

FIG. 1

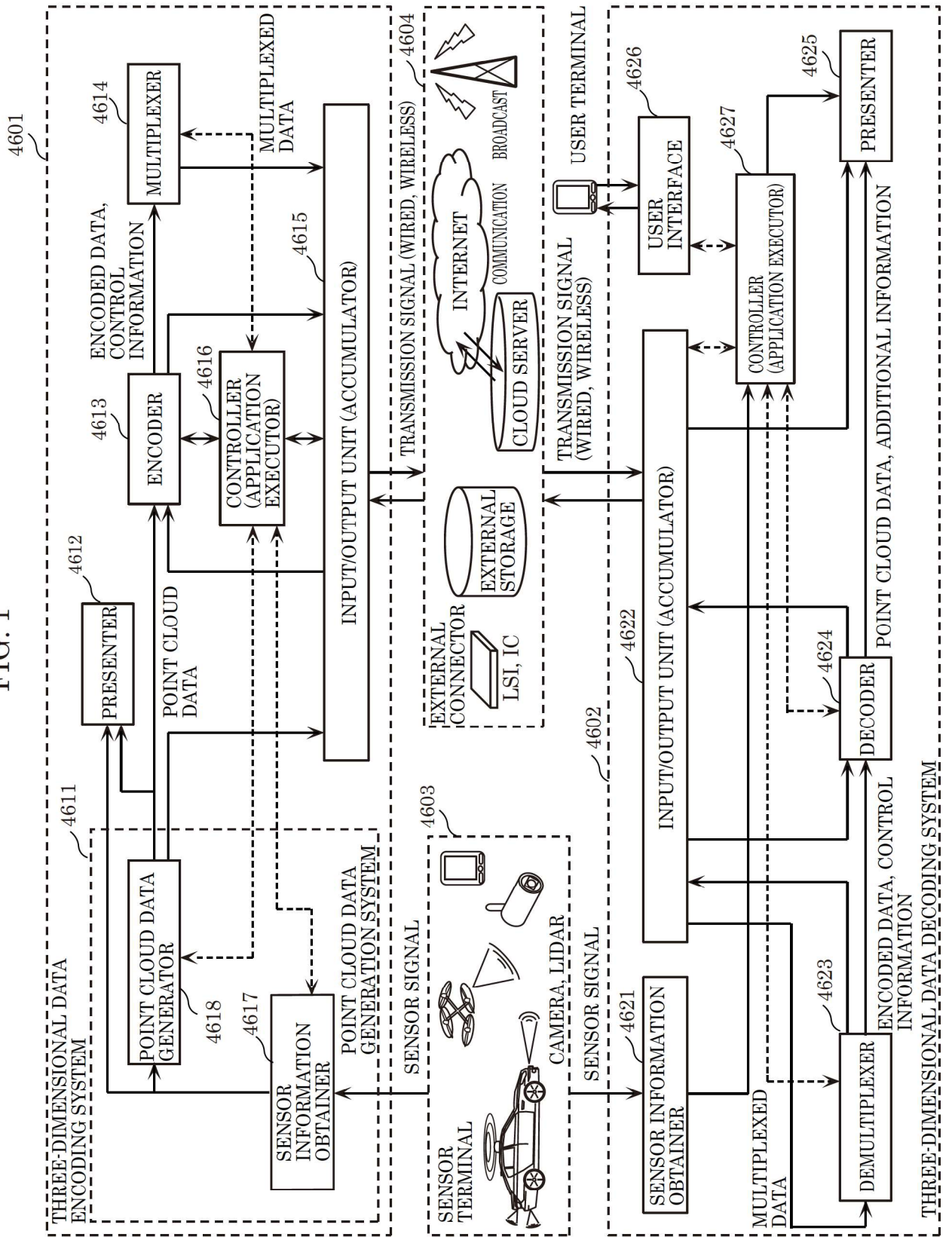


FIG. 2

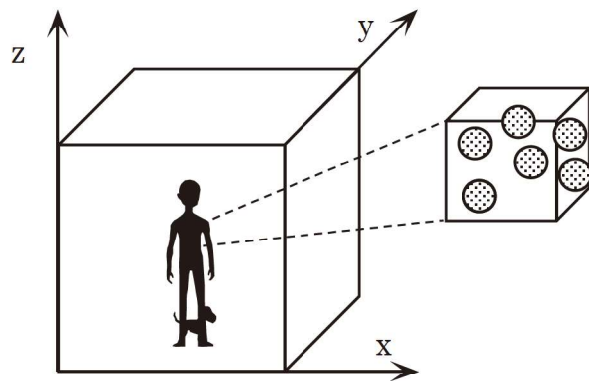


FIG. 3

	GEOMETRY INFORMATION	ATTRIBUTE INFORMATION
n = 1	$G(1) = (x_1, y_1, z_1)$	$A(1) = (R_1, G_1, B_1)$
n = 2	$G(2) = (x_2, y_2, z_2)$	$A(2) = (R_2, G_2, B_2)$
$\vdots$	$\vdots$	$\vdots$
n = N	$G(N) = (x_N, y_N, z_N)$	$A(N) = (R_N, G_N, B_N)$

FIG. 4

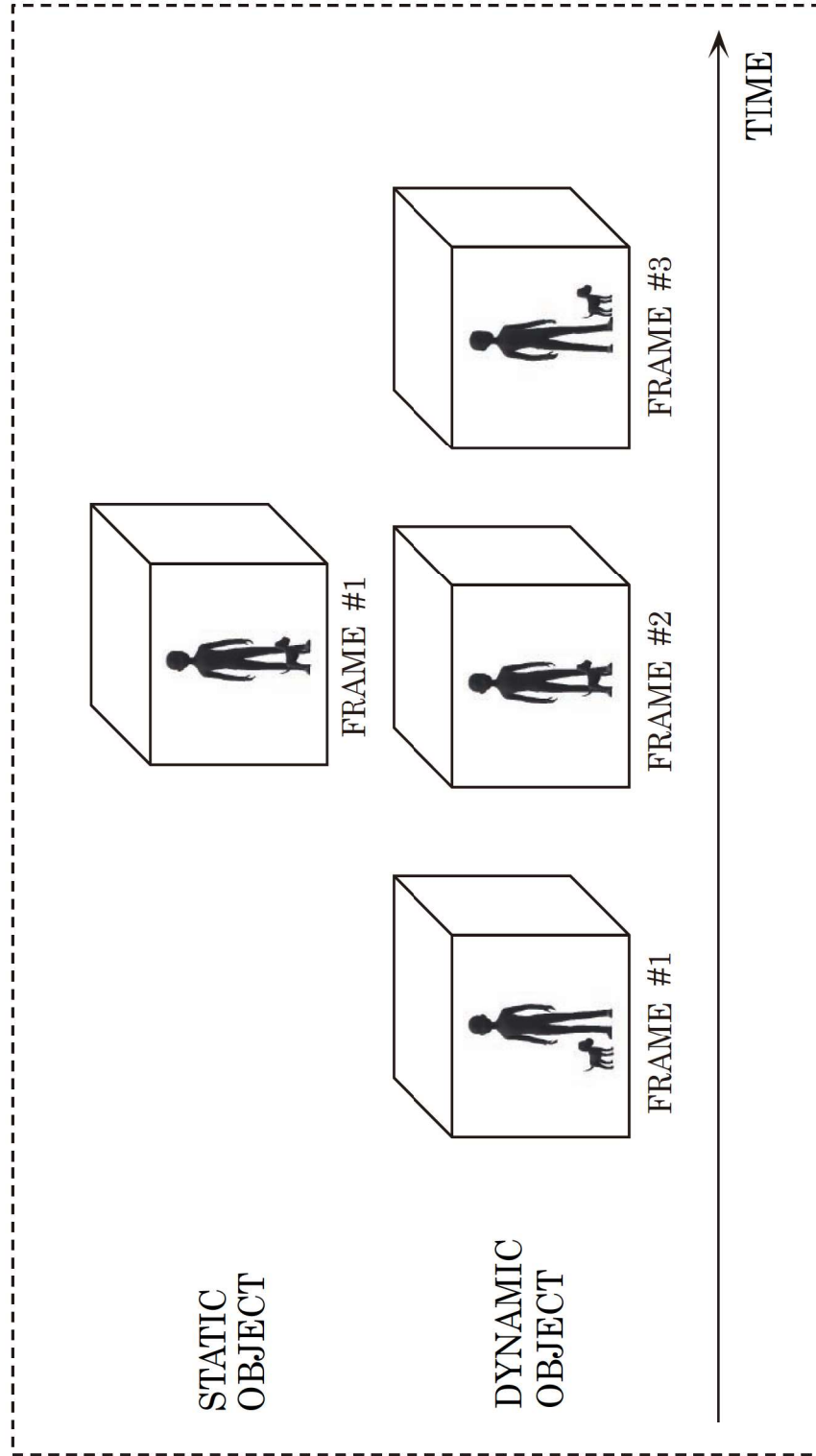


FIG. 5

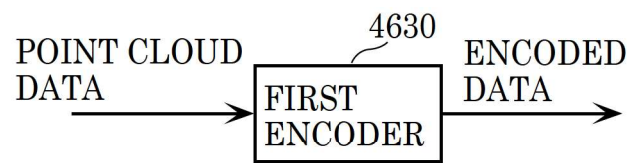


FIG. 6

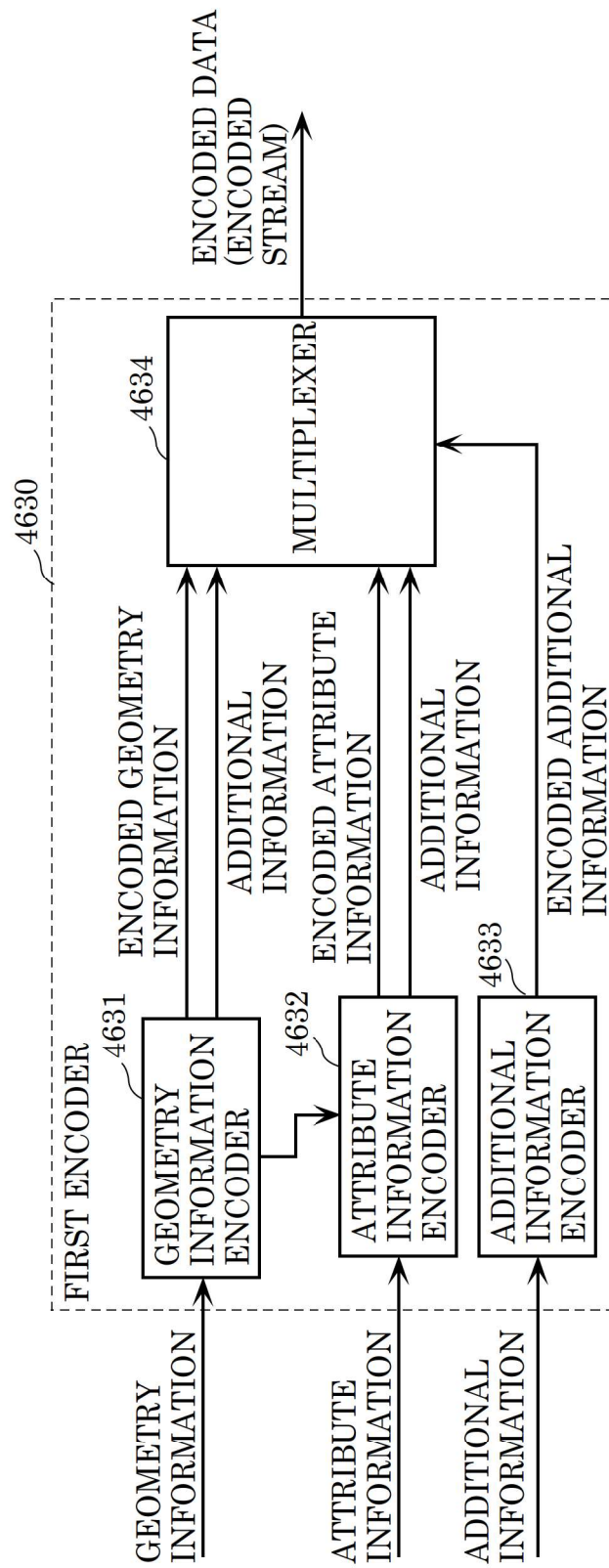


FIG. 7

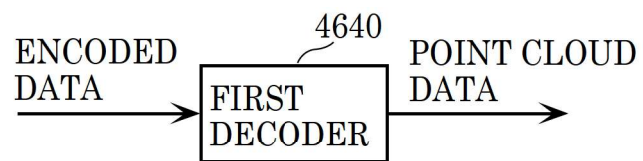


FIG. 8

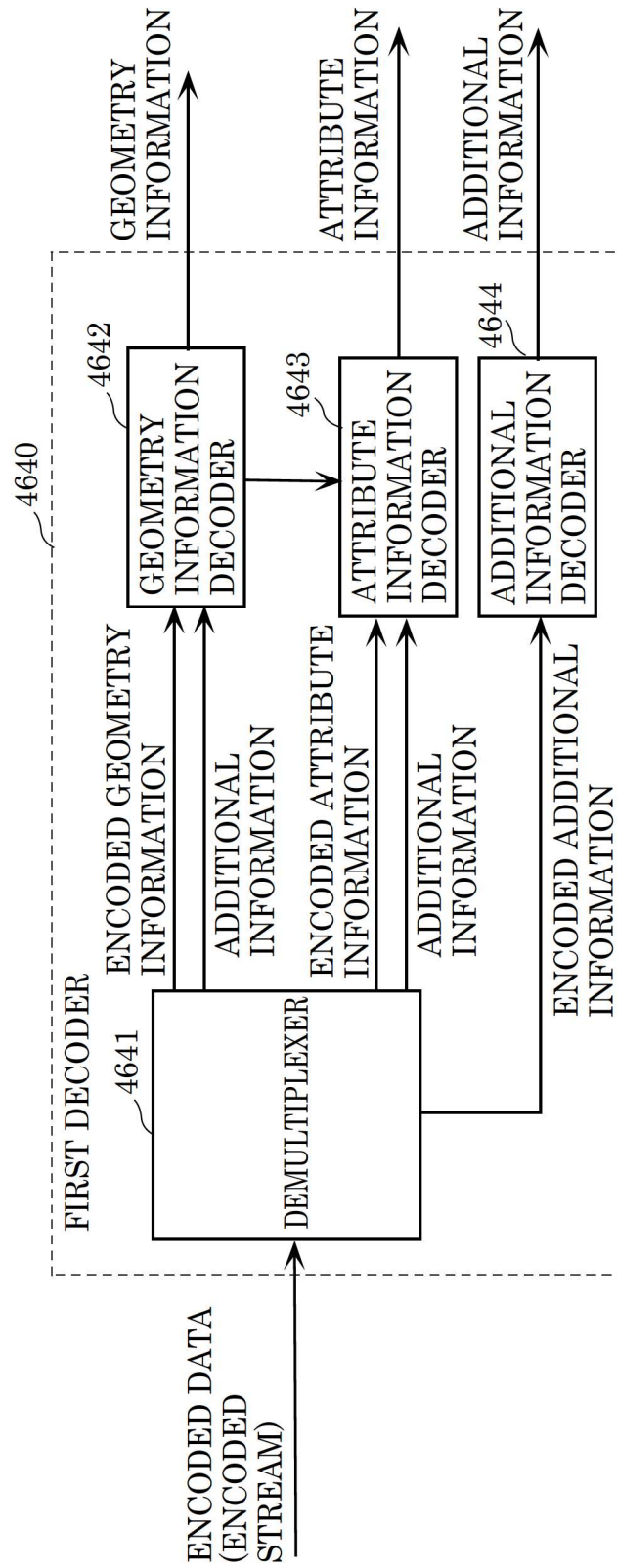


FIG. 9

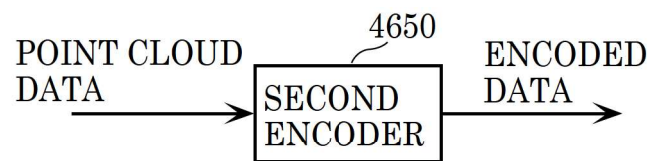


FIG. 10

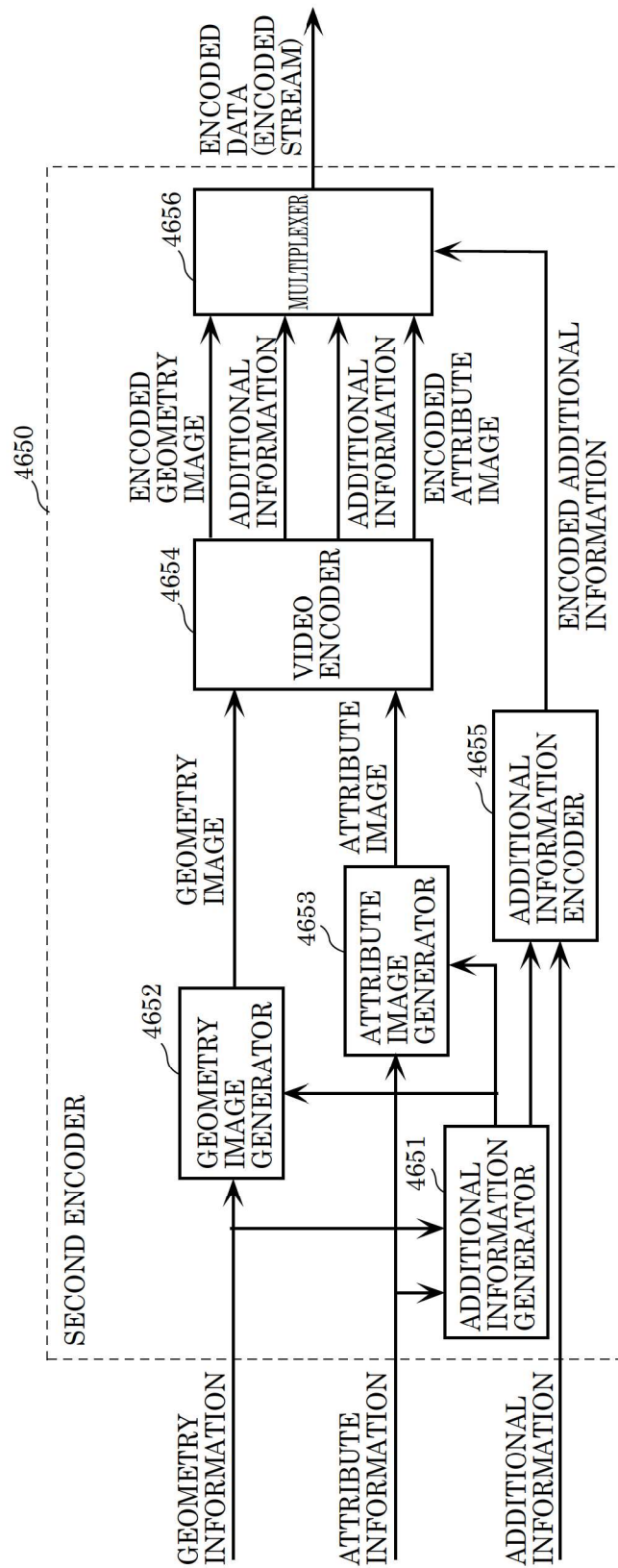


FIG. 11

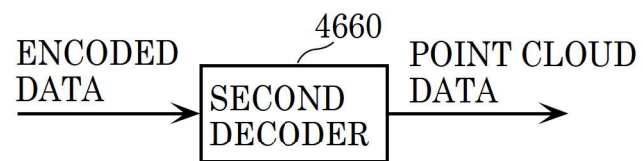


FIG. 12

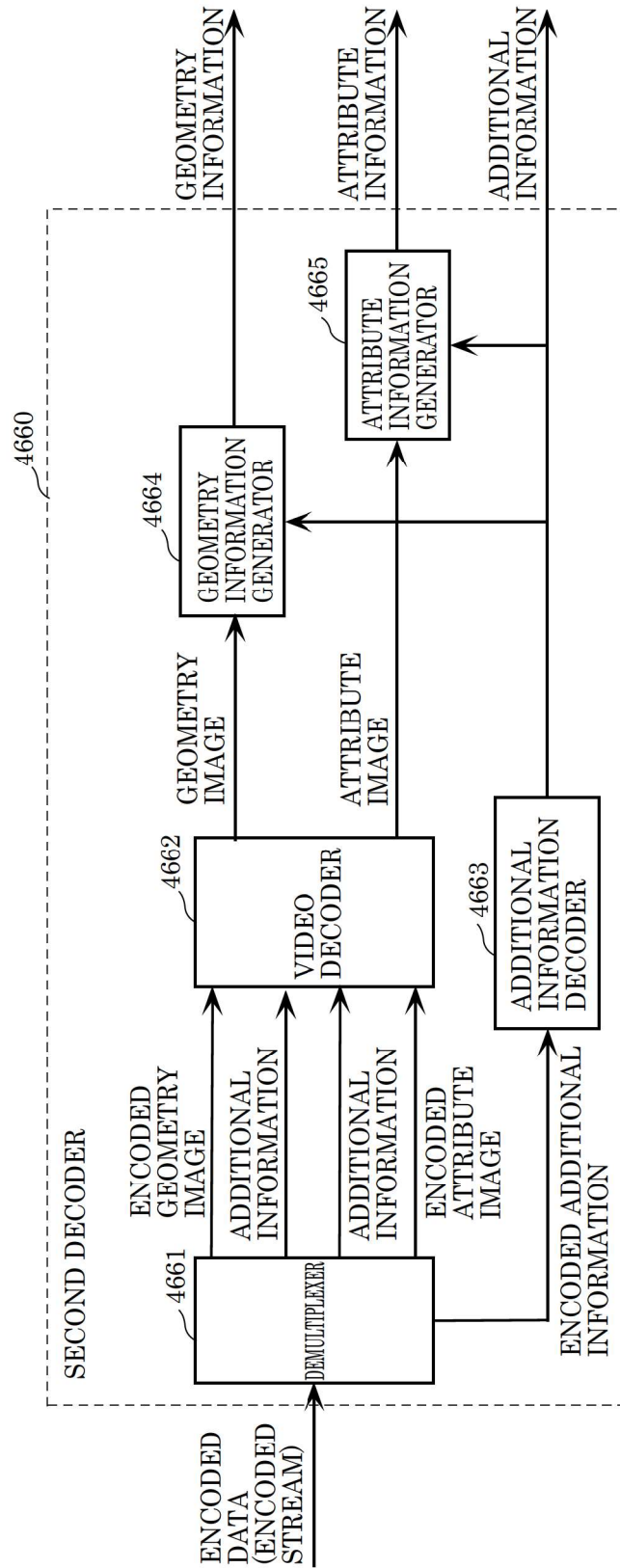


FIG. 13

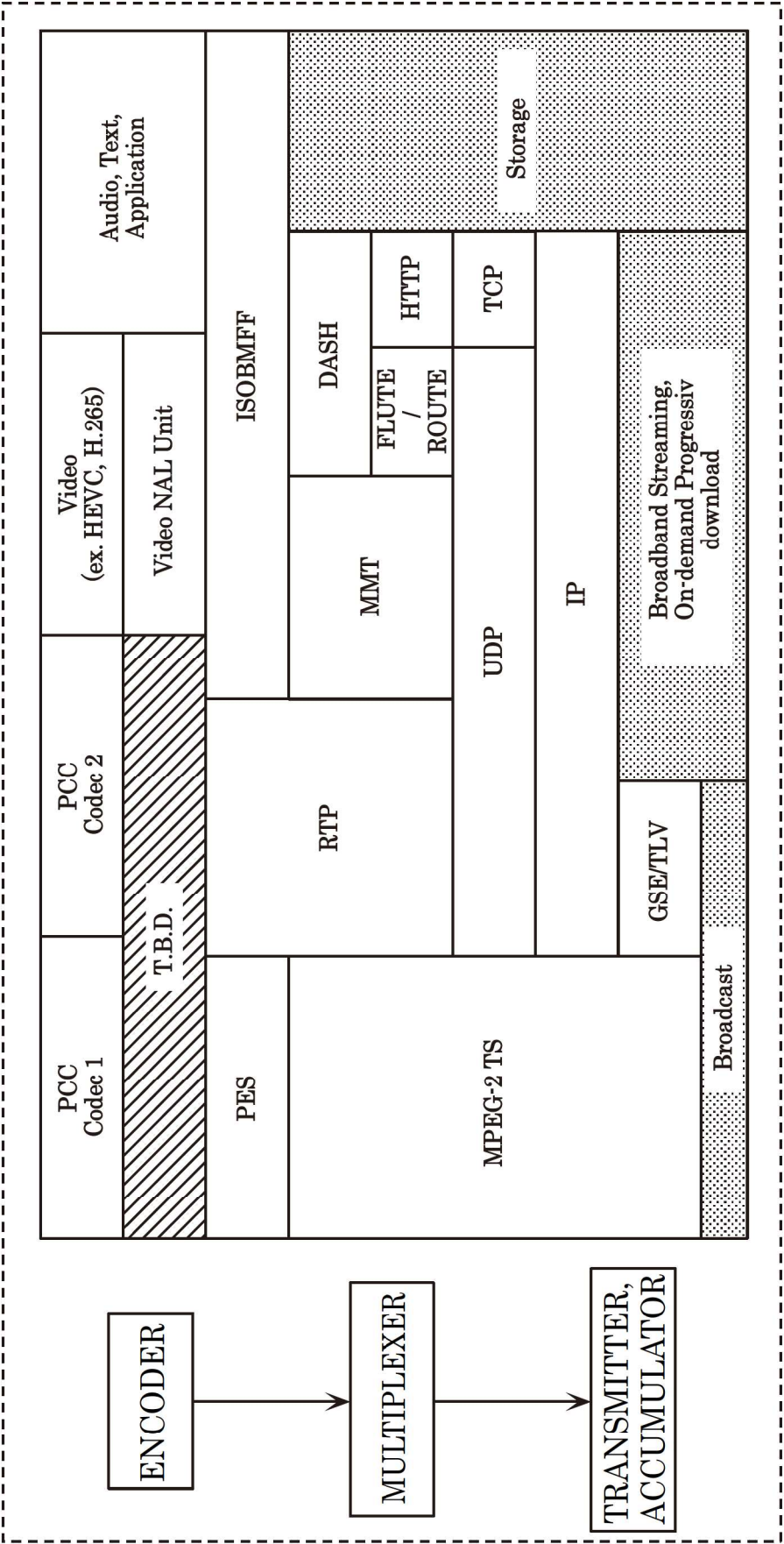


FIG. 14

ftyp	moov	mdat
------	------	------

FIG. 15

Codec 1	Codec 2
Codec 1 NAL unit	Codec 2 NAL unit
PCC NAL Unit	
Carriage of Codec 1	Carriage of Codec 2
ISOBMFF	

FIG. 16

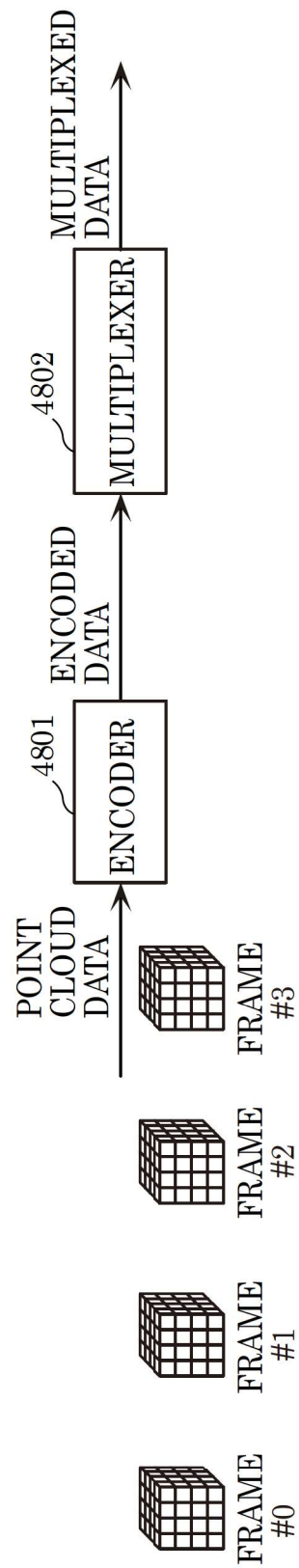


FIG. 17

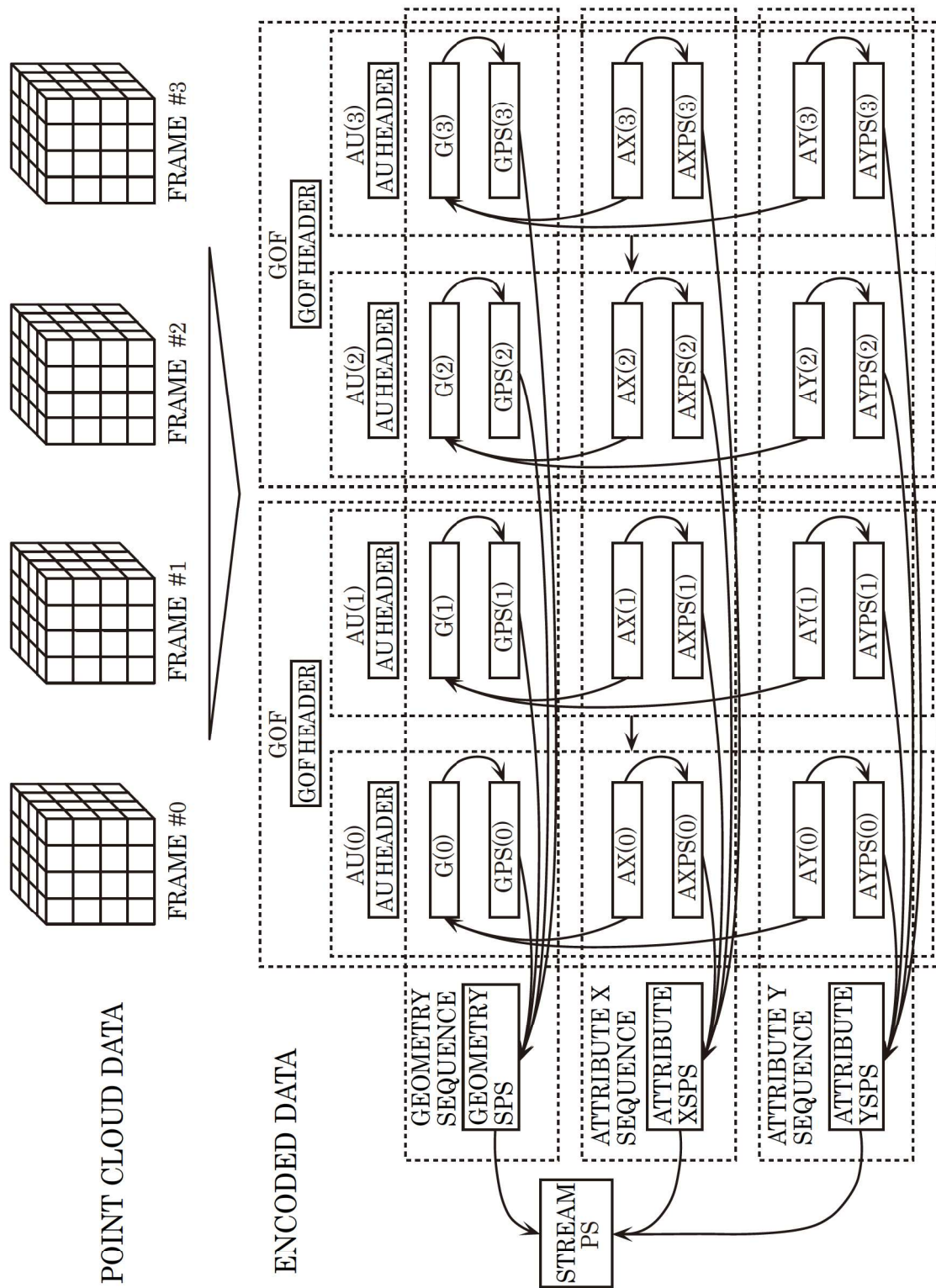


FIG. 18

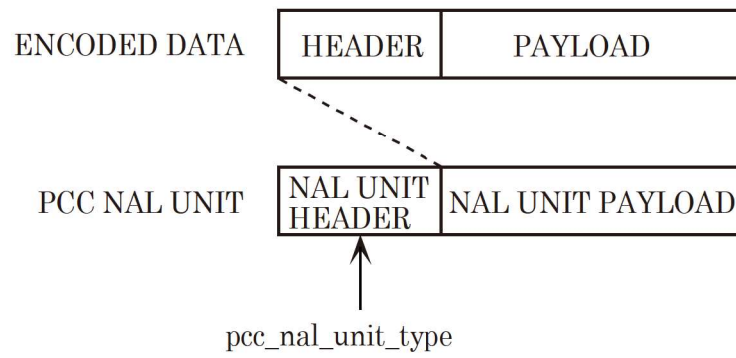


FIG. 19

EXAMPLES OF SEMANTICS OF pcc_nal_unit_type

(1) if pcc_codec_type == Codec 1

- 0:Codec1 Geometry
- 1:Codec1 AttributeX
- 2:Codec1 AttributeY
- 3:Codec1 Geom. PS
- 4:Codec1 AttrX. PS
- 5:Codec1 AttrX. PS
- 6:Codec1 Geometry Sequence PS
- 7:Codec1 AttributeX Sequence PS
- 8:Codec1 AttributeY Sequence PS
- 9:Codec1 AU Header
- 10:Codec1 GOF Header
- 11 ~:Codec1 reserved for future use

(2) if pcc_codec_type == Codec 2

- 0:Codec2 DataA
- 1:Codec2 MetaDataA
- 2:Codec2 MetaDataB
- 3 ~:Codec2 reserved for future use

FIG. 20

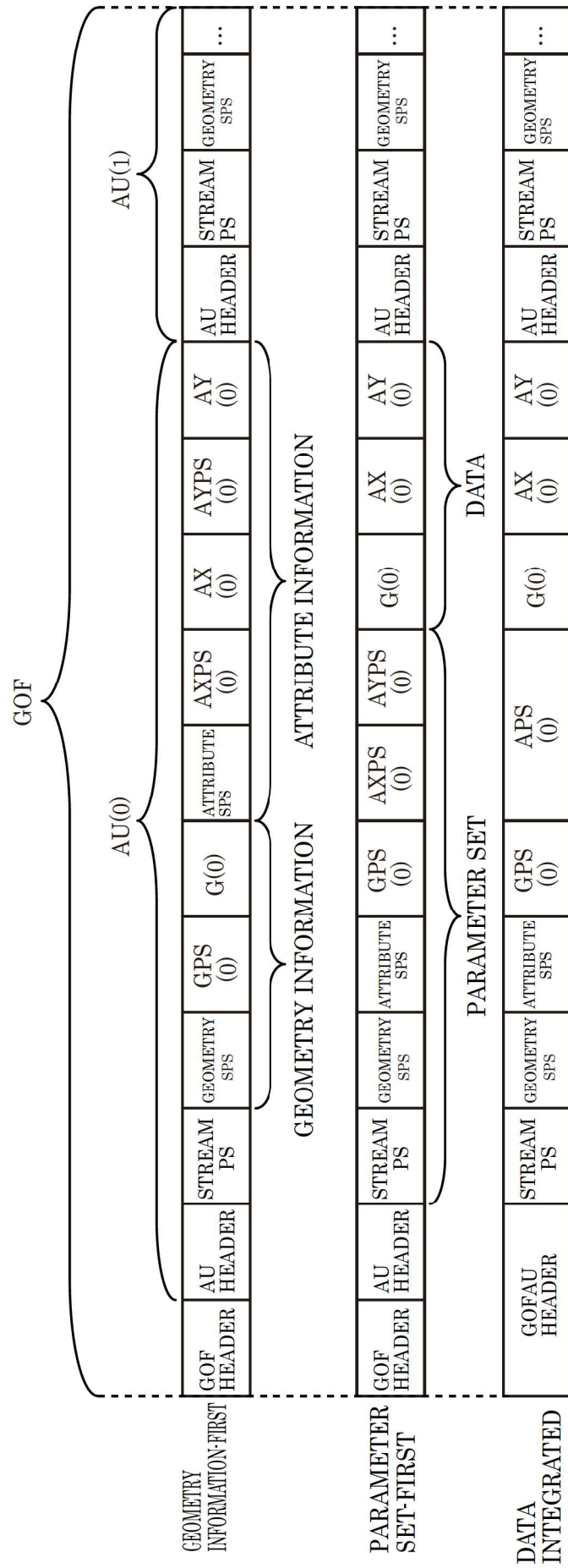


FIG. 21

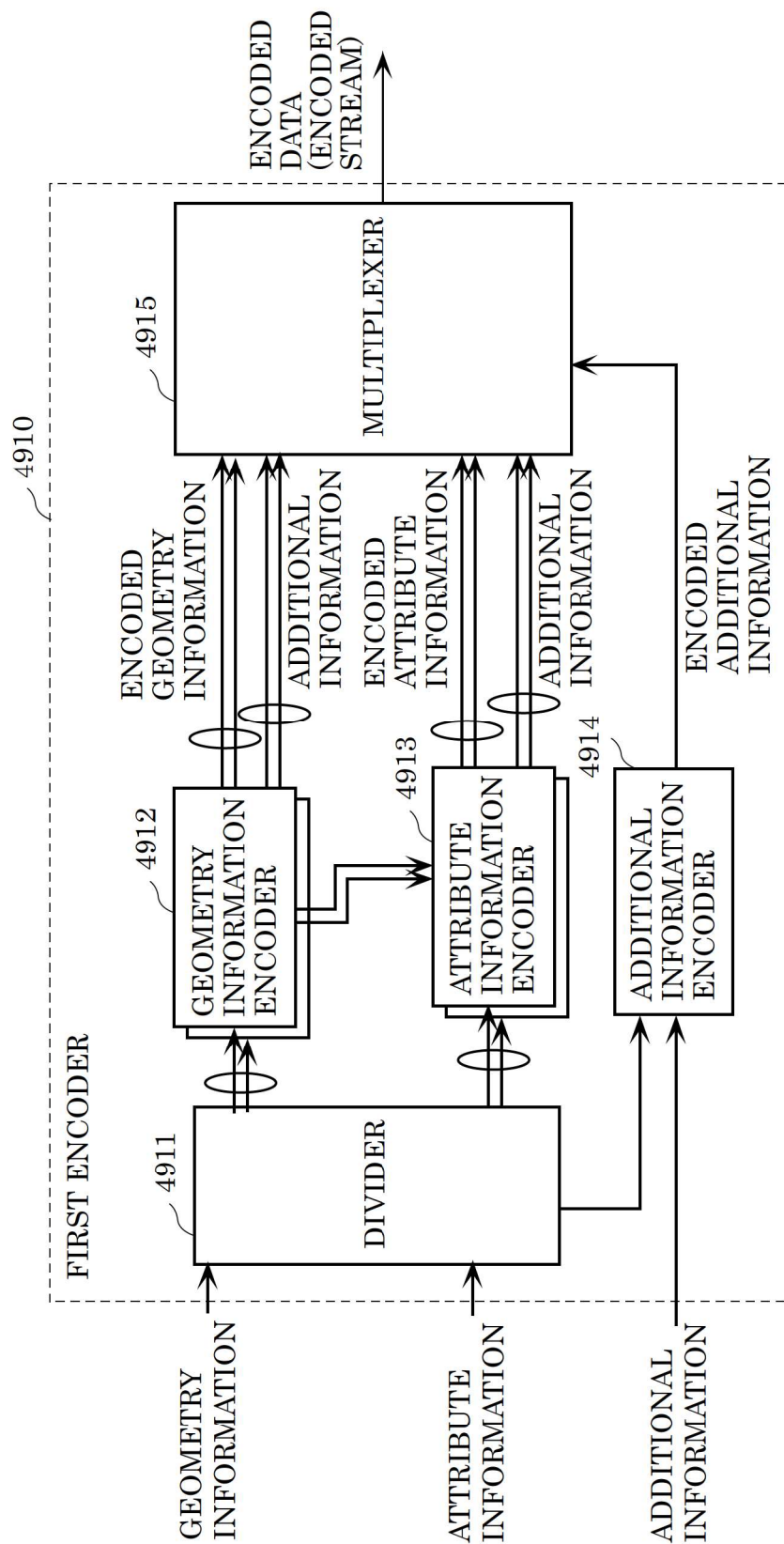


FIG. 22

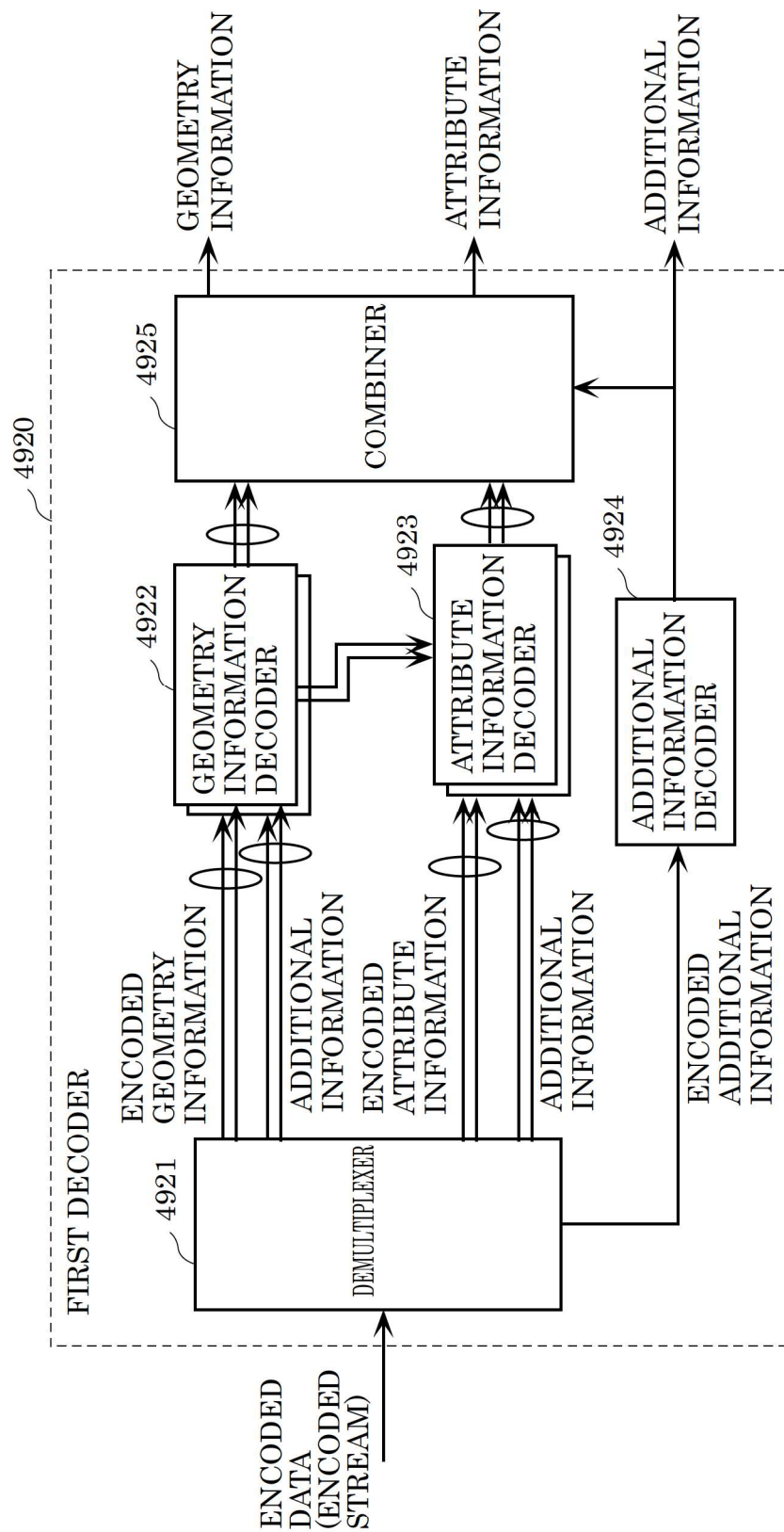


FIG. 23

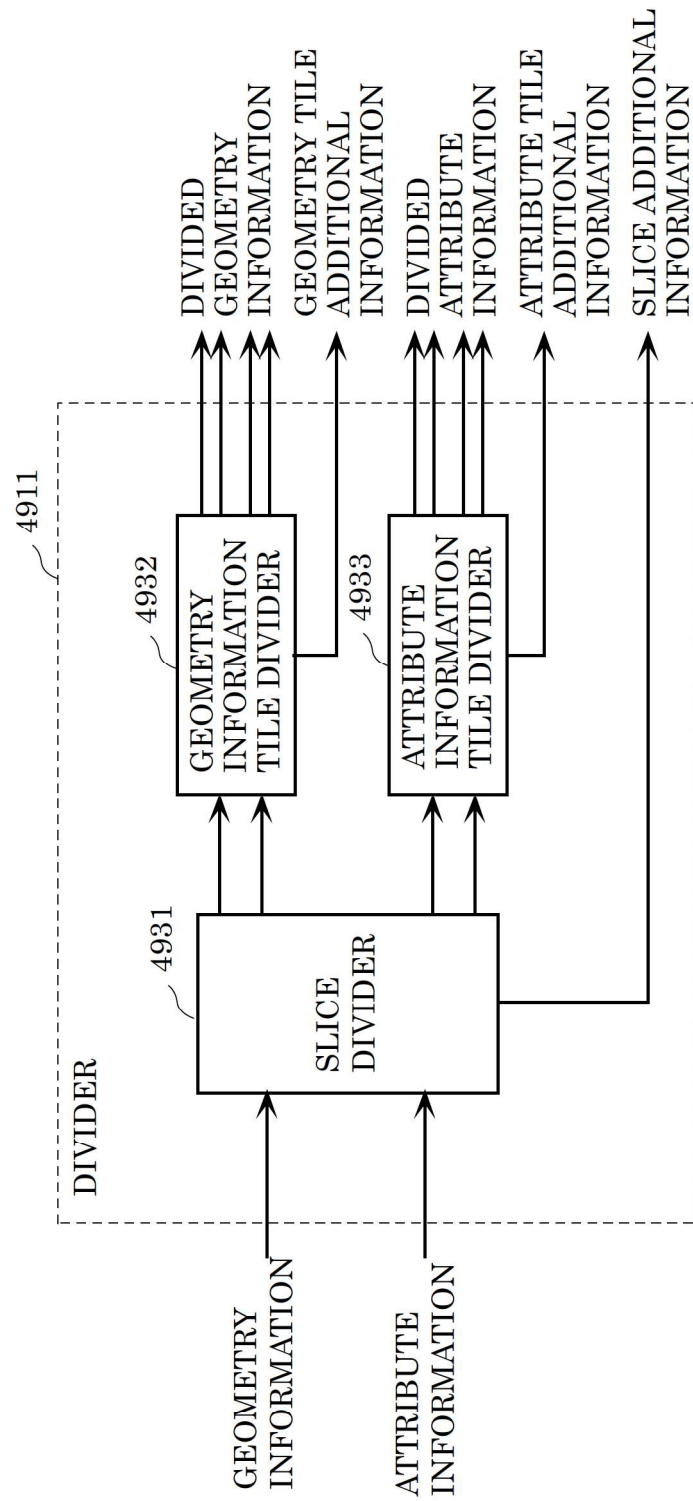


FIG. 24

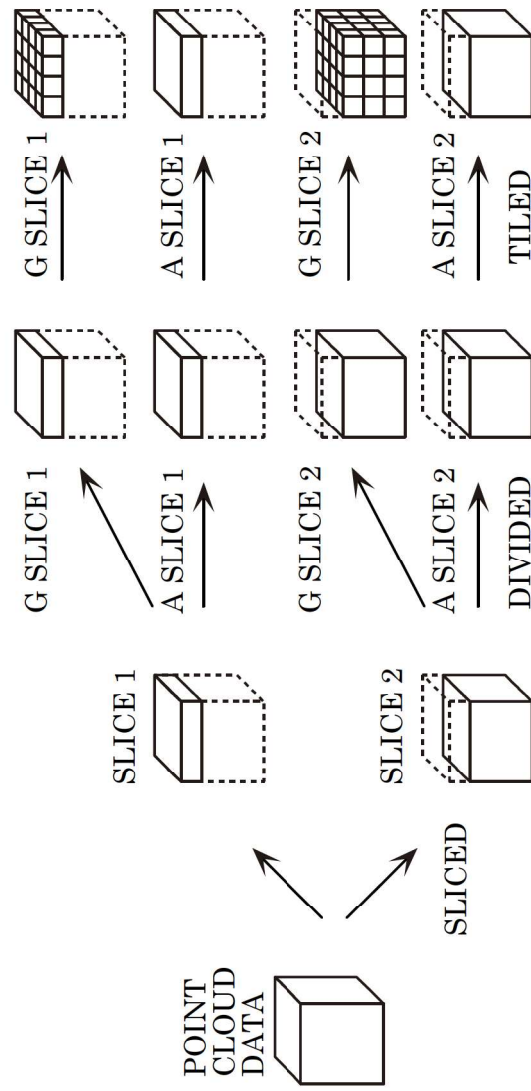


FIG. 25

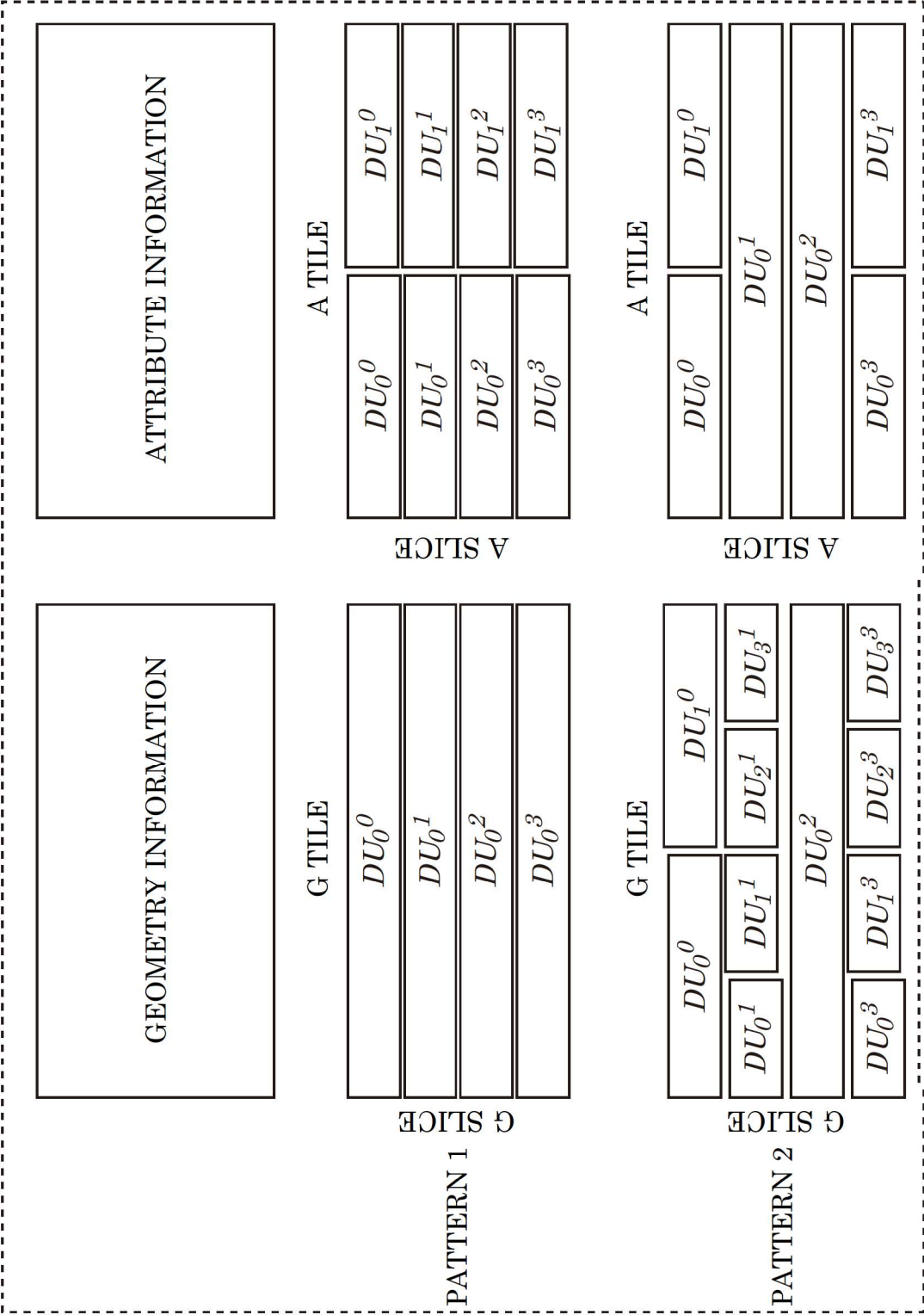


FIG. 26

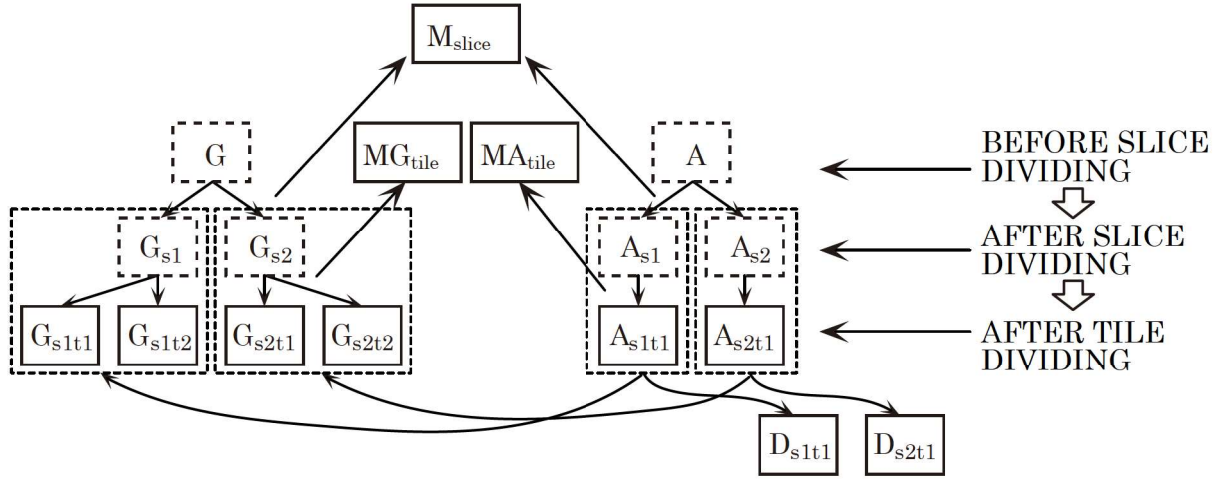


FIG. 27

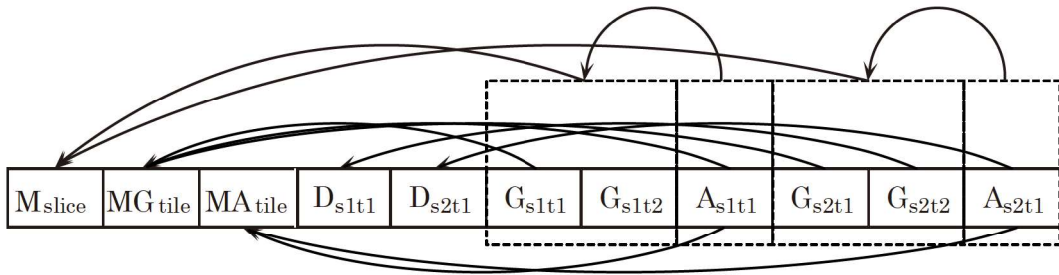


FIG. 28

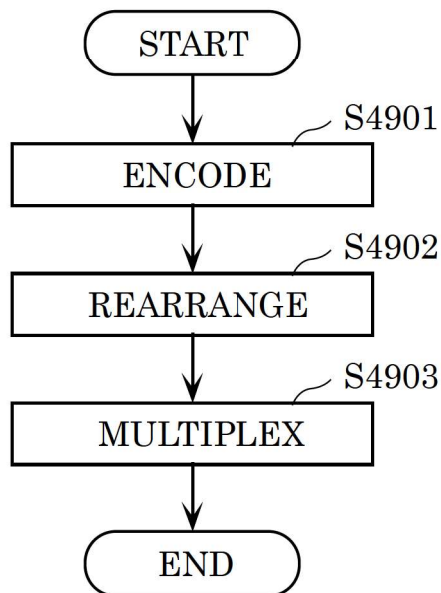


FIG. 29

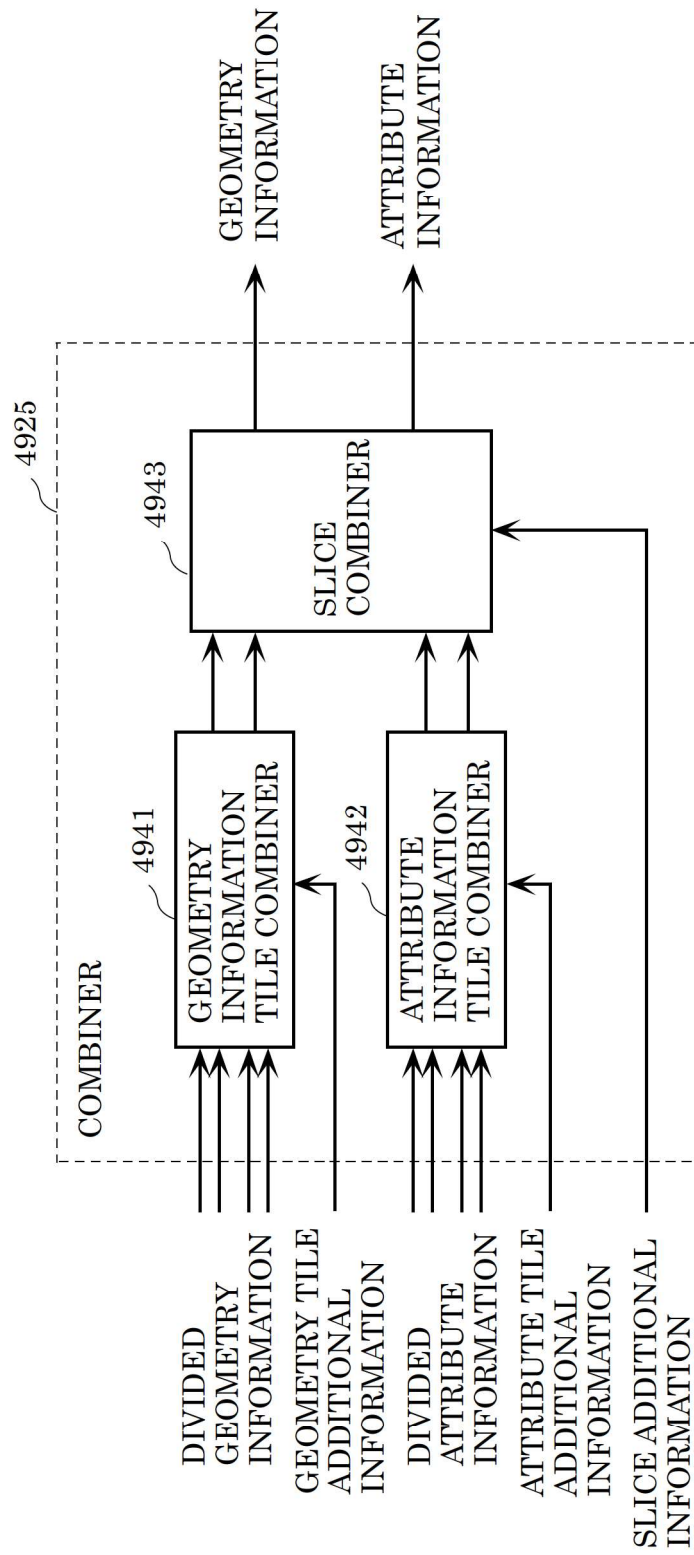


FIG. 30

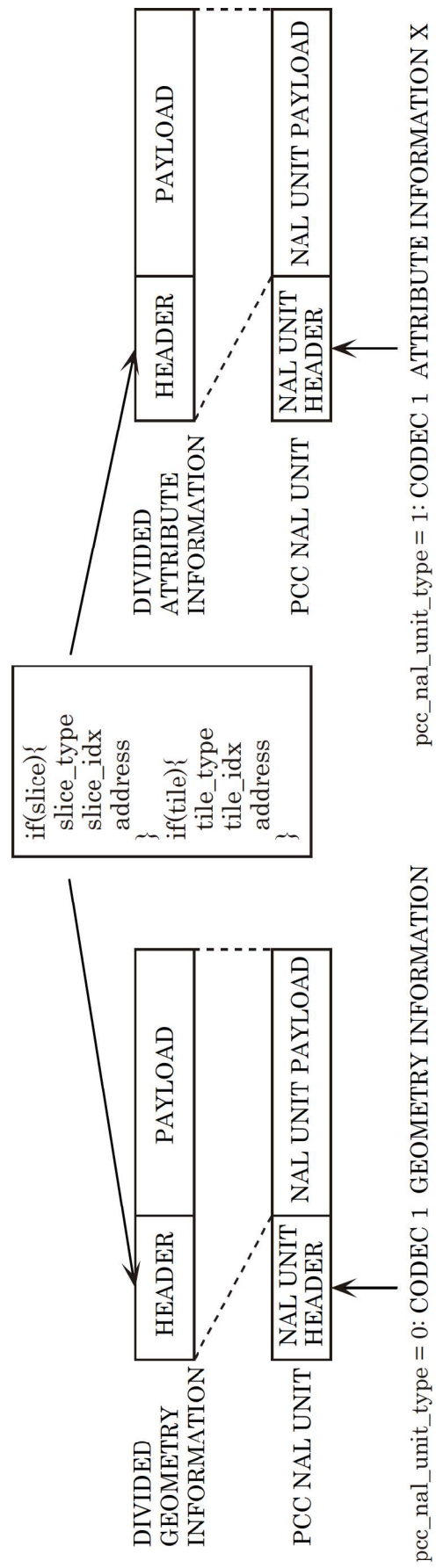


FIG. 31

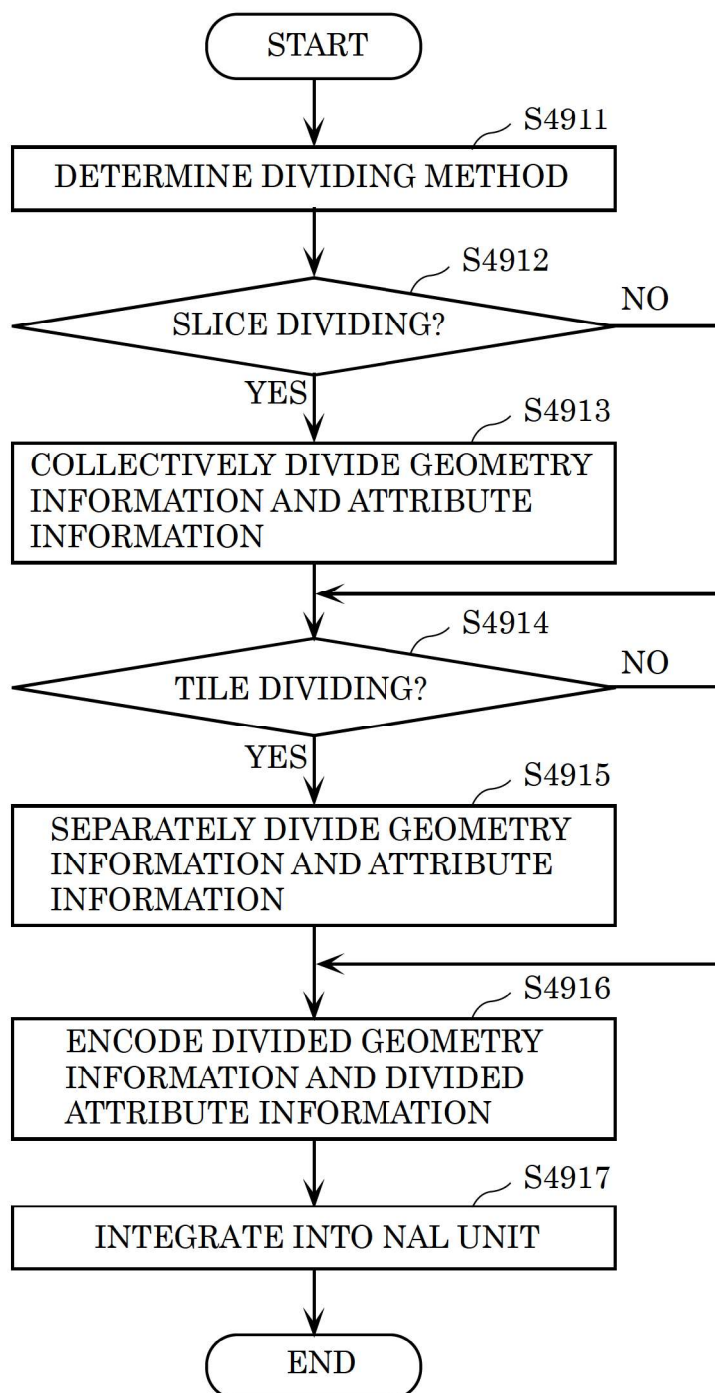


FIG. 32

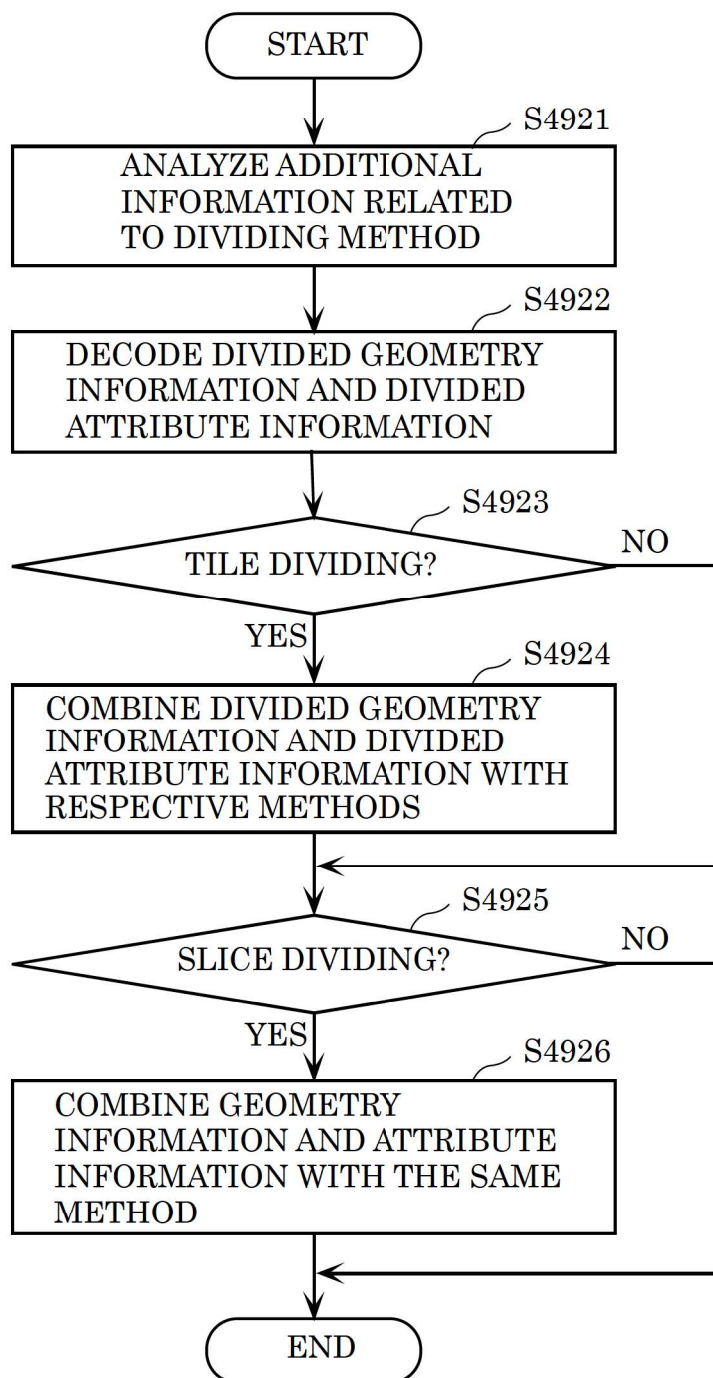


FIG. 33

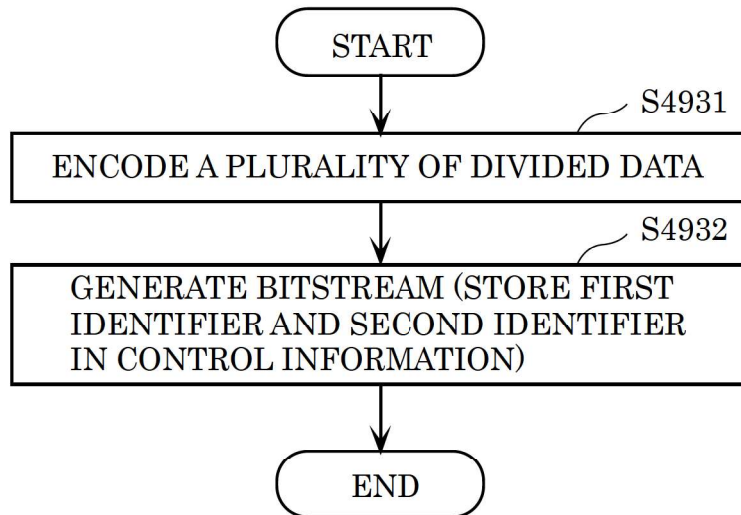


FIG. 34

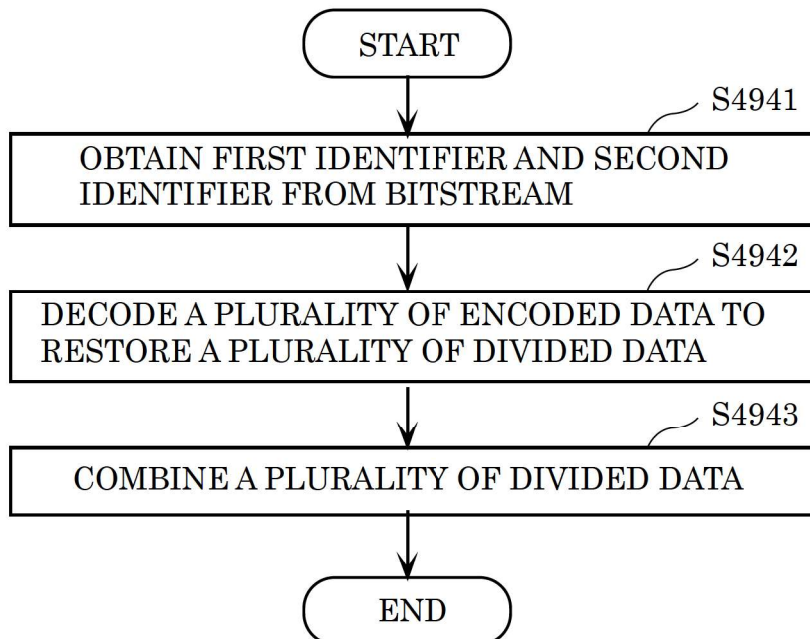


FIG. 35

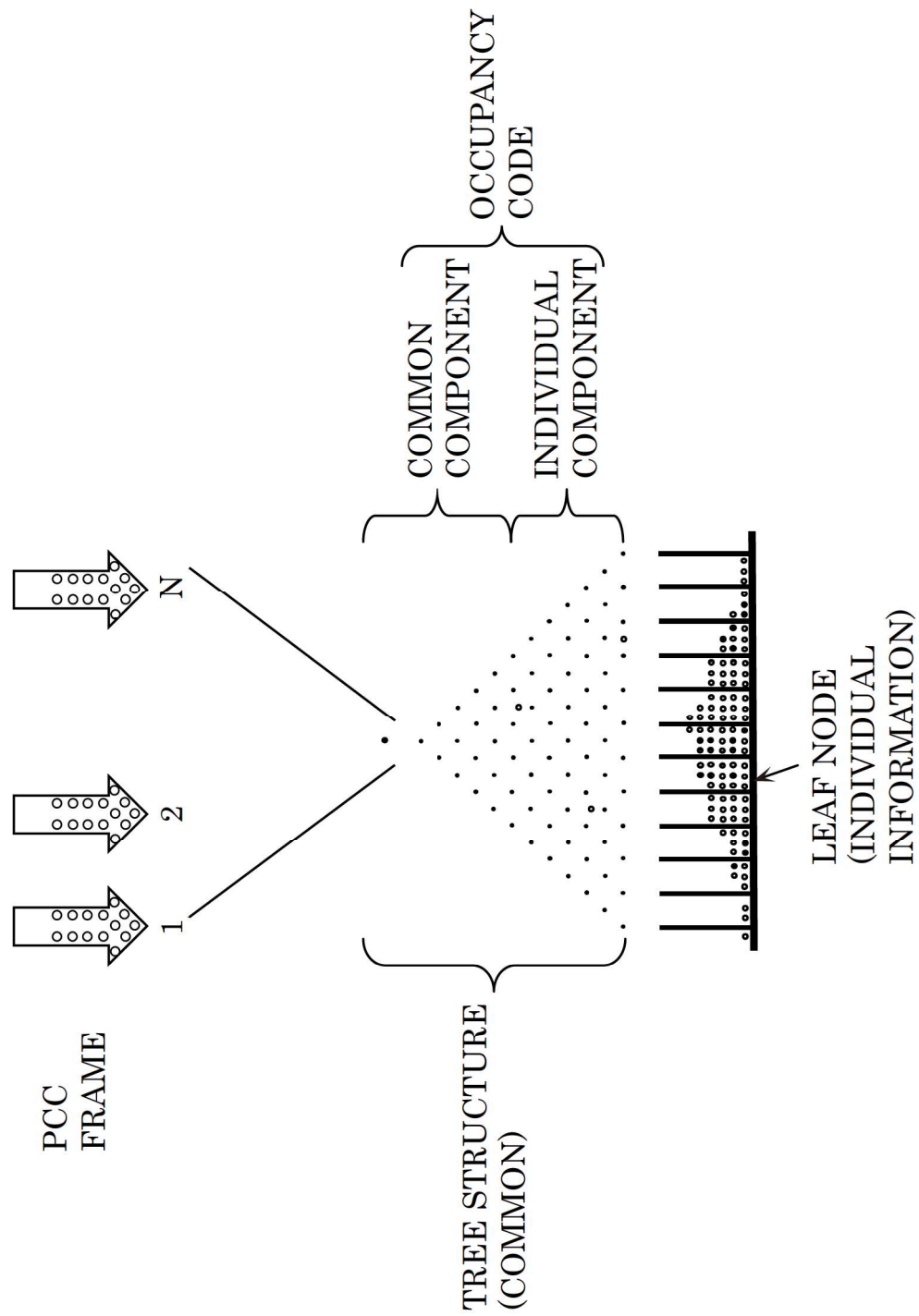


FIG. 36

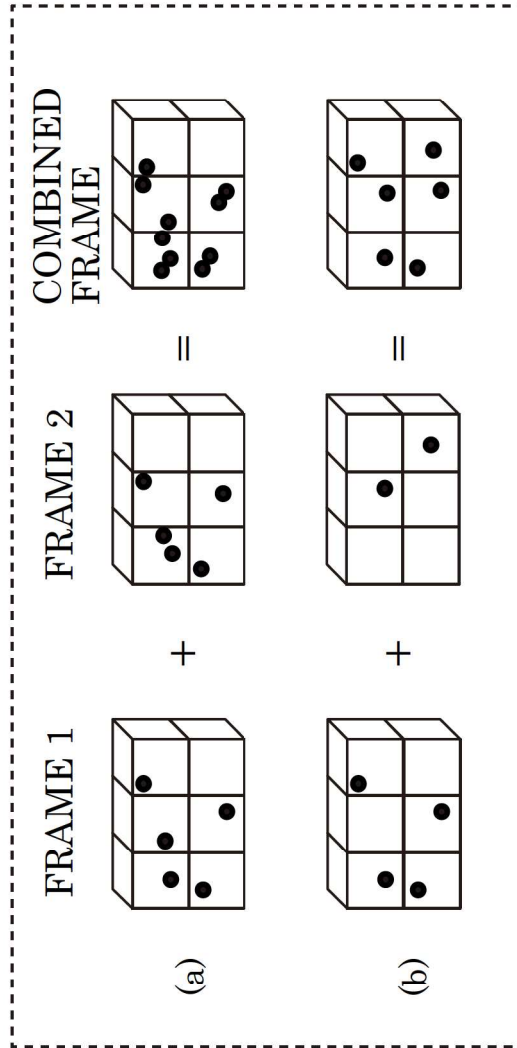


FIG. 37

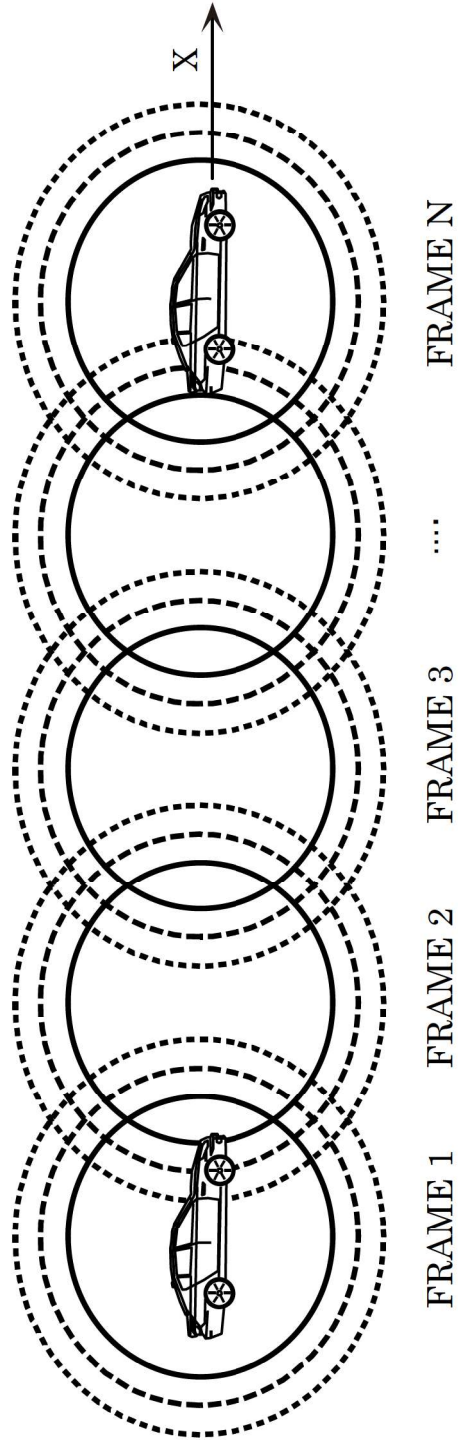


FIG. 38

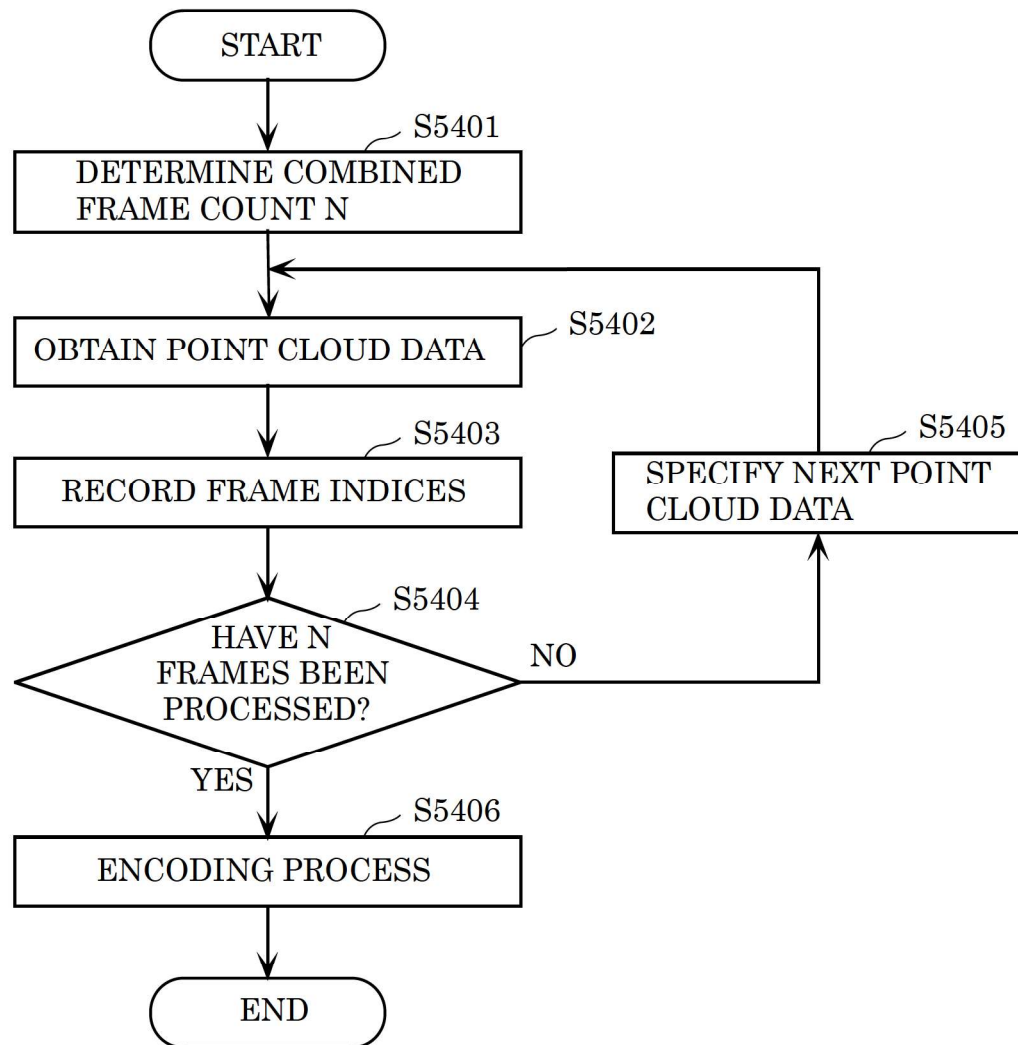


FIG. 39

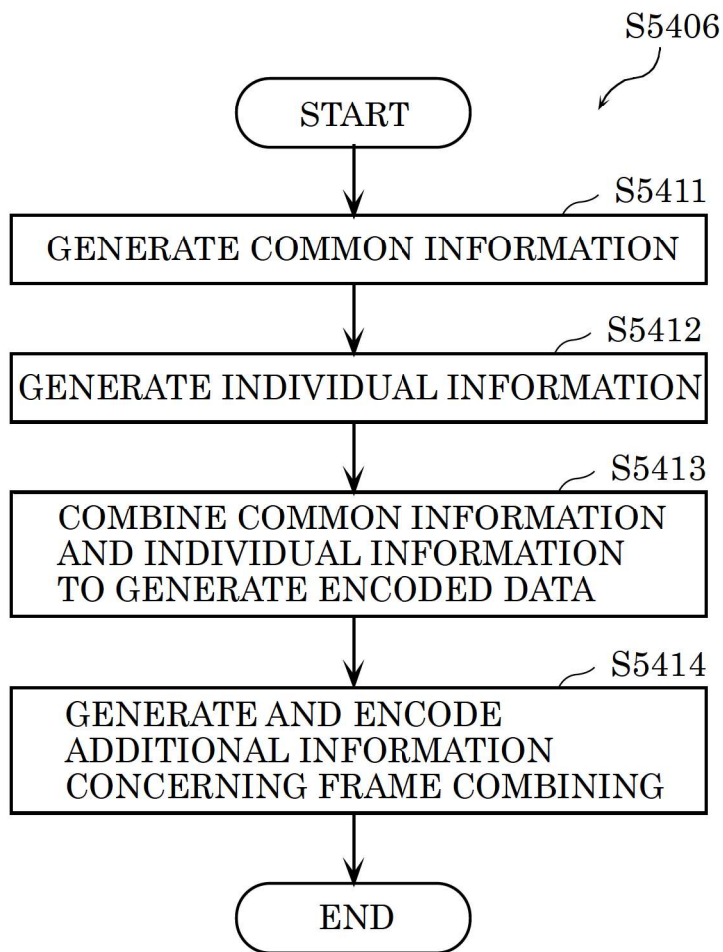


FIG. 40

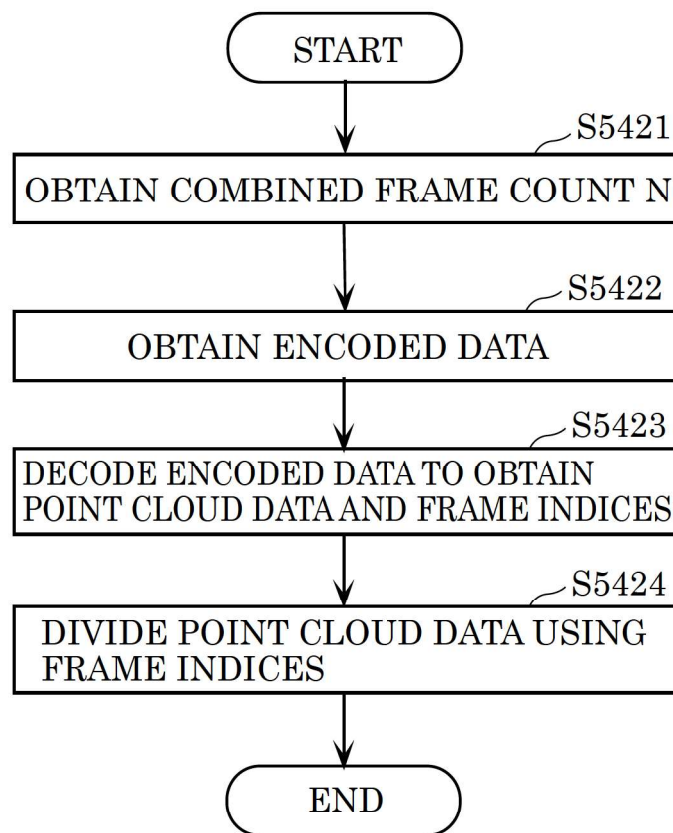


FIG. 41

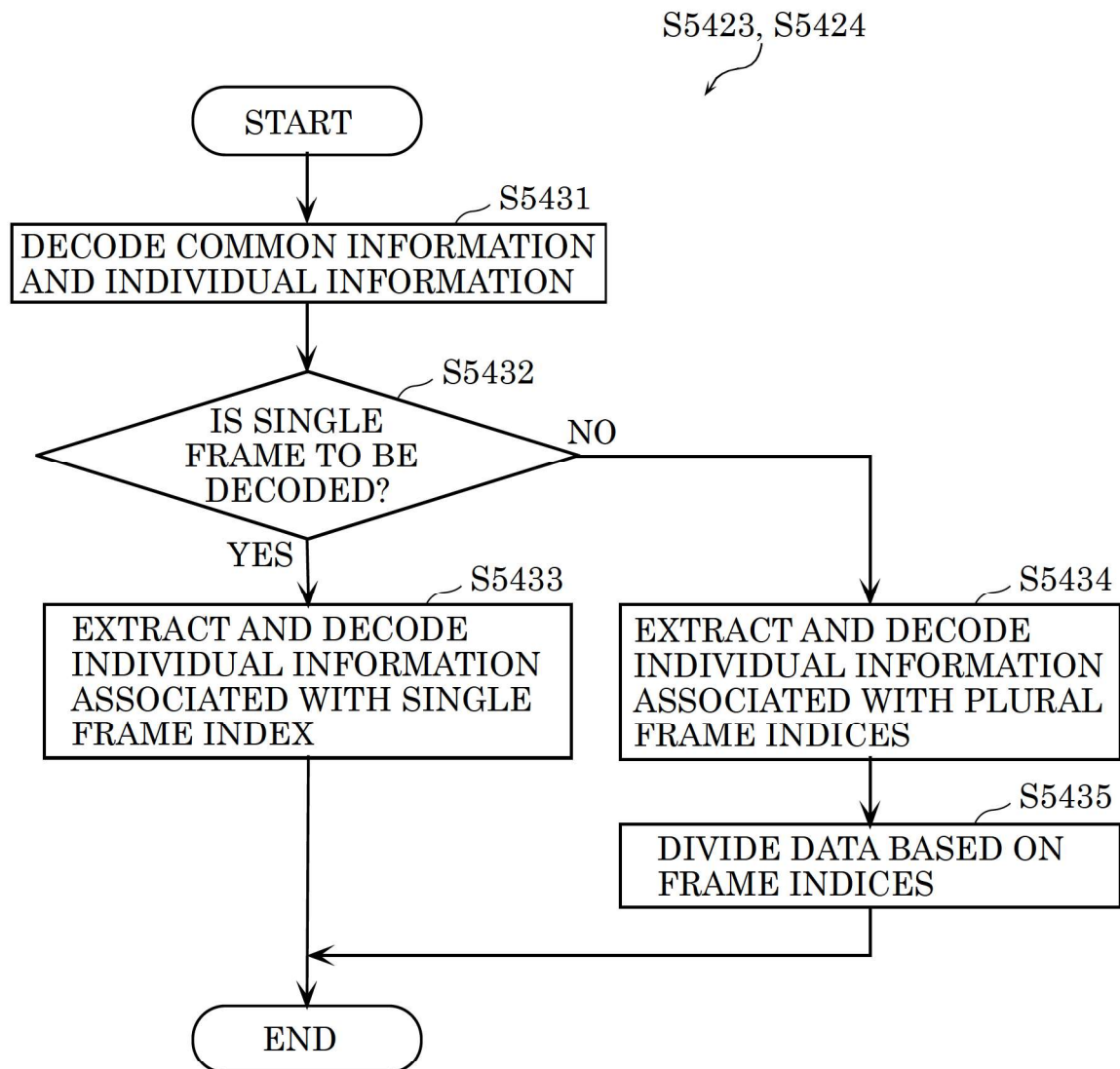


FIG. 42

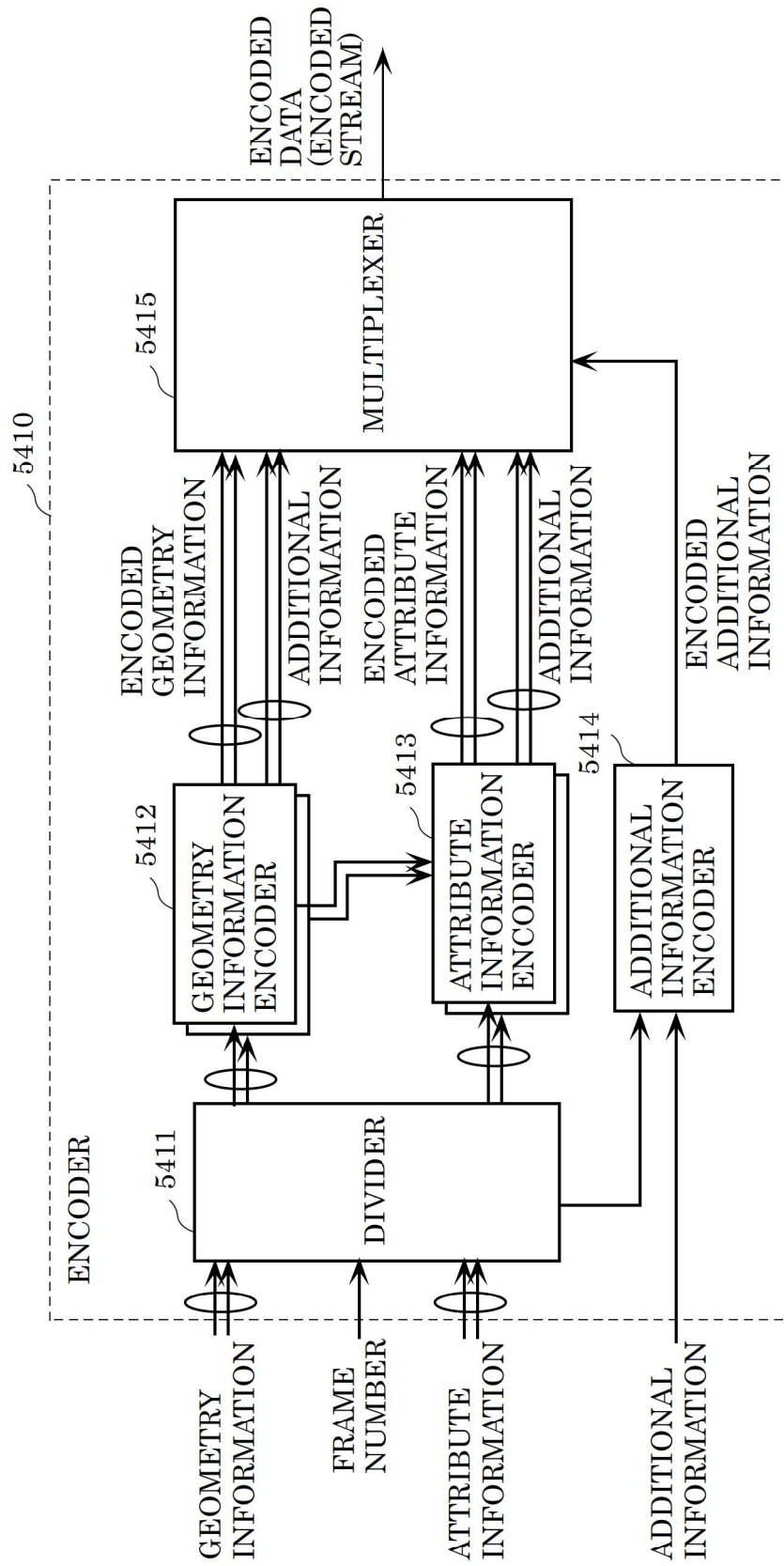


FIG. 43

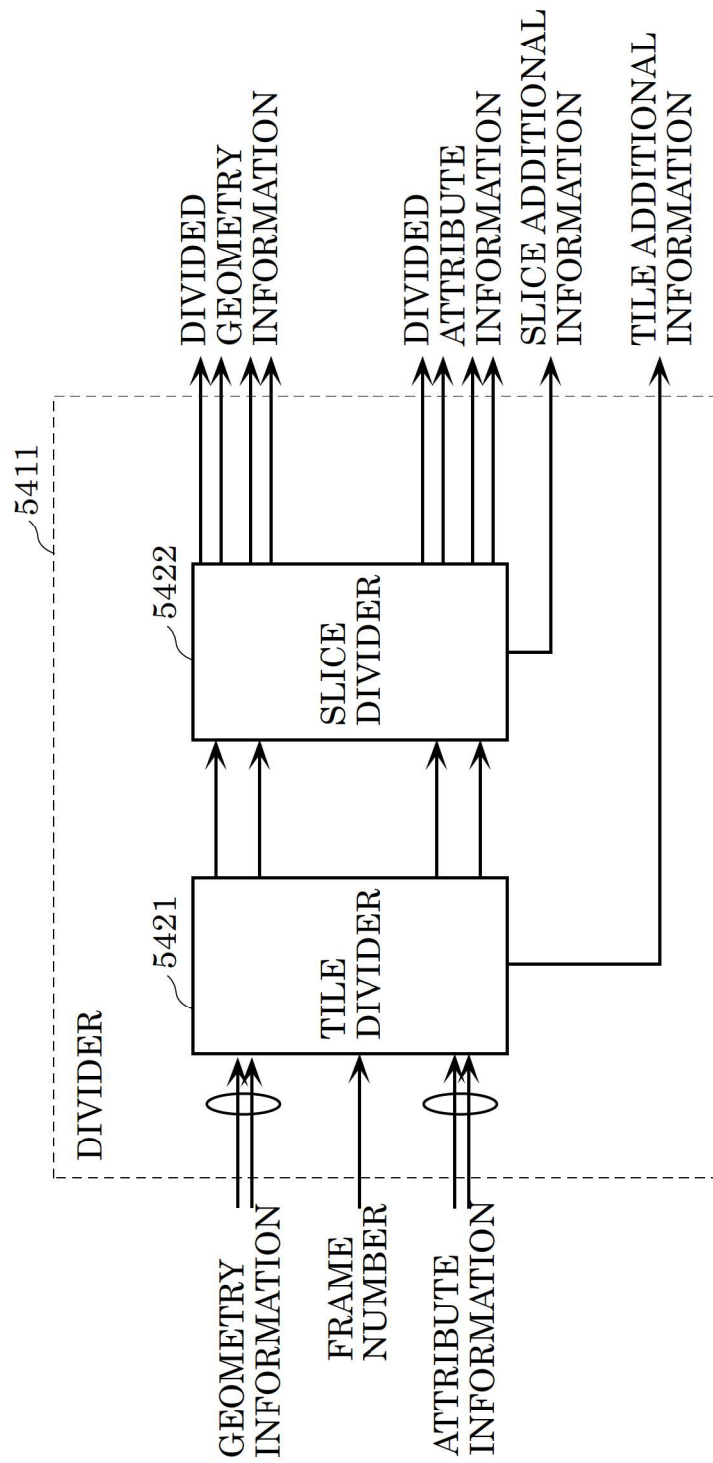


FIG. 44

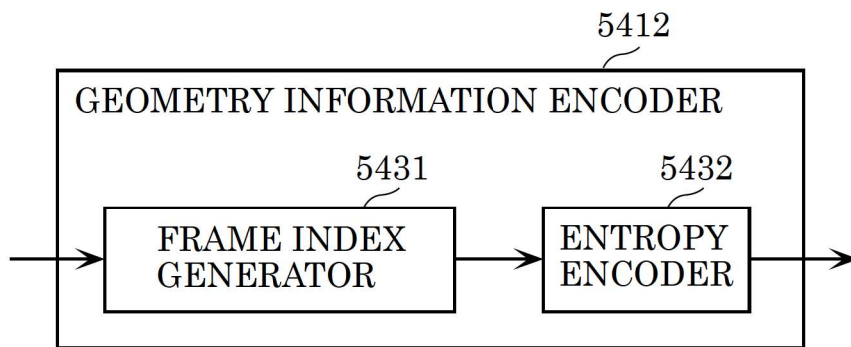


FIG. 45

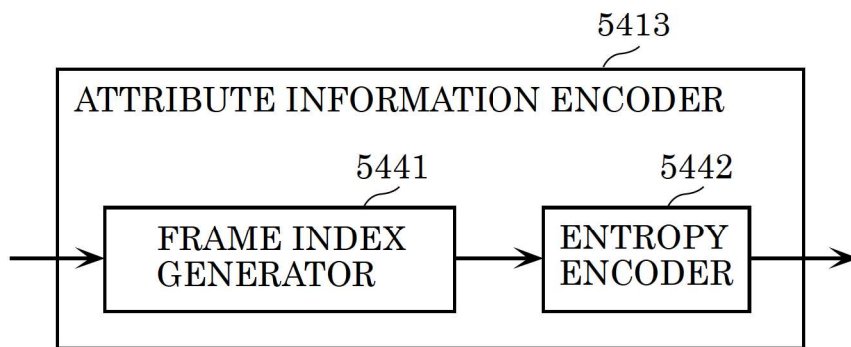


FIG. 46

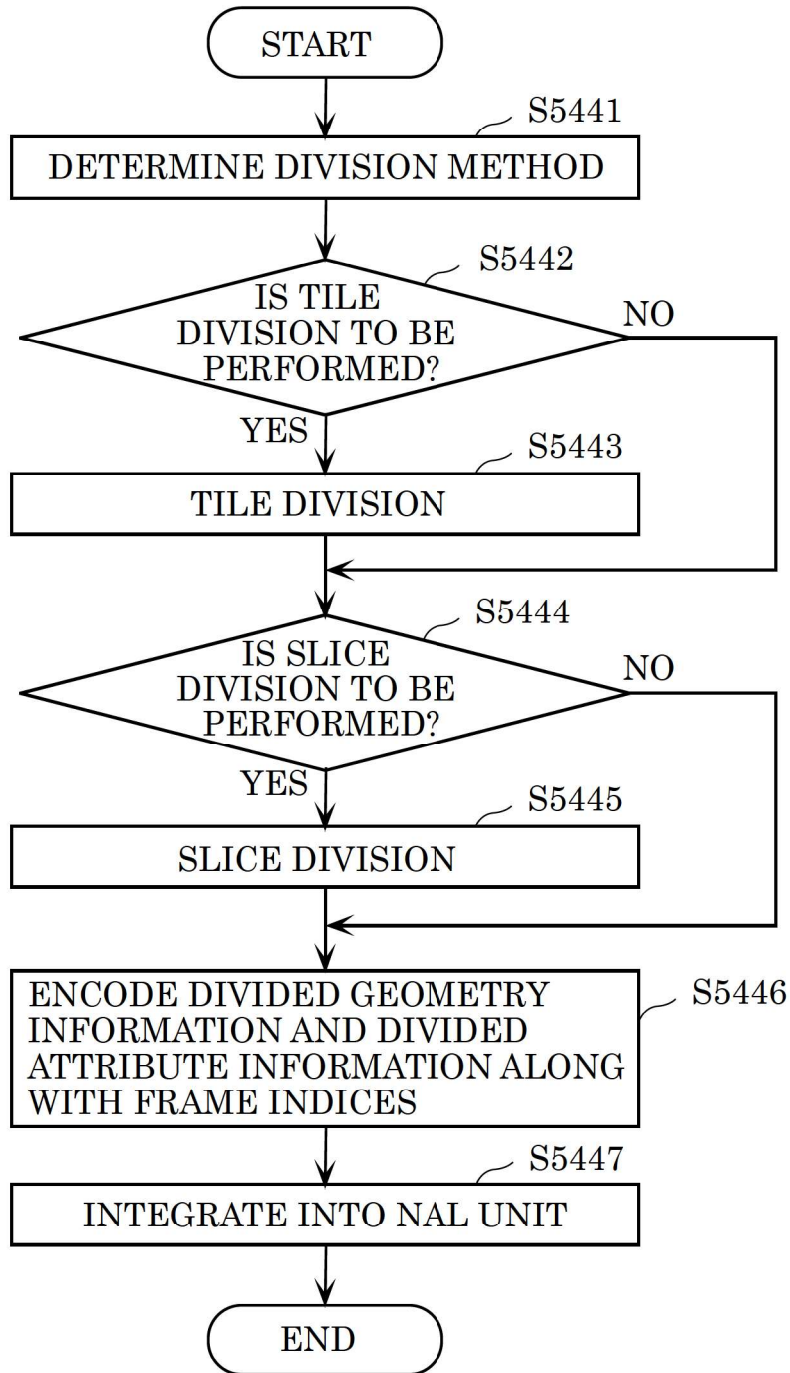


FIG. 47

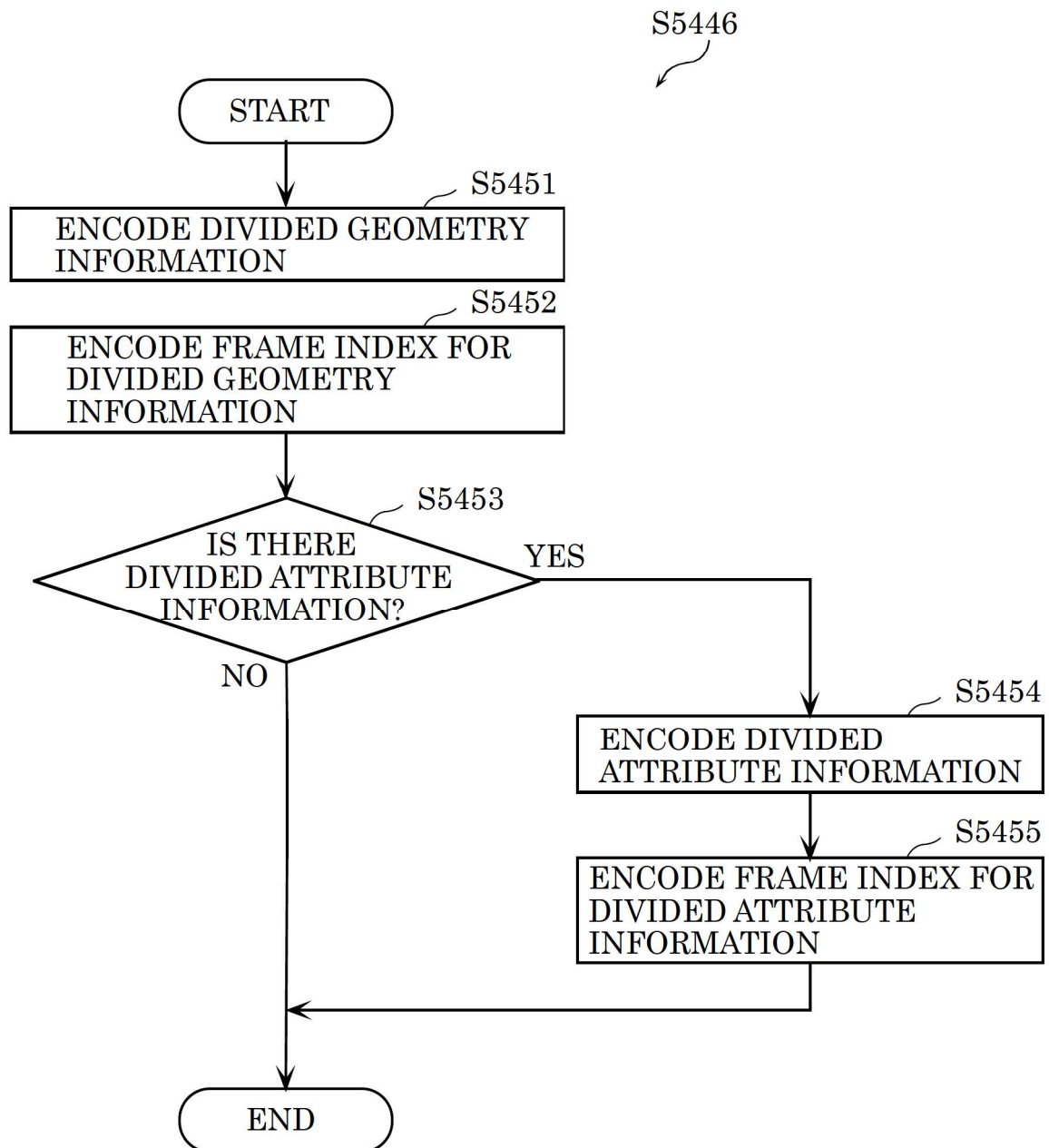


FIG. 48

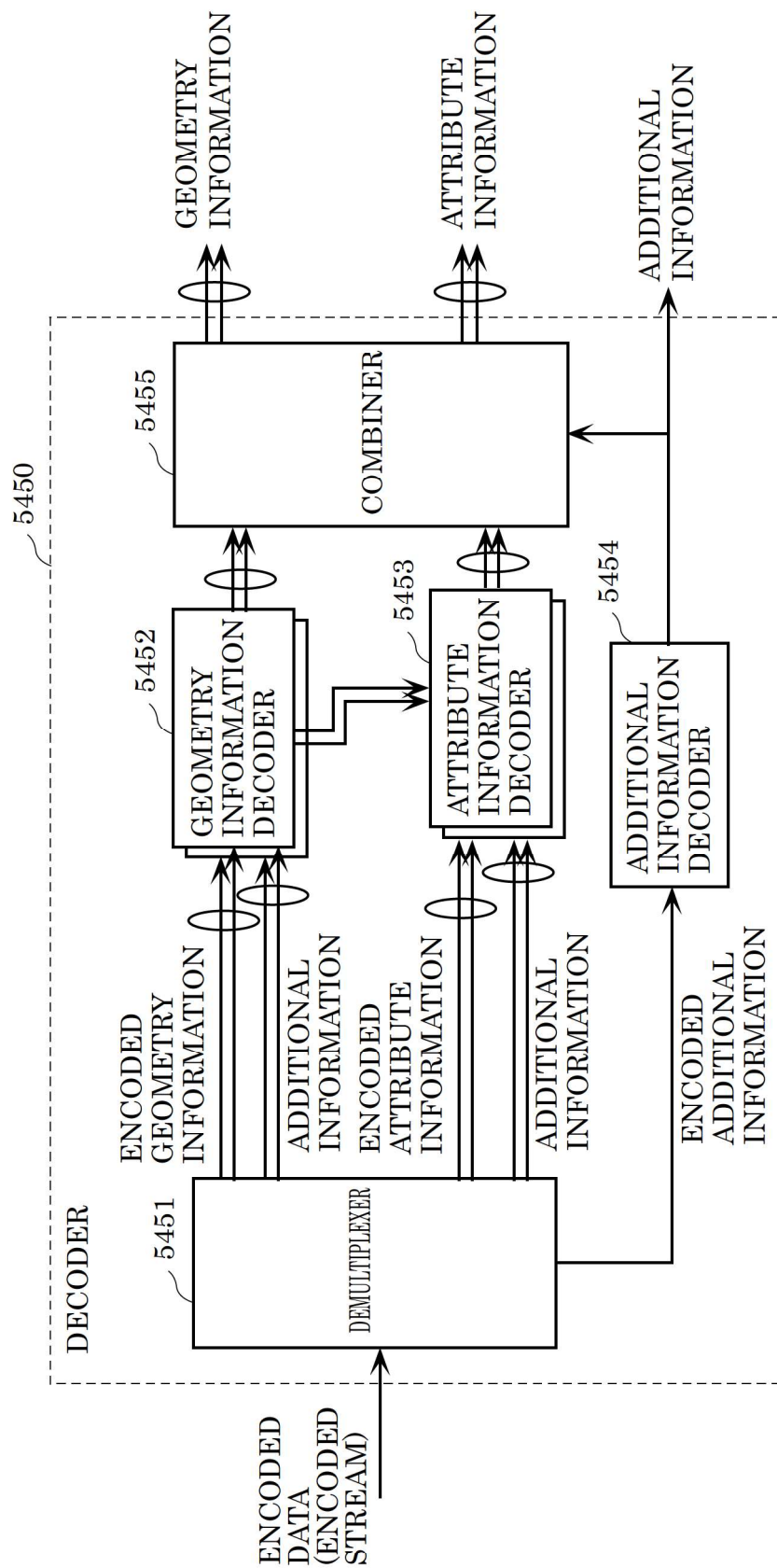


FIG. 49

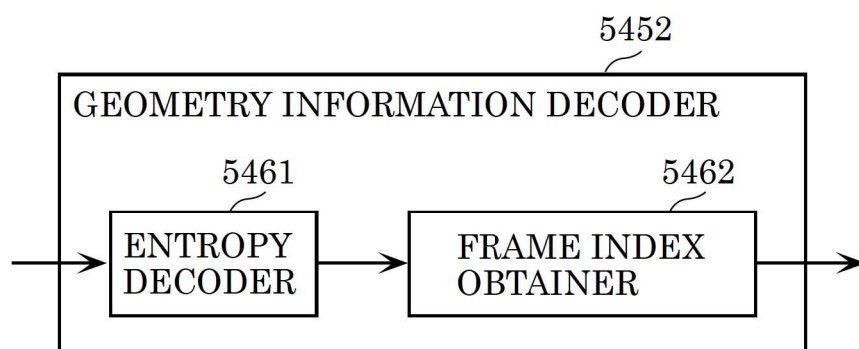


FIG. 50

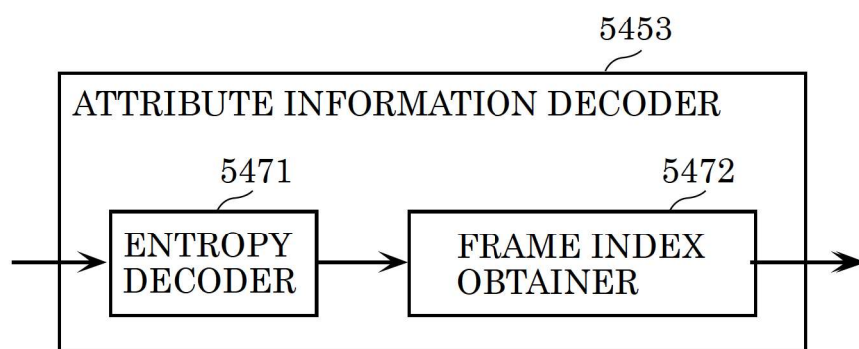


FIG. 51

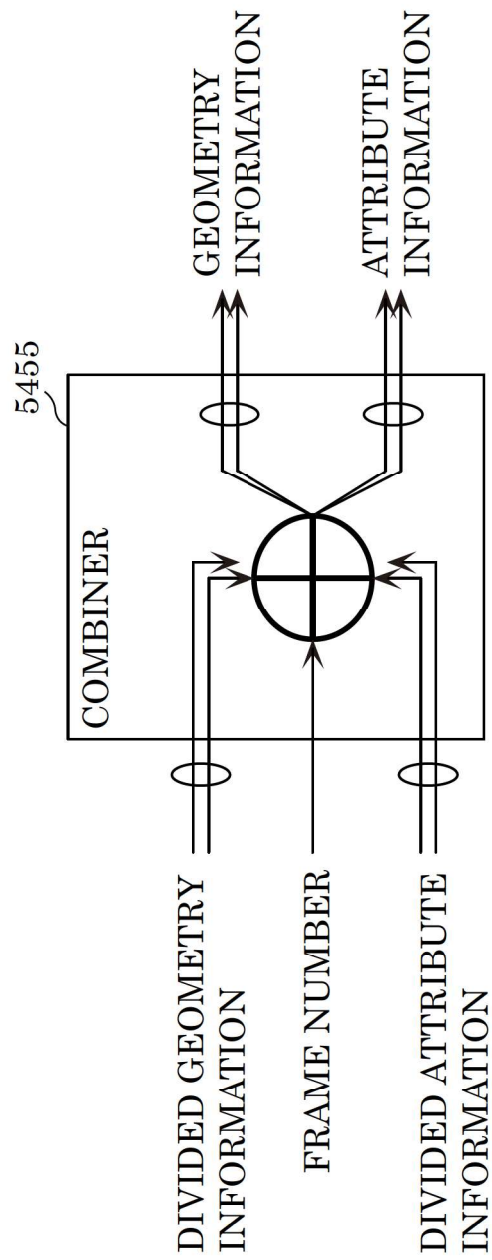


FIG. 52

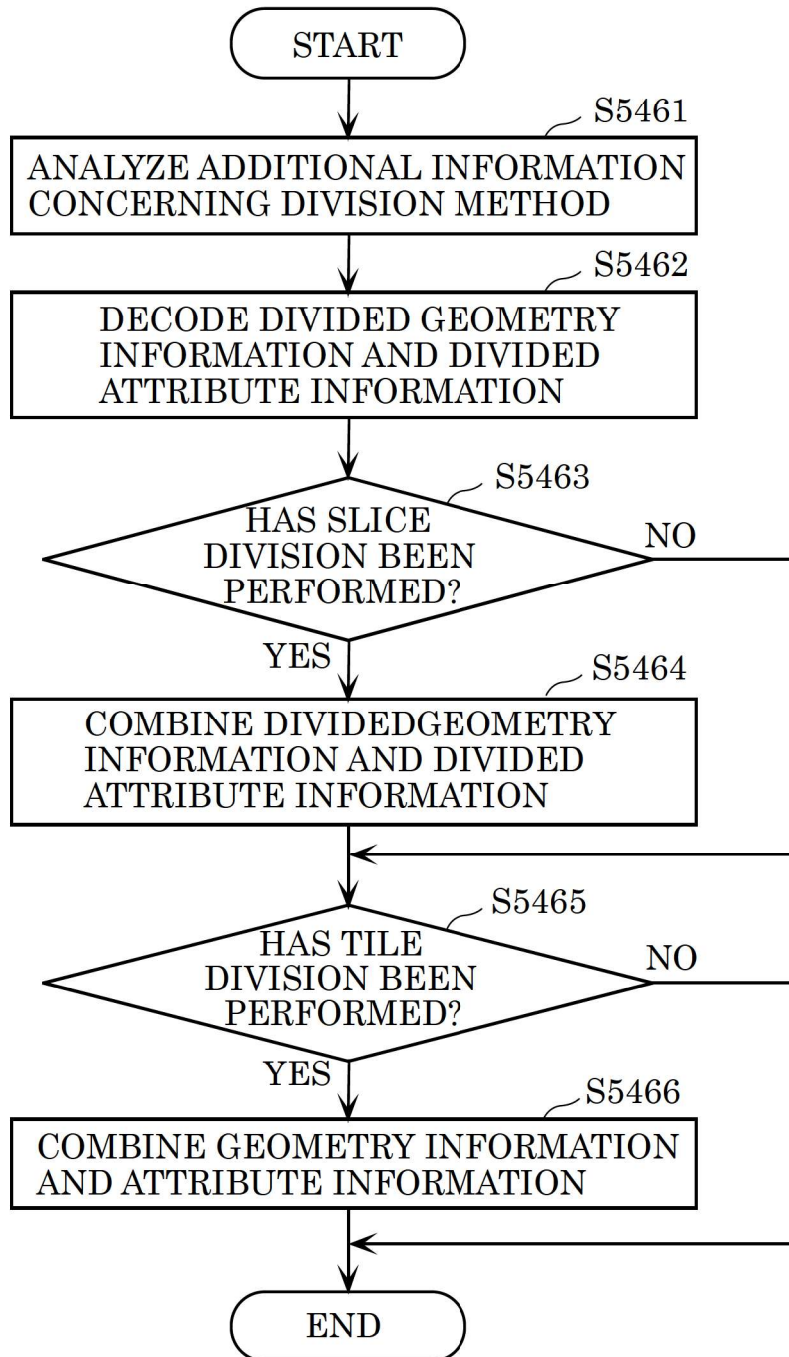


FIG. 53

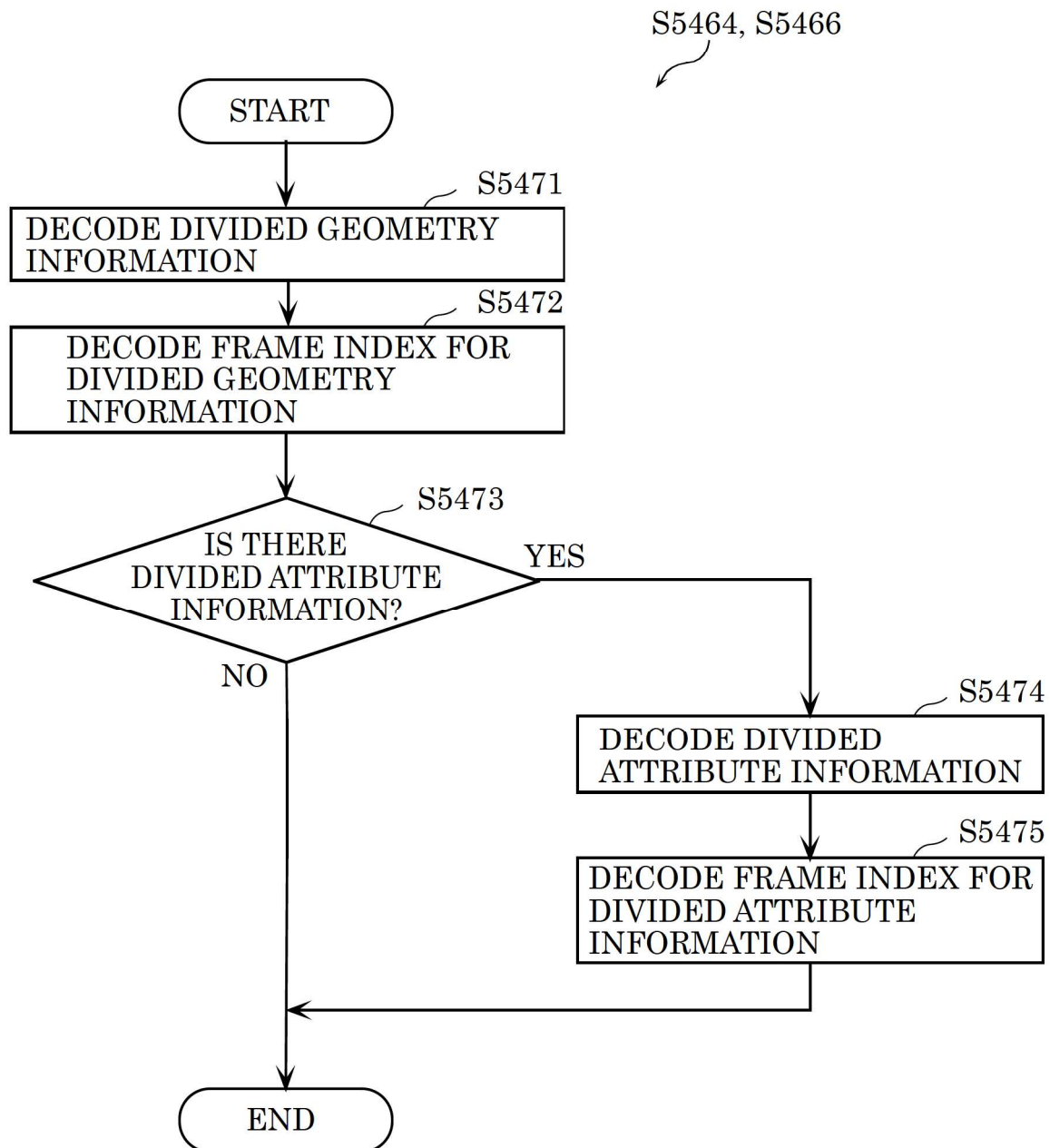


FIG. 54

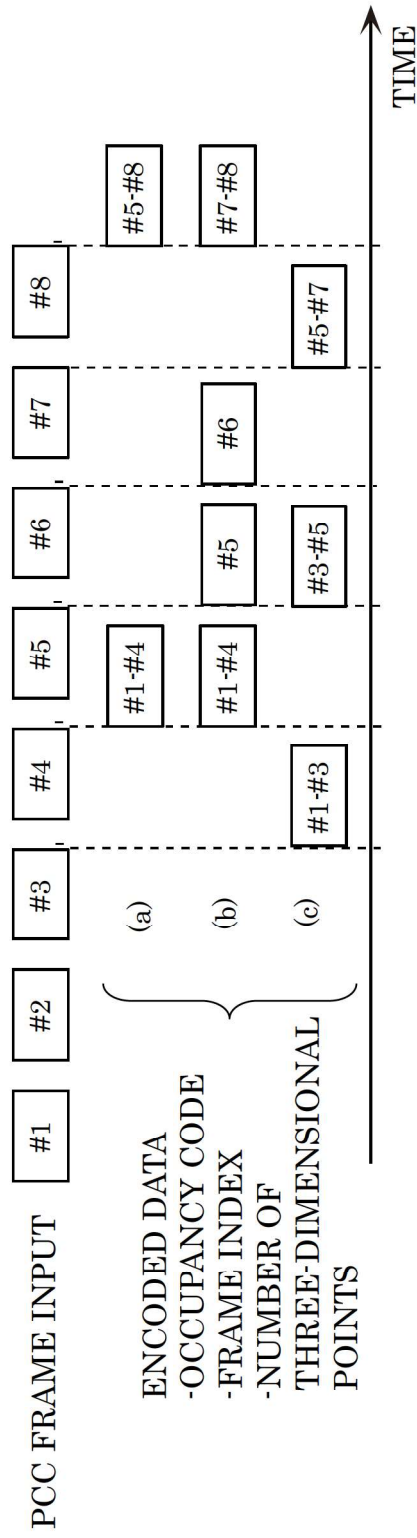


FIG. 55

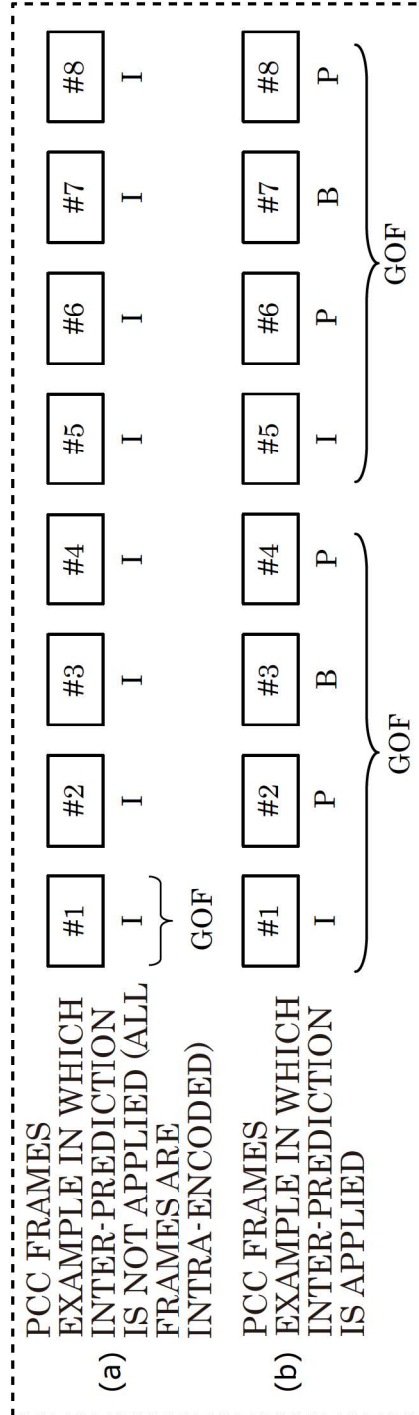


FIG. 56

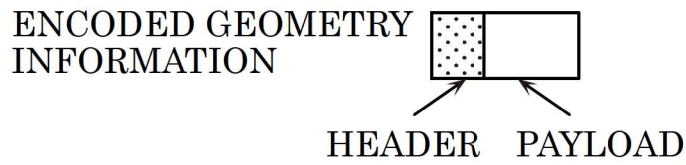


FIG. 57

```

Geometry_header0{
    gps_idx
    offset
    other_geometry_information
    if(combine_frame_flag){
        number_of_combine_frame
    }
    if(tile){
        tile_type
        tile_idx
    }
    if(slice){
        slice_type
        slice_idx
    }
}
    
```

FIG. 58

```

Geometry_data0{
    for(depth<maxDepth){
        for(i<MaxNodes){
            occupancy_Code(depth, i)
            node_information(depth, i){
                if(i==MaxNodes-1){
                    combine_information();
                }
            }
        }
    }
}
    
```

FIG. 59

```
combine_information(){  
    NumberOfPoints  
    for(i<NumberOfPoints){  
        FrameIndex (i)  
    }  
}
```

FIG. 60

```
combine_information(){  
    bitmapIsFramePointsFlag  
}
```

FIG. 61

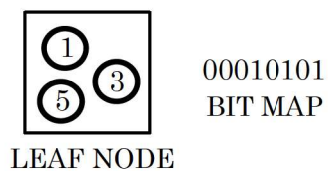


FIG. 62

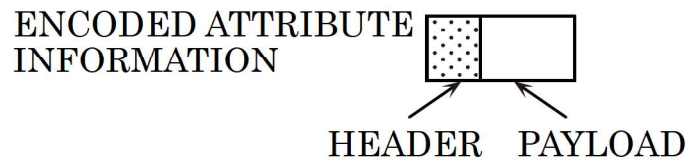


FIG. 63

```
Attribute_header(){  
    aps_idx  
    offset  
    other_attribute_information  
    if(combine_frame_flag){  
        number_of_combine_frame  
    }  
    if(tile){  
        tile_type  
        tile_idx  
    }  
    if(slice){  
        slice_type  
        slice_idx  
    }  
}
```

FIG. 64

```
Attribute_data(){  
    combine_information();  
}
```

FIG. 65

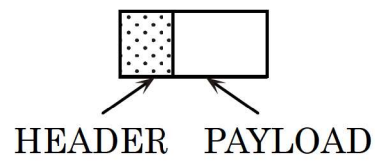


FIG. 66

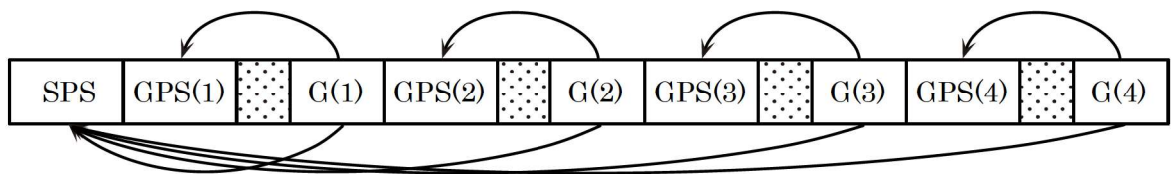


FIG. 67

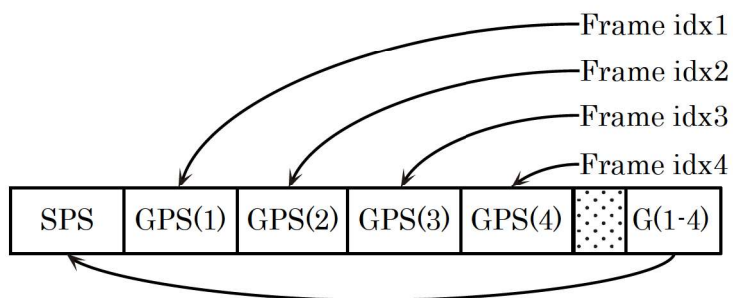


FIG. 68

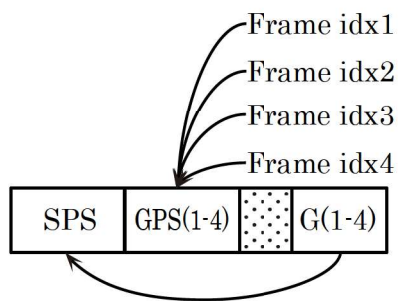


FIG. 69

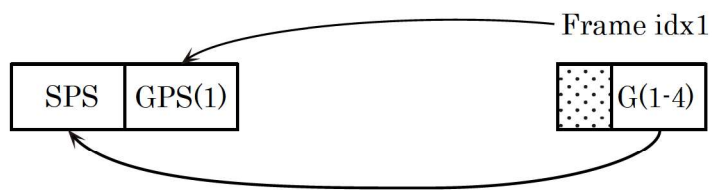


FIG. 70

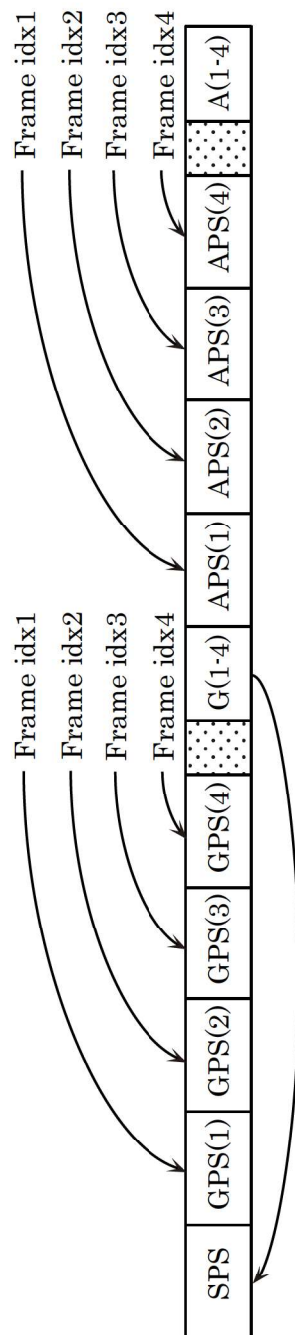


FIG. 71

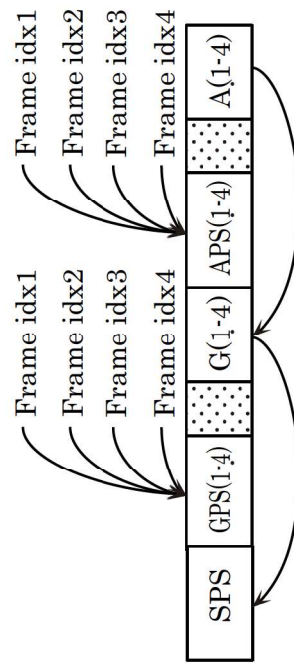


FIG. 72

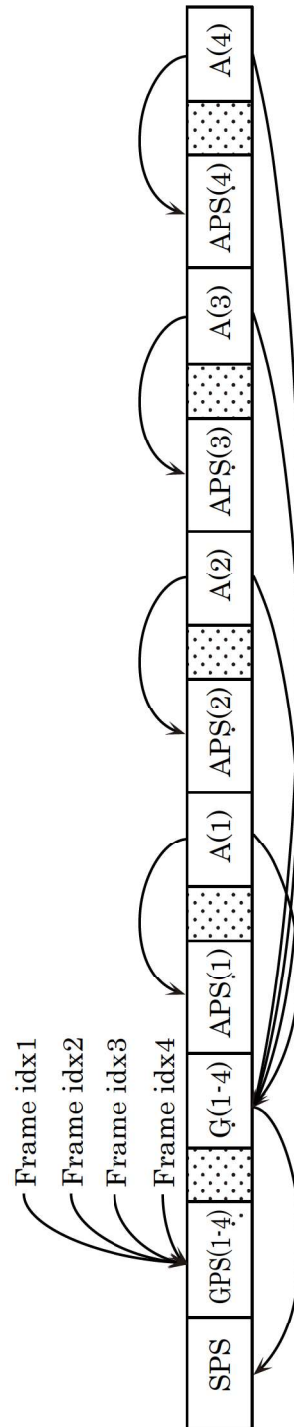


FIG. 73

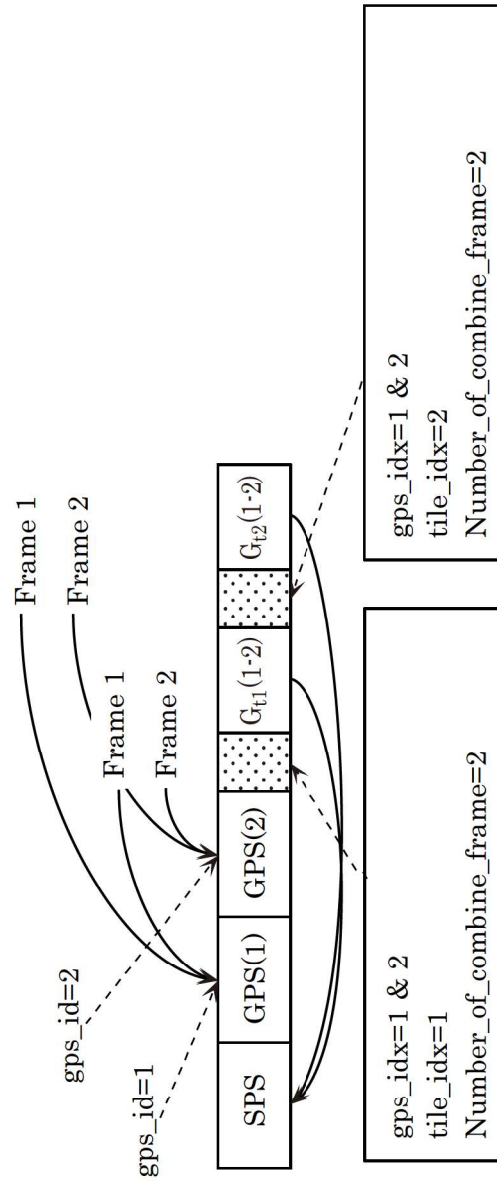


FIG. 74

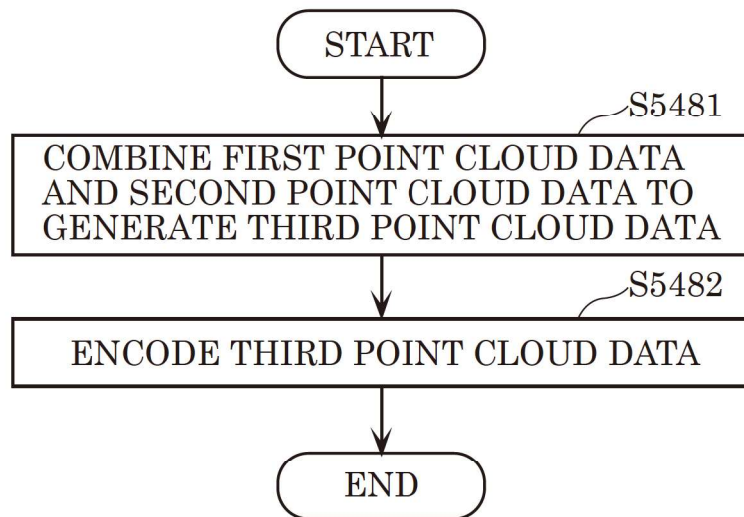


FIG. 75

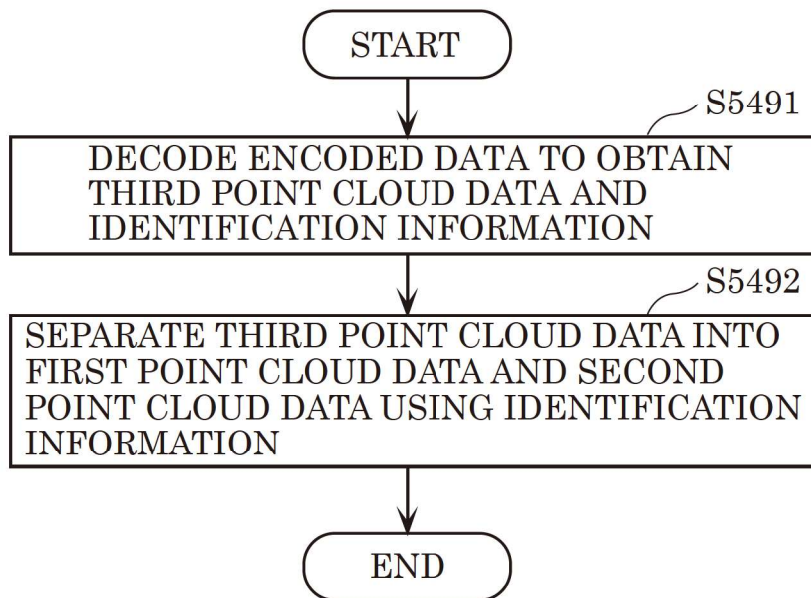


FIG. 76

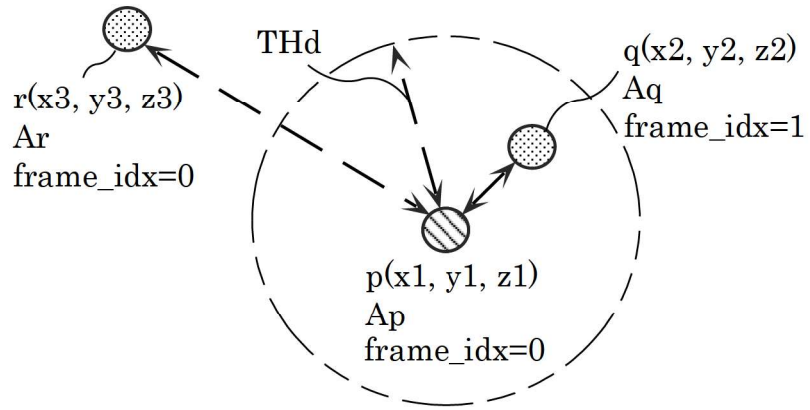


FIG. 77

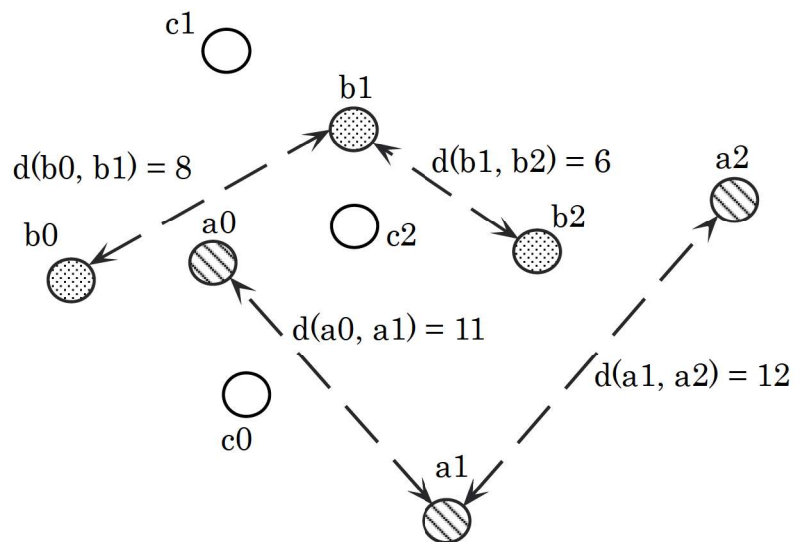


FIG. 78

a0	a1	a2	LoD0	Thres_LoD[0] = 10	SPARSE
b0	b1	b2	LoD1	Thres_LoD[1] = 5	
c0	c1	c2	LoD2	Thres_LoD[2] = 0	DENSE

FIG. 79

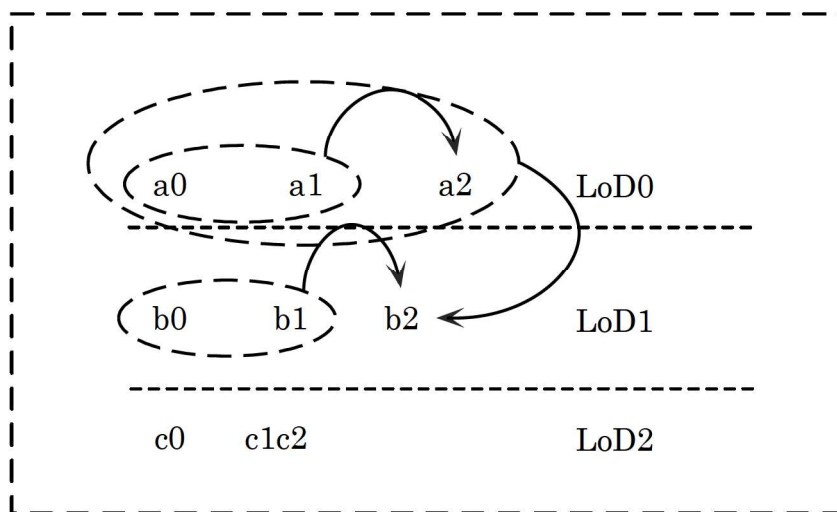


FIG. 80

VALUE	BIT
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
...	...

FIG. 81

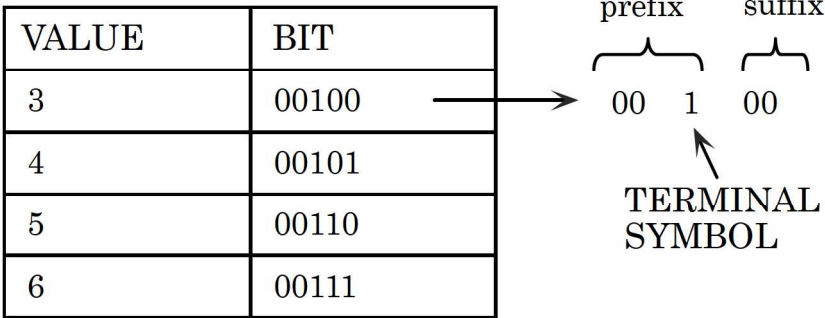


FIG. 82

```

attribute_header{
    ...
    NumLoD
    ...
    for (i=0; i<NumLoD; i++) {
        NumOfPoint[i]
    }
    ...
    for (i=0; i<NumLoD-1; i++) {
        Thres_Lod[i]
    }
    ...
    for (i=0; i<NumLoD; i++) {
        NumNeighborPoint[i]
    }
    ...
    for (i=0; i<NumLoD; i++) {
        THd[i]
    }
    ...
    for (i=0; i<NumLoD; i++) {
        QS[i]
    }
    ...
    for (i=0; i<NumLoD; i++) {
        R_TH[i]
    }
    ...
}

```

FIG. 83

```

attribute_data{
    ...
    for (i=0; i<NumLoD; i++) {
        for (j=0; j<NumOfPoint[i]; j++) {
            n-bit code
            if (n-bit code == R_TH[i])
                remaining code
        }
    }
    ...
}

```

FIG. 84

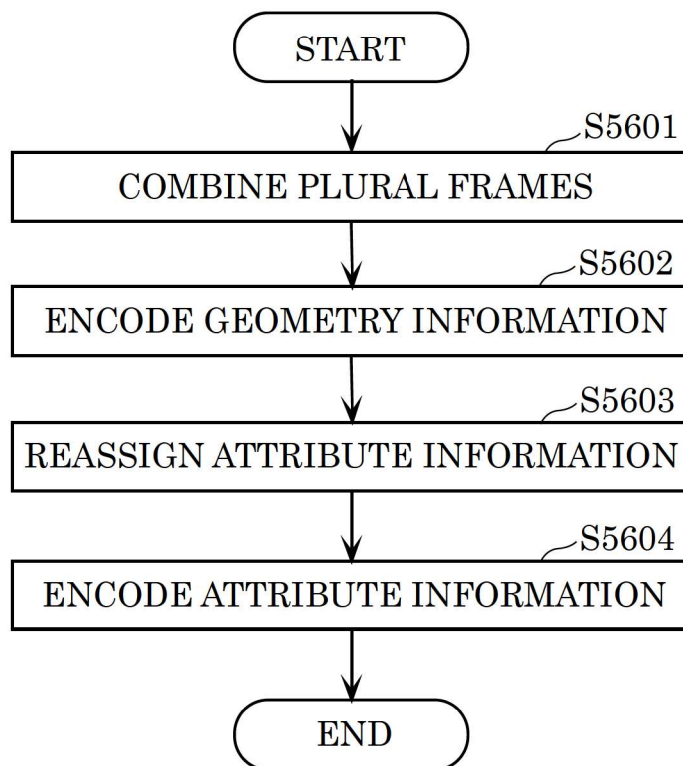


FIG. 85

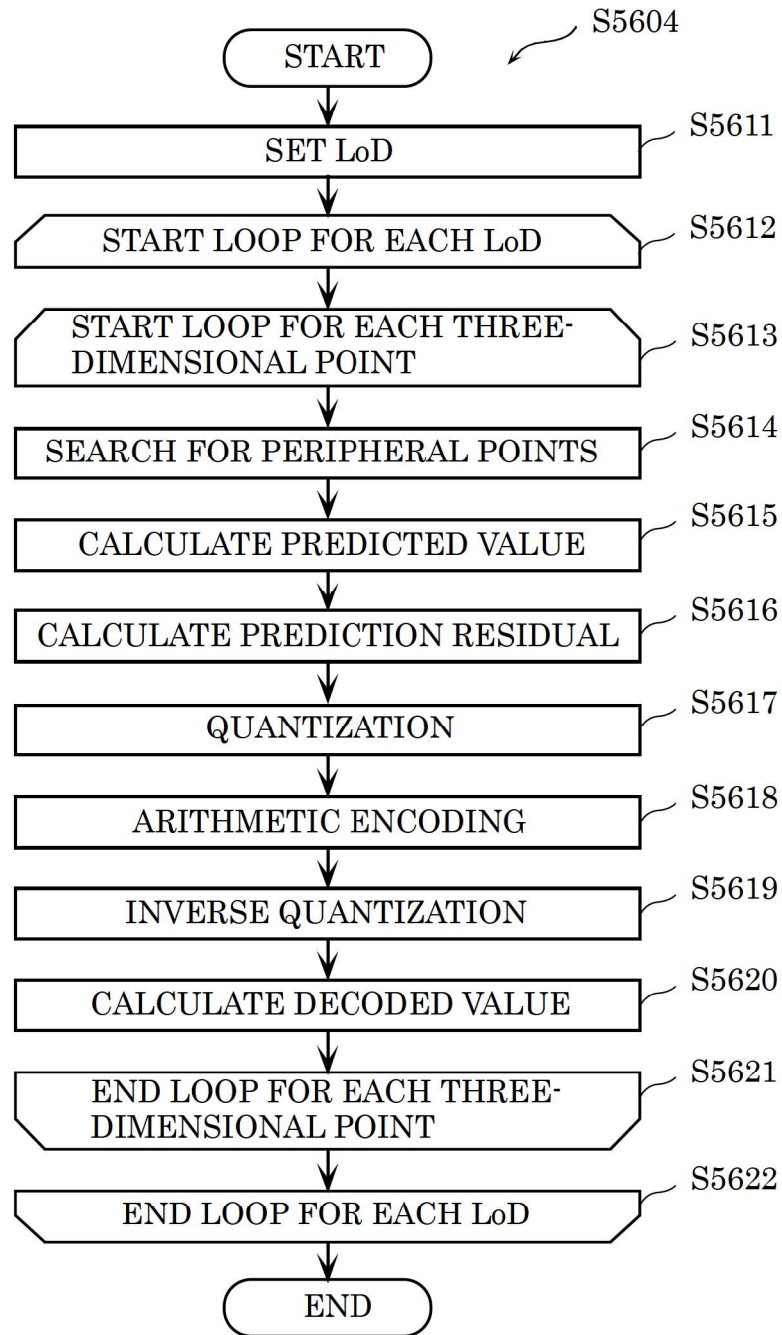


FIG. 86

VALUE	BIT
3	00100
4	00101
5	00110
6	00111

prefix

suffix

00100

←

←

TERMINAL SYMBOL

FIG. 87

VALUE	BIT
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
...	...

←

00100

REVERSE
REFERENCE

FIG. 88

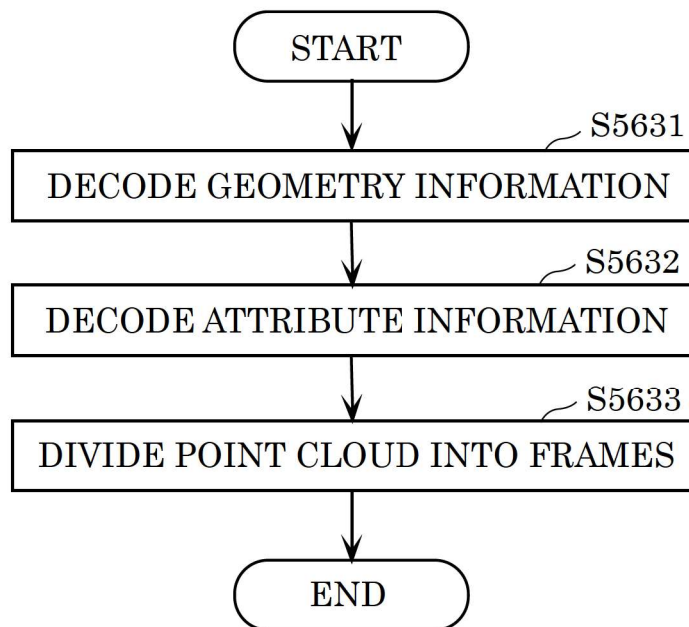


FIG. 89

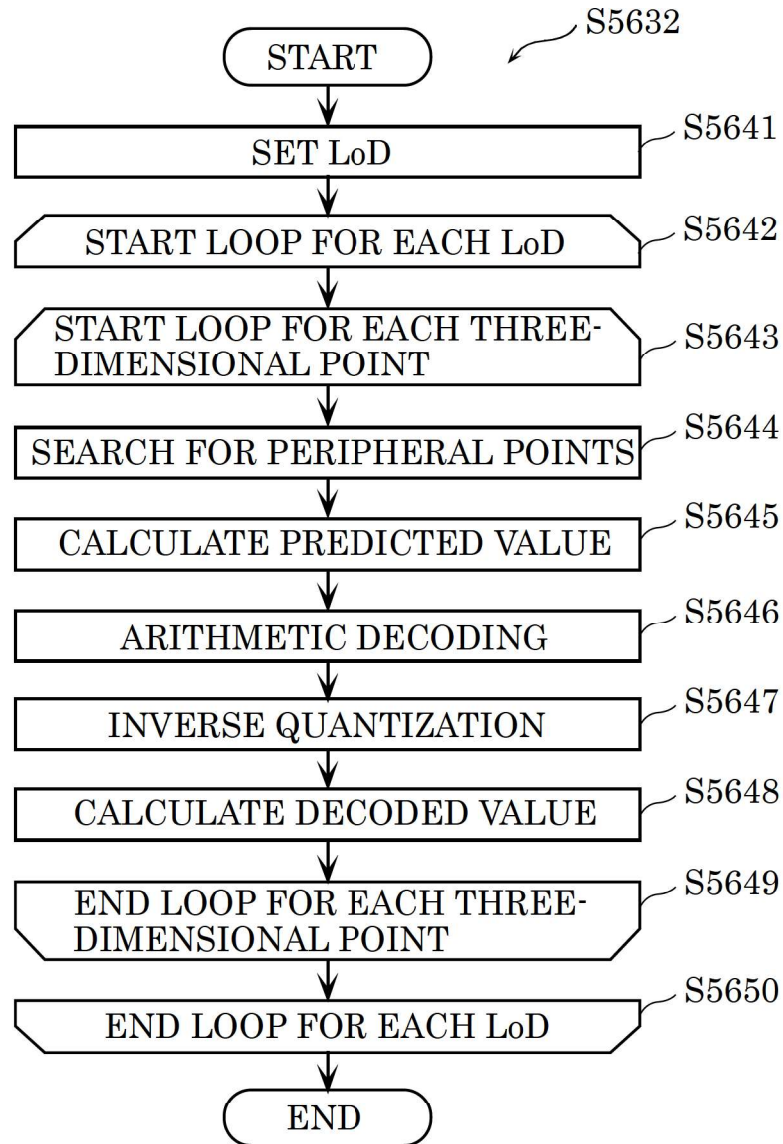


FIG. 90

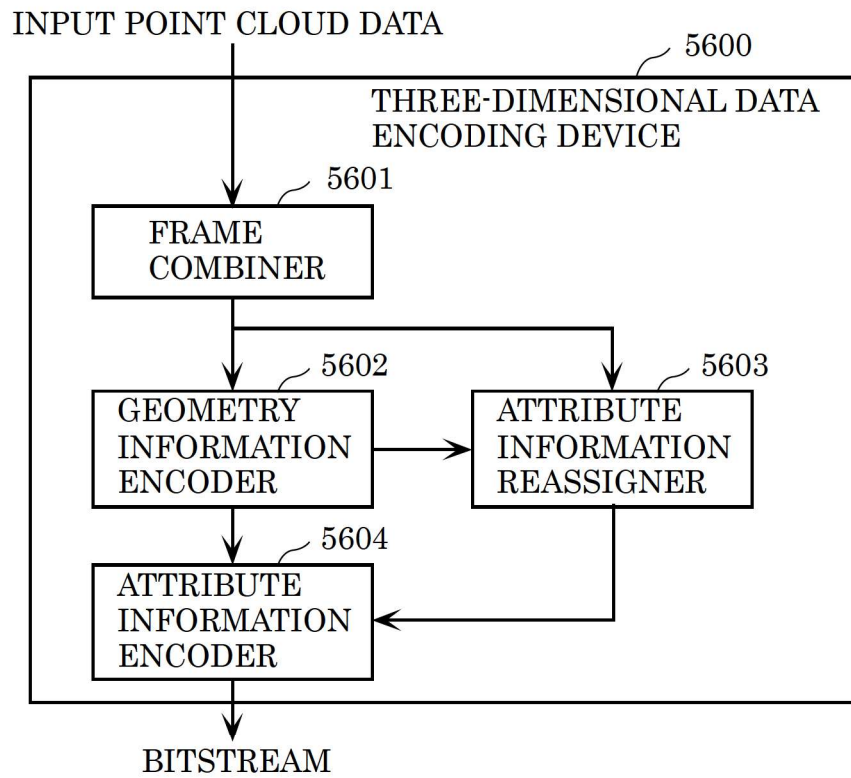


FIG. 91

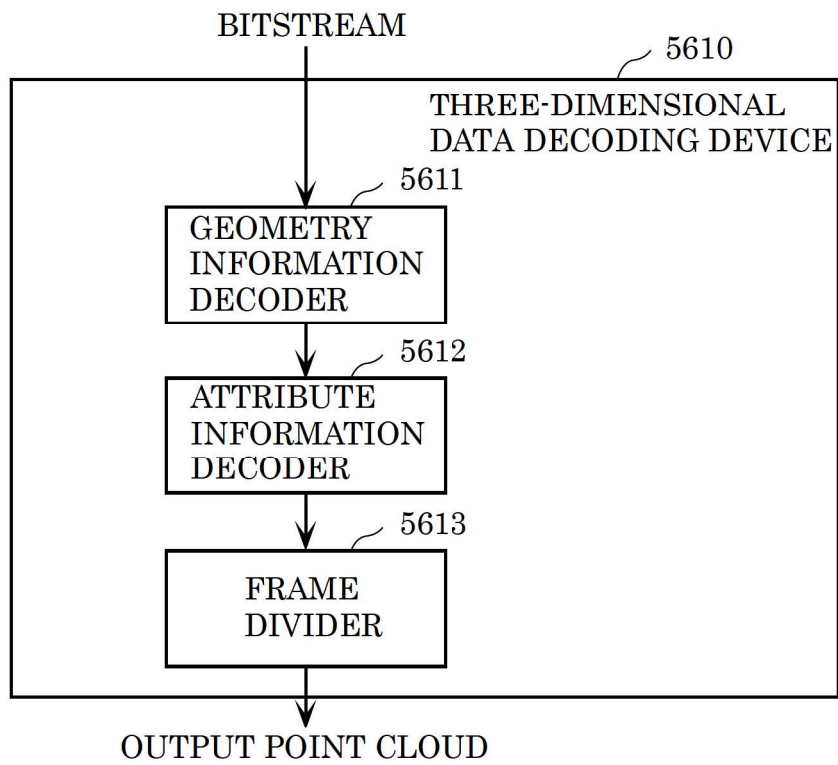
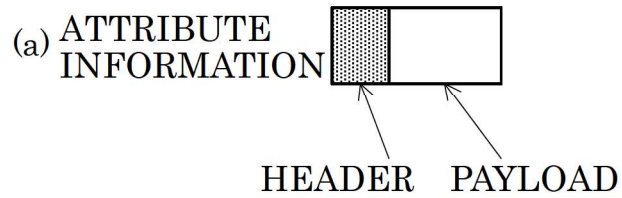


FIG. 92



(b)

```
Attribute_header(){
    aps_idx
    offset
    other_attribute_information
    if(combine_frame_flag){
        number_of _combine_frame
        refer_different_frame
    }
    if(tile){
        tile_type
        tile_idx
    }
    if(slice){
        slice_type
        slice_idx
    }
}
```

(c)

```
Attribute_data(){
    combine_information();
}
```

FIG. 93

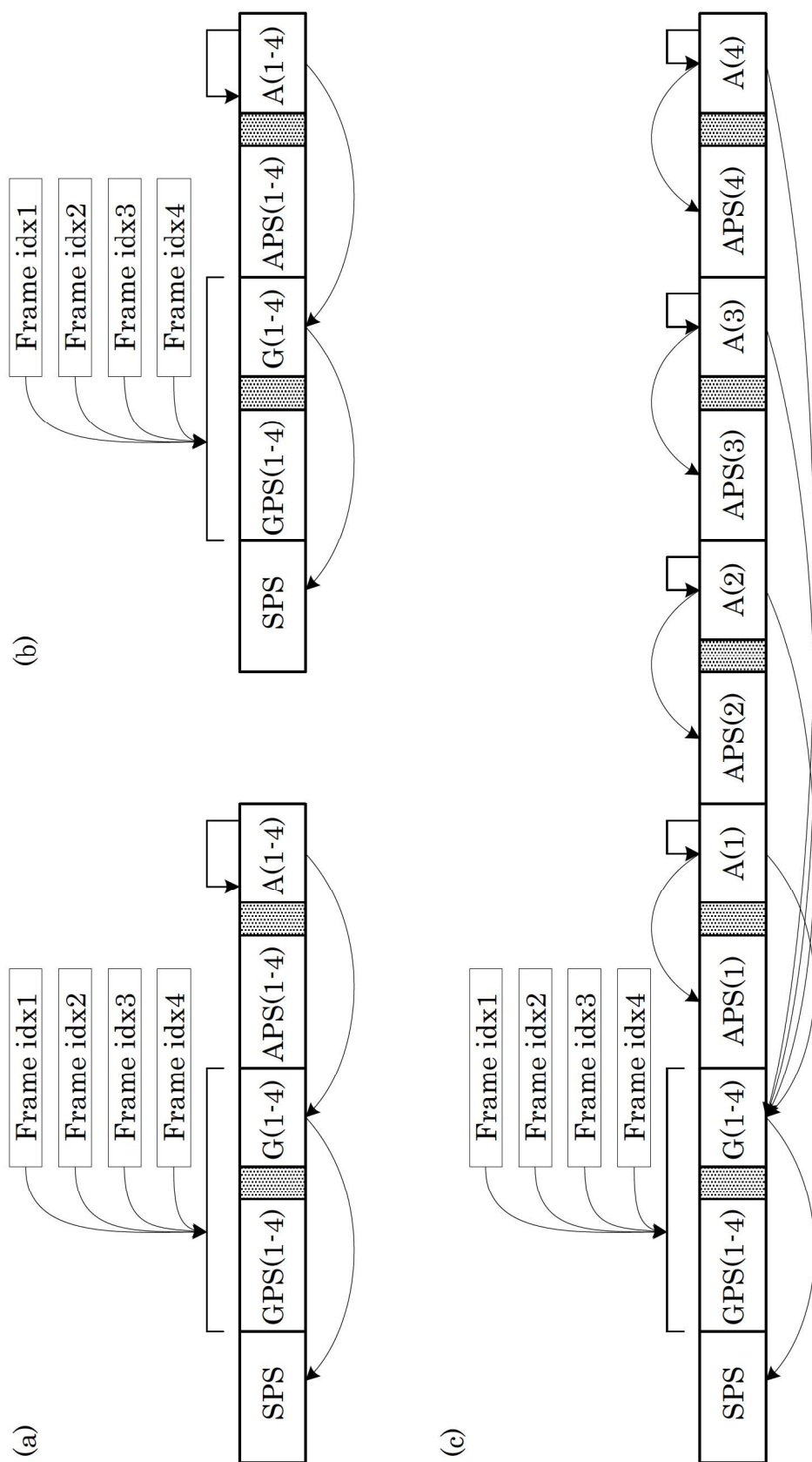


FIG. 94

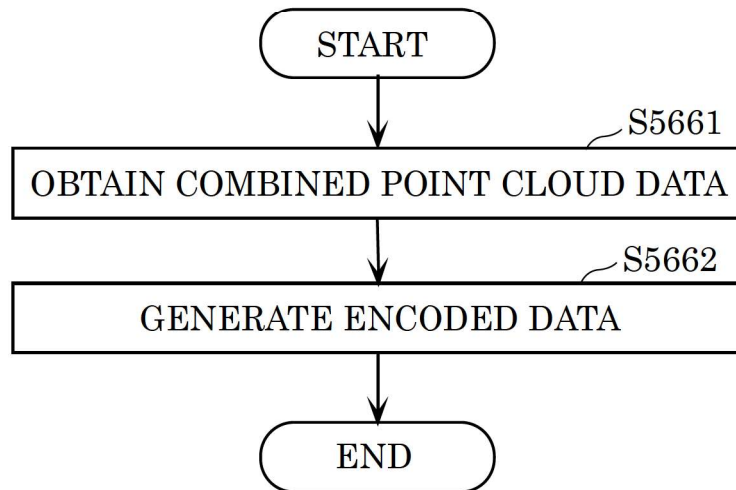


FIG. 95

