

FIG. 1

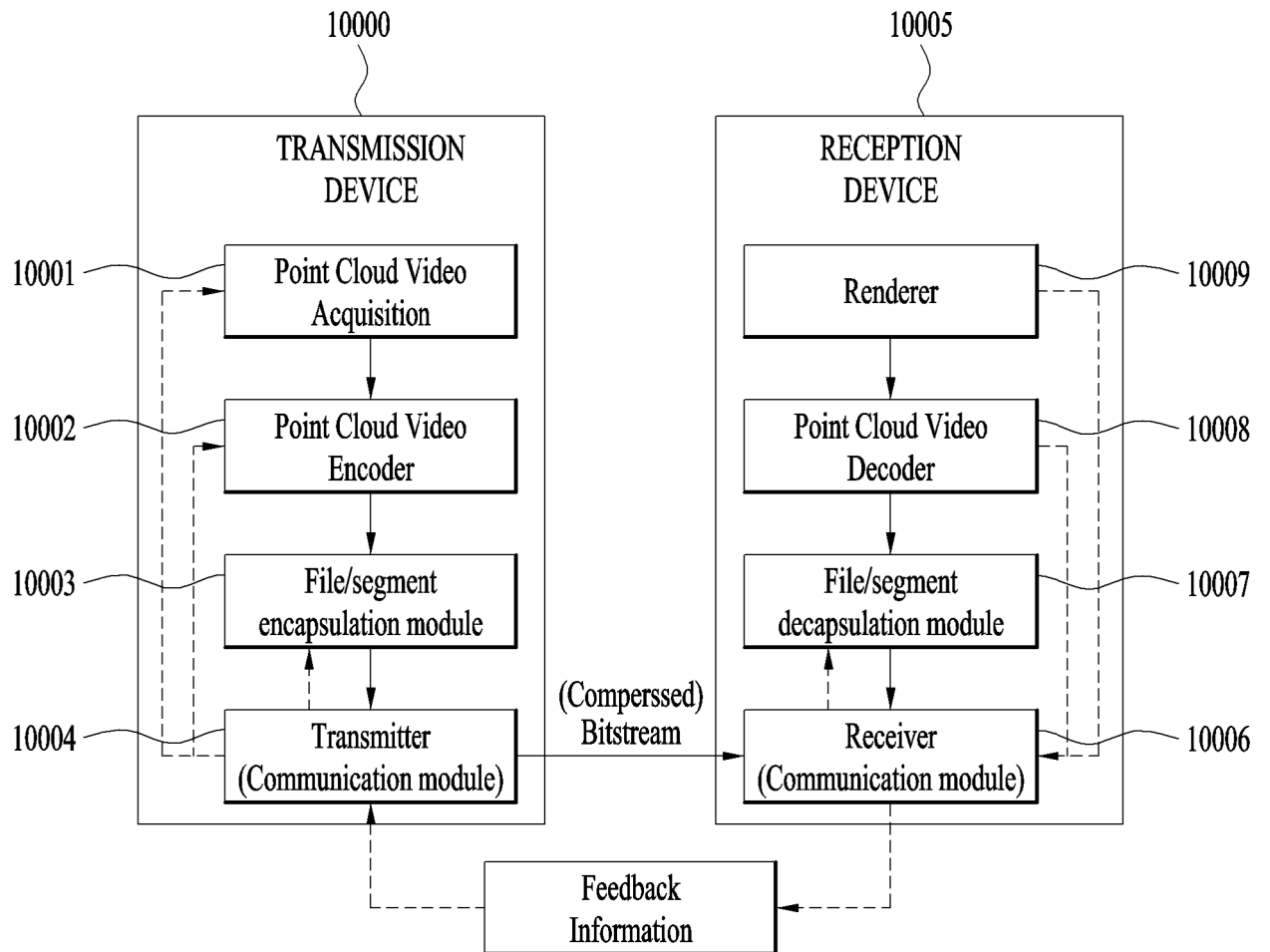


FIG. 2

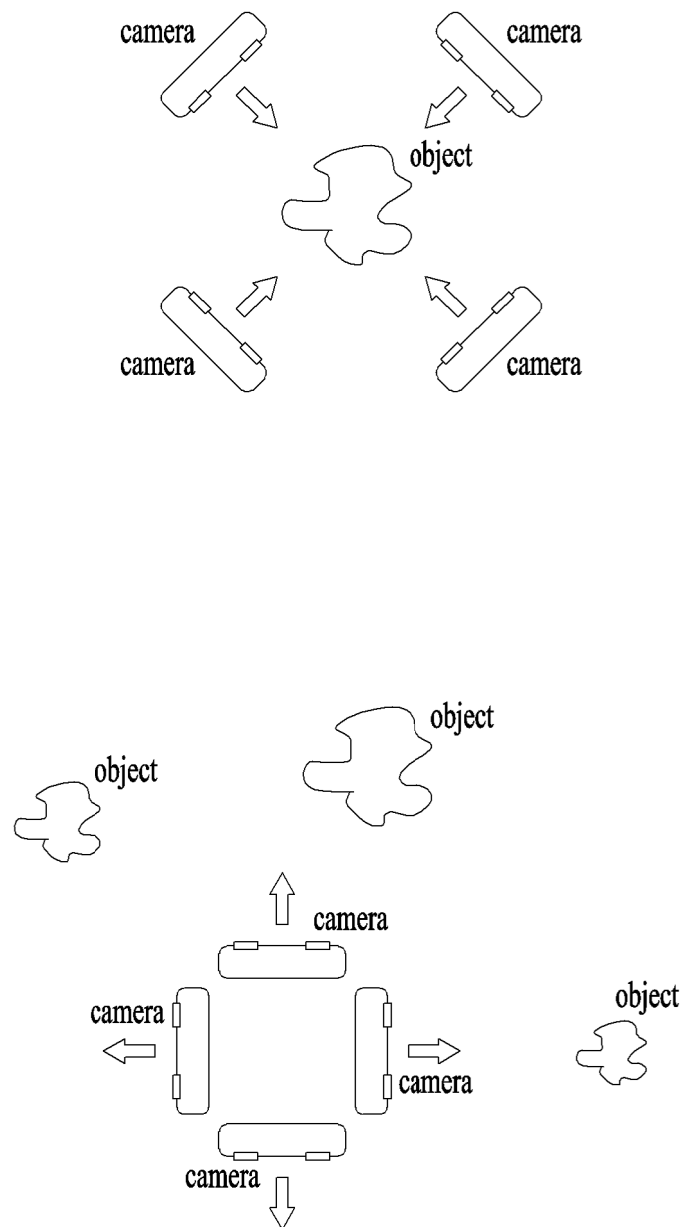


FIG. 3

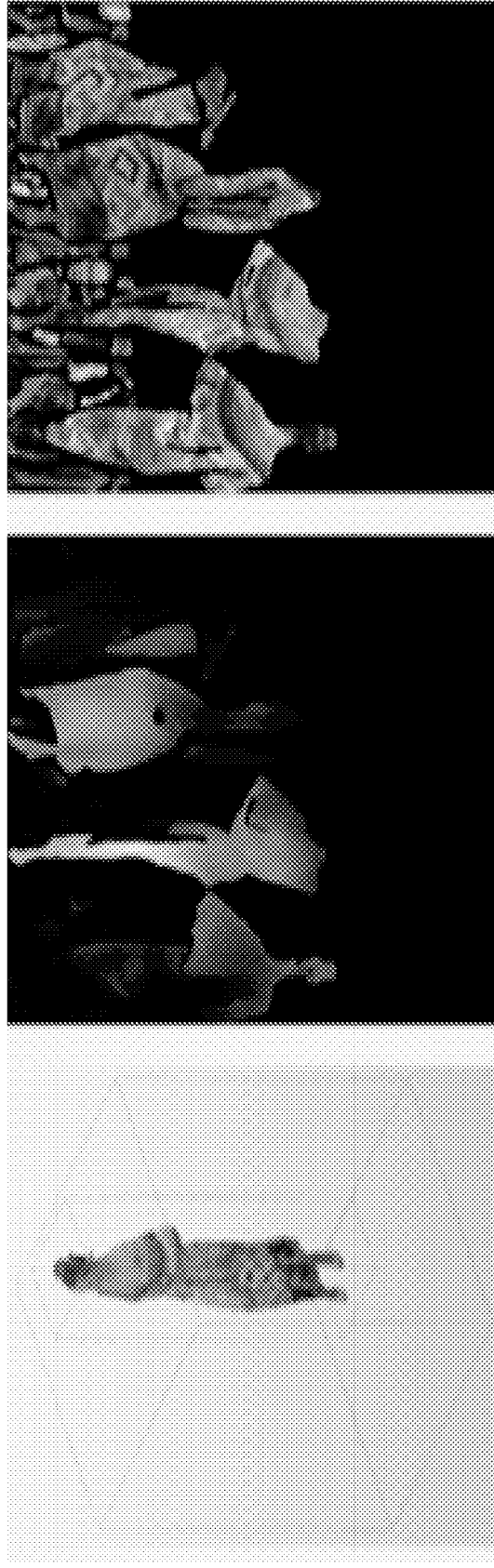


FIG. 4

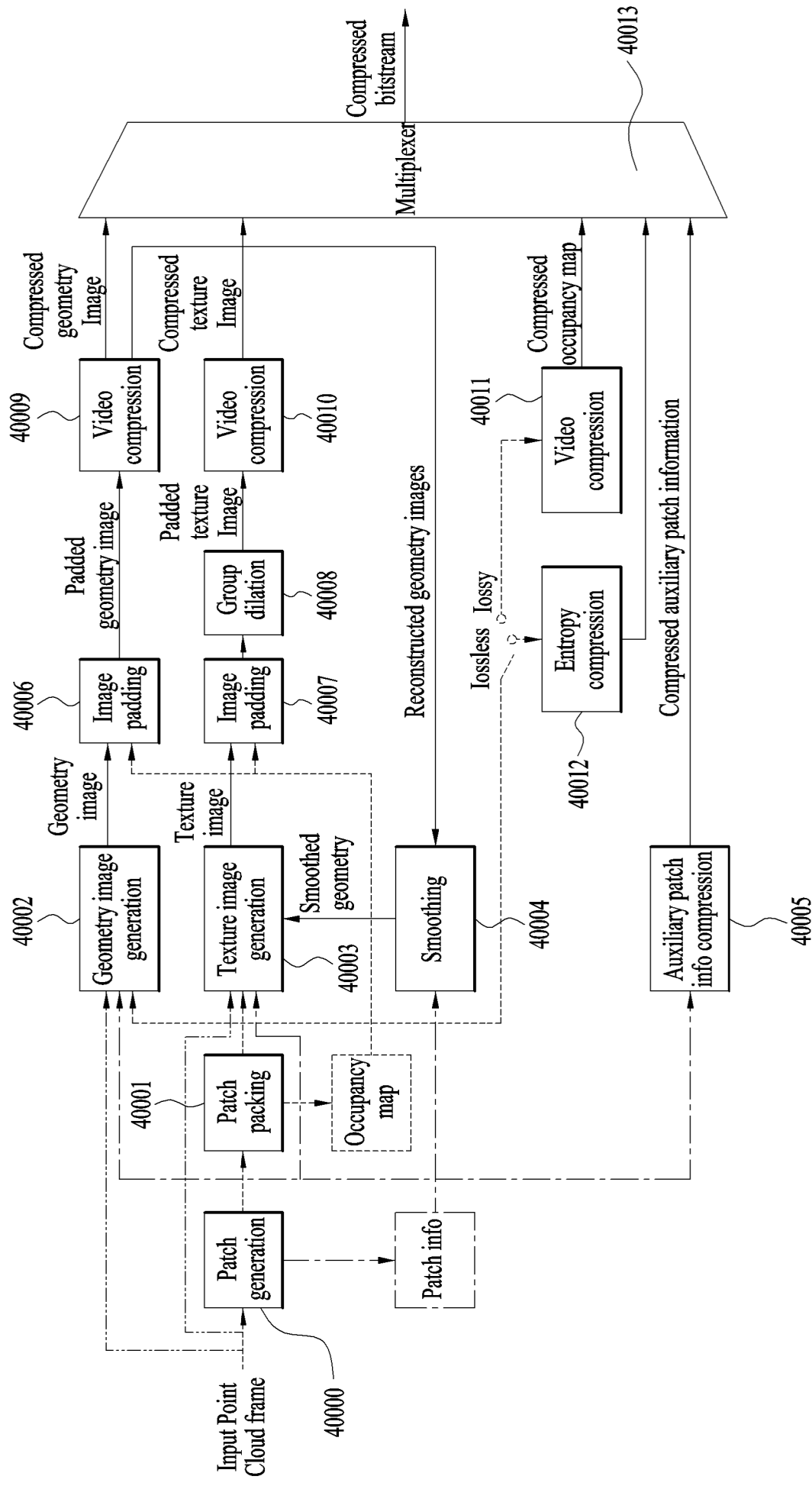


FIG. 5

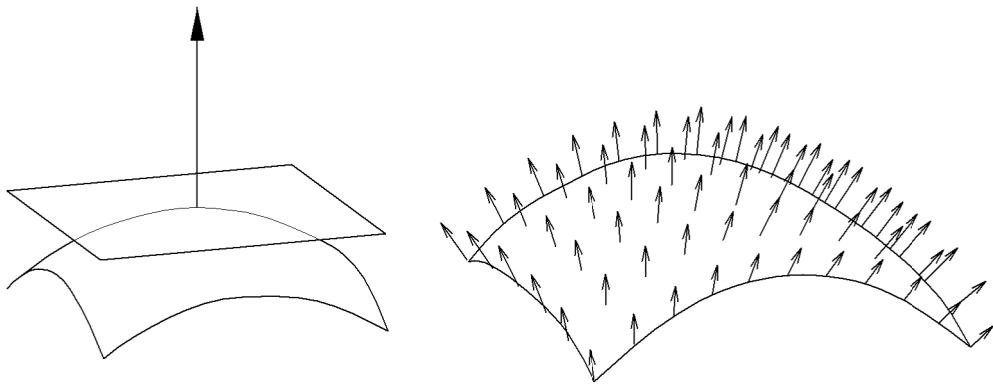


FIG. 6

FIG. 7

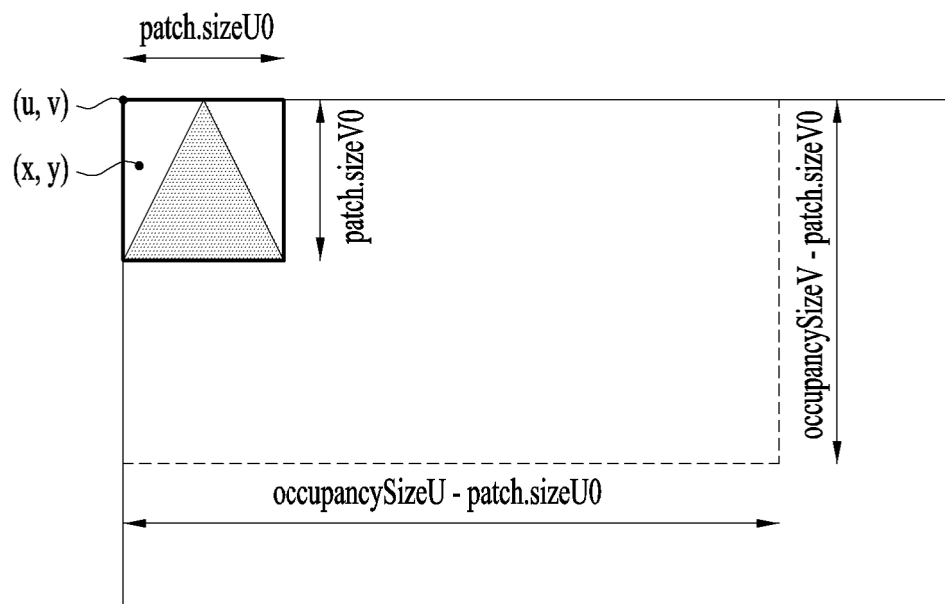


FIG. 8

FIG. 9

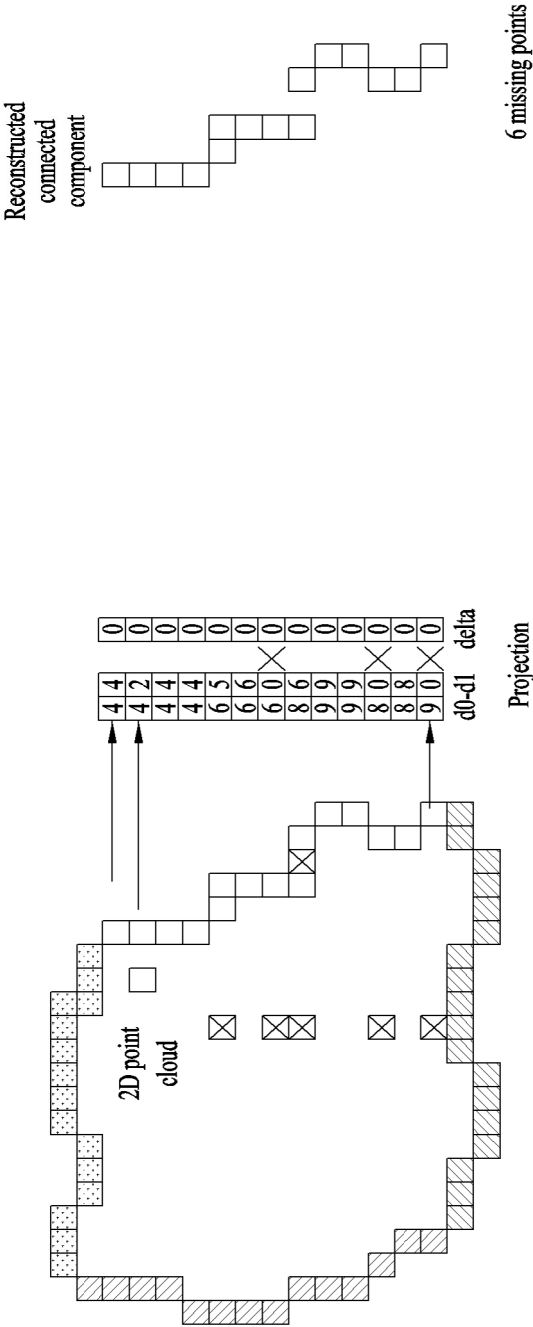
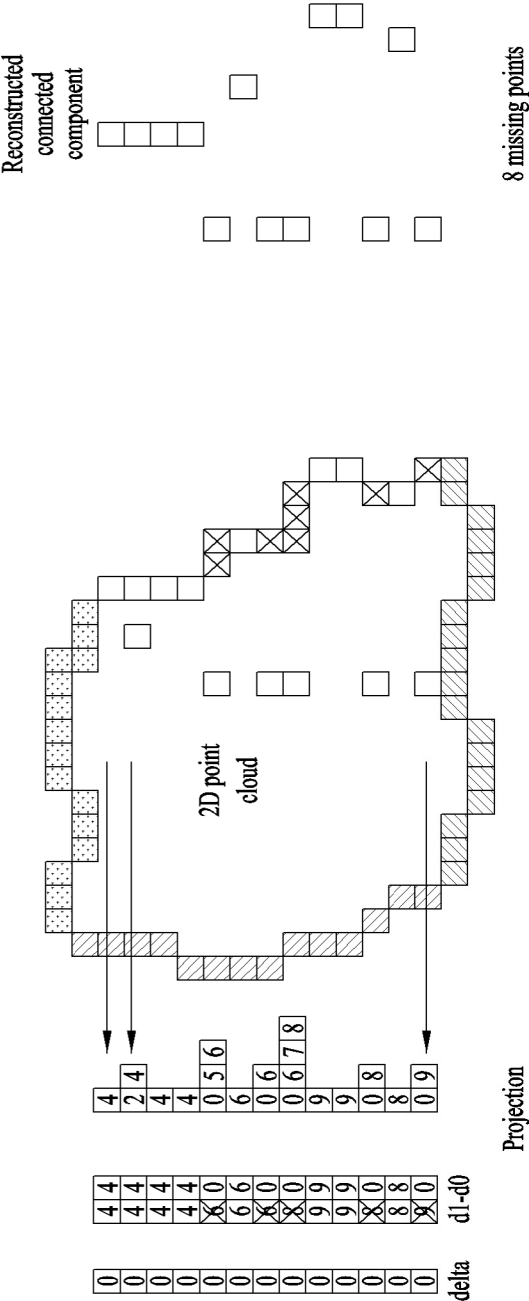


FIG. 10

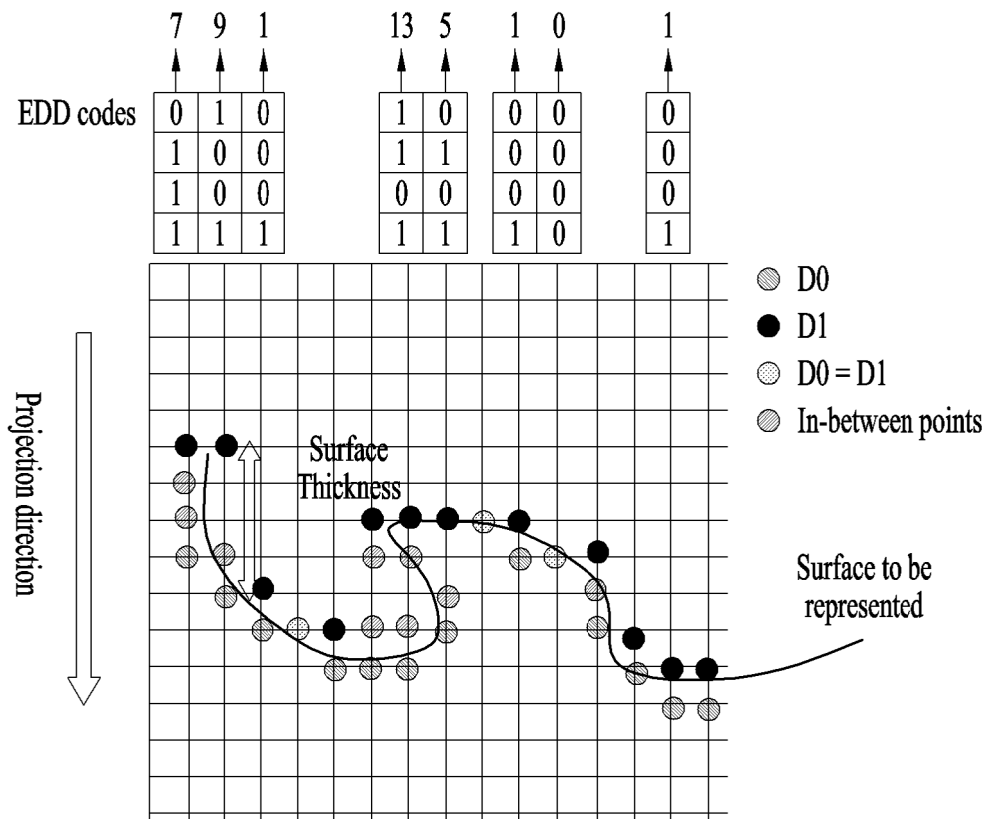


FIG. 11

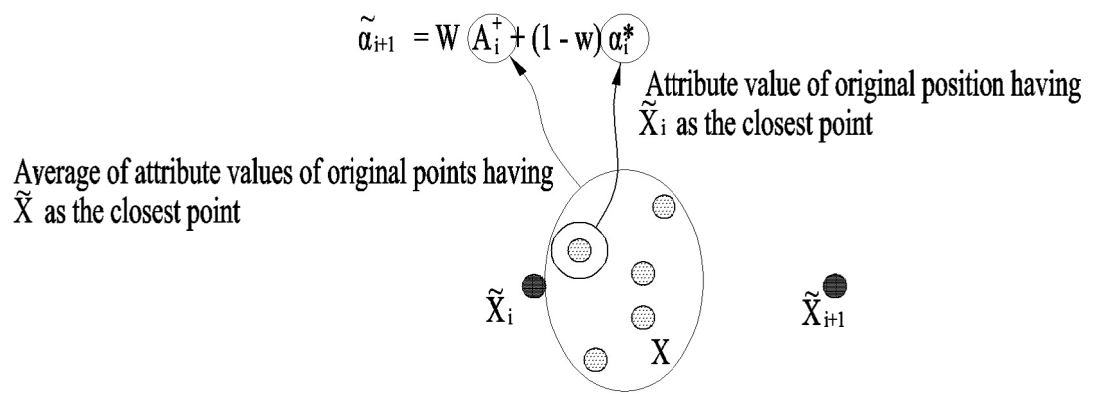


FIG. 12

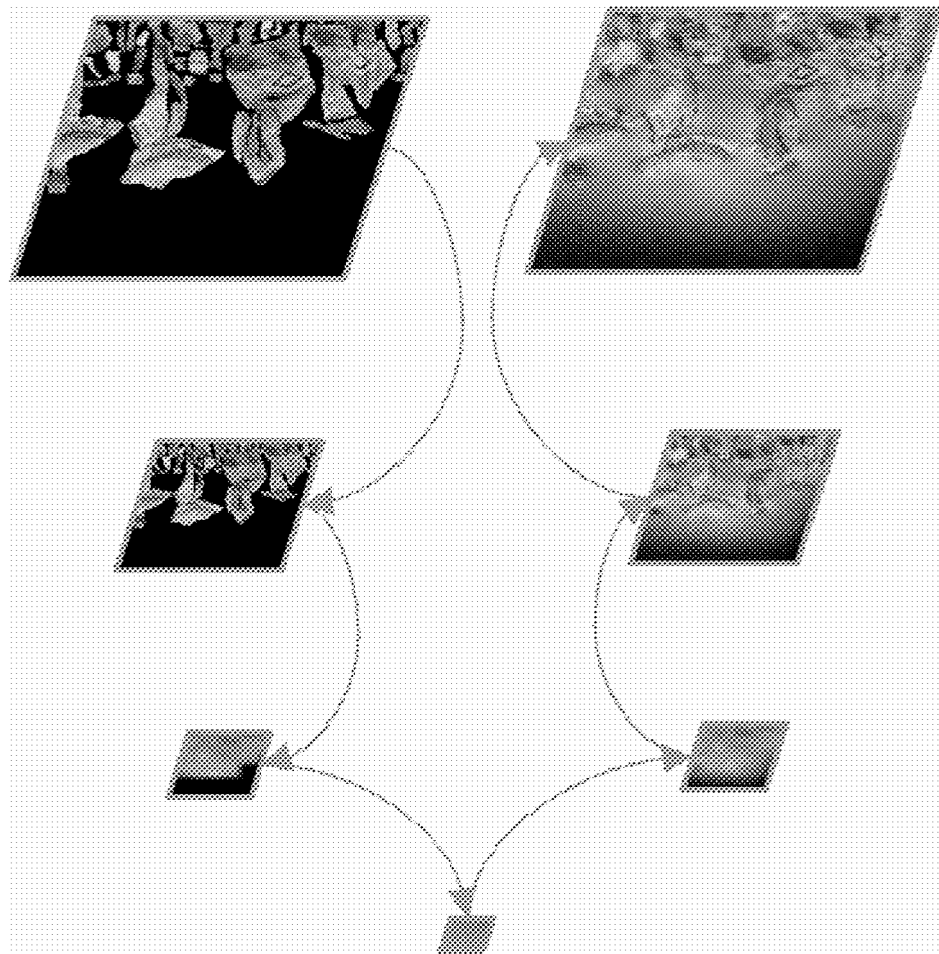


FIG. 13

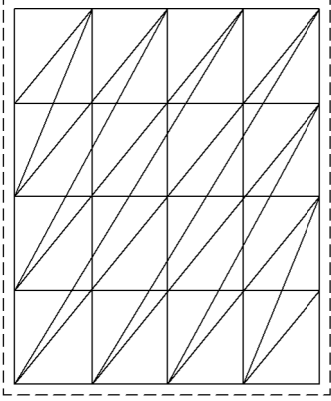
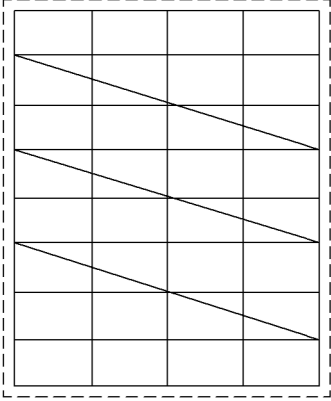
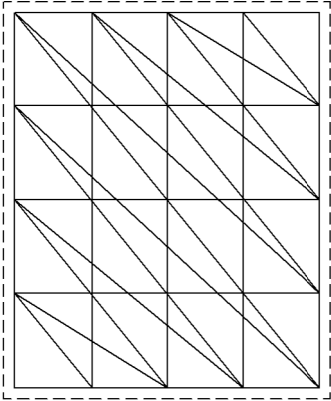
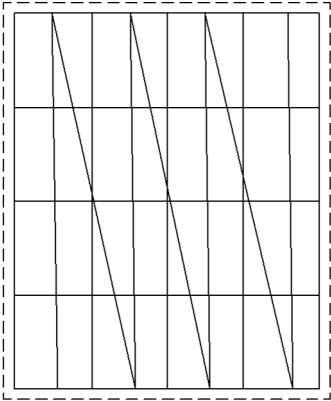
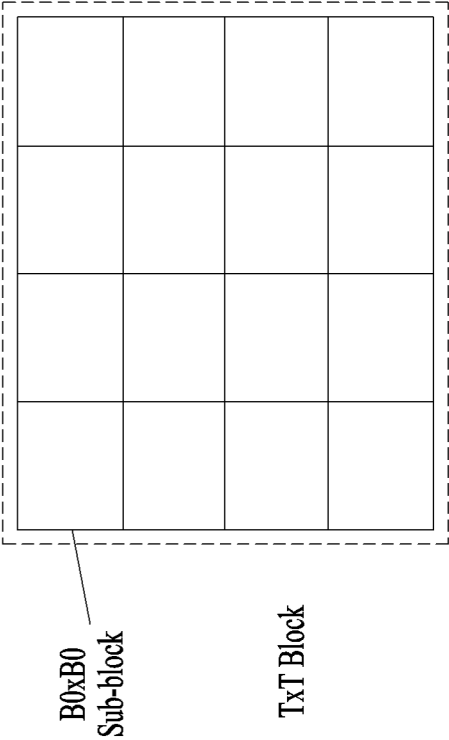


FIG. 14

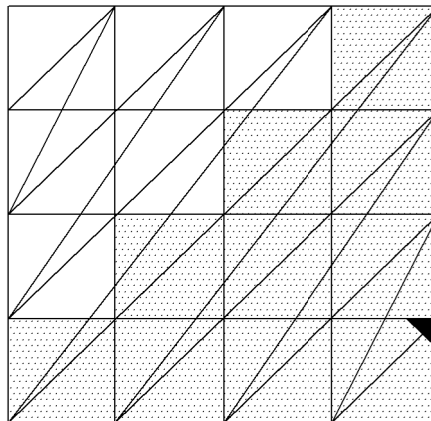


FIG. 15

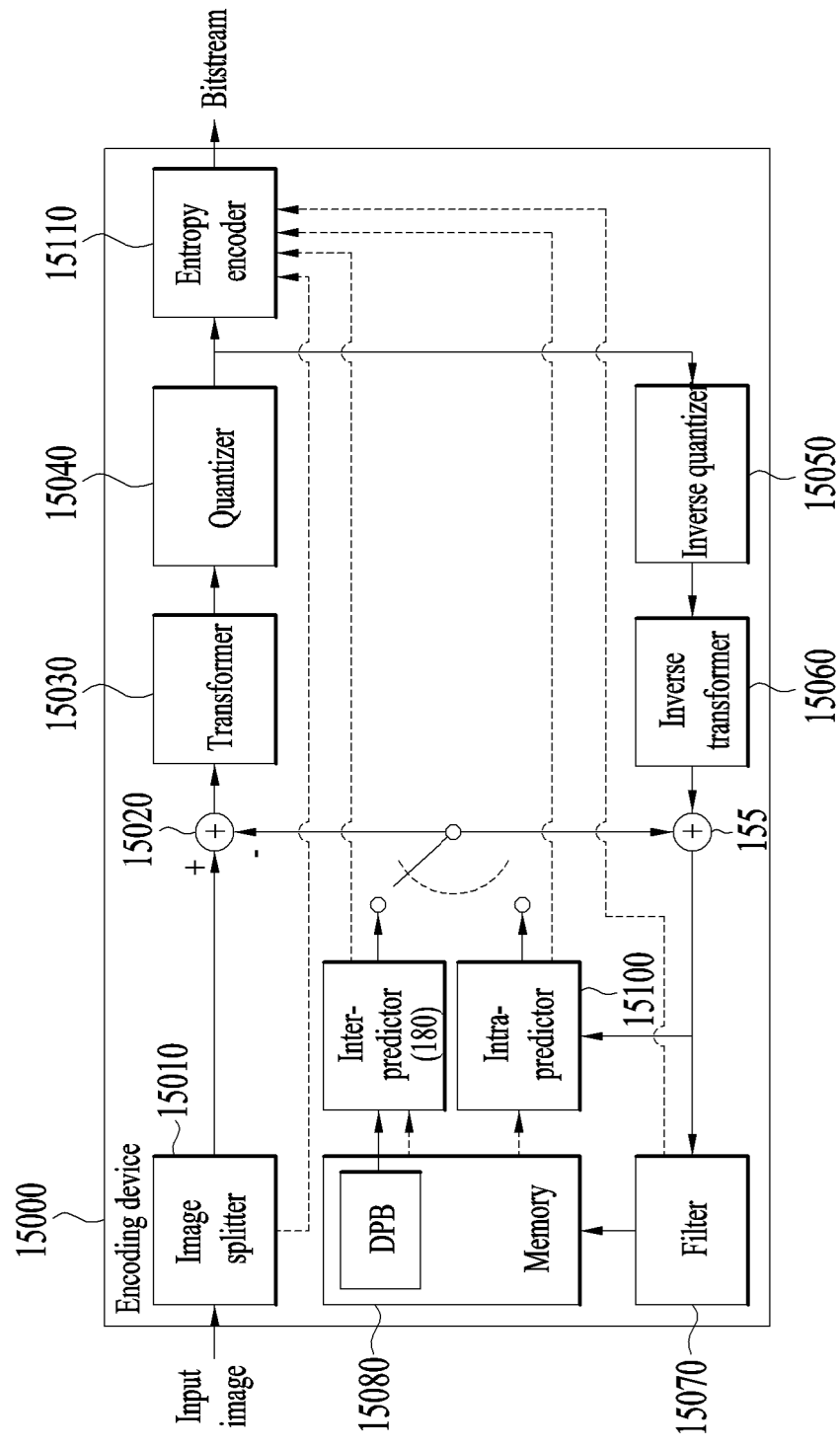


FIG. 16

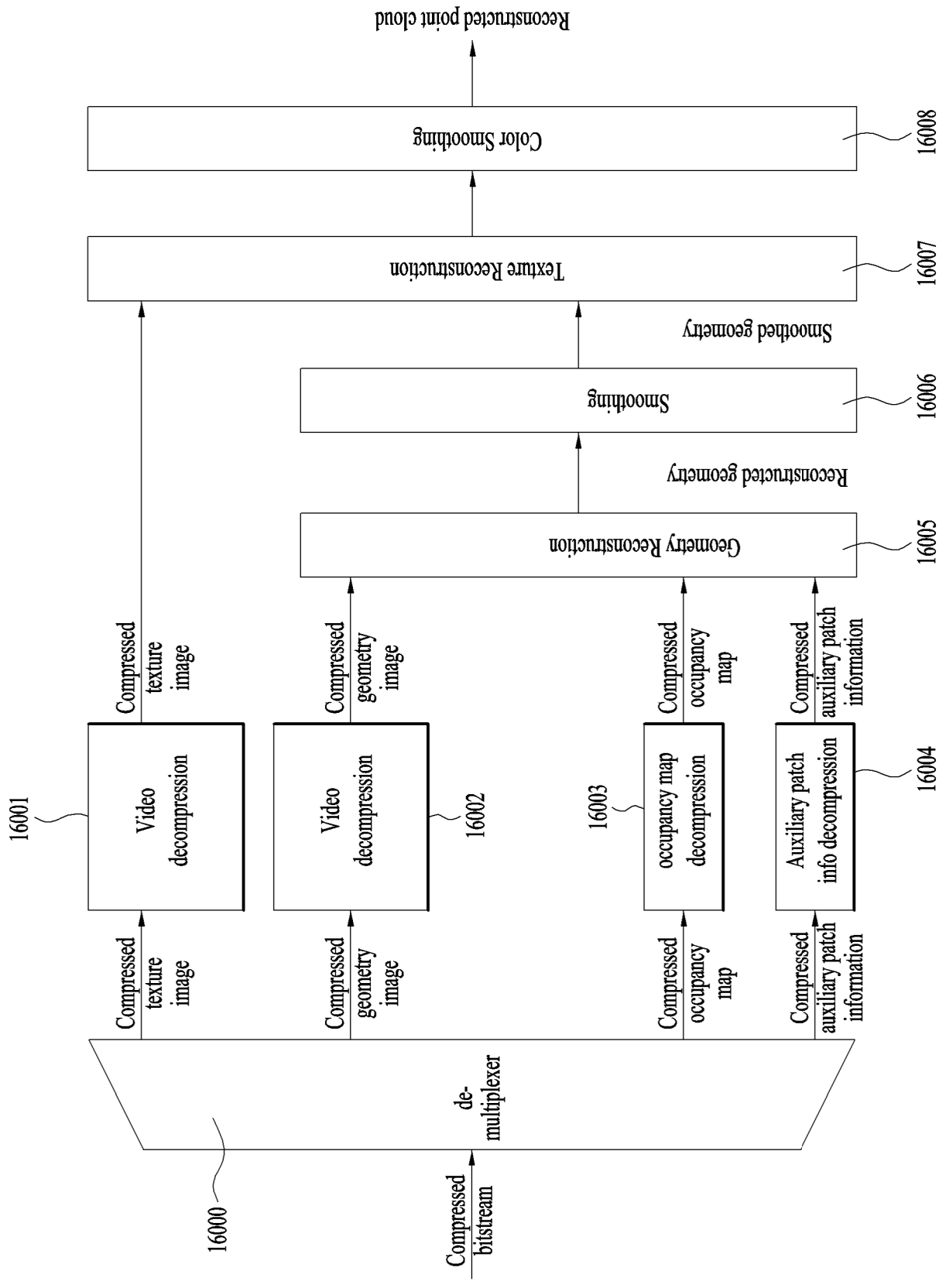


FIG. 17

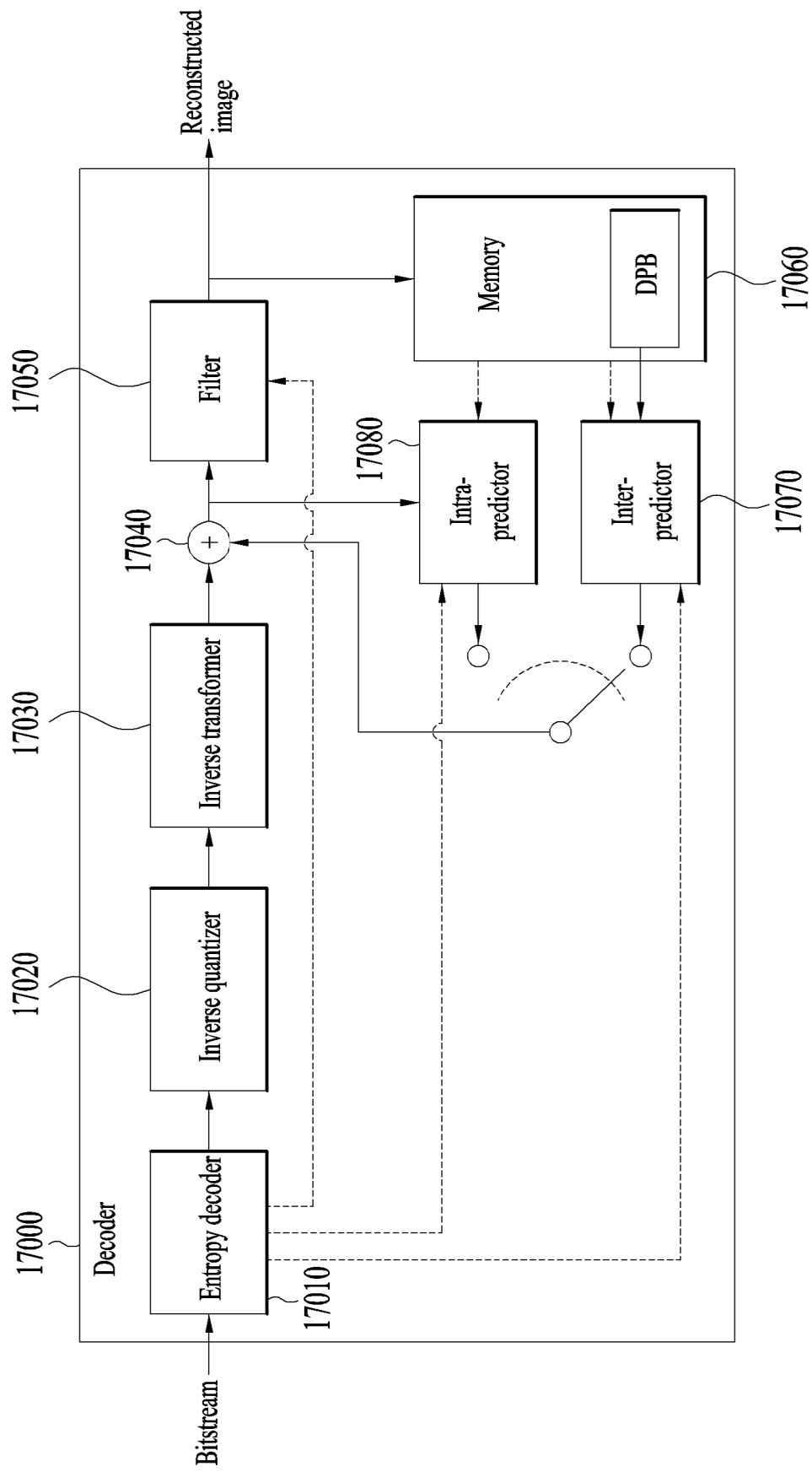


FIG. 18

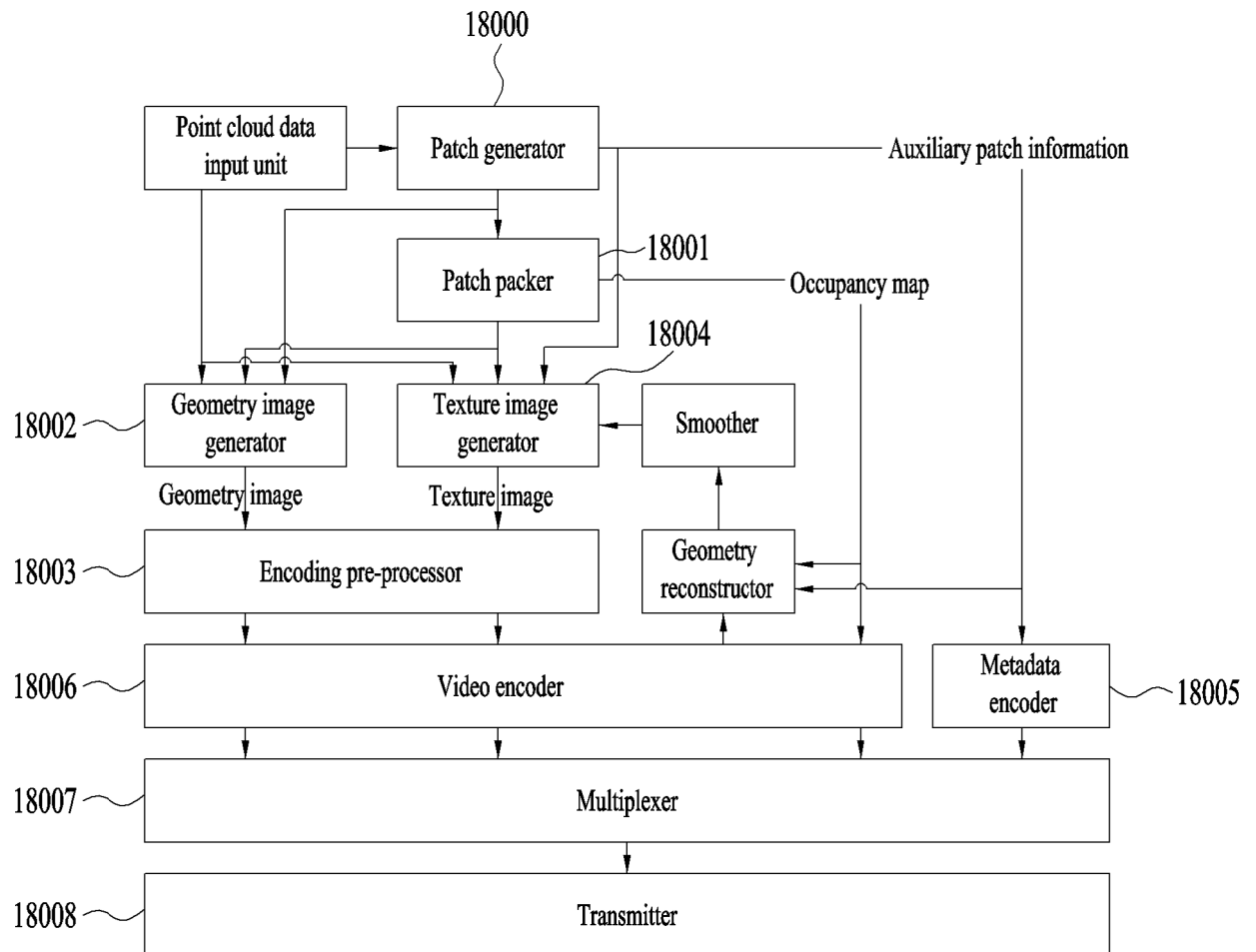


FIG. 19

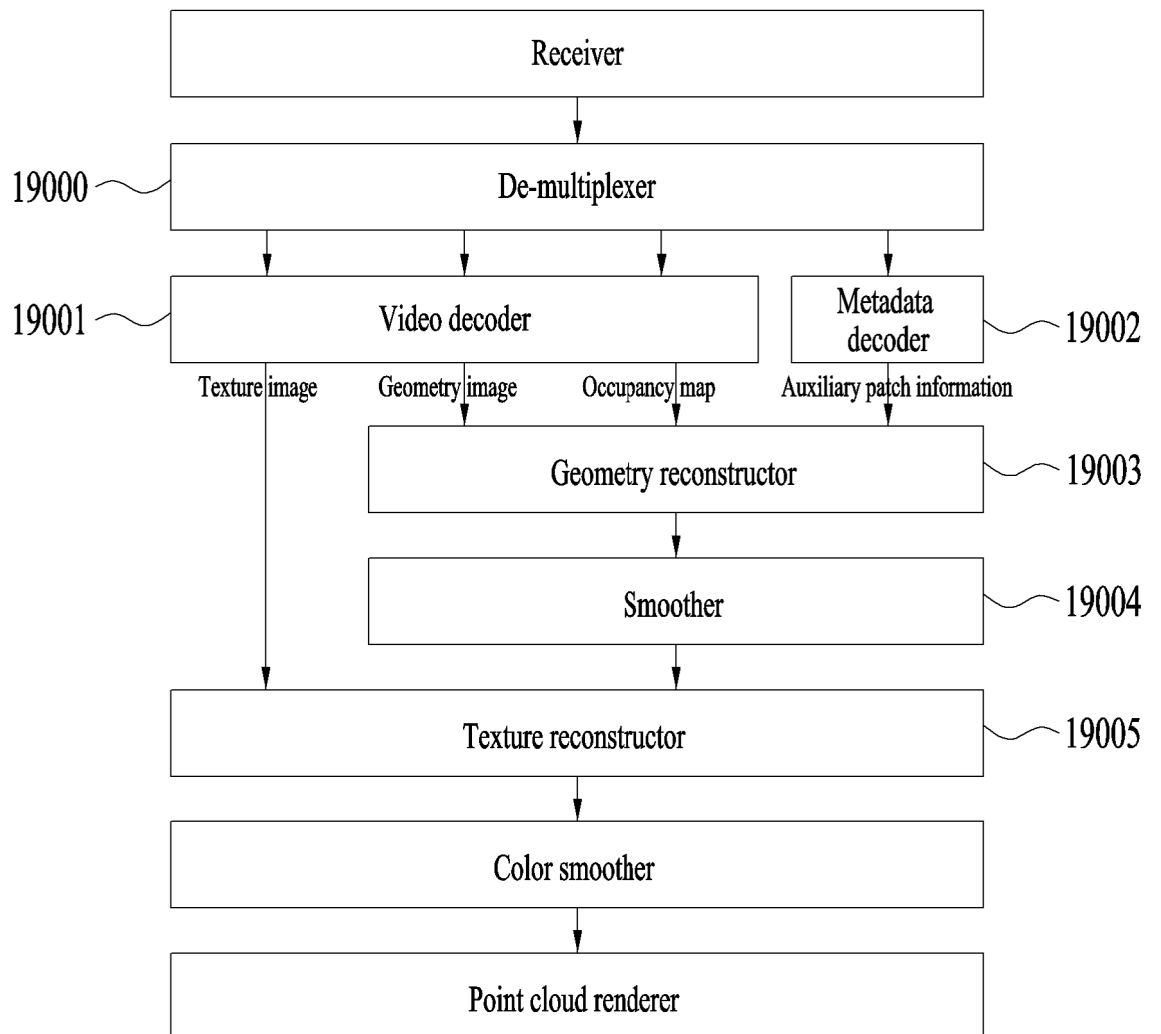


FIG. 20

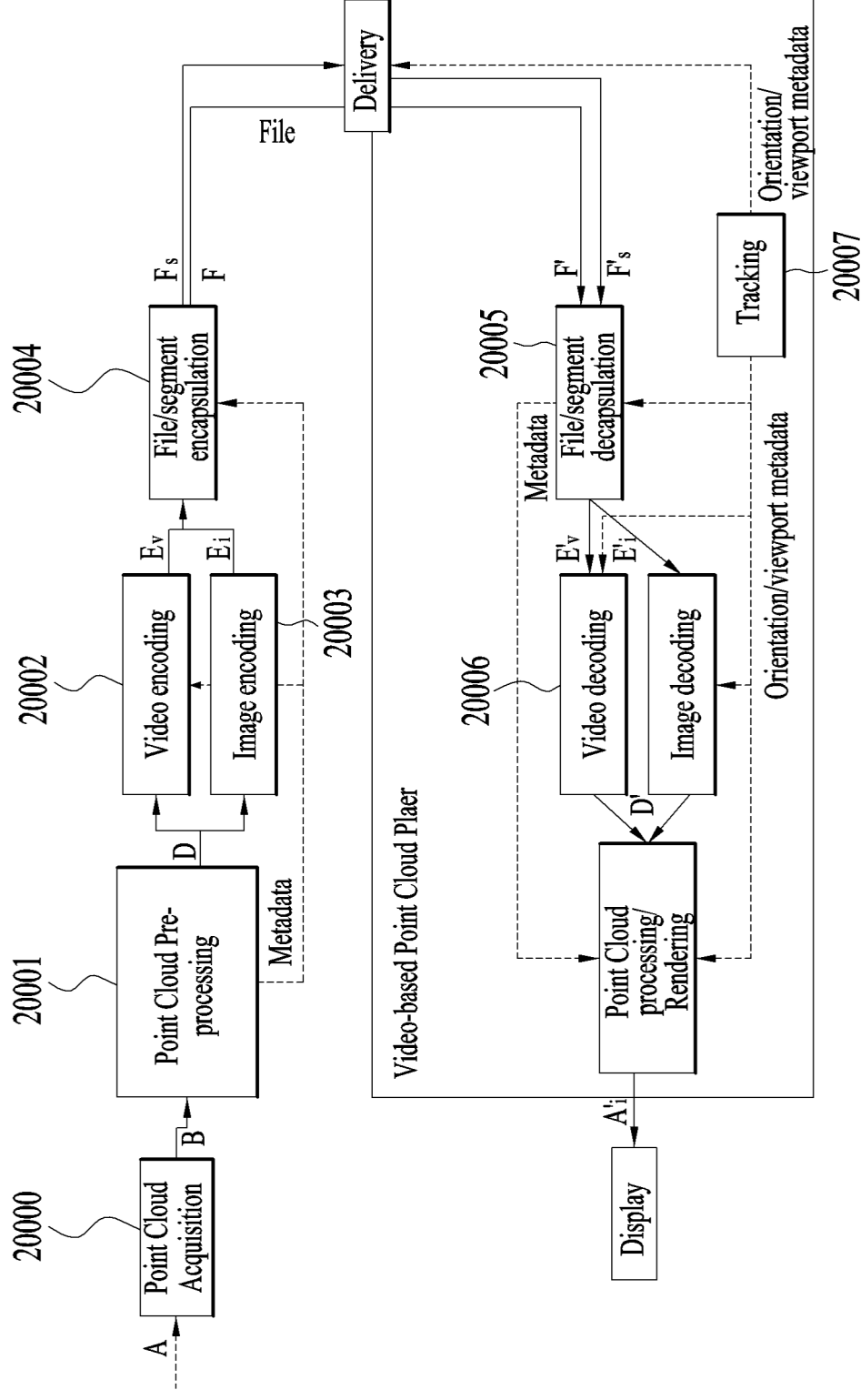


FIG. 21

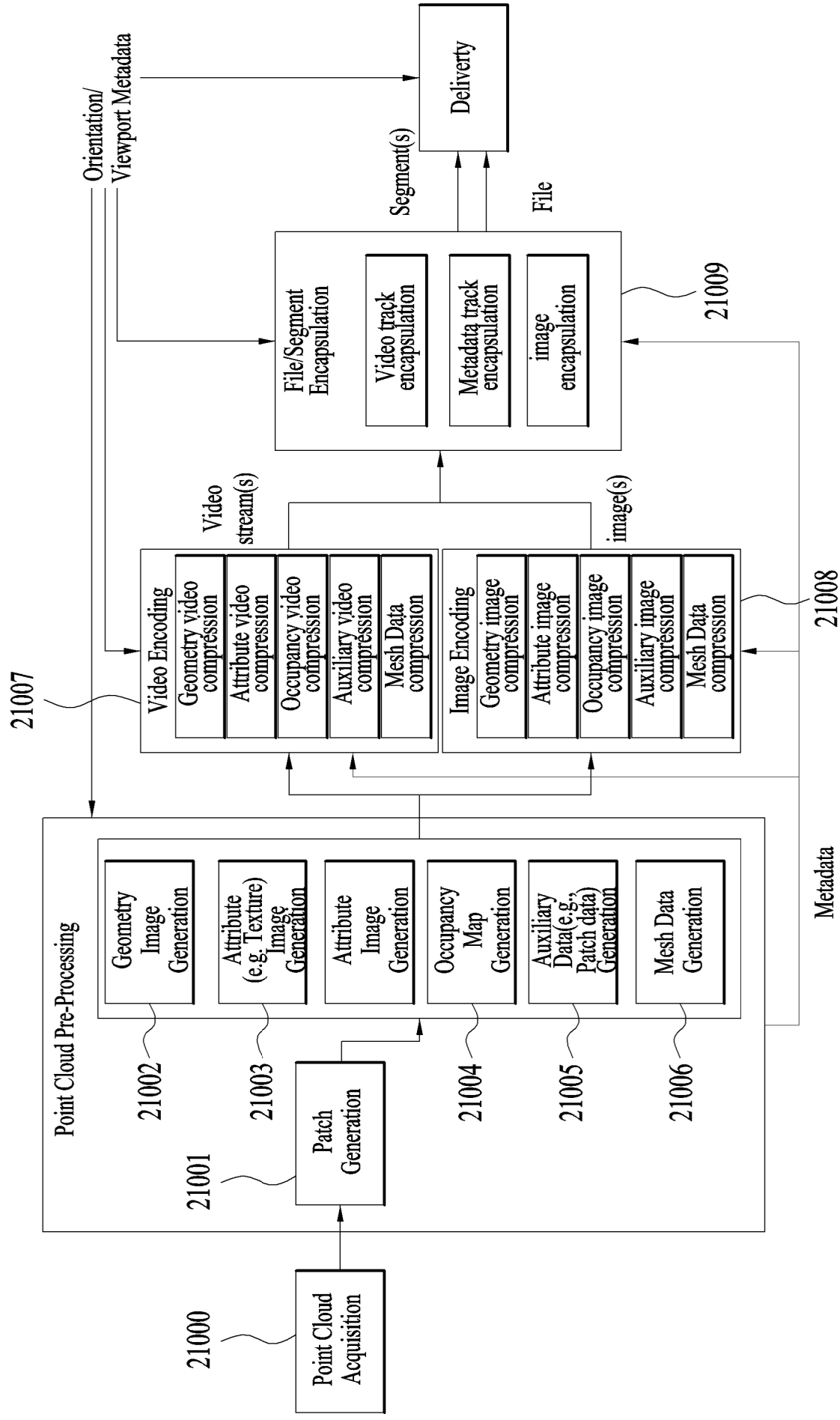


FIG. 22

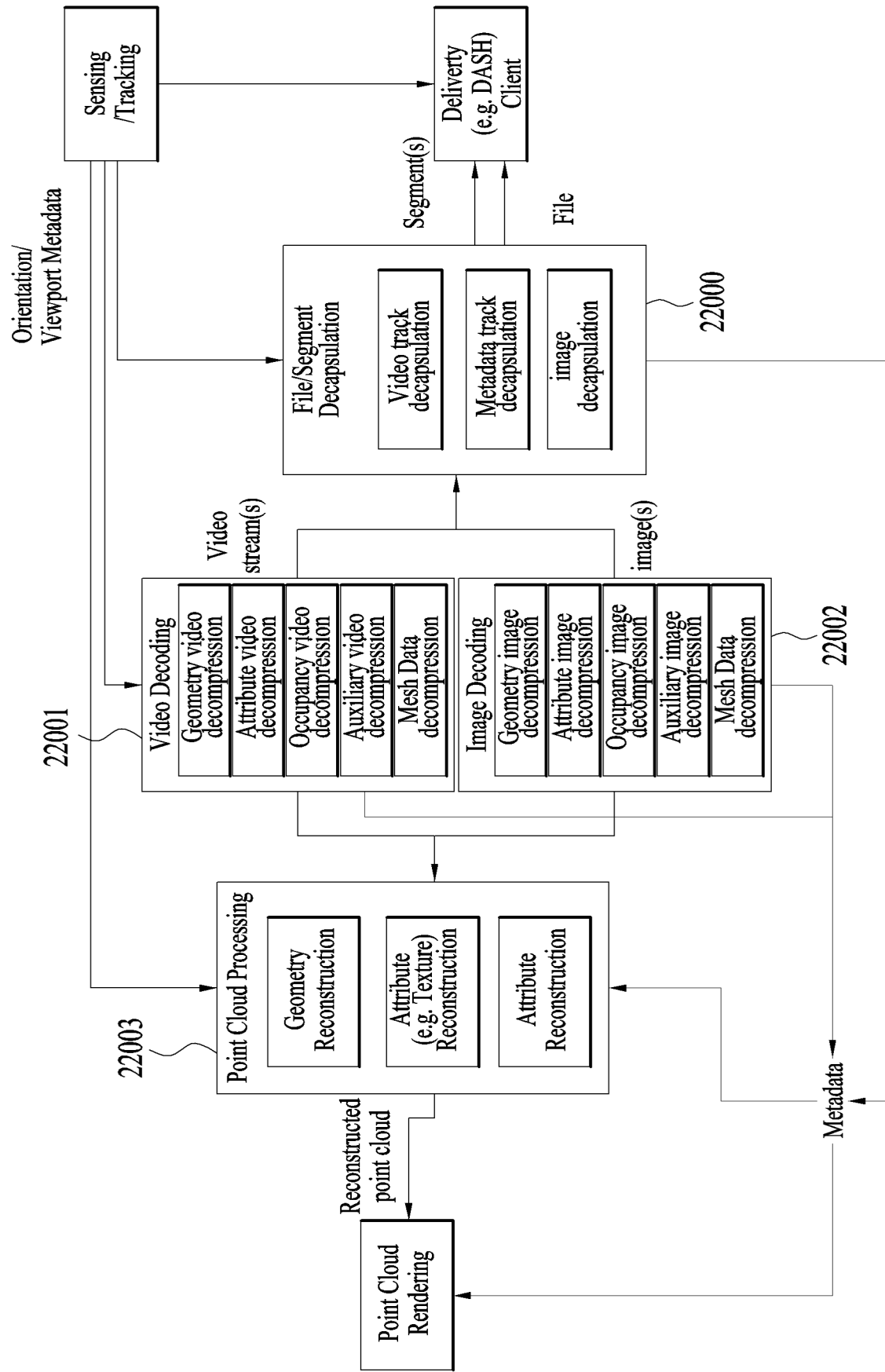


FIG. 23

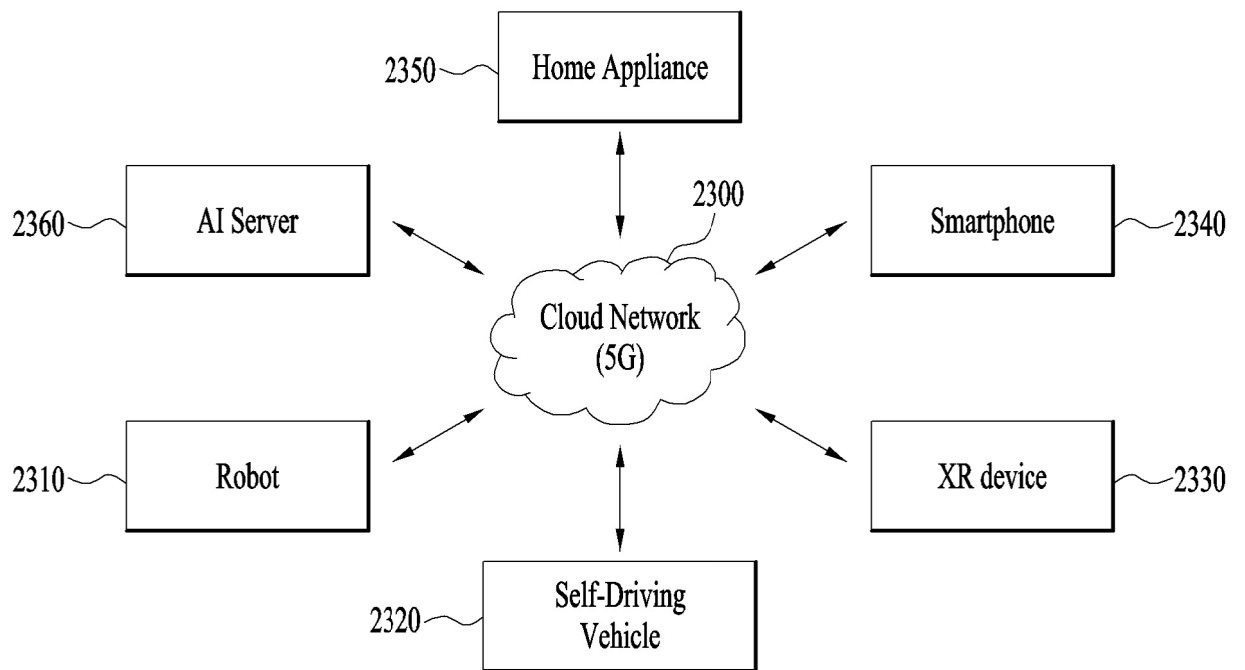


FIG. 24

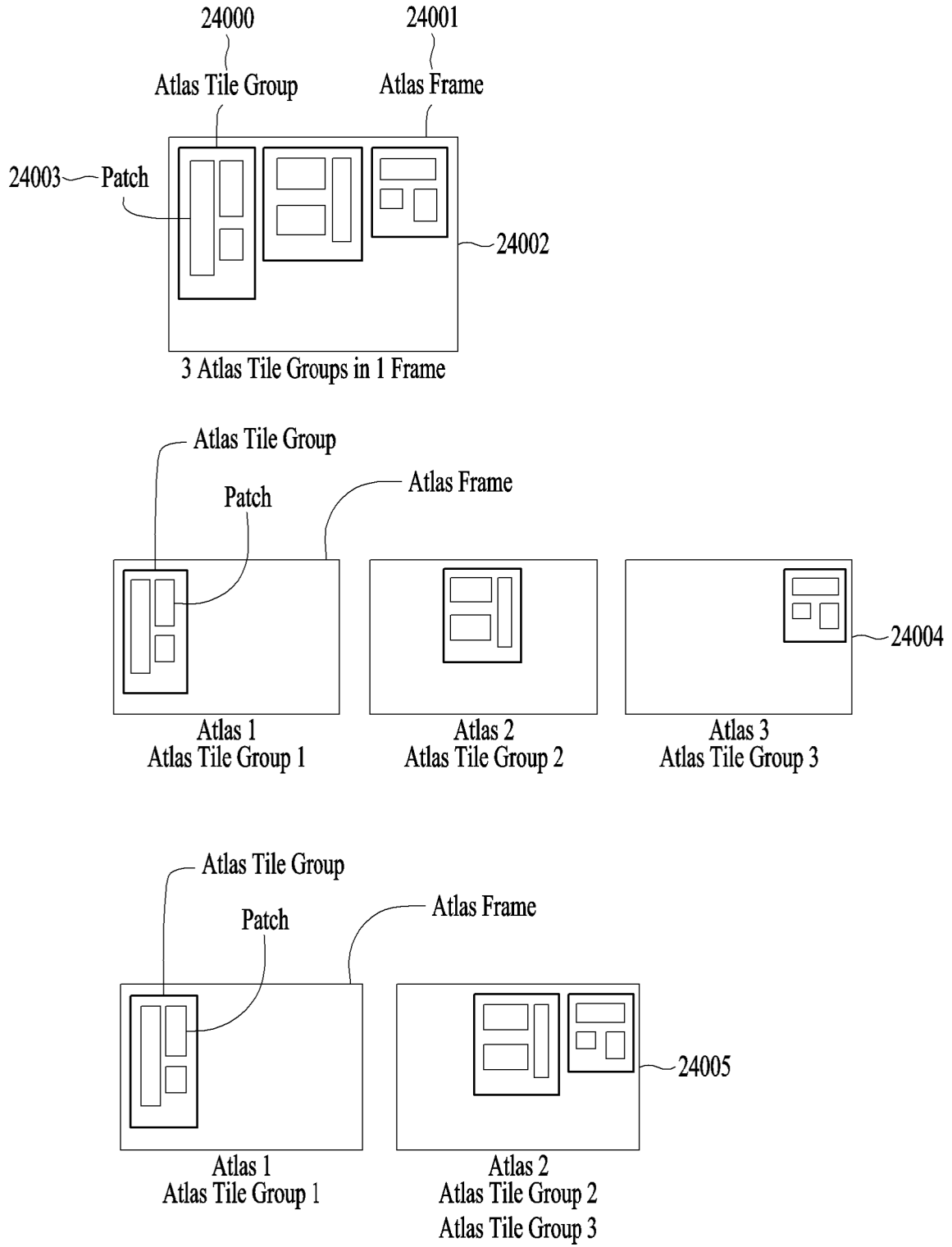


FIG. 25

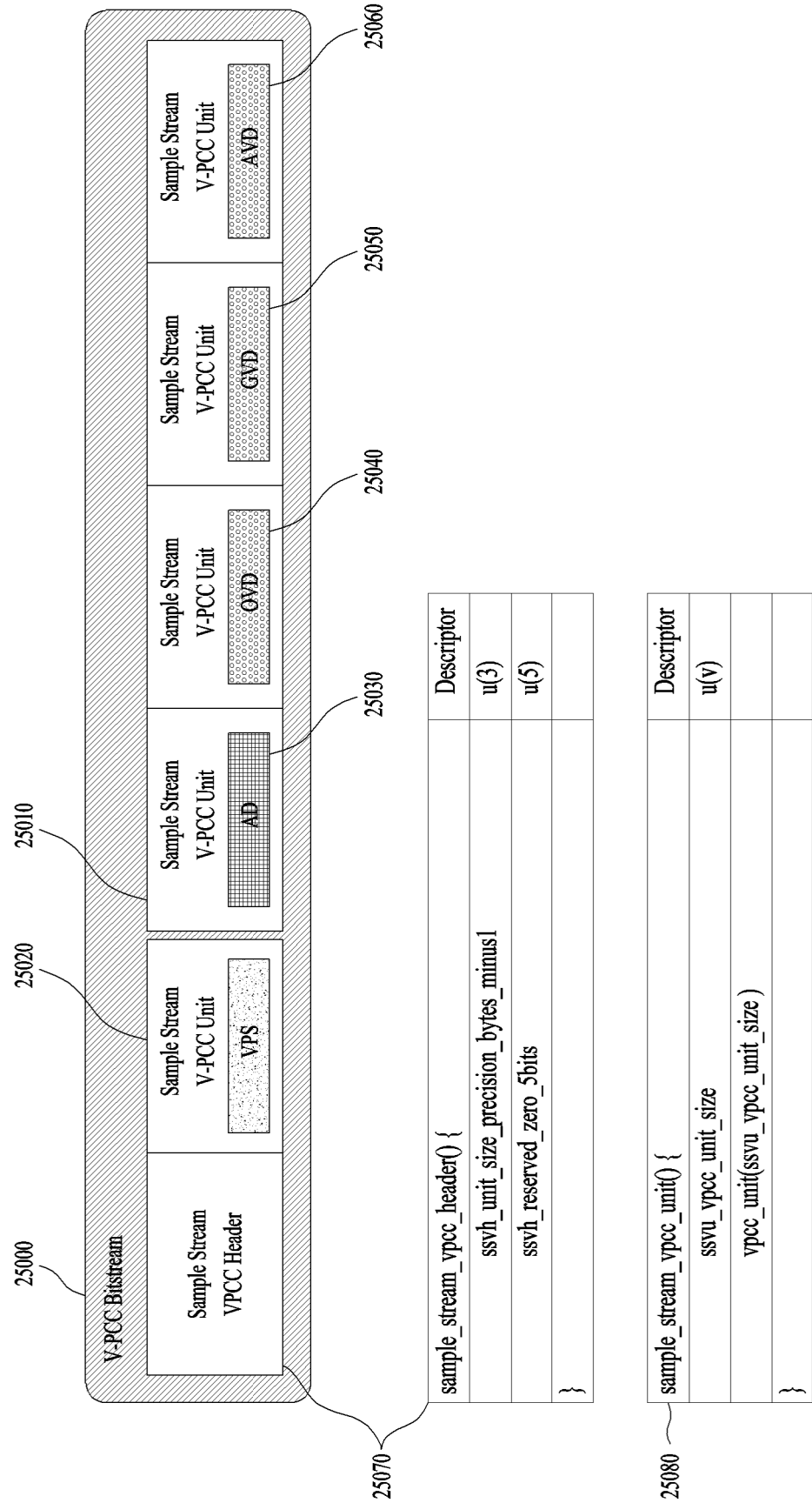


FIG. 26

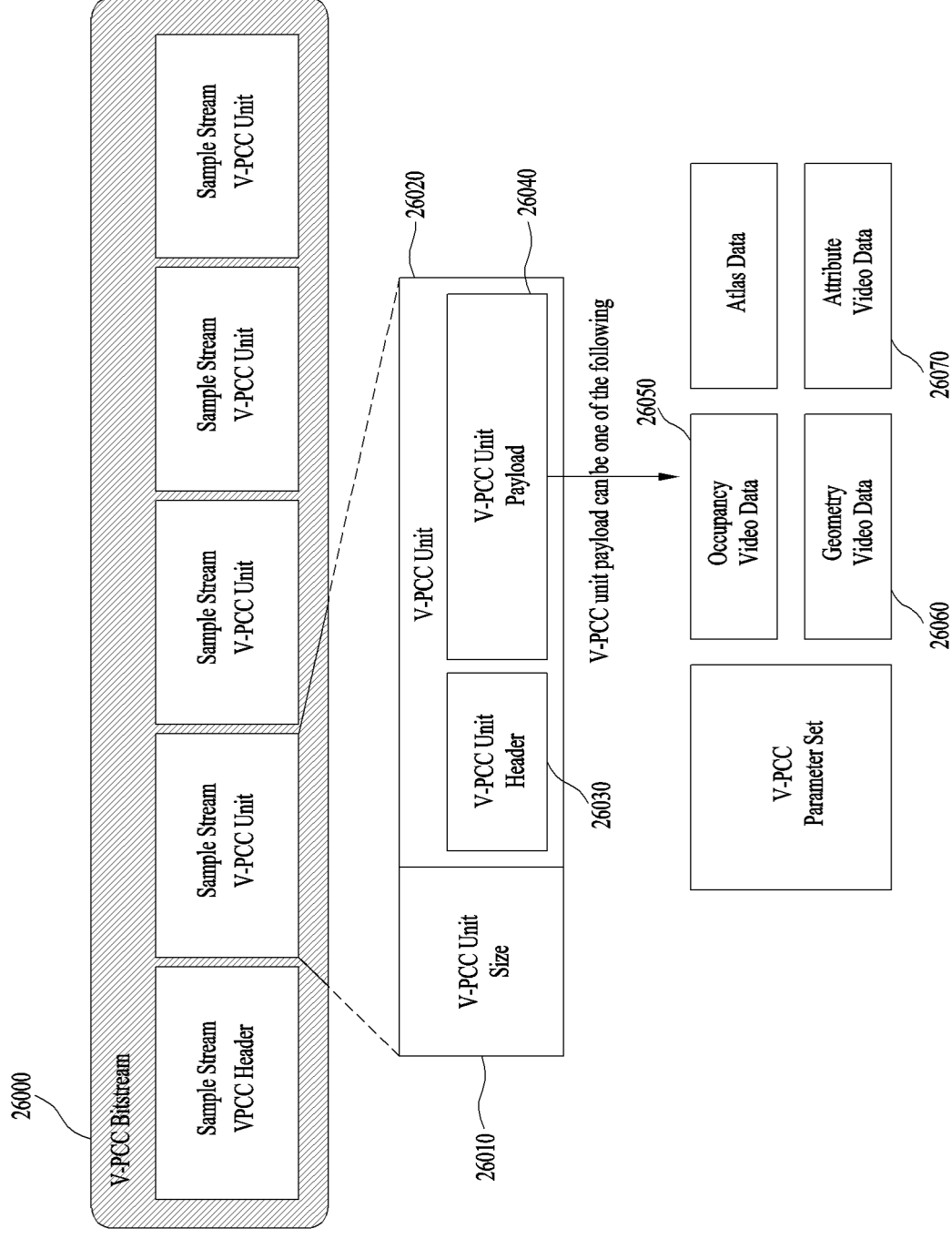


FIG. 27

vpcc_unit(numBytesInVPCCUnit) {	Descriptor
vpcc_unit_header()	
vpcc_unit_payload()	
while(more_data_in_vpcc_unit)	
trailing_zero_8bits /* equal to 0x00 */	f(8)
}	

vpcc_unit_header() {	Descriptor
vuh_unit_type	u(5)
if(vuh_unit_type == VPCC_AVD vuh_unit_type == VPCC_GVD	
vuh_unit_type == VPCC_OVD vuh_unit_type == VPCC_AD) {	
vuh_vpcc_parameter_set_id	u(4)
vuh_atlas_id	u(6)
}	
if(vuh_unit_type == VPCC_AVD) {	
vuh_attribute_index	u(7)
vuh_attribute_partition_index	u(5)
vuh_map_index	u(4)
vuh_auxiliary_video_flag	u(1)
} else if(vuh_unit_type == VPCC_GVD) {	
vuh_map_index	u(4)
vuh_auxiliary_video_flag	u(1)
vuh_reserved_zero_12bits	u(12)
} else if(vuh_unit_type == VPCC_OVD vuh_unit_type == VPCC_AD)	
vuh_reserved_zero_17bits	u(17)
else	
vuh_reserved_zero_27bits	u(27)
}	

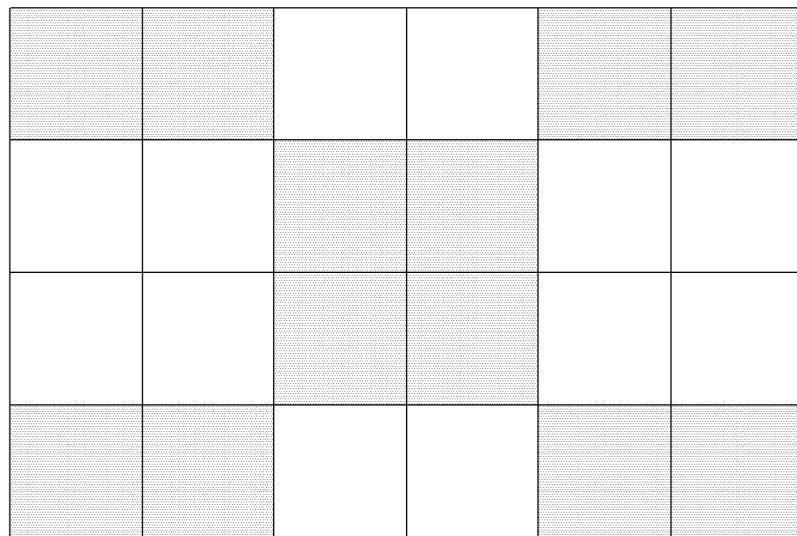
FIG. 28

vpcc_unit_payload() {	Descriptor
if(vuh_unit_type == VPCC_VPS)	
vpcc_parameter_set()	
else if(vuh_unit_type == VPCC_AD)	
atlas_sub_bitstream()	
else if(vuh_unit_type == VPCC_OVD	
vuh_unit_type == VPCC_GVD	
vuh_unit_type == VPCC_AVD)	
video_sub_bitstream()	
}	

FIG. 29

vpcc_parameter_set() {	Descriptor
profile_tier_level()	
vps_vpcc_parameter_set_id	u(4)
vps_atlas_count_minus1	u(6)
for(j = 0; j < vps_atlas_count_minus1 + 1; j++) {	
vps_frame_width[j]	u(16)
vps_frame_height[j]	u(16)
vps_map_count_minus1[j]	u(4)
if(vps_map_count_minus1[j] > 0)	
vps_multiple_map_streams_present_flag[j]	u(1)
vps_map_absolute_coding_enabled_flag[j][0] = 1	
for(i = 1; i <= vps_map_count_minus1[j]; i++) {	
if(vps_multiple_map_streams_present_flag[j])	
vps_map_absolute_coding_enabled_flag[j][i]	u(1)
else	
vps_map_absolute_coding_enabled_flag[j][i] = 1	
if(vps_map_absolute_coding_enabled_flag[j][i] == 0) {	
if(i > 0)	
vps_map_predictor_index_diff[j][i]	ue(v)
else	
vps_map_predictor_index_diff[j][i] = 0	
}	
}	
vps_auxiliary_video_present_flag[j]	u(1)
occupancy_information(j)	
geometry_information(j)	
attribute_information(j)	
}	
vps_extension_present_flag	u(1)
if(vps_extension_present_flag) {	
vps_extension_length_minus1	ue(v)
for(j = 0; j < vps_extension_length_minus1 + 1; j++) {	
vps_extension_data_byte	u(8)
}	
}	
byte_alignment()	
}	

FIG. 30



 Tile
30010

 Tile Group
30000

FIG. 31

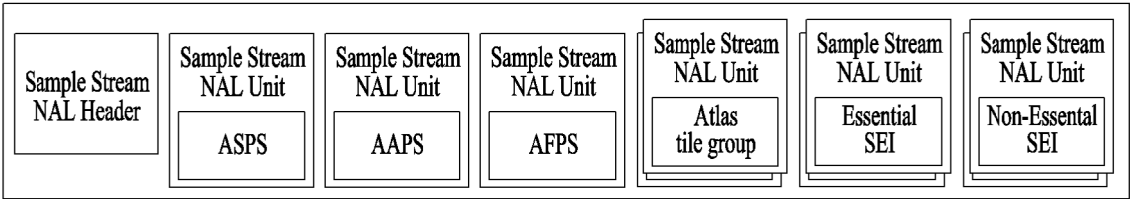


FIG. 32

sample_stream_nal_header() {	Descriptor
ssvh_unit_size_precision_bytes_minus1	u(3)
ssvh_reserved_zero_5bits	u(5)
}	

sample_stream_nal_unit() {	Descriptor
ssvu_nal_unit_size	u(v)
nal_unit(ssvu_nal_unit_size)	
}	

nal_unit(NumBytesInNalUnit) {	Descriptor
nal_unit_header	
NumBytesInRbsp = 0	
for(i = 2; i < NumBytesInNalUnit; i++)	
rbsp_byte[NumBytesInRbsp++]	b(8)
}	

nal_unit_header() {	Descriptor
nal_forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nal_layer_id	u(6)
nal_temporal_id_plus1	u(3)
}	

FIG. 33

nal_unit_type	Name of nal_unit_type	Content of NAL unit and Rbsp ⁸ syntax structure	NAL unit type class
0	NAL_TRAIL	Content of NAL unit and Rbsp ⁸ syntax structure	ACL
		Coded tile group of a non-TSA, non STSA trailing atlas frame	
		atlas_tile_group_layer_rbsp()	
1	NAL_TSA	Coded tile group of a TSA atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
2	NAL_STSA	Coded tile group of a STSA atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
3	NAL_RADL	Coded tile group of a RADL atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
4	NAL_RASL	Coded tile group of a RASL atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
5	NAL_SKIP	Coded tile group of a skipped atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
6..9	NAL_RSV_ACL_6.. NAL_RSV_ACL_9	Reserved non-IRAP ACL NAL unit types	ACL
10	NAL_BLA_W_LP	Coded tile group of a BLA atlas frame	ACL
11	NAL_BLA_W_RADL	atlas_tile_group_layer_rbsp()	
12	NAL_BLA_N_LP		
13	NAL_GBLA_W_LP	Coded tile group of a GBLA atlas frame	ACL
14	NAL_GBLA_W_RADL	atlas_tile_group_layer_rbsp()	
15	NAL_GBLA_N_LP		
16	NAL_IDR_W_RADL	Coded tile group of an IDR atlas frame	ACL
17	NAL_IDR_N_LP	atlas_tile_group_layer_rbsp()	
18	NAL_GIDR_W_RADL	Coded tile group of a GIDR atlas frame	ACL
19	NAL_GIDR_N_LP	atlas_tile_group_layer_rbsp()	
20	NAL_CRA	Coded tile group of a CRA atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
21	NAL_GCRA	Coded tile group of a GCRA atlas frame	ACL
		atlas_tile_group_layer_rbsp()	
22	NAL_IRAP_ACL_22	Reserved IRAP ACL NAL unit types	ACL
23	NAL_IRAP_ACL_23		

24..31	NAL_RSV_ACL_24.. NAL_RSV_ACL_31	Reserved non-IRAP ACL NAL unit types	ACL
32	NAL_ASPS	Atlas sequence parameter set atlas_sequence_parameter_set_rbsp()	non-ACL
33	NAL_AFPS	Atlas frame parameter set atlas_frame_parameter_set_rbsp()	non-ACL
34	NAL_AUD	Access unit delimiter access_unit_delimiter_rbsp()	non-ACL
35	NAL_VPCC_AUD	V-PCC access unit delimiter access_unit_delimiter_rbsp()	non-ACL
36	NAL_EOS	End of sequence end_of_seq_rbsp()	non-ACL
37	NAL_EOB	End of bitstream end_of_atlas_sub_bitstream_rbsp()	non-ACL
38	NAL_FD	Filler filler_data_rbsp()	non-ACL
39	NAL_PREFIX_NSEI	Non-essential supplemental enhancement information sei_rbsp()	non-ACL
40	NAL_SUFFIX_NSEI		
41	NAL_PREFIX_ESEI	Essential supplemental enhancement information sei_rbsp()	non-ACL
42	NAL_SUFFIX_ESEI		
43	NAL_AAPS	Atlas adaptation parameter set atlas_adaptation_parameter_set_rbsp()	non-ACL
44..47	NAL_RSV_NACL_44 NAL_RSV_NACL_47	Reserved non-ACL NAL unit types	non-ACL
48..63	NAL_UNSPEC_48 NAL_UNSPEC_63	Unspecified non-ACL NAL unit types	non-ACL

FIG. 34

atlas_sequence_parameter_set_rbsp() {	Descriptor
asps_atlas_sequence_parameter_set_id	ue(v)
asps_frame_width	u(16)
asps_frame_height	u(16)
asps_log2_patch_packing_block_size	u(3)
asps_log2_max_atlas_frame_order_cnt_lsb_minus4	ue(v)
asps_max_dec_atlas_frame_buffering_minus1	ue(v)
asps_long_term_ref_atlas_frames_flag	u(1)
asps_num_ref_atlas_frame_lists_in_asps	ue(v)
for(i = 0; i < asps_num_ref_atlas_frame_lists_in_asps; i++)	
ref_list_struct(i)	
asps_use_eight_orientations_flag	u(1)
asps_extended_projection_enabled_flag	u(1)
asps_normal_axis_limits_quantization_enabled_flag	u(1)
asps_normal_axis_max_delta_value_enabled_flag	u(1)
asps_remove_duplicate_point_enabled_flag	u(1)
asps_pixel_deinterleaving_enabled_flag	u(1)
asps_patch_precedence_order_flag	u(1)
asps_patch_size_quantizer_present_flag	u(1)
asps_raw_patch_enabled_flag	u(1)
asps_eom_patch_enabled_flag	u(1)
if(asps_raw_patch_enabled_flag asps_eom_patch_enabled_flag)	
asps_auxiliary_video_enabled_flag	u(1)
asps_point_local_reconstruction_enabled_flag	u(1)
asps_map_count_minus1	u(4)
if(asps_pixel_deinterleaving_enabled_flag)	
for(j = 0; j <= asps_map_count_minus1; j++)	
asps_pixel_deinterleaving_map_flag[j]	u(1)
if(asps_eom_patch_enabled_flag && asps_map_count_minus1 == 0)	
asps_eom_fix_bit_count_minus1	u(4)
if(asps_point_local_reconstruction_enabled_flag)	
asps_point_local_reconstruction_information(asps_map_count_minus1)	
if(asps_pixel_deinterleaving_enabled_flag asps_point_local_reconstruction_enabled_flag)	
asps_surface_thickness_minus1	u(8)
asps_vui_parameters_present_flag	u(1)
if(asps_vui_parameters_present_flag)	
vui_parameters()	
asps_extension_flag	u(1)
if(asps_extension_flag)	
while(more_rbsp_data())	
asps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

FIG. 35

atlas_frame_parameter_set_rbsp() {	Descriptor
afps_atlas_frame_parameter_set_id	ue(v)
afps_atlas_sequence_parameter_set_id	ue(v)
atlas_frame_tile_information()	
afps_output_flag_present_flag	u(1)
afps_num_ref_idx_default_active_minus1	ue(v)
afps_additional_lt_afoc_lsb_len	ue(v)
afps_3d_pos_x_bit_count_minus1	u(5)
afps_3d_pos_y_bit_count_minus1	u(5)
afps_lod_mode_enabled_flag	u(1)
afps_override_eom_for_depth_flag	u(1)
if(afps_override_eom_for_depth_flag) {	
afps_eom_number_of_patch_bit_count_minus1	u(4)
afps_eom_max_bit_count_minus1	u(4)
}	
afps_raw_3d_pos_bit_count_explicit_mode_flag	u(1)
afps_fixed_camera_model_flag	u(1)
afps_extension_flag	u(1)
if(afps_extension_flag)	
while(more_rbsp_data())	
afps_extension_data_flag	u(1)
rbps_trailing_bits()	
}	

FIG. 36

atlas_frame_tile_information() {	Descriptor
afti_single_tile_in_atlas_frame_flag	u(1)
if(!afti_single_tile_in_atlas_frame_flag) {	
afti_uniform_tile_spacing_flag	u(1)
if(afti_uniform_tile_spacing_flag) {	
afti_tile_cols_width_minus1	ue(v)
afti_tile_rows_height_minus1	ue(v)
} else {	
afti_num_tile_columns_minus1	ue(v)
afti_num_tile_rows_minus1	ue(v)
for(i = 0; i < afti_num_tile_columns_minus1; i++)	
afti_tile_column_width_minus1[i]	ue(v)
for(i = 0; i < afti_num_tile_rows_minus1; i++)	
afti_tile_row_height_minus1[i]	ue(v)
}	
afti_single_tile_per_tile_group_flag	u(1)
if(!afti_single_tile_per_tile_group_flag) {	
afti_num_tile_groups_in_atlas_frame_minus1	ue(v)
for(i = 0; i < afti_num_tile_groups_in_atlas_frame_minus1 + 1; i++) {	
if(i > 0)	
afti_top_left_tile_idx[i]	u(v)
afti_bottom_right_tile_idx_delta[i]	u(v)
}	
}	
afti_signalled_tile_group_id_flag	u(1)
if(afti_signalled_tile_group_id_flag) {	
afti_signalled_tile_group_id_length_minus1	ue(v)
for(i = 0; i < afti_num_tile_groups_in_atlas_frame_minus1 + 1; i++)	
afti_tile_group_id[i]	u(v)
}	
}	
}	

FIG. 37

atlas_adaptation_parameter_set_rbsp() {	Descriptor
aaps_atlas_adaptation_parameter_set_id	ue(v)
aaps_camera_parameters_present_flag	u(1)
if (aaps_camera_parameters_present_flag)	
atlas_camera_parameters()	
aaps_extension_flag	u(1)
if(aaps_extension_flag)	
while(more_rbsp_data())	
aaps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

FIG. 38

atlas_camera_parameters() {	Descriptor
acp_camera_model	u(8)
if(acp_camera_model == 1) {	
acp_scale_enabled_flag	u(1)
acp_offset_enabled_flag	u(1)
acp_rotation_enabled_flag	u(1)
if(acp_scale_enabled_flag)	
for(d = 0; d < 3; d++)	
acp_scale_on_axis[d]	u(32)
if(acp_offset_enabled_flag)	
for(d = 0; d < 3; d++)	
acp_offset_on_axis[d]	i(32)
if(acp_rotation_enabled_flag) {	
acp_rotation_qx	i(16)
acp_rotation_qy	i(16)
acp_rotation_qz	i(16)
}	
}	
}	

FIG. 39

atlas_tile_group_layer_rbsp() {	Descriptor
atlas_tile_group_header()	
if(atgh_type != SKIP_TILE_GRP)	
atlas_tile_group_data_unit()	
rbsp_trailing_bits()	
}	

atlas_tile_group_header() {	Descriptor
atgh_atlas_frame_parameter_set_id	ue(v)
atgh_atlas_adaptation_parameter_set_id	ue(v)
atgh_address	u(v)
atgh_type	ue(v)
if(afps_output_flag_present_flag)	
atgh_atlas_output_flag	u(1)
atgh_atlas_frm_order_cnt_lsb	u(v)
if(asps_num_ref_atlas_frame_lists_in_asps > 0)	
atgh_ref_atlas_frame_list_sps_flag	u(1)
if(atgh_ref_atlas_frame_list_sps_flag == 0)	
ref_list_struct(asps_num_ref_atlas_frame_lists_in_asps)	
else if(asps_num_ref_atlas_frame_lists_in_asps > 1)	
atgh_ref_atlas_frame_list_idx	u(v)
for(j = 0; j < NumLtrAtlasFrmEntries; j++) {	
atgh_additional_afoc_lsb_present_flag[j]	u(1)
if(atgh_additional_afoc_lsb_present_flag[j])	
atgh_additional_afoc_lsb_val[j]	u(v)
}	
if(atgh_type != SKIP_TILE_GRP) {	
if(asps_normal_axis_limits_quantization_enabled_flag) {	
atgh_pos_min_z_quantizer	u(5)
if(asps_normal_axis_max_delta_value_enabled_flag)	
atgh_pos_delta_max_z_quantizer	u(5)
}	
if(asps_patch_size_quantizer_present_flag) {	
atgh_patch_size_x_info_quantizer	u(3)
atgh_patch_size_y_info_quantizer	u(3)
}	
if(afps_raw_3d_pos_bit_count_explicit_mode_flag)	
atgh_raw_3d_pos_axis_bit_count_minus1	u(v)
if(atgh_type == P_TILE_GRP && num_ref_entries[RlsIdx] > 1) {	
atgh_num_ref_idx_active_override_flag	u(1)
if(atgh_num_ref_idx_active_override_flag)	
atgh_num_ref_idx_active_minus1	ue(v)
}	
}	
byte_alignment()	
}	

FIG. 40

ref_list_struct(rlsIdx) {	Descriptor
num_ref_entries[rlsIdx]	ue(v)
for(i = 0; i < num_ref_entries[rlsIdx]; i++) {	
if(asps_long_term_ref_atlas_frames_flag)	
st_ref_atlas_frame_flag[rlsIdx][i]	u(1)
if(st_ref_atlas_frame_flag[rlsIdx][i]) {	
abs_delta_afoc_st[rlsIdx][i]	ue(v)
if(abs_delta_afoc_st[rlsIdx][i] > 0)	
strpf_entry_sign_flag[rlsIdx][i]	u(1)
} else	
afoc_lsb_lt[rlsIdx][i]	u(v)
}	
}	

FIG. 41

atlas_tile_group_data_unit() {	Descriptor
p = 0	
atgdu_patch_mode[p]	ue(v)
while(atgdu_patch_mode[p] != I_END && atgdu_patch_mode[p] != P_END){	
patch_information_data(p, atgdu_patch_mode[p])	
p++	
atgdu_patch_mode[p]	ue(v)
}	
AtgduTotalNumberOfPatches = p	
byte_alignment()	
}	

FIG. 42

patch_information_data (patchIdx, patchMode) {	Descriptor
if(atgh_type == SKIP_TILE_GR)	
skip_patch_data_unit(patchIdx)	
else if(atgh_type == P_TILE_GR) {	
if(patchMode == P_SKIP)	
skip_patch_data_unit(patchIdx)	
else if(patchMode == P_MERGE)	
merge_patch_data_unit(patchIdx)	
else if(patchMode == P_INTRA)	
patch_data_unit(patchIdx)	
else if(patchMode == P_INTER)	
inter_patch_data_unit(patchIdx)	
else if(patchMode == P_RAW)	
raw_patch_data_unit(patchIdx)	
else if(patchMode == P_EOM)	
eom_patch_data_unit(patchIdx)	
}	
else if(atgh_type == I_TILE_GR) {	
if(patchMode == I_INTRA)	
patch_data_unit(patchIdx)	
else if(patchMode == I_RAW)	
raw_patch_data_unit(patchIdx)	
else if(patchMode == I_EOM)	
eom_patch_data_unit(patchIdx)	
}	
}	

FIG. 43

patch_data_unit(patchIdx) {	Descriptor
pdu_2d_pos_x[patchIdx]	ue(v)
pdu_2d_pos_y[patchIdx]	ue(v)
pdu_2d_size_x_minus1[patchIdx]	ue(v)
pdu_2d_size_y_minus1[patchIdx]	ue(v)
pdu_3d_pos_x[patchIdx]	u(v)
pdu_3d_pos_y[patchIdx]	u(v)
pdu_3d_pos_min_z[patchIdx]	u(v)
if(asps_normal_axis_max_delta_value_enabled_flag)	
pdu_3d_pos_delta_max_z[patchIdx]	u(v)
pdu_projection_id[patchIdx]	u(v)
pdu_orientation_index[patchIdx]	u(v)
if(afps_lod_mode_enabled_flag) {	
pdu_lod_enabled_flag[patchIndex]	u(1)
if(pdu_lod_enabled_flag[patchIndex] > 0) {	
pdu_lod_scale_x_minus1[patchIndex]	ue(v)
pdu_lod_scale_y[patchIndex]	ue(v)
}	
}	u(v)
if(asps_point_local_reconstruction_enabled_flag)	
point_local_reconstruction_data(patchIdx)	
}	

FIG. 44

```
class VPCCAtlasRelationBox extends Box('vpcA') {
    VPCCAtlasRelationRecord VPCCAtlas;
}

aligned(8) class VPCCAtlasRelationRecord {
    bit(1) entry_point_flag;
    unsigned int(8) num_atlas_data;
    for (i=0; i<num_atlas_data; i++) {
        unsigned int(8) atlas_id;
        unsigned int(8) num_atlas_tile_groups;
        for (j=0; j<num_atlas_tile_groups; j++) {
            unsigned int(16) atlas_tile_group_id;
        }
        unsigned int(8) num_vpcc_component_track_groups;
        for (k=0; k<num_vpcc_component_track_groups; k++) {
            unsigned int(32) vpcc_component_track_group_id;
        }
        unsigned int(8) num_vpcc_component_tracks;
        for (p=0; p<num_vpcc_component_tracks; p++) {
            unsigned int(32) vpcc_component_track_id;
        }
    }
}
```

FIG. 45

```
class VPCCGlobalAtlasInformationBox extends Box('vpcA') {
    VPCCGlobalAtlasInformationRecord GlobalAtlas;
}

aligned(8) class VPCCGlobalAtlasInformationRecord {
    unsigned int(8) num_vpcc_tracks;
    for (i=0; i<num_vpcc_tracks; i++) {
        unsigned int(8) track_id;
        bit(1) entry_point_flag;
        unsigned int(8) num_atlas_data;
        for(j=0; j<num_atlas_data; j++) {
            unsigned int(8) atlas_id;
            unsigned int(8) num_atlas_tile_groups;
            for (k=0; k<num_atlas_tile_groups; k++) {
                unsigned int(16) atlas_tile_group_id;
            }
            unsigned int(8) num_vpcc_component_track_groups;
            for (k=0; k<num_vpcc_component_track_groups; k++) {
                unsigned int(32) vpcc_component_track_group_id;
            }
            unsigned int(8) num_vpcc_component_tracks;
            for (p=0; p<num_vpcc_component_tracks; p++) {
                unsigned int(32) vpcc_component_track_id;
            }
        }
    }
}
```

FIG. 46

```
aligned(8) class DynamicAtlasRelationSampleGroupDescriptionEntry()  
extends SampleGroupDescriptionEntry('dyar') {  
    VPCCAtlasRelationBox();  
}
```

```
aligned(8) class DynamicGlobalAtlasInformationSampleGroupDescriptionEntry()  
extends SampleGroupDescriptionEntry('dyar') {  
    VPCCGlobalAtlasInformationBox();  
}
```

FIG. 47

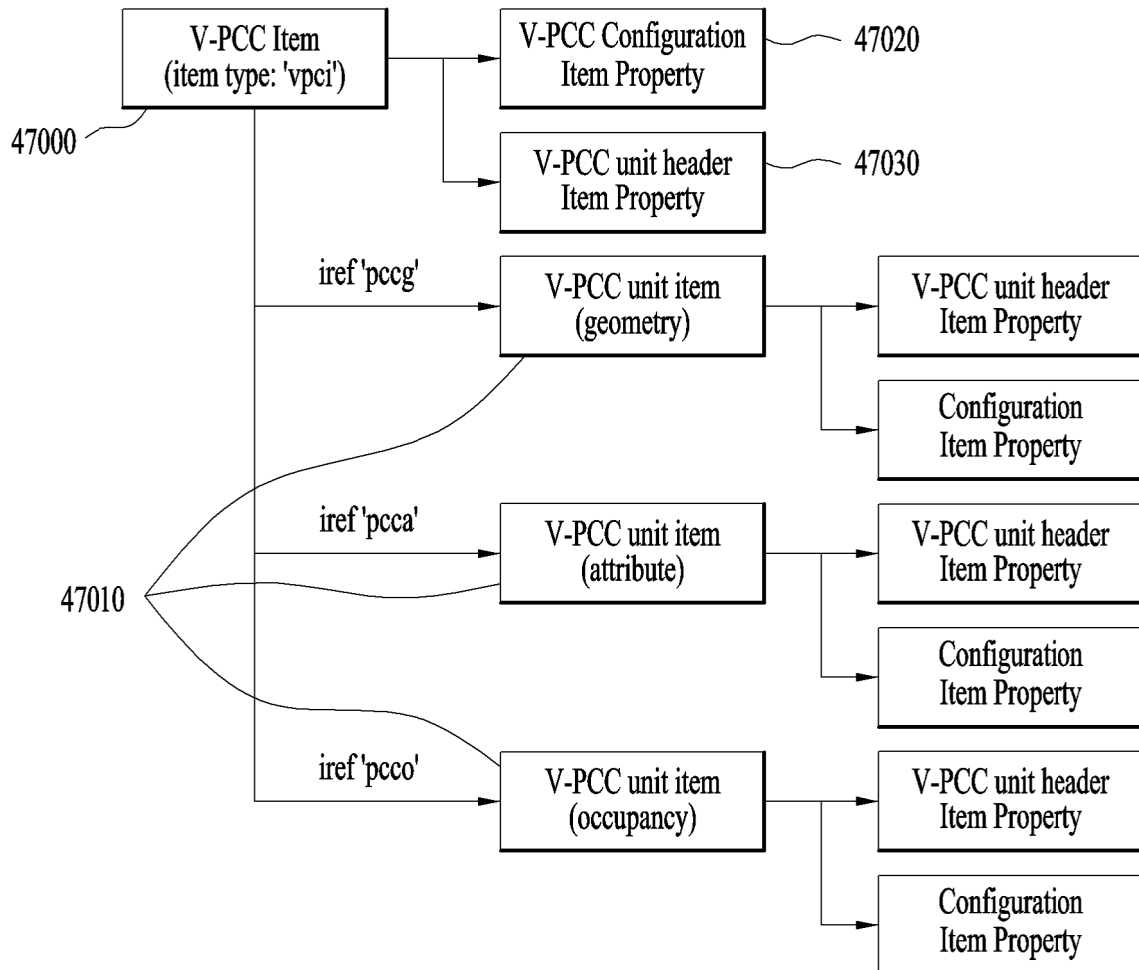


FIG. 48

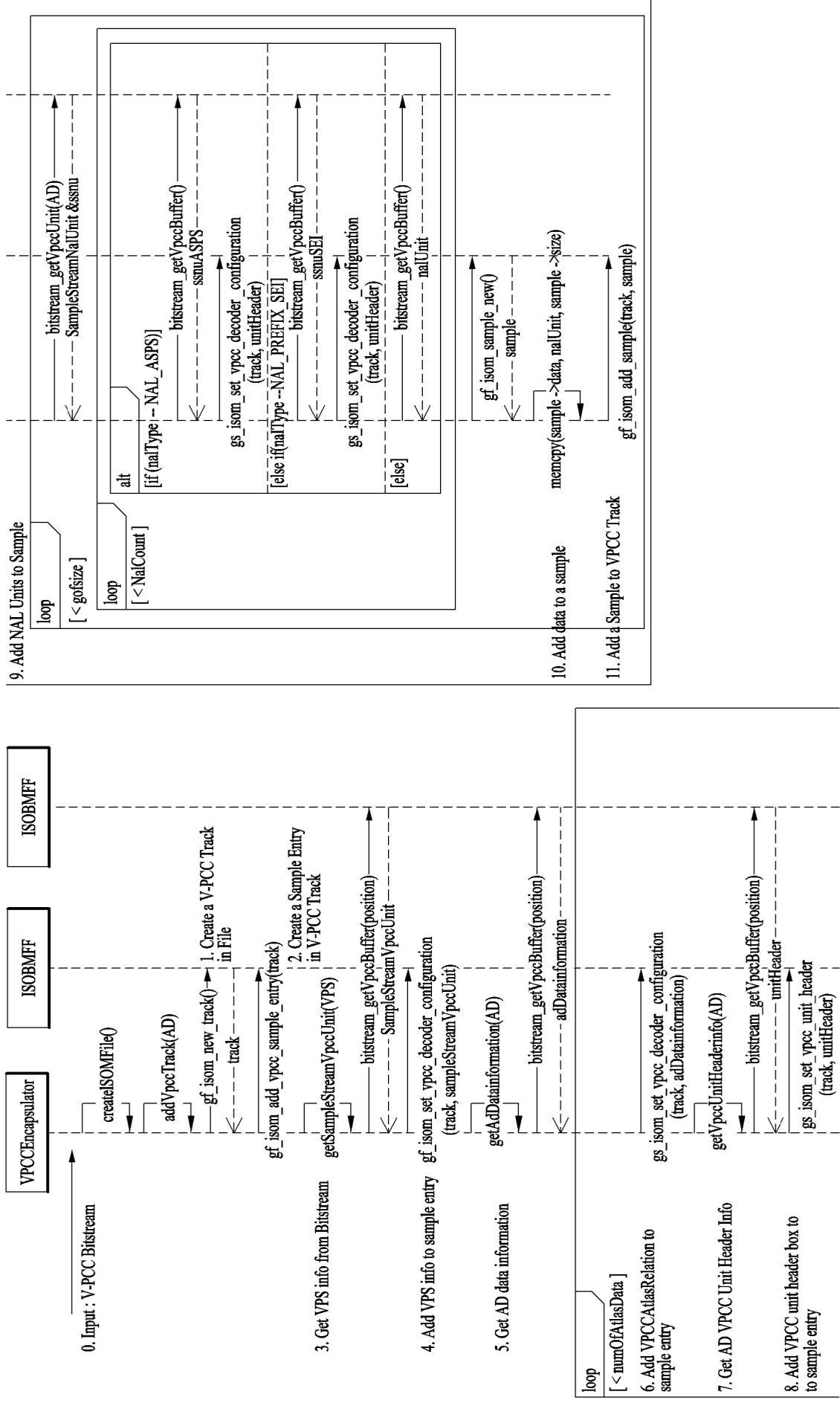


FIG. 49

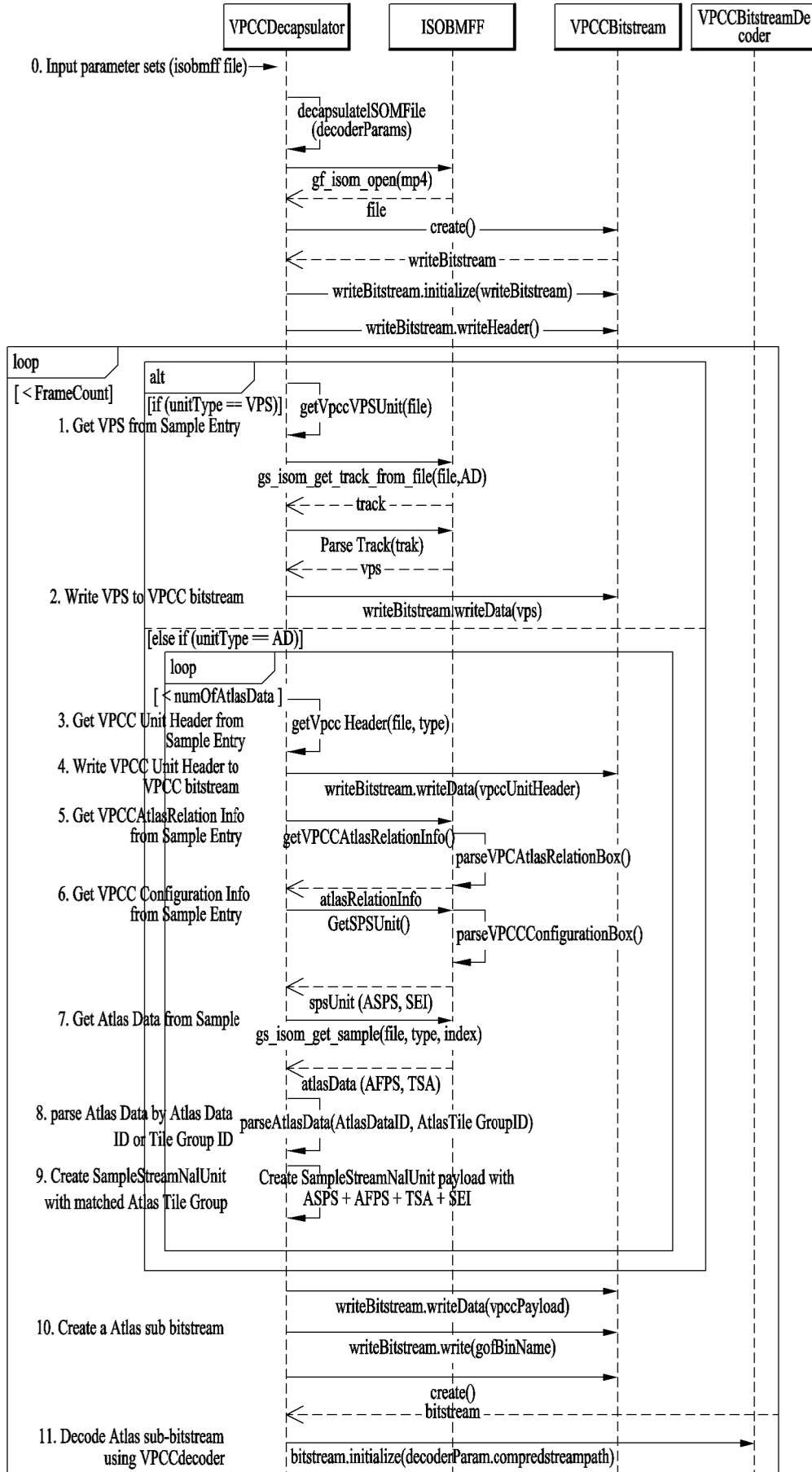


FIG. 50

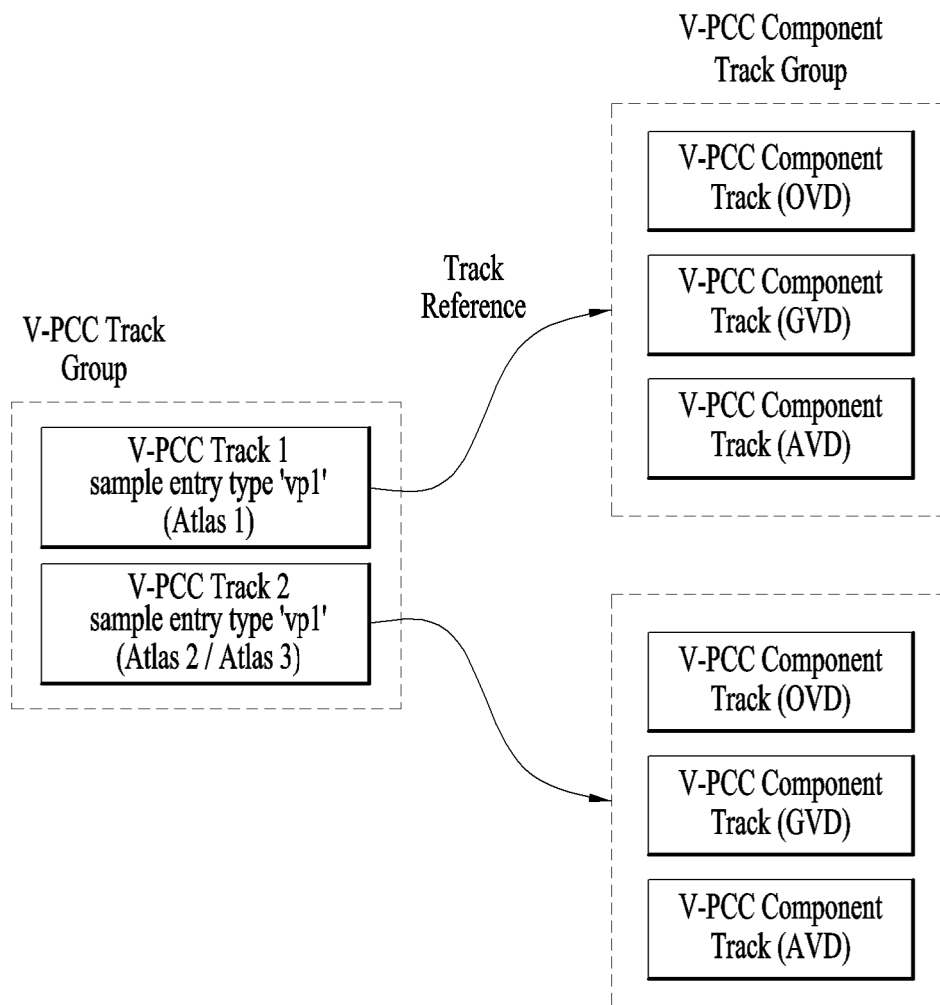


FIG. 51

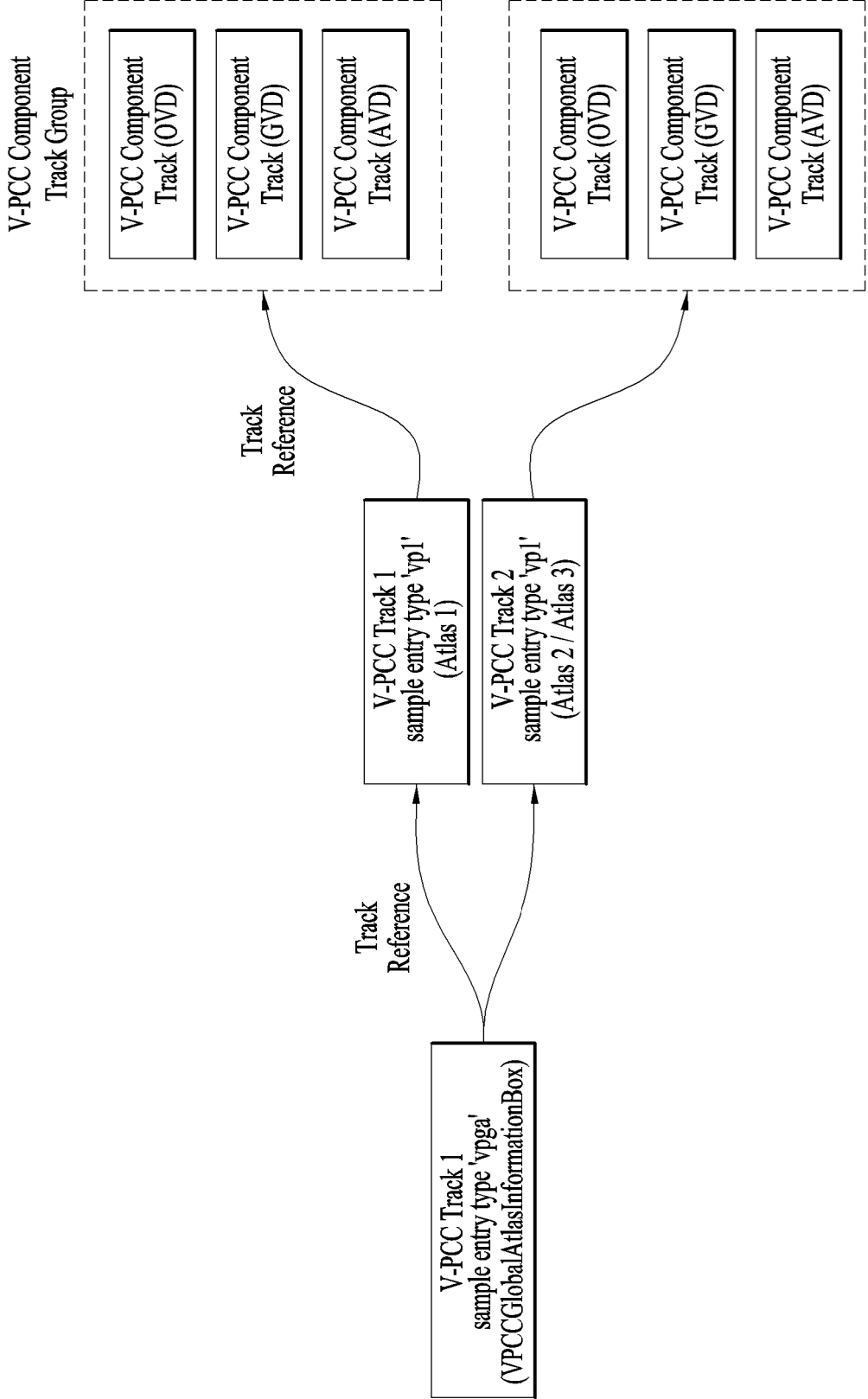


FIG. 52

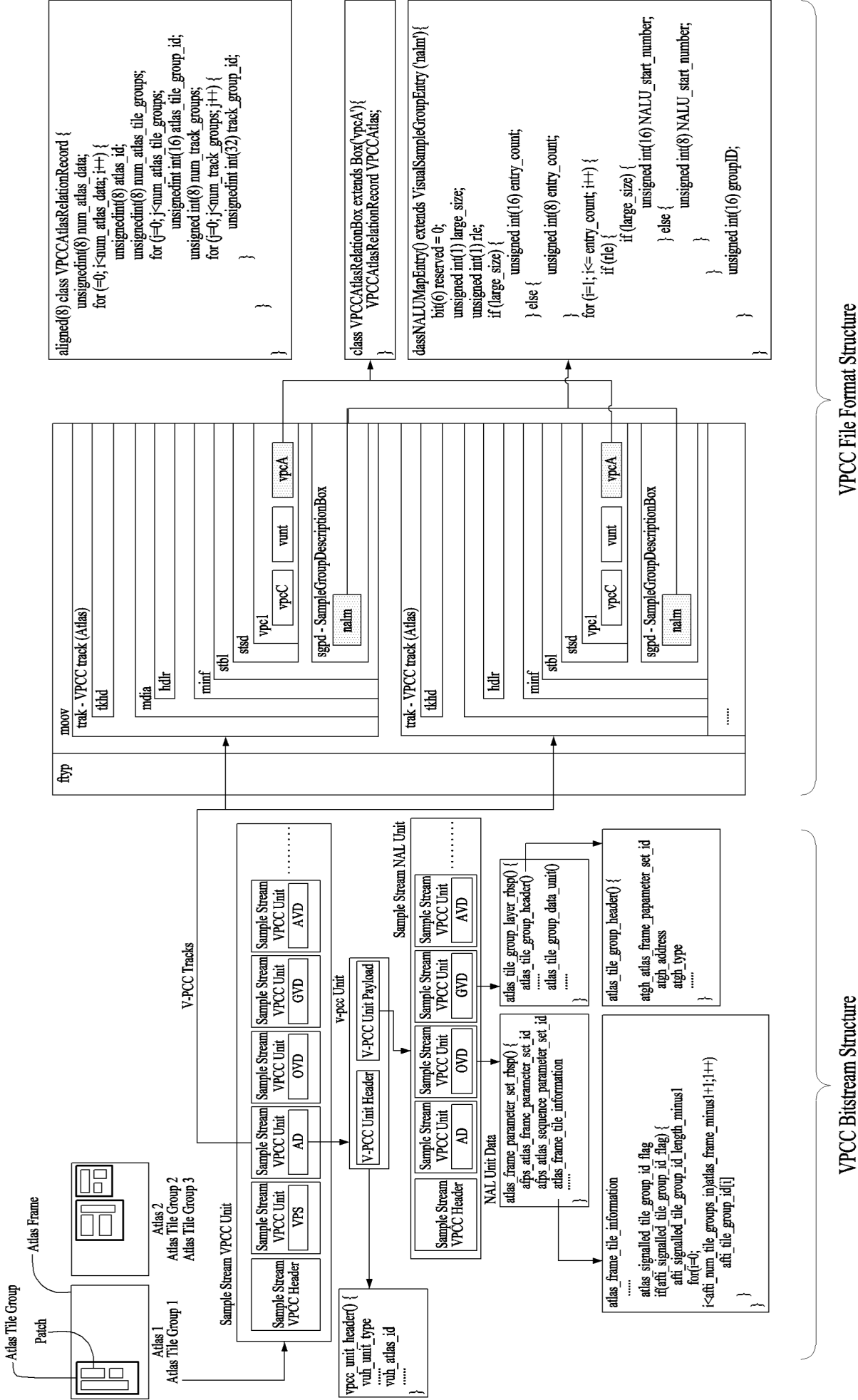


FIG. 53

scene_object_information(payloadSize) {	Descriptor
soi_cancel_flag	u(1)
soi_num_object_updates	ue(v)
if (soi_num_object_updates > 0) {	
soi_simple_objects_flag	u(1)
if (soi_simple_objects_flag == 0) {	
soi_object_label_present_flag	u(1)
soi_priority_present_flag	u(1)
soi_object_hidden_present_flag	u(1)
soi_object_dependency_present_flag	u(1)

soi_visibility_cones_present_flag	u(1)
soi_3d_bounding_box_present_flag	u(1)
soi_collision_shape_present_flag	u(1)
soi_point_style_present_flag	u(1)
soi_material_id_present_flag	u(1)
soi_extension_present_flag	u(1)
}	
else {	
soi_object_label_present_flag = 0	
soi_priority_present_flag = 0	
soi_object_hidden_present_flag = 0	
soi_object_dependency_present_flag = 0	
soi_visibility_cones_present_flag = 0	
soi_3d_bounding_box_present_flag = 0	
soi_collision_shape_present_flag = 0	
soi_point_style_present_flag = 0	
soi_material_id_present_flag = 0	
soi_extension_present_flag = 0	
}	
if (soi_3d_bounding_box_present_flag) {	
soi_3d_bounding_box_scale_log2	u(5)
soi_3d_bounding_box_precision_minus8	u(5)
}	

FIG. 54

soi_log2_max_object_idx_updated	u(5)
if(soi_object_dependency_present_flag)	
soi_log2_max_object_dependency_idx	u(5)
for(i = 0; i <= soi_num_object_updates; i++) {	
soi_object_idx[i]	u(v)
k = soi_object_idx[i]	
soi_object_cancel_flag[k]	u(1)
ObjectTracked[k] = ! pi_object_cancel_flag[k]	
if (!soi_object_cancel_flag[k]) {	
if(soi_object_label_present_flag) {	
soi_object_label_update_flag[k]	u(1)
if(soi_object_label_update_flag[k])	
soi_object_label_idx[k]	ue(v)
}	
if(soi_priority_present_flag) {	
soi_priority_update_flag[k]	u(1)
if(soi_priority_update_flag[k])	
soi_priority_value[k]	u(4)
}	
if(soi_object_hidden_present_flag)	
soi_object_hidden_flag[k]	u(1)
if(soi_object_dependency_present_flag) {	
soi_object_dependency_update_flag[k]	u(1)

if(soi_object_dependency_update_flag[k]) {	
soi_object_num_dependencies[k]	u(4)
for(j = 0; j < soi_object_num_dependencies[k]; j++)	
soi_object_dependency_idx[k][j]	u(v)
}	
}	
if(soi_visibility_cones_present_flag) {	
soi_visibility_cones_update_flag[k]	u(1)
if(soi_visibility_cones_update_flag[k]) {	
soi_direction_x[k]	fl(32)
soi_direction_y[k]	fl(32)
soi_direction_z[k]	fl(32)
soi_angle[k]	fl(16)
}	
}	

FIG. 55

if(soi_3d_bounding_box_present_flag) {	
soi_3d_bounding_box_update_flag[k]	u(1)
if(soi_3d_bounding_box_update_flag[k]) {	
soi_3d_bounding_box_x[k]	u(v)
soi_3d_bounding_box_y[k]	u(v)
soi_3d_bounding_box_z[k]	u(v)
soi_3d_bounding_box_delta_x[k]	u(v)
soi_3d_bounding_box_delta_y[k]	u(v)
soi_3d_bounding_box_delta_z[k]	u(v)
}	
}	
if(soi_collision_shape_present_flag) {	
soi_collision_shape_update_flag[k]	u(1)
if(soi_collision_shape_update_flag[k])	
soi_collision_shape_id[k]	u(16)
}	
if(soi_point_style_present_flag) {	
soi_point_style_update_flag[k]	u(1)
if(soi_point_style_update_flag[k]) {	
soi_point_shape_id[k]	u(8)
soi_point_size[k]	u(16)
}	
}	
if(soi_material_id_present_flag) {	
soi_material_id_update_flag[k]	u(1)
if(soi_material_id_update_flag[k])	
soi_material_id[k]	u(16)
}	
}	
}	
}	

FIG. 56

object_label_information (payload_size) {	Descriptor
oli_cancel_flag	u(1)
if (!oli_cancel_flag) {	
oli_label_language_present_flag	u(1)
if(oli_label_language_present_flag) {	
while(!byte_aligned())	
oli_bit_equal_to_zero /* equal to 0 */	f(1)
oli_label_language	st(v)
}	
oli_num_label_updates	ue(v)
for(i = 0; i < oli_num_label_updates ; i++) {	
oli_label_idx[i]	ue(v)
oli_label_cancel_flag	u(1)
LabelAssigned[pi_label_idx[i]] = !pi_label_cancel_flag	
if (!oli_label_cancel_flag) {	
while(!byte_aligned())	
oli_bit_equal_to_zero /* equal to 0 */	f(1)
oli_label[oli_label_idx[i]]	st(v)
}	
}	
}	

FIG. 57

patch_information (payload_size) {	Descriptor
pi_cancel_flag	u(1)
pi_num_tile_group_updates	ue(v)
if (pi_num_tile_group_updates > 0) {	ue(v)
pi_log2_max_object_idx_tracked	u(5)
pi_log2_max_patch_idx_updated	u(4)
}	
for(i = 0; i <= pi_num_tile_group_updates; i++) {	
pi_tile_group_address[i]	ue(v)
j = pi_tile_group_address[i]	
pi_tile_group_cancel_flag[j]	u(1)
pi_num_patch_updates[j]	ue(v)
for(k = 0; k < pi_num_patch_updates[j]; k++) {	
pi_patch_idx[j][k]	u(v)
p = pi_patch_idx[j][k]	
pi_patch_cancel_flag[j][p]	u(1)
if(!pi_patch_cancel_flag[j][p]) {	
pi_patch_number_of_objects_minus1[j][p]	ue(v)
m = pi_patch_number_of_objects_minus1[j][p] + 1	
for(n = 0; n < m; n++)	
pi_patch_object_idx[j][p][n]	u(v)
}	
}	
}	
}	

FIG. 58

volumetric_rectangle_information (payload_size) {	Descriptor
vri_cancel_flag	u(1)
vri_num_rectangles_updates	ue(v)
if (vri_num_rectangles_updates > 0) {	ue(v)
vri_log2_max_object_idx_tracked	u(5)
vri_log2_max_rectangle_idx_updated	u(4)
}	
for(k = 0; k < vri_num_rectangles_updates; k++) {	
vri_rectangle_idx[k]	u(v)
p = vri_rectangle_idx[k]	
vri_rectangle_cancel_flag[p]	u(1)
if(!vri_rectangle_cancel_flag[p]) {	
vri_bounding_box_update_flag[p]	u(1)
if(vri_bounding_box_update_flag[p]) {	
vri_bounding_box_top[p]	u(v)
vri_bounding_box_left[p]	u(v)
vri_bounding_box_width[p]	u(v)
vri_bounding_box_height[p]	u(v)
}	
vri_rectangle_number_of_objects_minus1[p]	ue(v)
m = vri_rectangle_number_of_objects_minus1[p] + 1	
for(n = 0; n < m; n++)	
vri_rectangle_object_idx[p][n]	u(v)
}	
}	
}	

FIG. 59

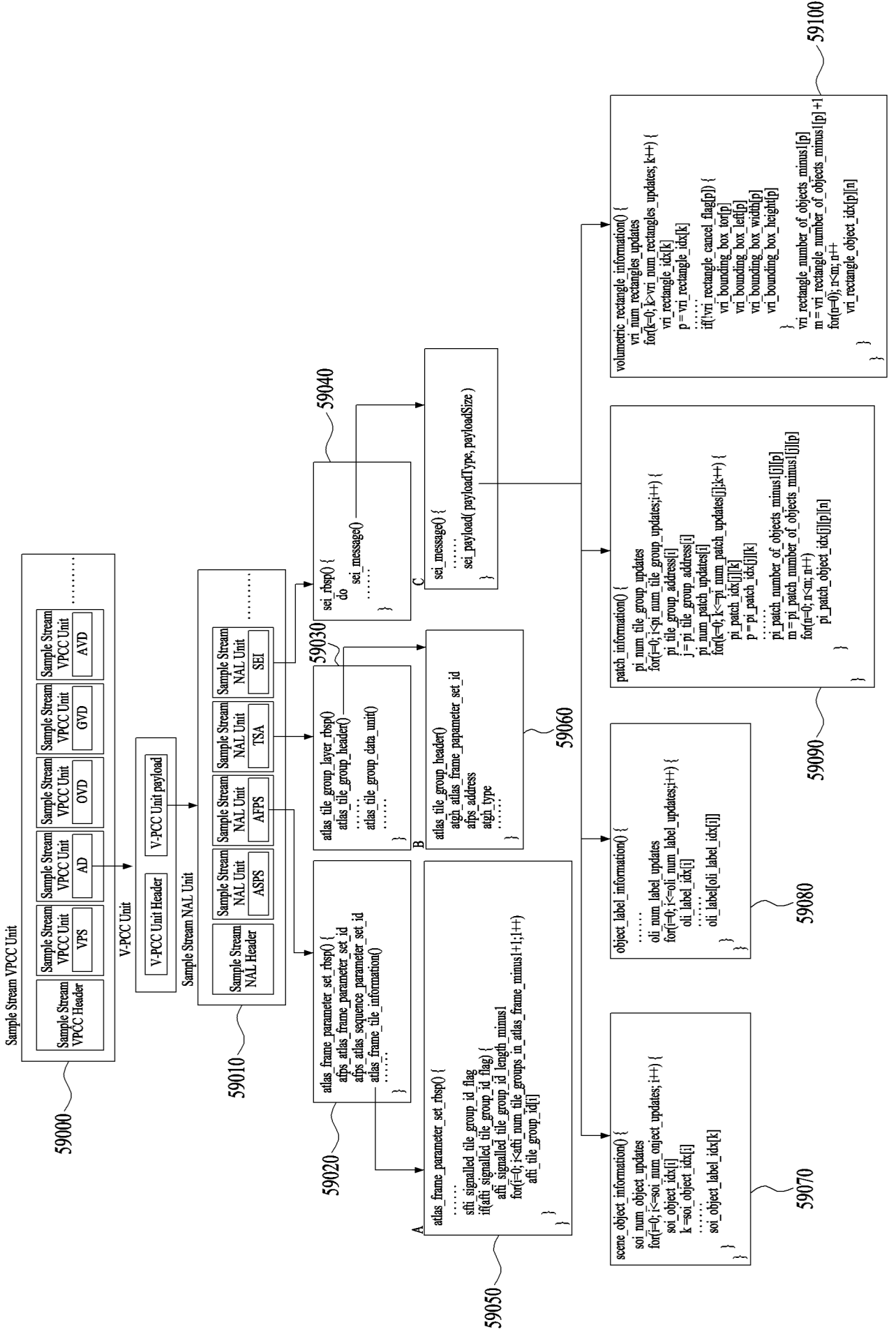


FIG. 60

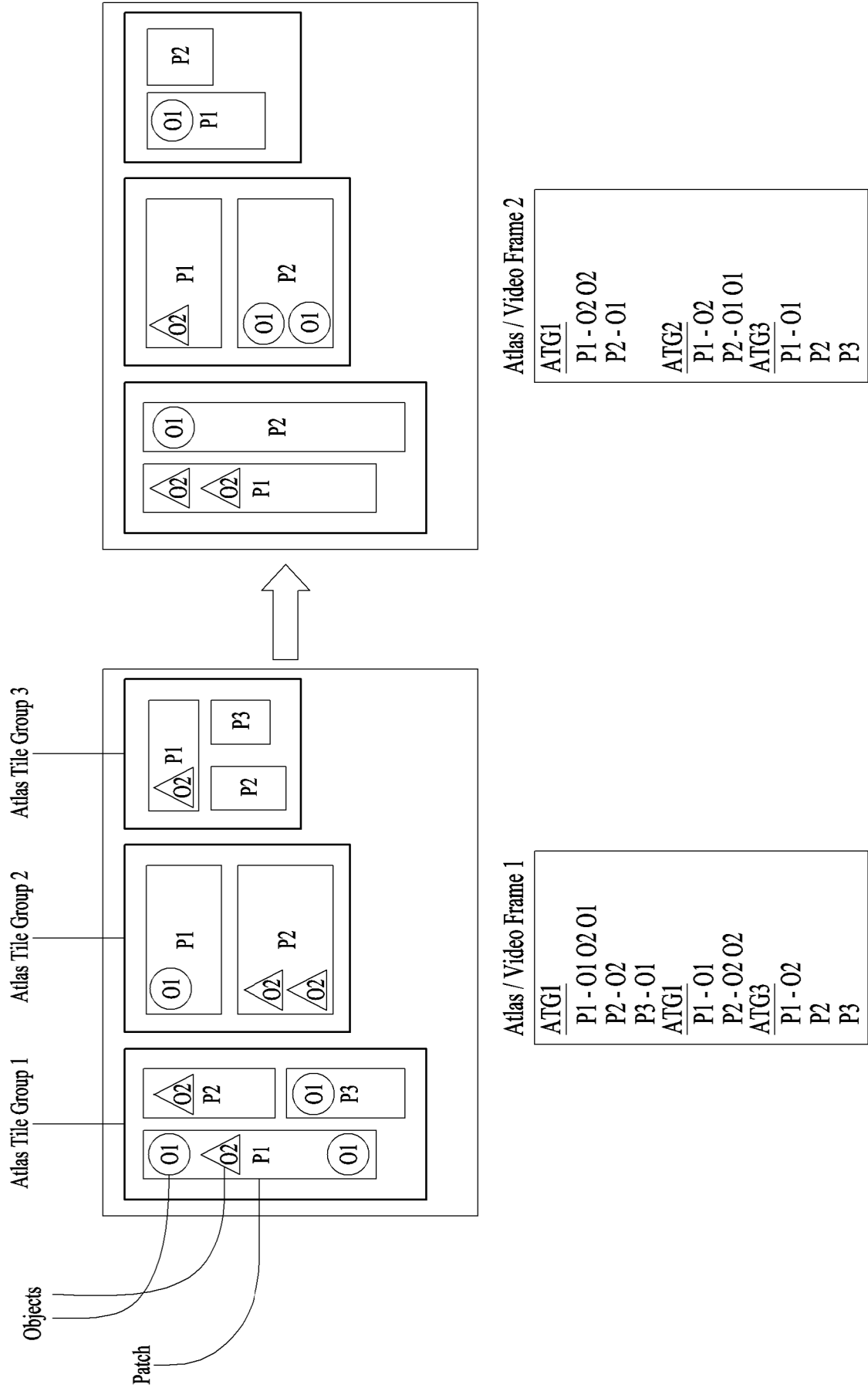


FIG. 61

```
aligned(8) class 3DPoint() {
    unsigned int(16) x;
    unsigned int(16) y;
    unsigned int(16) z;
}
aligned(8) class CuboidRegionStruct() {
    unsigned int(16) cuboid_dx;
    unsigned int(16) cuboid_dy;
    unsigned int(16) cuboid_dz;
}
aligned(8) class 3DSpatialRegionStruct(dimensions_included_flag) {
    unsigned int(16) 3d_region_id;
    3DPoint anchor;
    if (dimensions_included_flag) {
        CuboidRegionStruct();
    }
}
```

FIG. 62

```
aligned(8) class VPCCSpatialRegionsBox extends FullBox('vpsr',0,0) {
    unsigned int(16) num_regions;
    for (i=0; i < num_regions; i++) {
        3DSpatialRegionStruct(1);
        LabelInfoStruct();
        unsigned int(16) num_atlas;
        for(m=0; m< num_atlas ; m++)
        {
            unsigned int(16) atlas_id;

            unsigned int(16) num_region_tile_groups;
            for (k=0; k < num_region_tile_groups; k++) {
                unsigned int(16) region_tile_group_id;
                unsigned int(8) num_patch_updates;
                for (p=0; p < num_patch_updates; p++)
                    unsigned int(16) patch_id;
            }
        }

        unsigned int(8) num_track_groups;
        for (j=0; j < num_track_groups; j++) {
            unsigned int(32) track_group_id;
        }
    }
}

aligned(8) class LabelInfoStruct() {
    unsigned int(16) label_id;
    string label_language;
    string label_name;
}
```

FIG. 63

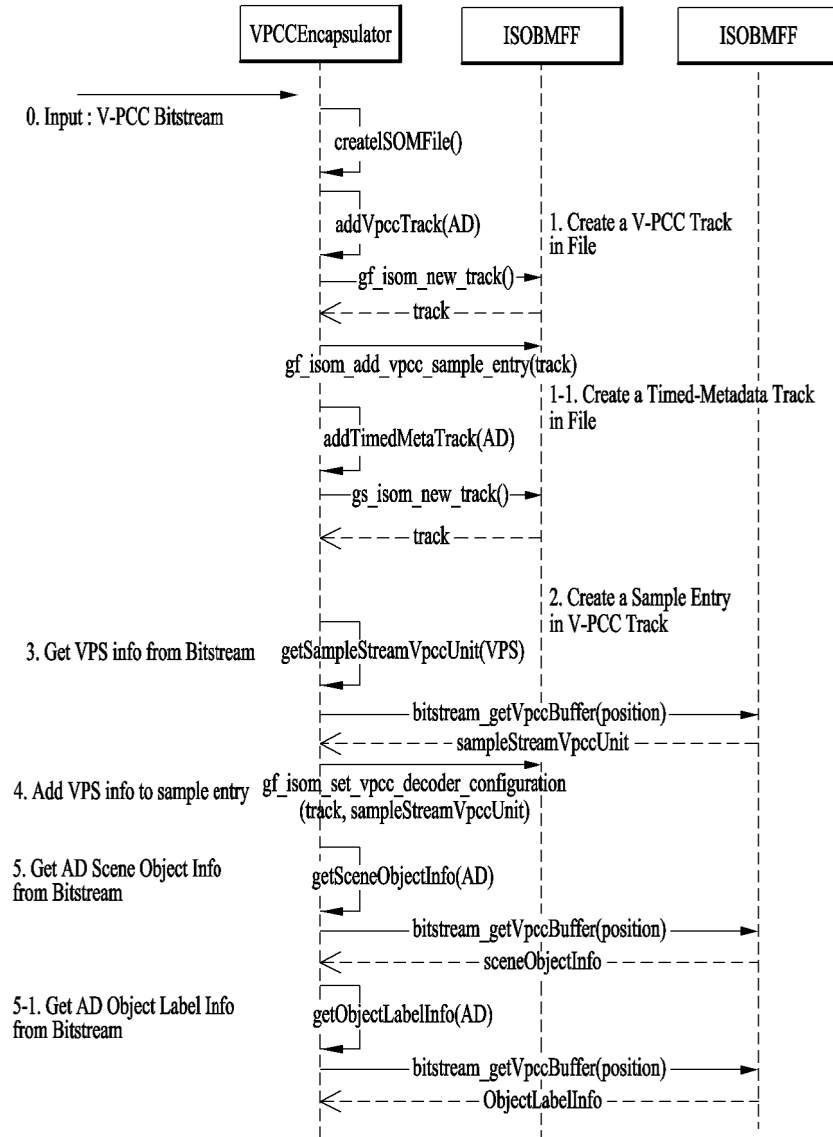


FIG. 64

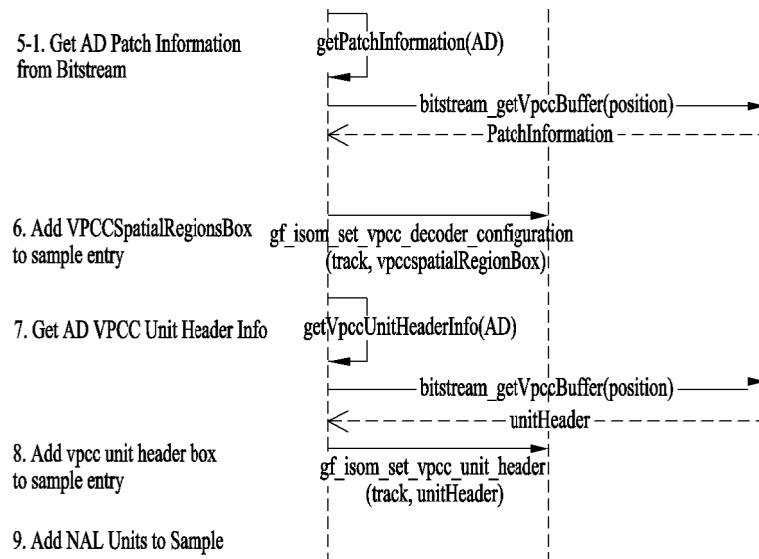


FIG. 65

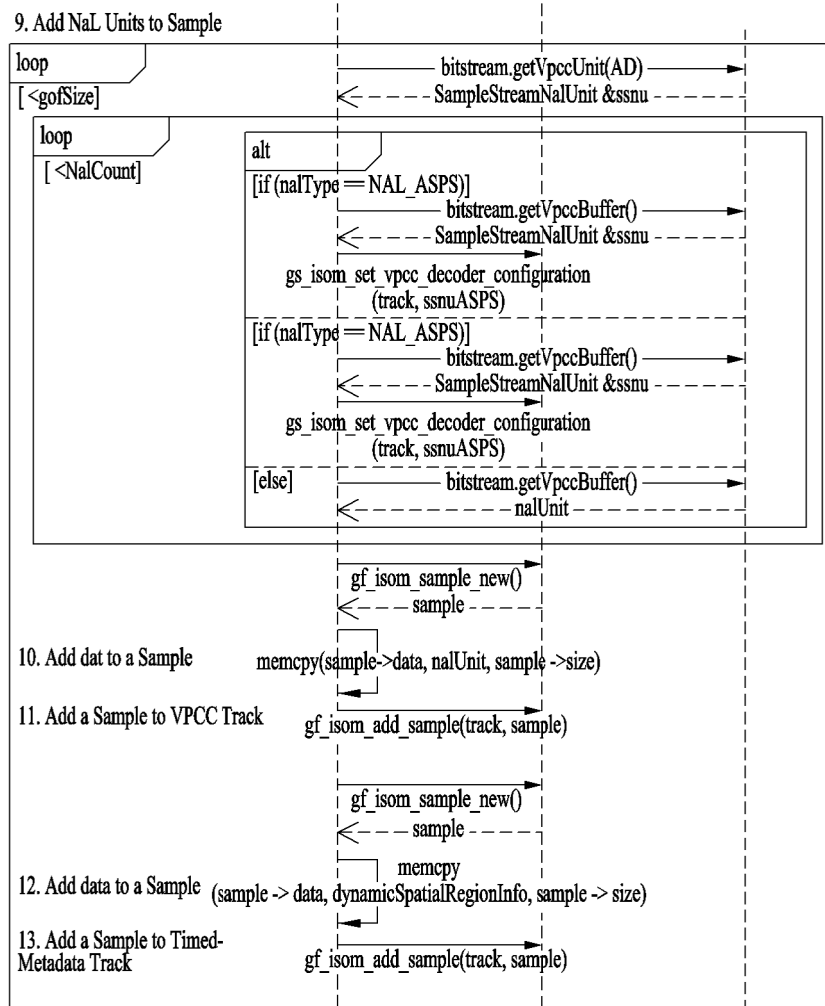


FIG. 66

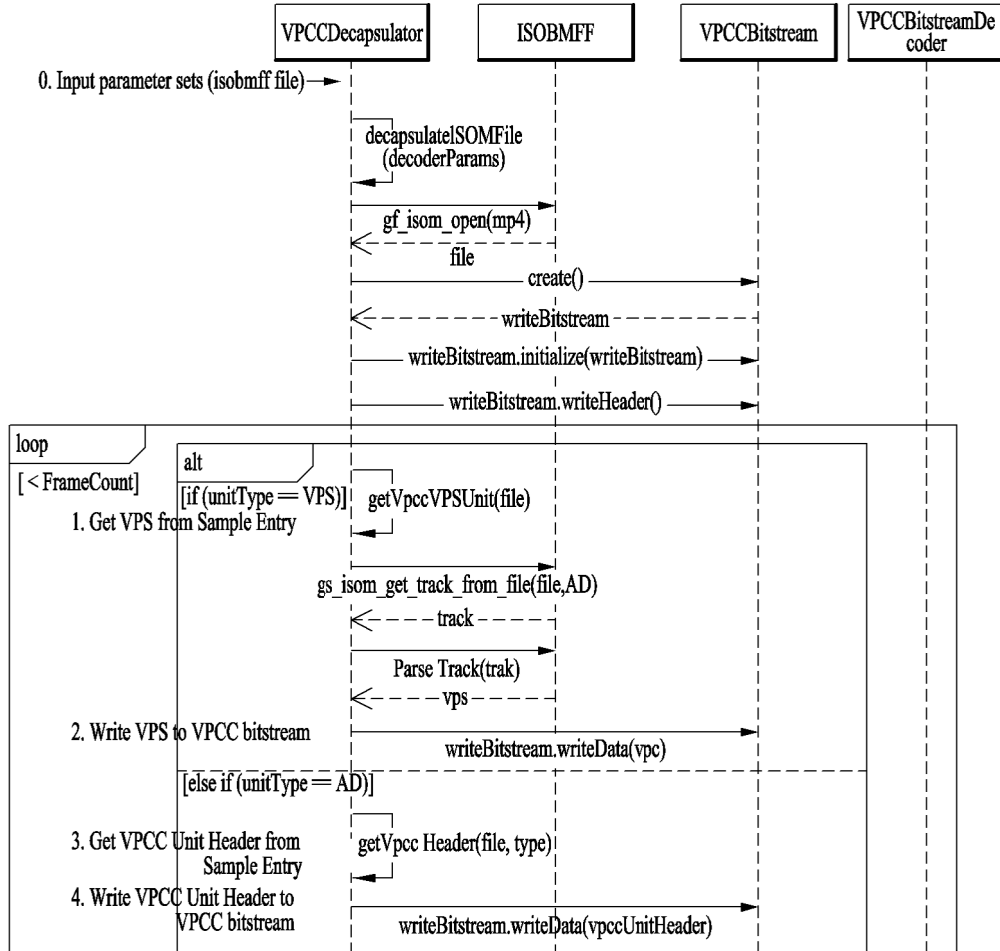


FIG. 67

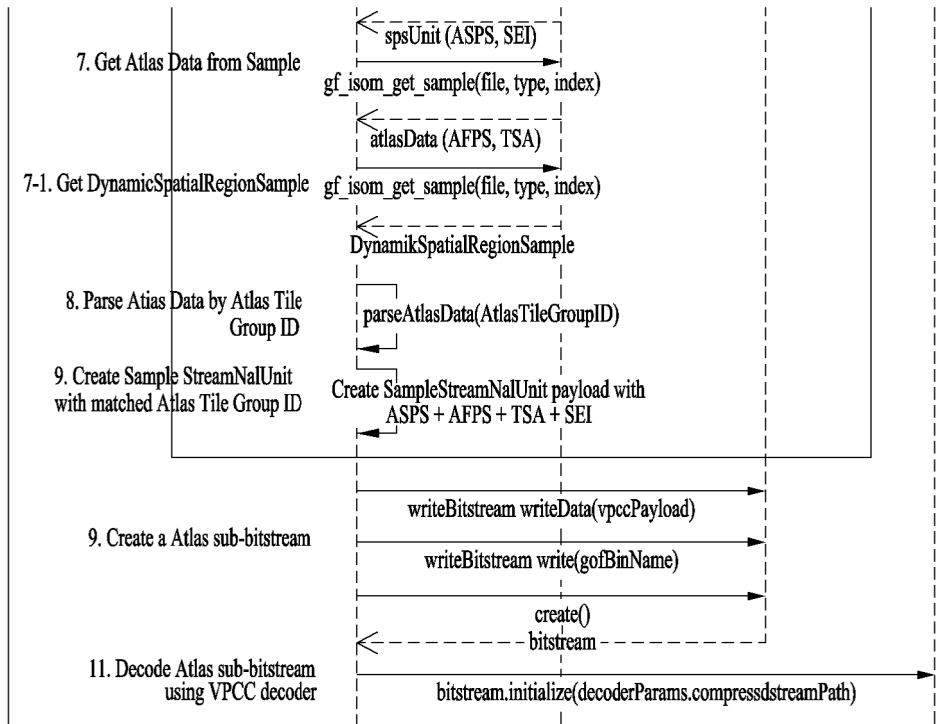


FIG. 68

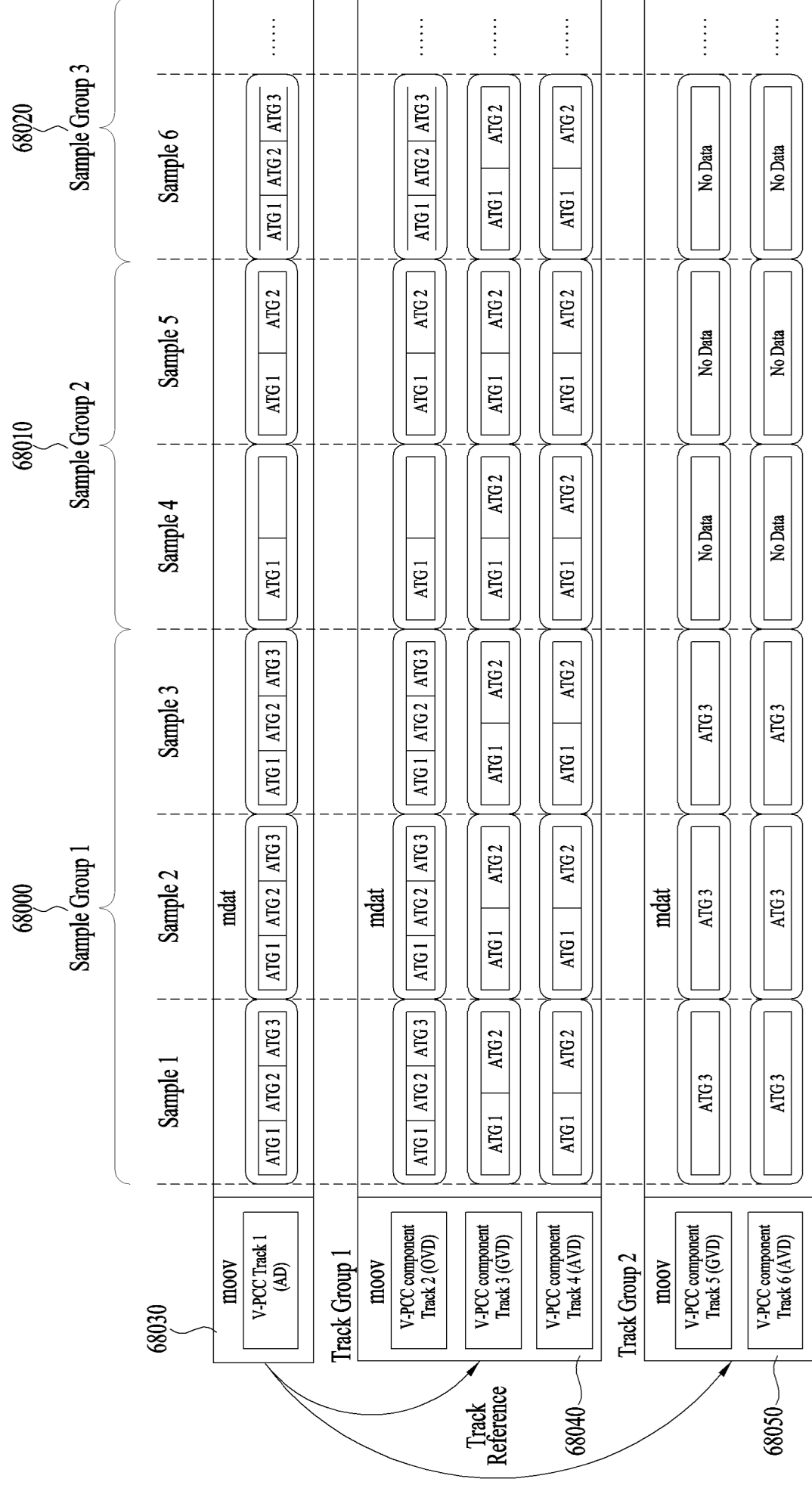


FIG. 69

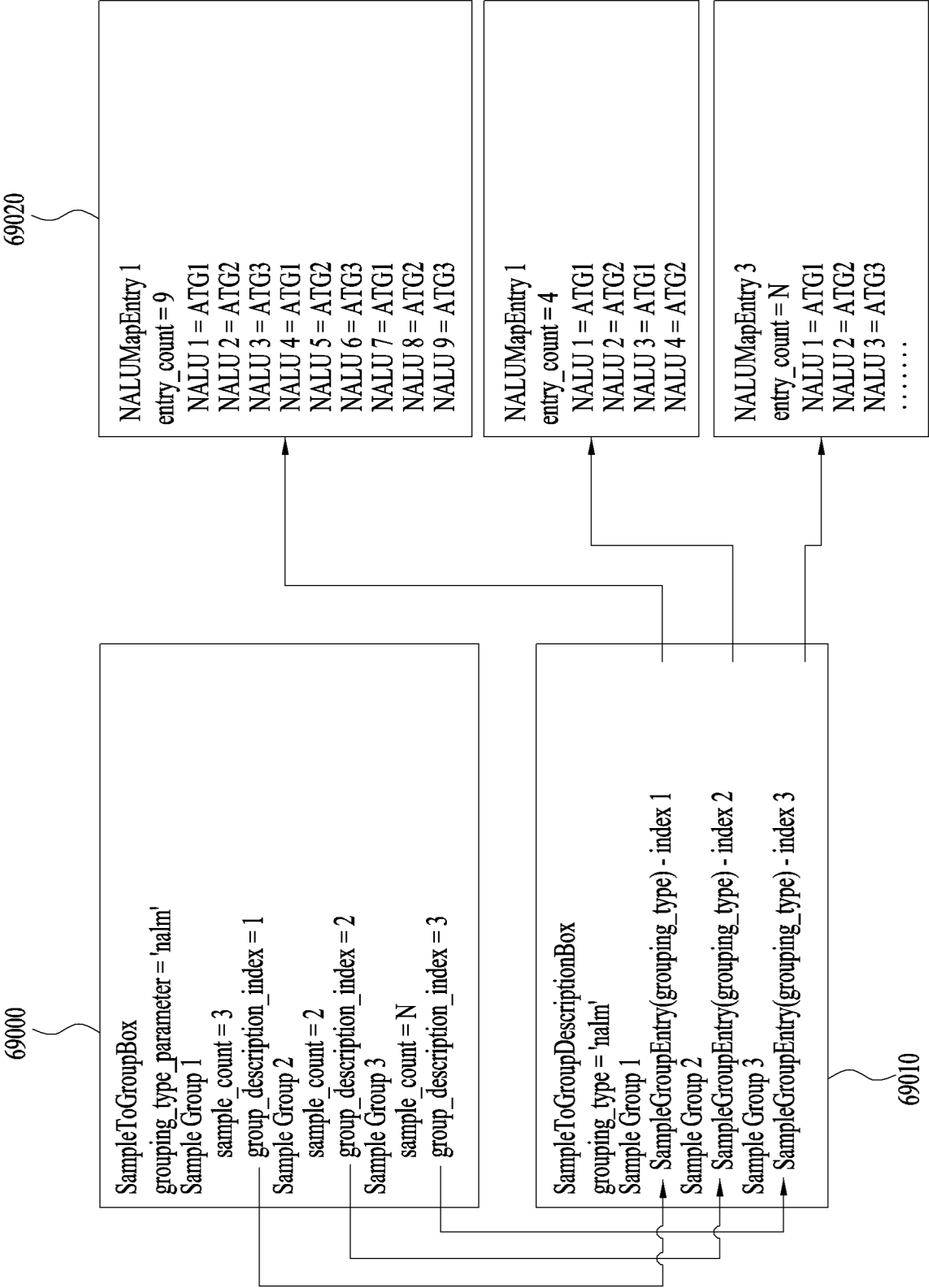


FIG. 70

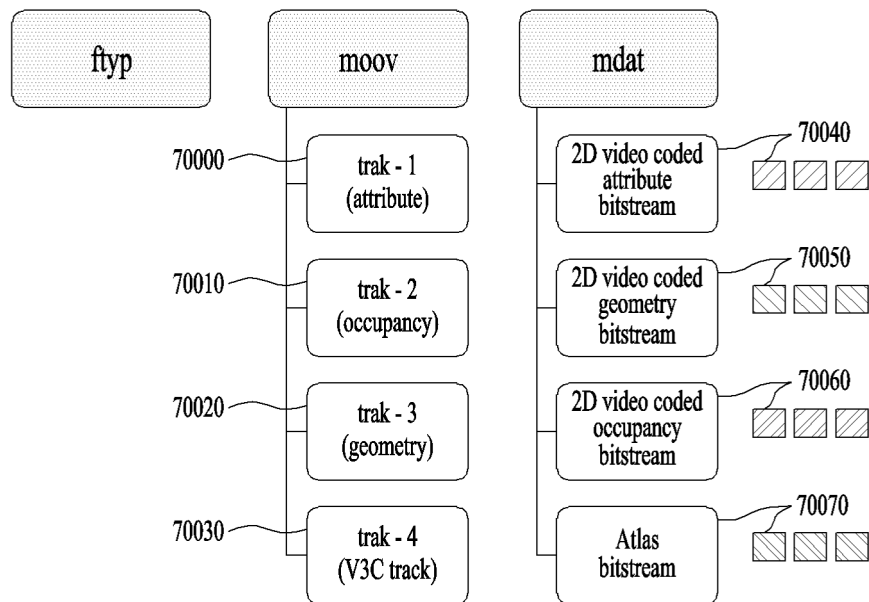


FIG. 71

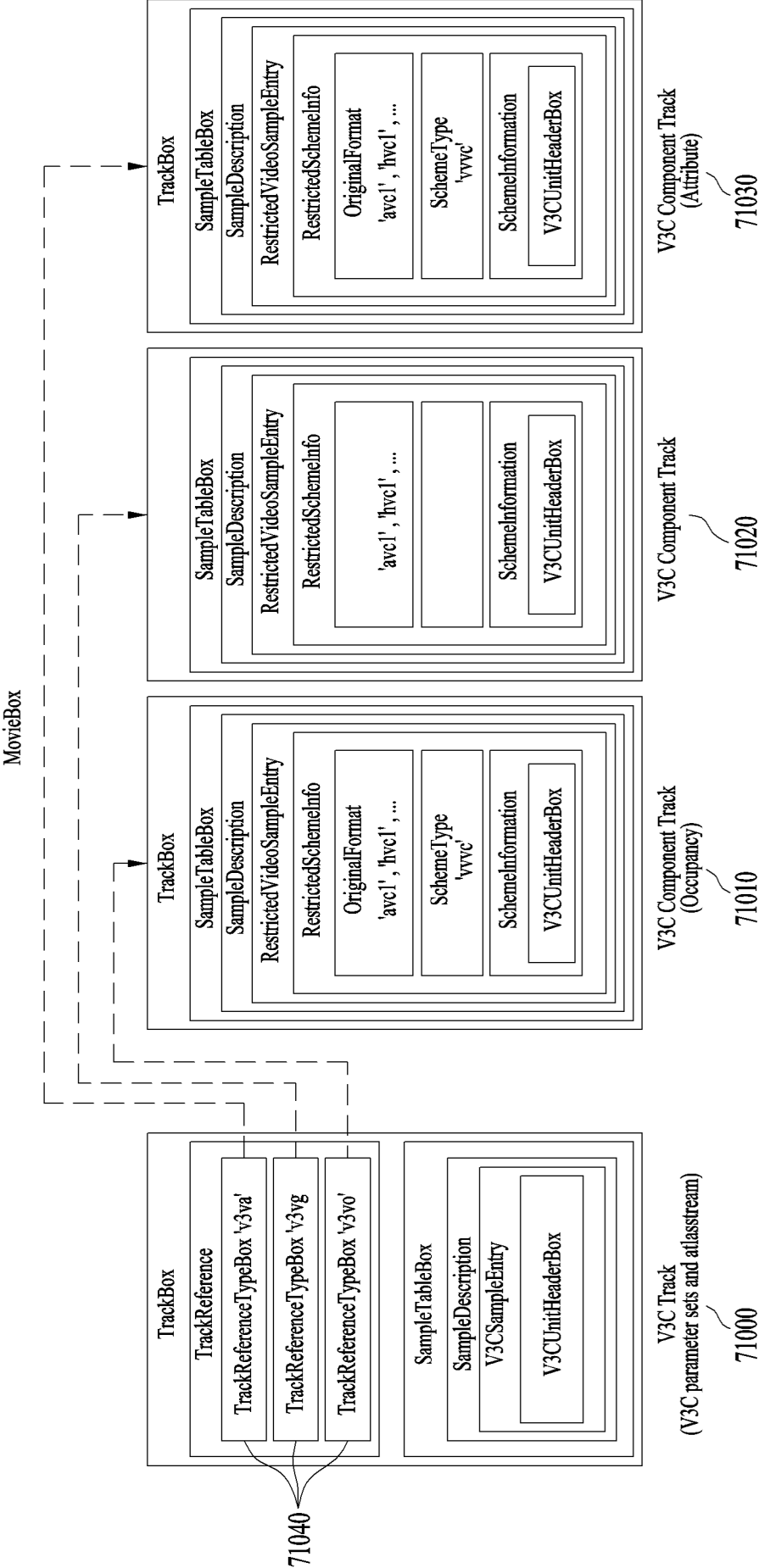


FIG. 72

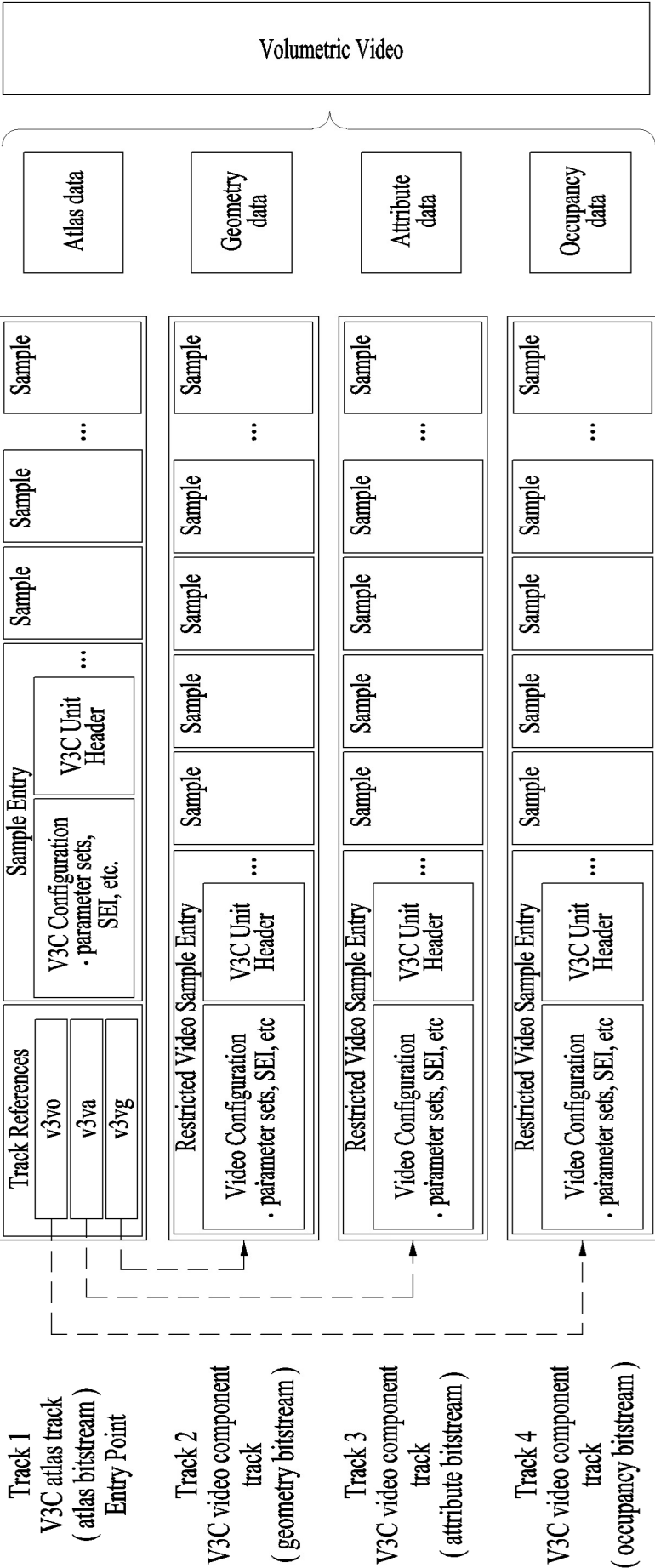


FIG. 73

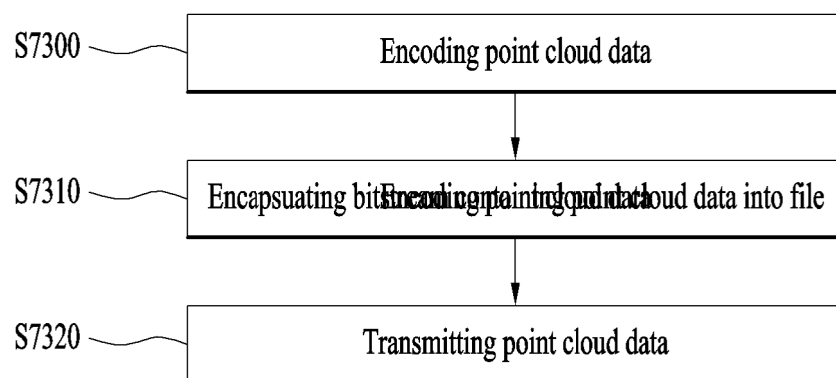


FIG. 74

