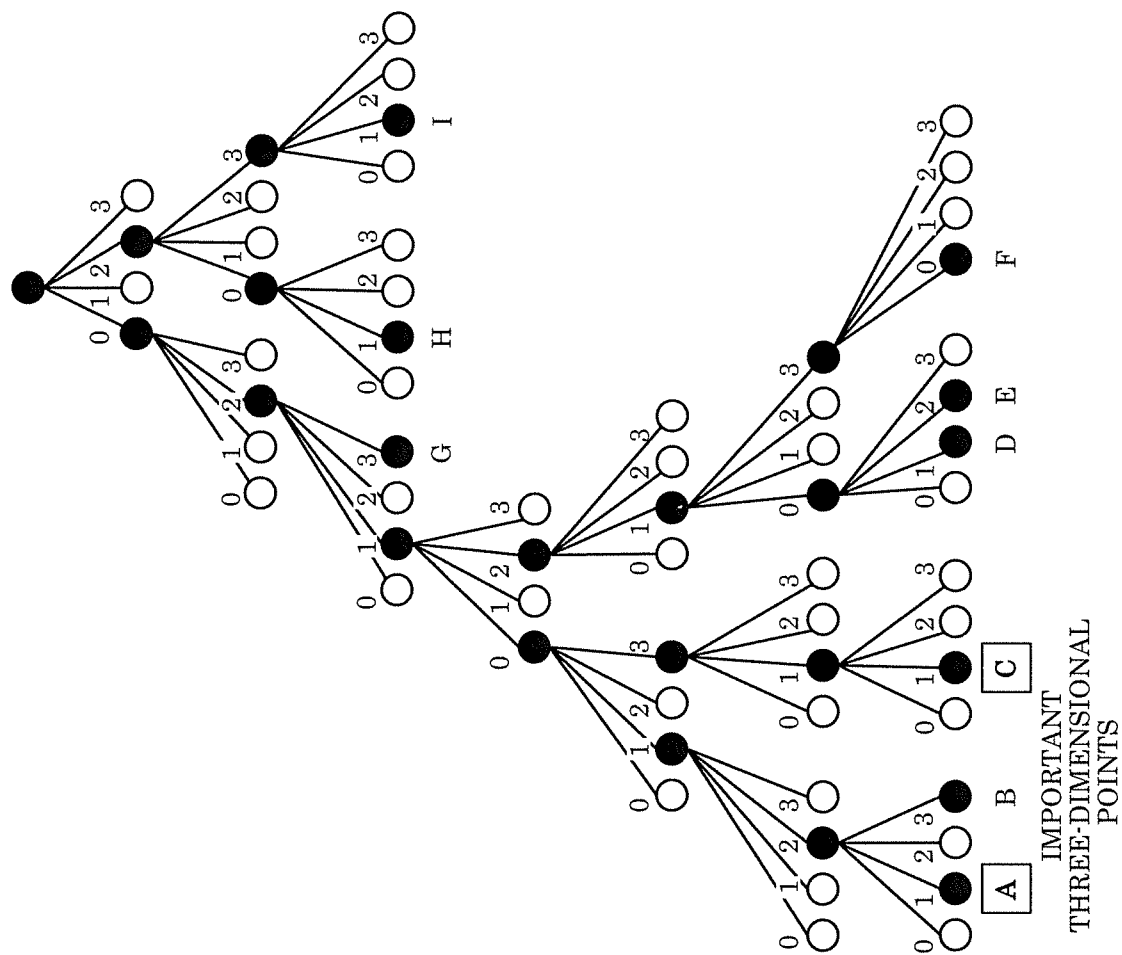


FIG. 73



FIG. 74

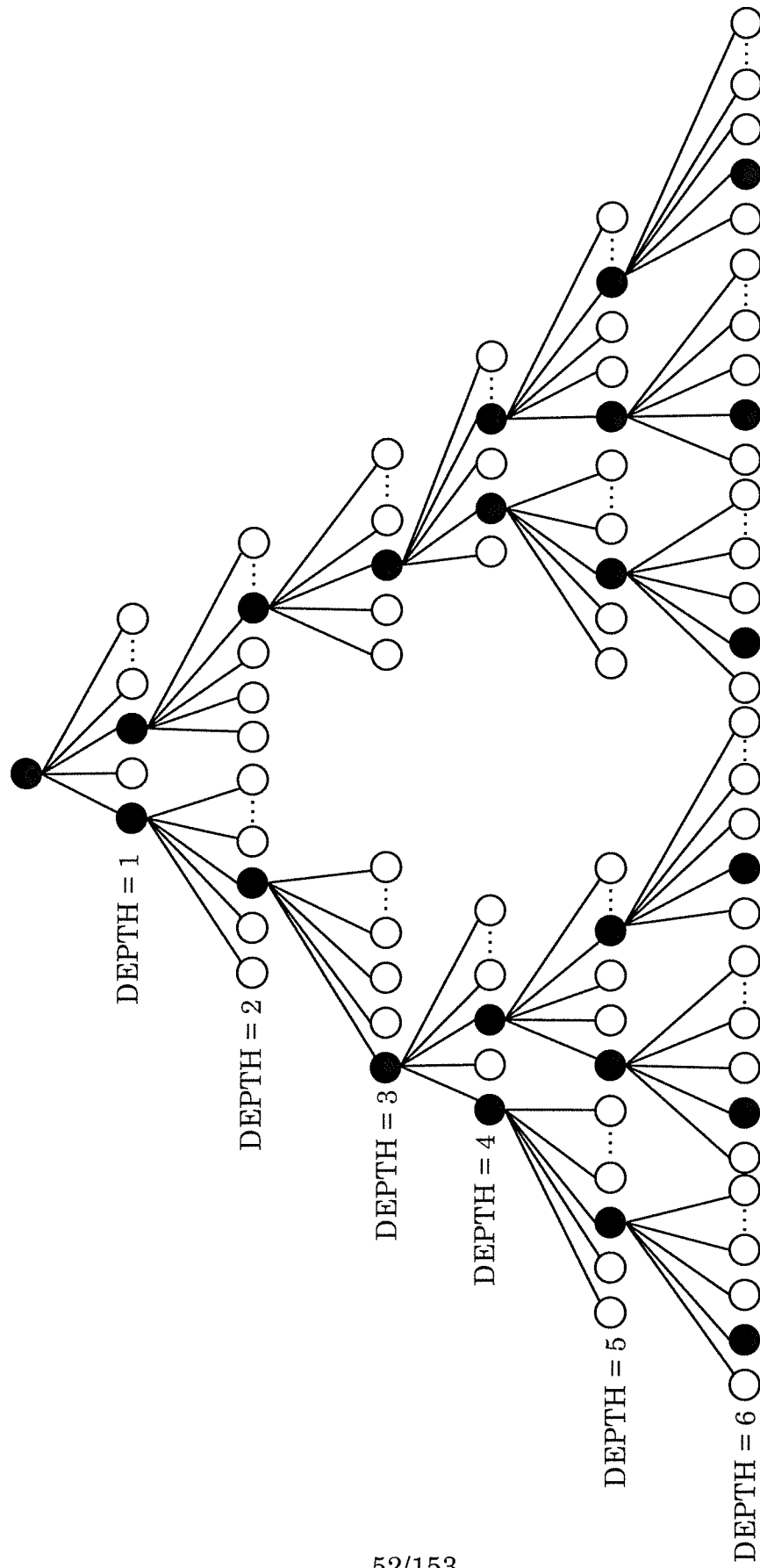


FIG. 75

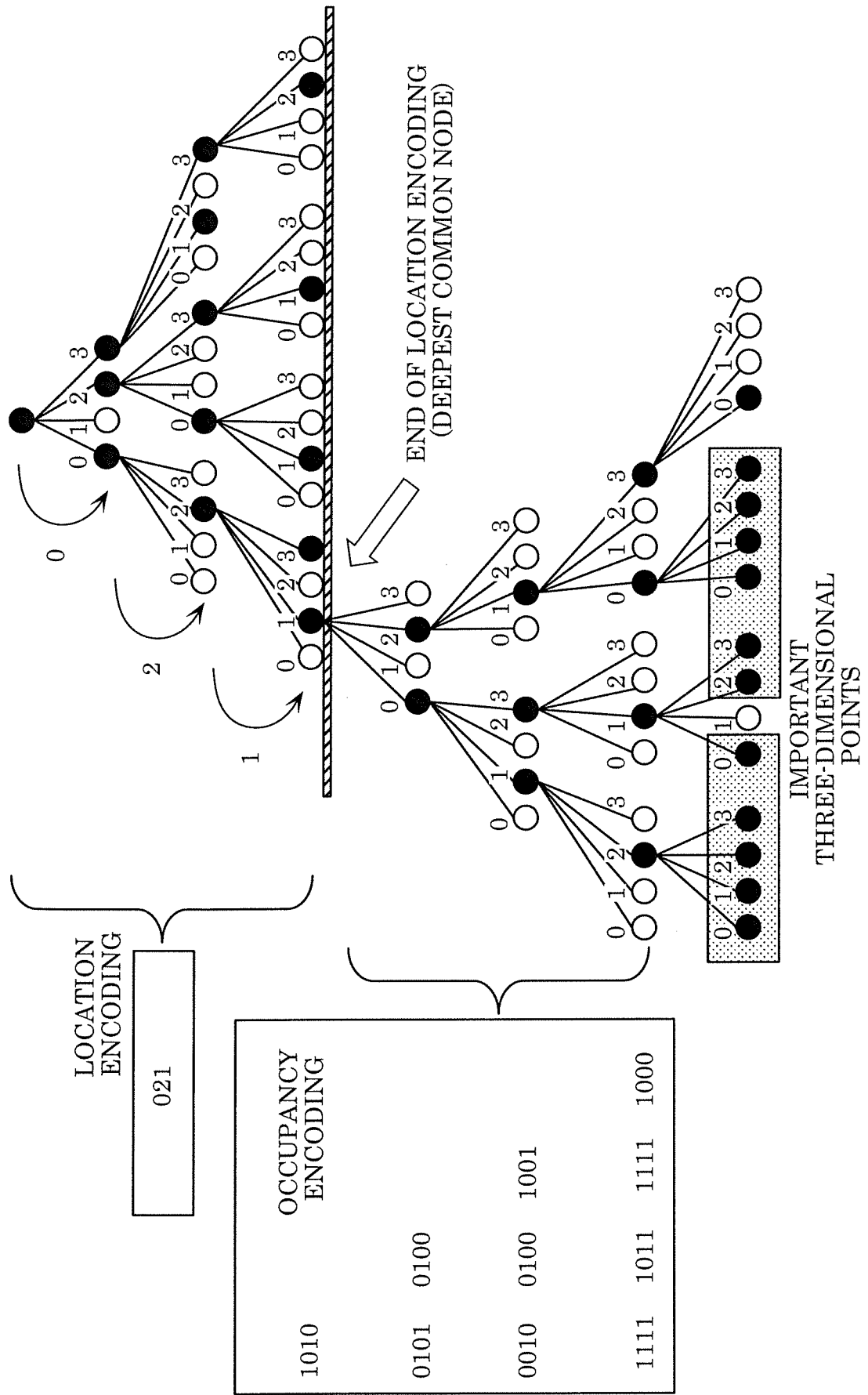


FIG. 76

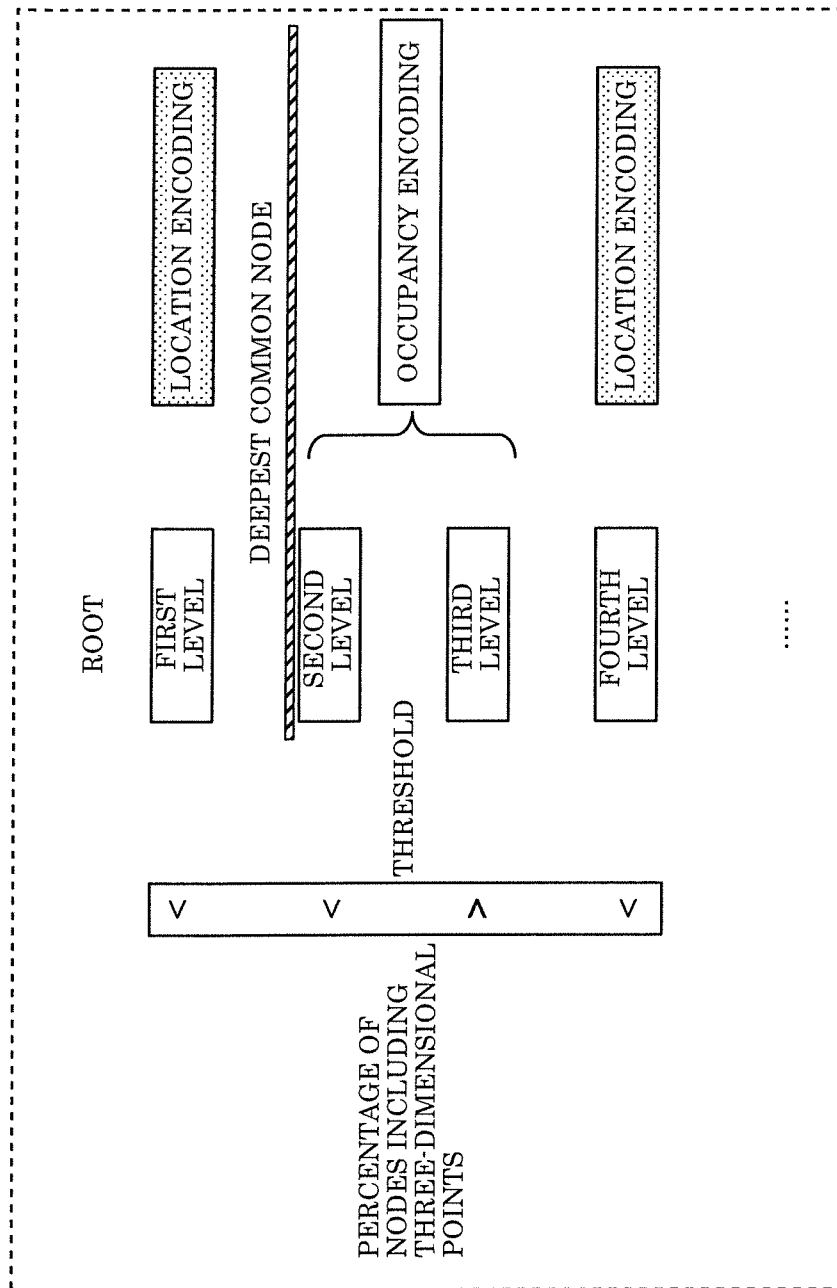


FIG. 77

HEADER	0101	0121	0123	0212	0221	0223	0233	1121	1201	1221	2101	2121	2201	2222
LOCATION CODE														

FIG. 78

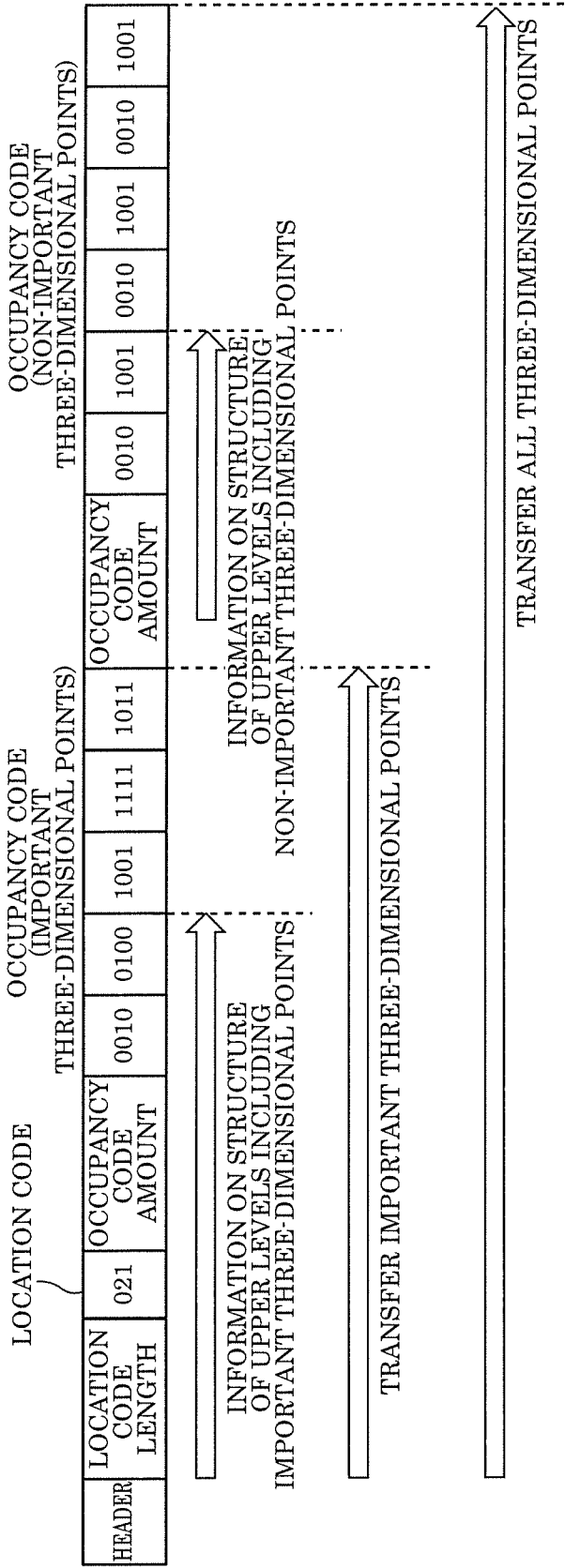


FIG. 79

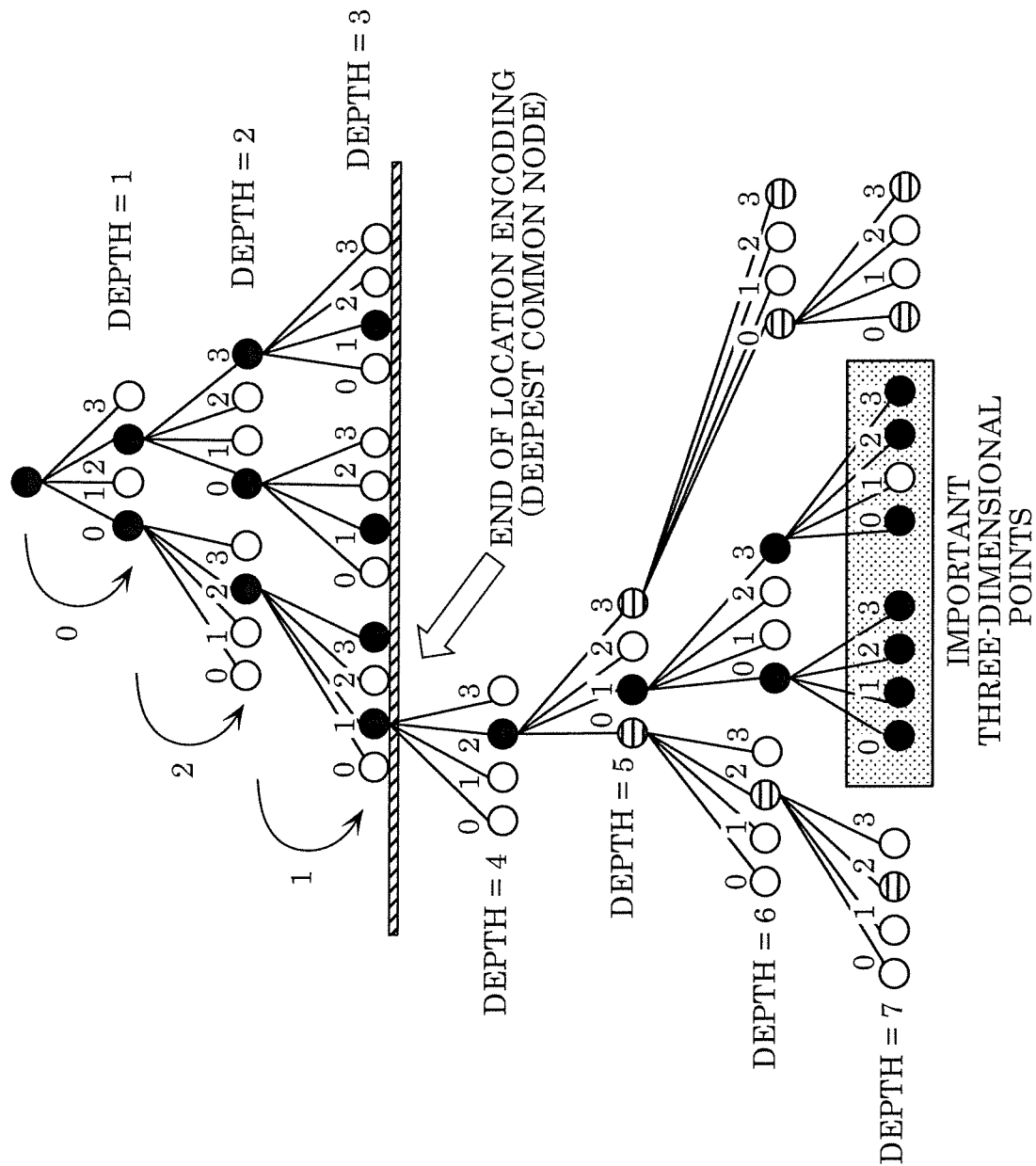


FIG. 80

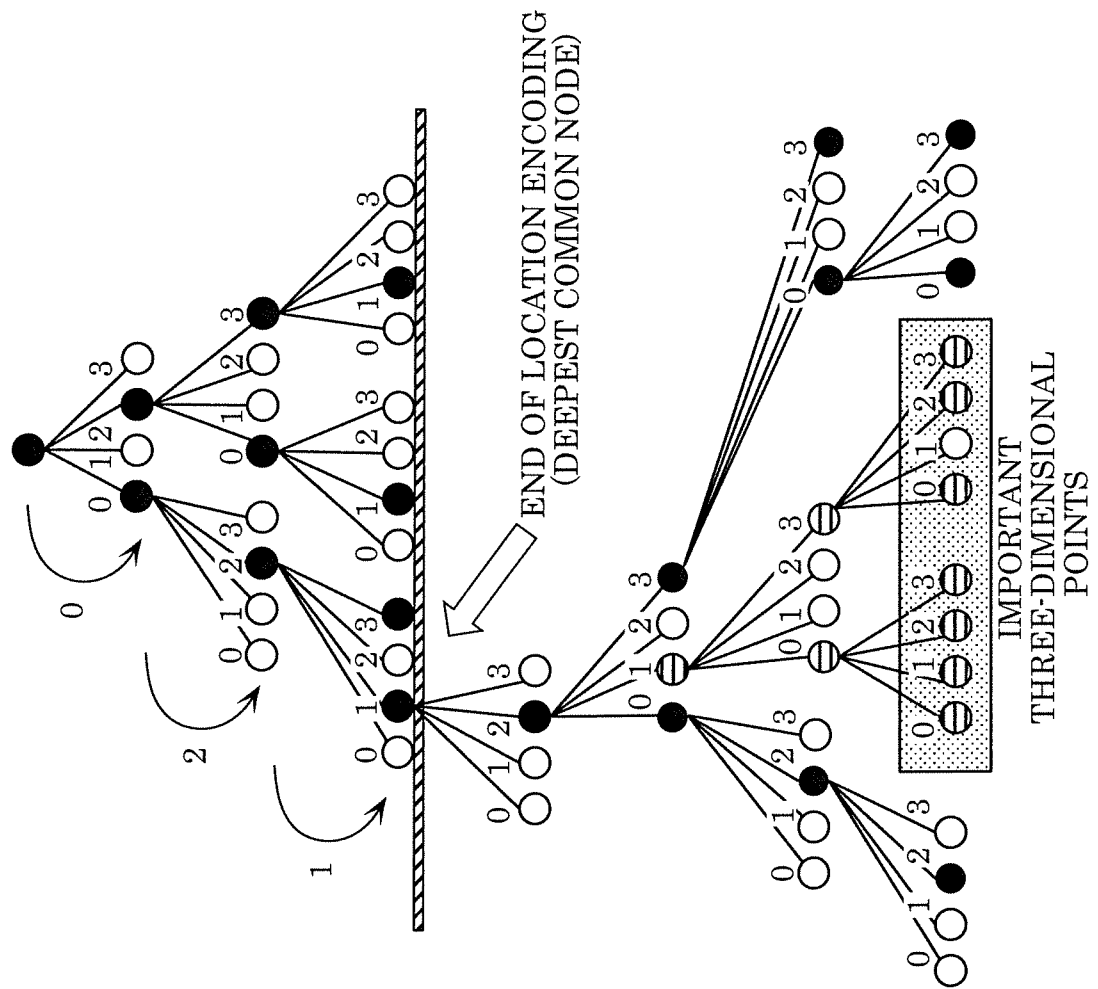


FIG. 81

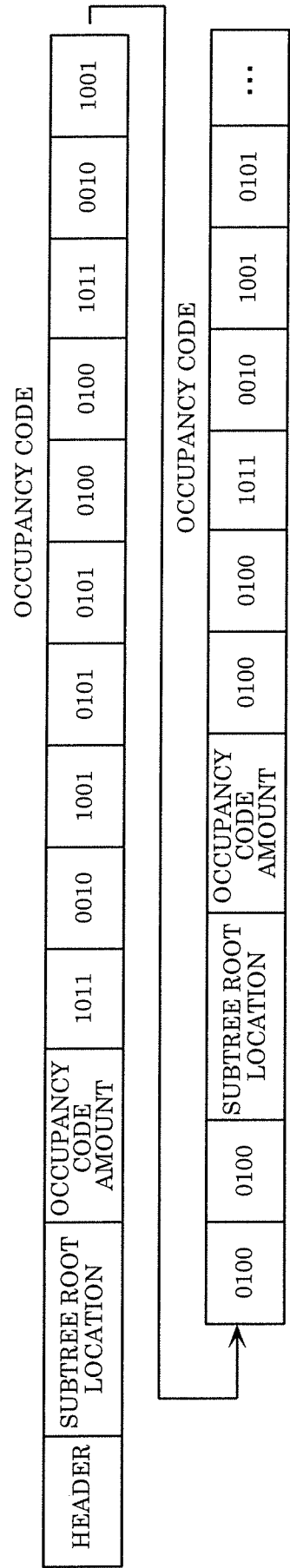


FIG. 82

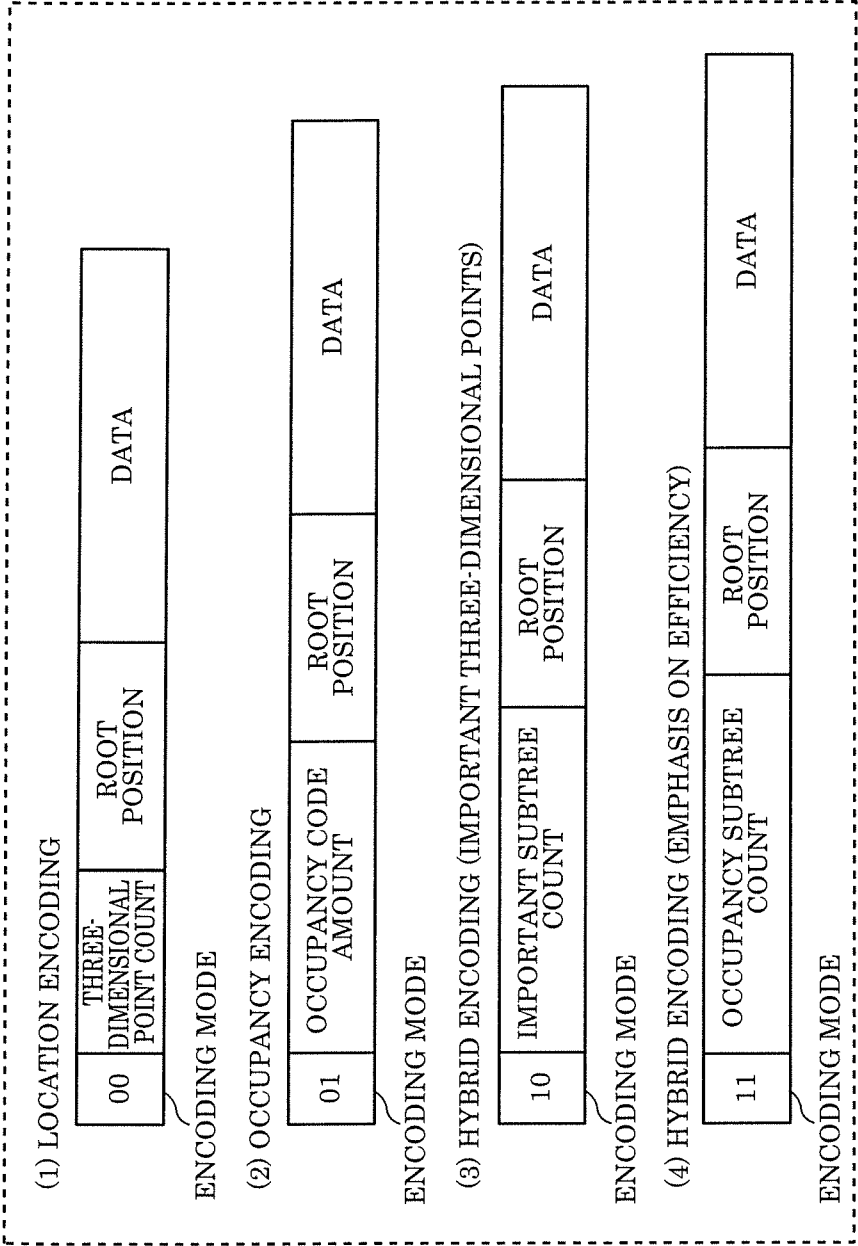


FIG. 83

```

node(depth, index) {
    . . .
    isleaf
    if (isleaf) { //leaf
        point_flag
    }
    else { //node
        coding_type
        if (coding_type == location coding)
        {
            numPoint
            for (i=0; i<numPoint; i++) {
                num_idx[i]
                for (j=0; j<num_idx[i]; j++) {
                    idx[i][j]
                }
            }
        }
        else if (coding_type==occupancy coding) {
            occupancy_code
            for (i=0; i<8; i++) {
                if (occupancy_code&(0x01<i) == 1) {
                    node(depth+1, i)
                }
            }
            . . .
        }
    }
}

```

FIG. 84

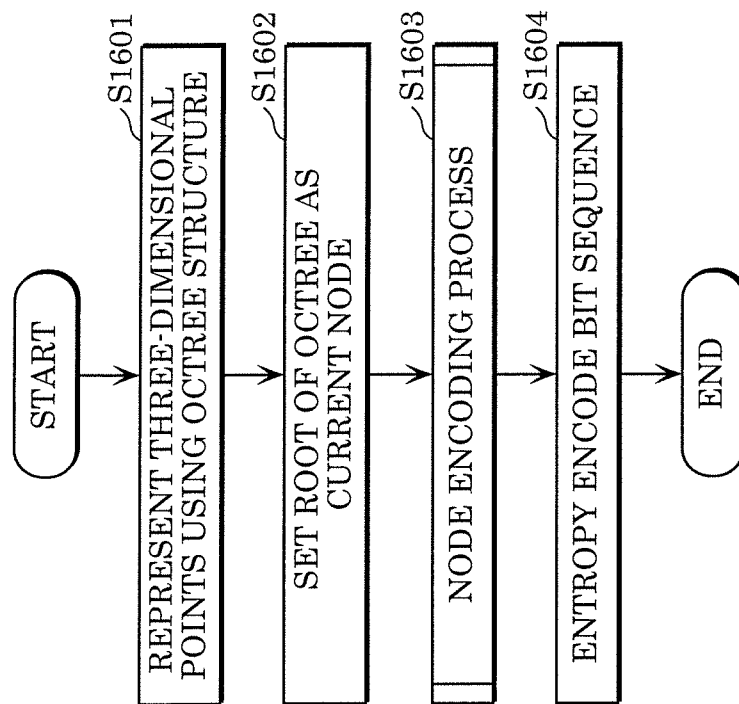


FIG. 85

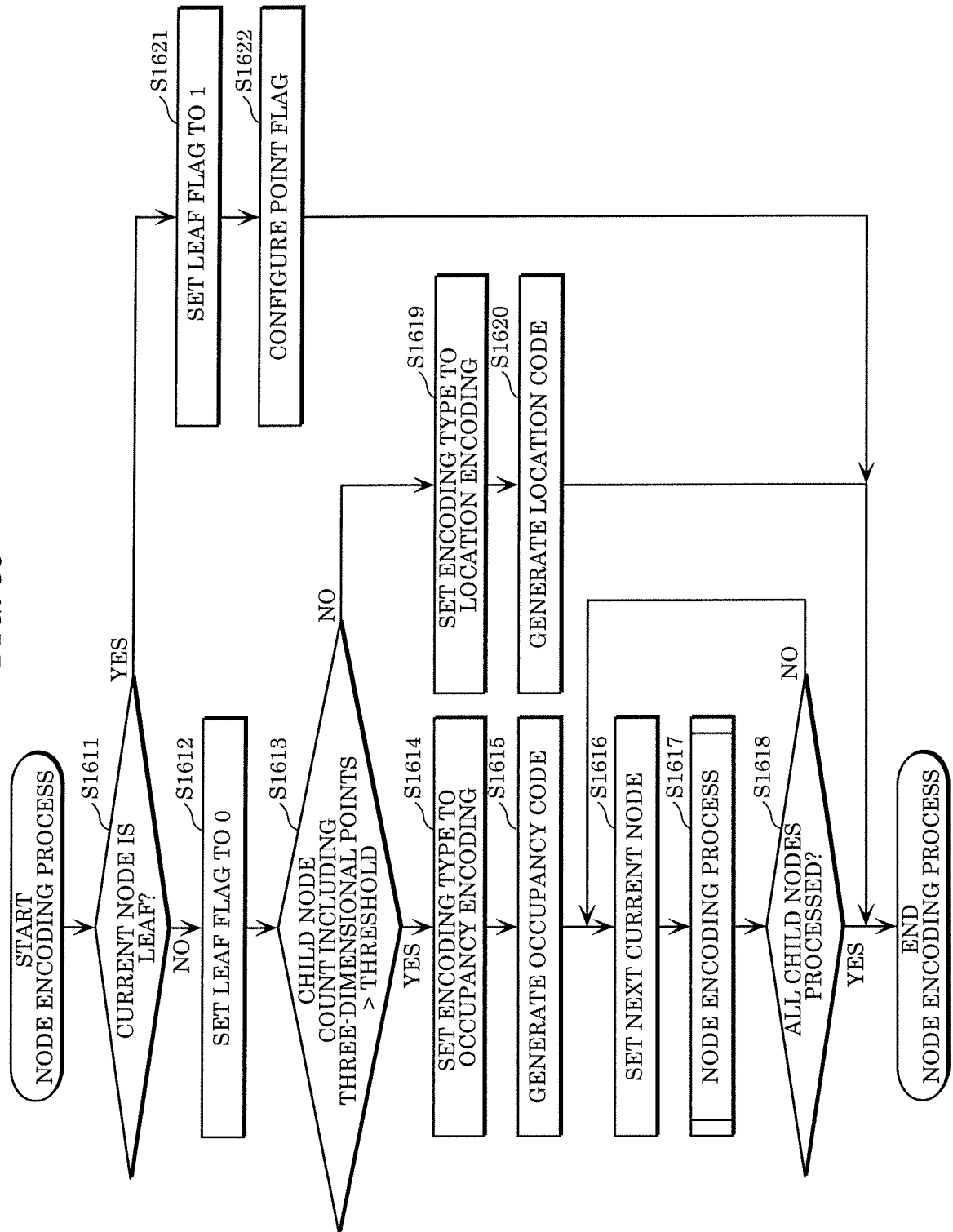


FIG. 86

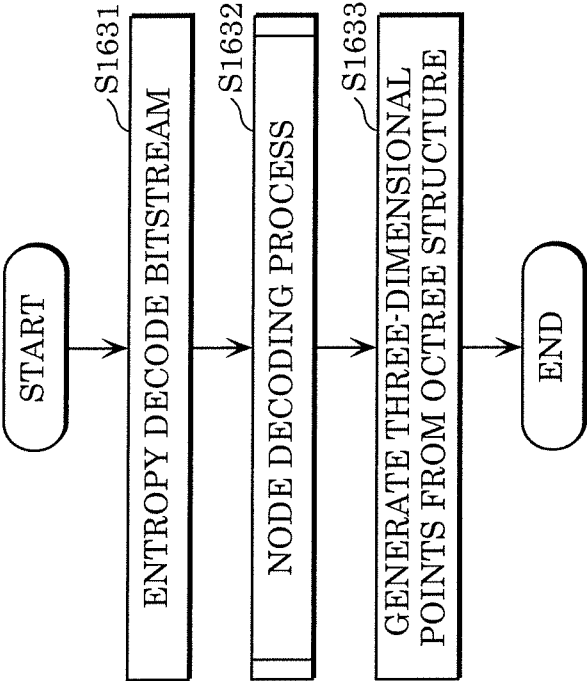


FIG. 87

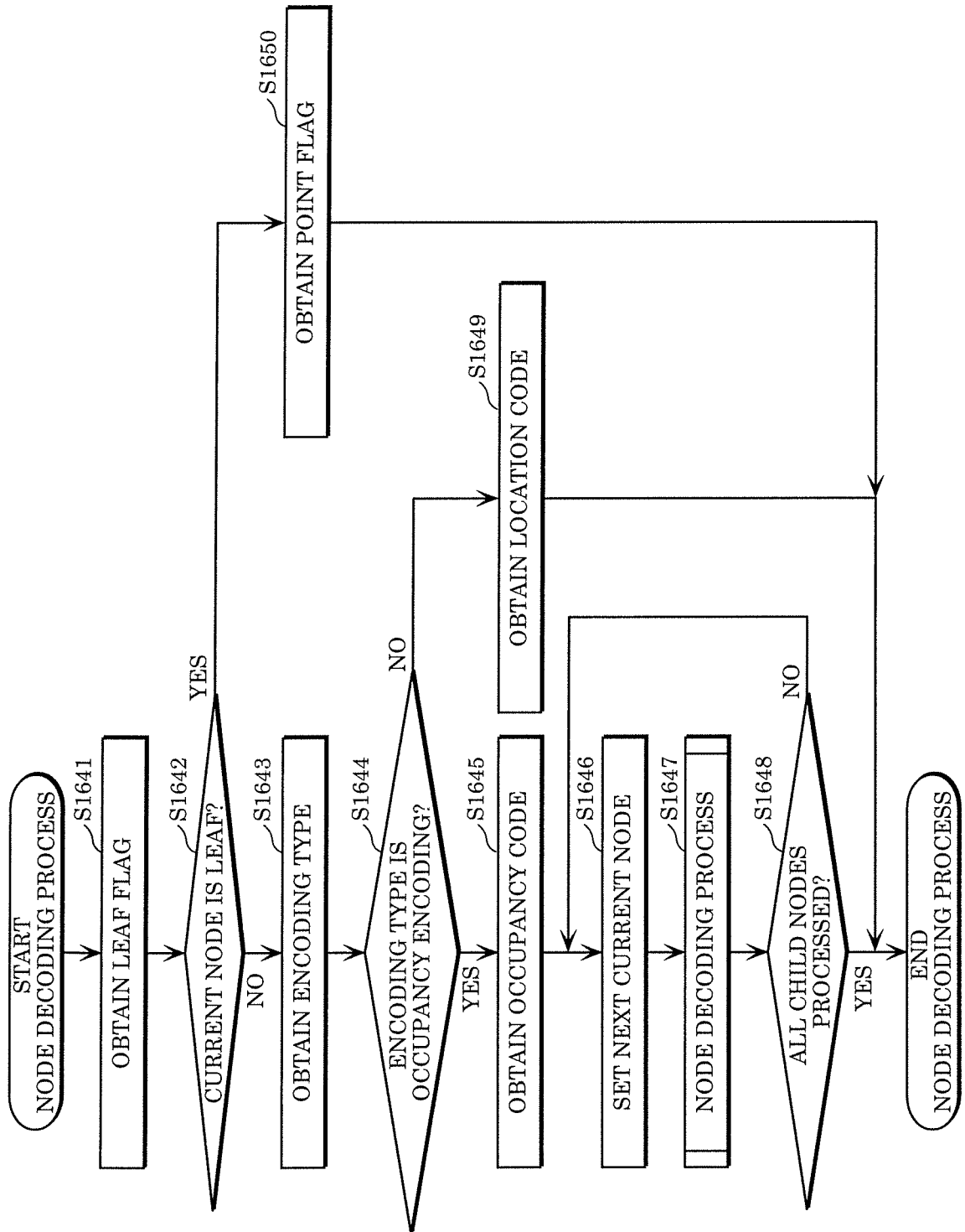


FIG. 88

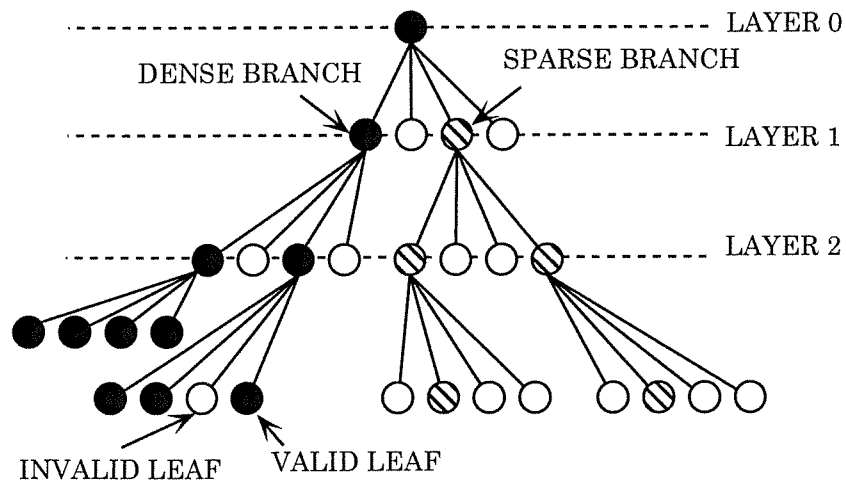


FIG. 89

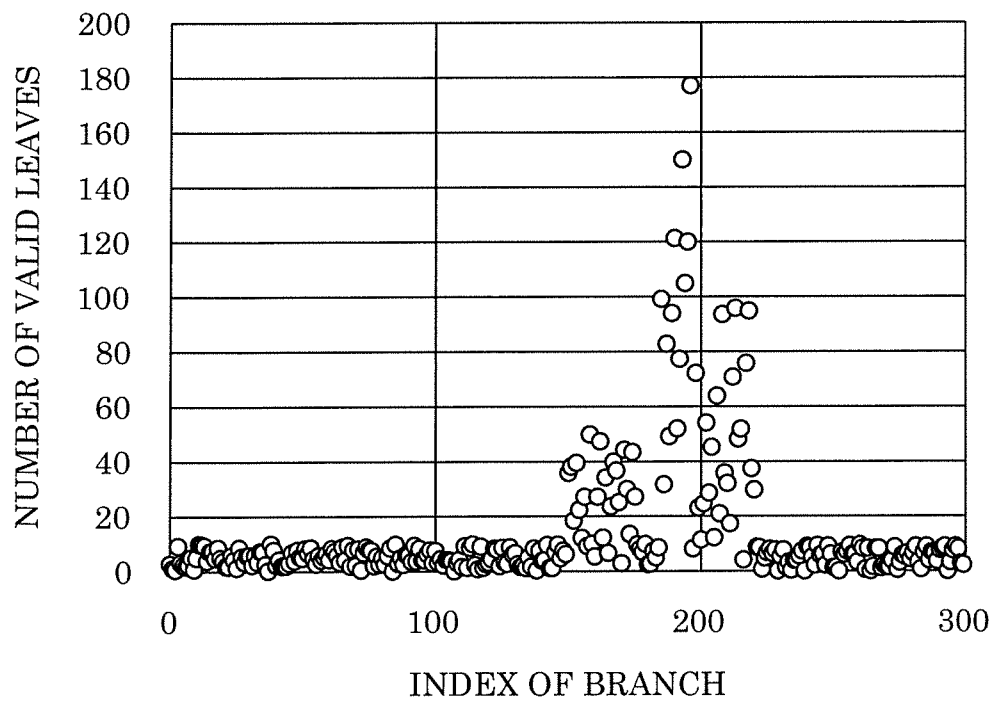


FIG. 90

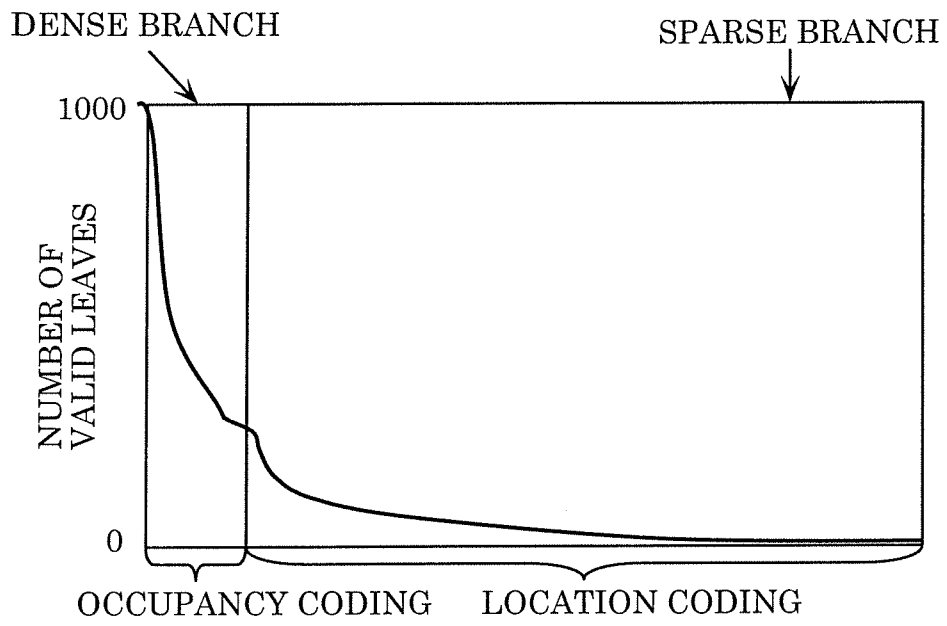


FIG. 91

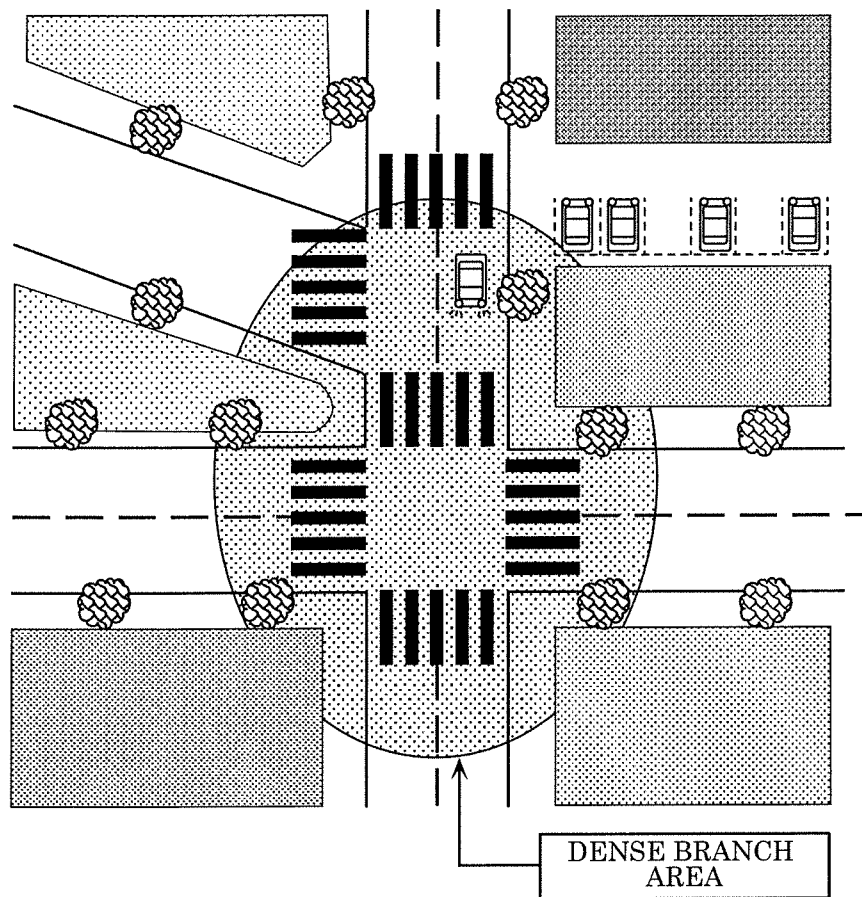


FIG. 92

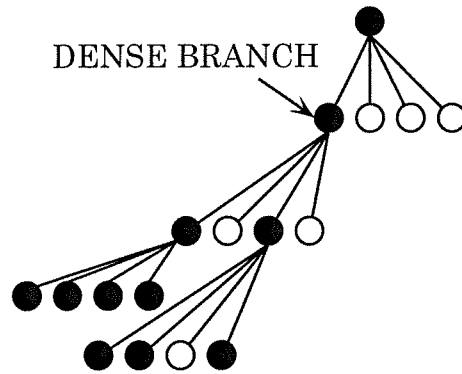


FIG. 93

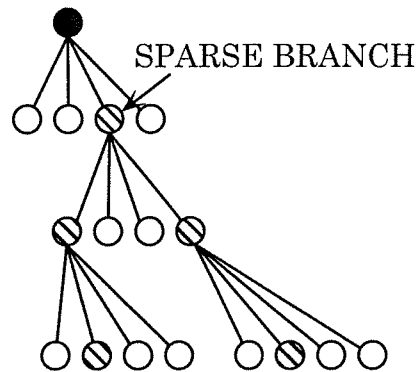


FIG. 94

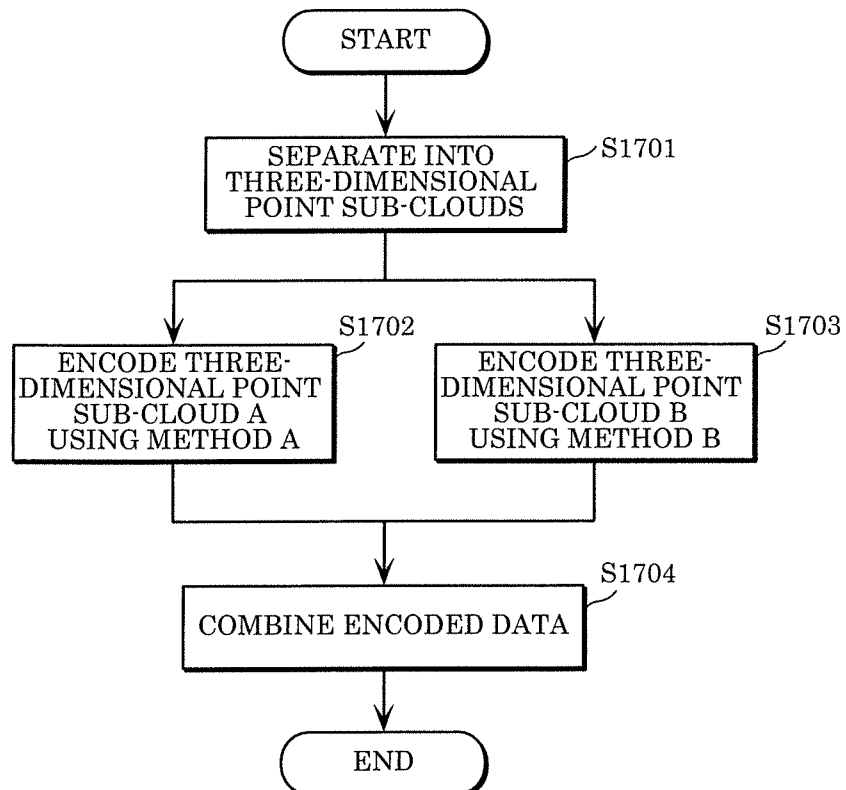


FIG. 95

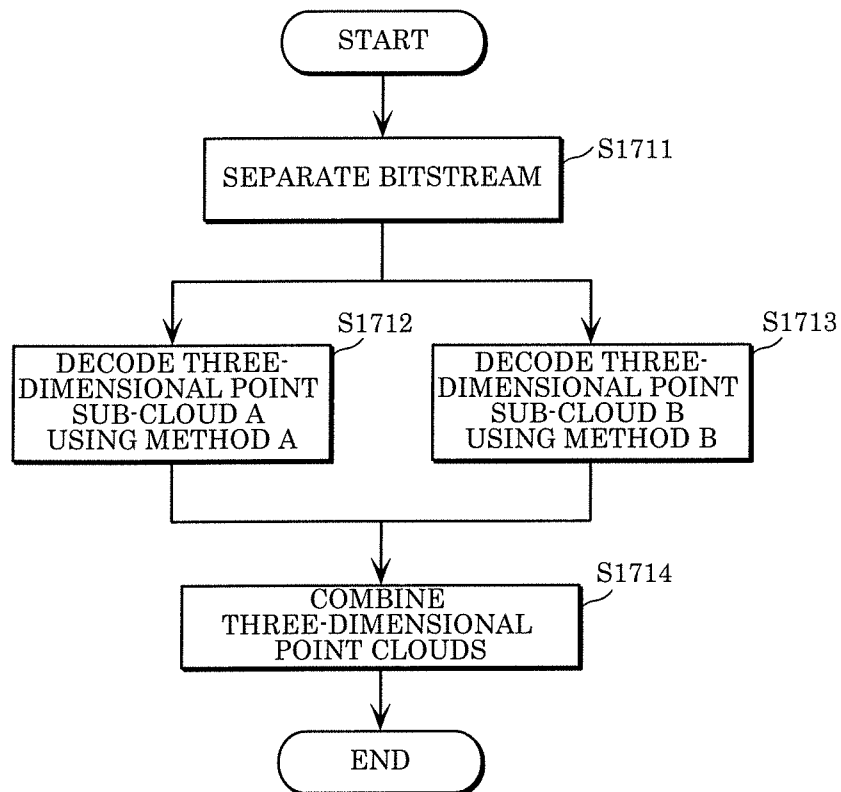


FIG. 96

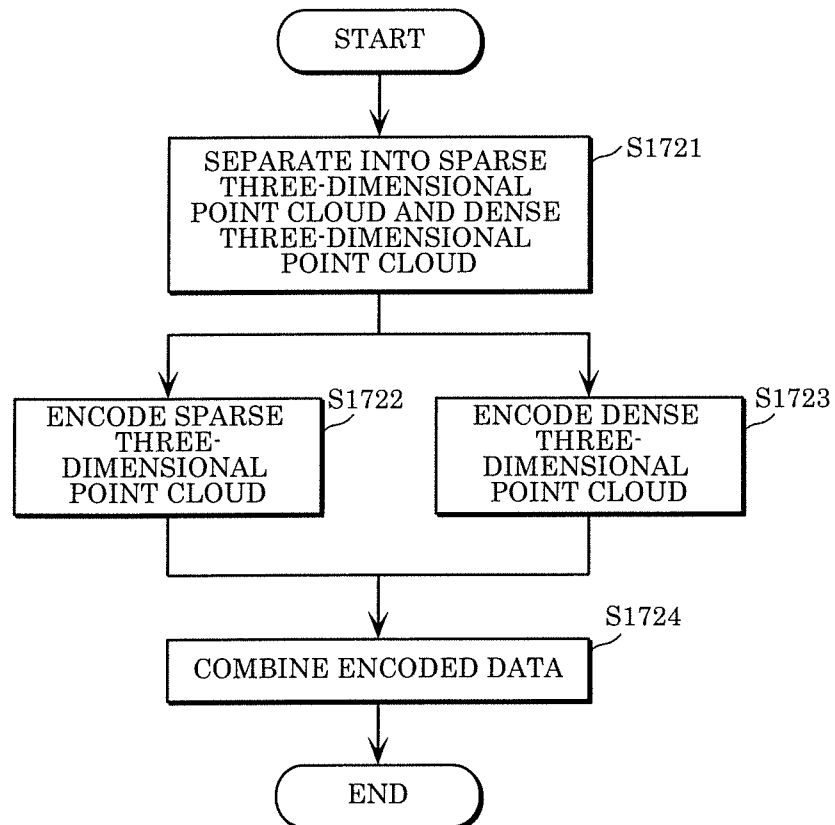


FIG. 97

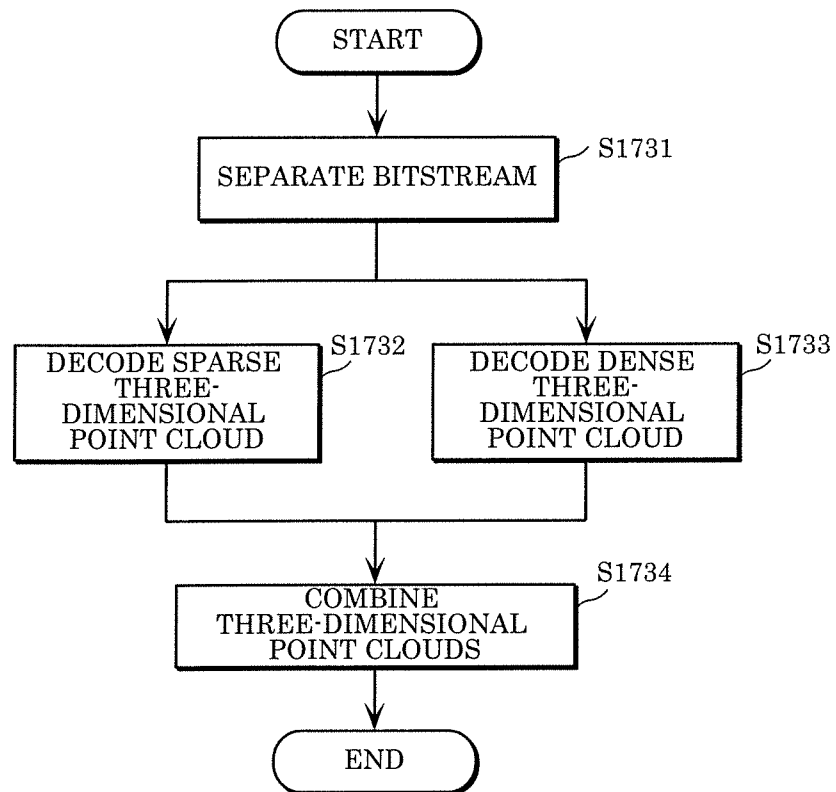


FIG. 98

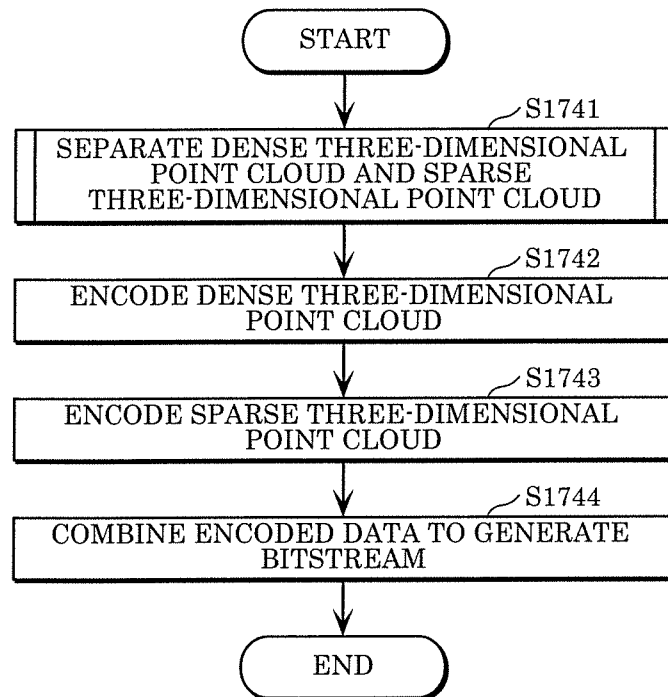


FIG. 99

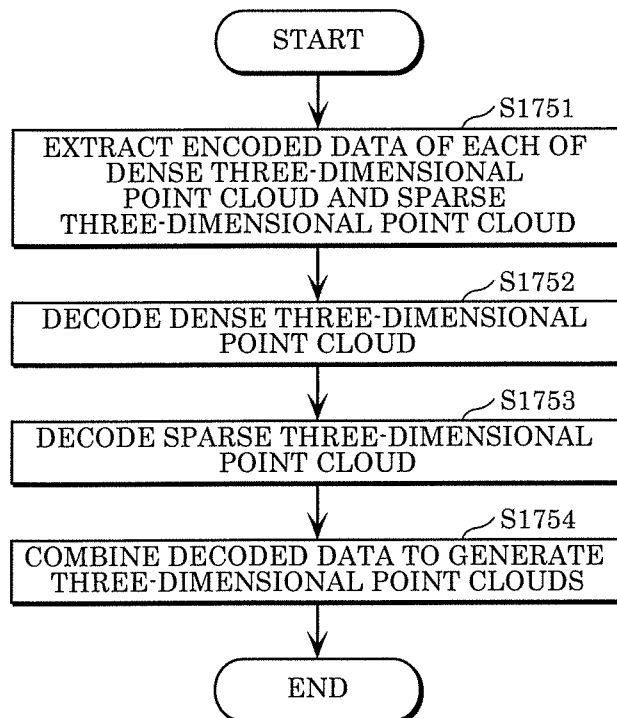


FIG. 100

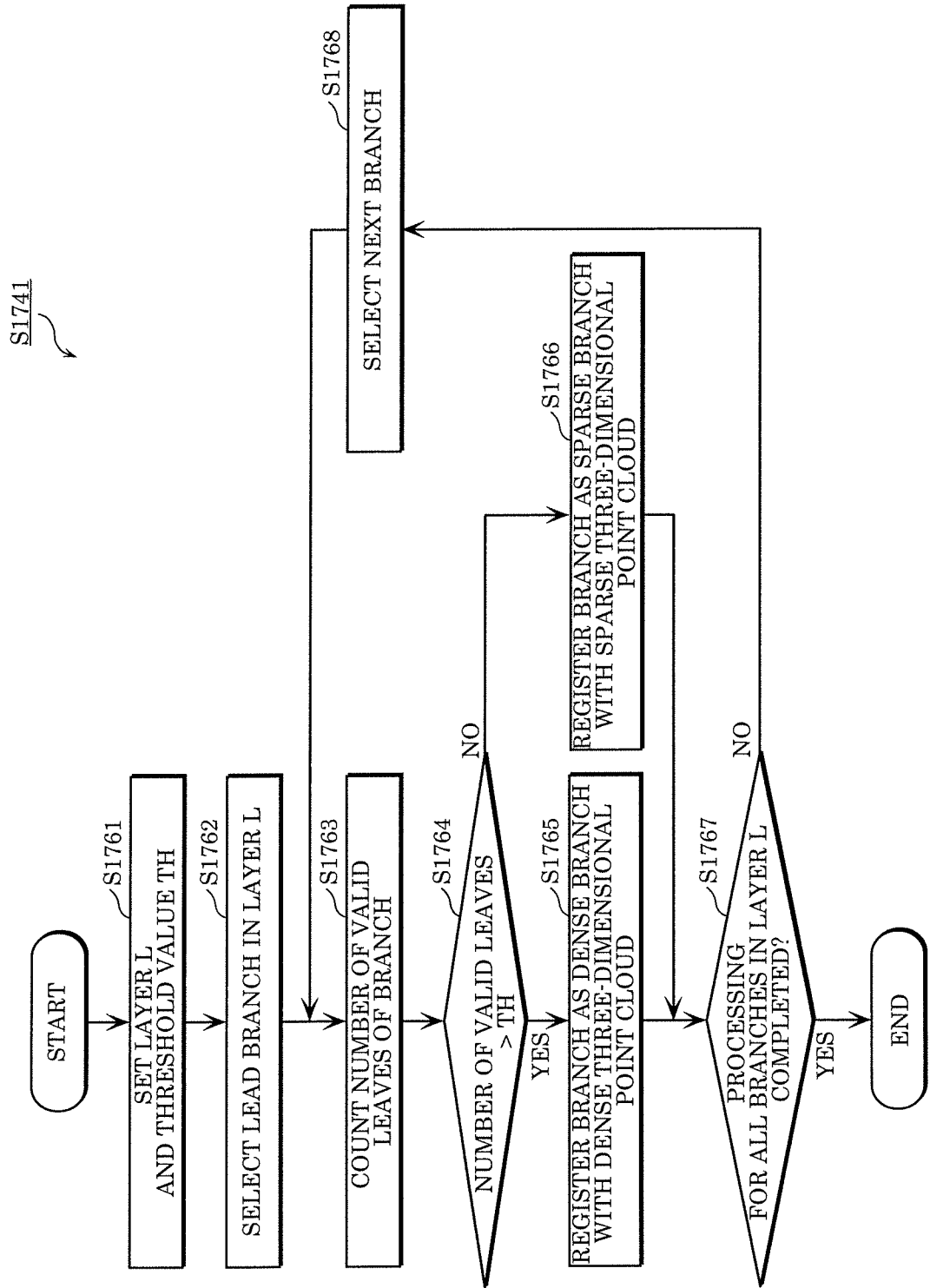


FIG. 101

```

pc_header() {
    . . .
    num_sub_pc
    for (i=0; i<num_sub_pc; i++)
        numPoint[i]
        coding_type[i]
        data_sub_cloud[i]()
    }

    data_sub_cloud() {

        if (coding_type == 00)
        {
            coding_type_00_data
        }
        else if (coding_type == 01)
        {
            coding_type_01_data
        }
        . . .
        end_of_data
    }
}

```

FIG. 102

DENSE BRANCH

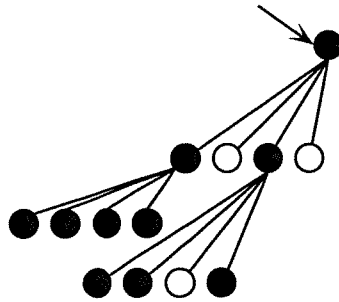


FIG. 103

SPARSE BRANCH

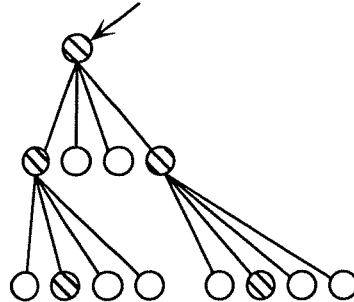


FIG. 104

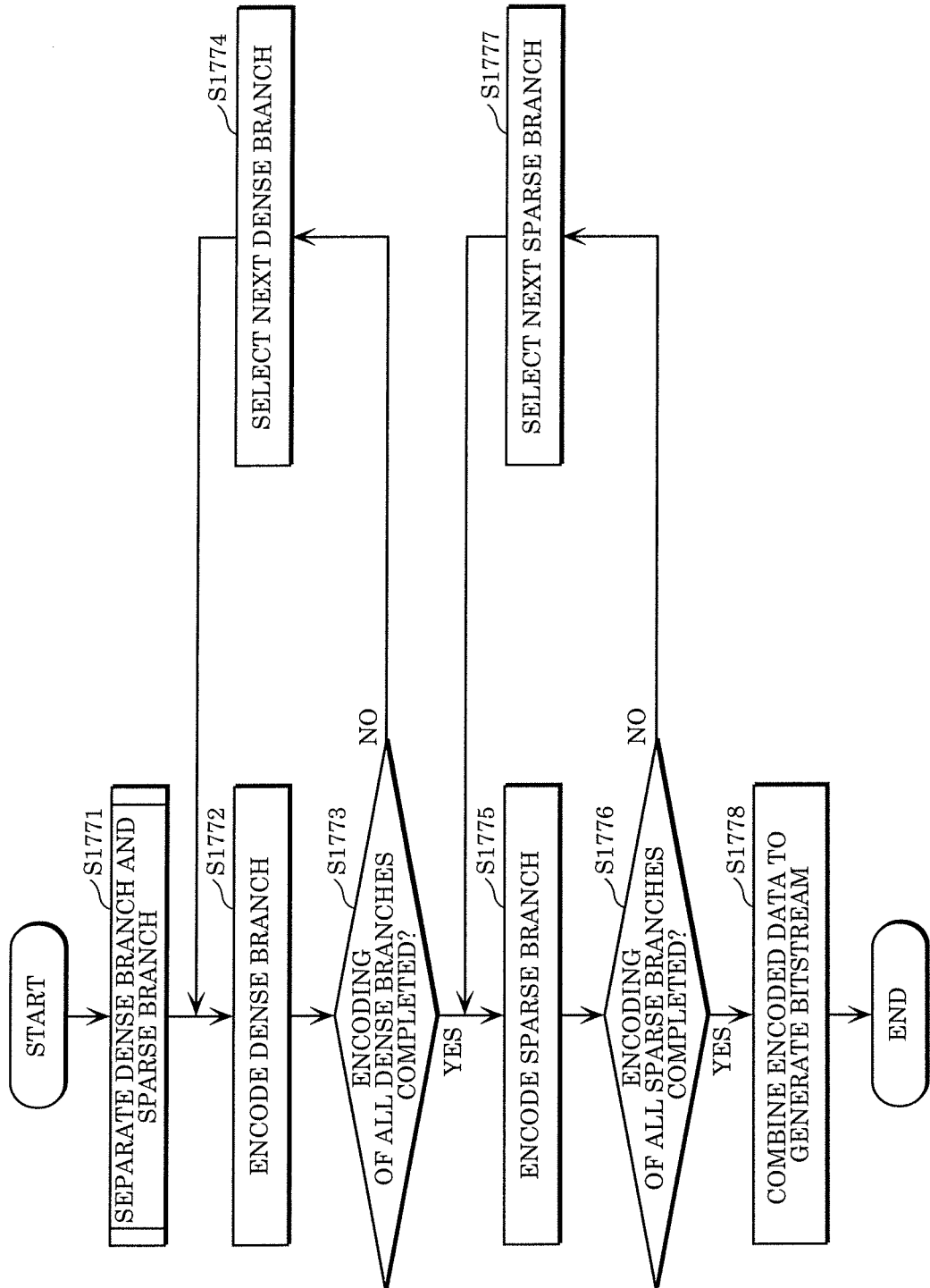


FIG. 105

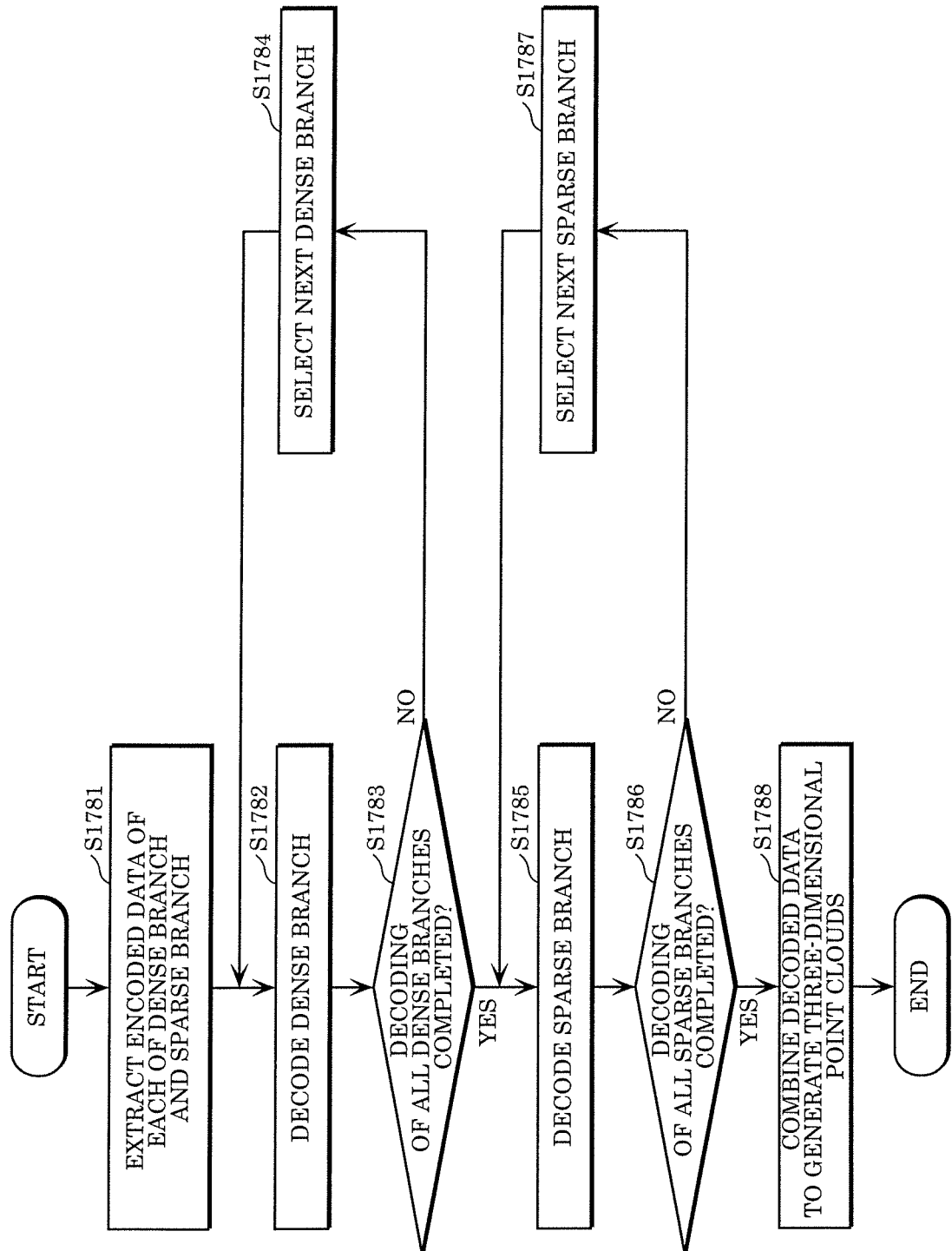


FIG. 106

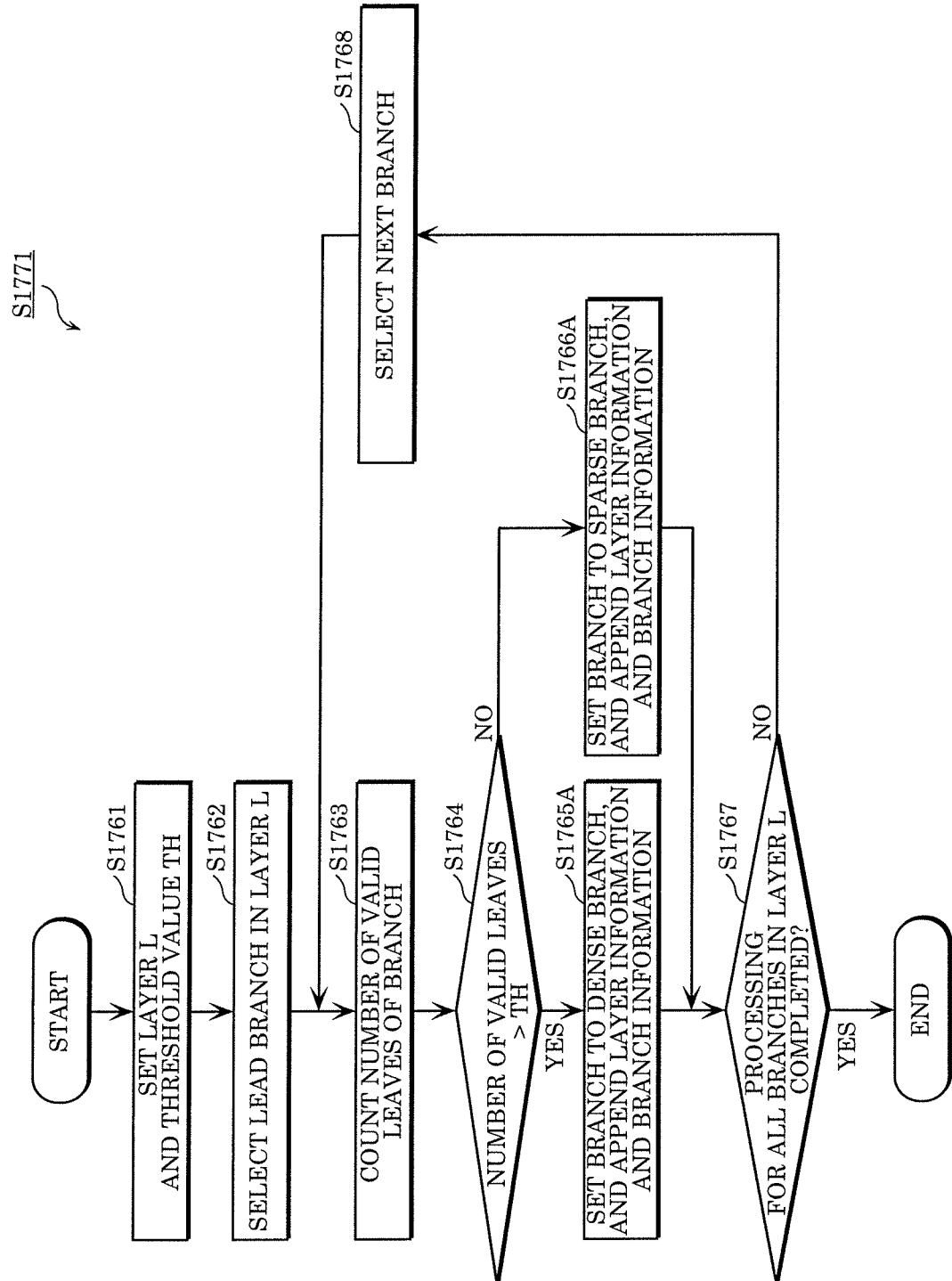


FIG. 107

```
pc_header() {  
  . . .  
  num_sub_pc  
  for (i=0; i<num_sub_pc; i++)  
    layer_id[i]  
    branch_id[i]  
    numPoint[i]  
    coding_type[i]  
    data_sub_cloud[i]()  
}  
  
data_sub_cloud() {  
  
  if (coding_type == 00)  
  {  
    coding_type_00_data  
  }  
  else if (coding_type == 01)  
  {  
    coding_type_01_data  
  }  
  . . .  
  end_of_data  
}
```

FIG. 108

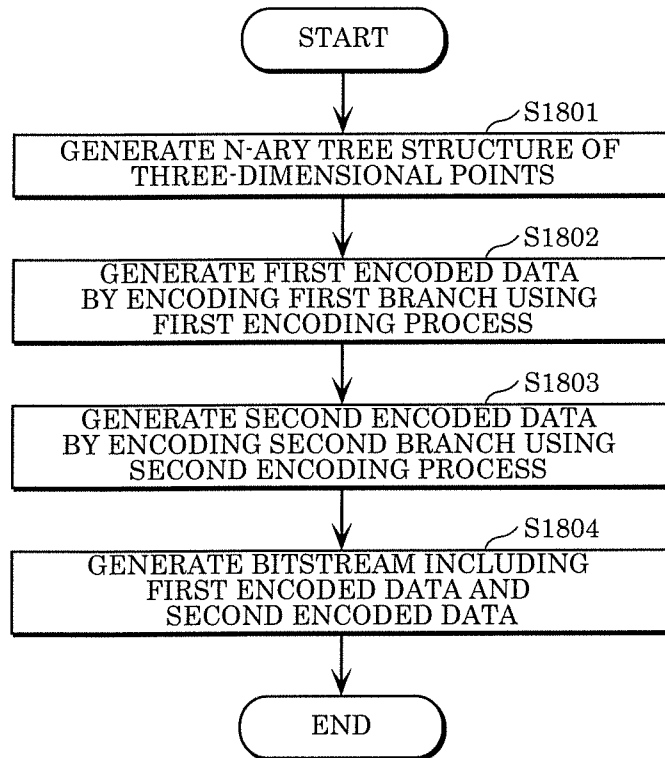


FIG. 109

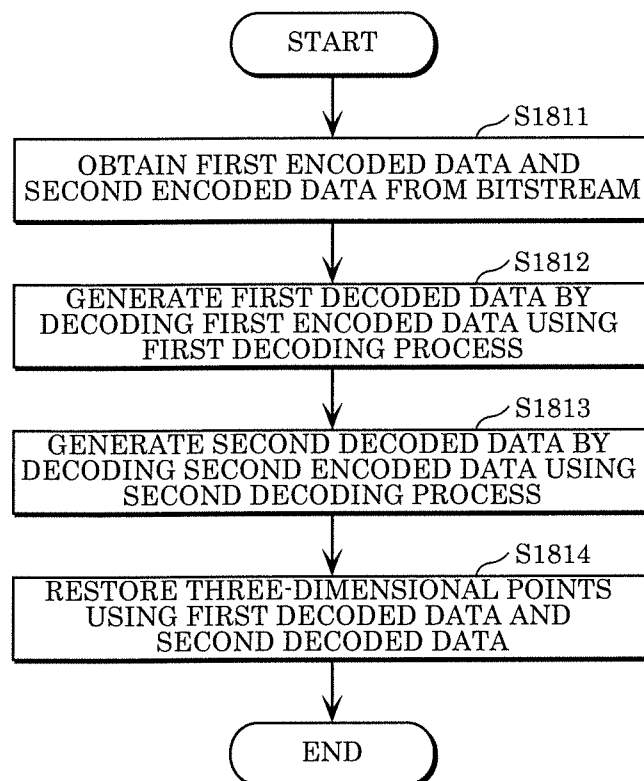


FIG. 110

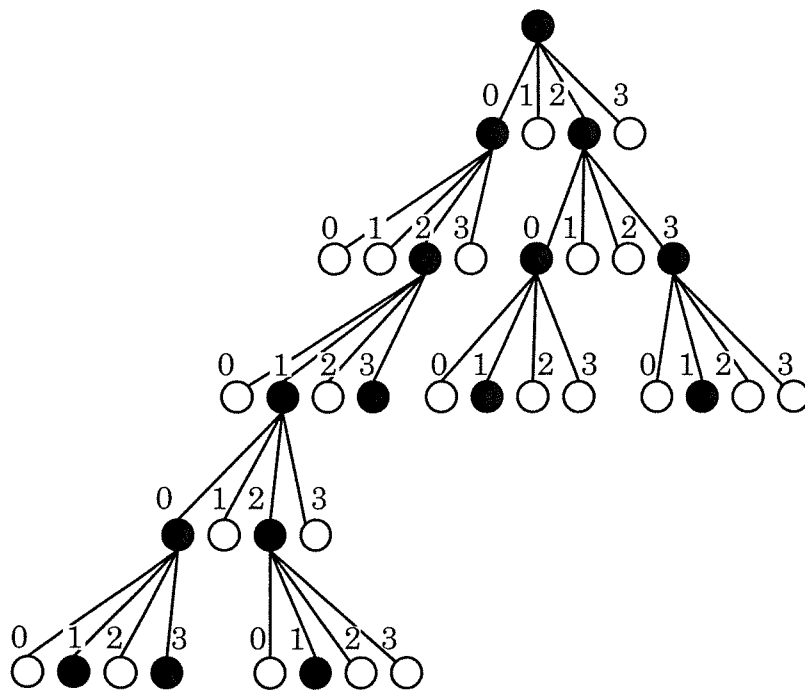


FIG. 111

<u>OCCUPANCY CODE</u>		
1010		
0010	1001	
0101	0100	0100
1010		
0101	0100	

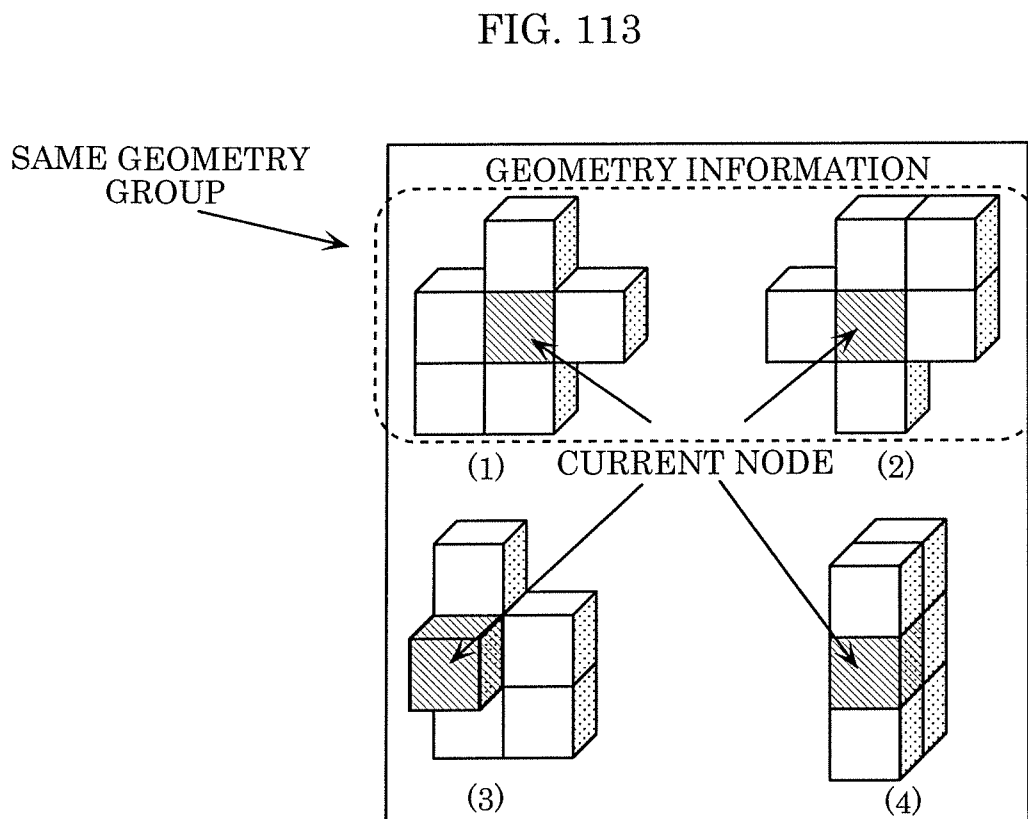
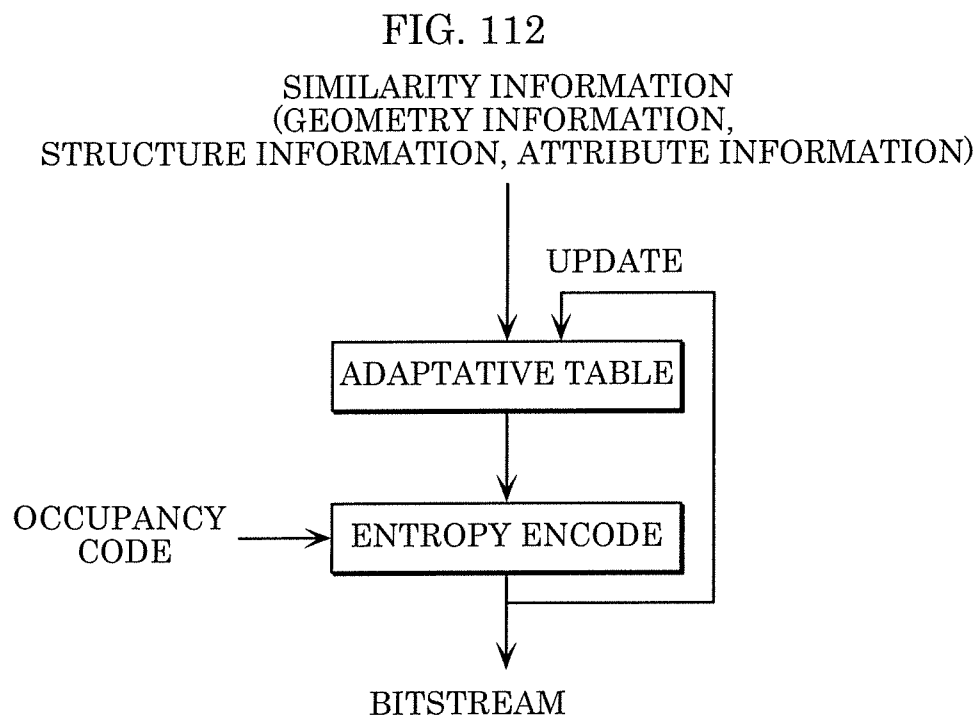


FIG. 114

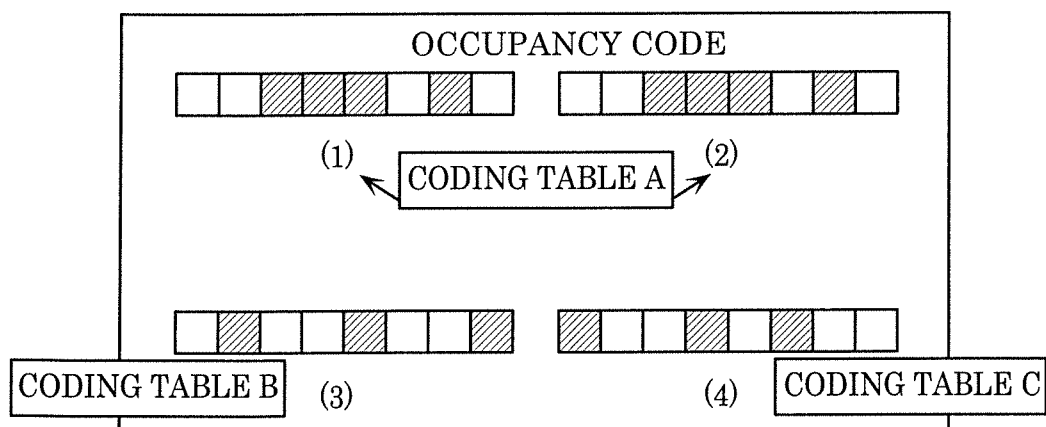


FIG. 115

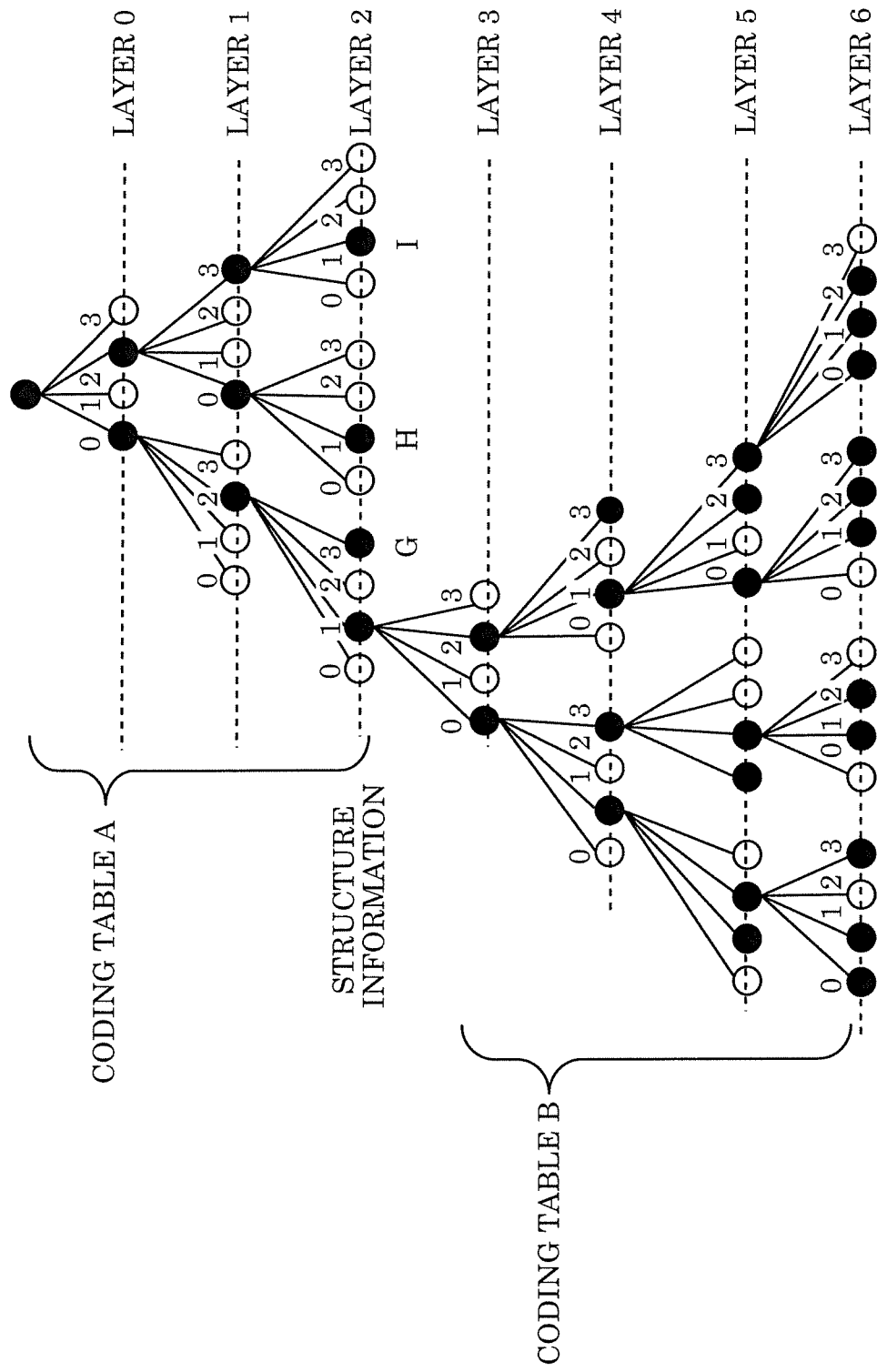


FIG. 116

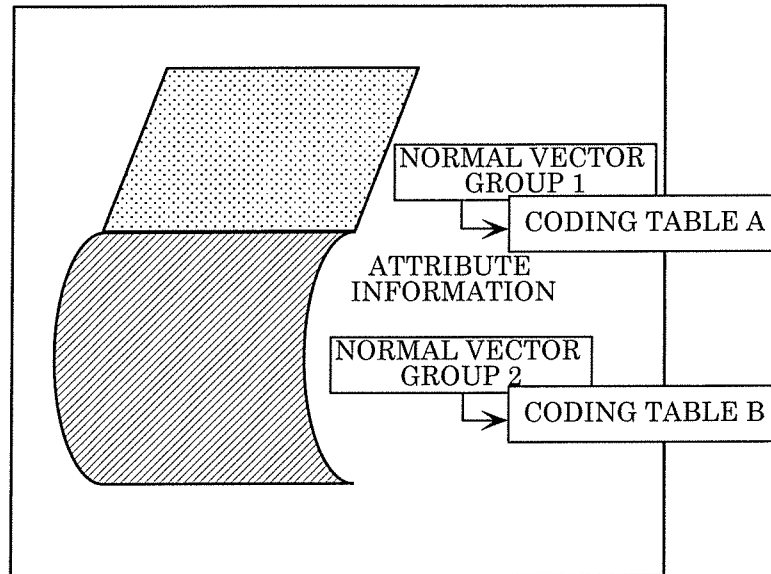


FIG. 117

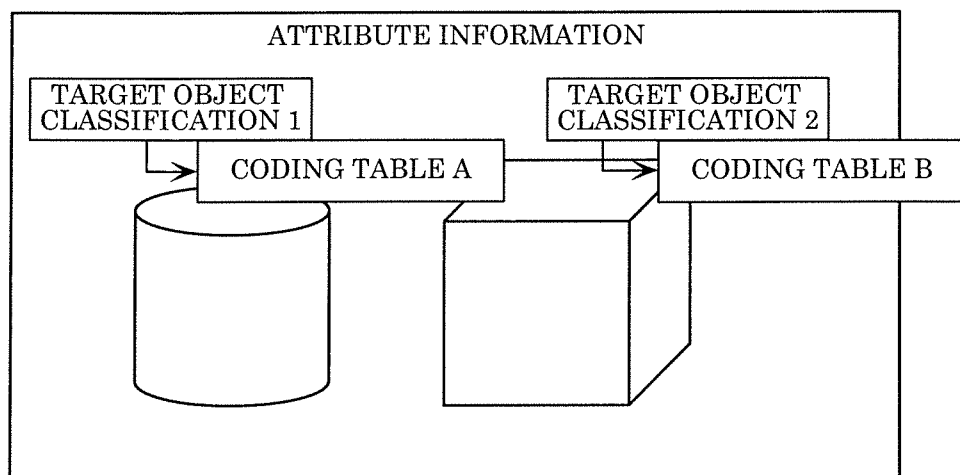


FIG. 118

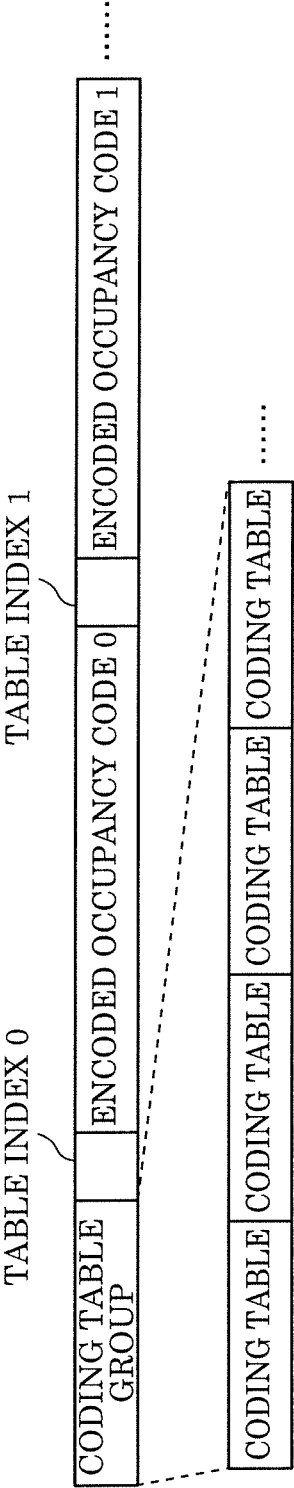


FIG. 119

TABLE INDEX 0

OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
00000000	MODEL 0-0	TYPE A
00000001		
.....		
11111111	MODEL 0-X	TYPE X

FIG. 120

TABLE INDEX 1

OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
00000000	MODEL 1-0	TYPE A
00000001	MODEL 1-1	TYPE B
.....		
11111111	MODEL 1-Y	TYPE Y

FIG. 121

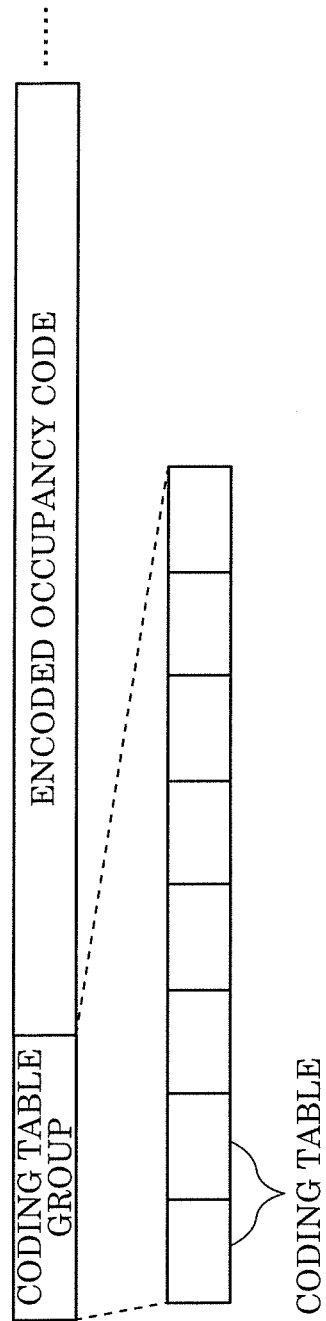


FIG. 122

TABLE INDEX 0

BIT NUMBER OF OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
1	MODEL 0-0	TYPE A
2	MODEL 0-1	
.....		
8	MODEL 0-7	TYPE B

FIG. 123

TABLE INDEX 1

BIT NUMBER OF OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
1	MODEL 1-0	TYPE A
2		TYPE A
.....		
8	MODEL 1-Y	TYPE B

FIG. 124

OCCUPANCY CODE: 0 0 0 1 0 0 0 1

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

BIT NUMBER: 1 2 3 4 5 6 7 8

FIG. 125

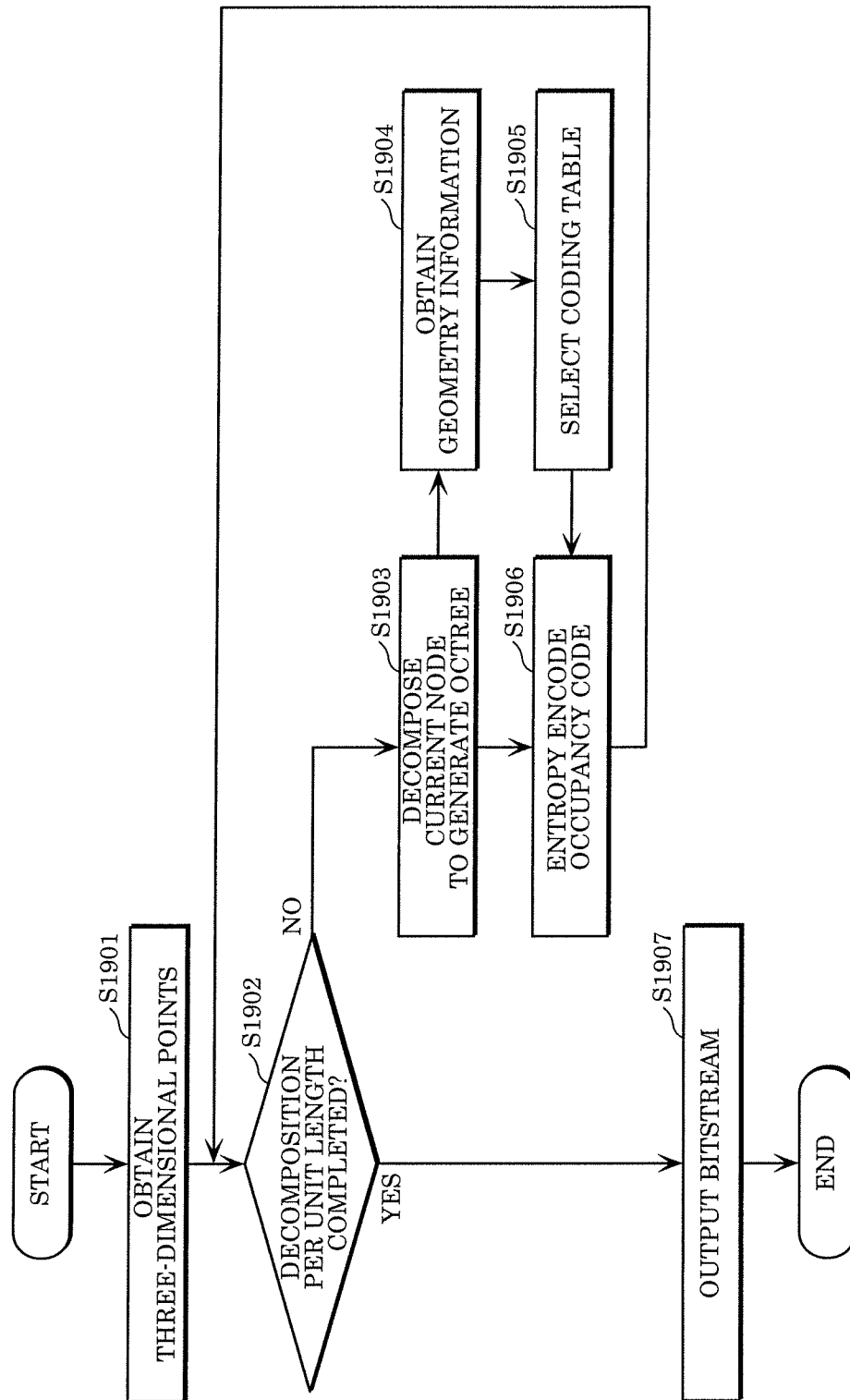


FIG. 126

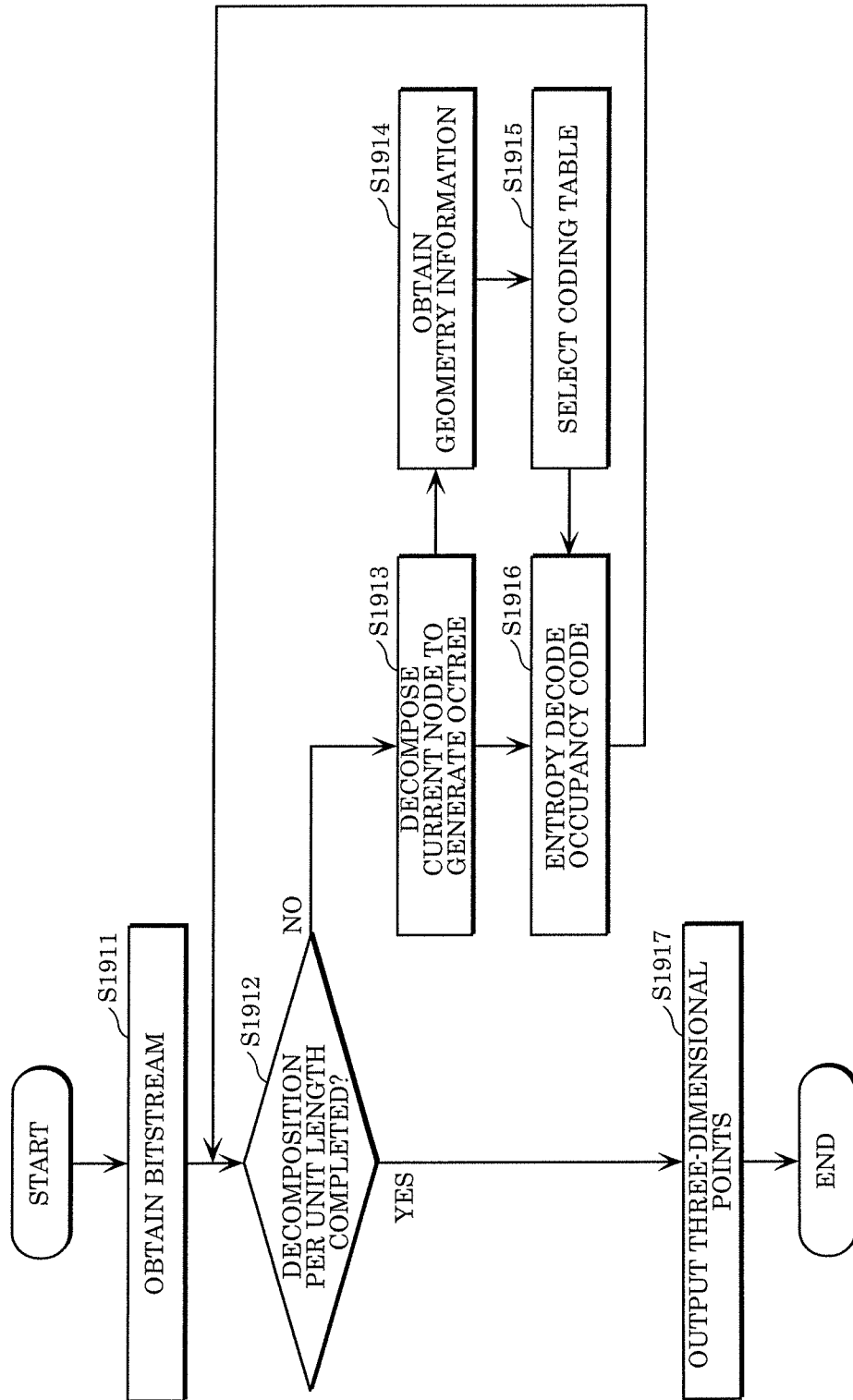


FIG. 127

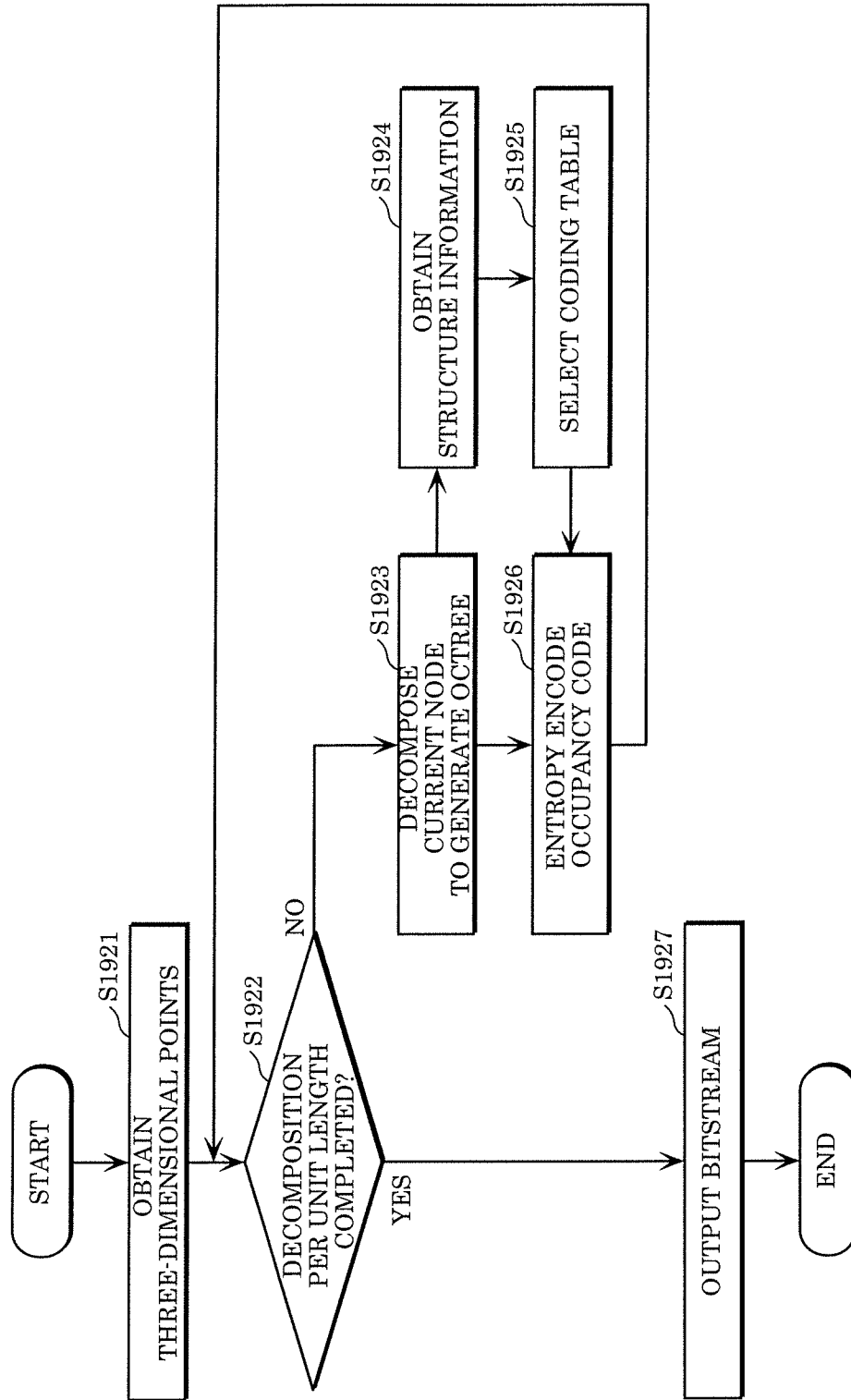


FIG. 128

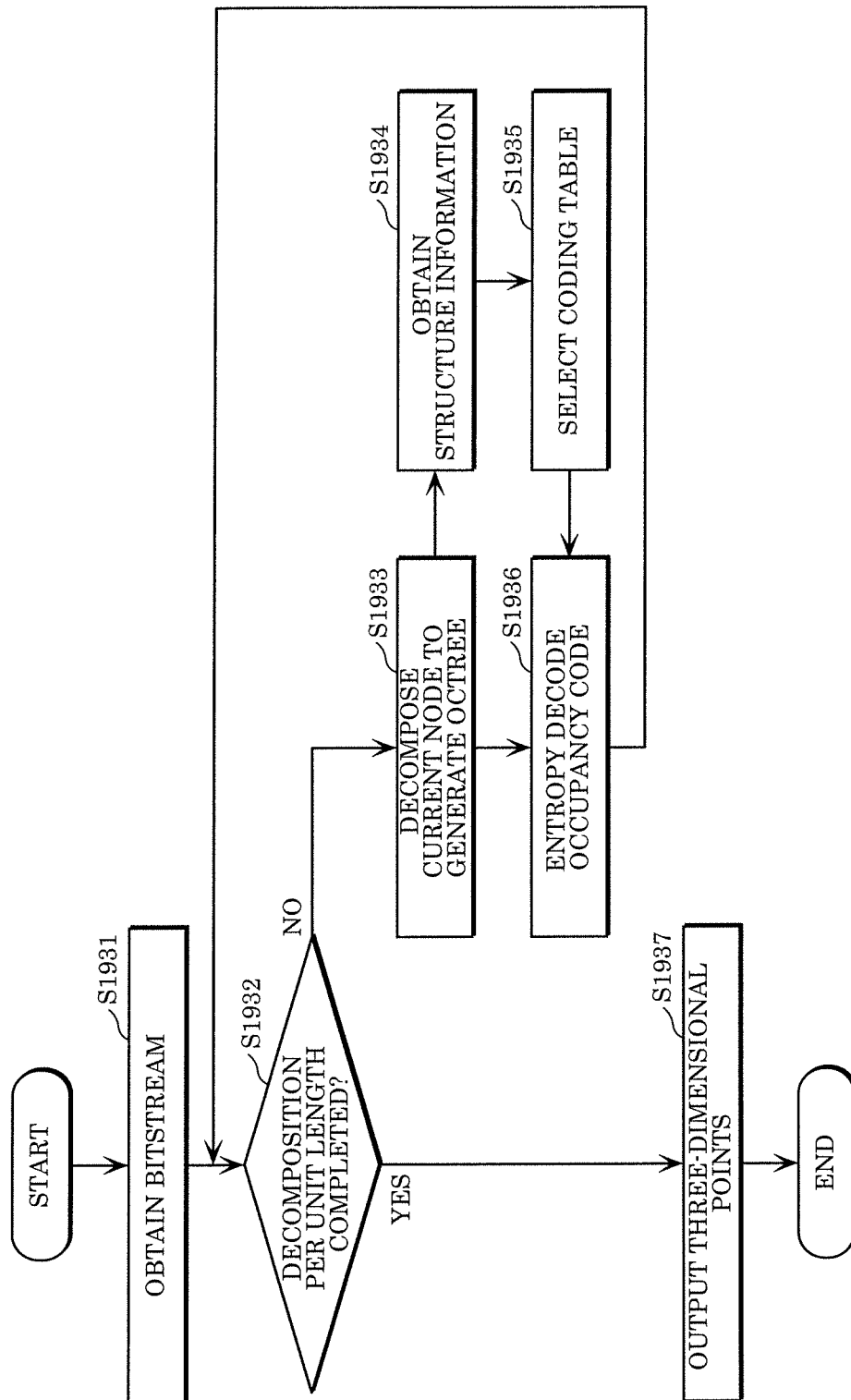


FIG. 129

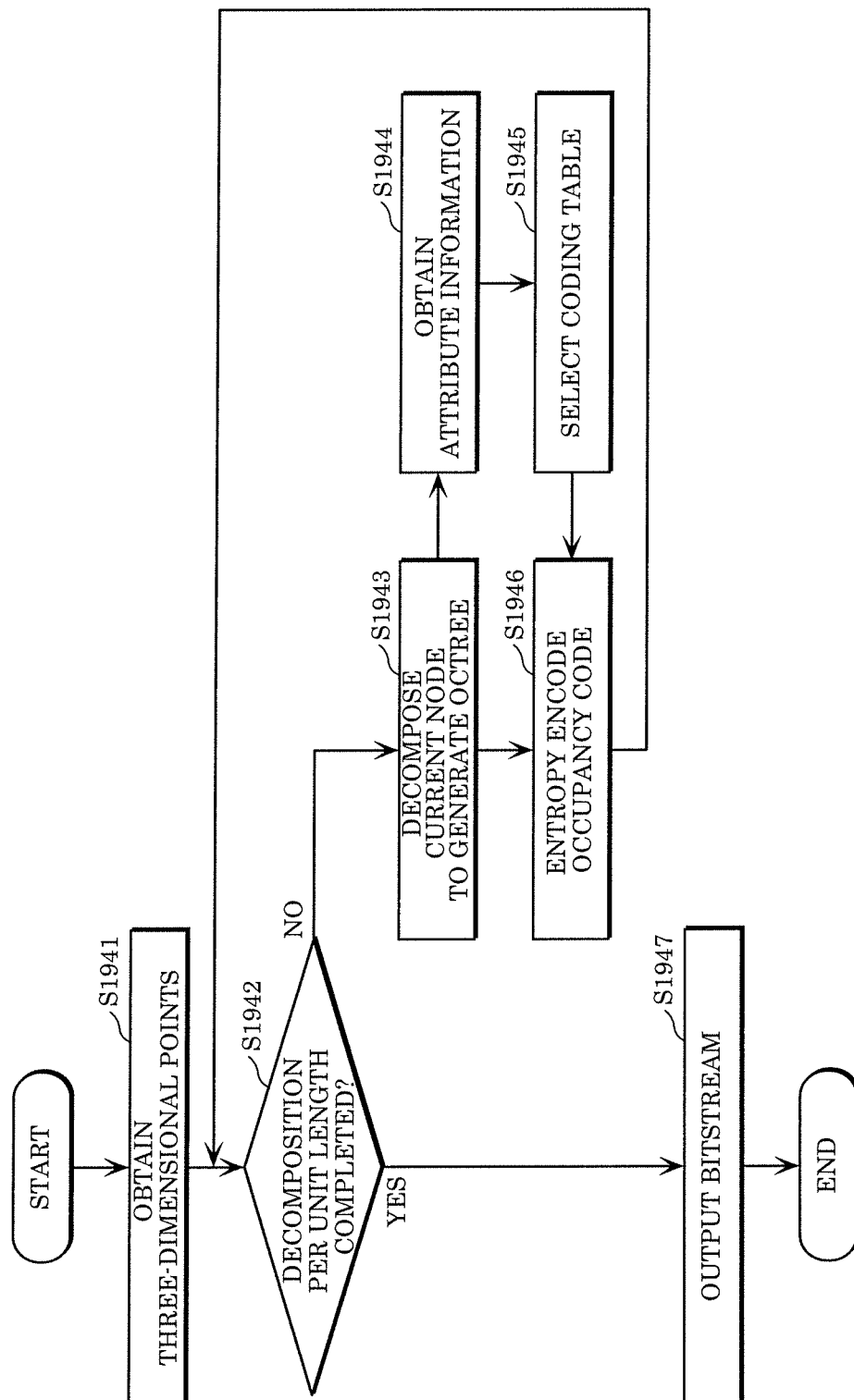


FIG. 130

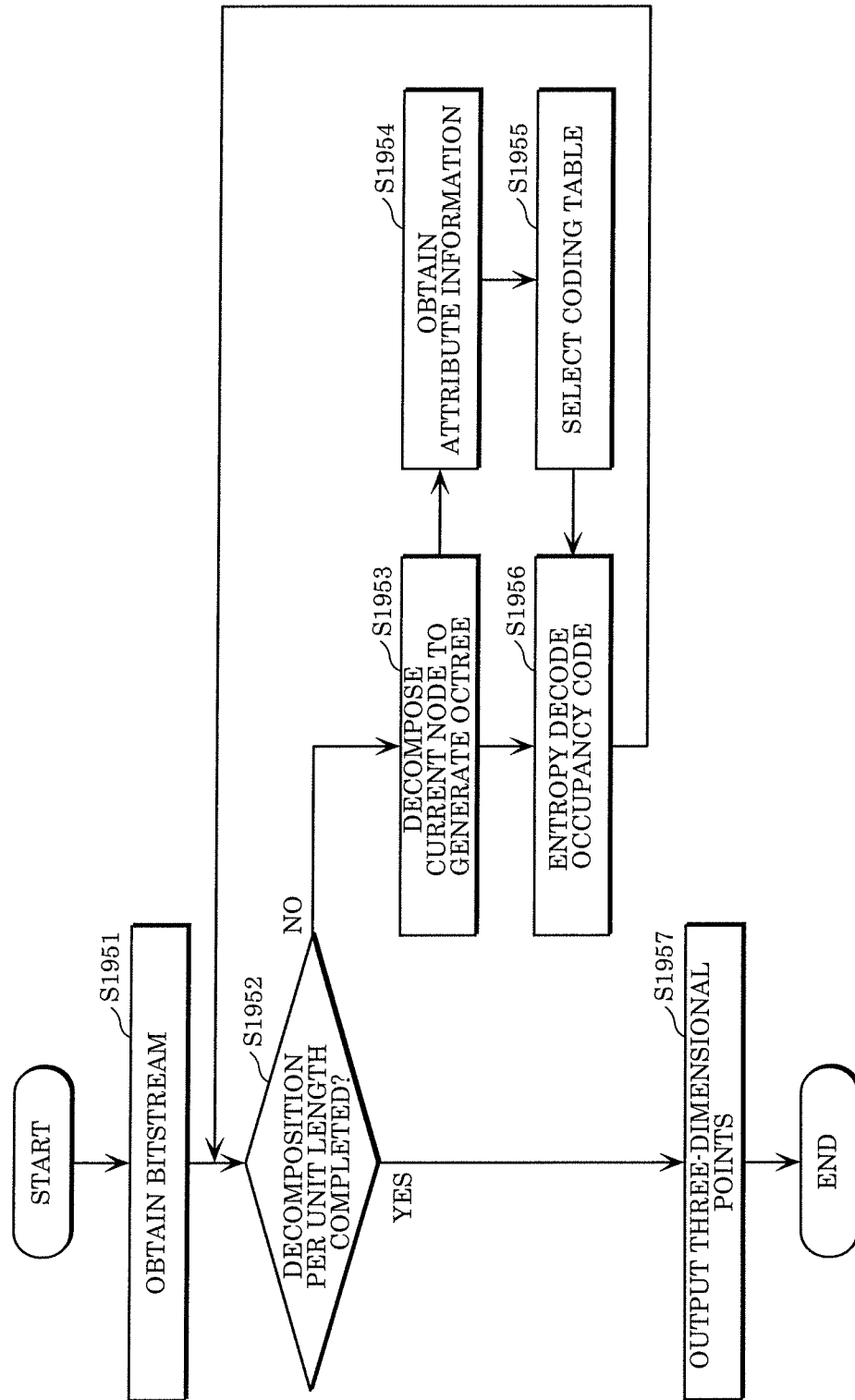


FIG. 131

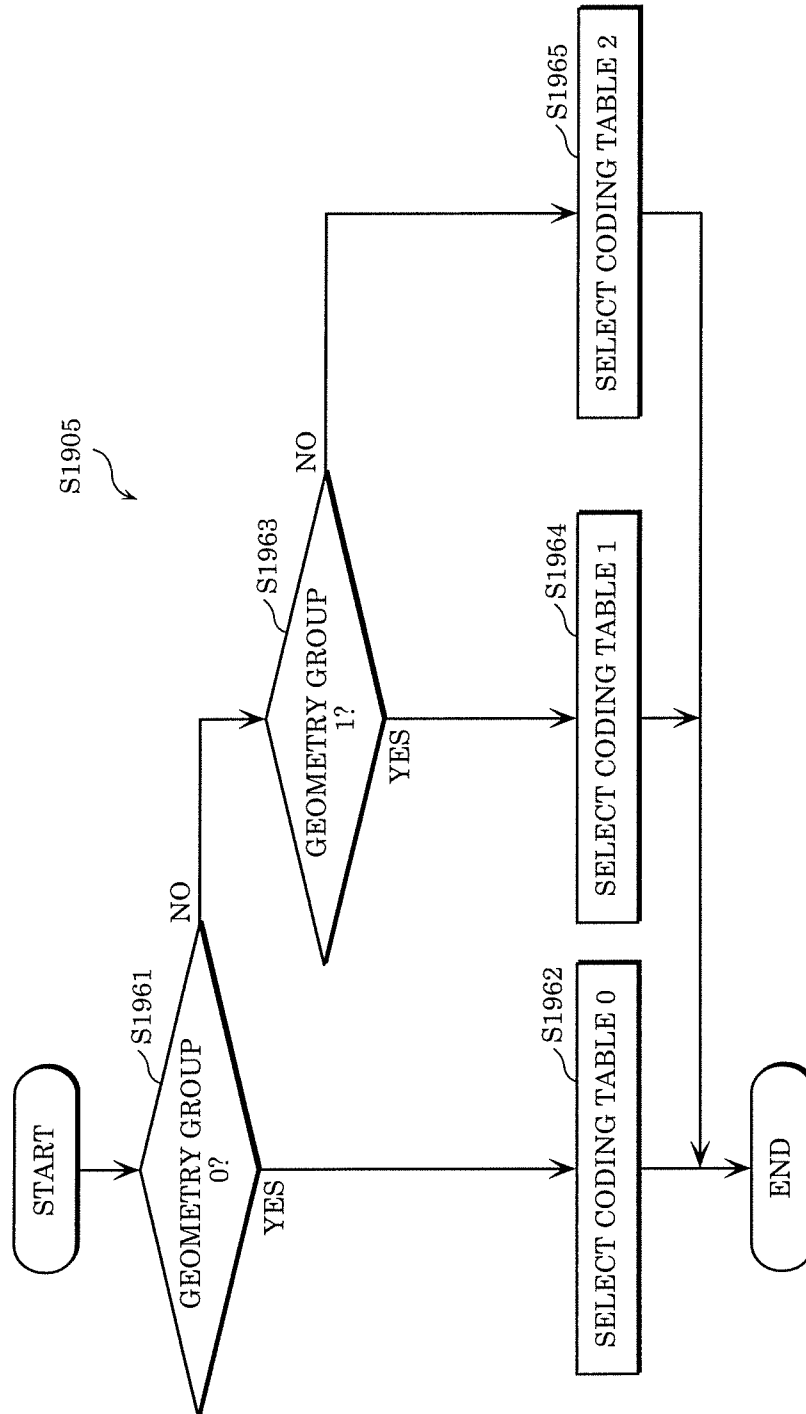


FIG. 132

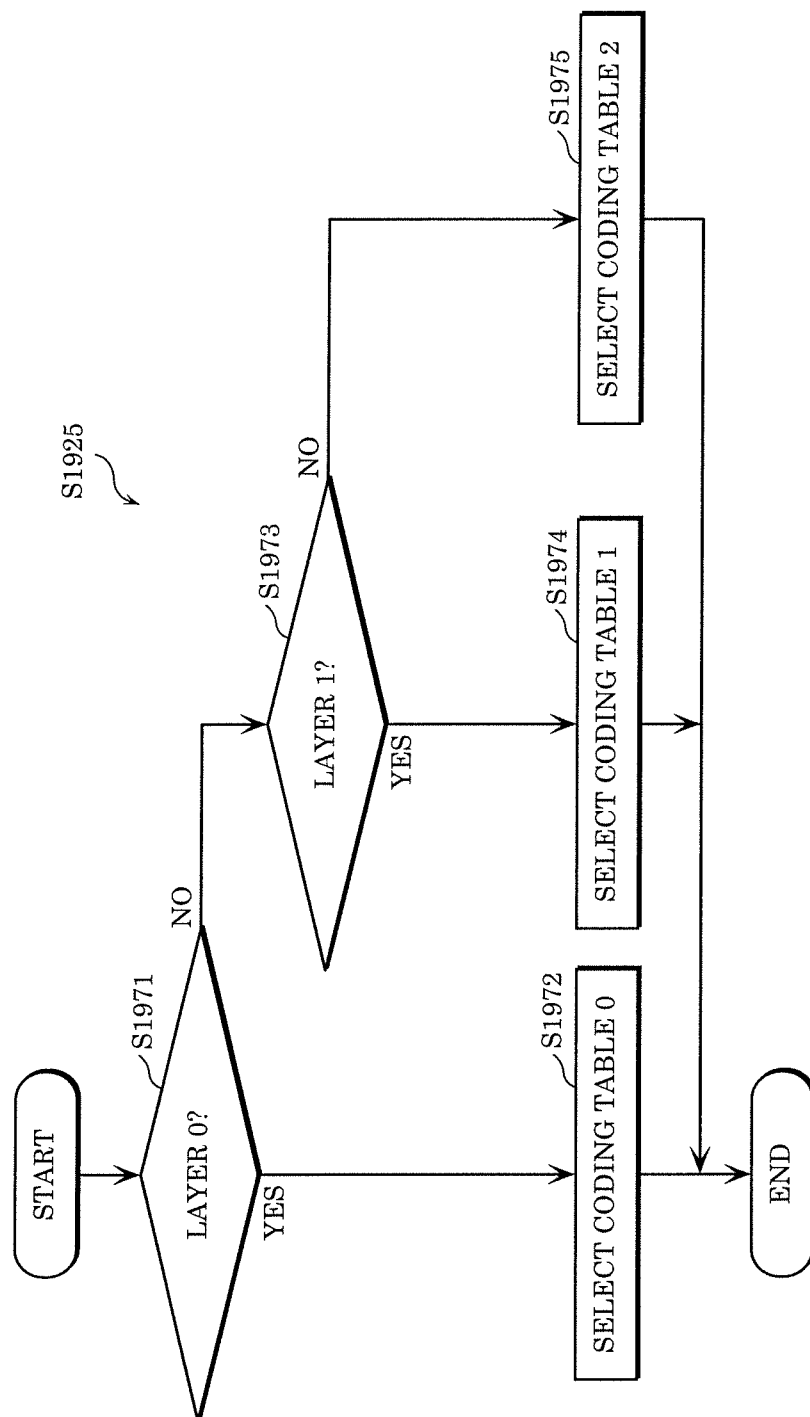


FIG. 133

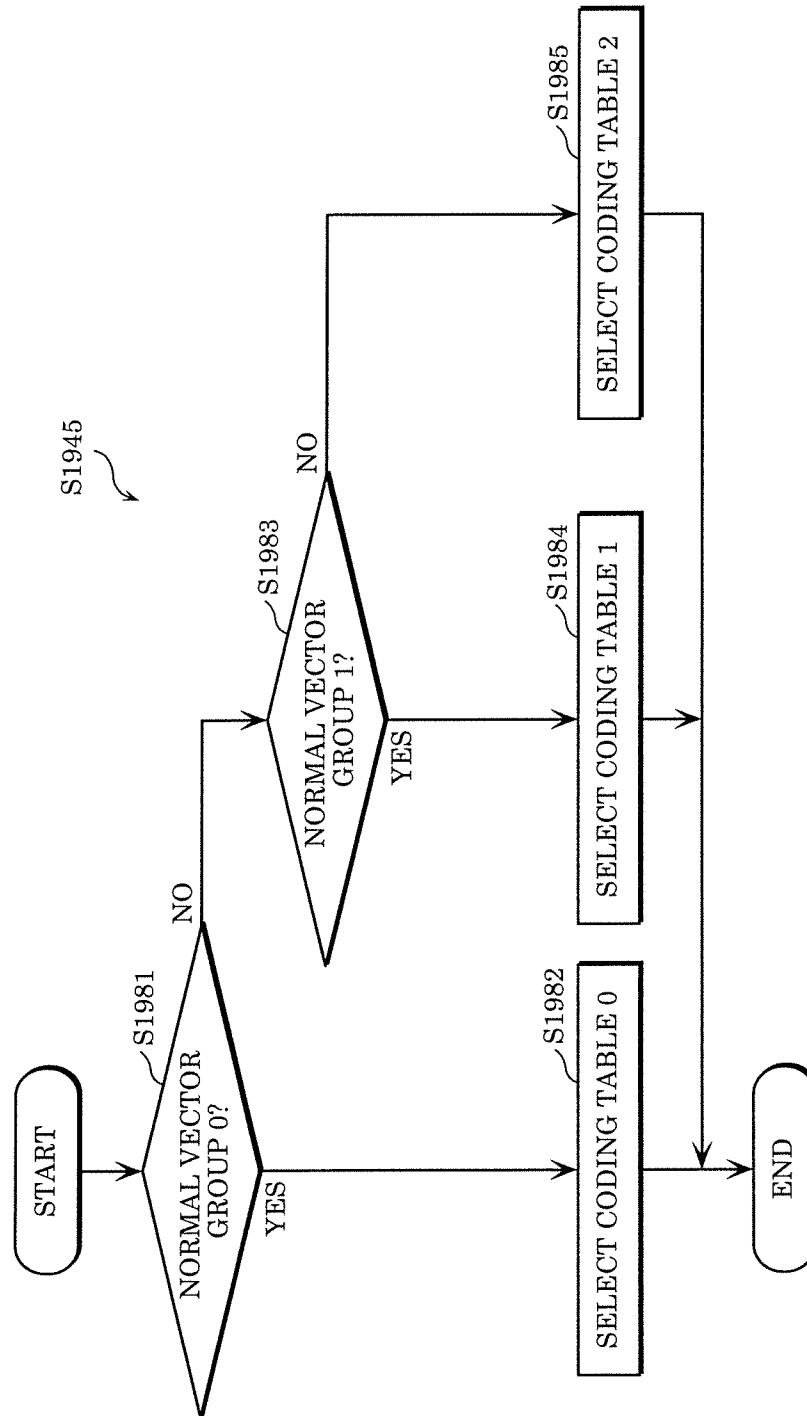


FIG. 134

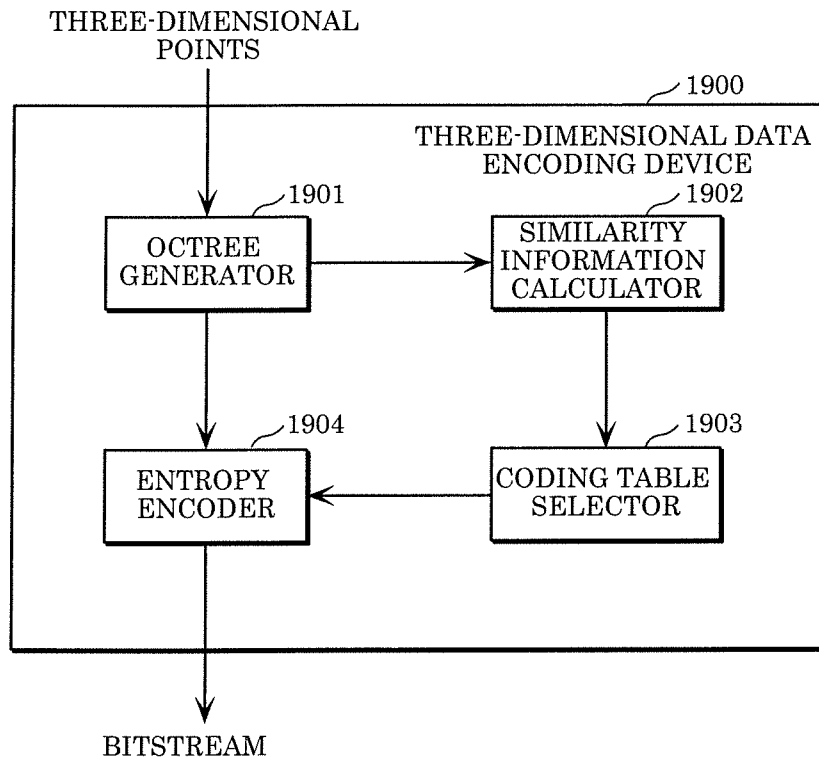


FIG. 135

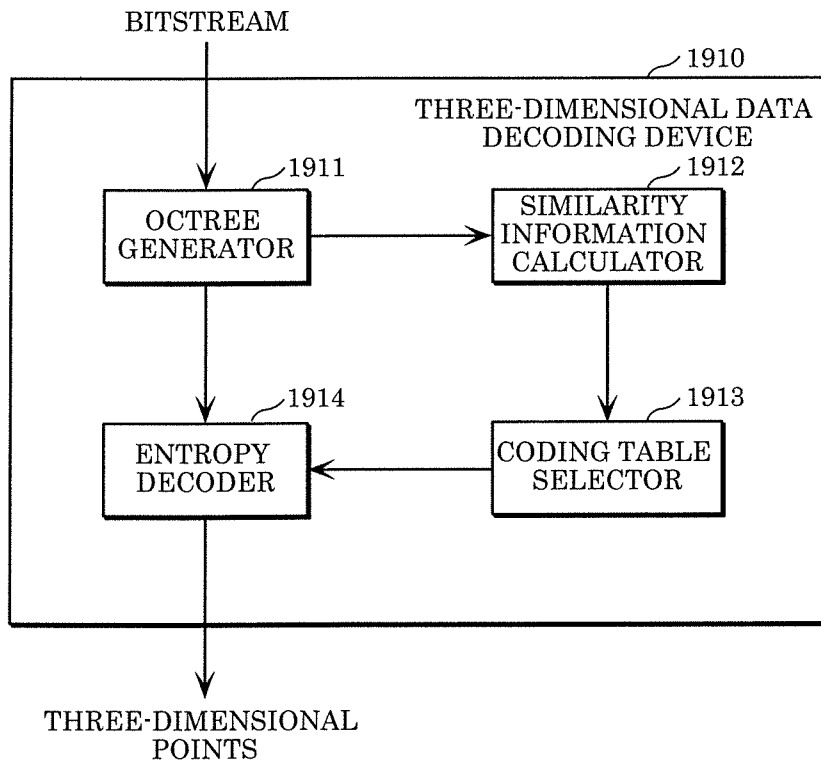


FIG. 136

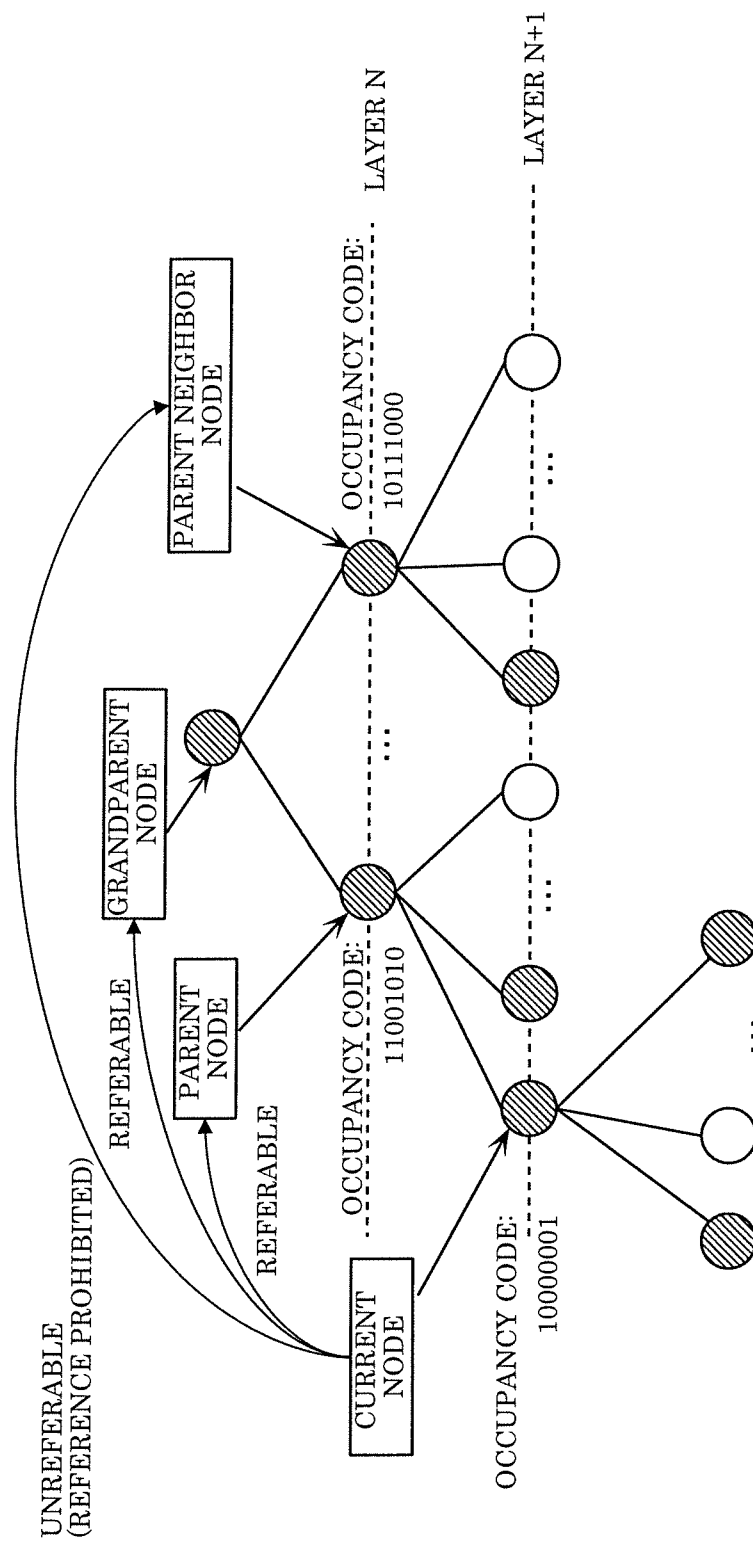


FIG. 137

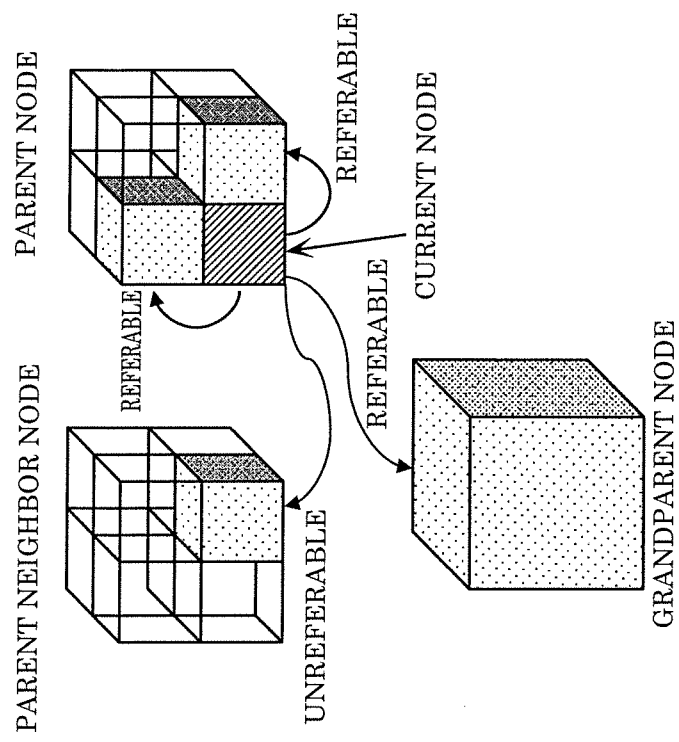


FIG. 138

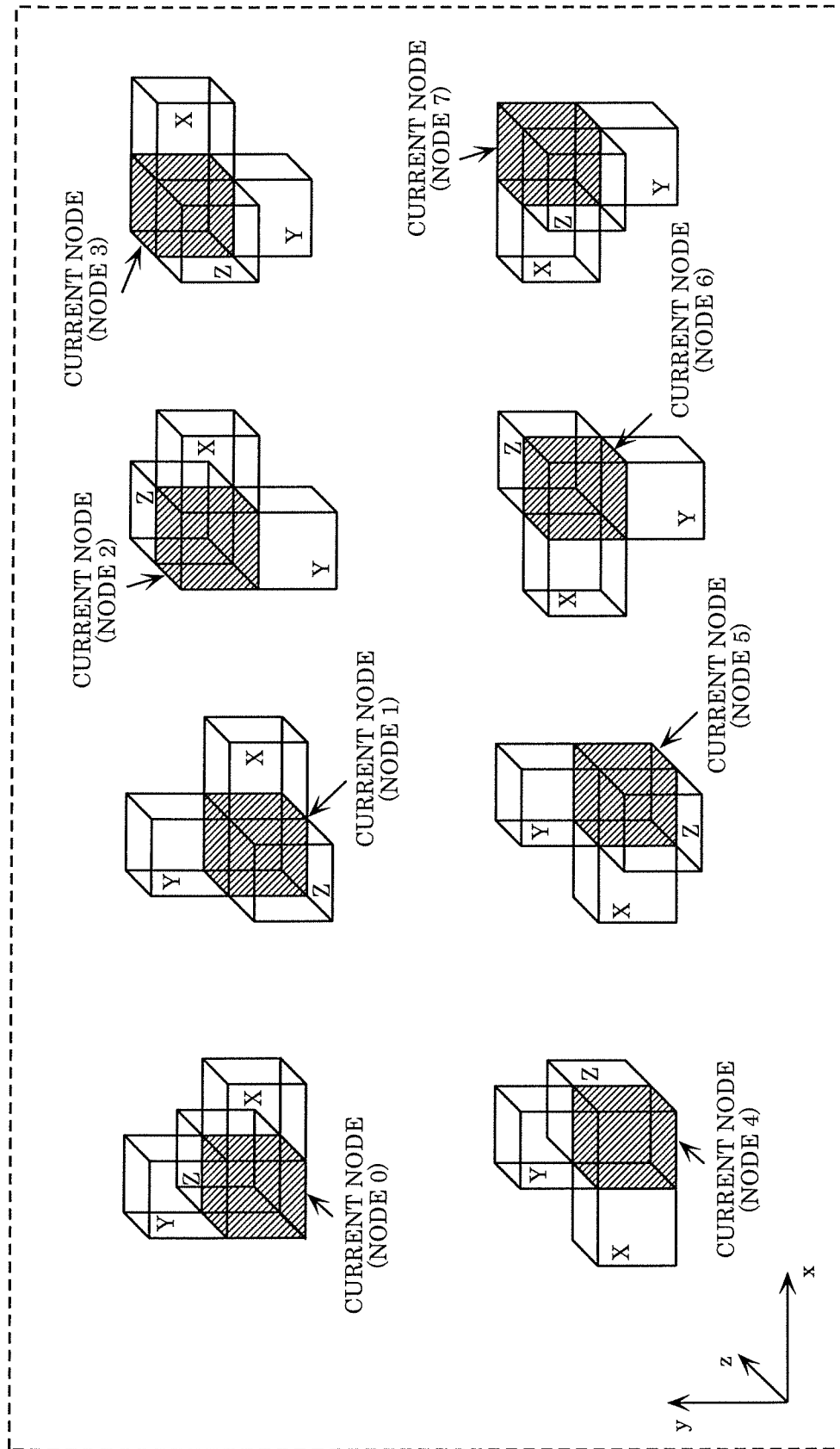


FIG. 139

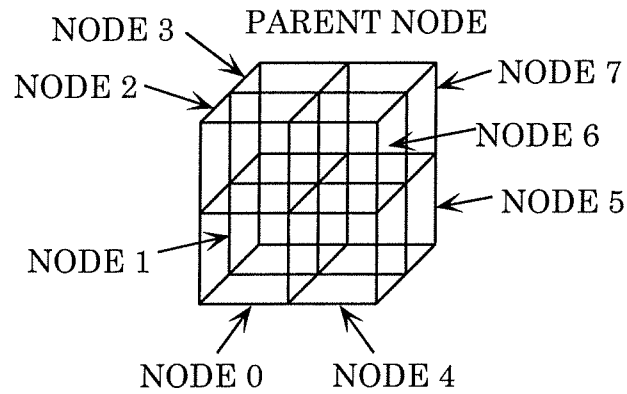


FIG. 140

EXAMPLE OF OCCUPANCY CODE
OF PARENT NODE:

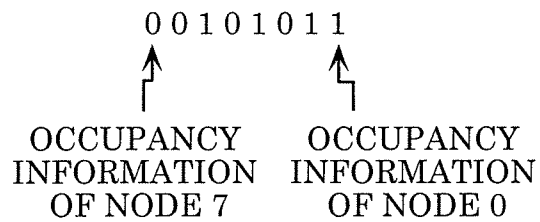


FIG. 141

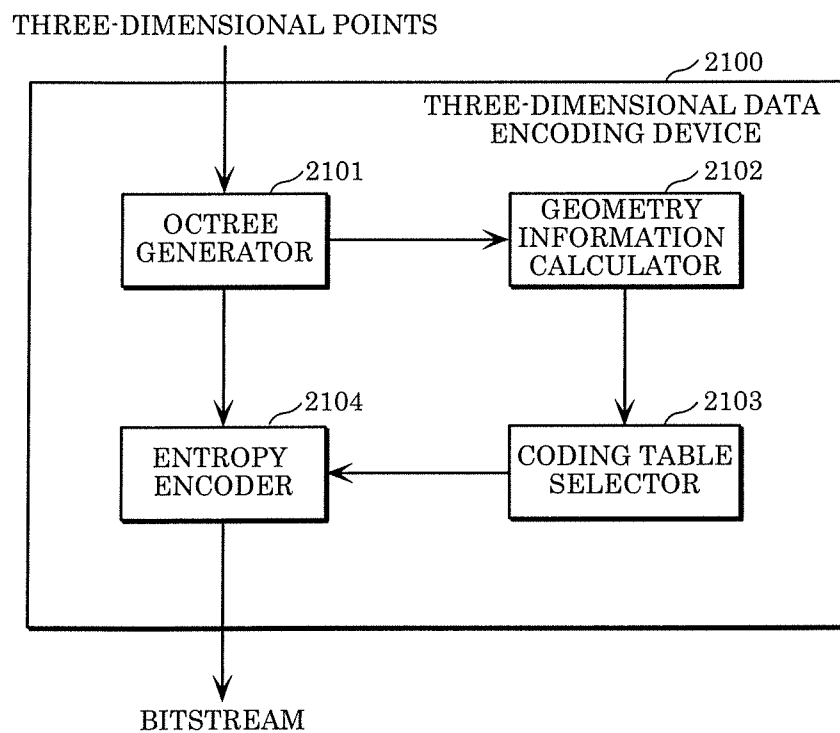


FIG. 142

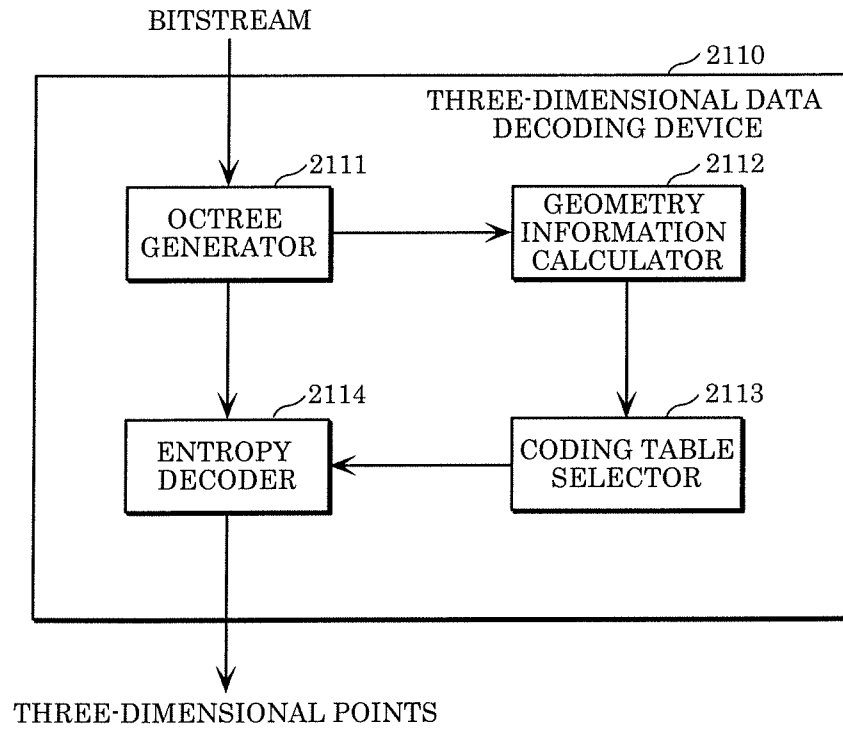


FIG. 143

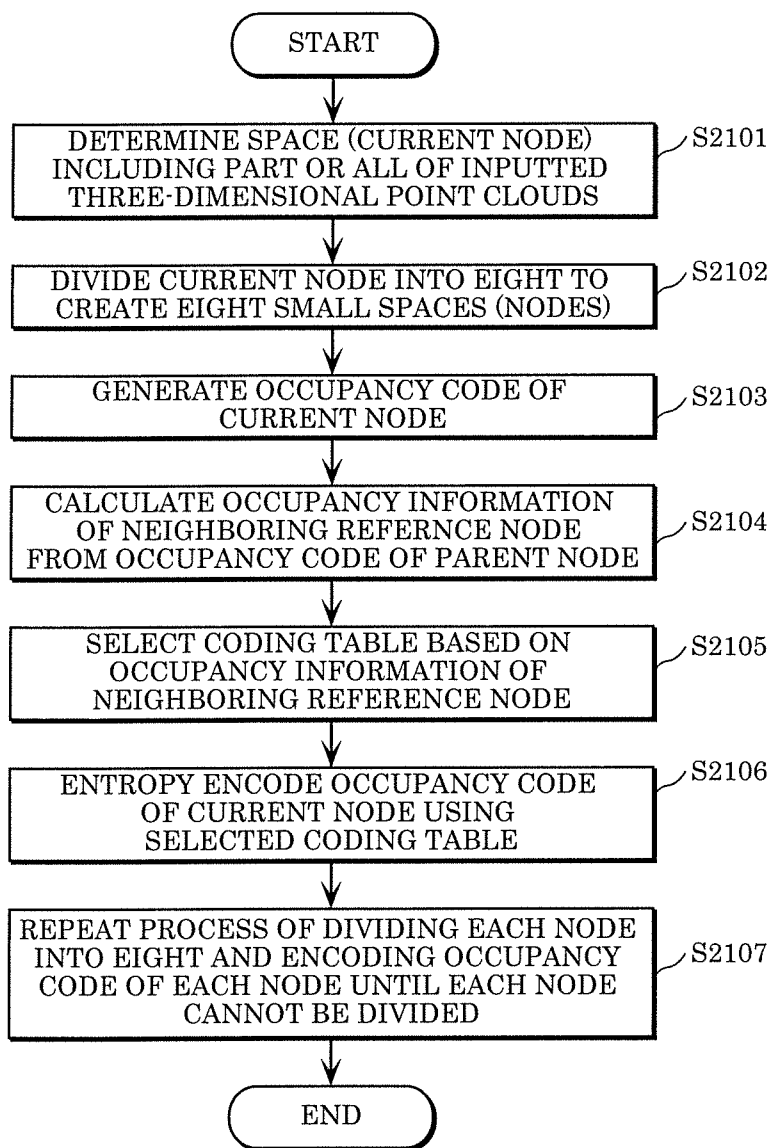


FIG. 144

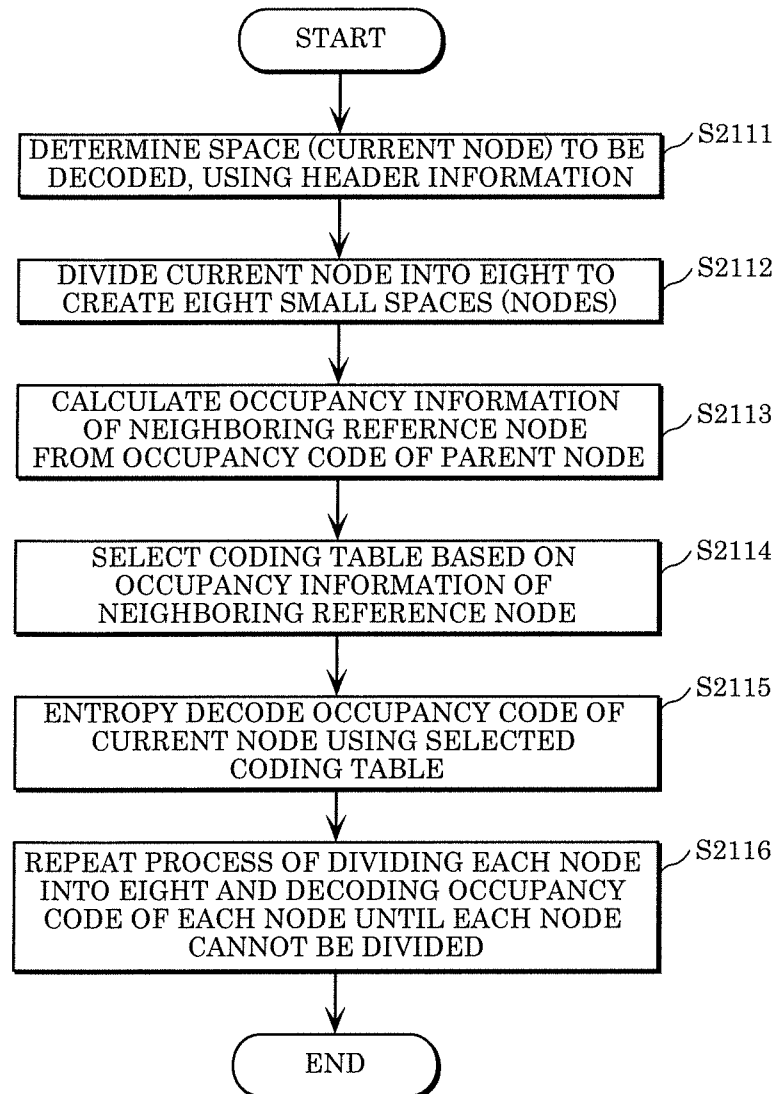


FIG. 145

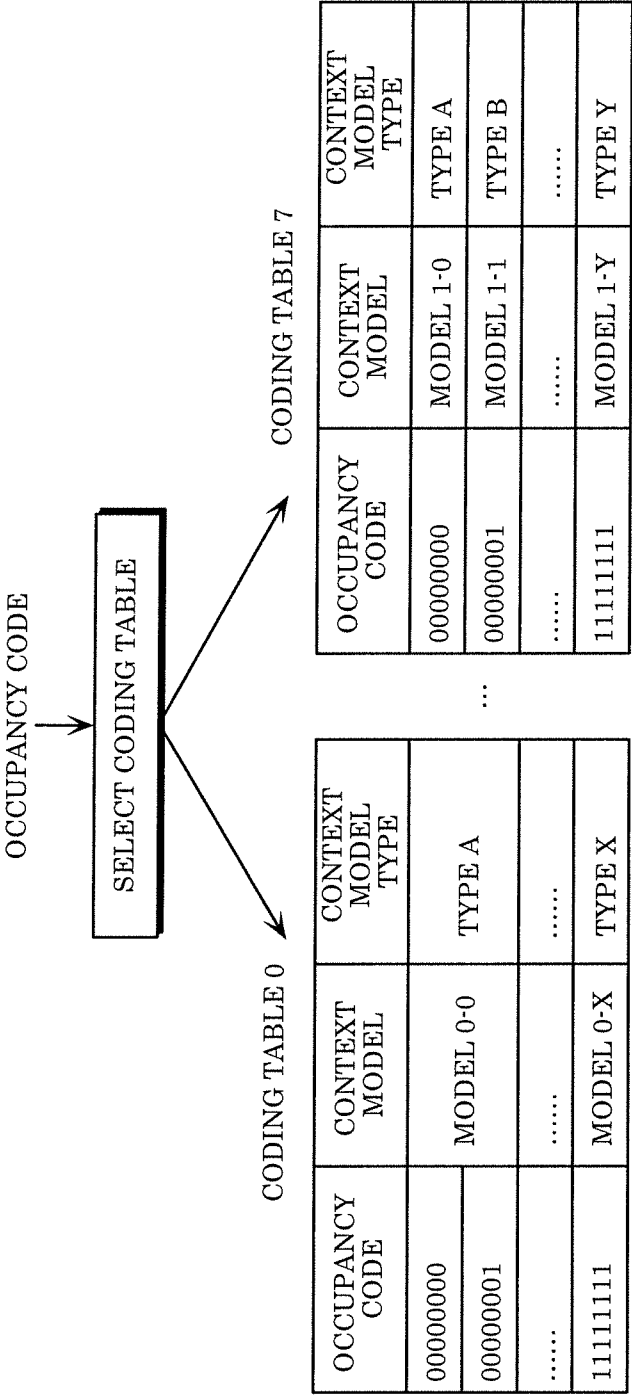


FIG. 146

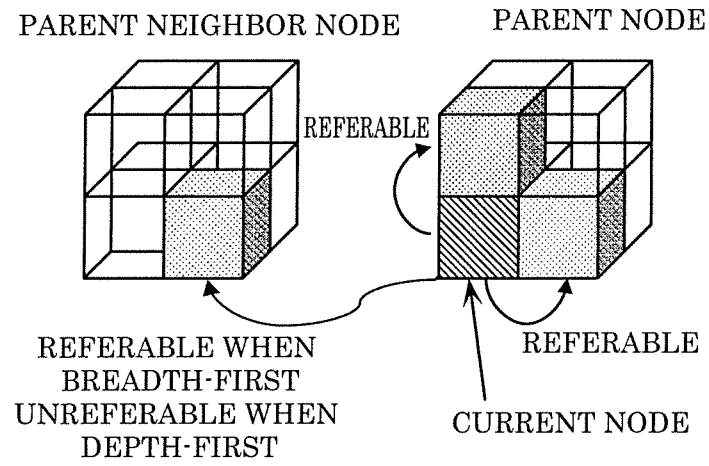


FIG. 147

```
pc_header 0 {
...
octree_scan_order
...
}
```

FIG. 148

```
pc_header 0 {
...
limit_refer_flag
...
}
```

FIG. 149

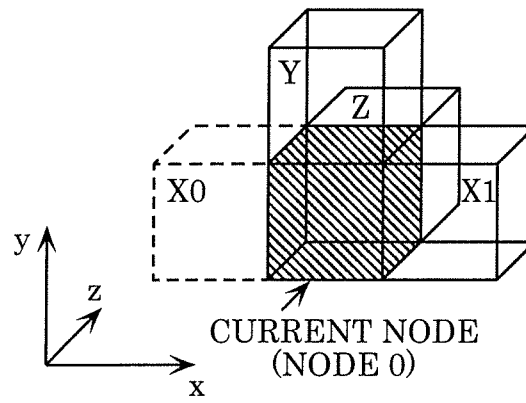


FIG. 150

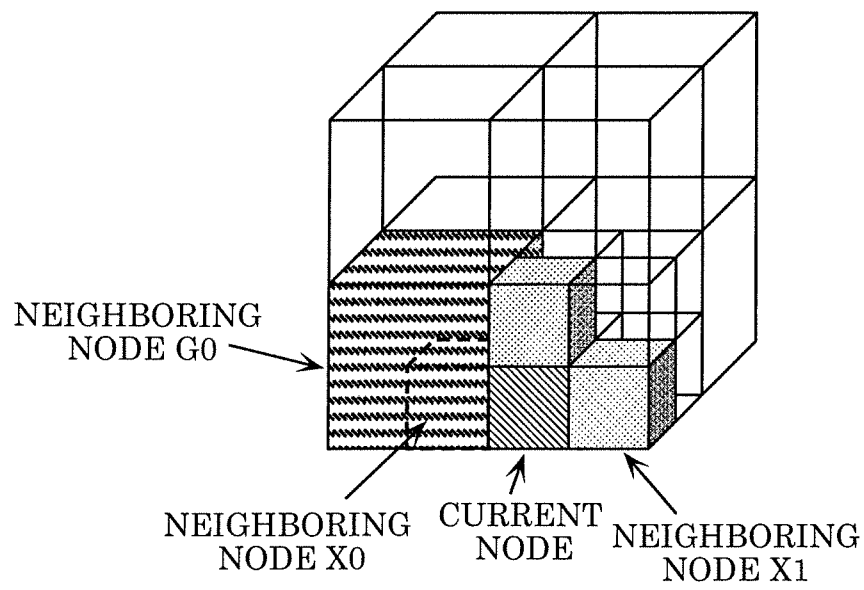


FIG. 151

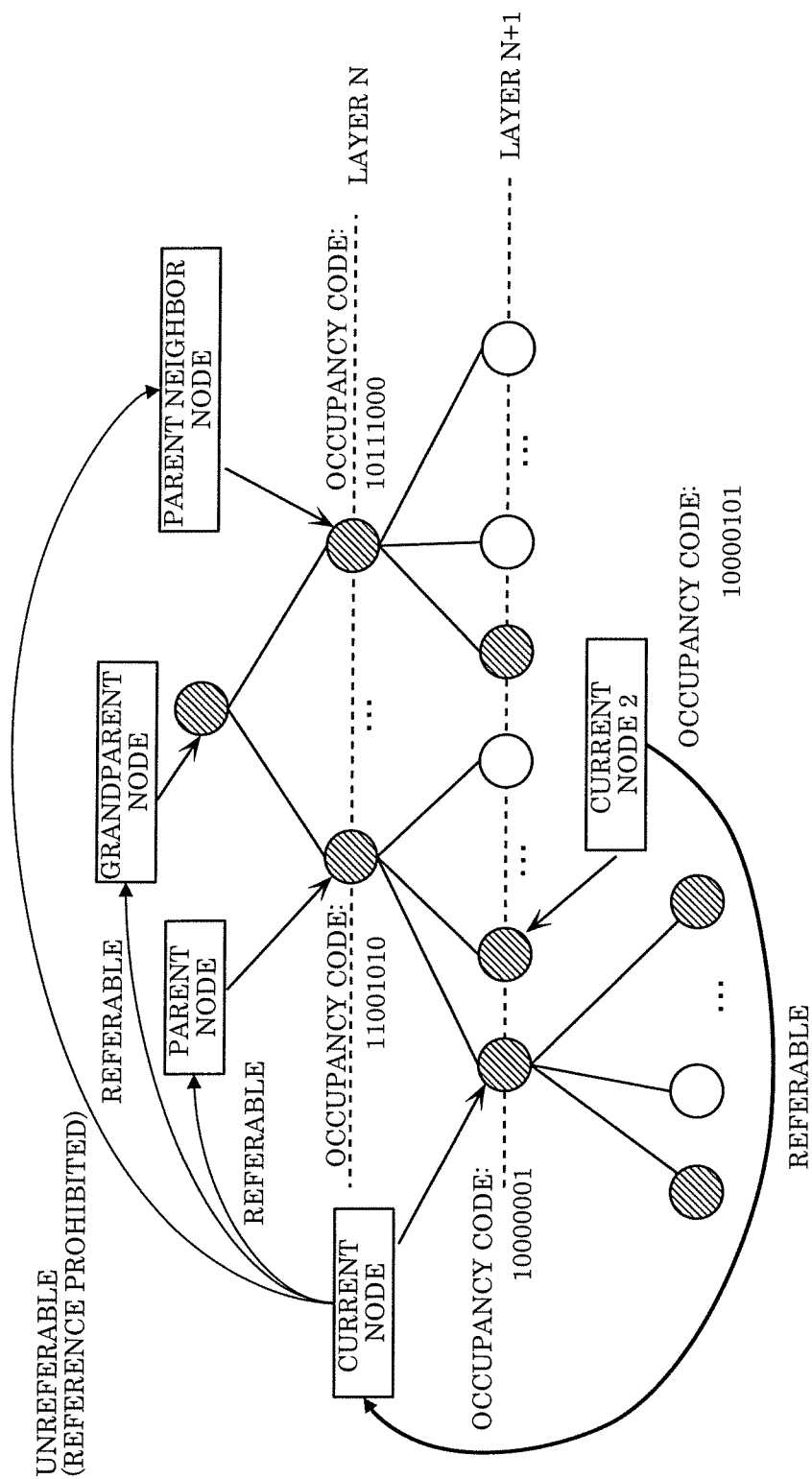


FIG. 152

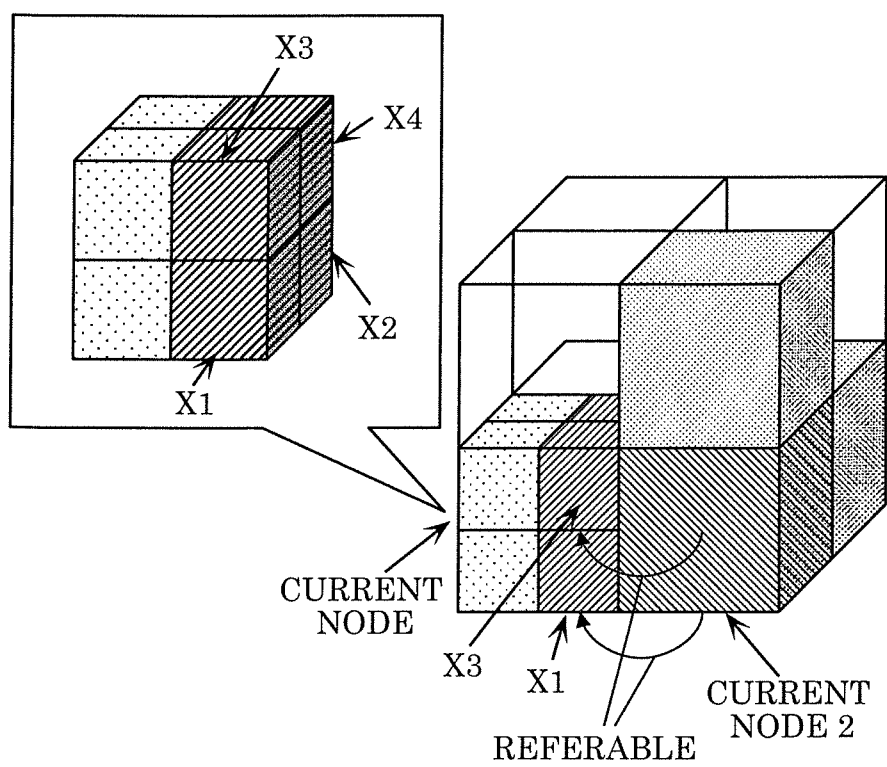


FIG. 153

```
pc_header 0 {  
  ...  
  single_point_per_leaf  
  ...  
}
```

FIG. 154

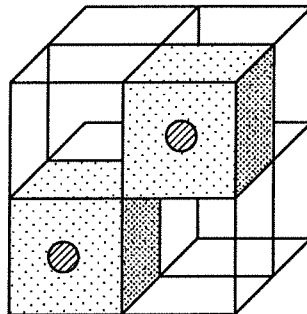


FIG. 155

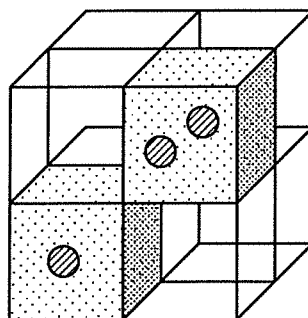


FIG. 156

```
node(depth, index) {  
  ...  
  isleaf  
  if (isleaf) { //leaf  
    if (single_point_per_leaf == 0)  
      num_point_per_leaf  
  }  
  else { //node  
    ...  
  }  
  ...  
}
```

FIG. 157

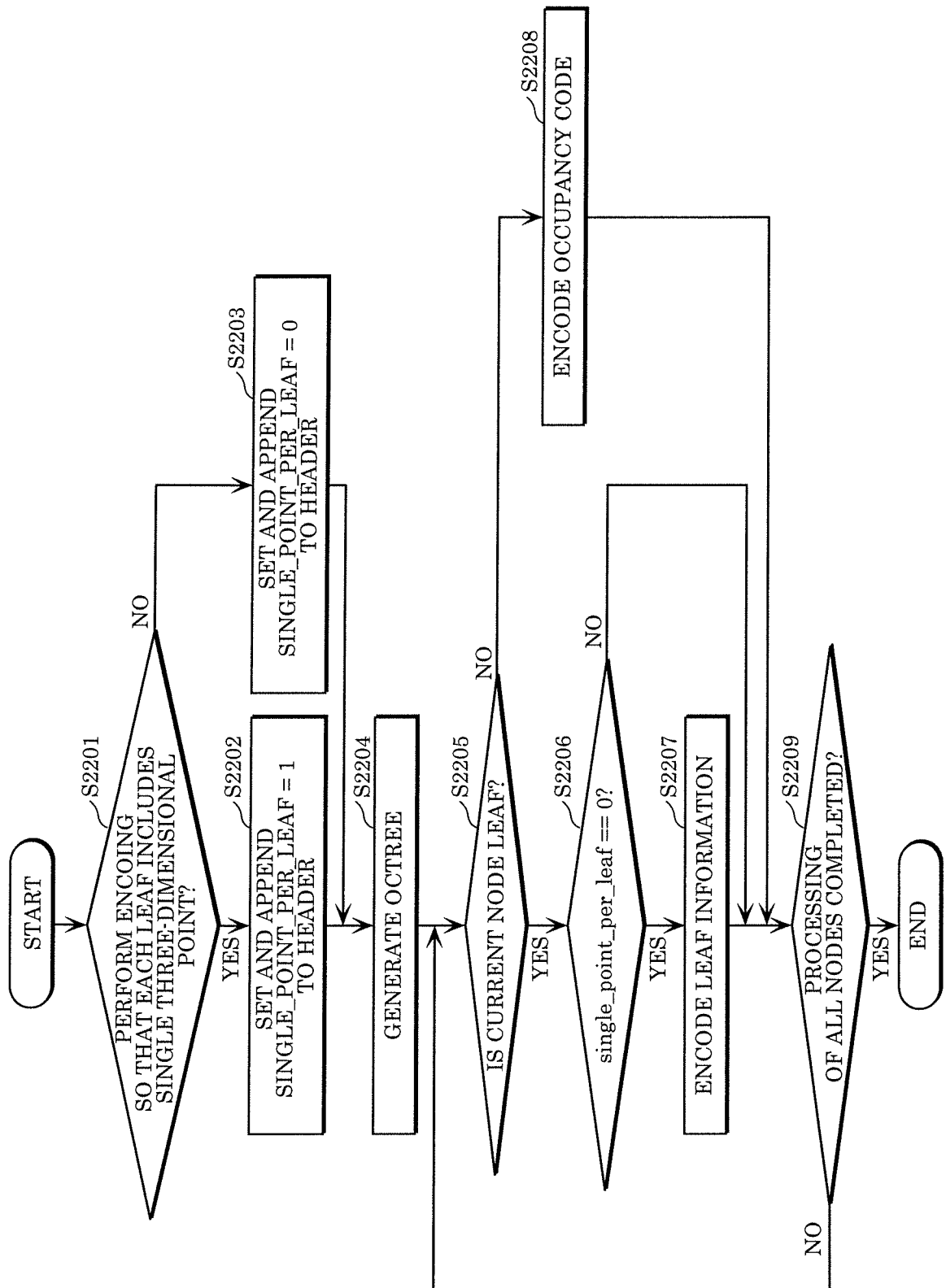


FIG. 158

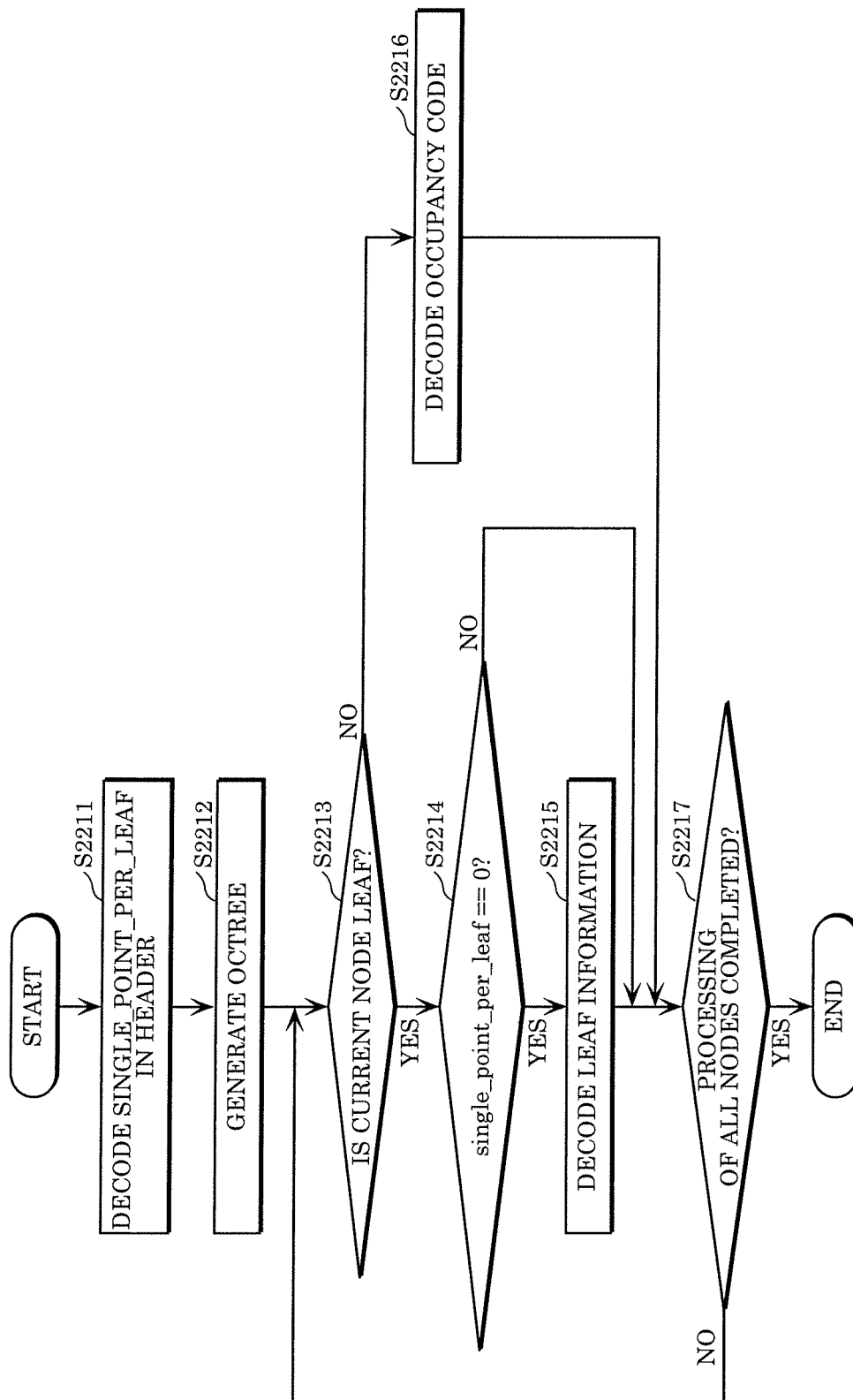


FIG. 159

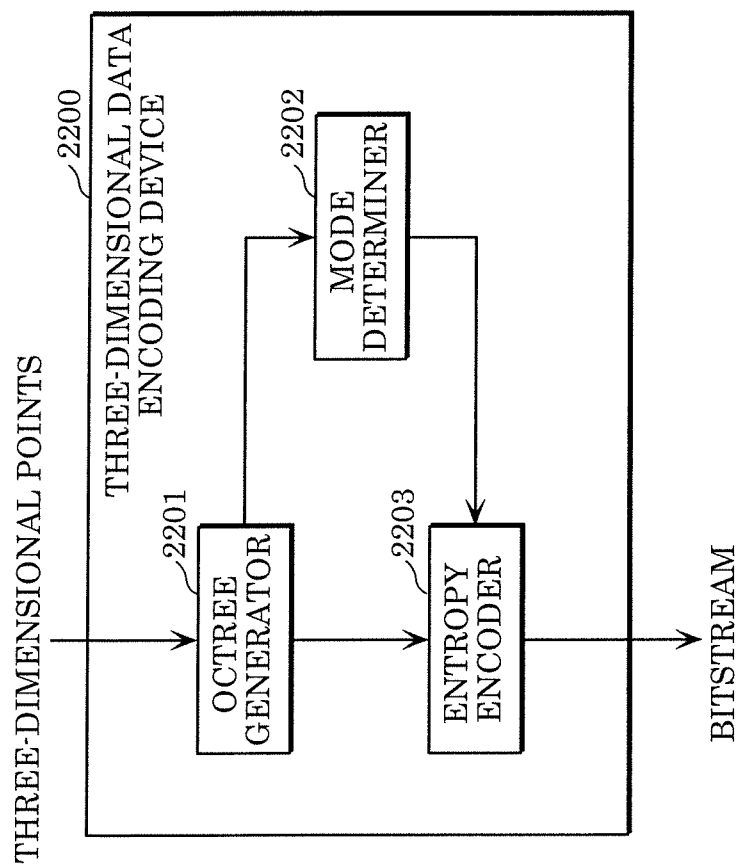


FIG. 160

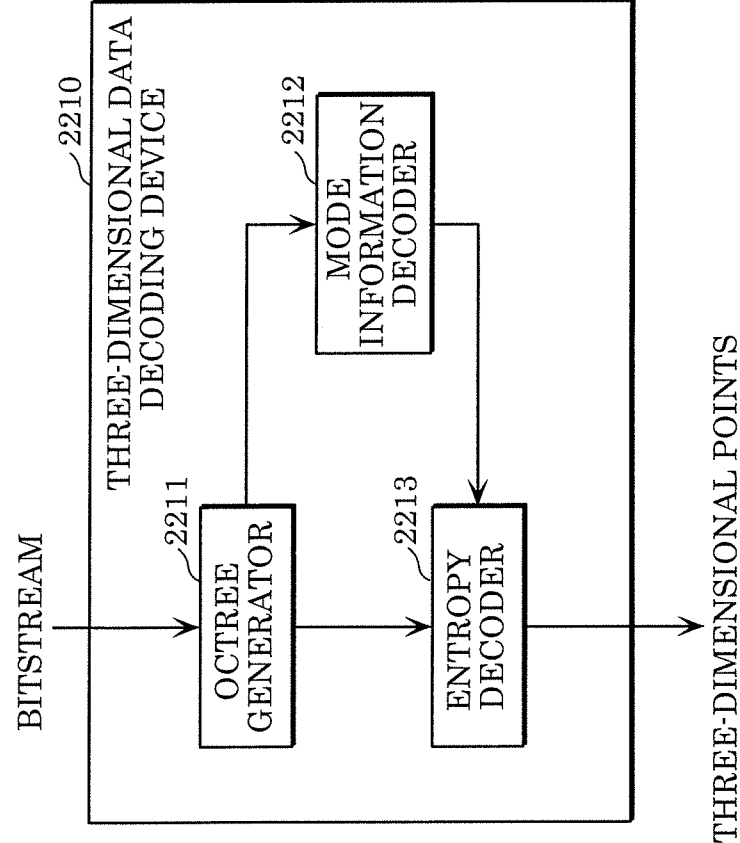


FIG. 161

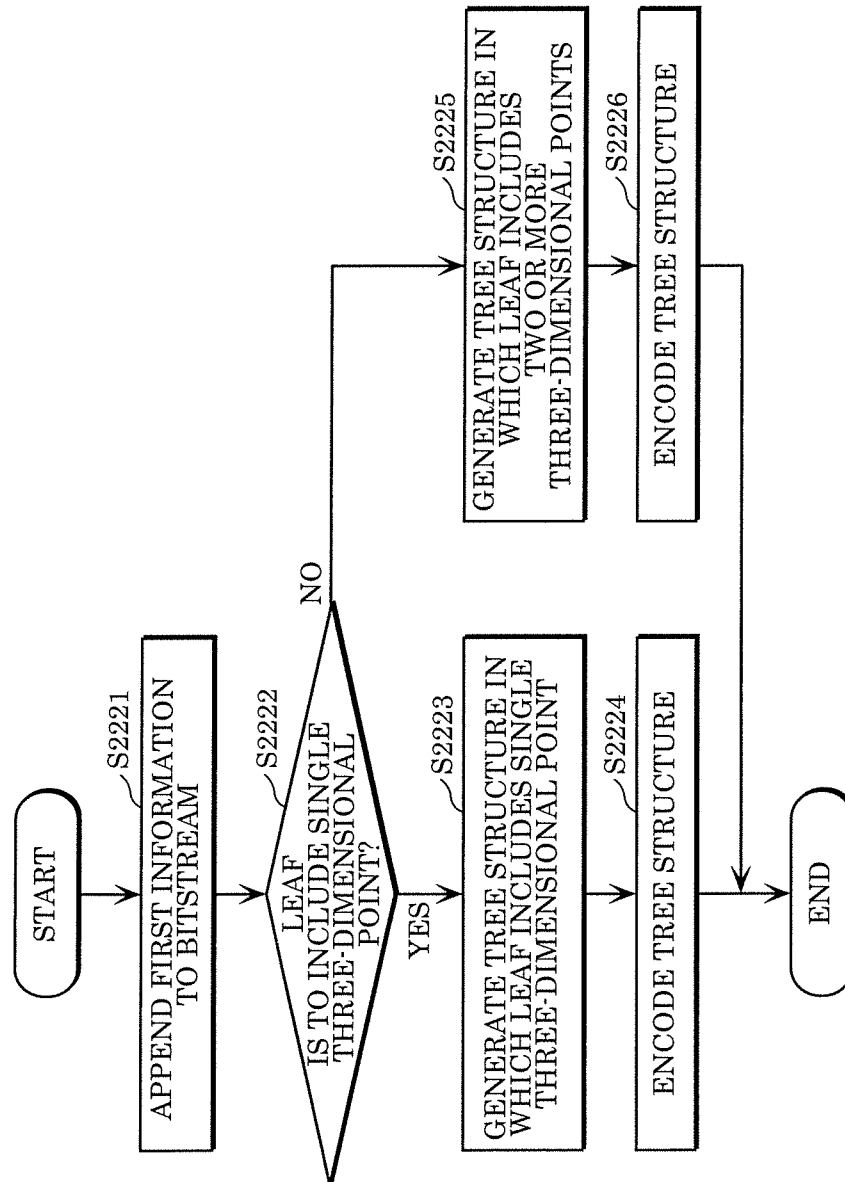


FIG. 162

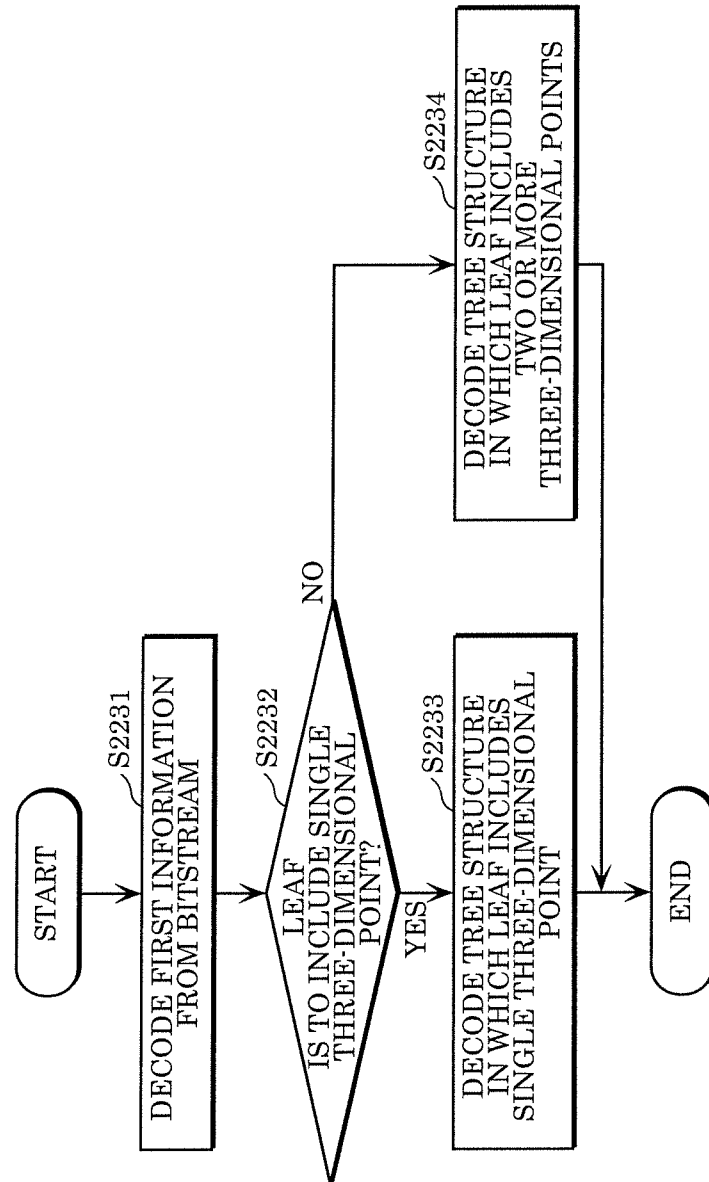


FIG. 163

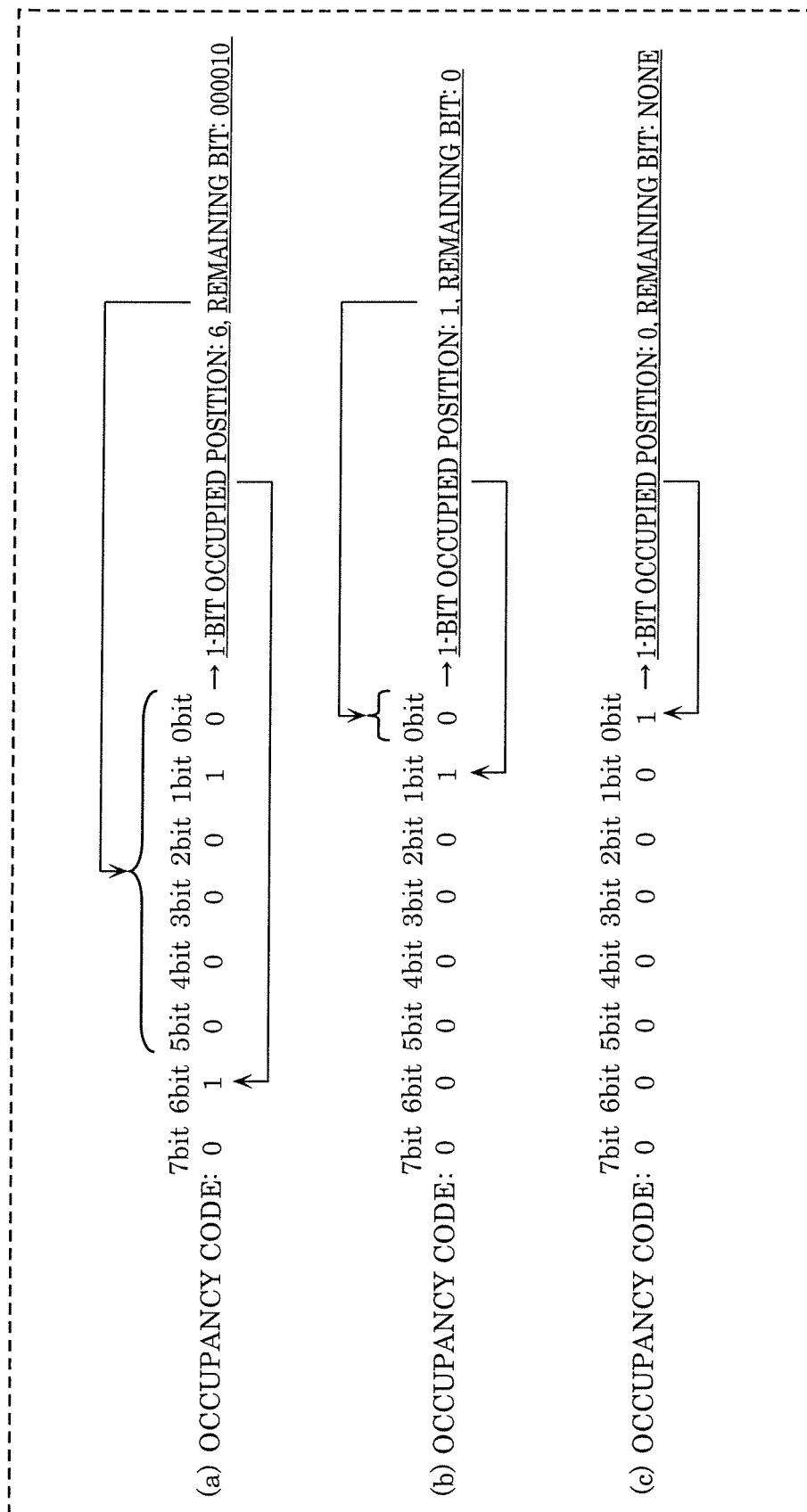


FIG. 164

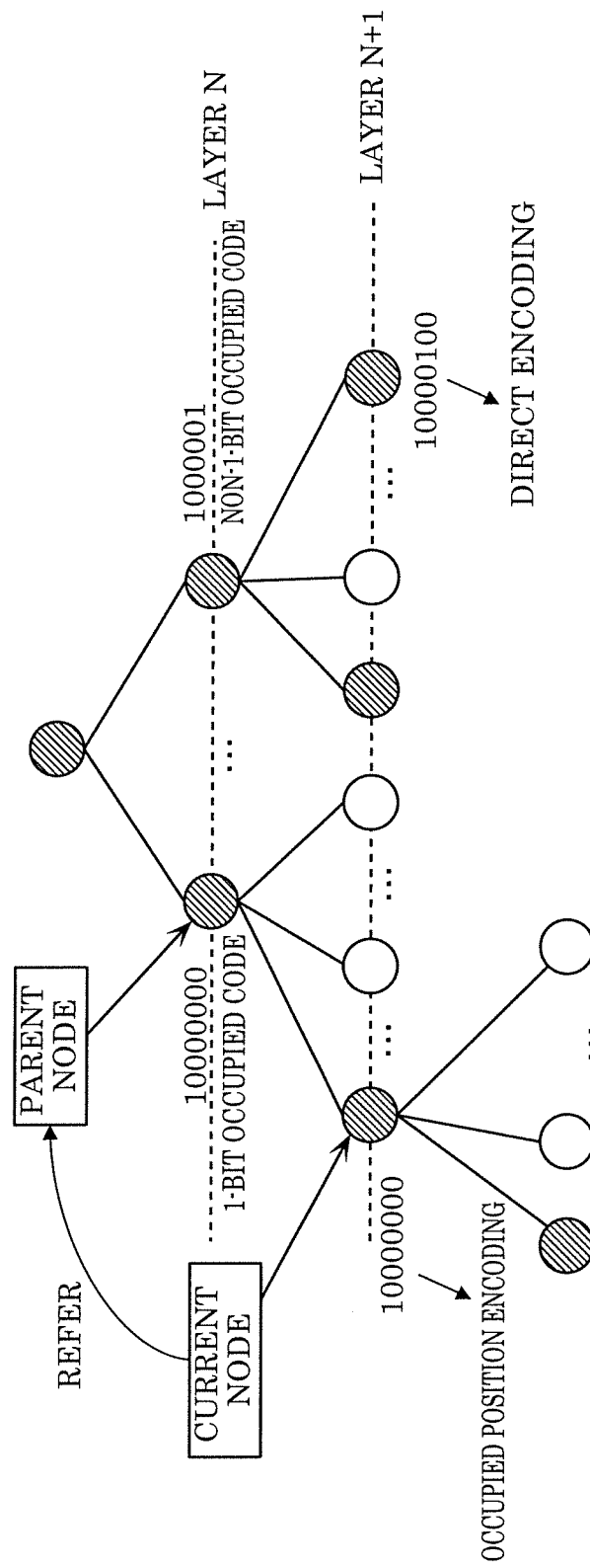


FIG. 165

```

node(depth, index) {
    . . .
    if (parent_occupancy_code == 1bit_occupied_code) {
        1bit_occupied_position
        if (1bit_occupied_position != 0) {
            remaining_bit
        }
    }
    Else {
        occupancy_code
    }
    . . .
}

```

FIG. 166

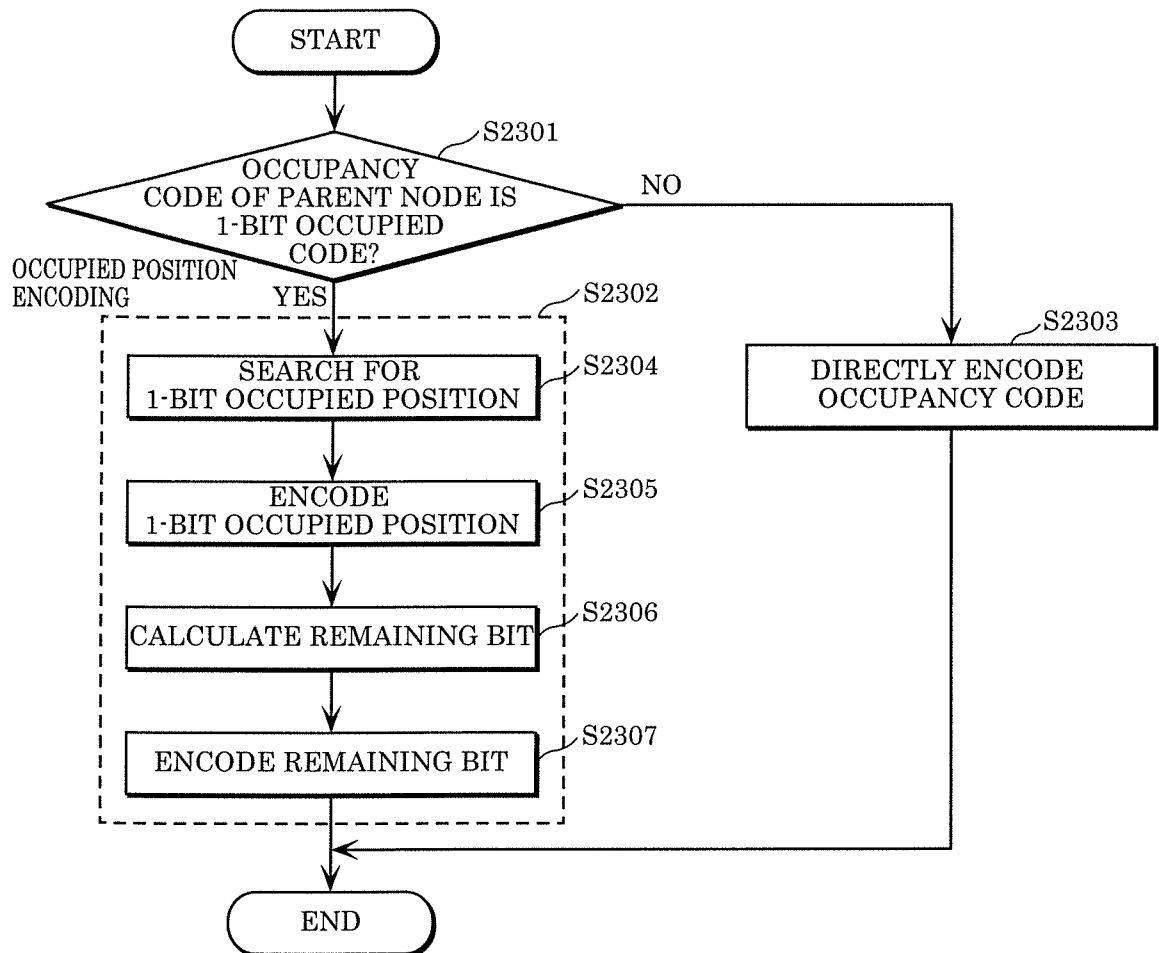


FIG. 167

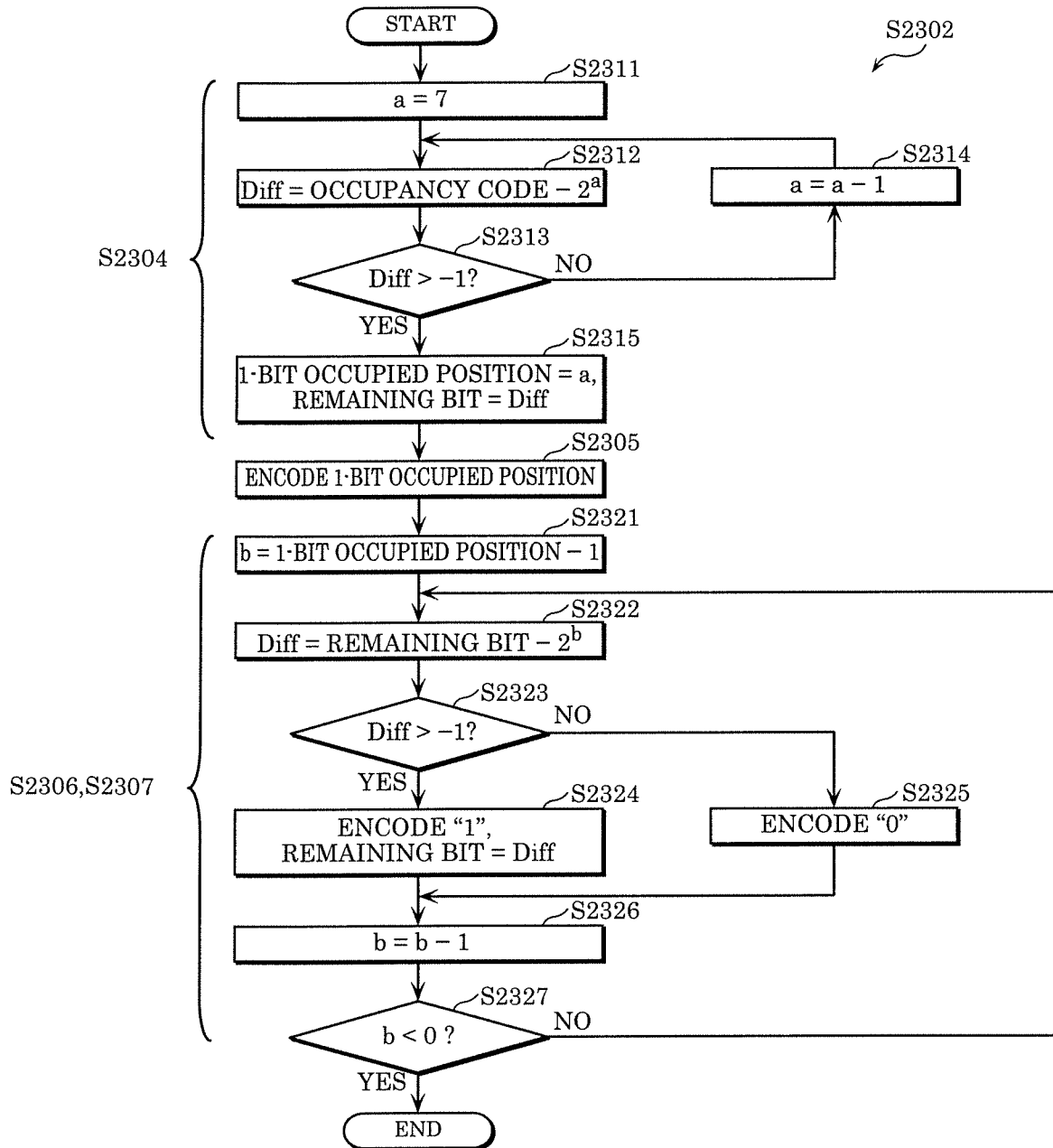


FIG. 168

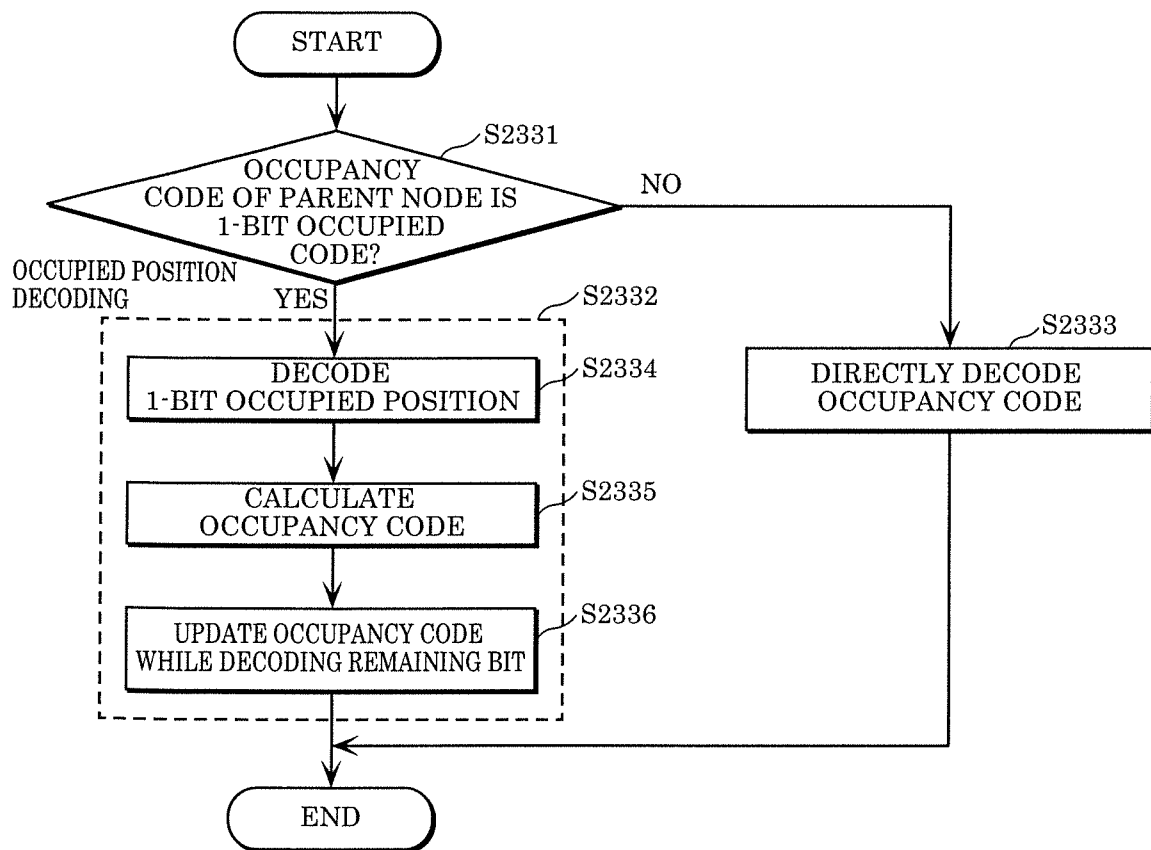


FIG. 169

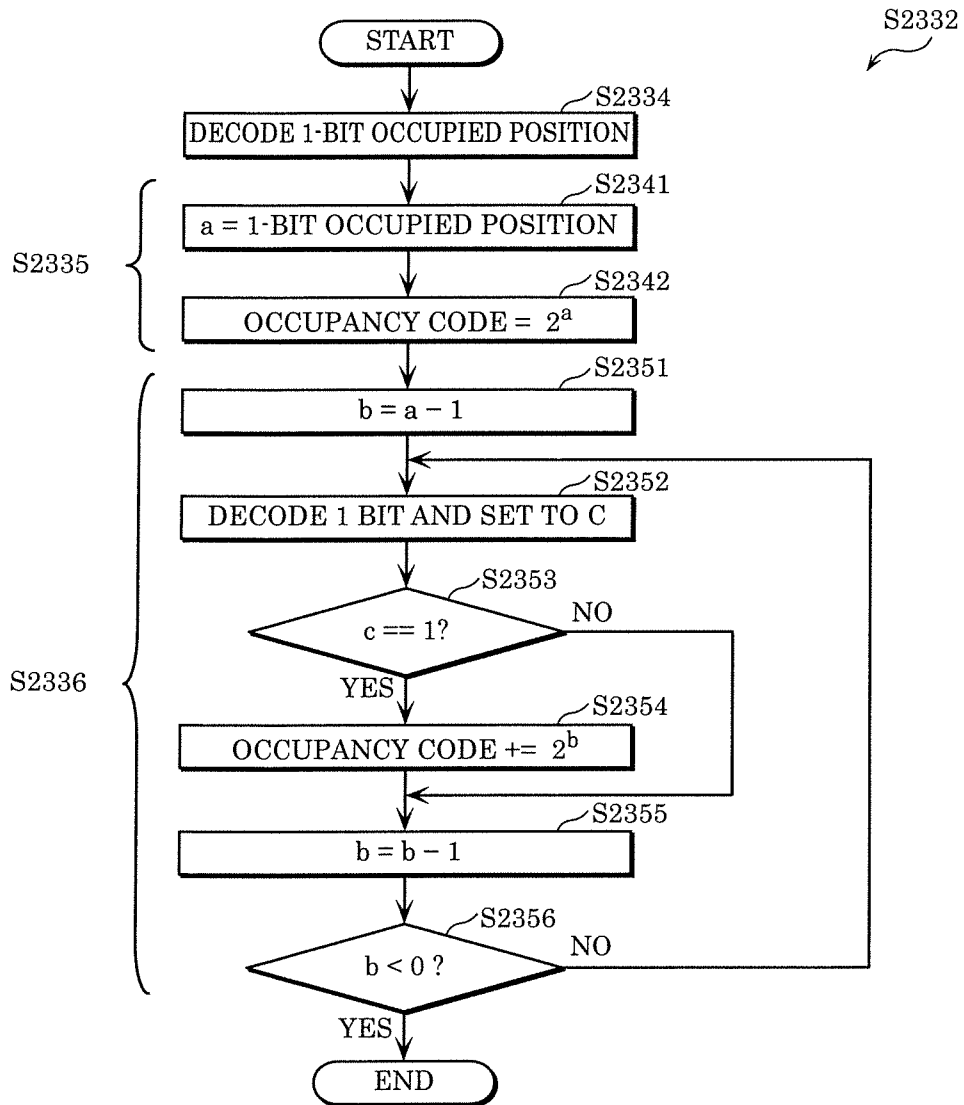


FIG. 170

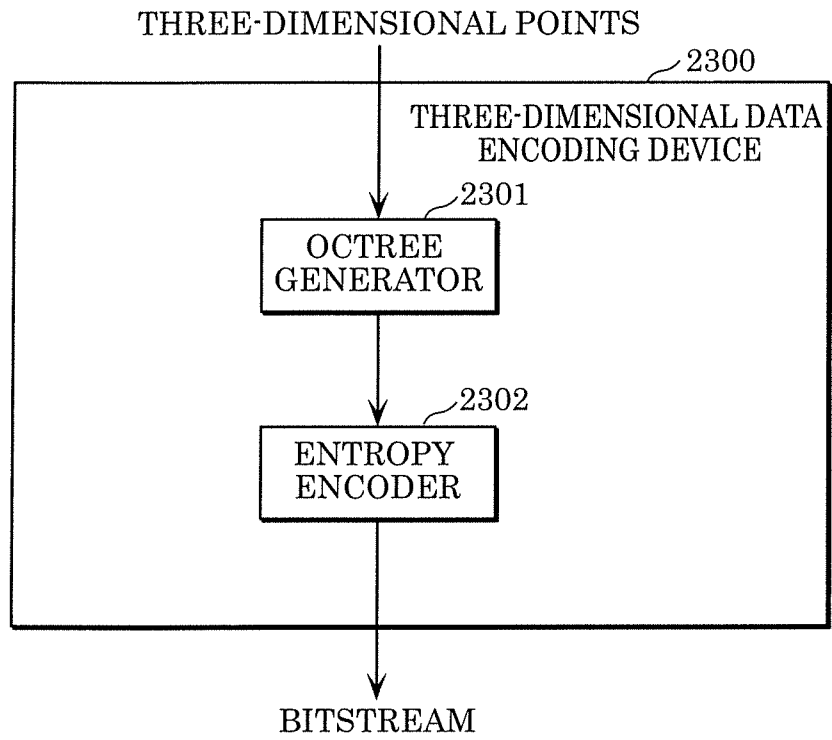


FIG. 171

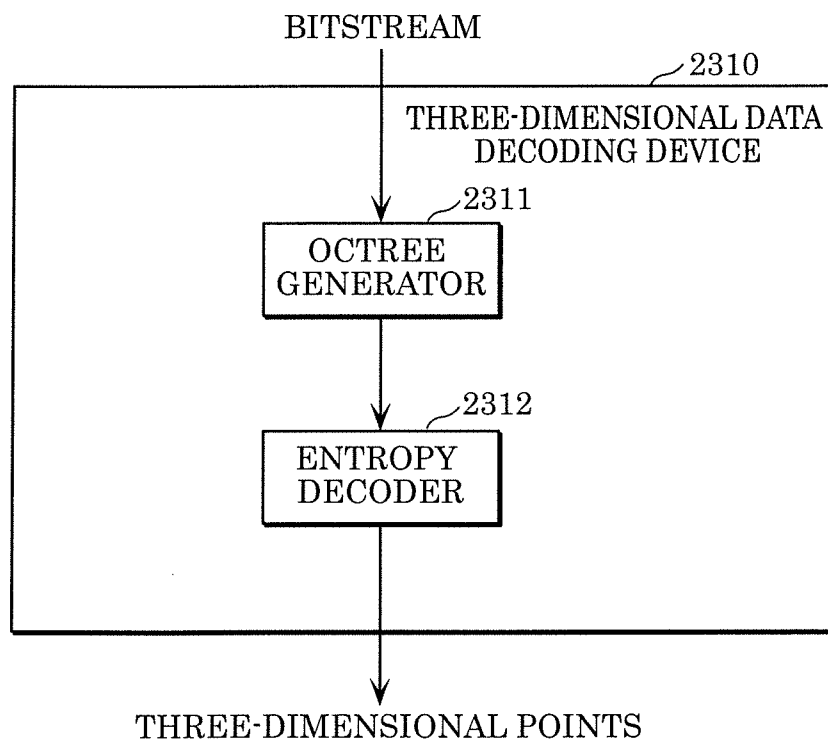


FIG. 172

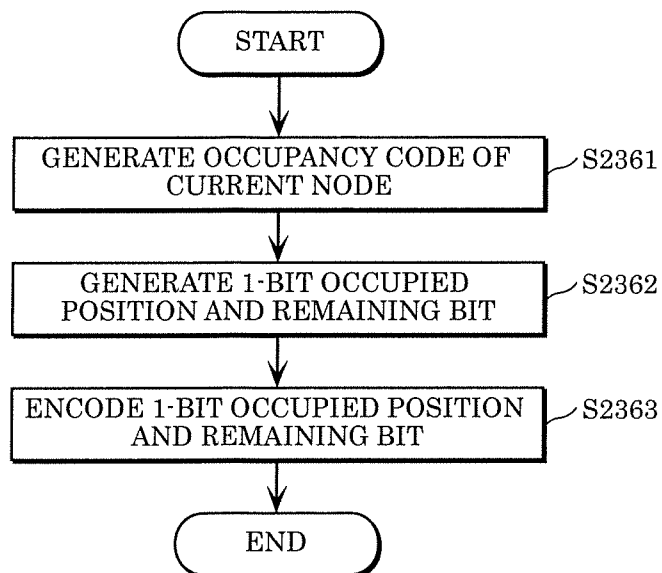


FIG. 173

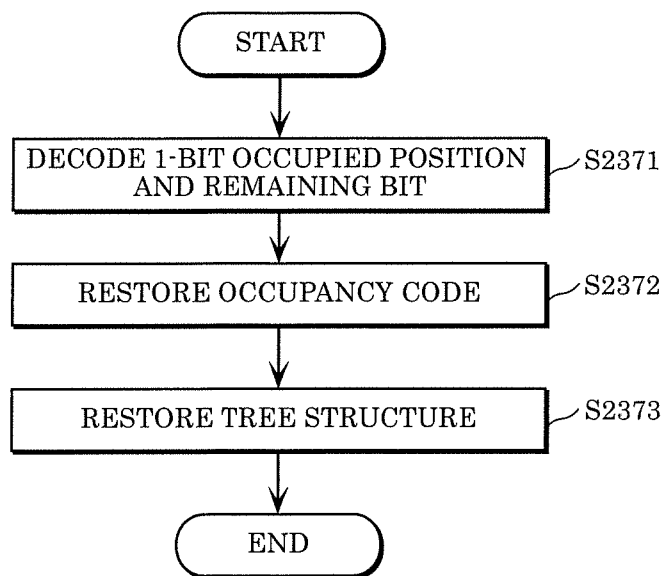


FIG. 174

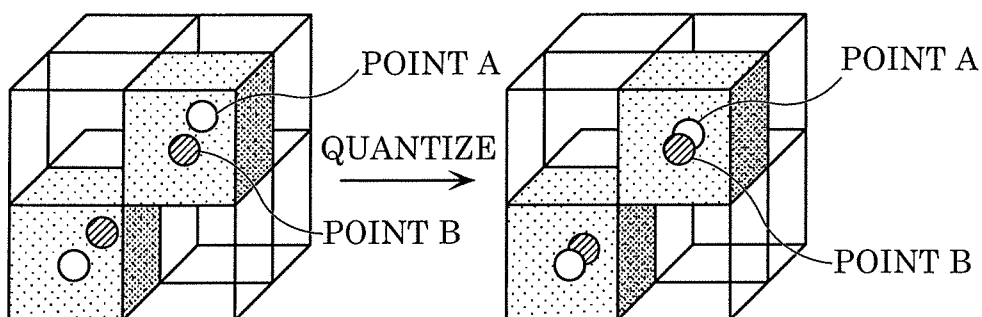


FIG. 175

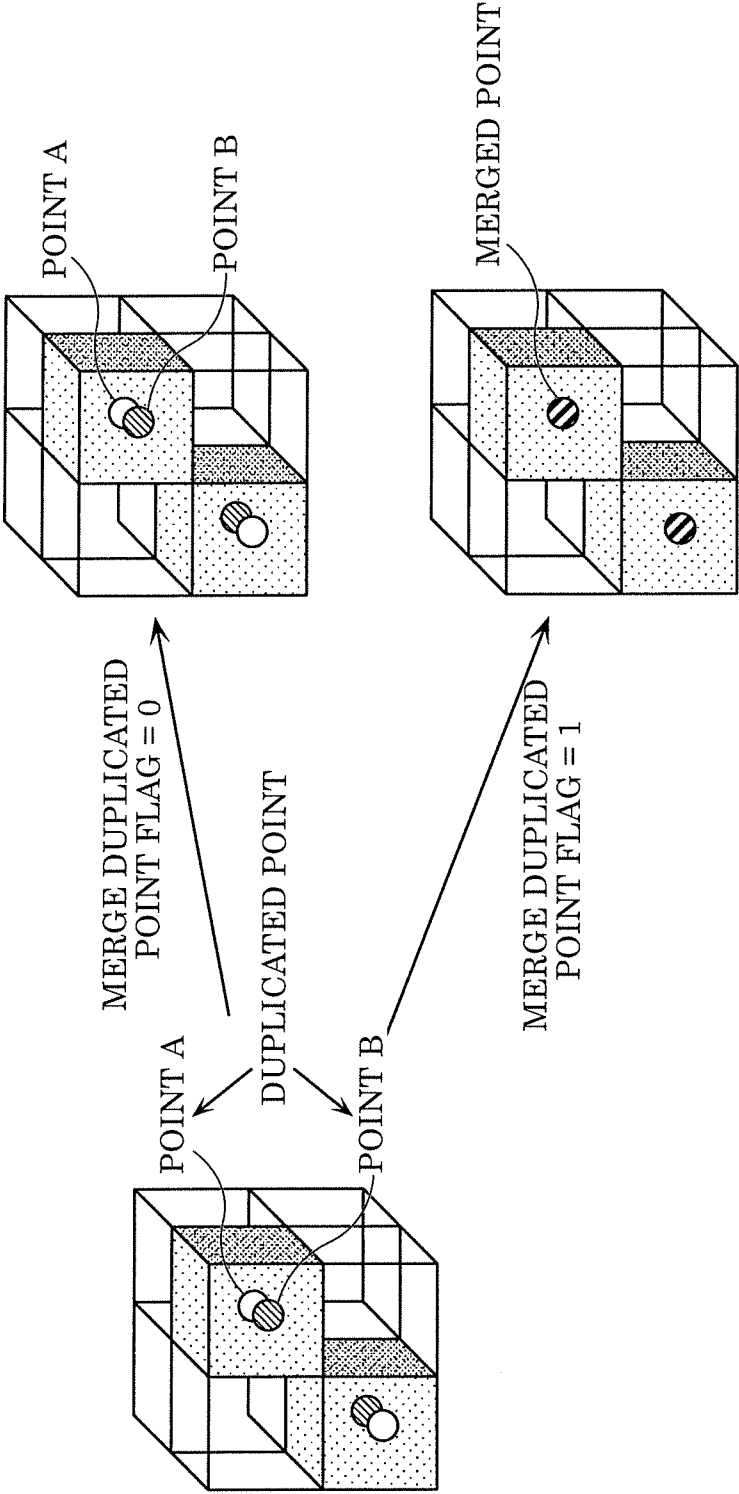


FIG. 176

```
pc_header() {  
  . . .  
  MergeDuplicatedPointFlag  
  . . .  
}
```

FIG. 177

```
node(depth, index) {  
  . . .  
  isleaf  
  if (isleaf) { //leaf  
    if (MergeDuplicatedPointFlag == 0)  
      num_point_per_leaf  
  }  
  else {      //node  
    . . .  
  }  
  . . .  
}
```

FIG. 178

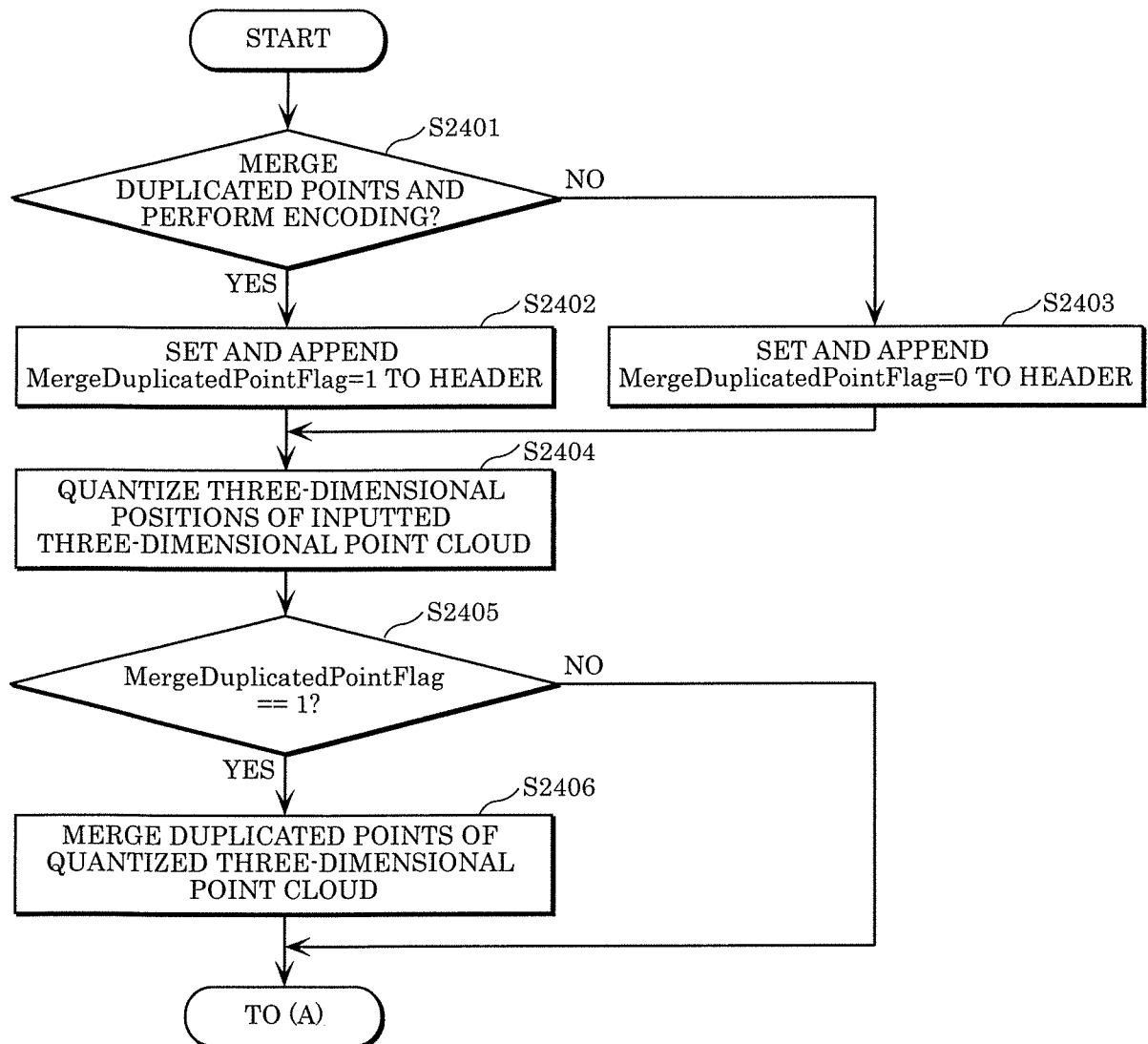


FIG. 179

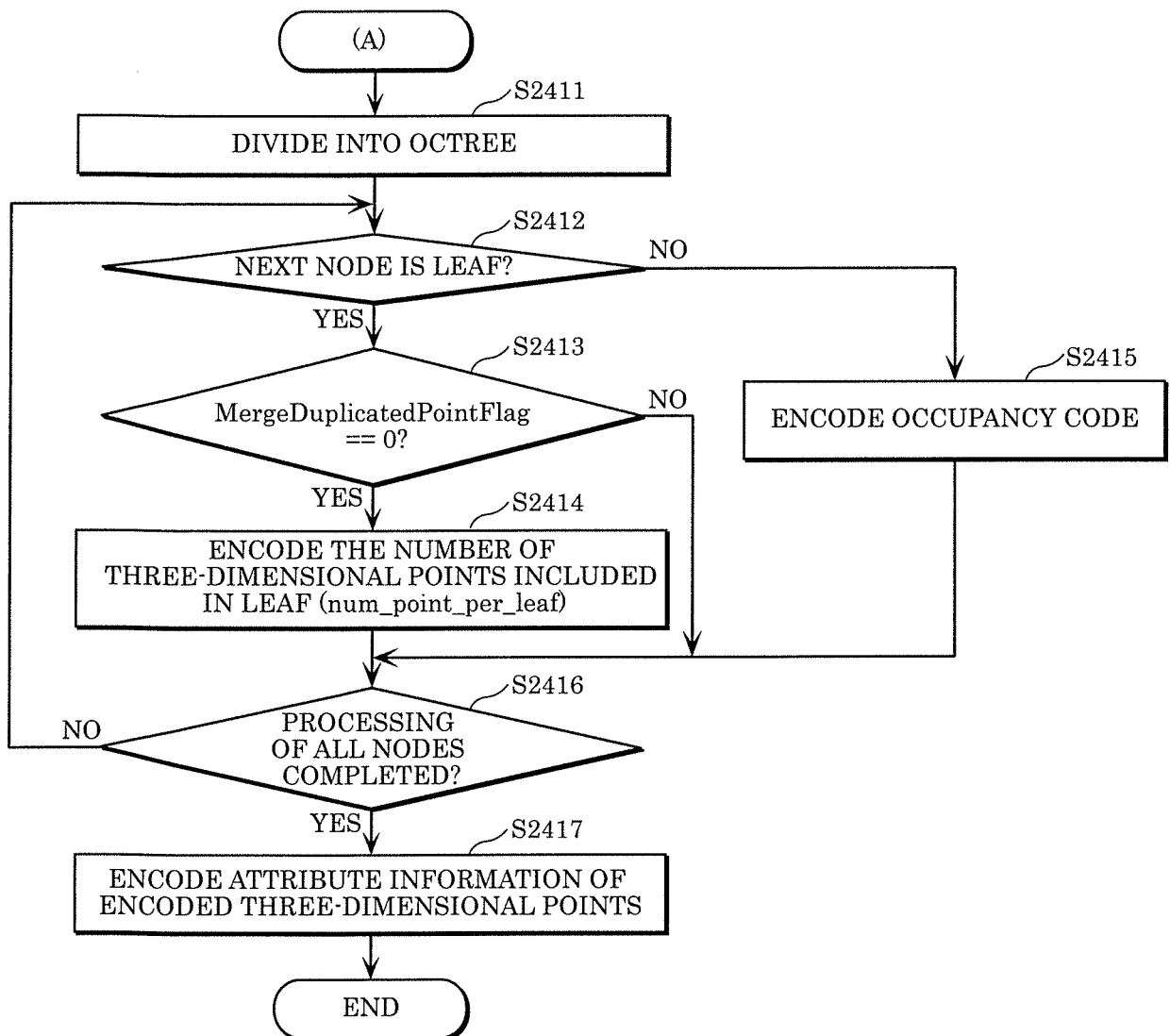


FIG. 180

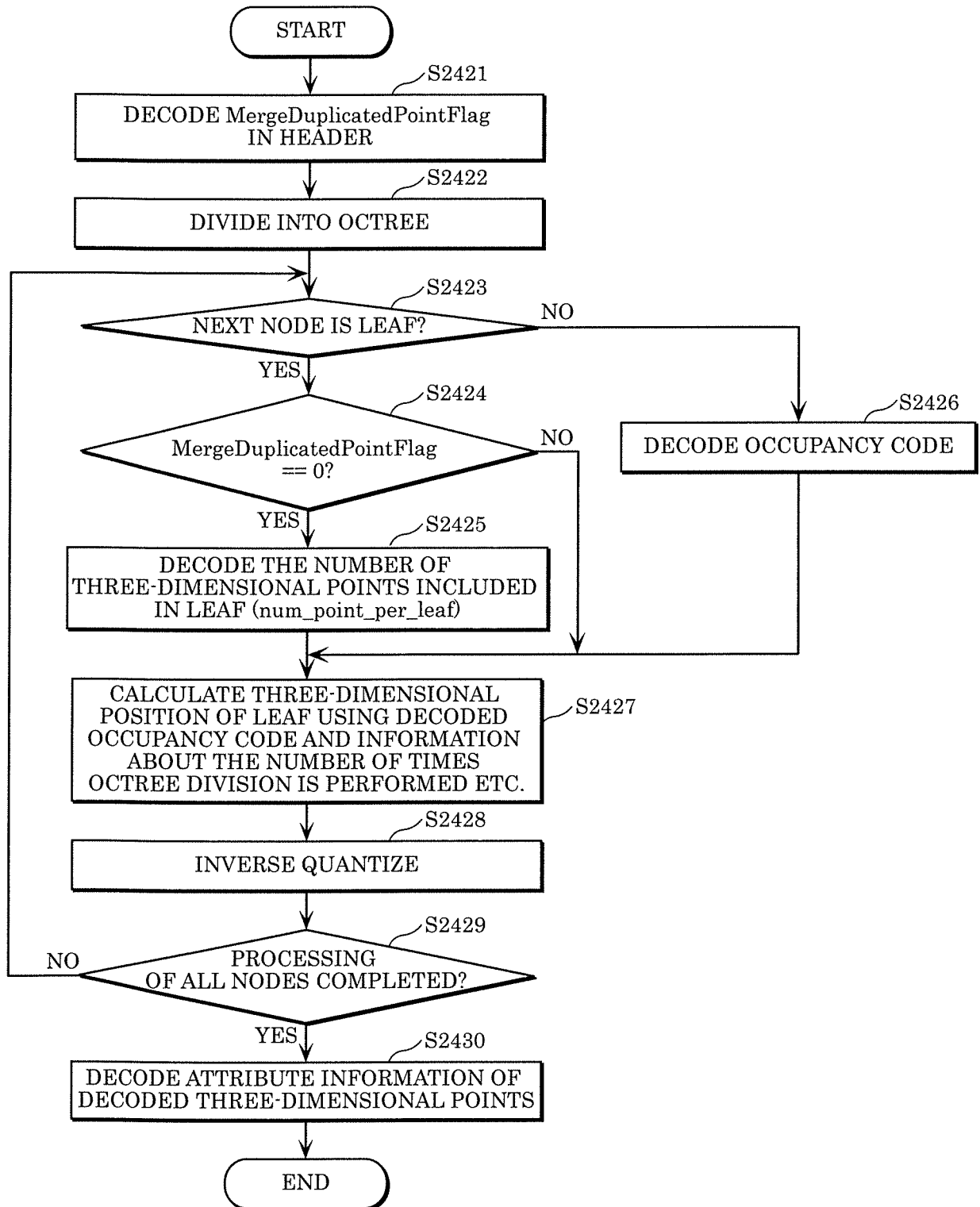


FIG. 181
THREE-DIMENSIONAL POINTS

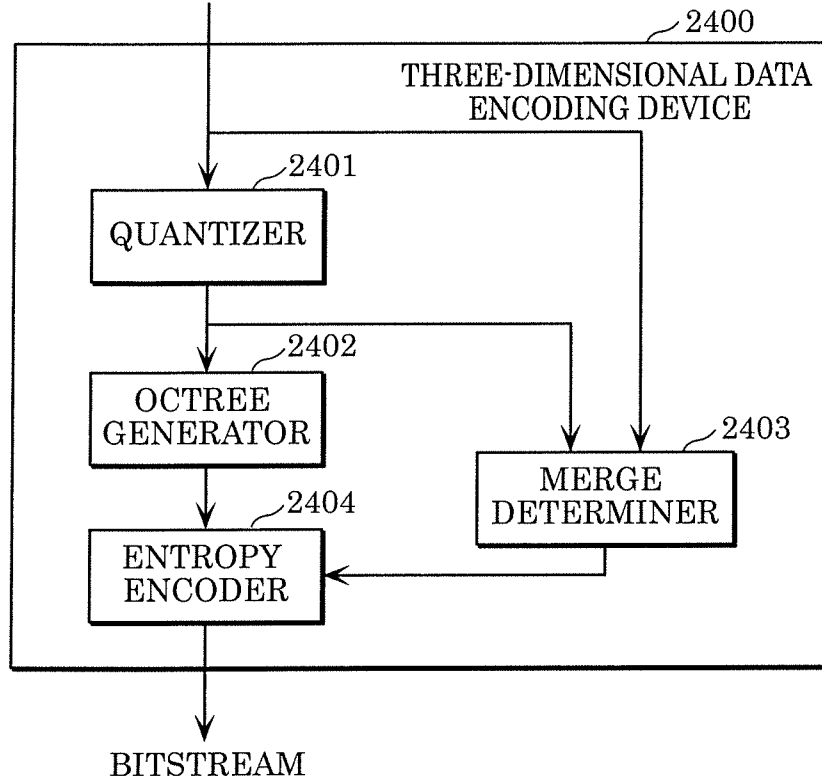


FIG. 182
BITSTREAM

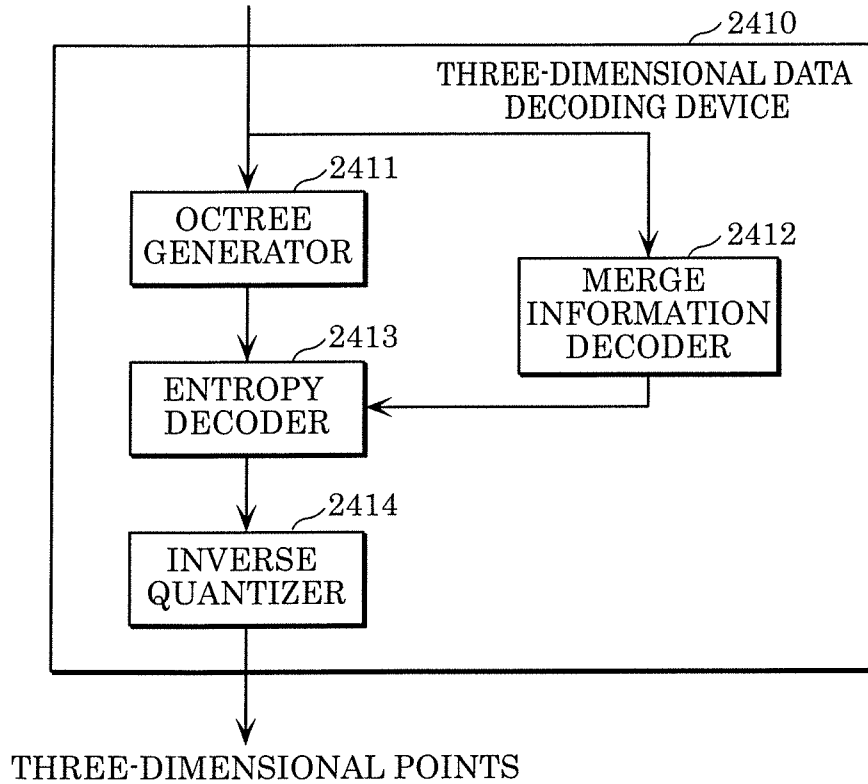


FIG. 183

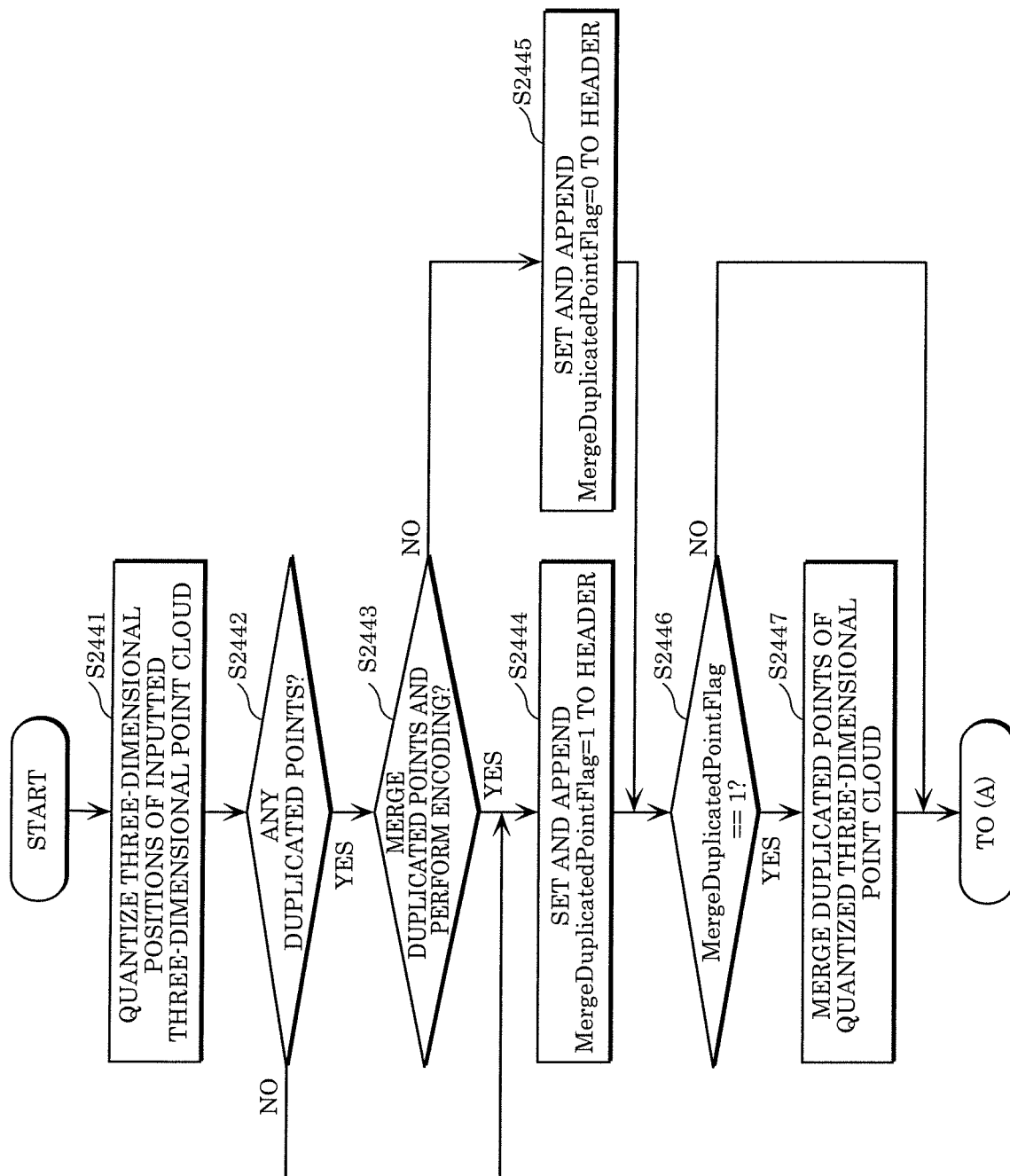


FIG. 184

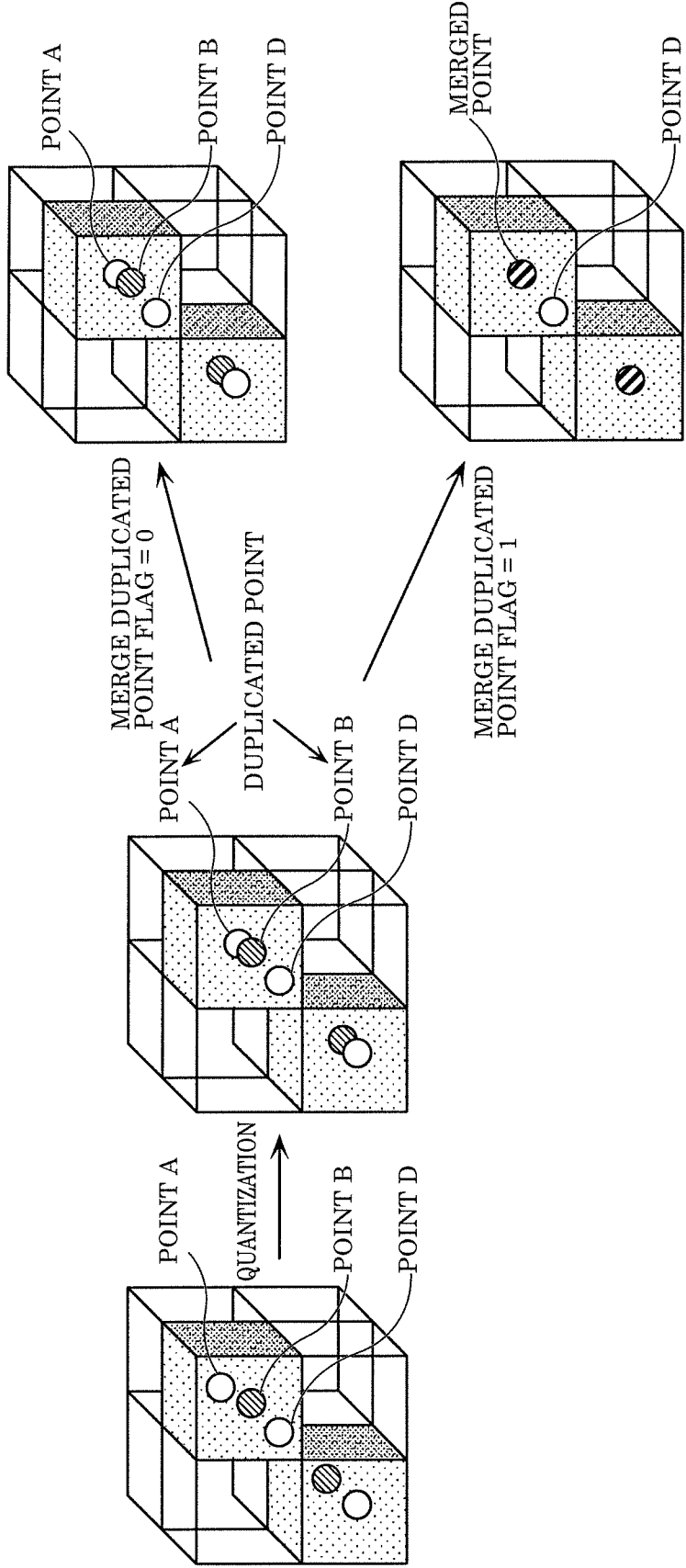


FIG. 185

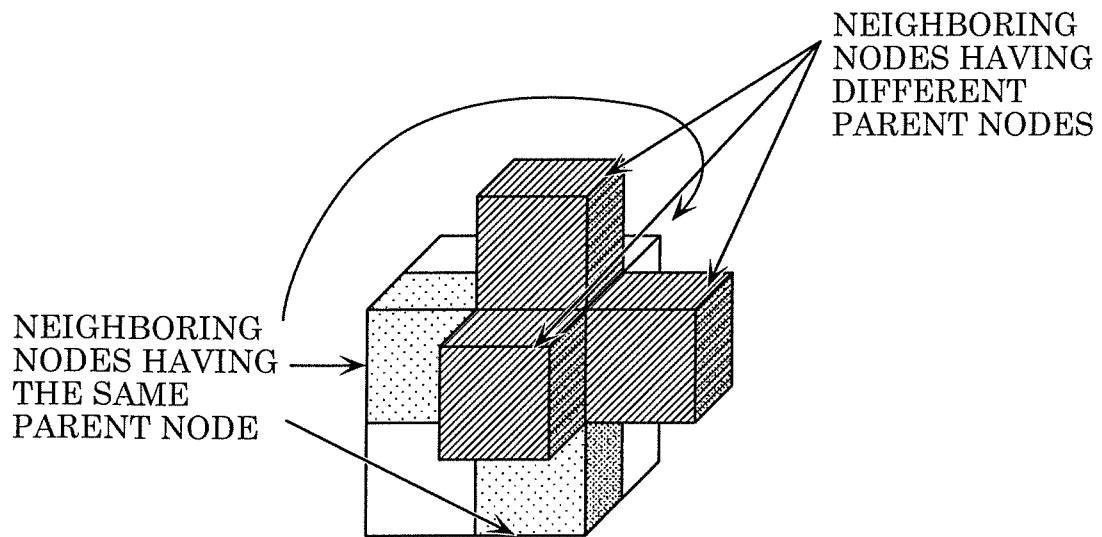


FIG. 186

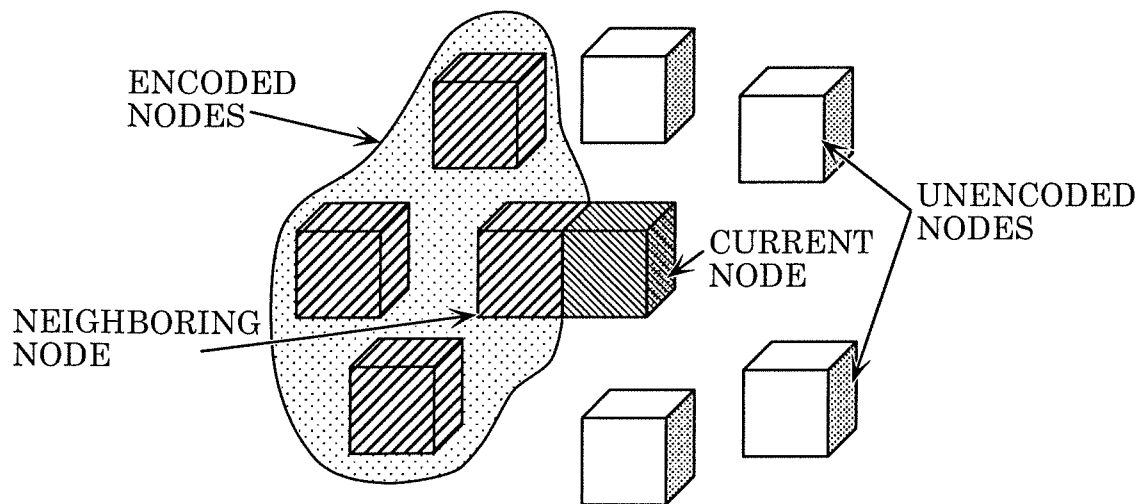


FIG. 187

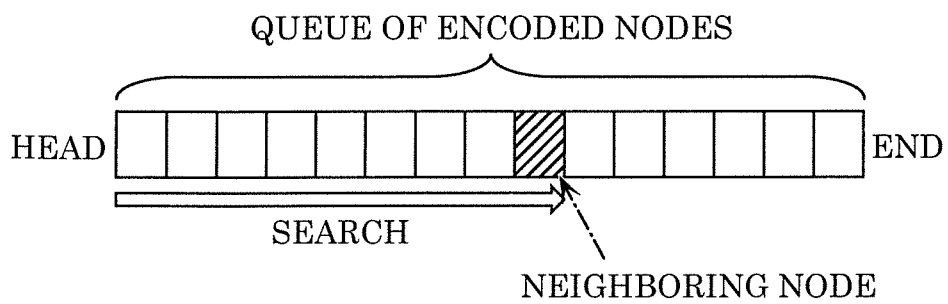


FIG. 188

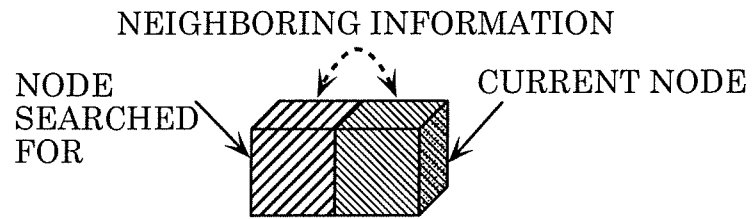


FIG. 189

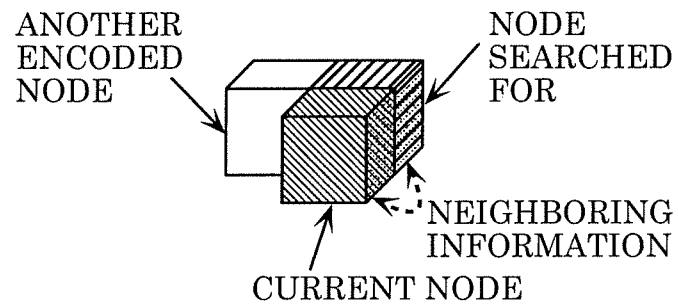


FIG. 190

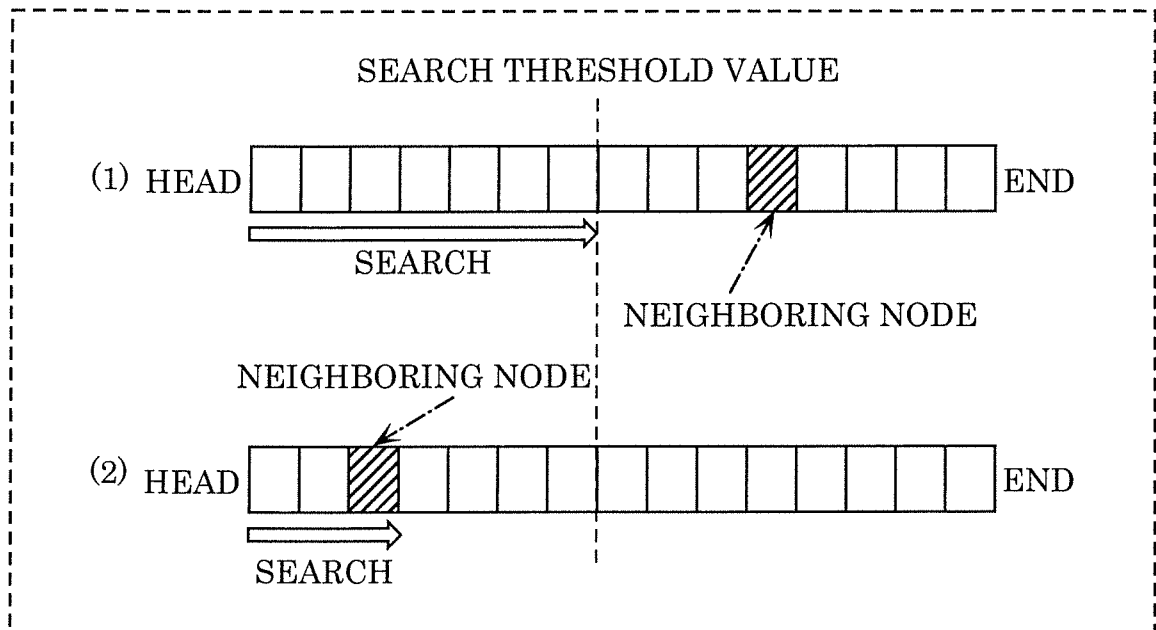


FIG. 191

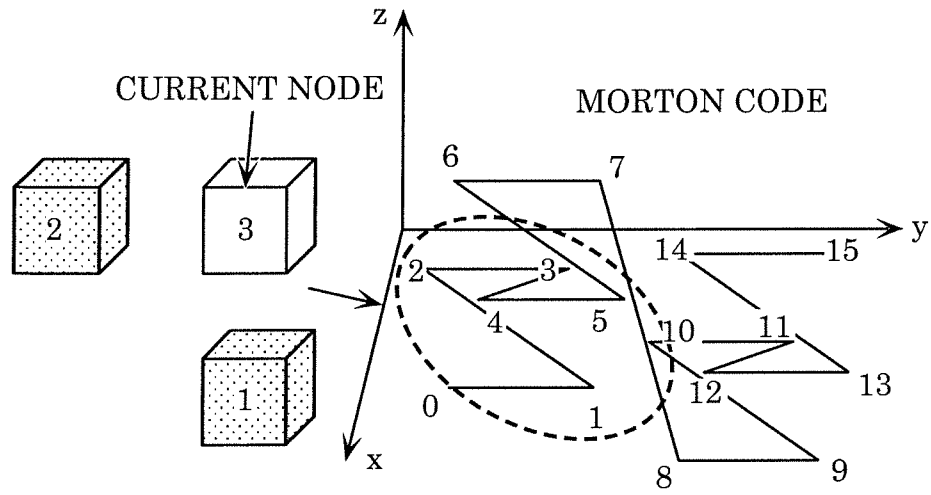


FIG. 192

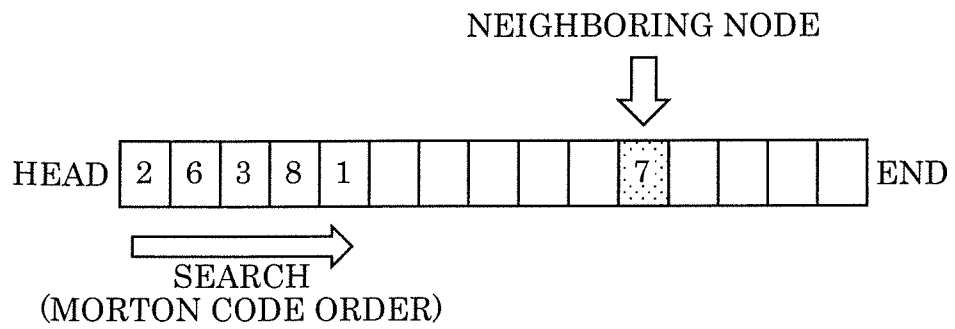


FIG. 193

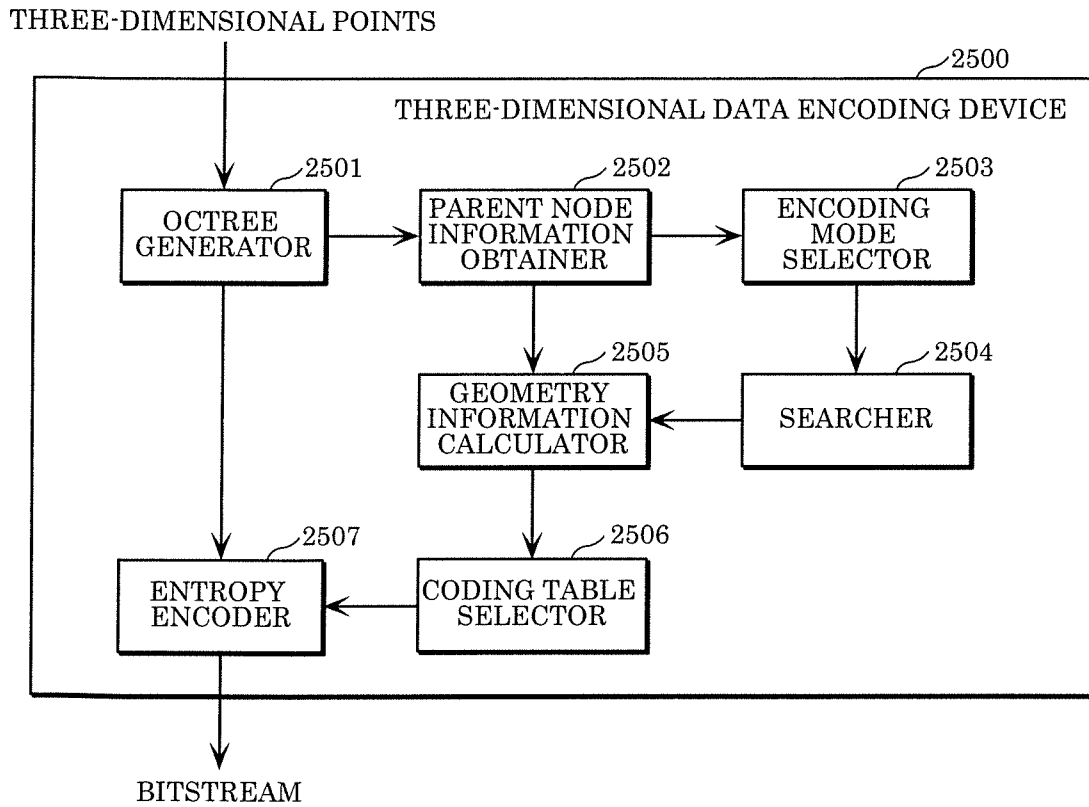


FIG. 194

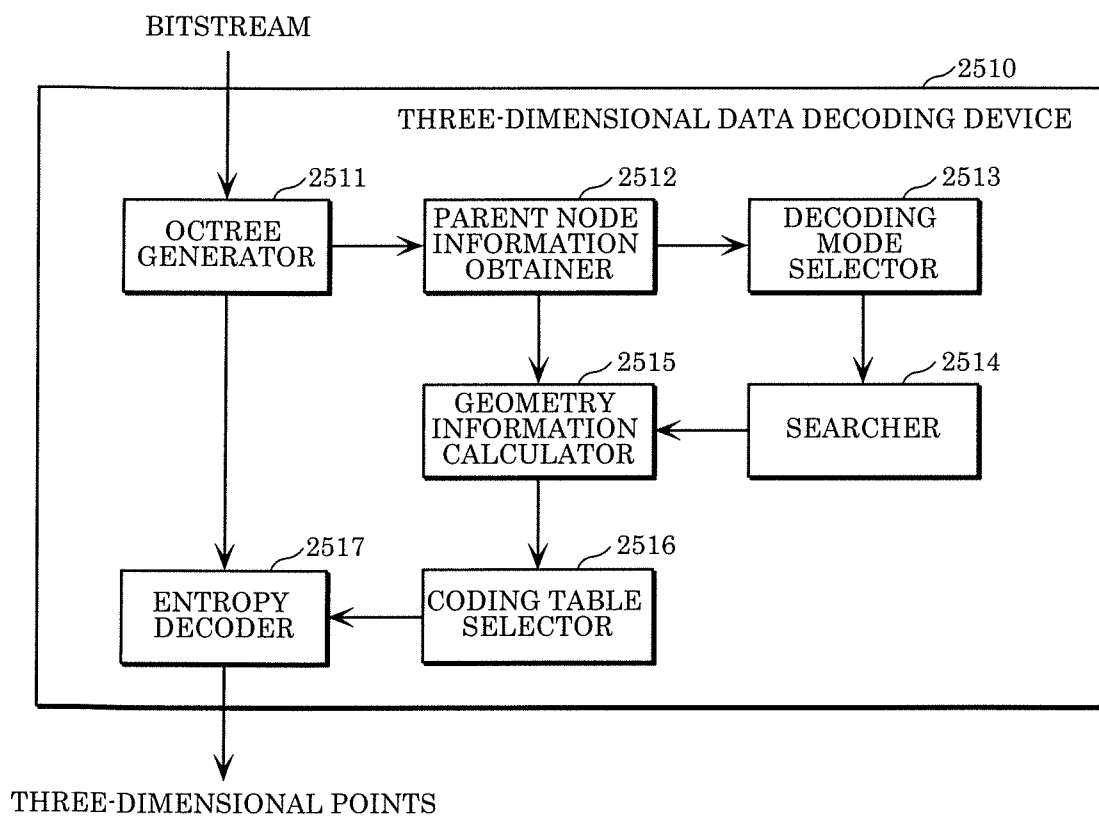


FIG. 195

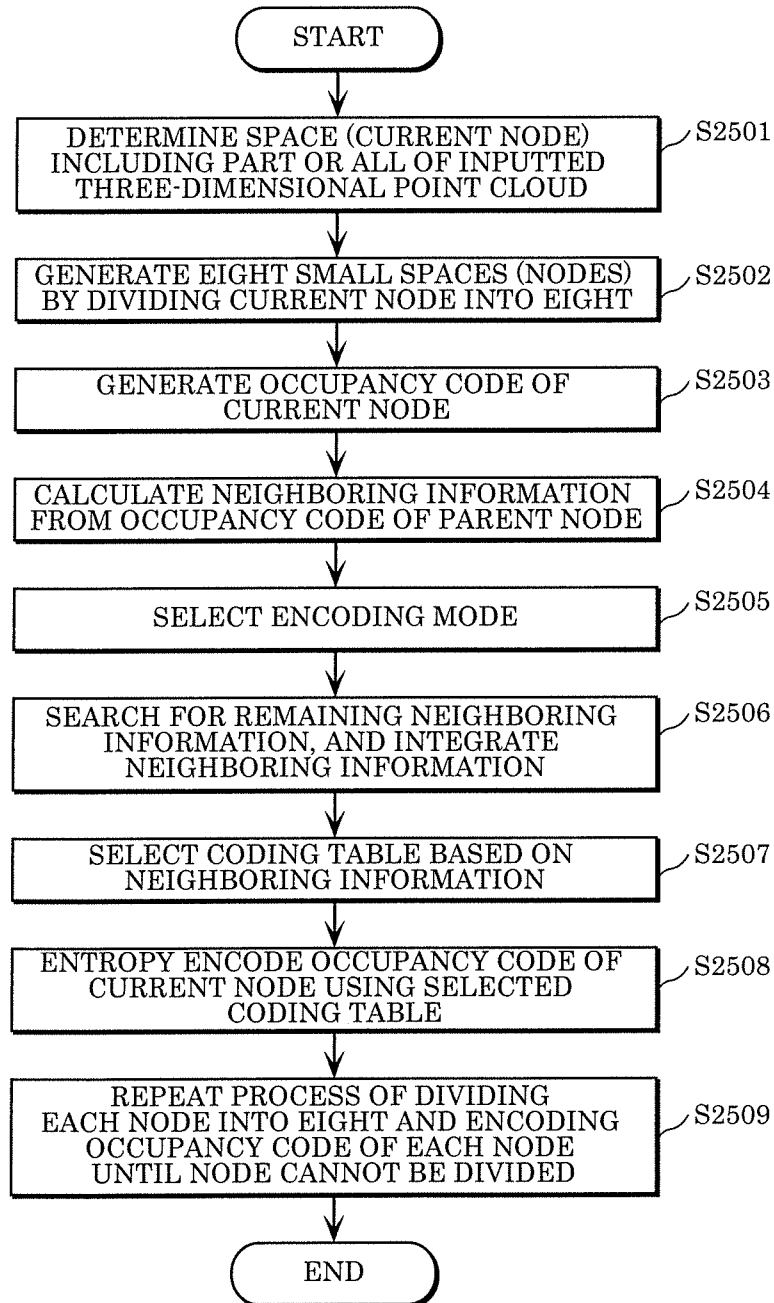


FIG. 196

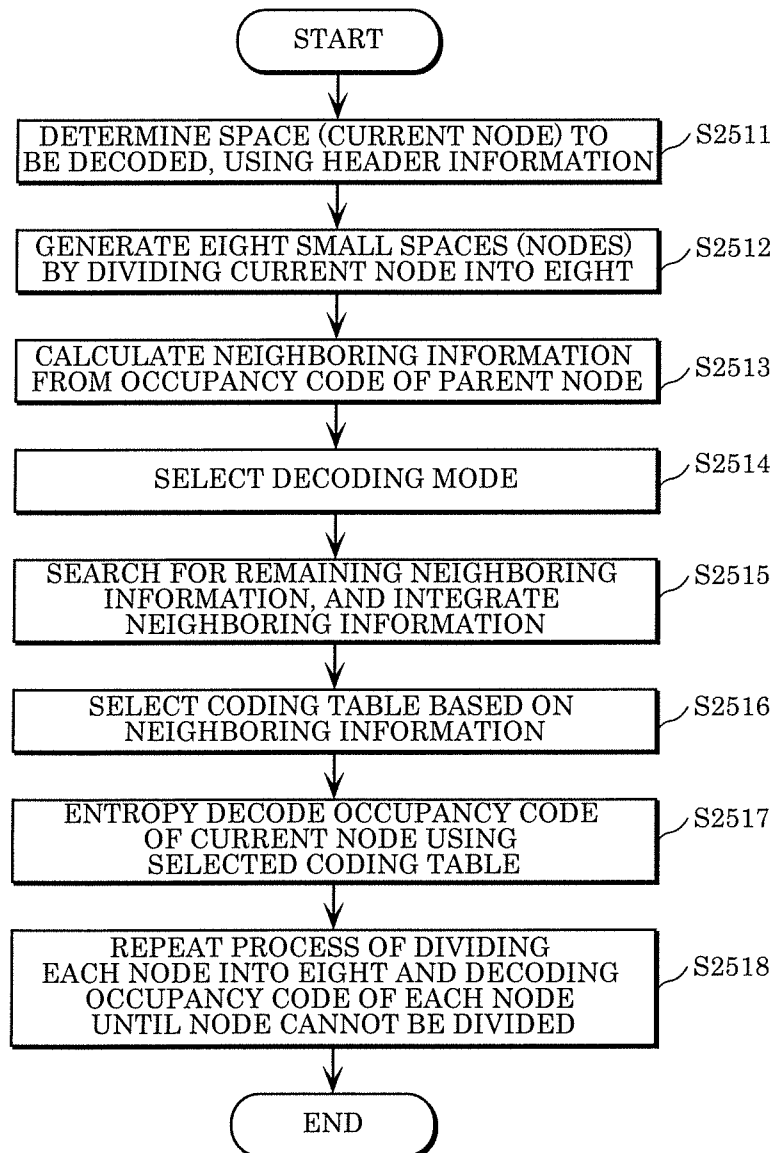


FIG. 197

```
pc_header () {  
    ...  
    coding_mode1  
    if (coding_mode1 >= 1) {  
        limit_num_of_search  
    }  
    ...  
}
```

FIG. 198

```
node(depth, index) {  
    ...  
    if (isleaf) { //leaf  
        ...  
    }  
    else { //node  
        if (coding_mode1 == 2) {  
            coding_mode2  
        }  
        occupancy_code  
    }  
    ...  
}
```

FIG. 199

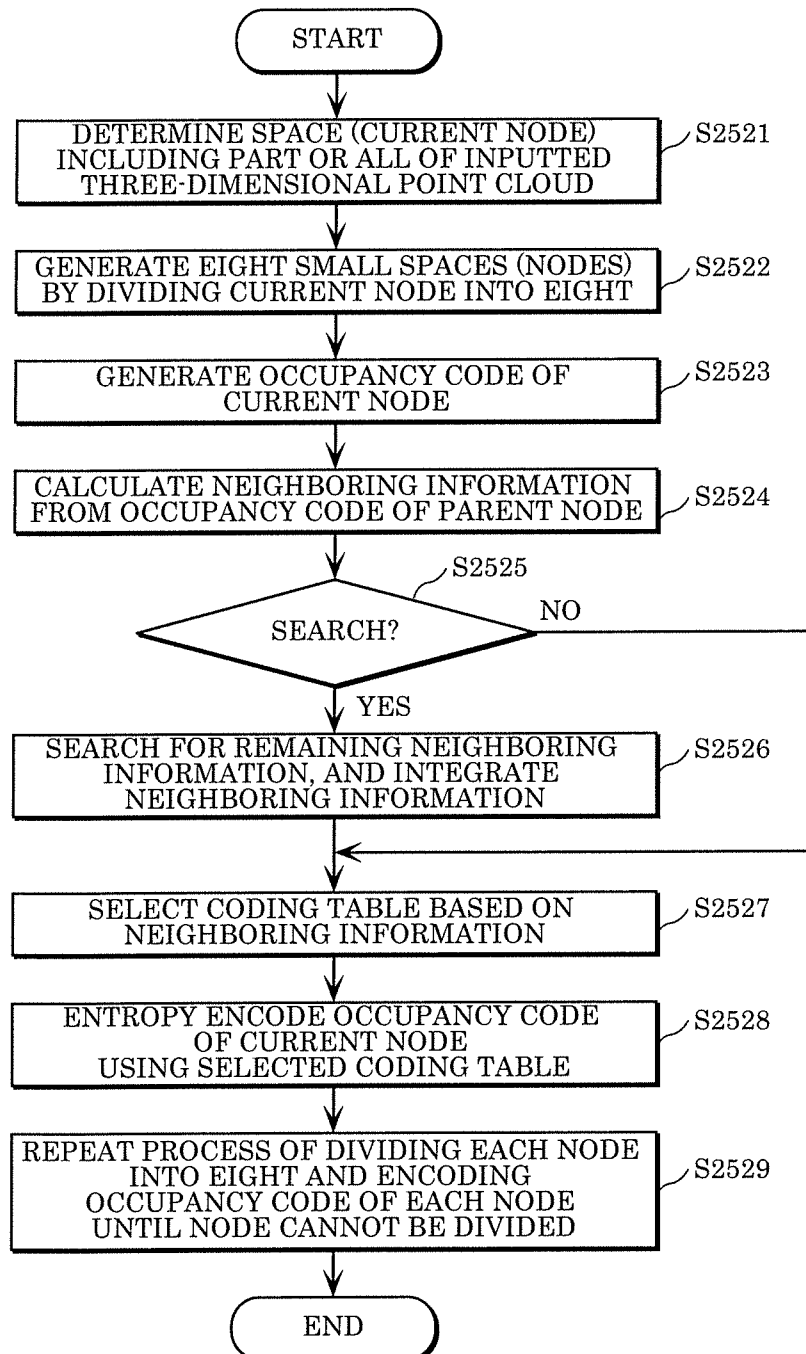


FIG. 200

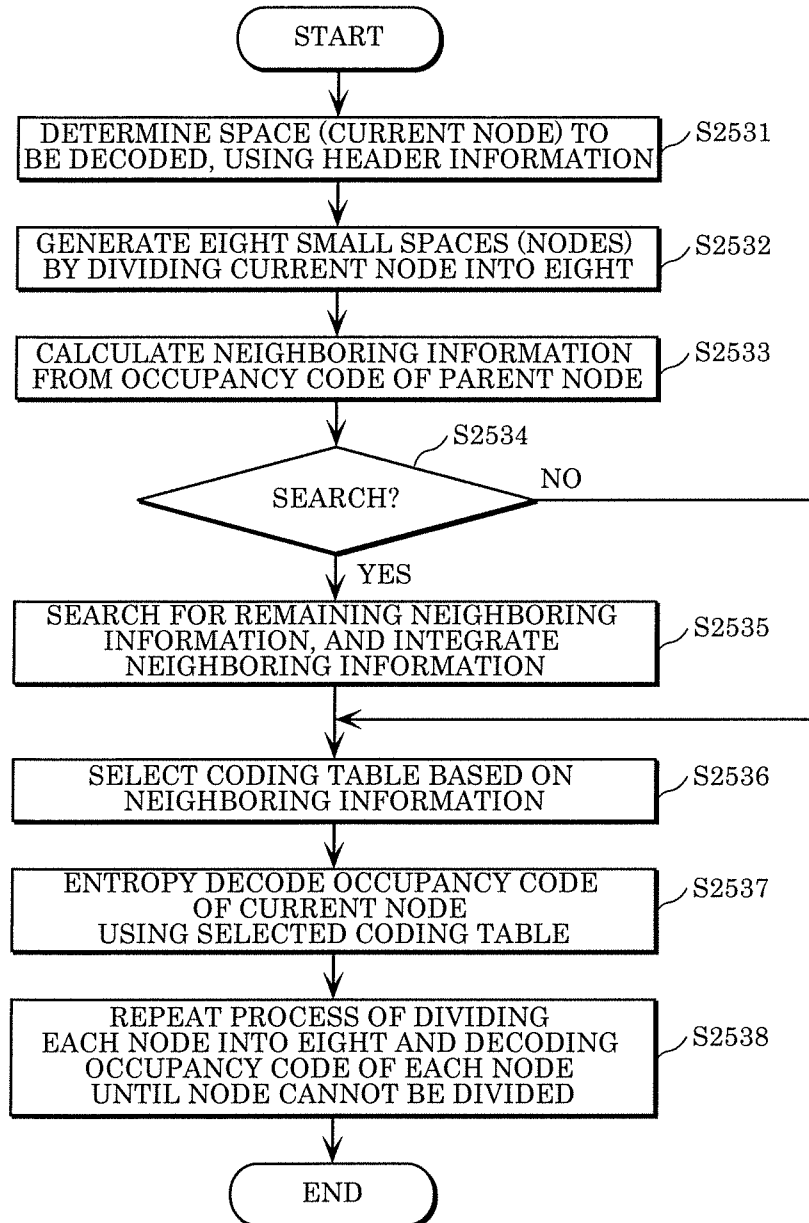


FIG. 201

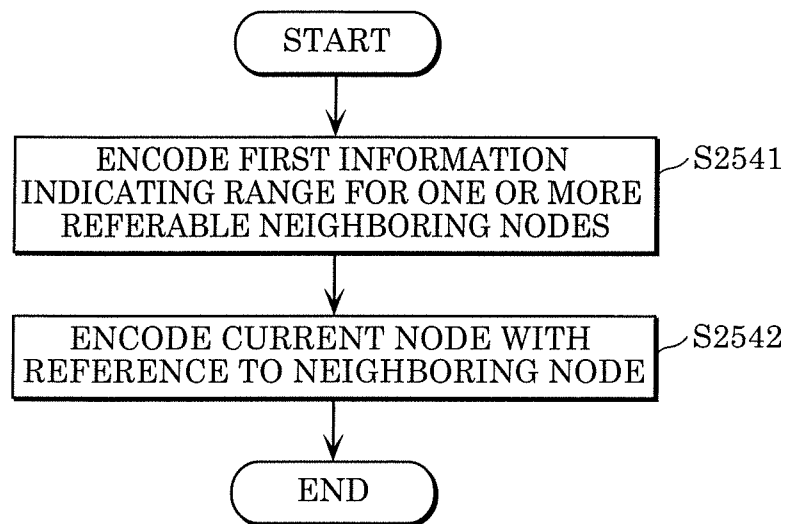


FIG. 202

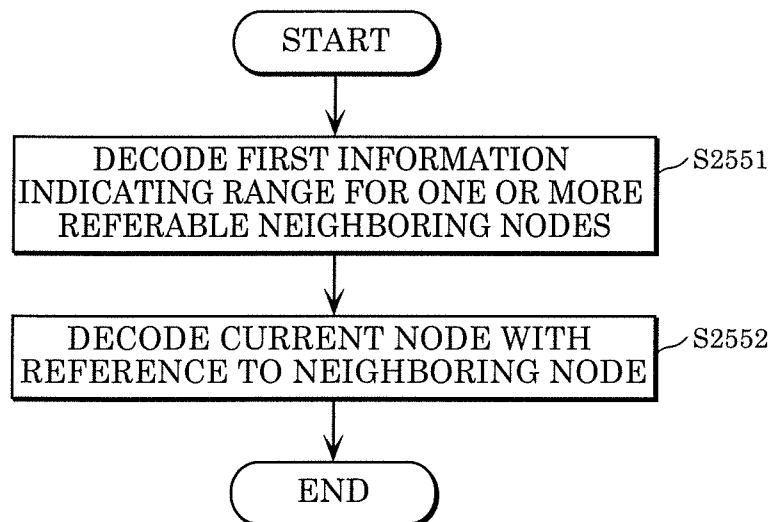


FIG. 203

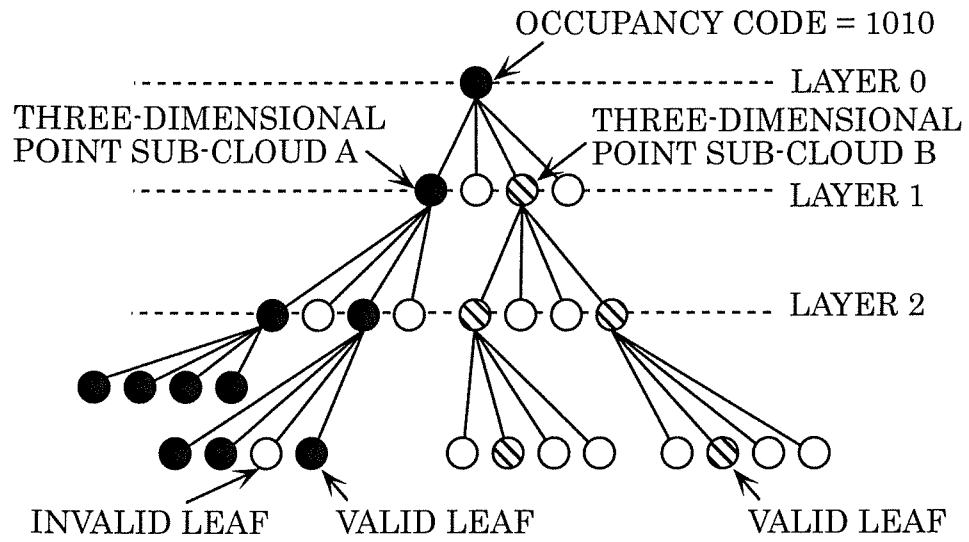


FIG. 204

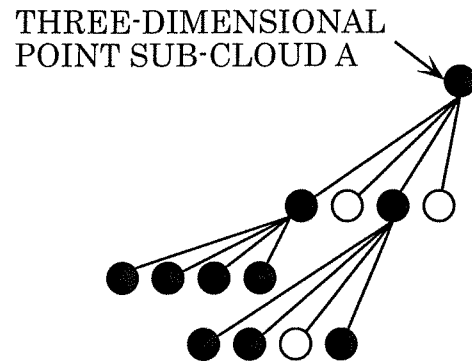


FIG. 205

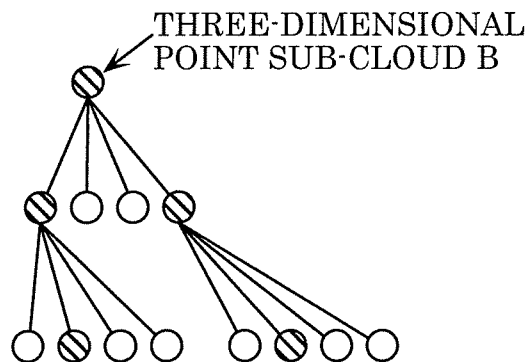


FIG. 206

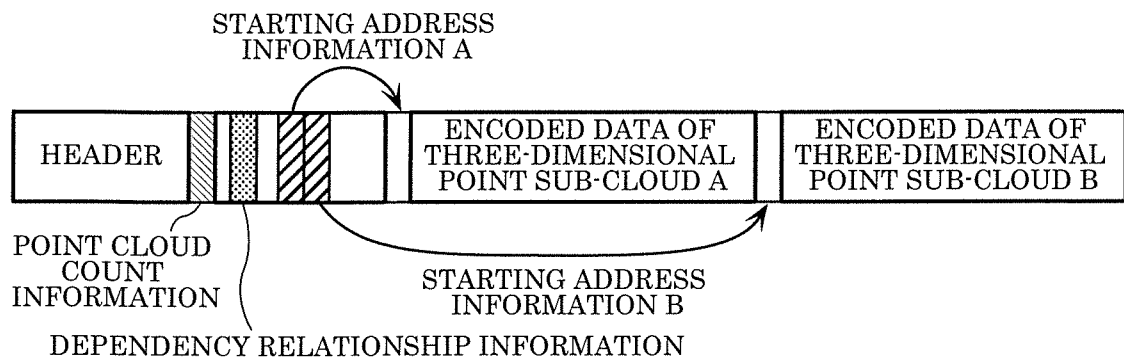


FIG. 207

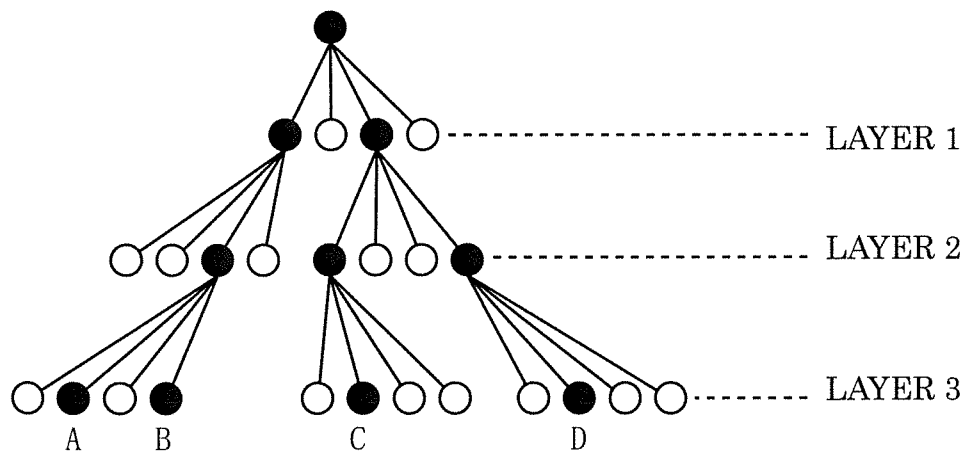


FIG. 208

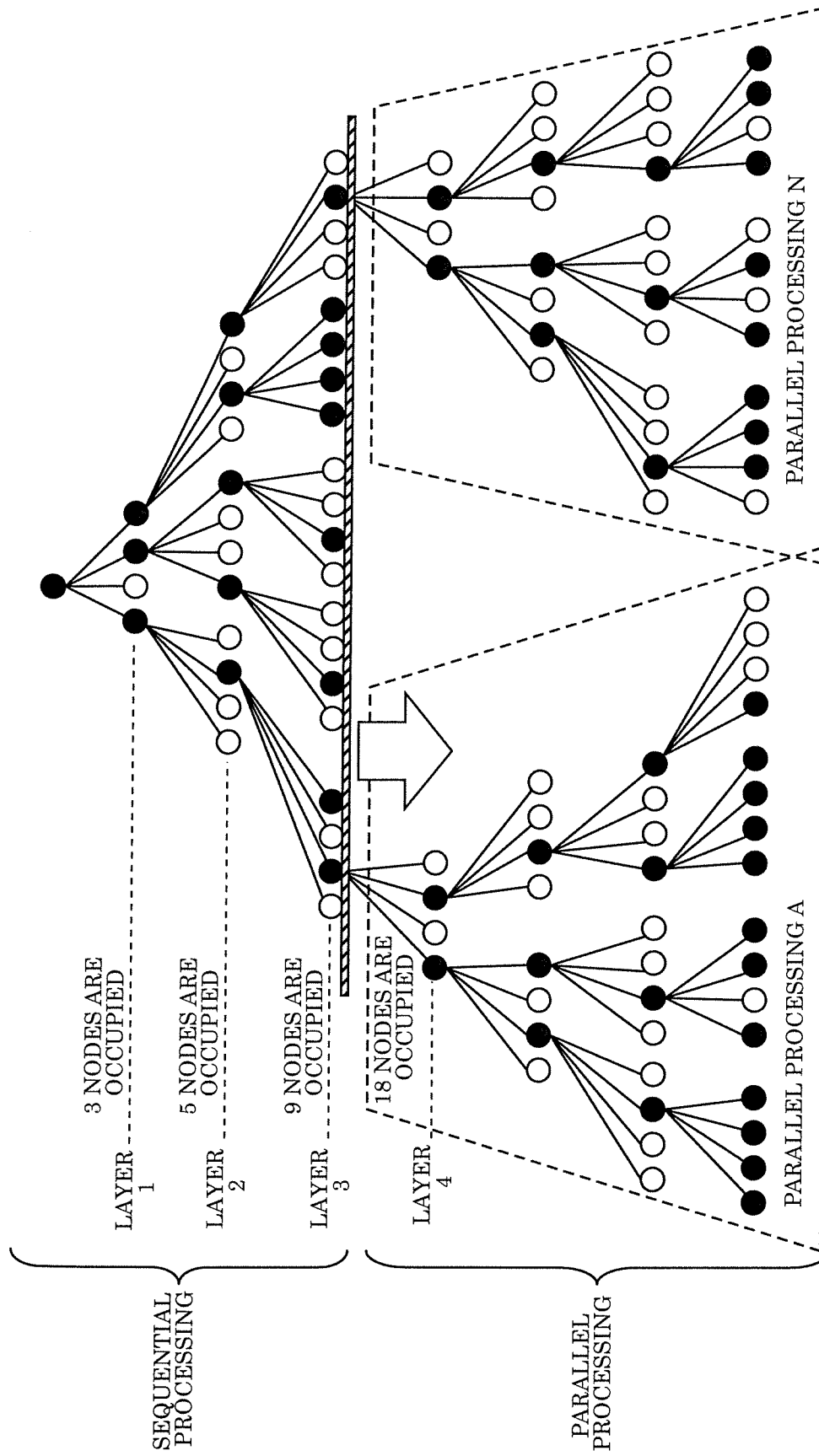


FIG. 209

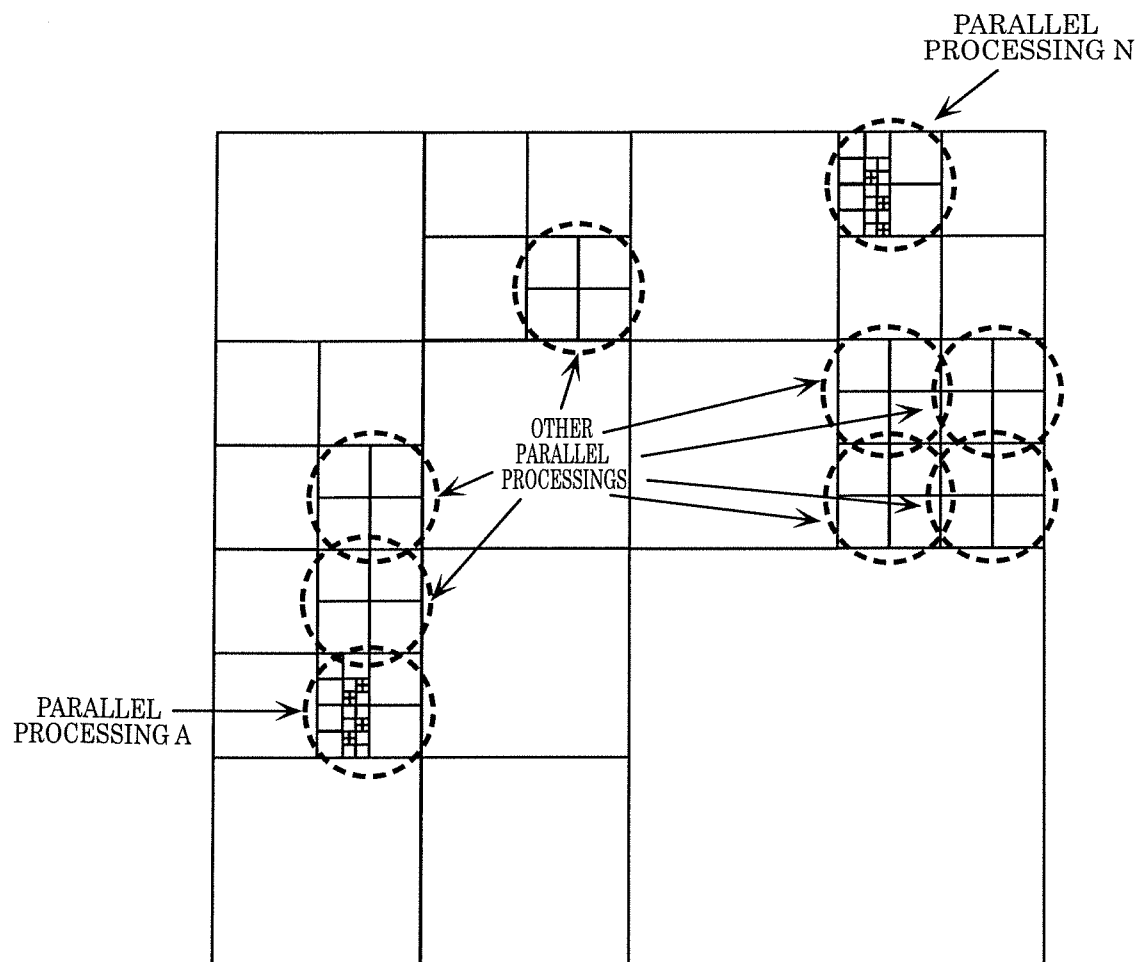


FIG. 210

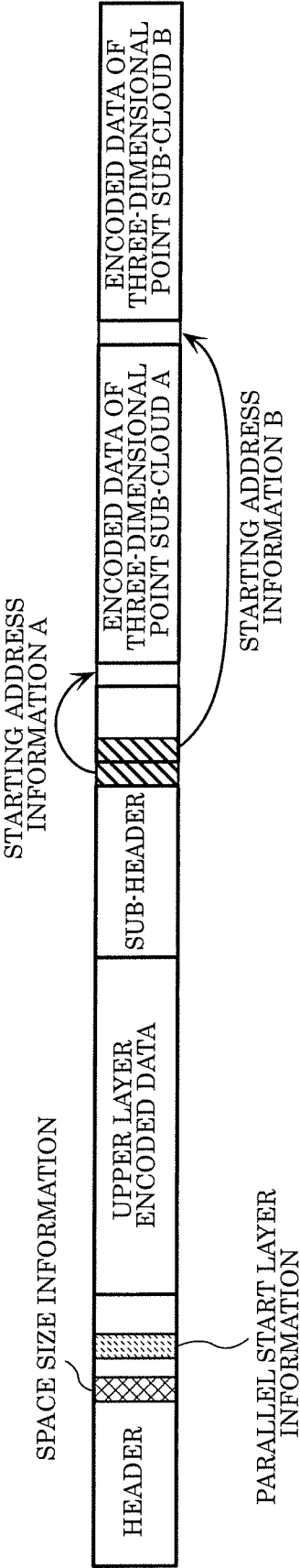


FIG. 211

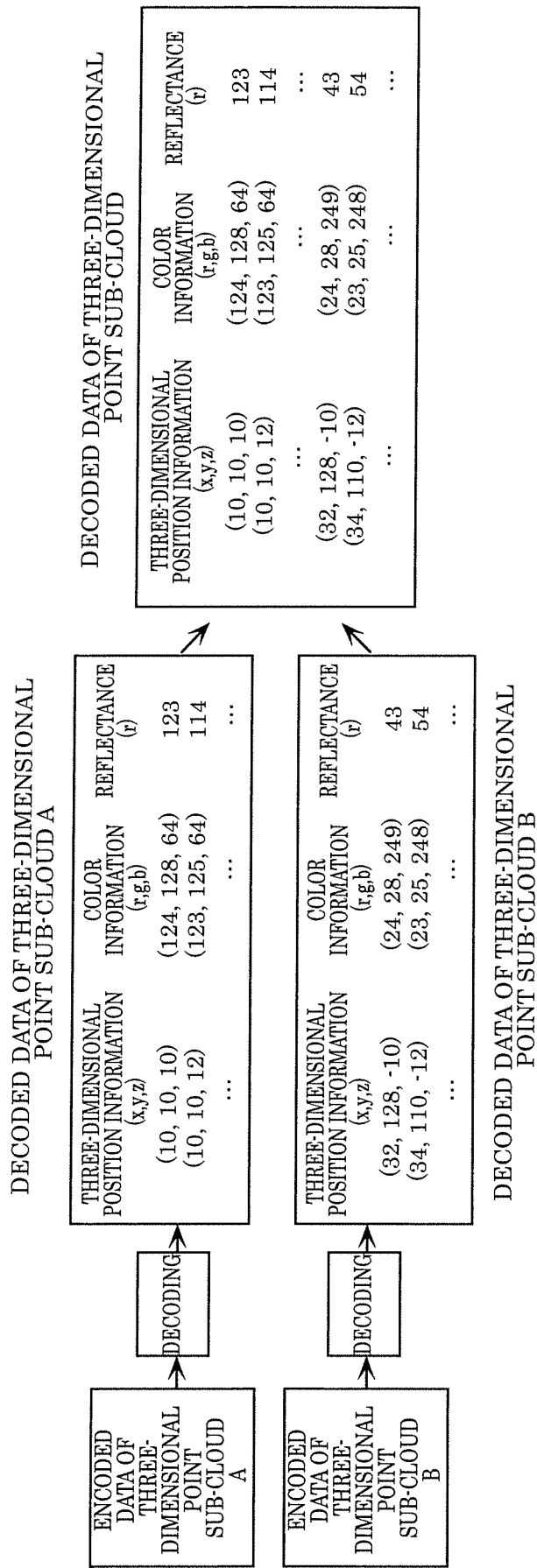


FIG. 212

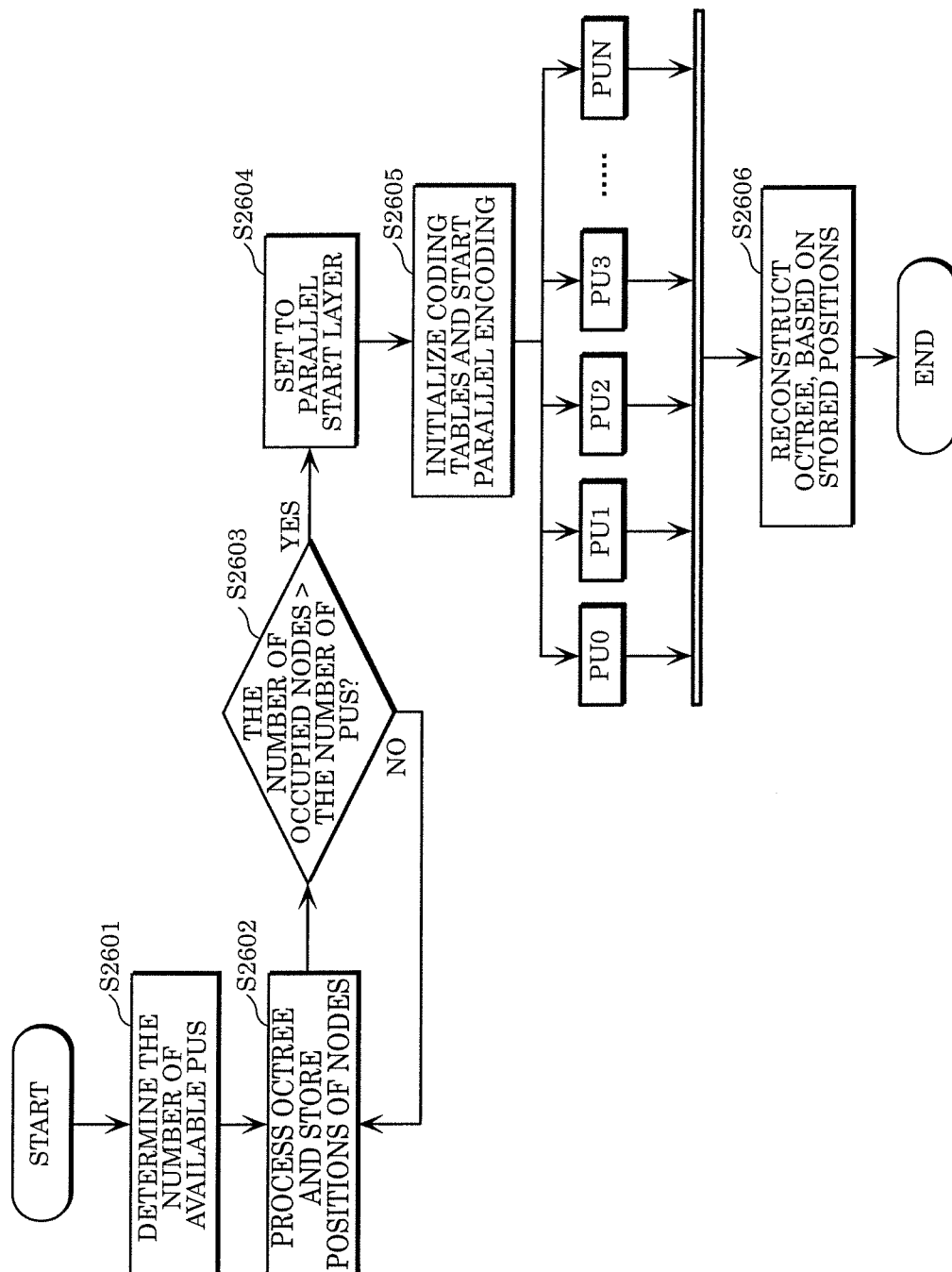


FIG. 213

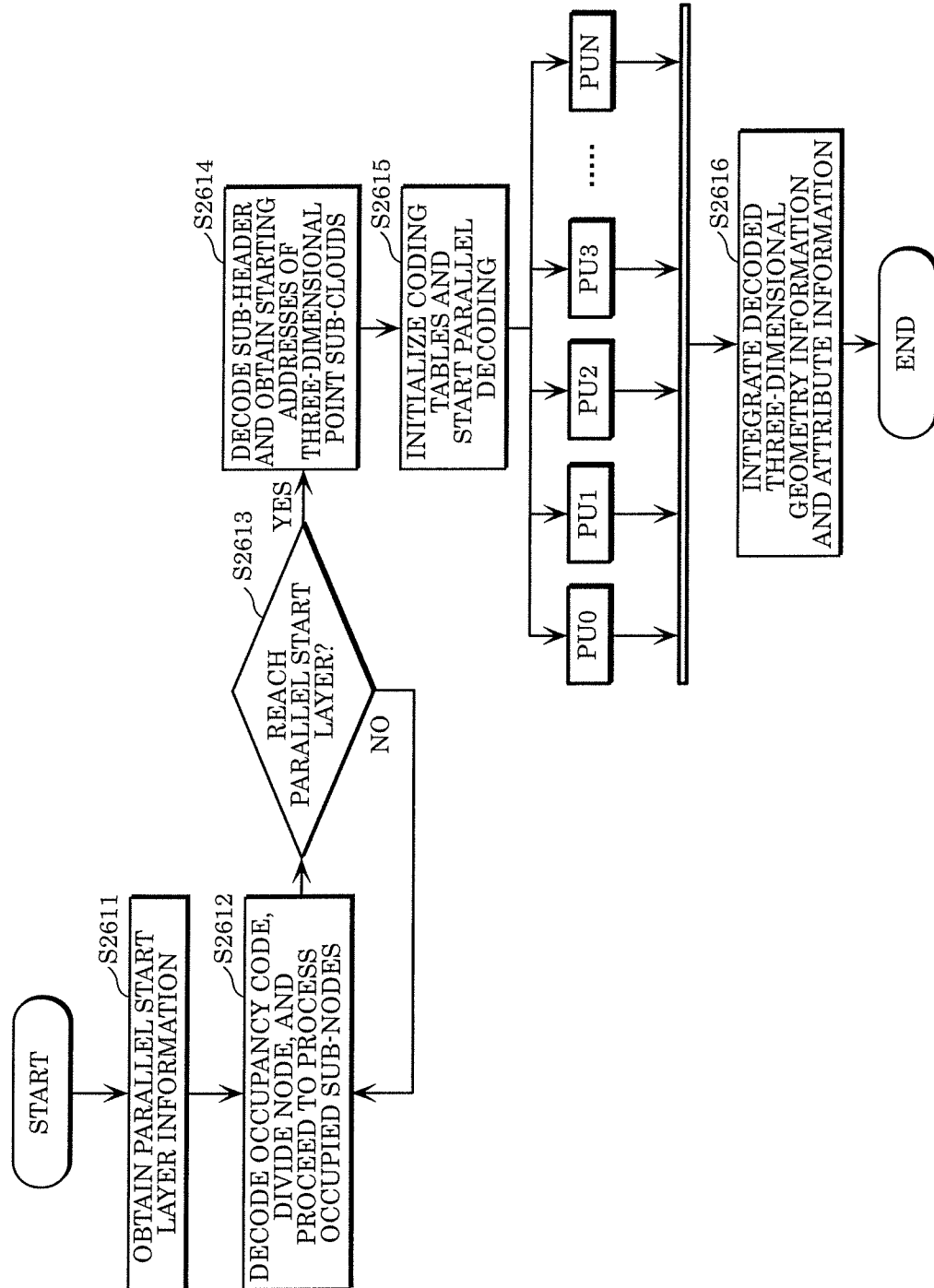


FIG. 214

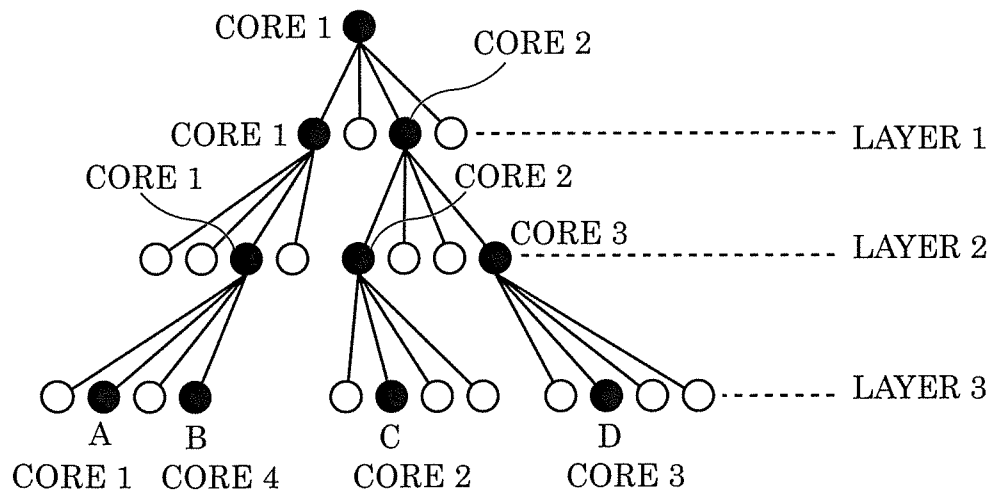


FIG. 215

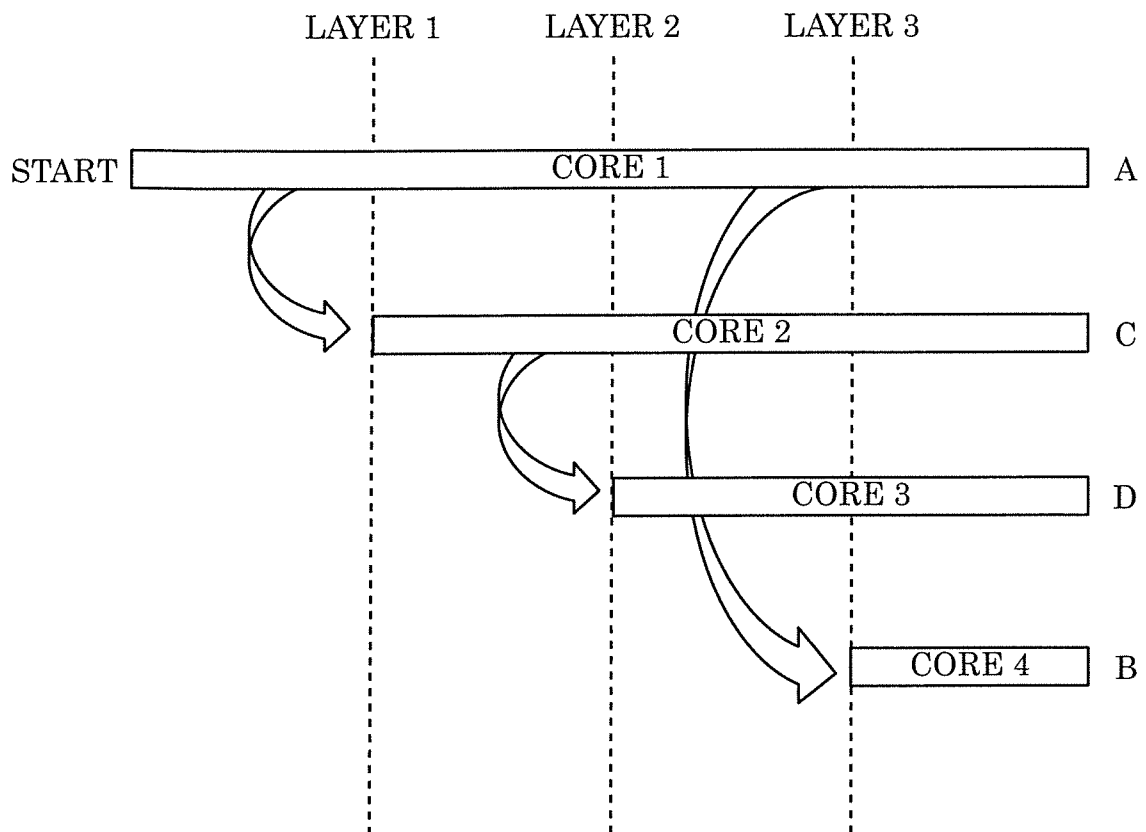


FIG. 216

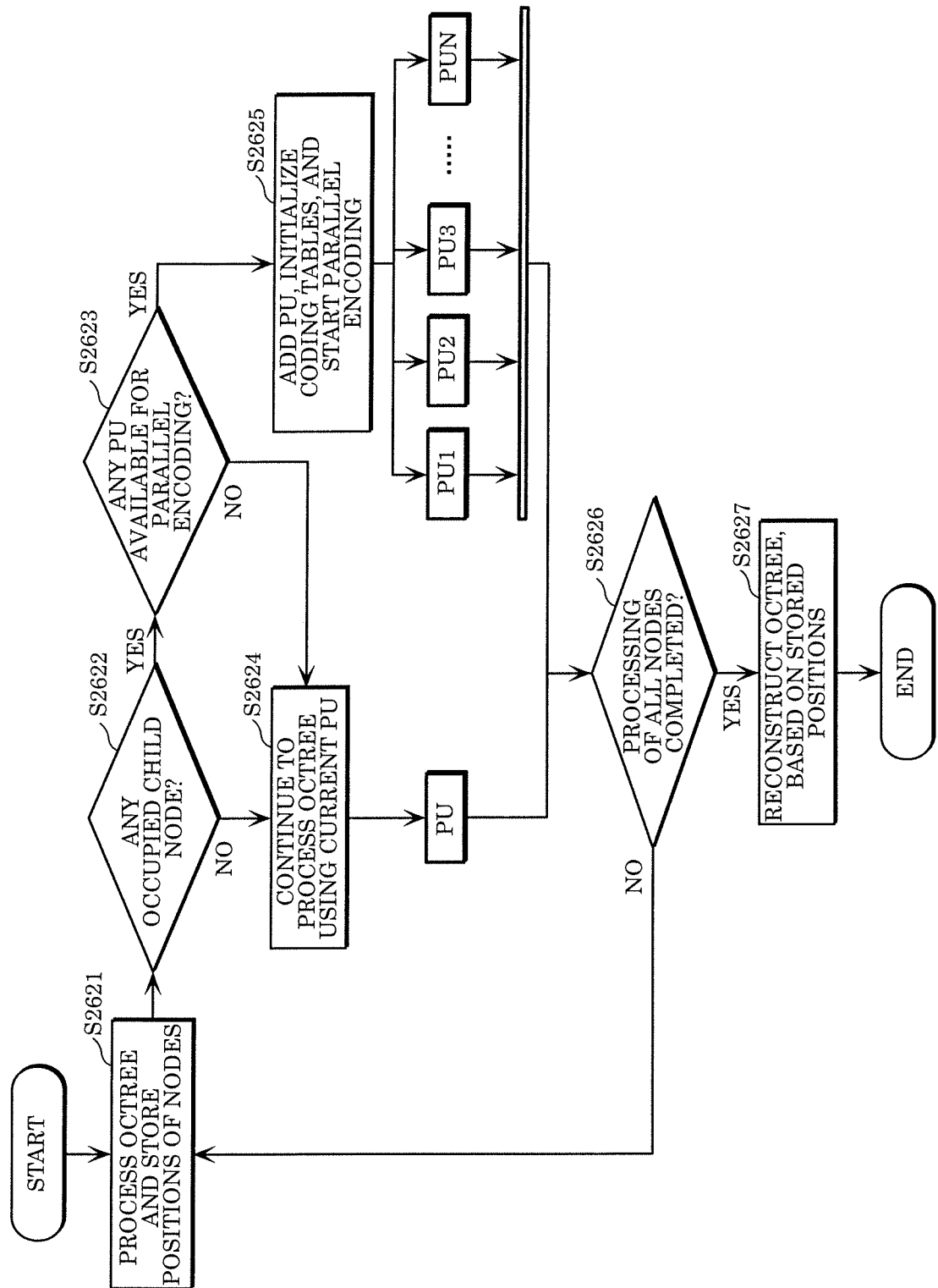


FIG. 217

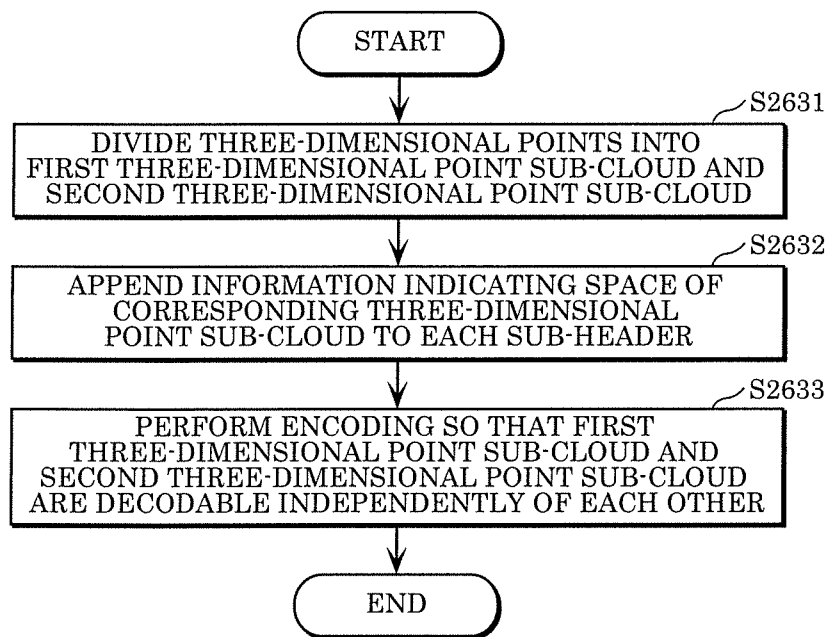


FIG. 218

