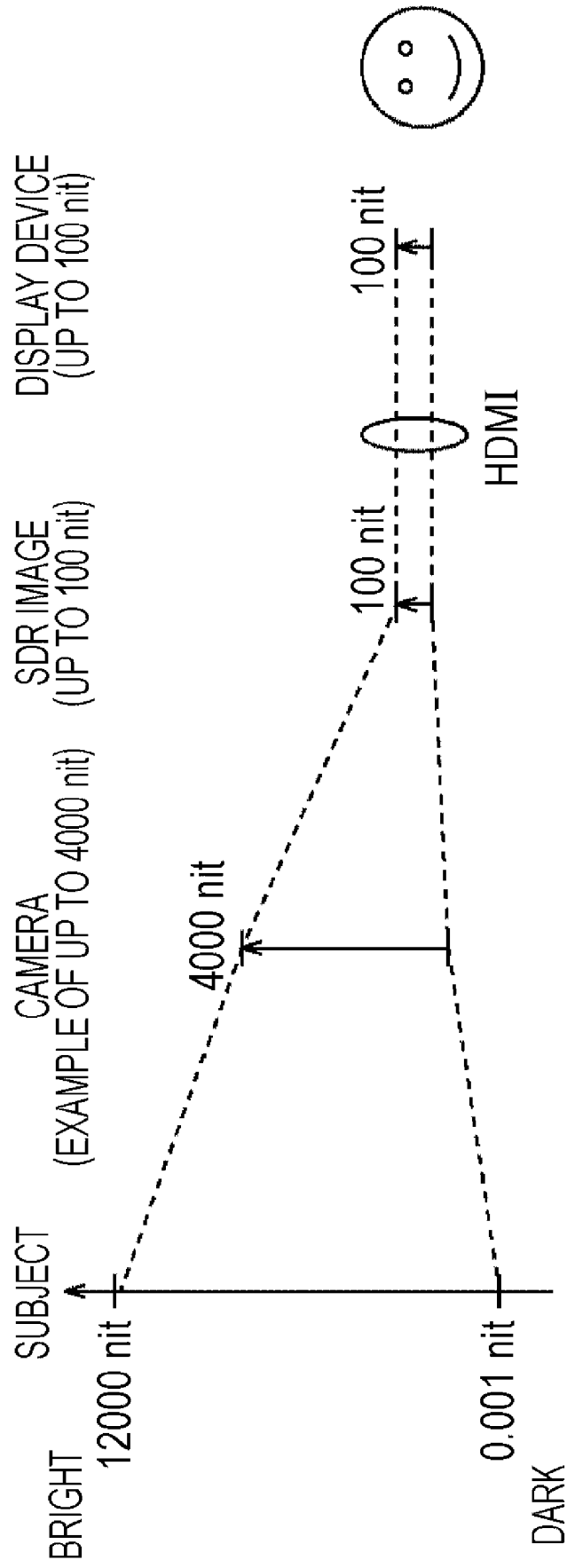


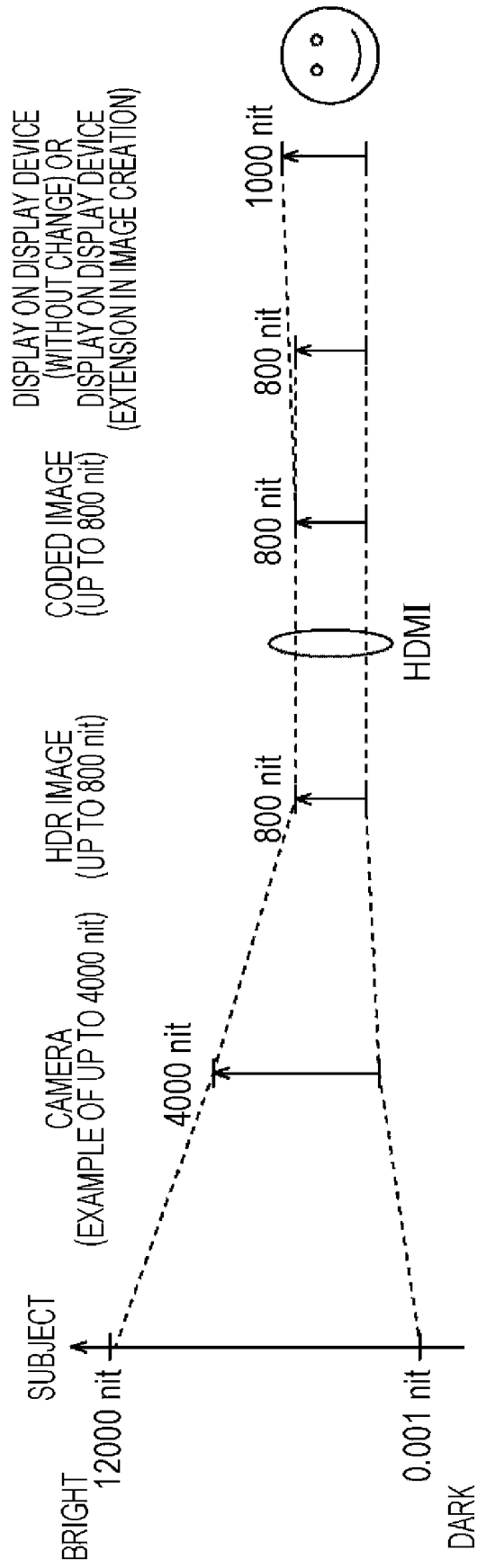
[Fig. 1]

FIG. 1



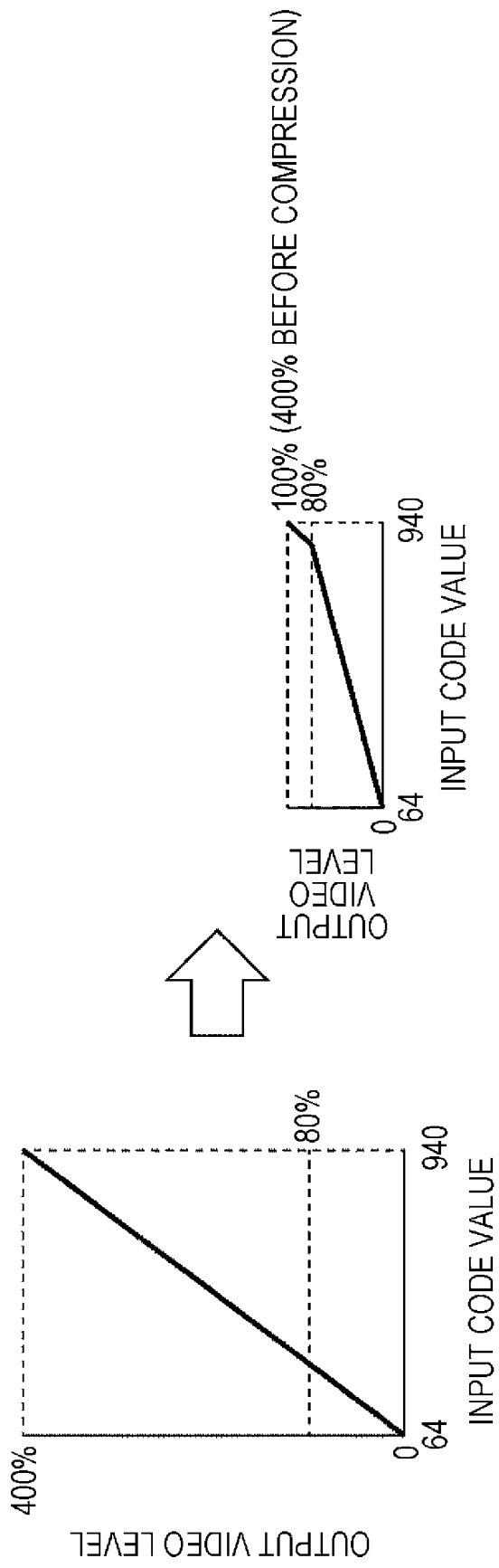
[Fig. 2]

FIG. 2



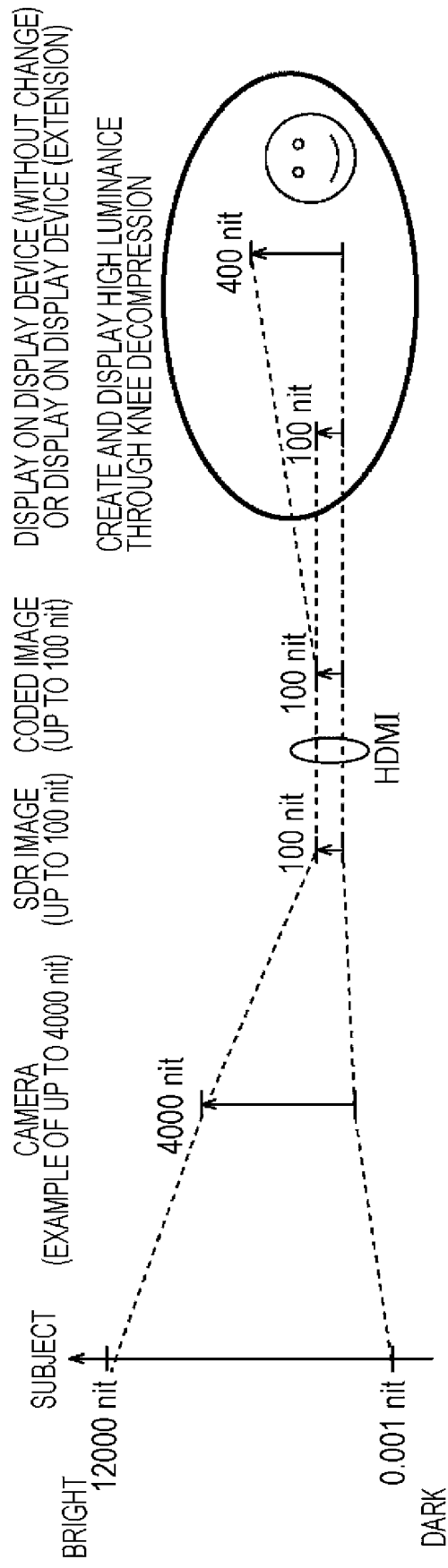
[Fig. 3]

FIG. 3



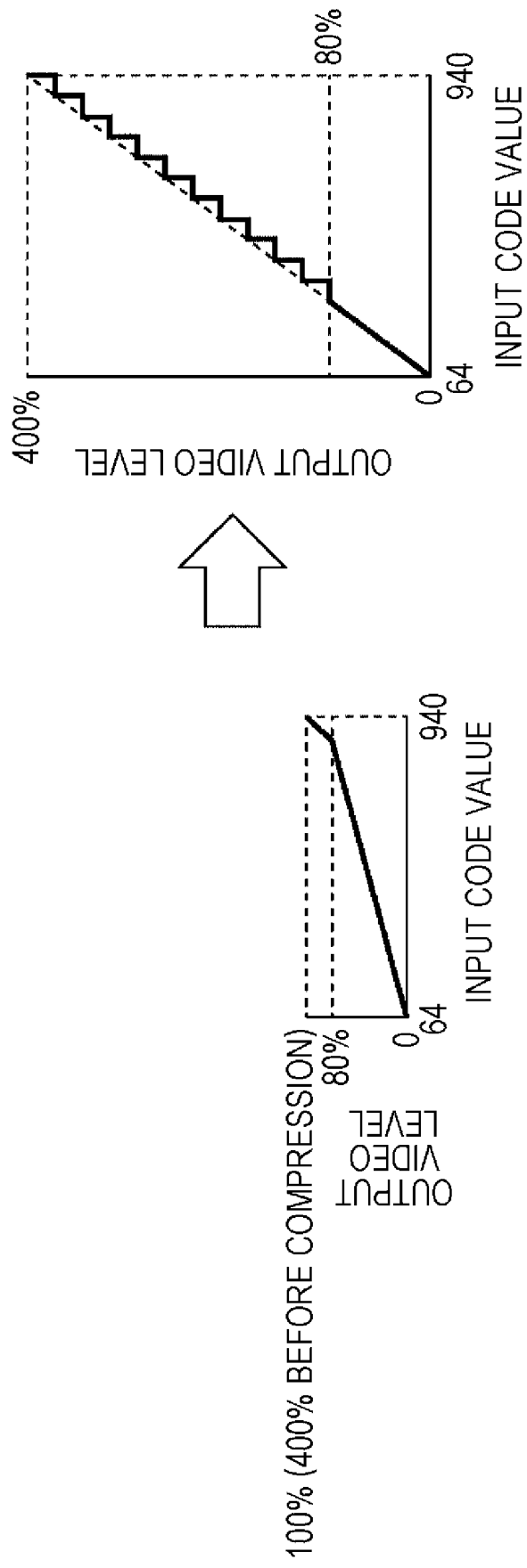
[Fig. 4]

FIG. 4



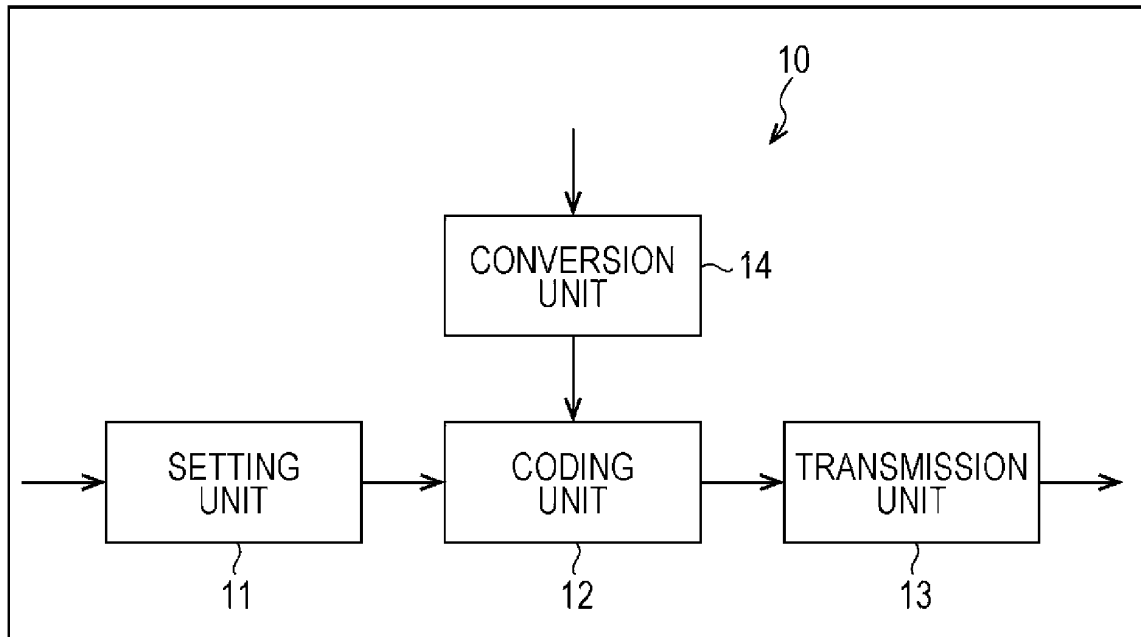
[Fig. 5]

FIG. 5



[Fig. 6]

FIG. 6



[Fig. 7]

FIG. 7

	Descriptor
knee_function_info(payloadSize) {	
knee_point_of_input	u(16)
knee_point_of_output	u(16)
output_white_level_range	u(32)
output_white_level_range_luminance	u(32)
kfi_extension_flag	u(1)
if(kfi_extension_flag)	
while(more_rbsp_data())	
kfi_extension_data_flag	u(1)
}	

[Fig. 8]

FIG. 8

knee_point_of_input indicates the knee point on input-axis. This value increment is in units of one-thousandth of the value that normalized linear input code value to 1.0. Therefore, This value shall be less than or equal to 1000.

knee_point_of_output indicates the knee point on output-axis. This value increment is in units of one-thousandth of the value that normalized linear output code value to 1.0. Therefore, This value shall be less than or equal to 1000.

★when **knee_point_of_input** > **knee_point_of_output**, knee decompression is performed, when **knee_point_of_input** < **knee_point_of_output**, knee compression is performed, and when **knee_point_of_input** = **knee_point_of_output**, no knee conversion is performed. (any process is performed as post process. knee compression is performed when HDR image is converted into SDR image, and knee decompression is performed when SDR image is converted into HDR image)

output_white_level_range indicates the luminance dynamic range for output of the associated pictures, after conversion to the linear light domain for display, expressed as an integer percentage relative to the nominal white level.

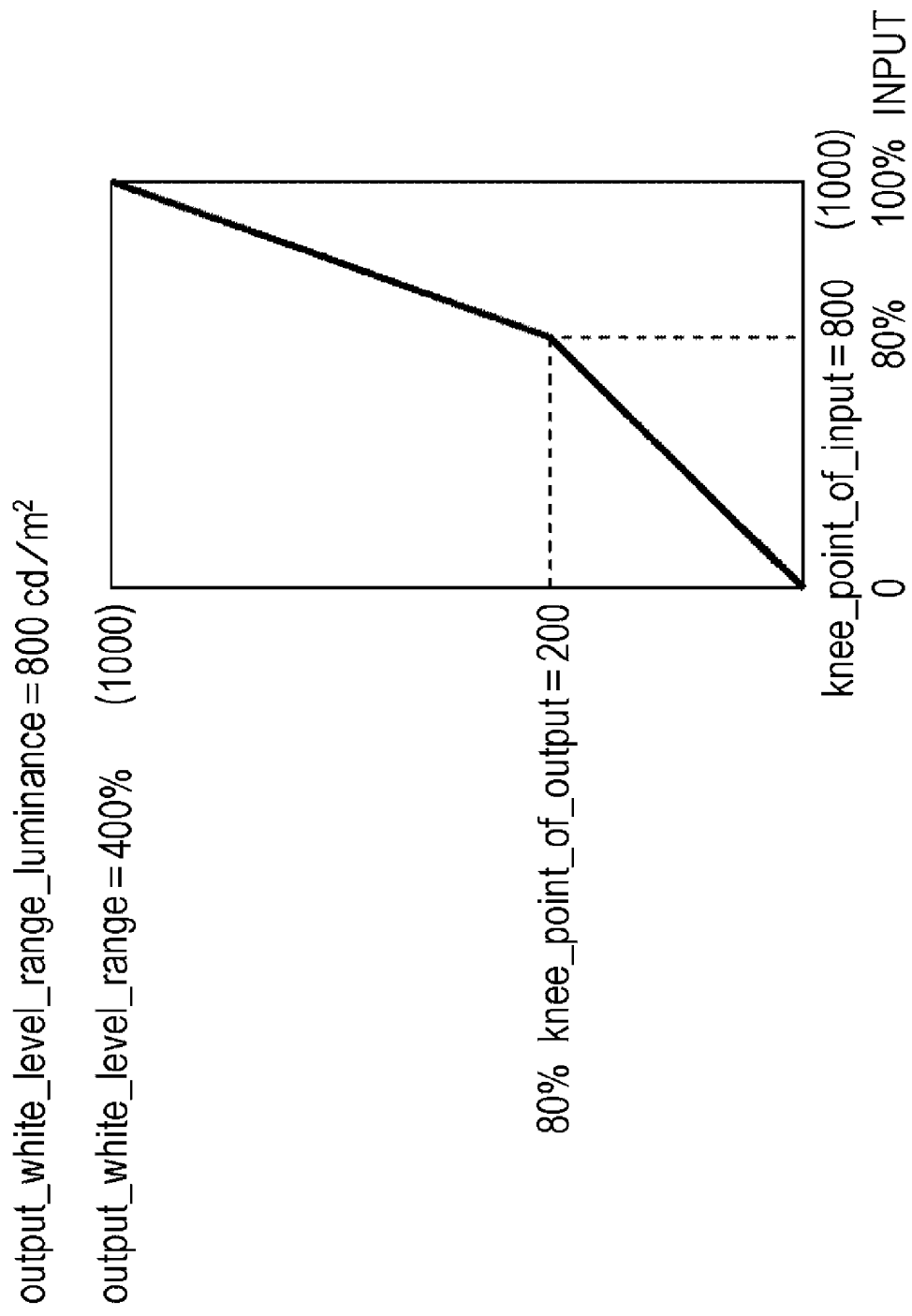
output_white_level_range_luminance indicates the display luminance for the output white level range in units of candela per square metre.

kfi_extension_flag equal to 0 specifies that no **kfi_extension_data_flag** syntax elements are present in the KFI RBSP syntax structure. **kfi_extension_flag** shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for **kfi_extension_flag** is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all **kfi_extension_data_flag** syntax elements that follow the value 1 for **kfi_extension_flag** in an KFI NAL unit.

kfi_extension_data_flag may have any value. Its presence and value do not affect decoder conformance to profiles specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all **kfi_extension_data_flag** syntax elements.

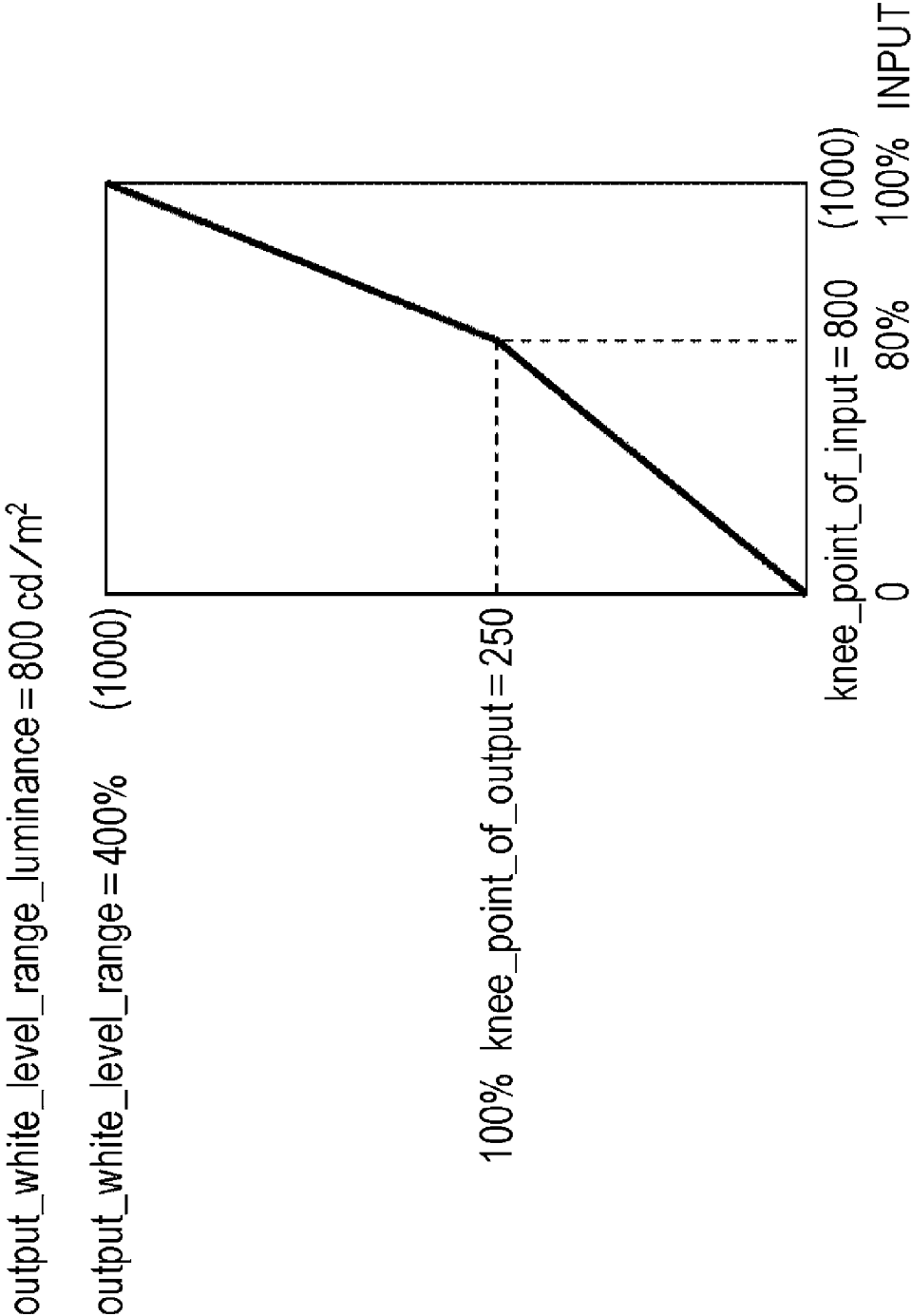
[Fig. 9]

FIG. 9



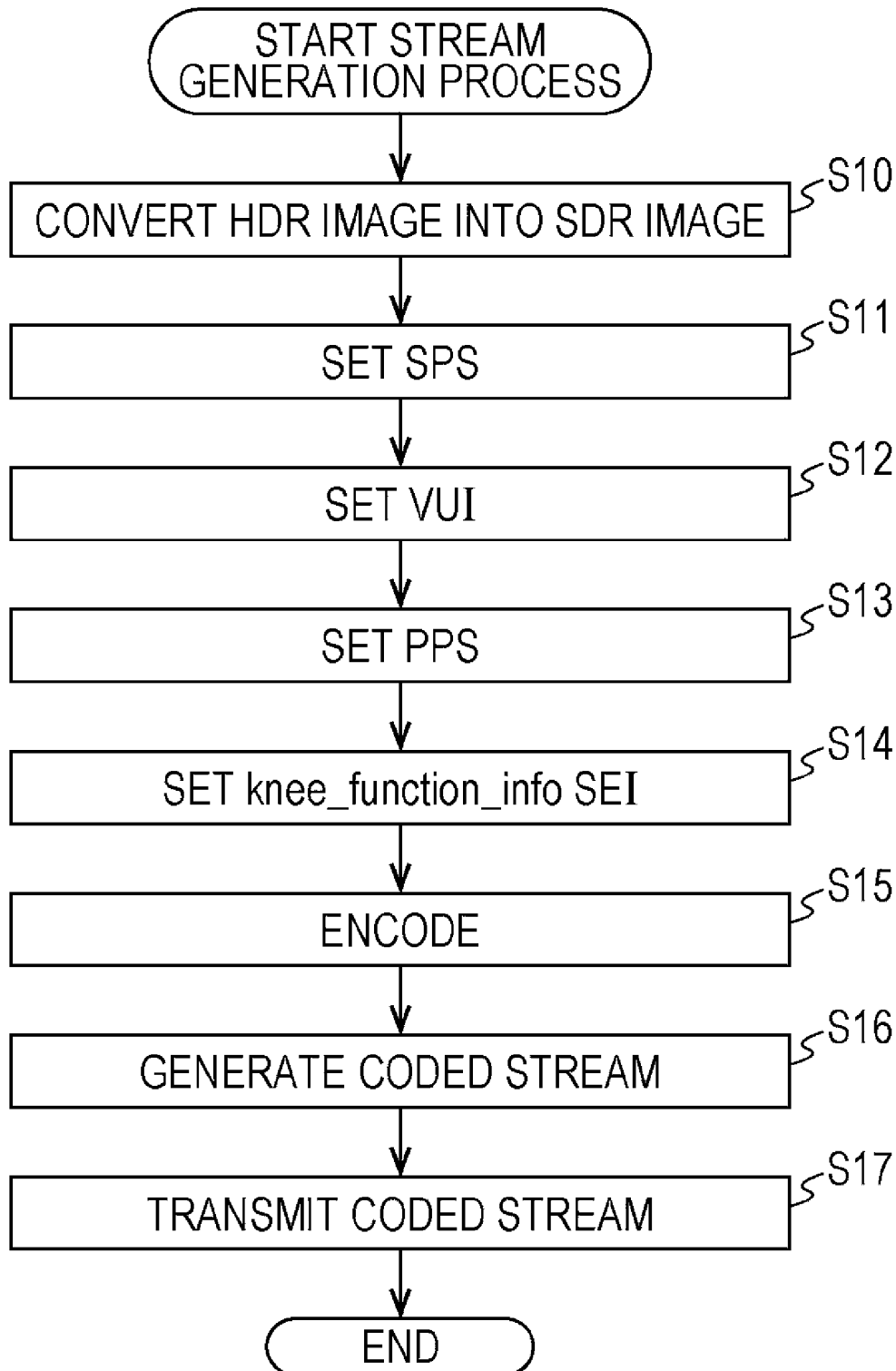
[Fig. 10]

FIG. 10

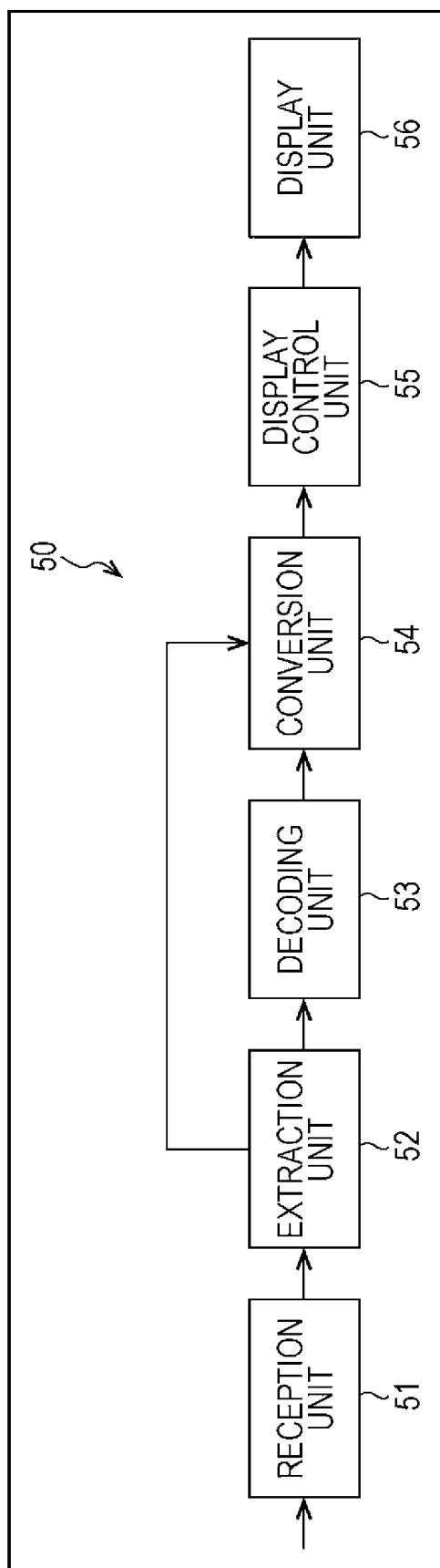


[Fig. 11]

FIG. 11

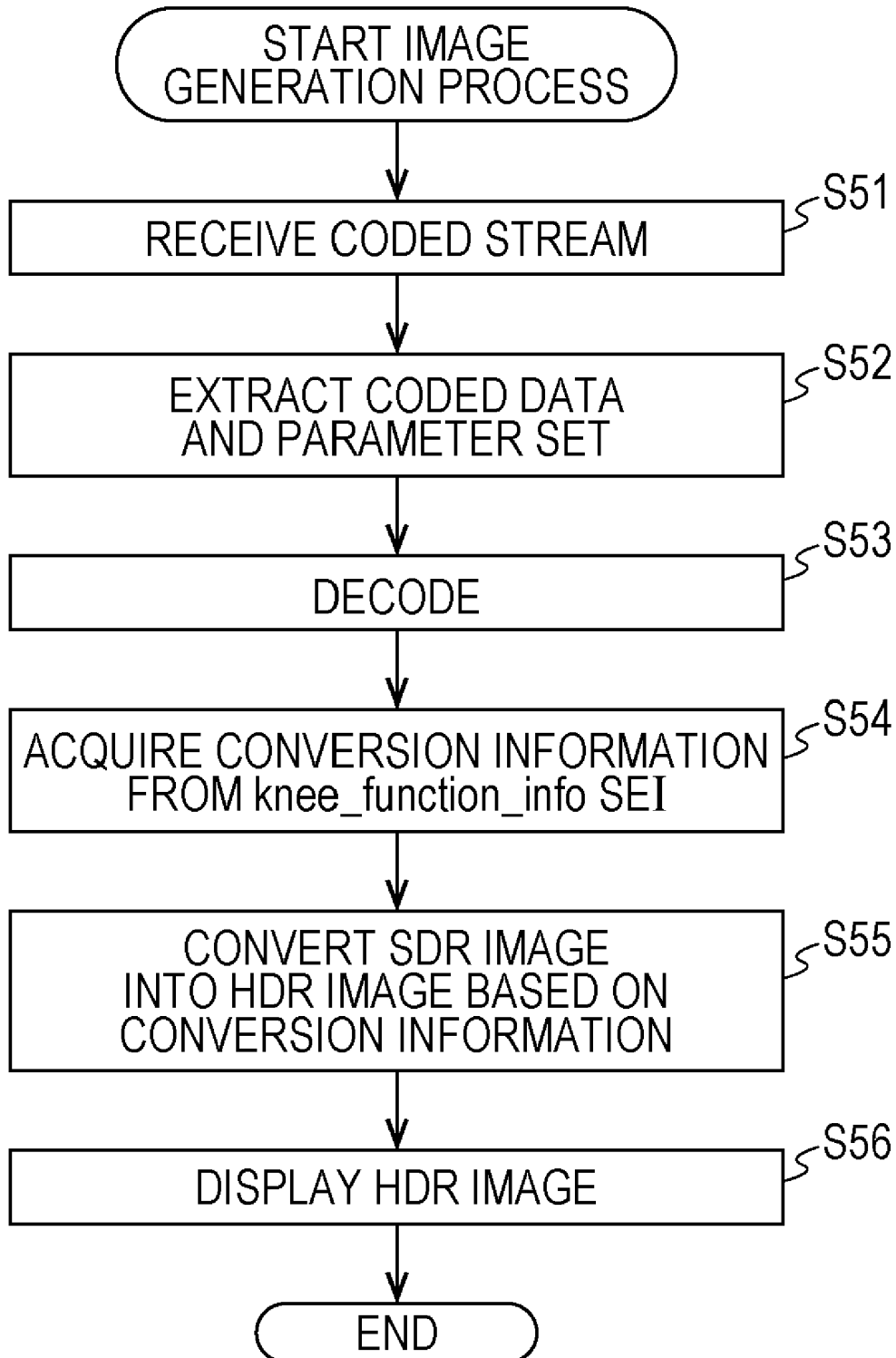


[Fig. 12]



[Fig. 13]

FIG. 13



[Fig. 14]

FIG. 14

	Descriptor
knee_function_info(payloadSize) {	
knee_point_of_input	u(16)
knee_point_of_output	u(16)
white_level_range	u(32)
white_level_range_luminance	u(32)
kfi_extension_flag	u(1)
if(kfi_extension_flag)	
while(more_rbsp_data())	
kfi_extension_data_flag	u(1)
}	

[Fig. 15]

FIG. 15

knee_point_of_input indicates the knee point on input-axis. This value increment is in units of one-thousandth of the value that normalized linear input code value to 1.0. Therefore, This value shall be less than or equal to 1000.

knee_point_of_output indicates the knee point on output-axis. This value increment is in units of one-thousandth of the value that normalized linear output code value to 1.0. Therefore, This value shall be less than or equal to 1000.

★when **knee_point_of_input** > **knee_point_of_output**, knee decompression is performed, when **knee_point_of_input** < **knee_point_of_output**, knee compression is performed, and when **knee_point_of_input** = **knee_point_of_output**, no knee conversion is performed. (any process is performed as post process. knee compression is performed when HDR image is converted into SDR image, and knee decompression is performed when SDR image is converted into HDR image)

white_level_range indicates the luminance dynamic range for output of the associated pictures, after conversion to the linear light domain for display, expressed as an integer percentage relative to the nominal white level.

white_level_range_luminance indicates the display luminance for the output white level range in units of candela per square metre.

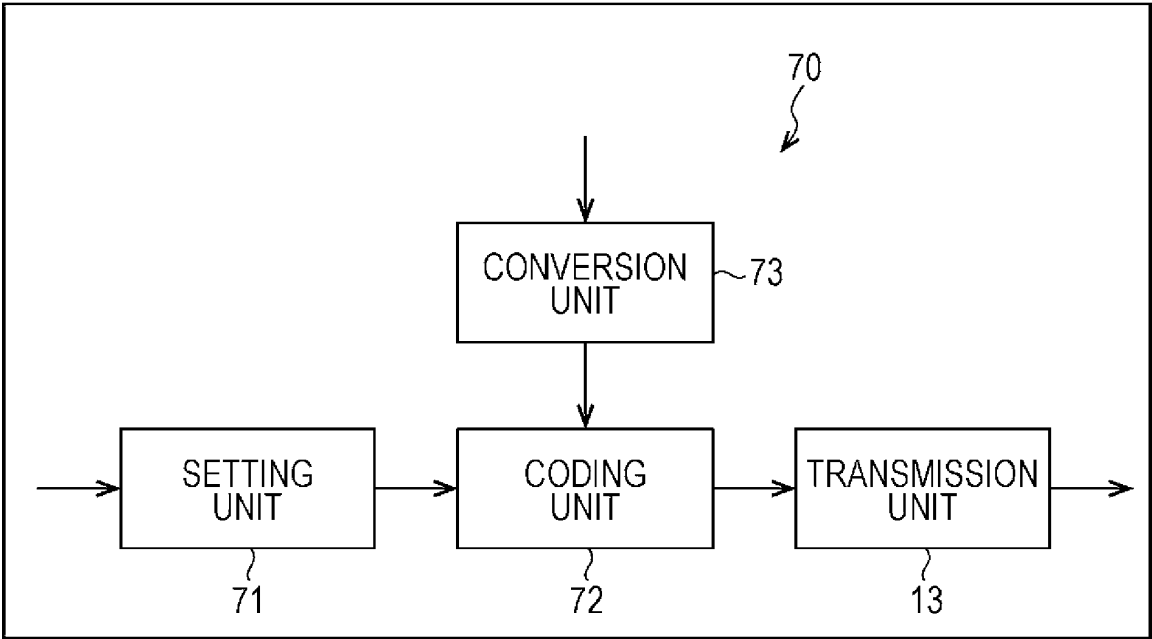
★when **knee_point_of_input** >= **knee_point_of_output**, **white_level_range** == **output_white_level_range**, when **knee_point_of_input** < **knee_point_of_output**, **white_level_range** == **input_white_level_range**

kfi_extension_flag equal to 0 specifies that no **kfi_extension_data_flag** syntax elements are present in the KFI RBSP syntax structure. **kfi_extension_flag** shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for **kfi_extension_flag** is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all **kfi_extension_data_flag** syntax elements that follow the value 1 for **kfi_extension_flag** in an KFI NAL unit.

kfi_extension_data_flag may have any value. Its presence and value do not affect decoder conformance to profiles specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all **kfi_extension_data_flag** syntax elements.

[Fig. 16]

FIG. 16



[Fig. 17]

FIG. 17

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
input_knee_point	u(16)
output_knee_point	u(16)
d_range	u(32)
d_range_disp_luminance	u(32)
}	
}	

[Fig. 18]

FIG. 18

knee_function_id contains an identifying number that may be used to identify the purpose of the knee functions. The value of **knee_function_id** shall be in the range of 0 to 232 - 2, inclusive.

- Values of **knee_function_id** from 0 to 255 and from 512 to 231 - 1 may be used as determined by the application. Values of **knee_function_id** from 256 to 511, inclusive, and from 231 to 232 - 2, inclusive are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all knee function information SEI messages containing a value of **knee_function_id** in the range of 256 to 511, inclusive, or in the range of 231 to 232 - 2, inclusive, and bitstreams shall not contain such values.
- NOTE 1 - The **knee_function_id** can be used to support knee function process that are suitable for different display scenarios. For example, different values of **knee_function_id** may correspond to different display bit depths.

knee_function_cancel_flag equal to 1 indicates that the knee function information SEI message cancels the persistence of any previous knee function information SEI message in output order. **knee_function_cancel_flag** equal to 0 indicates that knee function information follows.

input_knee_point specifies the luminance level of a point at which the decompression or compression starts to flatten the contrast curve for the input picture. The luminance level of the knee point for input picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **input_knee_point** is expressed in range of 0 to 1000.

output_knee_point specifies the luminance level of a point at which the decompression or compression starts to flatten the contrast curve for the output picture. The luminance level of the knee point for output picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **output_knee_point** is expressed in range of 0 to 1000.

- NOTE 2 - The coordinate of the knee point is defined as (**input_knee_point**, **output_knee_point**).

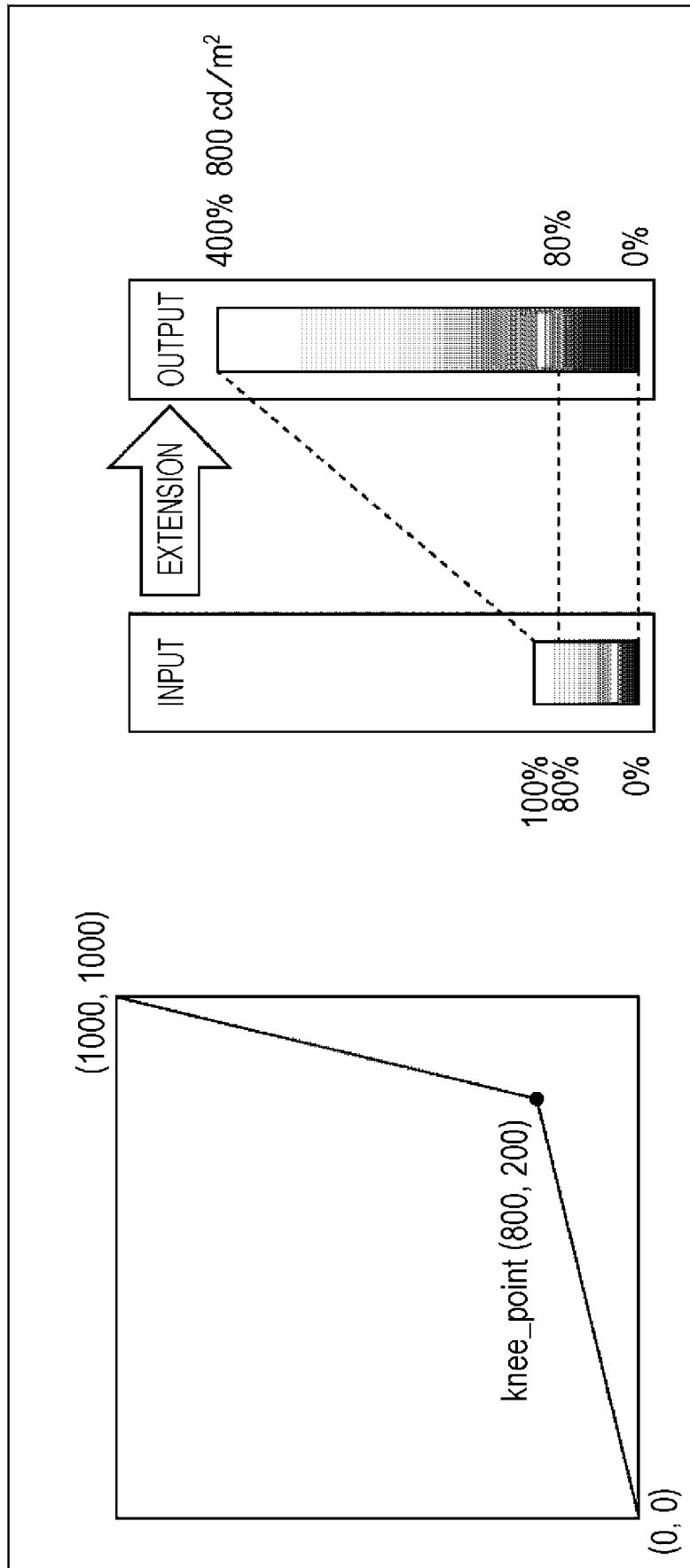
d_range specifies the peak luminance level of either the decompressed picture or uncompressed picture in relative to the nominal luminance level in units of 0.1%.

- The value of **d_range** specifies the peak luminance level of the decompressed picture (output picture of the knee process) when the value of **input_knee_point** is greater than or equal to the value of **output_knee_point**. The value of **d_range** specifies the peak luminance level of the uncompressed picture (input picture of the knee process) when the value of **input_knee_point** is less than the value of **output_knee_point**.

d_range_disp_luminance specifies the expected display brightness of peak luminance level specified as **d_range**. The value of **d_range_disp_luminance** is in units of candela per square metre.

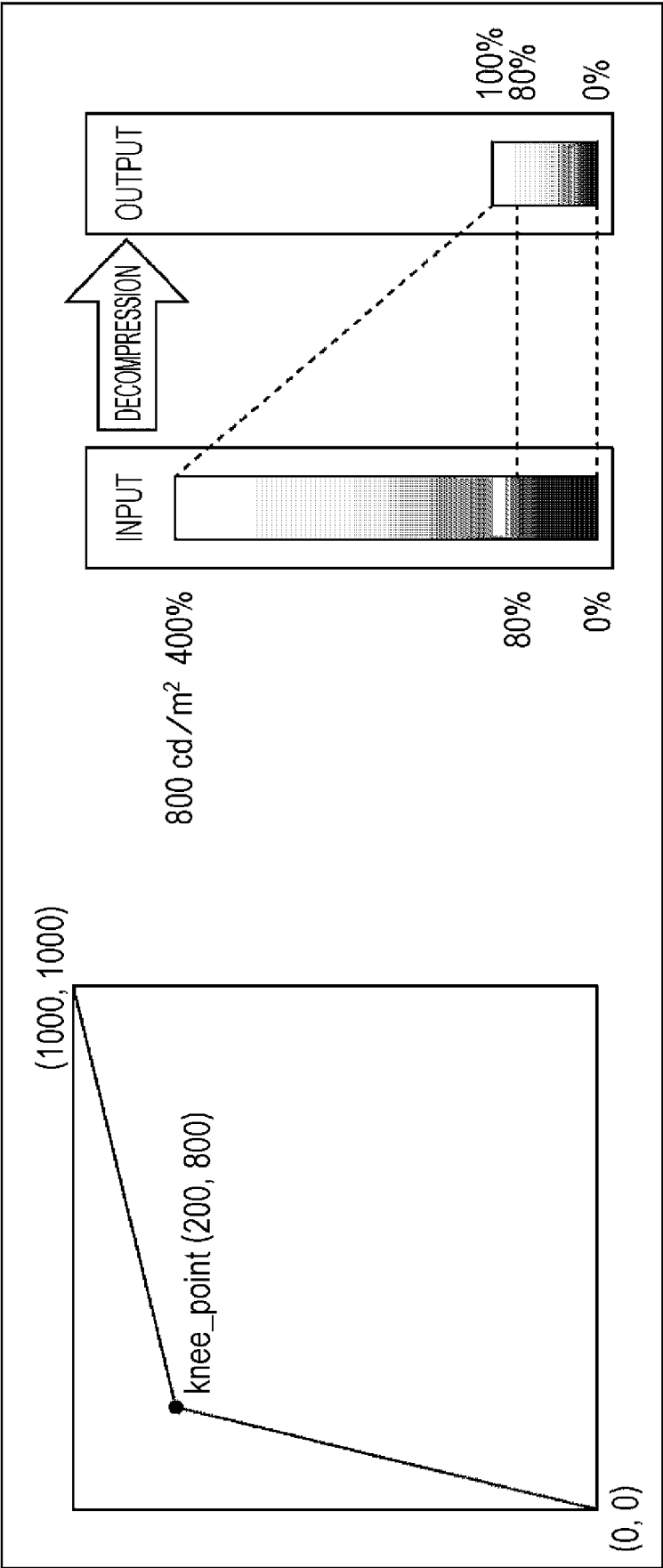
[Fig. 19]

FIG. 19



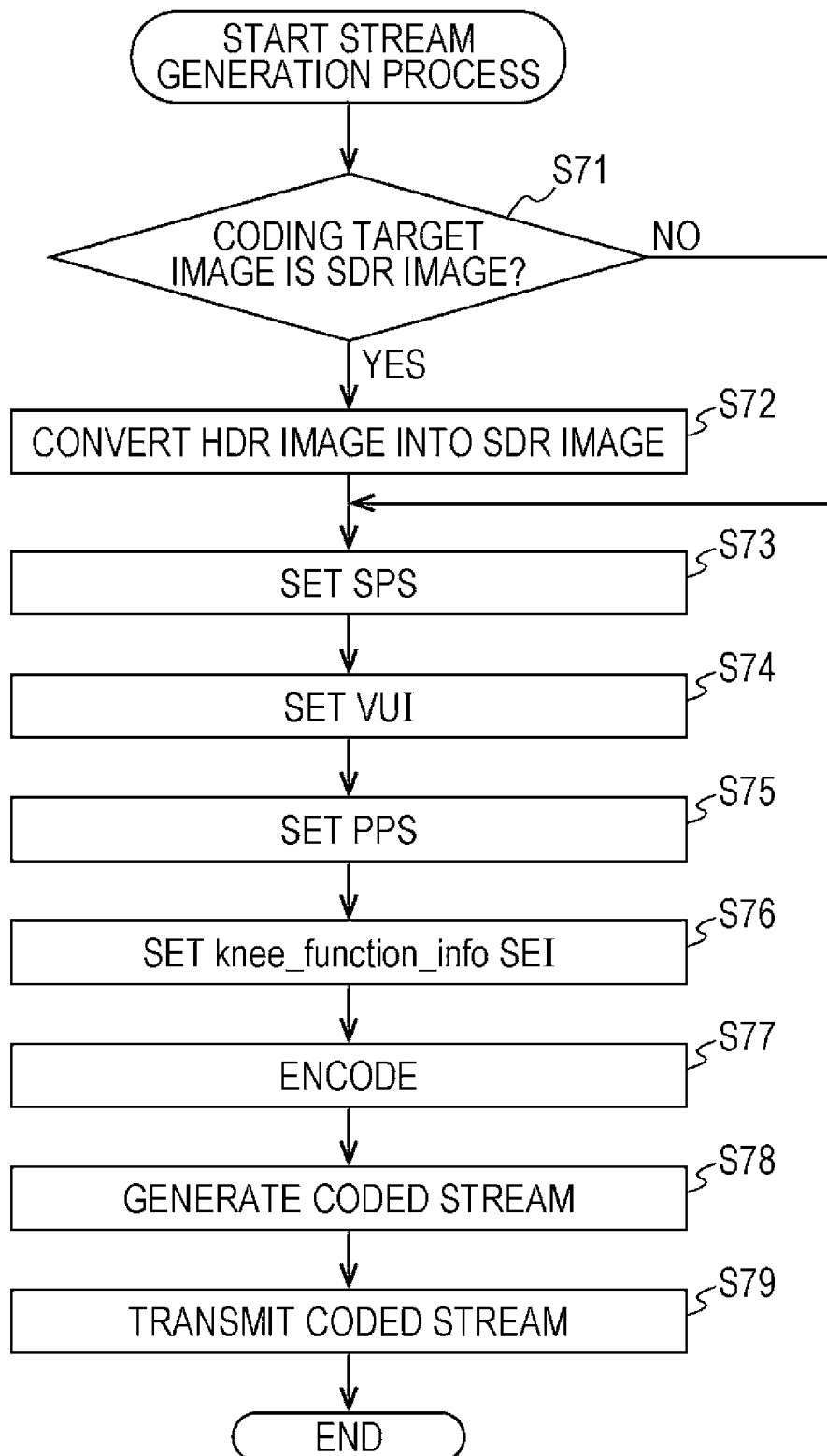
[Fig. 20]

FIG. 20



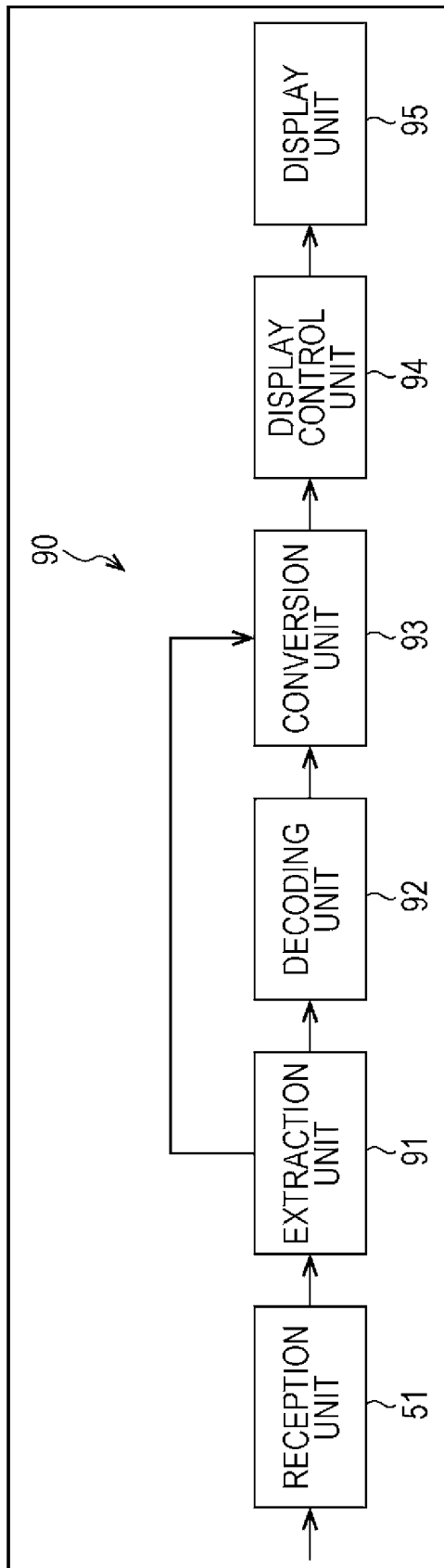
[Fig. 21]

FIG. 21



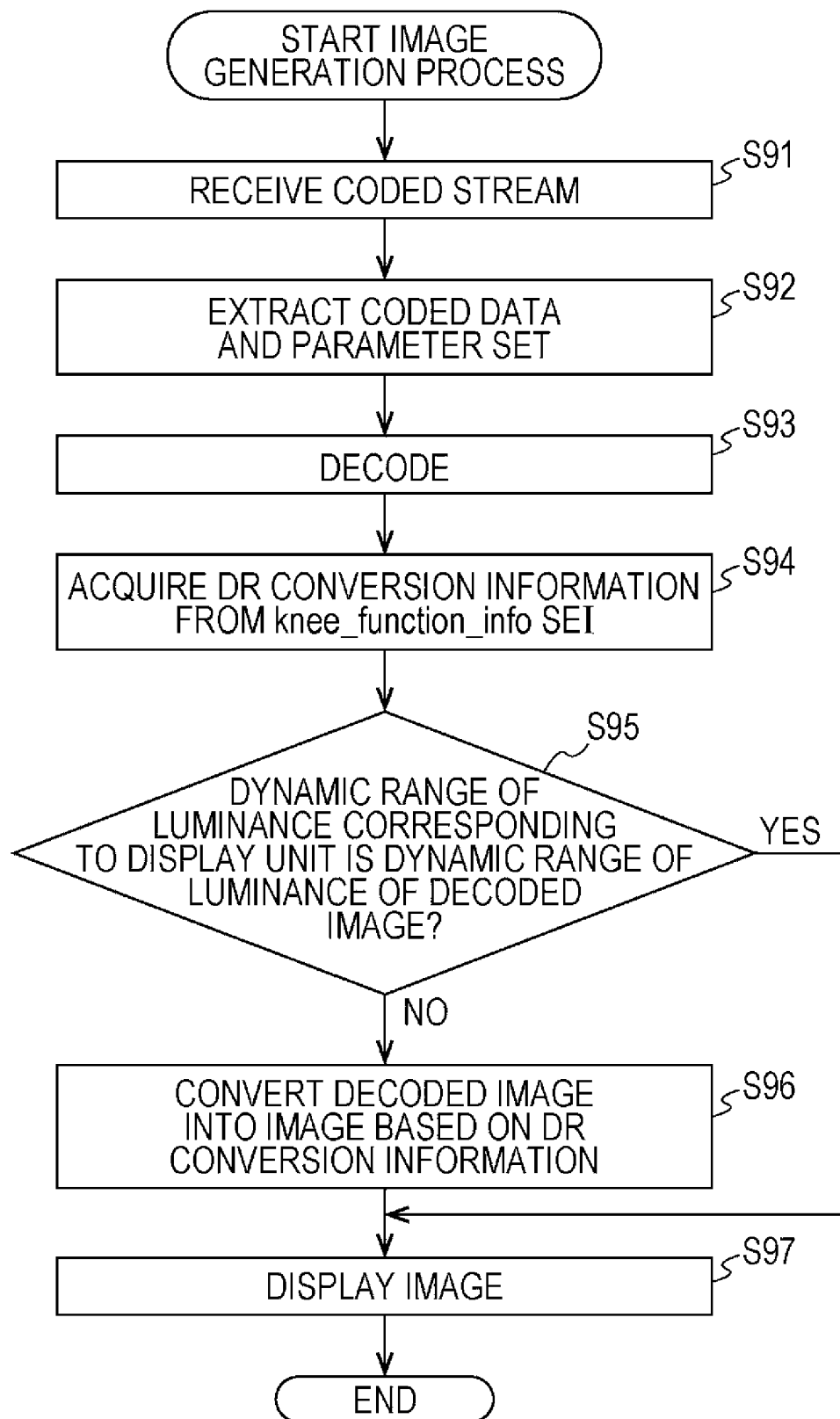
[Fig. 22]

FIG. 22



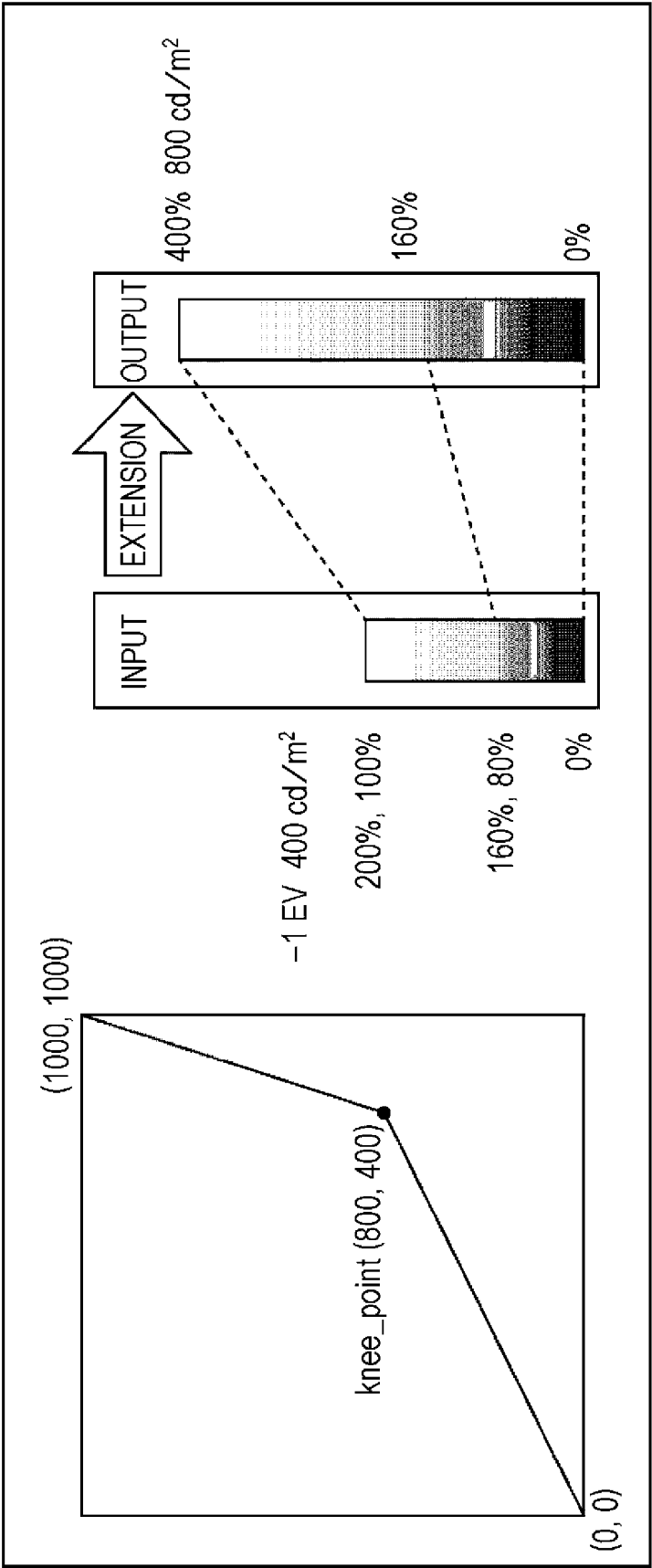
[Fig. 23]

FIG. 23



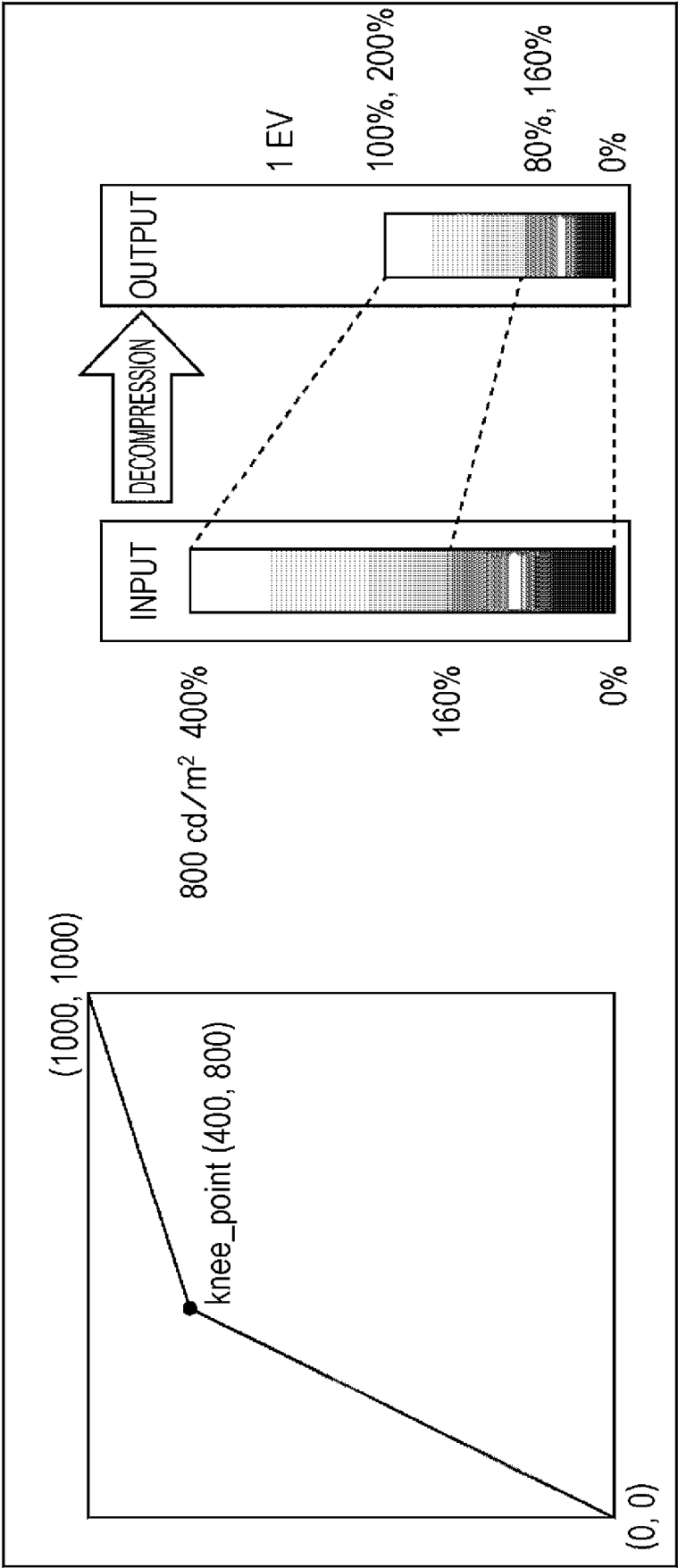
[Fig. 24]

FIG. 24



[Fig. 25]

FIG. 25



[Fig. 26]

FIG. 26

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}else if(tone_map_model_id == 5)	
{	
input_knee_point	u(16)
output_knee_point	u(16)
d_range	u(32)
d_range_disp_luminance	u(32)
}	
}	
}	

[Fig. 27]

FIG. 27

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}else if(tone_map_model_id == 5)	
{	
input_knee_point	u(16)
output_knee_point	u(16)
}	
}	
}	

[Fig. 28]

FIG. 28

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
compression_flag	u(1)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	ue(v)
for(i=0;i<=num_knee_point_minus1;i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	

[Fig. 29]

FIG. 29

compression_flag equals to 0 indicates that the knee function is decompression function, **compression_flag** equals to 1 indicates that the knee function is compression function.

num_knee_point_minus1 plus 1 specifies the number of knee points without counting the two default end points, (0,0) and (1,1).

input_knee_point[i] specifies the luminance level of i-th knee point for input picture. The luminance level of the knee point for input picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **input_knee_point** is expressed in range of 0 to 1000. **input_knee_point[i]** shall be greater than **input_knee_point[i-1]**.

NOTE 2—The coordinate of the representative knee point is defined as (**input_knee_point[i]**, **output_knee_point[i]**).

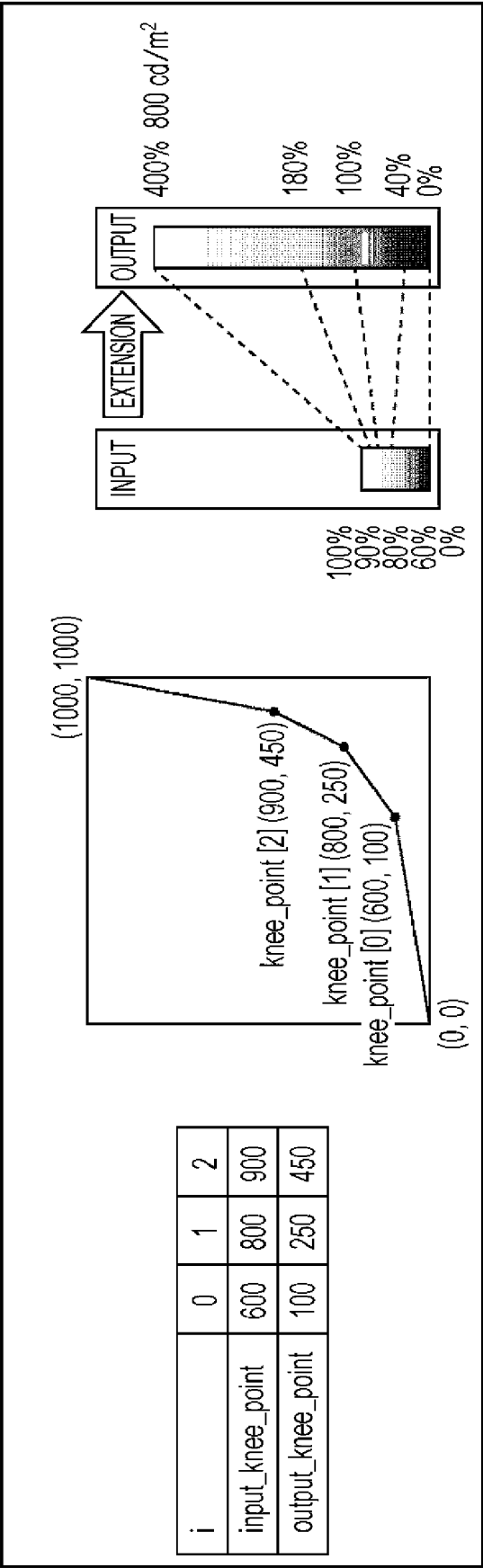
output_knee_point[i] specifies the luminance level of i-th knee for output picture. The luminance level of the knee point for output picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **output_knee_point** is expressed in range of 0 to 1000.

NOTE 3 - The coordinate of the i-th knee point is defined as (**input_knee_point[i]**, **output_knee_point[i]**).

Knee function is composed of several lines which starts with the point (0,0) ends with the point (1,1) through all knee points in ascending order of i.

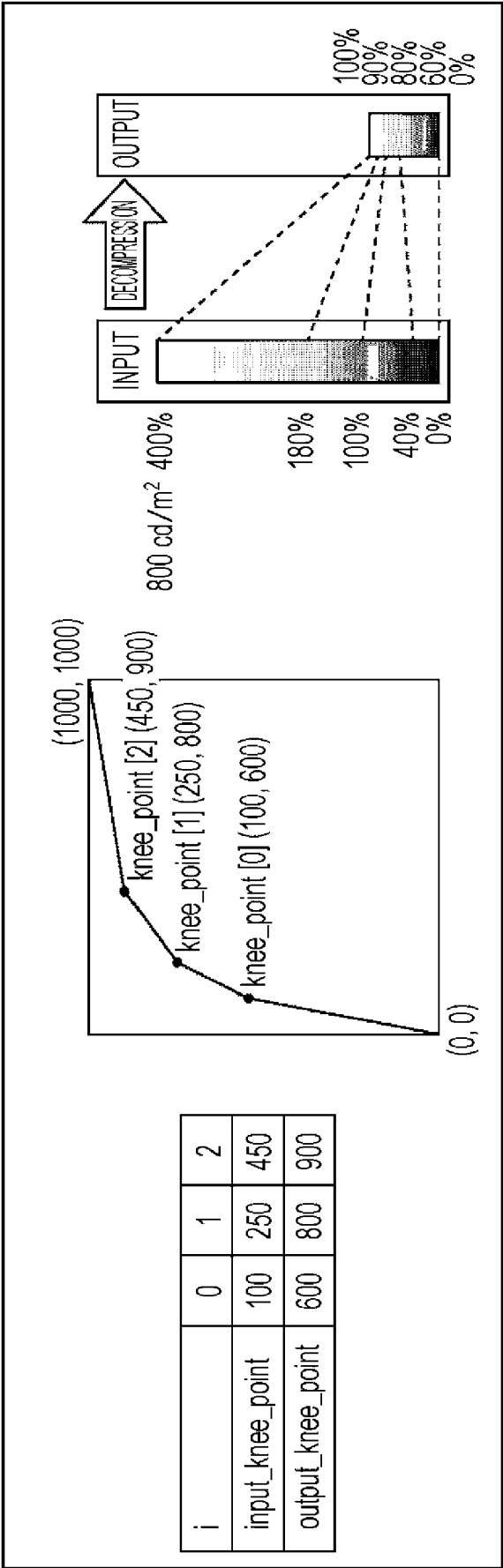
[Fig. 30]

FIG. 30



[Fig. 31]

FIG. 31



[Fig. 32]

FIG. 32

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
} else if(tone_map_model_id == 5) {	
compression_flag	u(1)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	ue(v)
for(i=0;i<=num_knee_point_minus1;i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	
}	

[Fig. 33]

FIG. 33

num_knee_point_minus1 plus 1 specifies the number of knee points without counting the two default end points, (0, 0) and (1, 1). The value of num_knee_point_minus1 shall be 0, 1 or 2.

[Fig. 34]

FIG. 34

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
compression_flag	u(1)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	u(2)
for(i=0; i<=num_knee_point_minus1; i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	

[Fig. 35]

FIG. 35

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}else if(tone_map_model_id == 5) {	
compression_flag	u(1)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	u(2)
for (i=0; i<=num_knee_point_minus1; i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	
}	

[Fig. 36]

FIG. 36

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
compression_flag	u(1)
representative_input_knee_point	u(16)
representative_output_knee_point	u(16)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	u(2)
for(i=0; i<=num_knee_point_minus1; i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	

FIG. 37

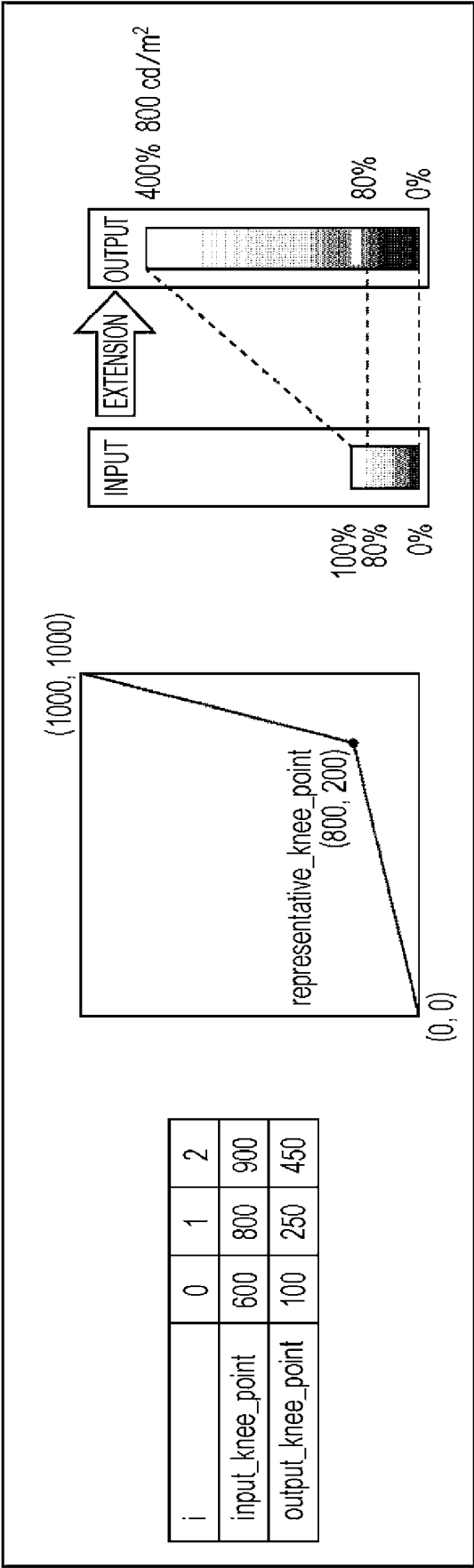
representative_input_knee_point specifies the luminance level of a point at which the decompression or compression starts to flatten the contrast curve for the input picture. The luminance level of the knee point for input picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **representative_input_knee_point** is expressed in range of 0 to 1000.

representative_output_knee_point specifies the luminance level of a point at which the decompression or compression starts to flatten the contrast curve for the output picture. The luminance level of the knee point for output picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **representative_output_knee_point** is expressed in range of 0 to 1000.

NOTE 2 – The coordinate of the representative knee point is defined as (**representative_input_knee_point**, **representative_output_knee_point**).

[Fig. 38]

FIG. 38



[Fig. 39]

FIG. 39

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}else if(tone_map_model_id == 5) {	
compression_flag	u(1)
representative_input_knee_point	u(16)
representative_output_knee_point	u(16)
d_range	u(32)
d_range_disp_luminance	u(32)
num_knee_point_minus1	u(2)
for (i=0; i<=num_knee_point_minus1; i++)	
{	
input_knee_point[i]	u(16)
output_knee_point[i]	u(16)
}	
}	
}	
}	

[Fig. 40]

FIG. 40

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
knee_function_persistence_flag	u(1)
compression_flag	u(1)
input_d_range	u(32)
input_disp_luminance	u(32)
output_d_range	u(32)
output_disp_luminance	u(32)
num_knee_point_minus1	ue(v)
for(i=0;i<=num_knee_point_minus1;i++)	
{	
input_knee_point[i]	u(10)
output_knee_point[i]	u(10)
}	
}	
}	

[Fig. 41]

FIG. 41

knee_function_persistence_flag specifies the persistence of the knee function information SEI message.

knee_function_persistence_flag equal to 0 specifies that the knee function information applies to the current decoded picture only. **knee_function_persistence_flag** equal to 1 specifies that the knee function information persists in output order until any of the following conditions are true:

- A new QoS begins.
- A picture in an access unit containing a knee function information SEI message with the same value of **knee_function_id** is output having **PicOrderCntVal** greater than **PicOrderCnt(CurrPic)**.

compression_flag equals to 0 indicates that the knee function is decompression function. **compression_flag** equals to 1 indicates that the knee function is compression function.

num_knee_point_minus1 plus 1 specifies the number of knee points without counting the two default end points, (0, 0) and (1, 1).

input_d_range specifies the peak luminance level of the input picture in relative to the nominal luminance level in units of 0.1%. When the value of **input_d_range** is 0, the peak luminance level of the input picture is unspecified.

input_disp_luminance specifies the expected display brightness of peak luminance level for the input picture. The value of

input_disp_luminance is in units of candela per square metre. When the value of **input_disp_luminance** is 0, the expected display brightness of peak luminance level for the input picture is unspecified.

output_d_range specifies the peak luminance level of the output picture in relative to the nominal luminance level in units of 0.1%. When the value of **output_d_range** is 0, the peak luminance level of the output picture is unspecified.

output_disp_luminance specifies the expected display brightness of peak luminance level for the output picture. The value of **output_disp_luminance** is in units of candela per square metre. When the value of **output_disp_luminance** is 0, the expected display brightness of peak luminance level for the output picture is unspecified.

input_knee_point[i] specifies the luminance level of *i*-th knee point for input picture. The luminance level of the knee point for input picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **input_knee_point** shall be greater than 0 and less than 1000. The value of *i*-th input knee point shall be greater than the value of (*i*-1)-th input knee point.

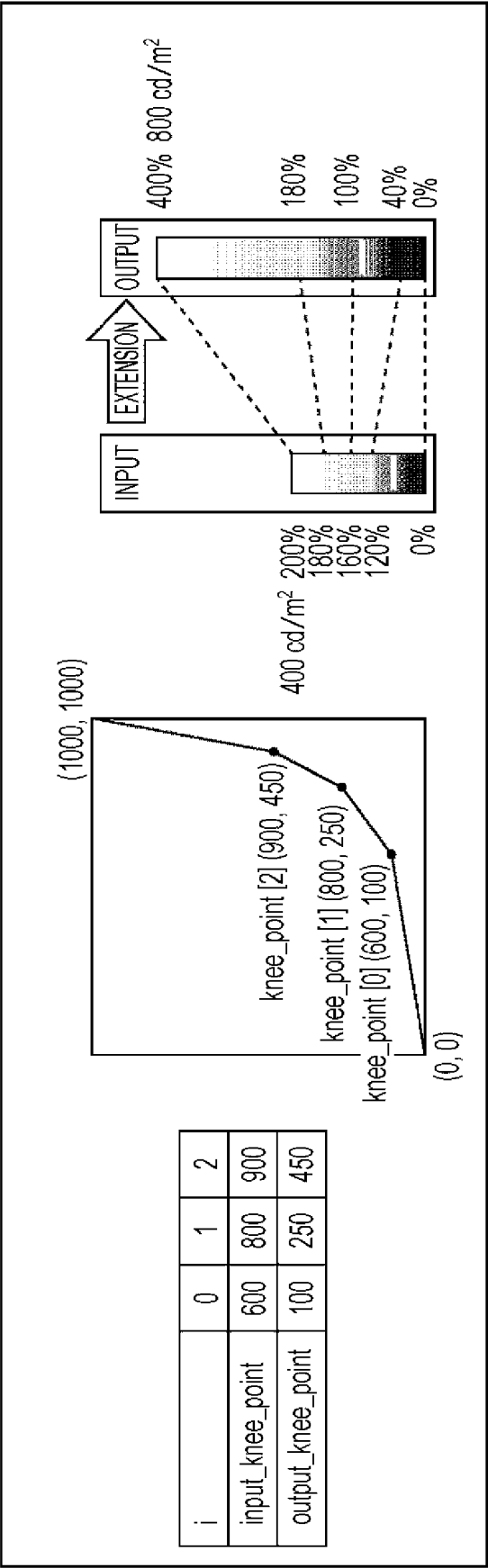
output_knee_point[i] specifies the luminance level of *i*-th knee for output picture. The luminance level of the knee point for output picture is normalized in range of 0 to 1.0 in units of 0.1%. The value of **output_knee_point** is expressed in range of 0 to 1000.

NOTE 2 - The coordinate of the *i*-th knee point is defined as (**input_knee_point[i]**, **output_knee_point[i]**).

Knee function is composed of several lines which starts with the point (0,0) ends with the point (1,1) through all knee points in ascending order of *i*.

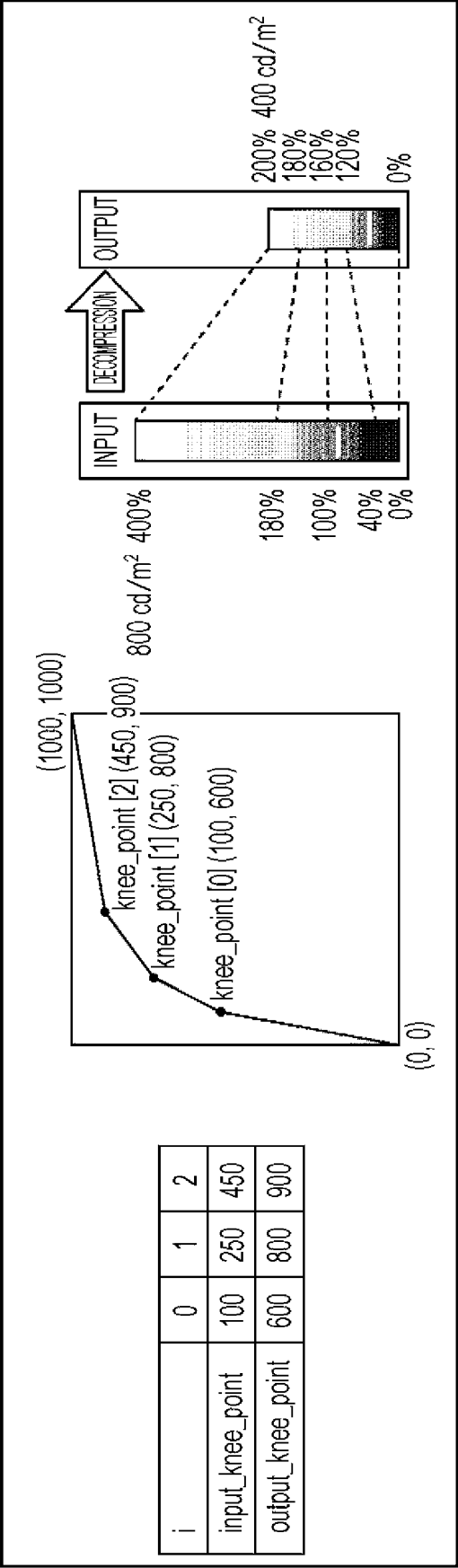
[Fig. 42]

FIG. 42



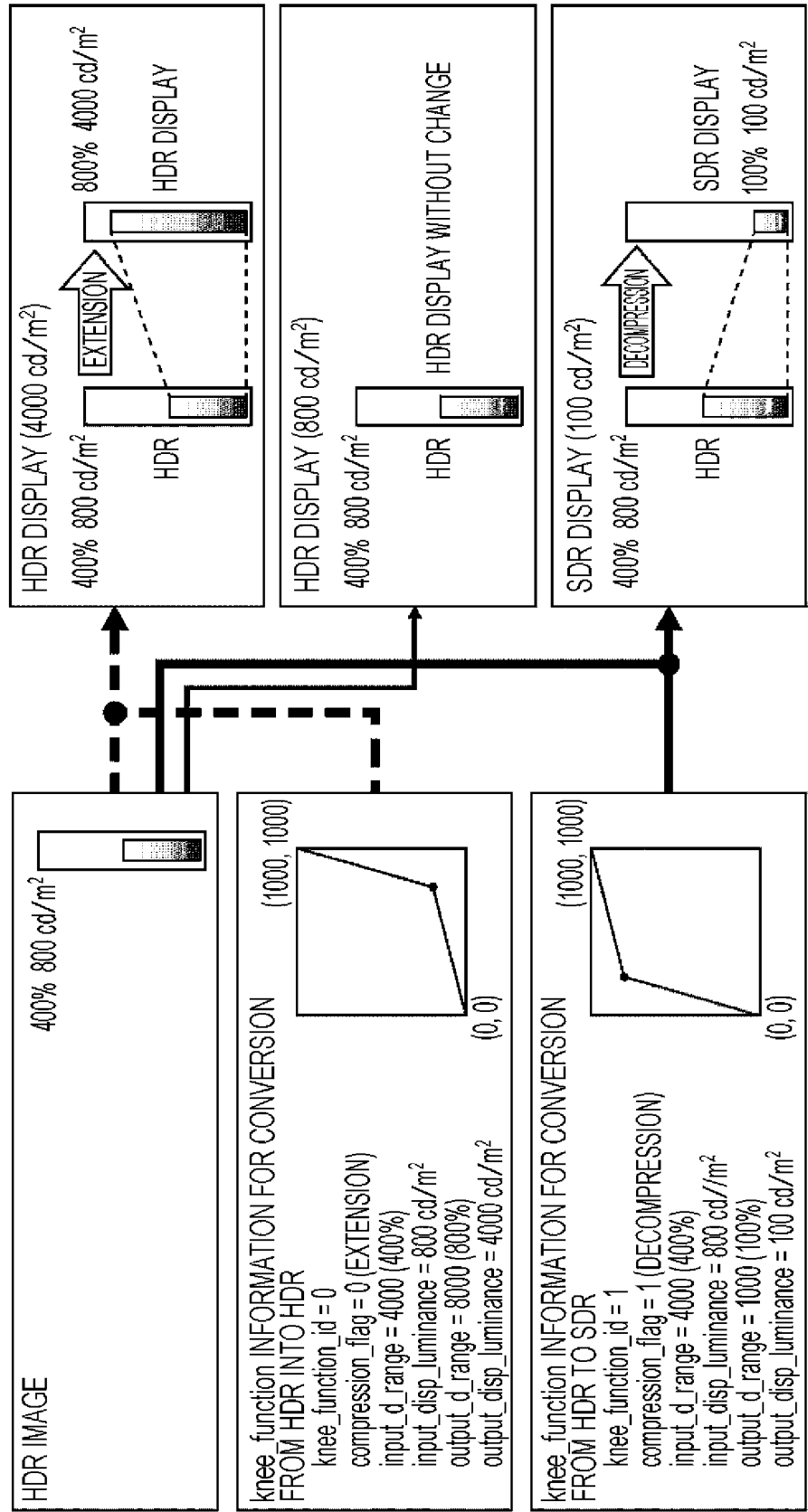
[Fig. 43]

FIG. 43



[Fig. 44]

FIG. 44



[Fig. 45]

FIG. 45

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
-----snip-----	
if(tone_map_model_id == 0) {	
-----snip-----	
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}else if(tone_map_model_id == 5) {	
compression_flag	u(1)
input_d_range	u(32)
input_disp_luminance	u(32)
output_d_range	u(32)
output_disp_luminance	u(32)
num_knee_point_minus1	ue(v)
for (i=0; i<=num_knee_point_minus1; i++)	
{	
input_knee_point[i]	u(10)
output_knee_point[i]	u(10)
}	
}	
}	
}	

[Fig. 46]

FIG. 46

Tone Mapping Information Box('tinf')**Box Type** 'tinf'**Container** Sample Table Box('stbl') or
Track Fragment Box('traf')**Mandatory** No**Quantity** Zero, or one in every track/track fragment

```

aligned(8) class ToneMappingInformationBox
    extends FullBox('tinf', version=0, flags=0)
{
    unsigned int(32) number_of_tone_mapping_info
    for (i=0; i<number_of_tone_mapping_info; i++)
    {
        ToneMapInfo tonemap;
    }
}

```

[Fig. 47]

FIG. 47

```

aligned(8) class ToneMapInfo
{
    unsigned int(32) tone_map_id
    unsigned int(7) padding_value
    unsigned int(1) tone_map_cancel_flag
    if(!tone_map_cancel_flag) {
        unsigned int(7) padding_value
        unsigned int(1) tone_map_persistence_flag
        unsigned int(8) coded_data_bit_depth
        unsigned int(8) target_bit_depth
        unsigned int(8) tone_map_model_id
        if(tone_map_model_id == 0) {
            unsigned int(32) min_value
            unsigned int(32) max_value
        } else if(tone_map_model_id == 1) {
            unsigned int(32) sigmoid_midpoint
            unsigned int(32) sigmoid_width
        } else if(tone_map_model_id == 2) {
            for (j=0; j<(1<<target_bit_depth); j++)
                unsigned int(256) start_of_coded_interval[j]
        } else if(tone_map_model_id == 3) {
            unsigned int(16) num_pivots
            for (j=0; j<num_pivots; j++) {
                unsigned int(256) coded_pivot_value[j]
                unsigned int(256) target_pivot_value[j]
            }
        } else if(tone_map_model_id == 4) {
            unsigned int(8) camera_iso_speed_idc
            if(camera_iso_speed_idc == EXTENDED_ISO)
                unsigned int(32) camera_iso_speed_value
            unsigned int(8) exposure_index_idc
            if(exposure_index_idc == EXTENDED_ISO)
                unsigned int(32) exposure_index_value
            unsigned int(7) padding_value
            unsigned int(1) exposure_compensation_value_sign_flag
            unsigned int(16) exposure_compensation_value_numerator
            unsigned int(16) exposure_compensation_value_denom_idc
            unsigned int(32) ref_screen_luminance_white
            unsigned int(32) extended_range_white_level
            unsigned int(16) nominal_black_level_code_value
            unsigned int(16) nominal_white_level_code_value
            unsigned int(16) extended_white_level_code_value
        } else if(tone_map_model_id == 5) {
            unsigned int(1) compression_flag
            unsigned int(32) d_range
            unsigned int(32) d_range_disp_luminance
            unsigned int(2) num_knee_point_minus1
            for (i=0; i<=num_knee_point_minus1; i++) {
                unsigned int(16) input_knee_point[i]
                unsigned int(16) output_knee_point[i]
            }
        }
    }
}

```

[Fig. 48]

FIG. 48

input_knee_point[i] MAY BE EQUAL TO OR LESS THAN input_knee_point[i-1].

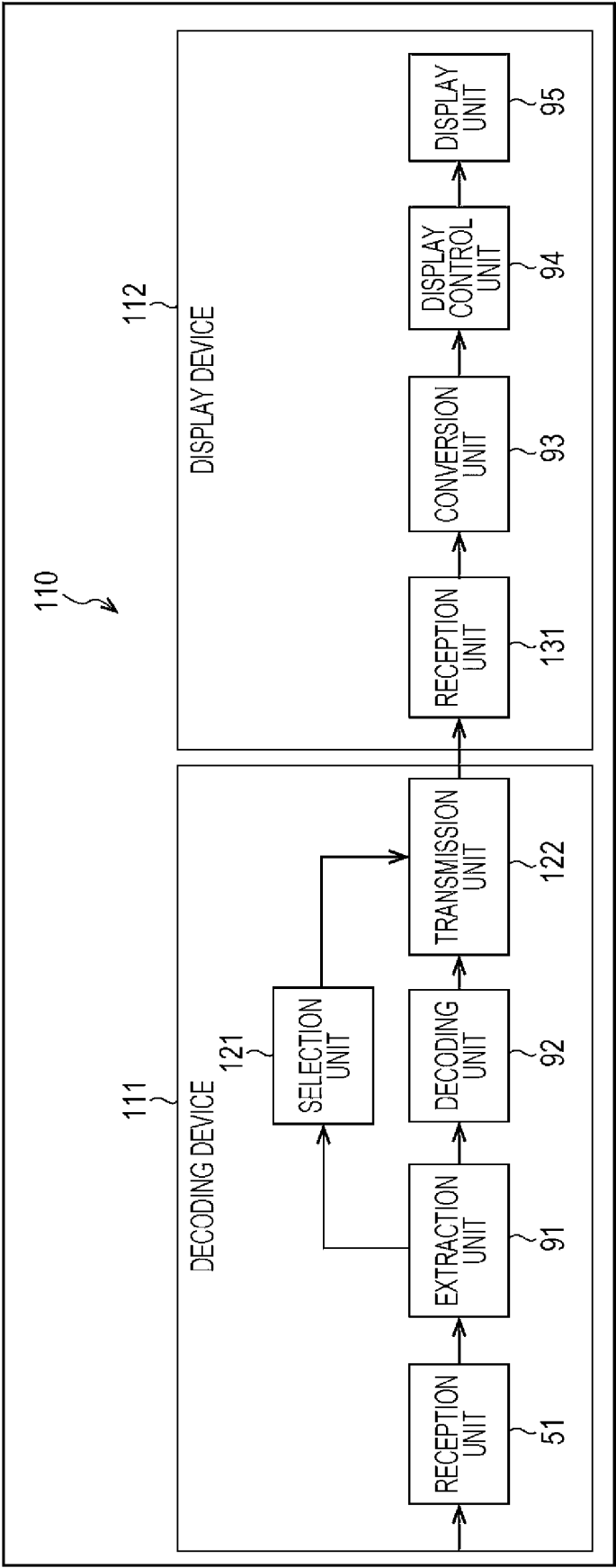
knee_function IS FORMED BY PLURAL STRAIGHT LINES WHICH CONNECT ALL knee points TO EACH OTHER IN ASCENDING ORDER OF input_knee_point[i] FROM (0.0, 0.0) TO (1.0, 1.0).

WHEN DECODED IMAGE IS CONVERTED INTO IMAGE THROUGH KNEE CONVERSION, CONVERSION MAY BE PERFORMED BY USING APPROXIMATE FUNCTION OF knee_function.

APPROXIMATE FUNCTION OF knee_function CAN BE OBTAINED BY USING PLURAL STRAIGHT LINES WHICH CONNECT knee_points[0] TO [N] IN ASCENDING ORDER OF input_knee_point[i] FROM (0.0, 0.0) TO (1.0, 1.0) FOR ANY N OF $0 \leq N \leq \text{num_knee_points_minus1}$.

[Fig. 49]

FIG. 49



[Fig. 50A]

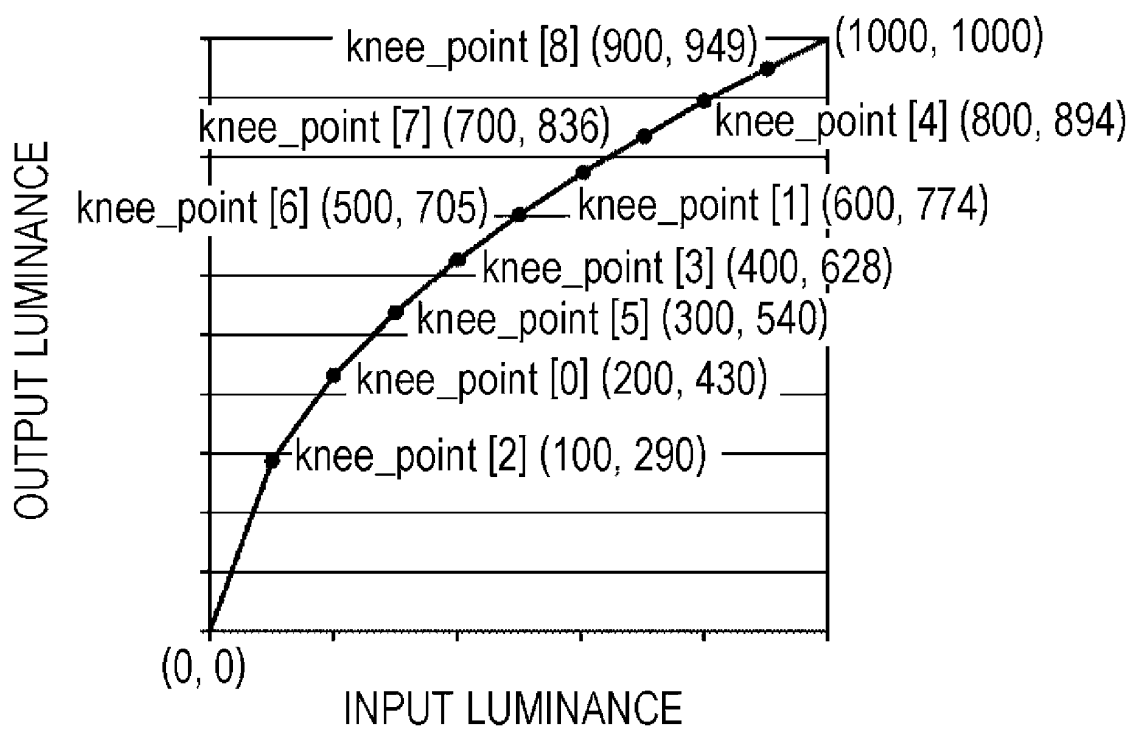
FIG. 50A

num_knee_points_minus1 = 8

i	0	1	2	3	4	5	6	7	8
input_knee_point	200	600	100	400	800	300	500	700	900
output_knee_point	433	774	290	628	894	540	705	836	949

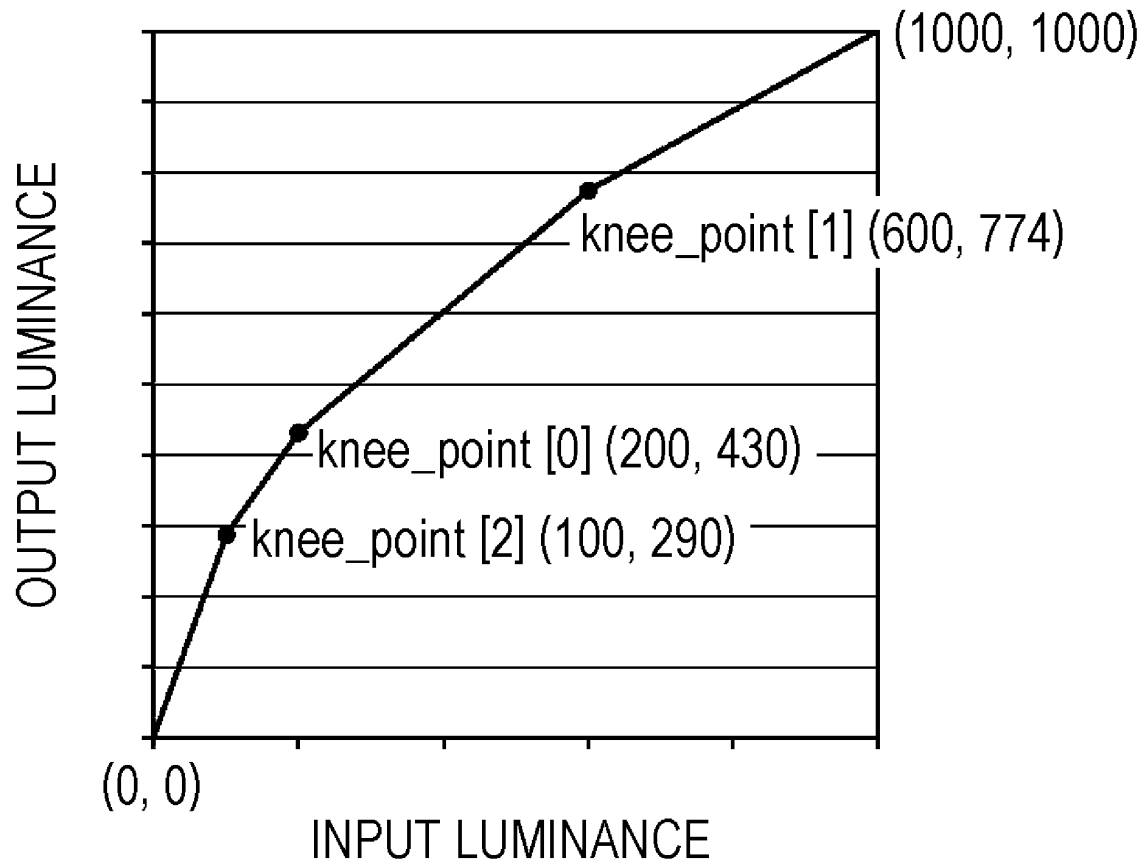
[Fig. 50B]

FIG. 50B



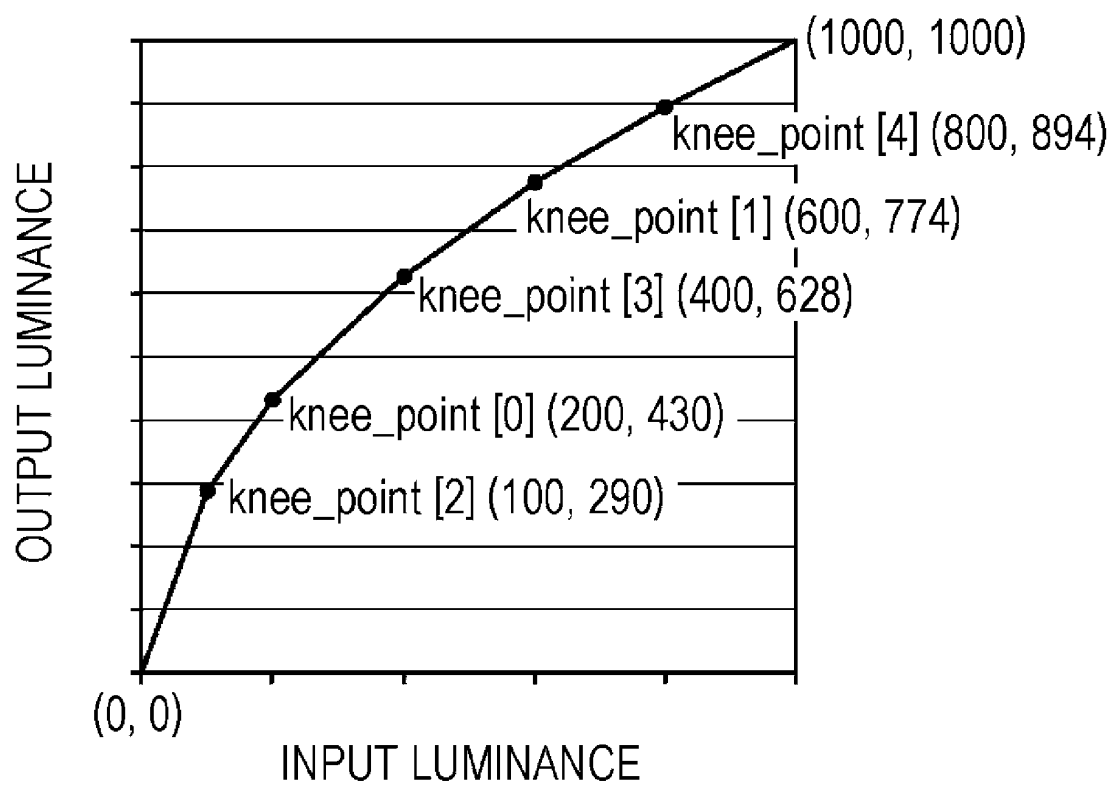
[Fig. 51]

FIG. 51



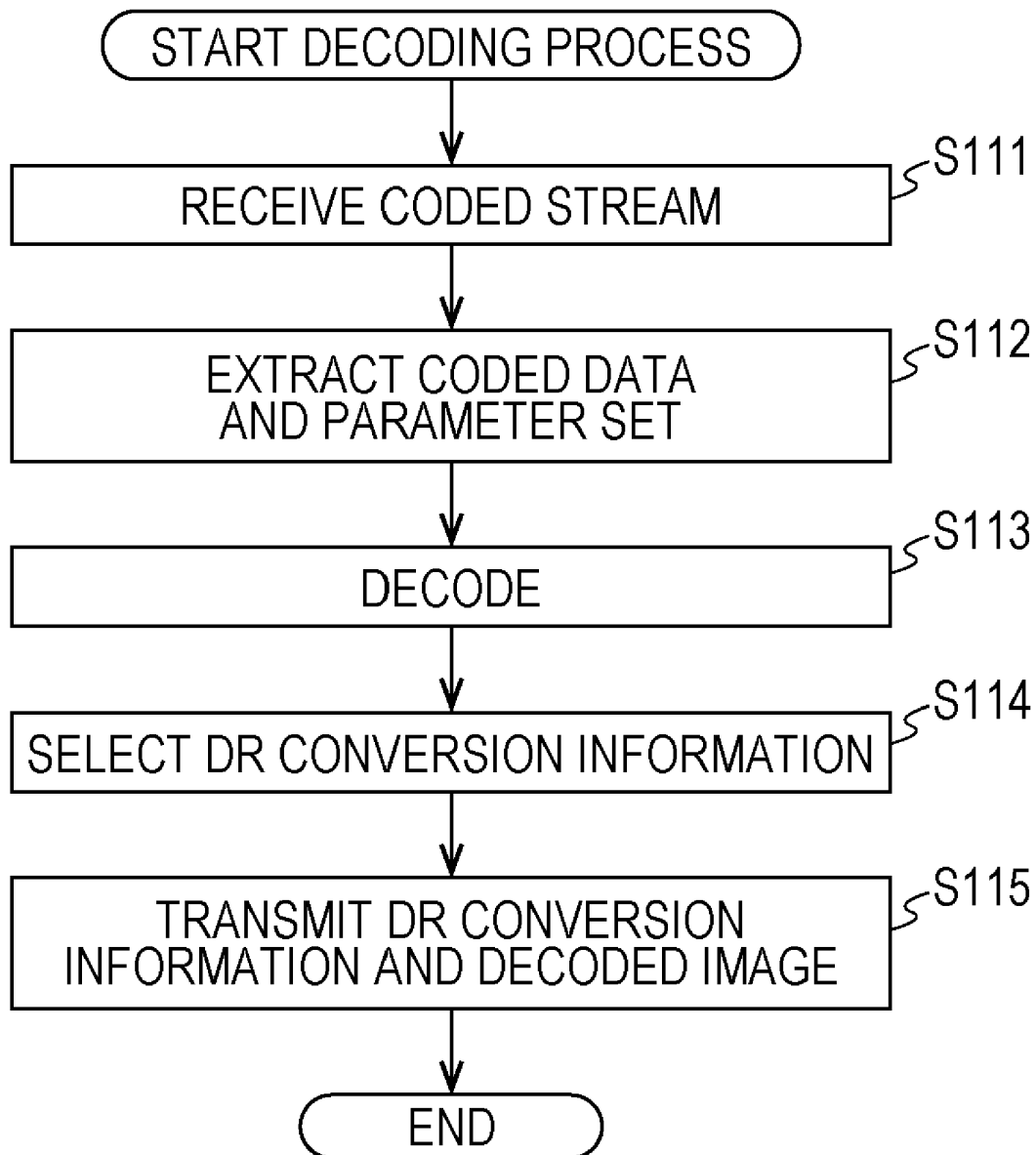
[Fig. 52]

FIG. 52



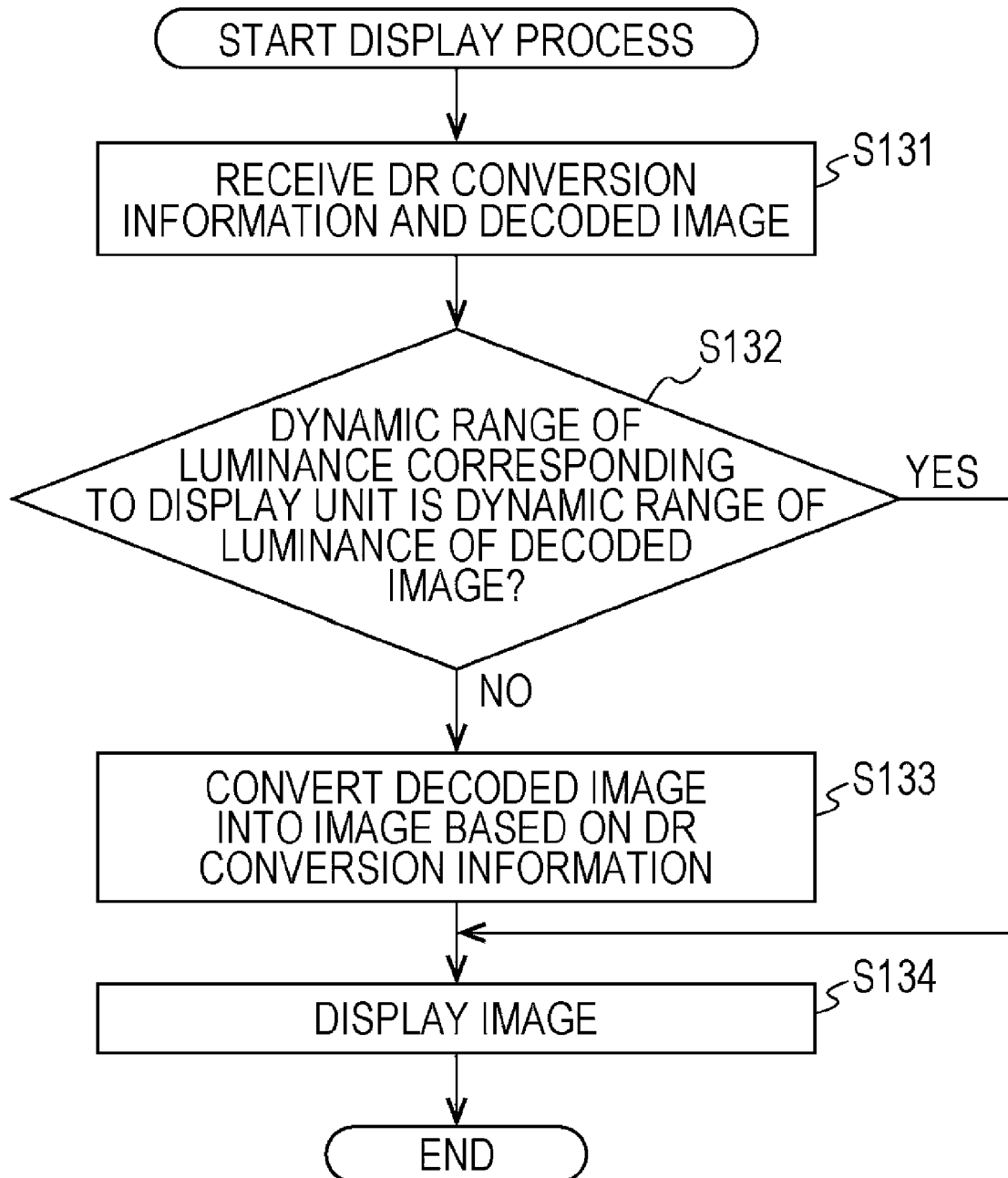
[Fig. 53]

FIG. 53



[Fig. 54]

FIG. 54



[Fig. 55]

FIG. 55

knee_function_info (payloadSize) {	Descriptor
knee_function_id	ue (v)
knee_function_cancel_flag	u (1)
if (!knee_function_cancel_flag) {	
knee_function_persistence_flag	u (1)
mapping_flag	u (1)
input_d_range	u (32)
input_disp_luminance	u (32)
output_d_range	u (32)
output_disp_luminance	u (32)
num_knee_points_minus1	ue (v)
for (i = 0; i <= num_knee_points_minus1; i++) {	
input_knee_point[i]	u (10)
output_knee_point[i]	u (10)
}	
for (j = 0; j <= num_knee_points_minus1; j++) {	
approximate_knee_point_index[j]	u (10)
}	
}	
}	

[Fig. 56]

FIG. 56

WHEN DECODED IMAGE IS CONVERTED INTO IMAGE THROUGH KNEE CONVERSION, CONVERSION MAY BE PERFORMED BY USING APPROXIMATE FUNCTION OF `knee_function`.

APPROXIMATE FUNCTION OF `knee_function` CAN BE OBTAINED BY USING PLURAL STRAIGHT LINES WHICH CONNECT `knee_points` [`approximate_knee_point_index`[0]] TO [`approximate_knee_point_index`[N]] IN ASCENDING ORDER OF `i` FROM (0.0, 0.0) TO (1.0, 1.0) FOR ANY `N` OF $0 \leq N \leq \text{num_knee_points_minus1}$.

[Fig. 57A]

FIG. 57A

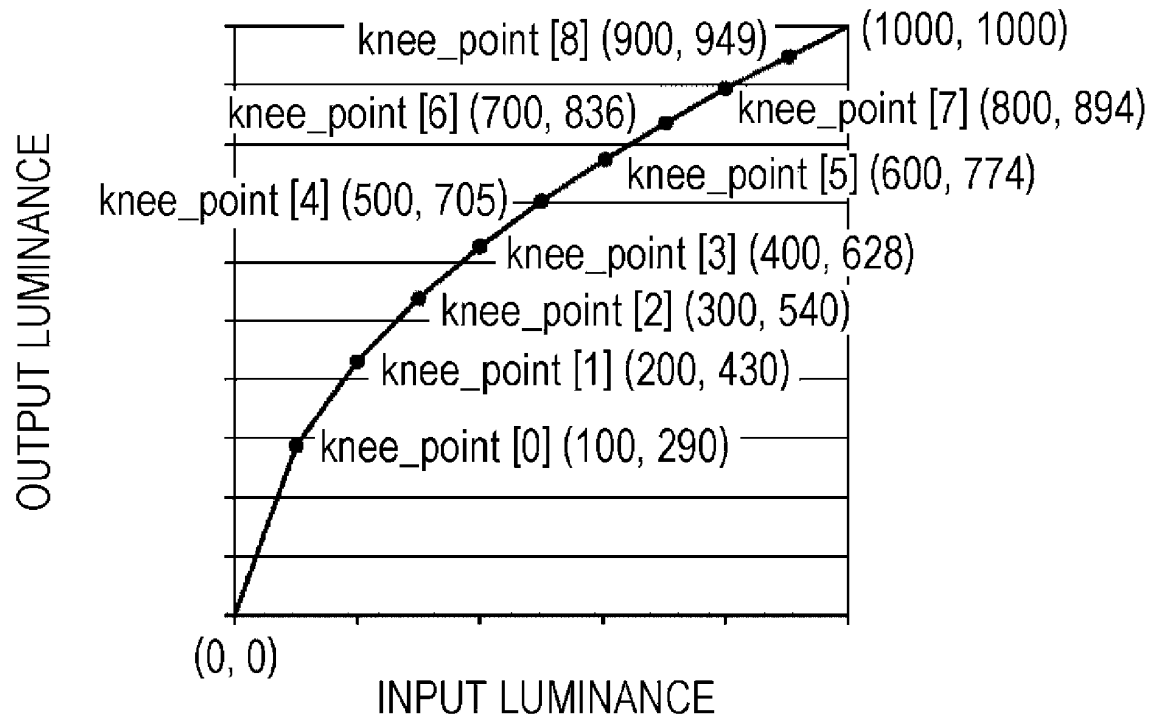
`num_knee_points_minus1` = 8

<code>i</code>	0	1	2	3	4	5	6	7	8
<code>input_knee_point</code>	100	200	300	400	500	600	700	800	900
<code>output_knee_point</code>	290	433	540	628	705	774	836	894	949

<code>j</code>	0	1	2	3	4	5	6	7	8
<code>approximate_knee_point_index</code>	1	5	0	3	7	2	4	6	8

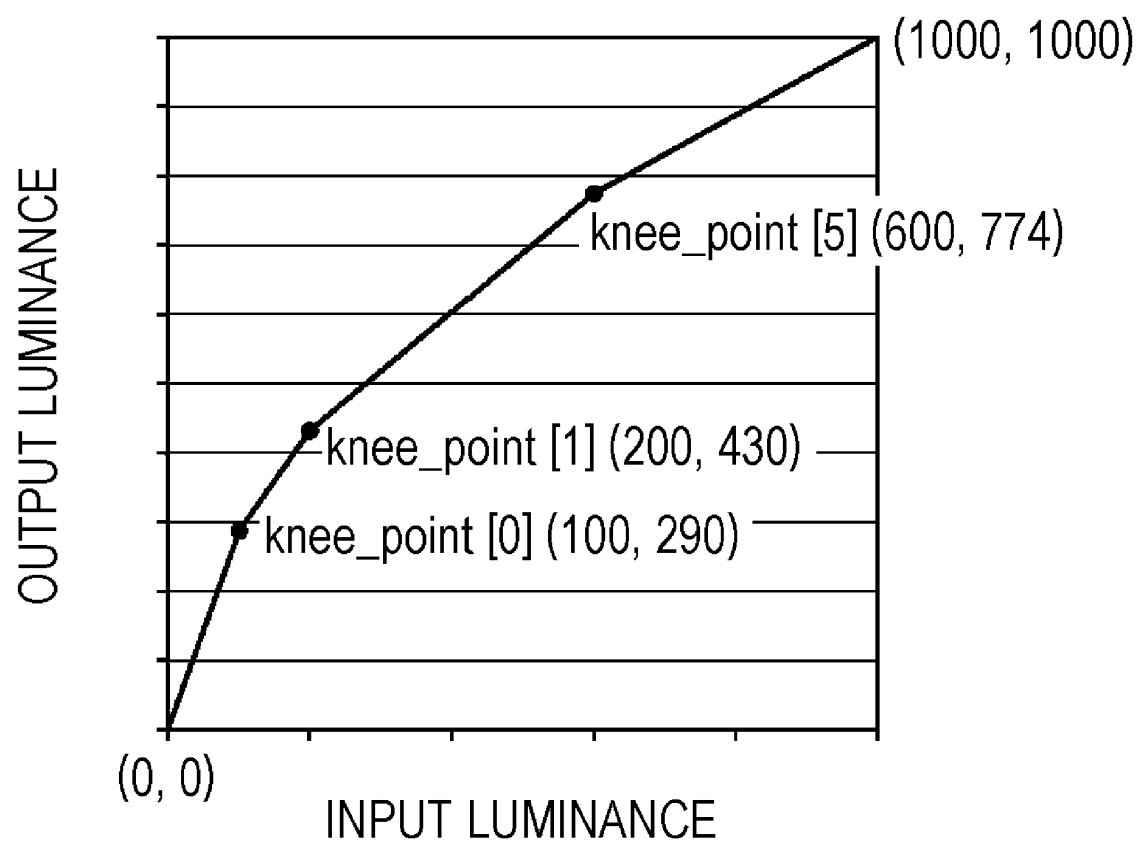
[Fig. 57B]

FIG. 57B



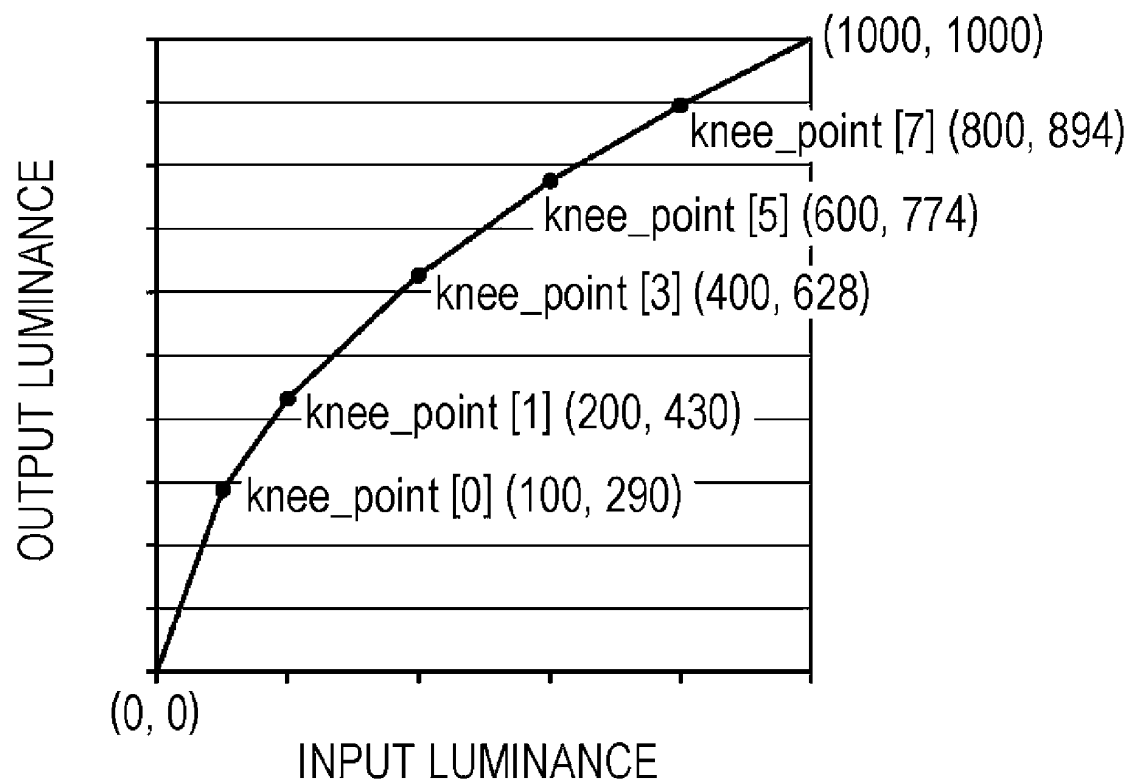
[Fig. 58]

FIG. 58



[Fig. 59]

FIG. 59



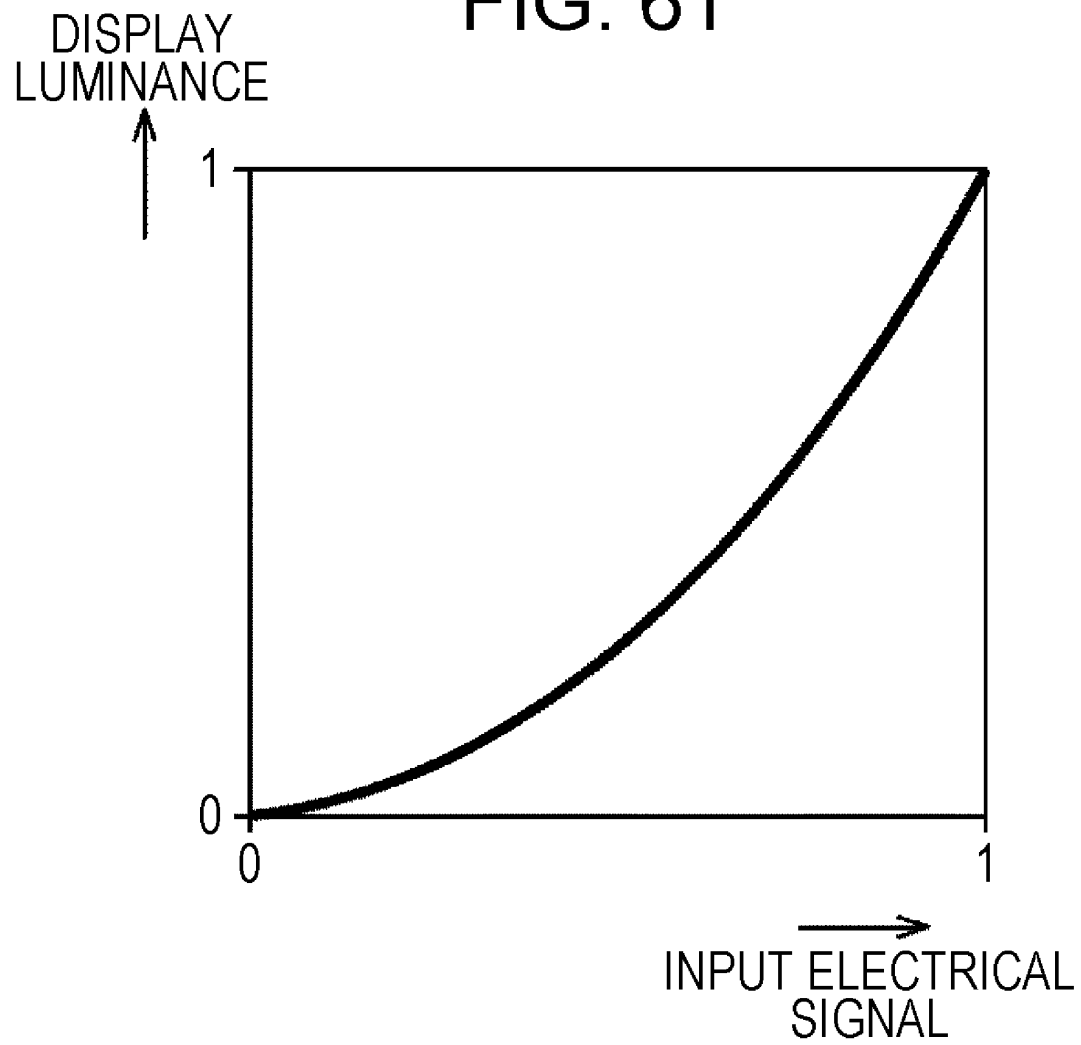
[Fig. 60]

FIG. 60

approximate_knee_function_info (payloadSize) {	Descriptor
approximate_knee_function_id	ue(v)
approximate_knee_function_cancel_flag	u(1)
if(! approximate_knee_function_cancel_flag) {	
ref_knee_function_id	ue(v)
approximate_knee_function_persistence_flag	u(1)
num_approximate_knee_point_indices_minus1	ue(v)
for(j = 0; j <= num_approximate_knee_point_indices_minus1; j++) {	
approximate_knee_point_index[j]	u(10)
}	
}	
}	

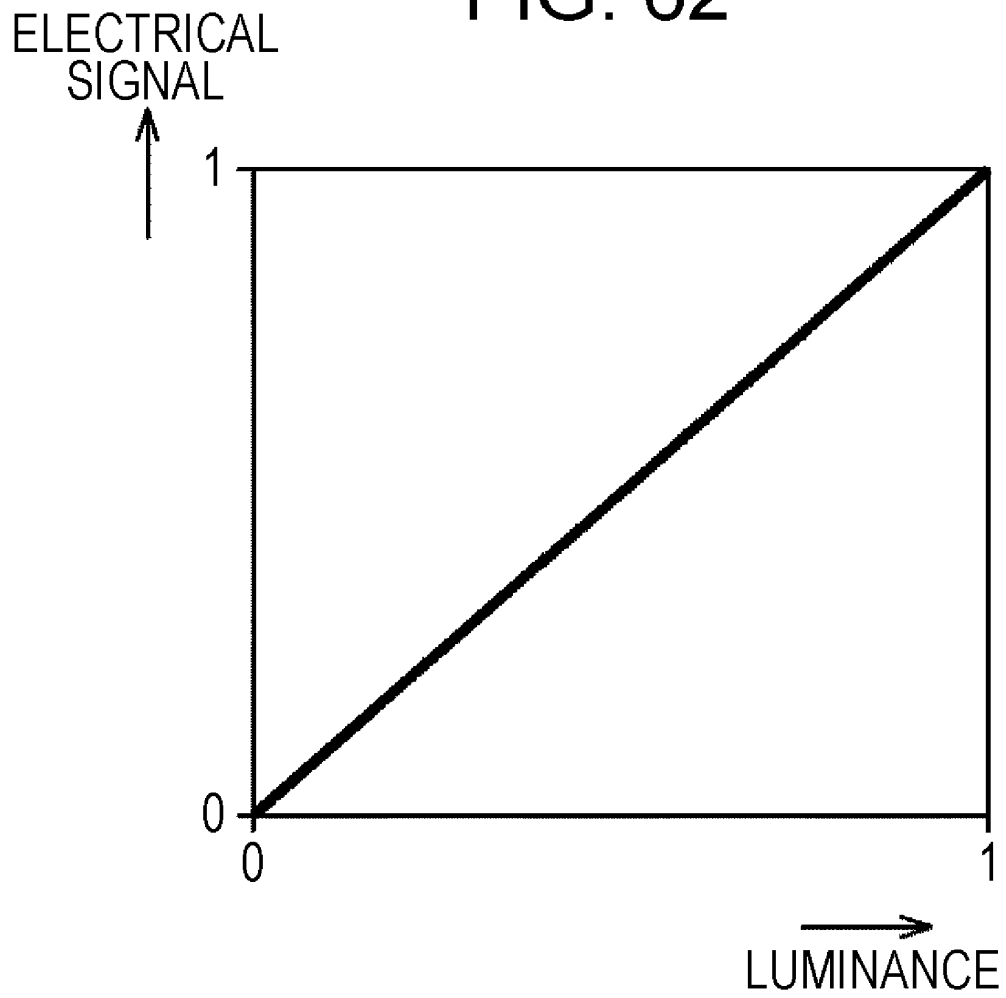
[Fig. 61]

FIG. 61



[Fig. 62]

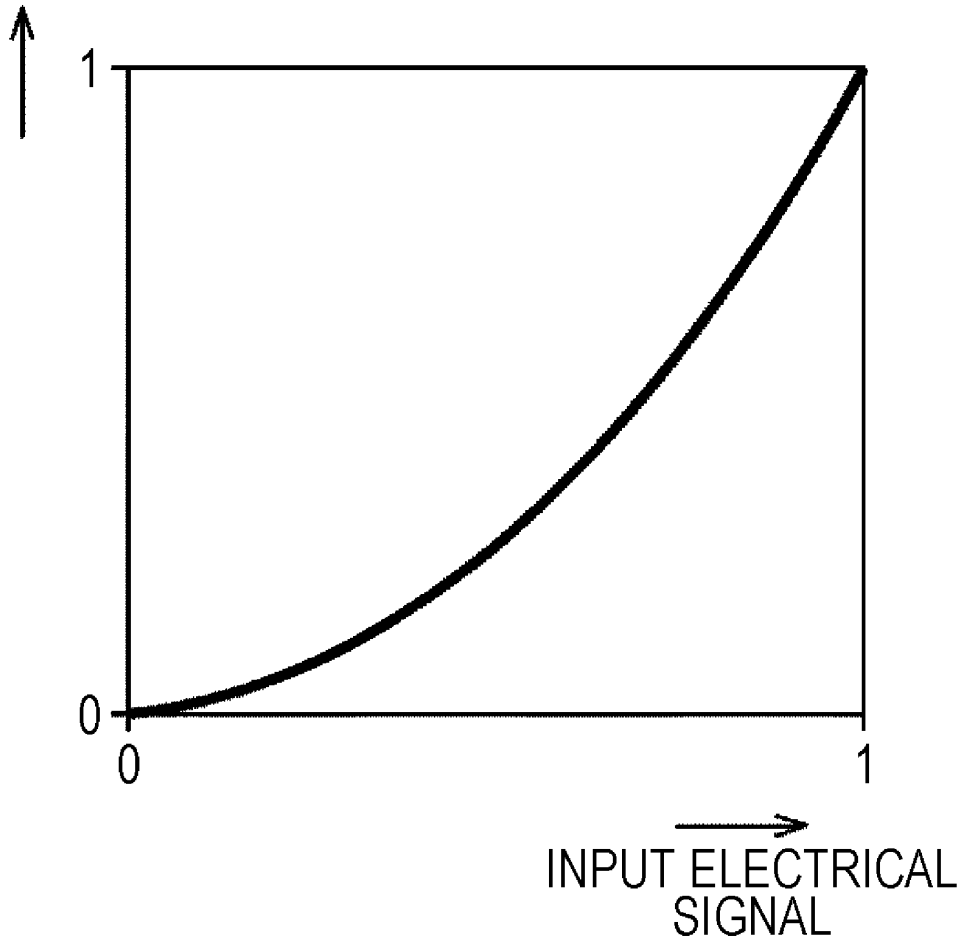
FIG. 62



[Fig. 63]

DISPLAY
LUMINANCE

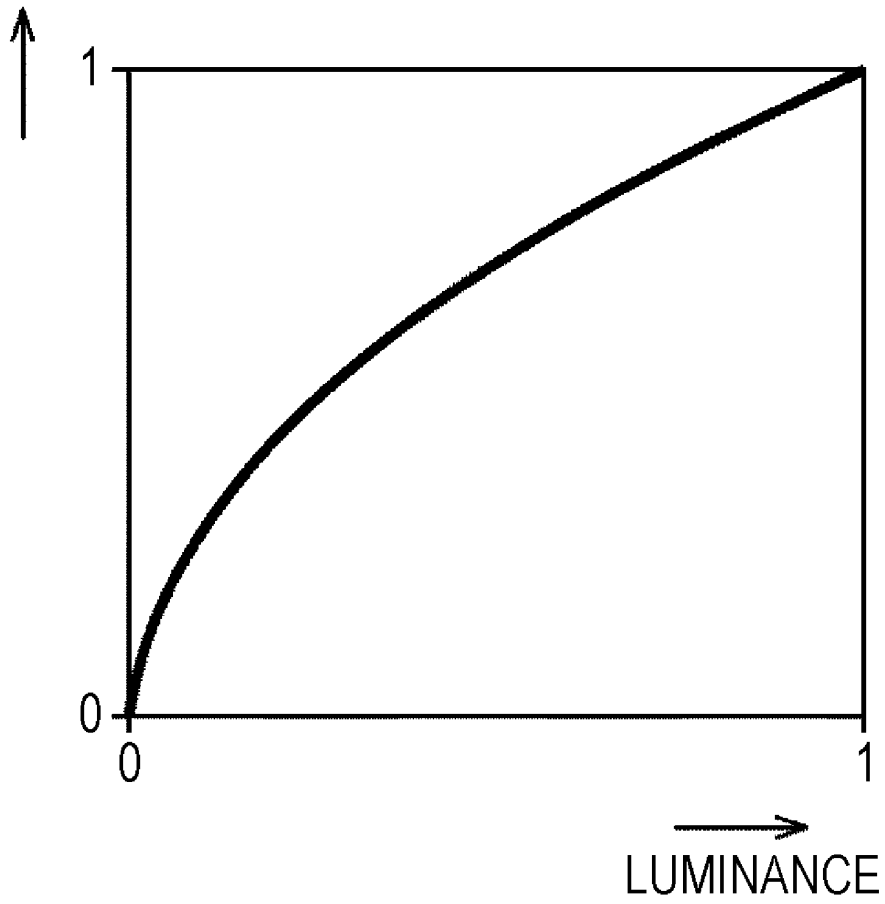
FIG. 63



[Fig. 64]

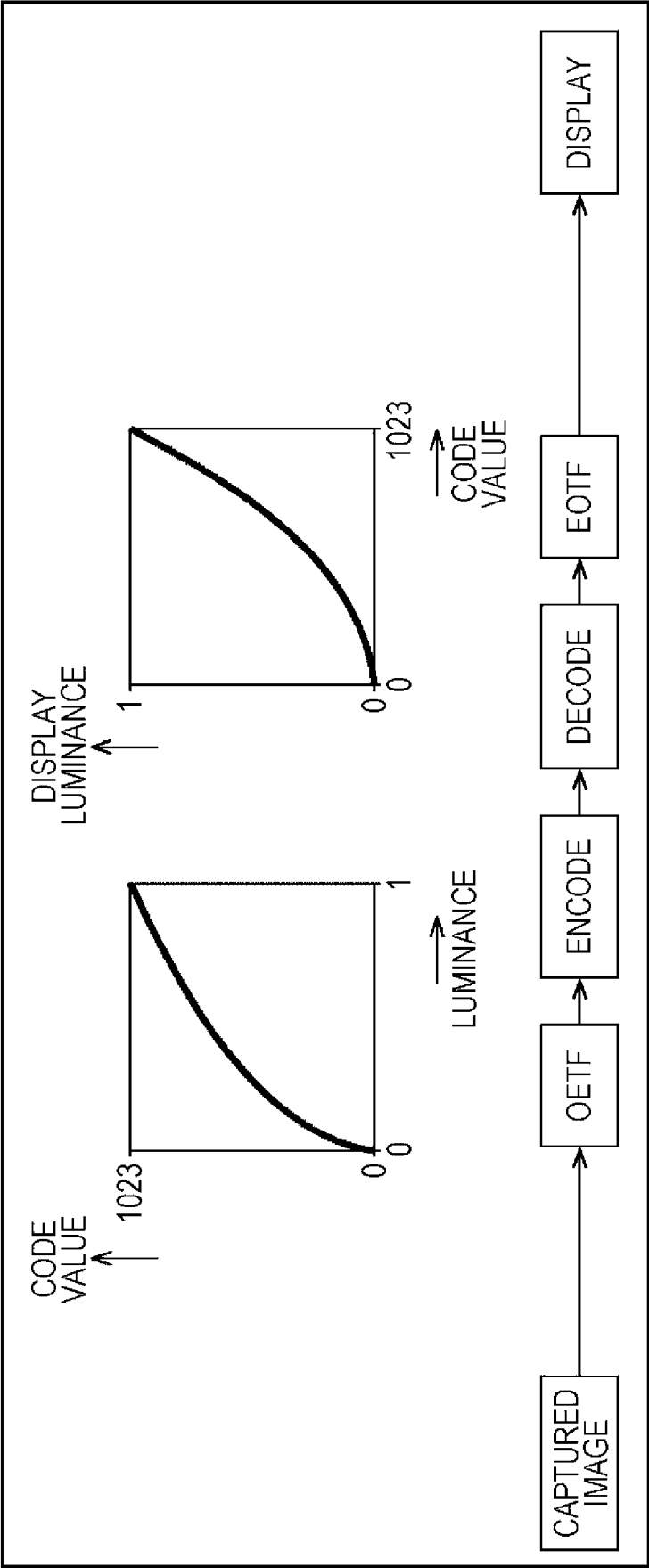
ELECTRICAL
SIGNAL

FIG. 64



[Fig. 65]

FIG. 65



[Fig. 66]

CODE
VALUE

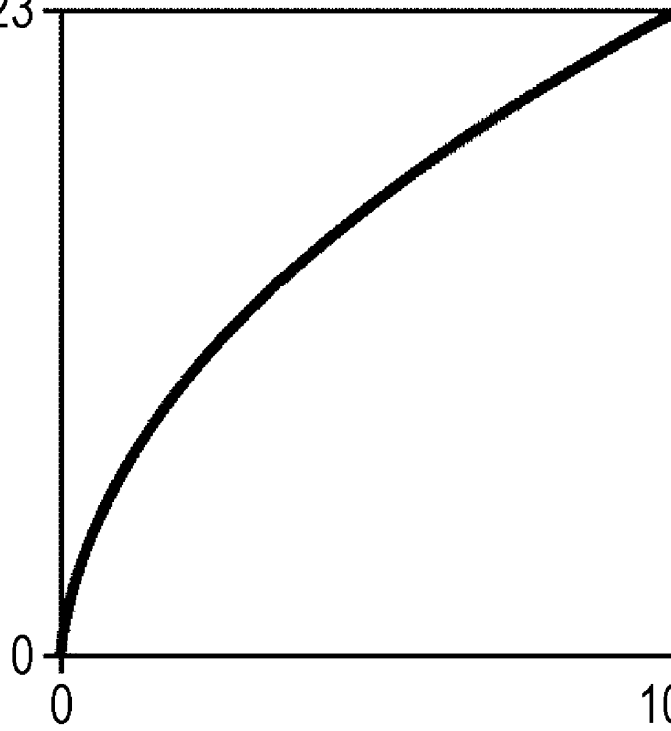
FIG. 66

1023

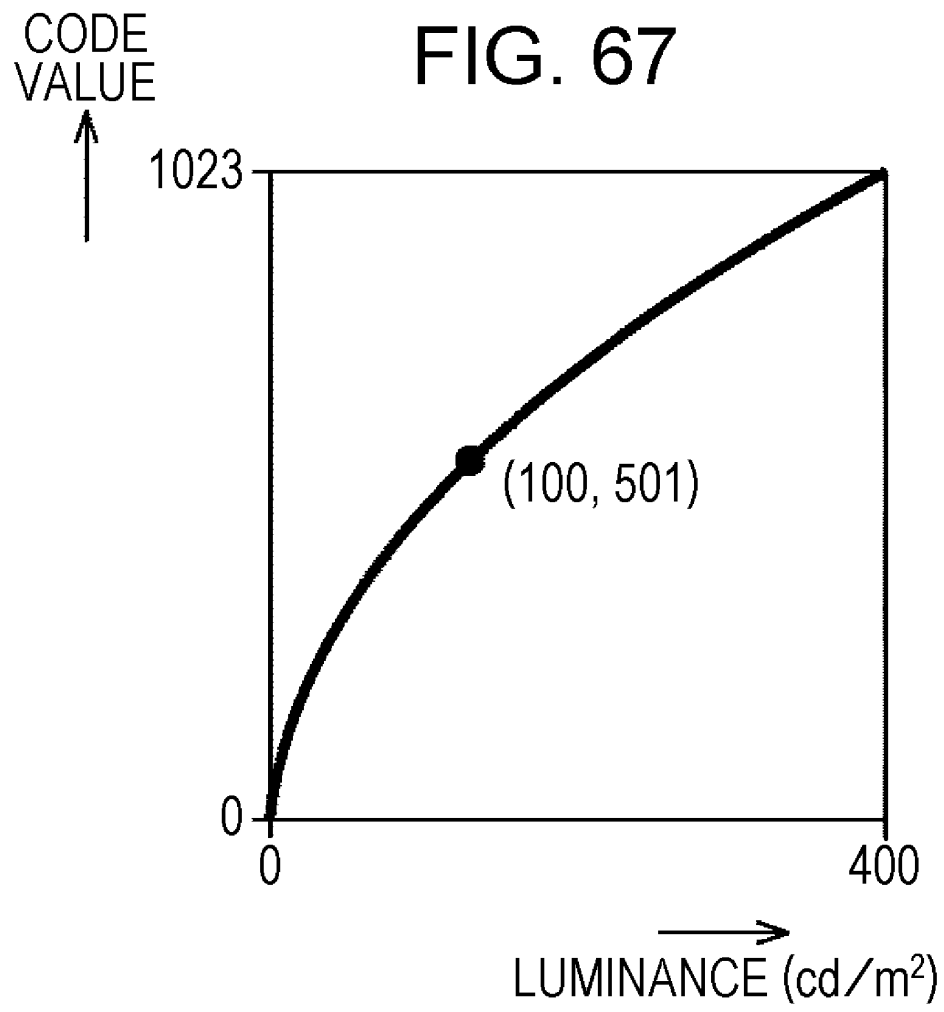
0

0

100

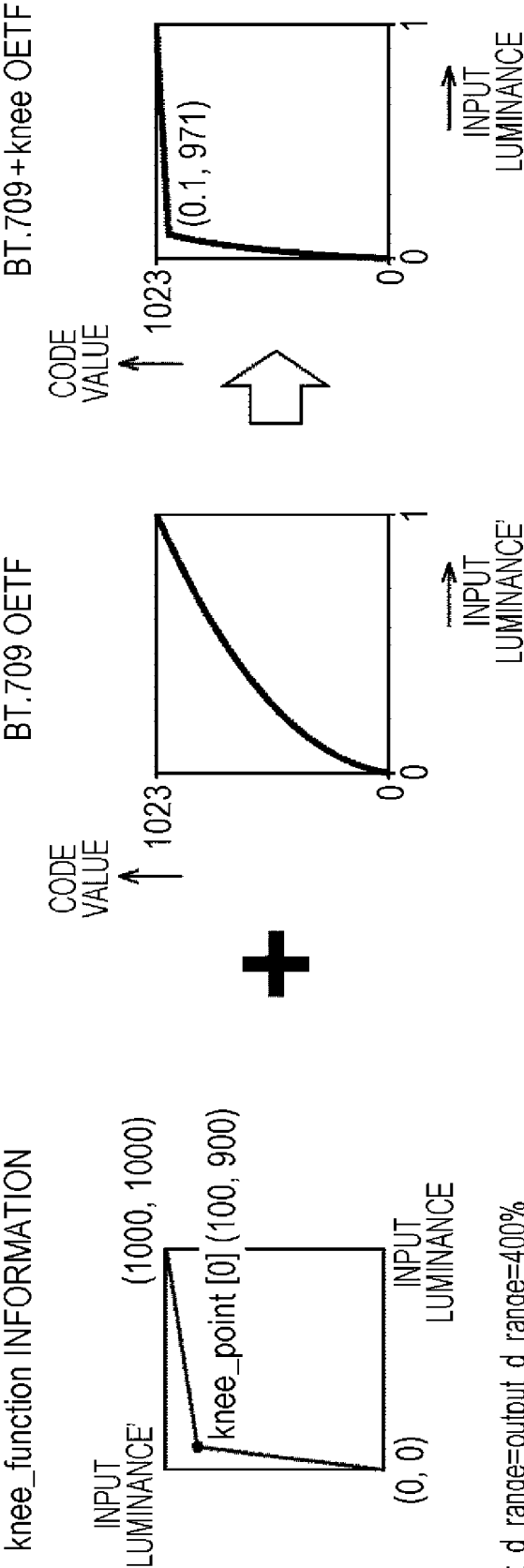
LUMINANCE (cd/m²)

[Fig. 67]



[Fig. 68]

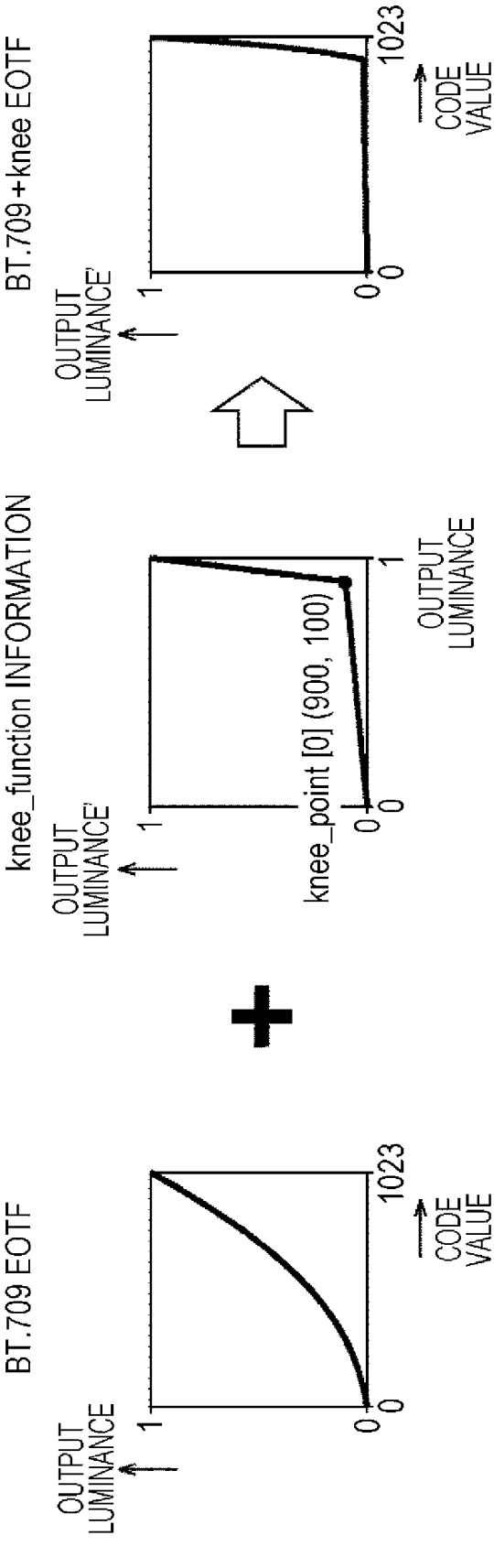
FIG. 68



input_d_range=output_d_range=400%
input_disp_luminance=output_disp_luminance=400 cd/m²

[Fig. 69]

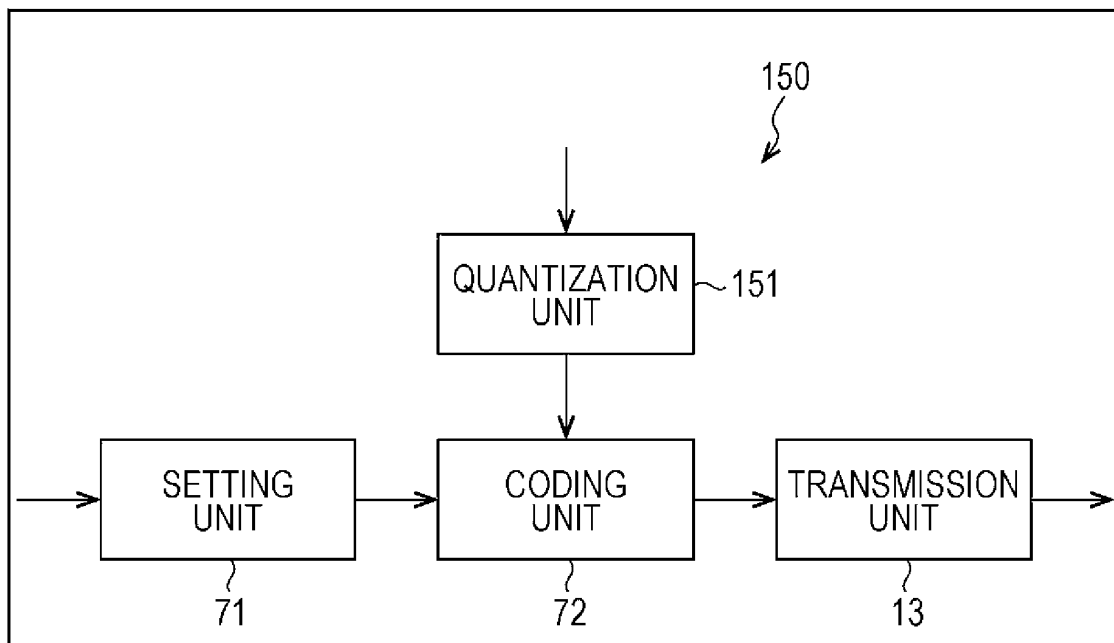
FIG. 69



input_d_range=output_d_range=400%
input_disp_luminance=output_disp_luminance=400 cd/m²

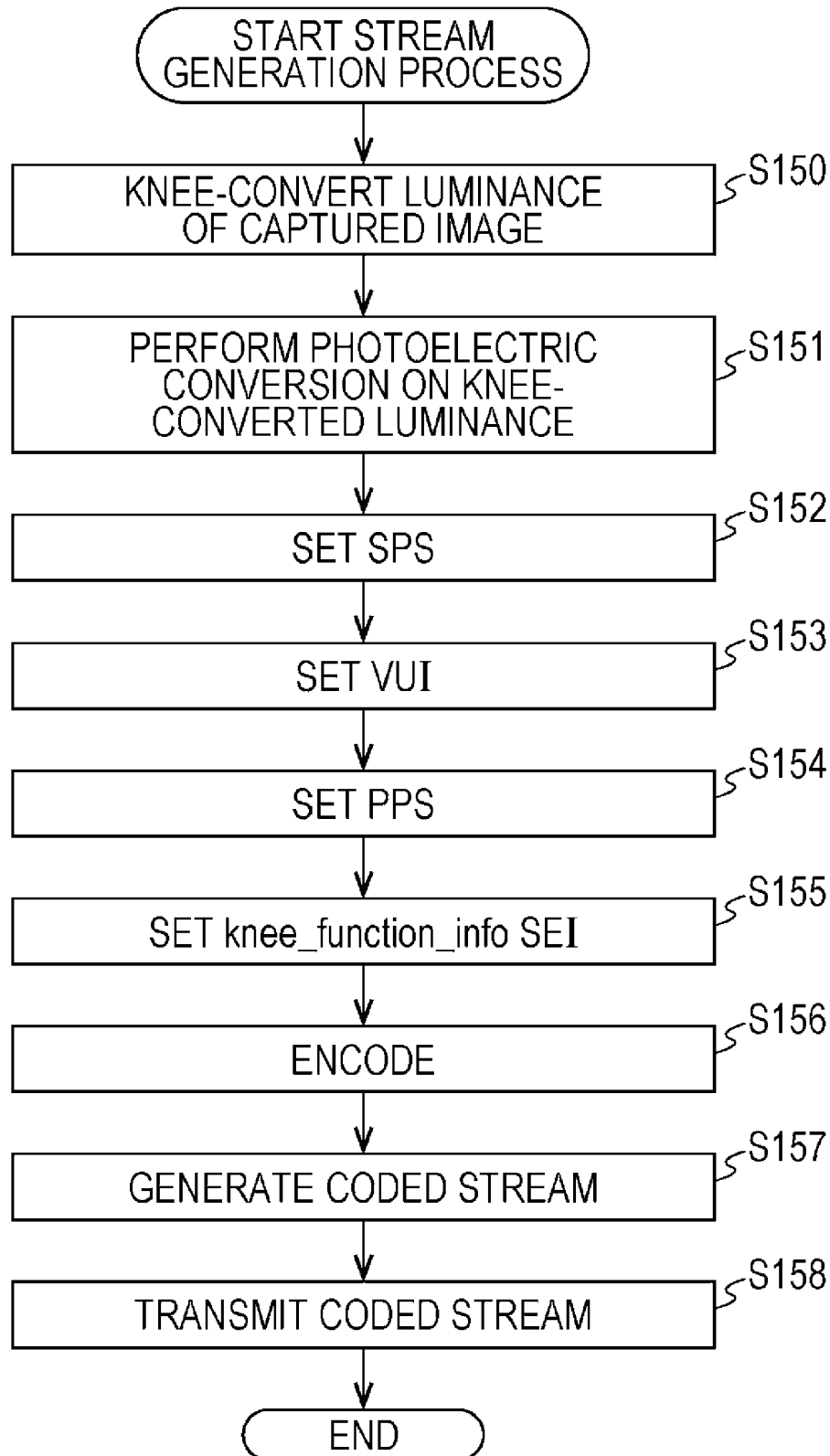
[Fig. 70]

FIG. 70



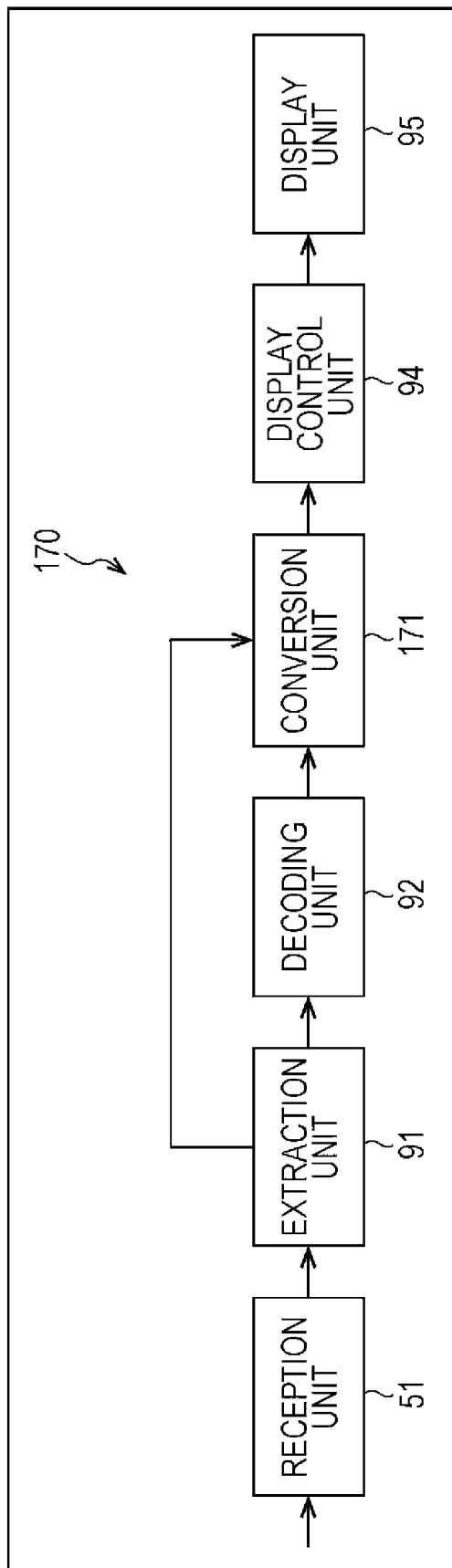
[Fig. 71]

FIG. 71



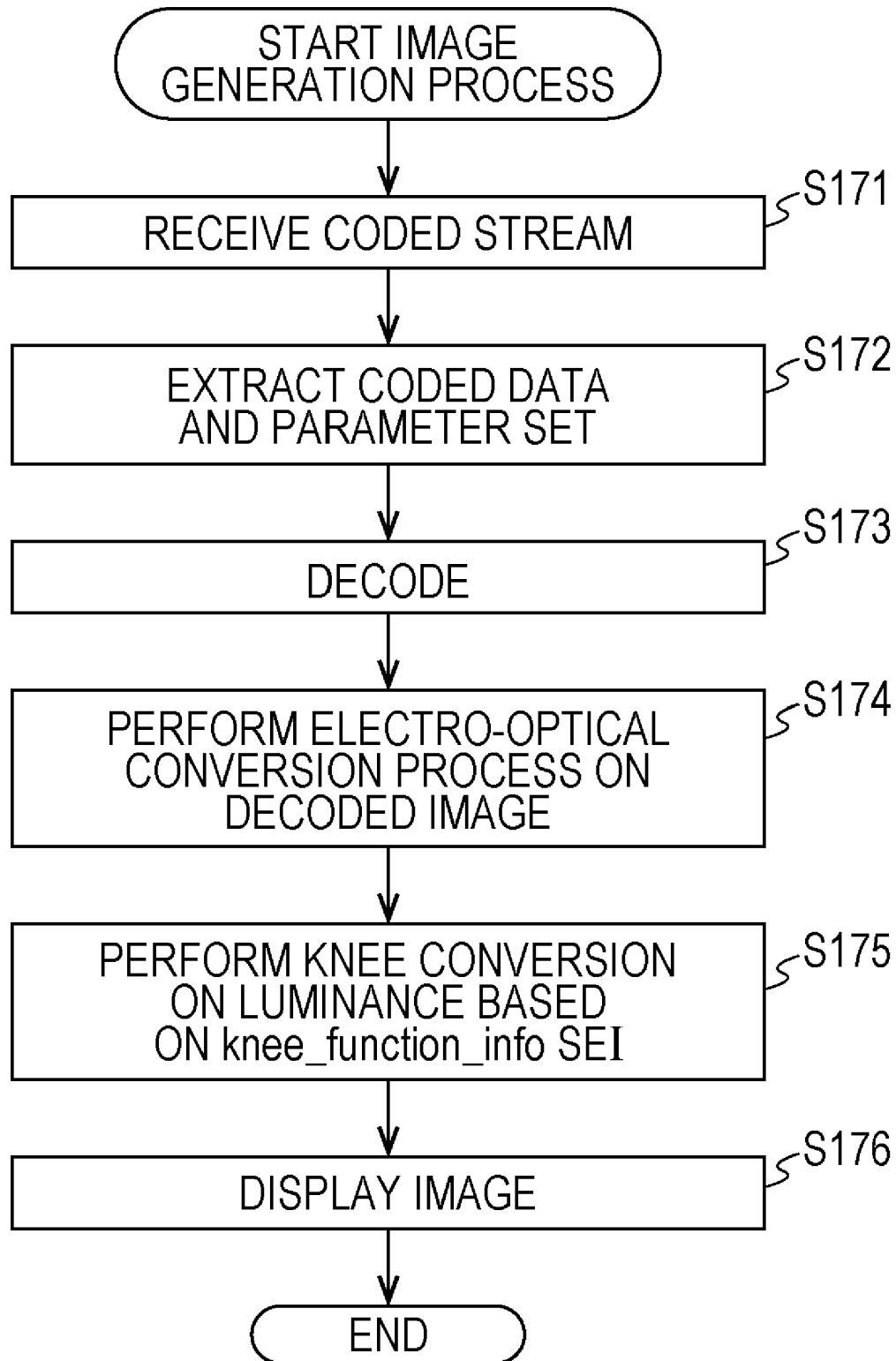
[Fig. 72]

FIG. 72



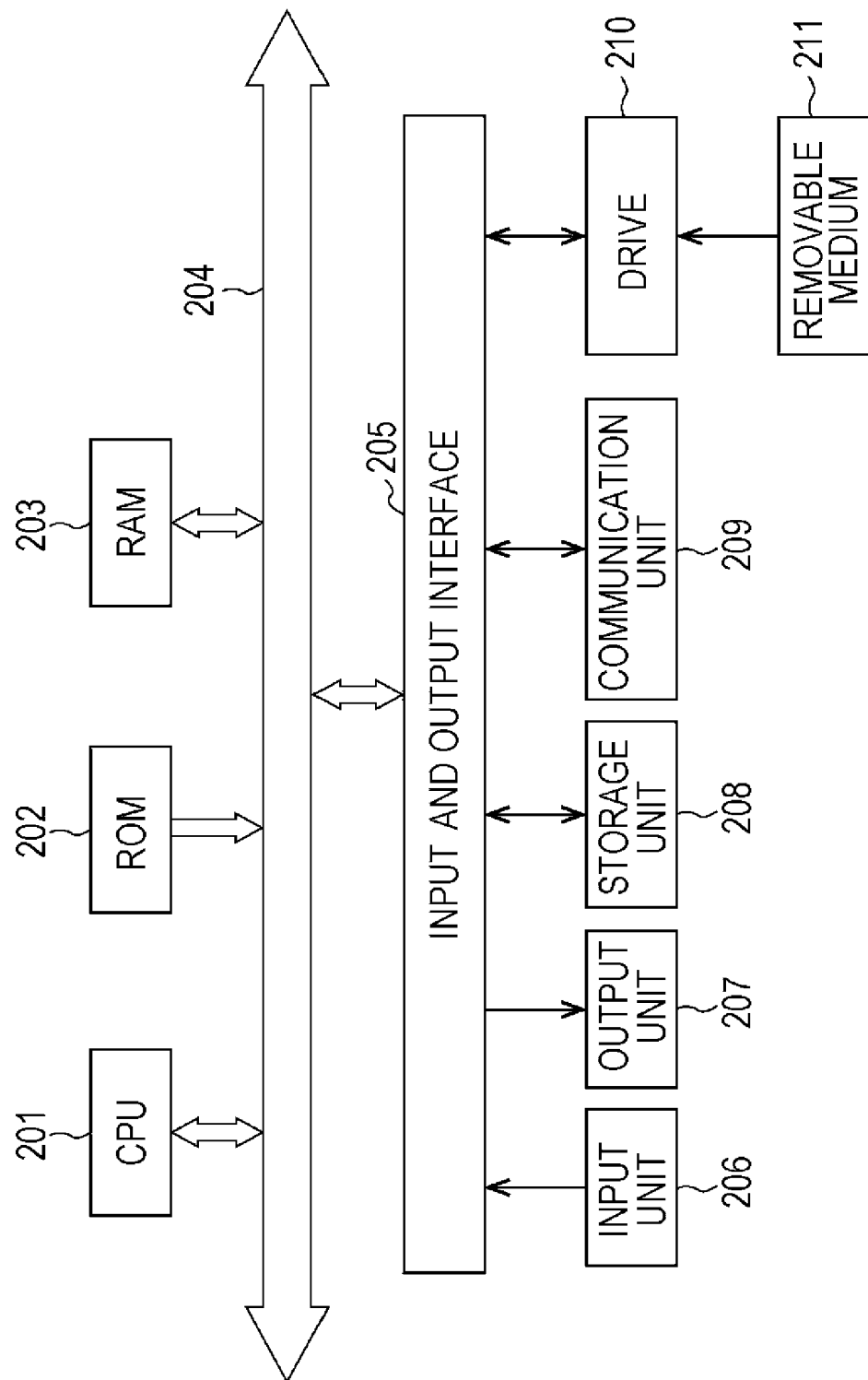
[Fig. 73]

FIG. 73



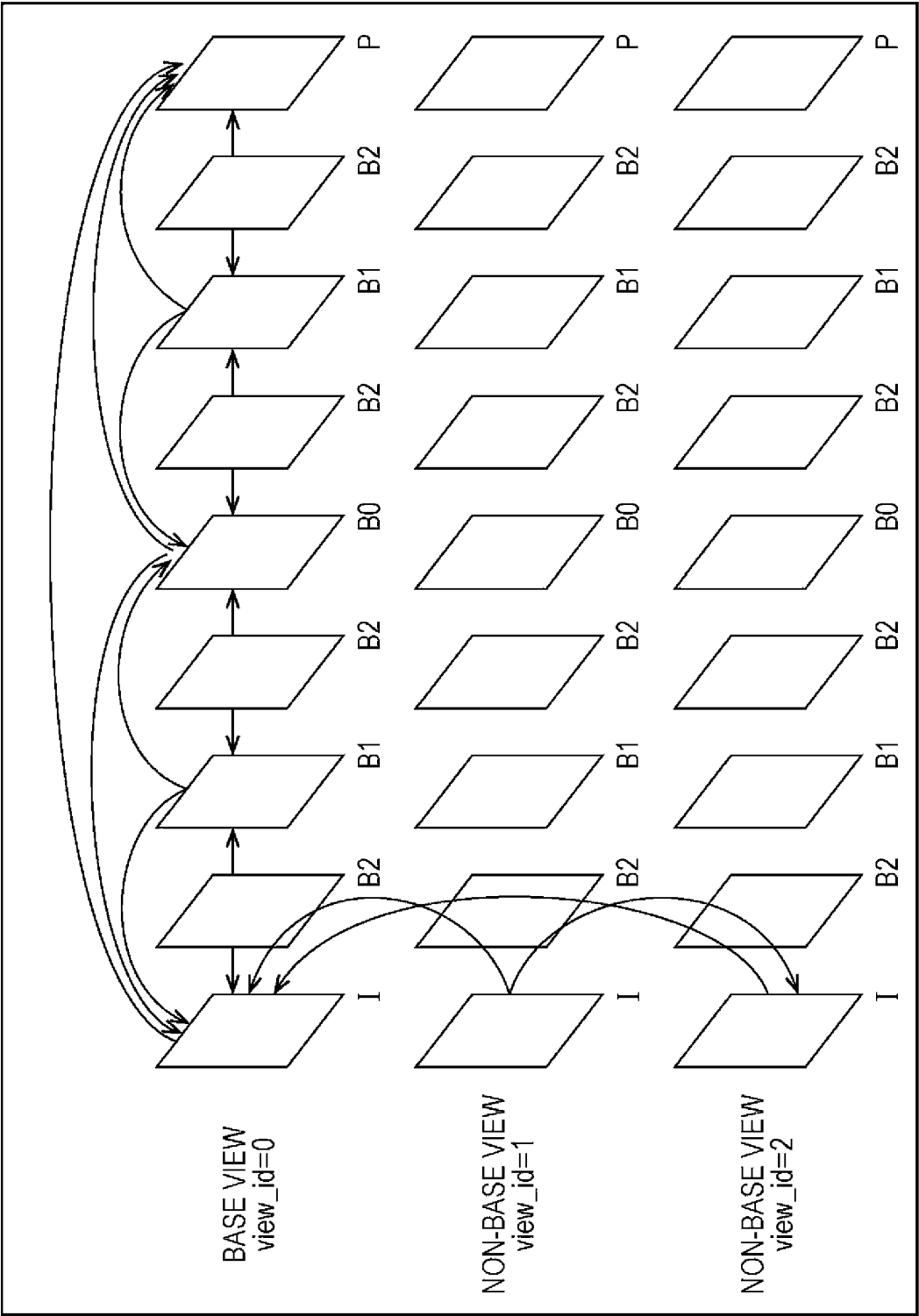
[Fig. 74]

FIG. 74



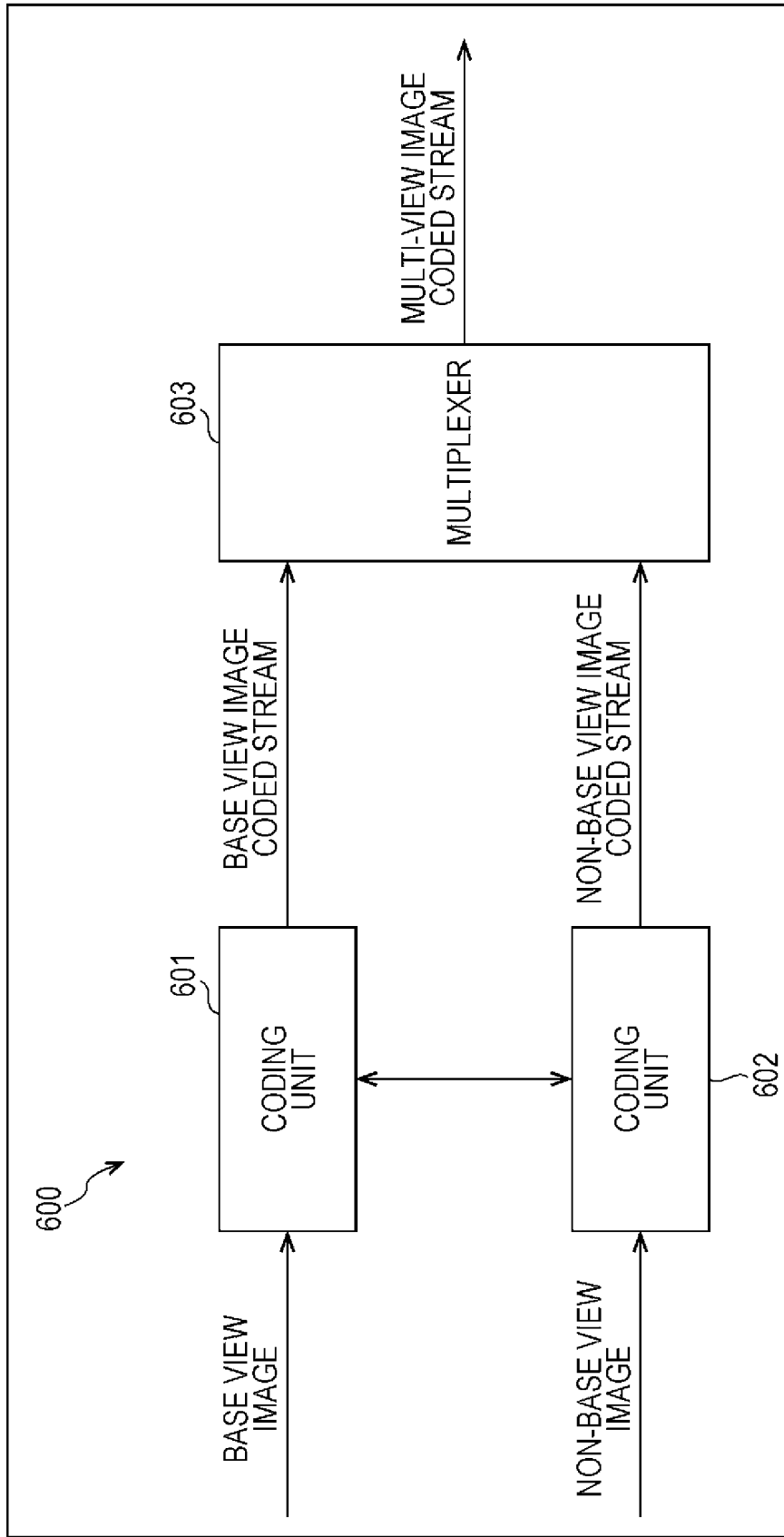
[Fig. 75]

FIG. 75



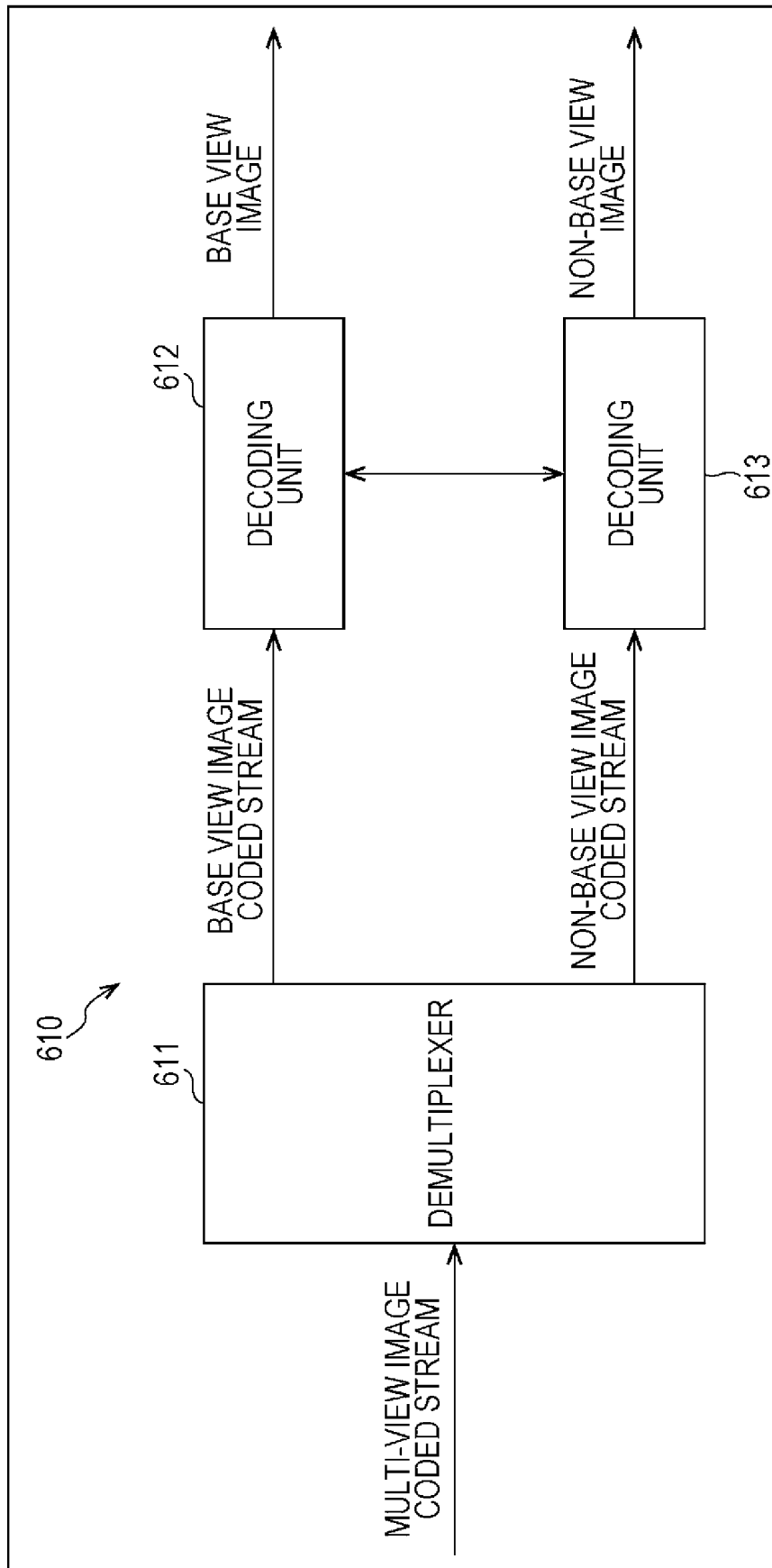
[Fig. 76]

FIG. 76

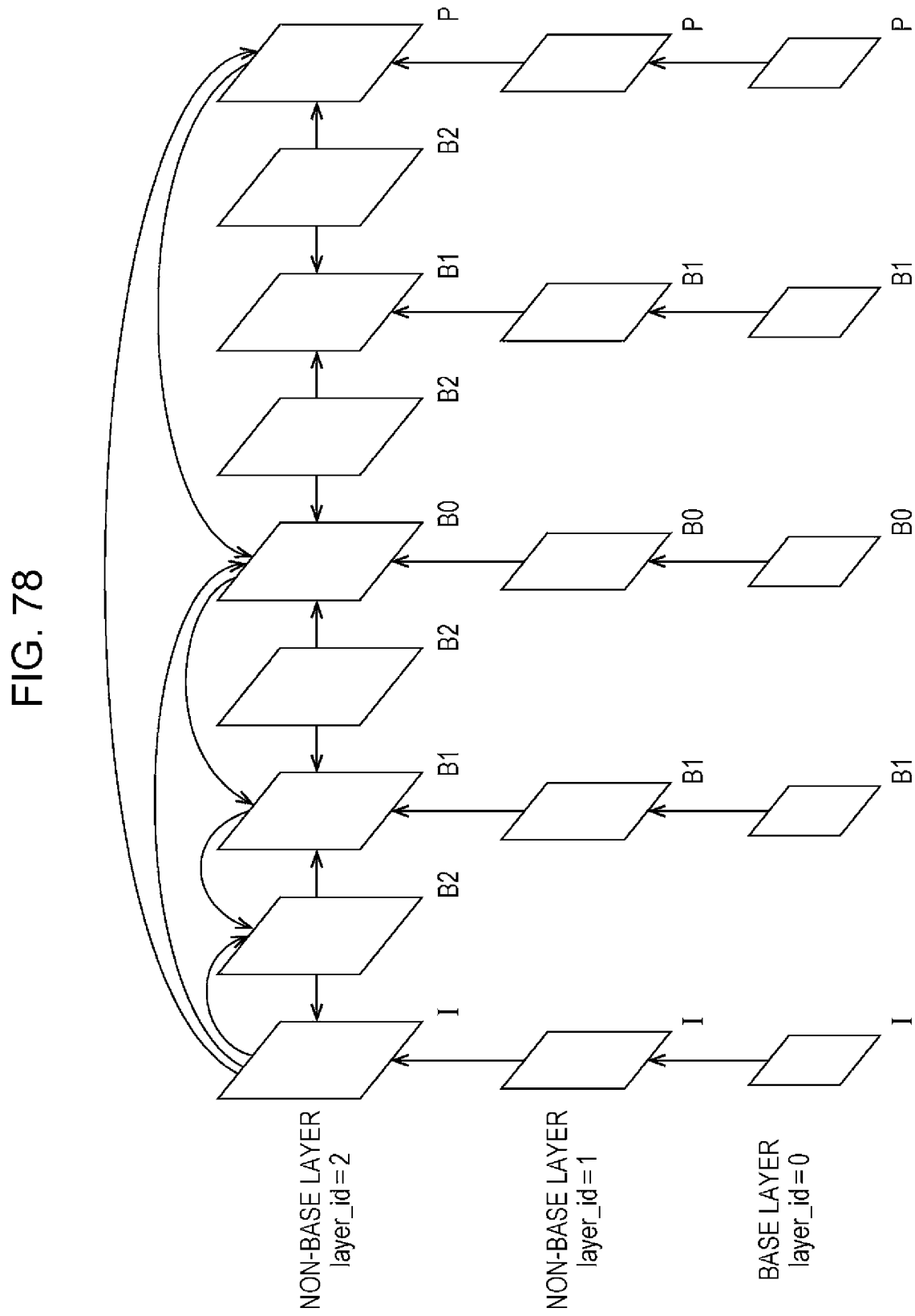


[Fig. 77]

FIG. 77

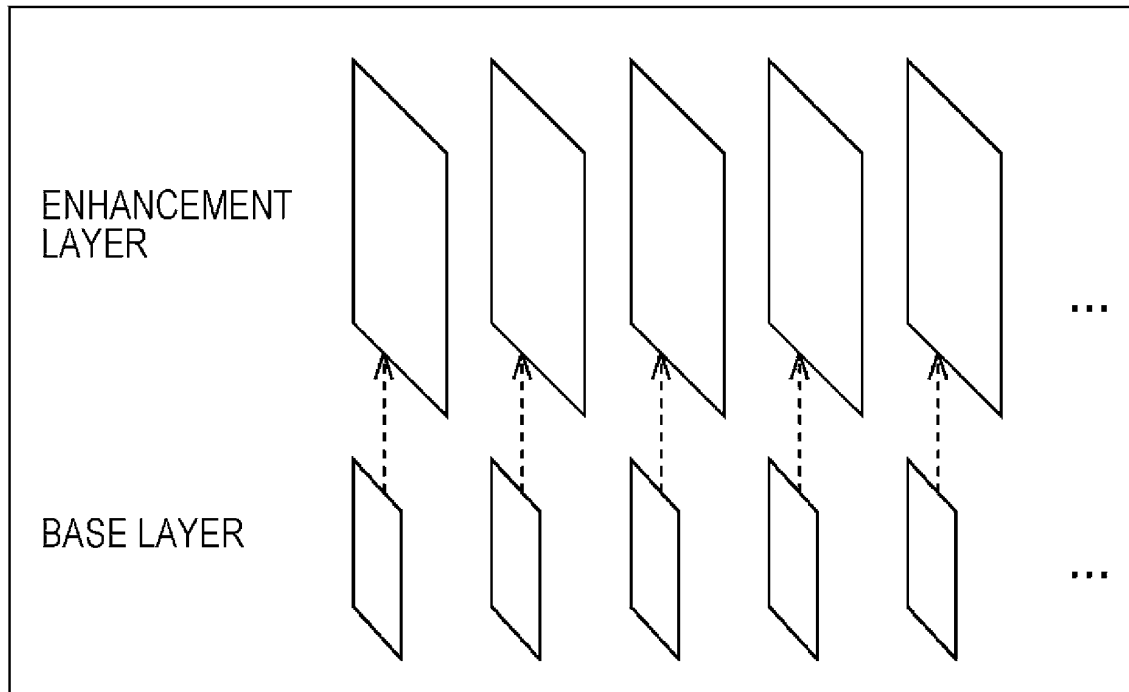


[Fig. 78]



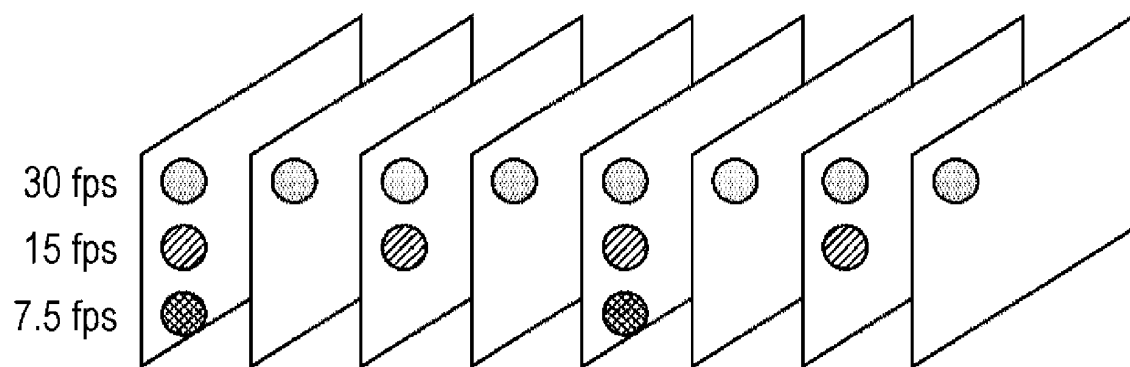
[Fig. 79]

FIG. 79



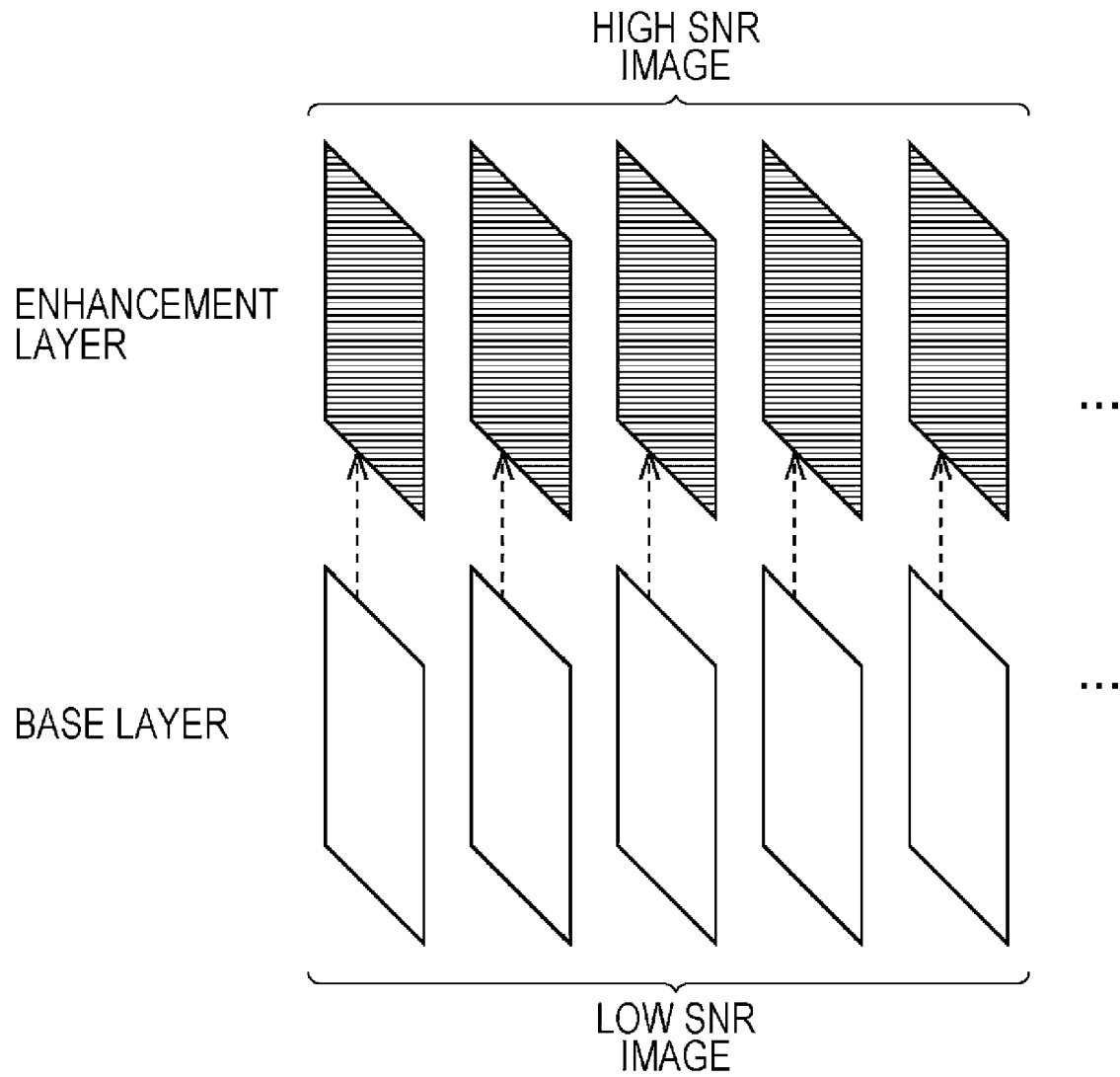
[Fig. 80]

FIG. 80

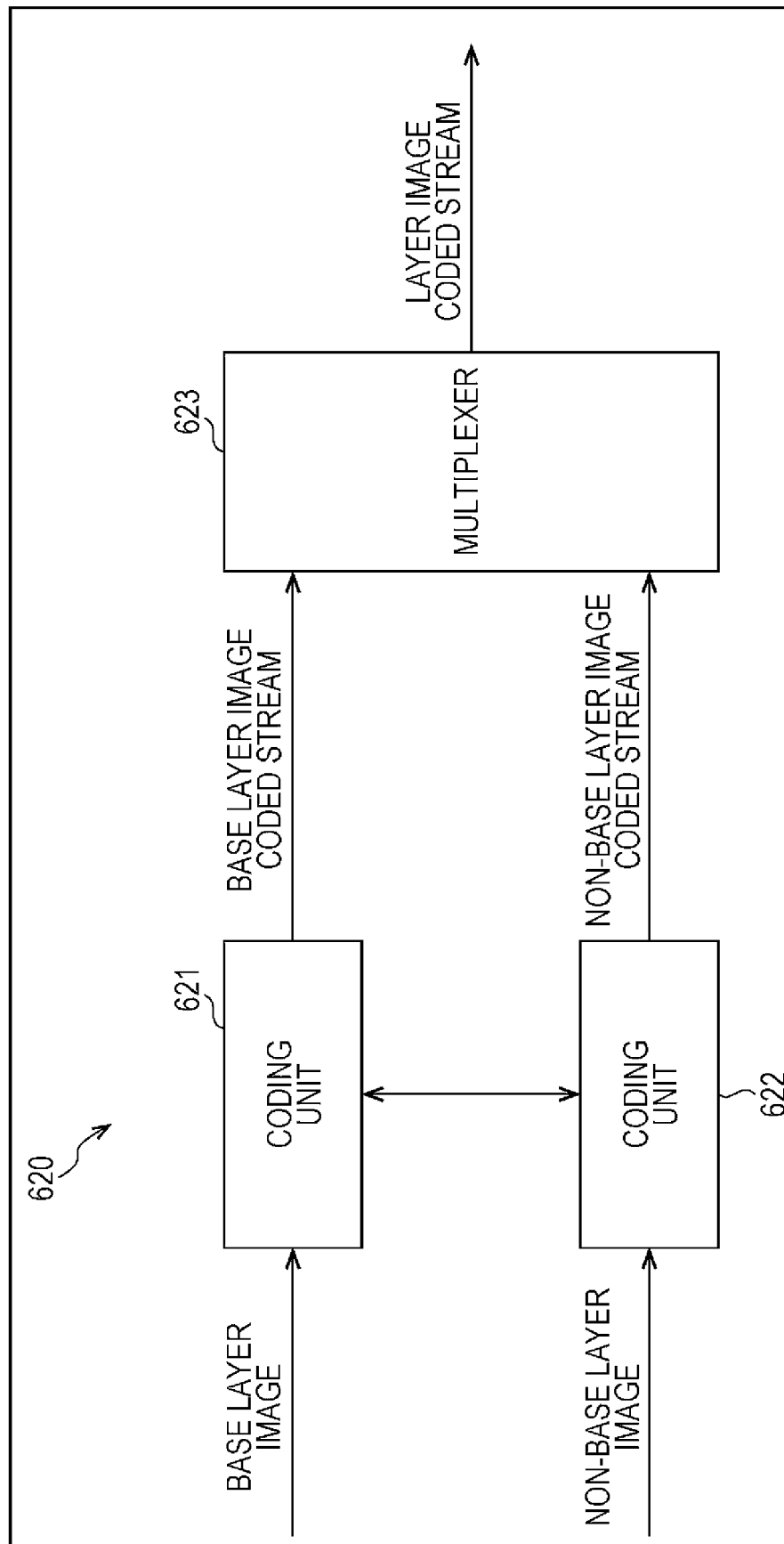


[Fig. 81]

FIG. 81

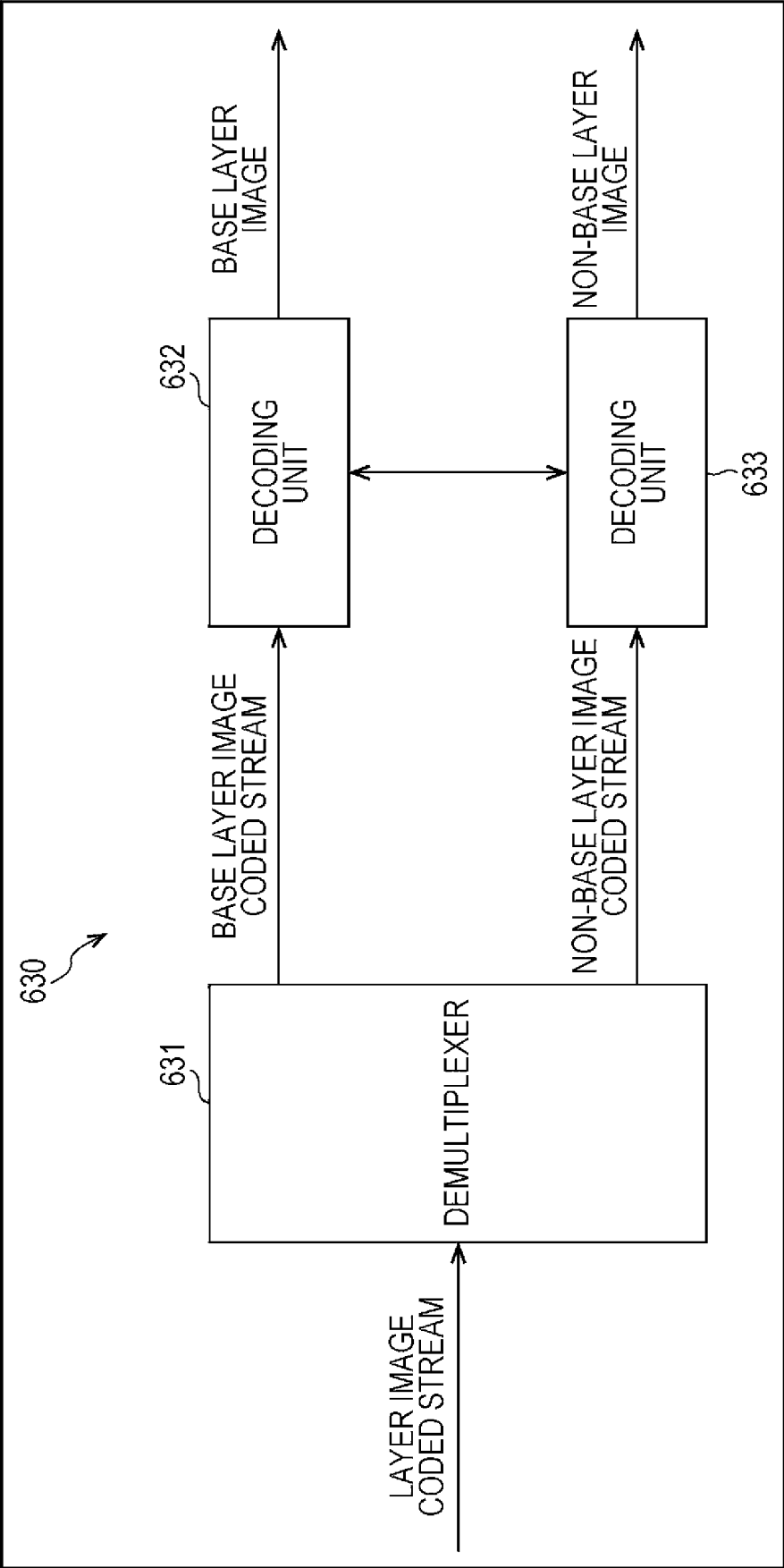


[Fig. 82]



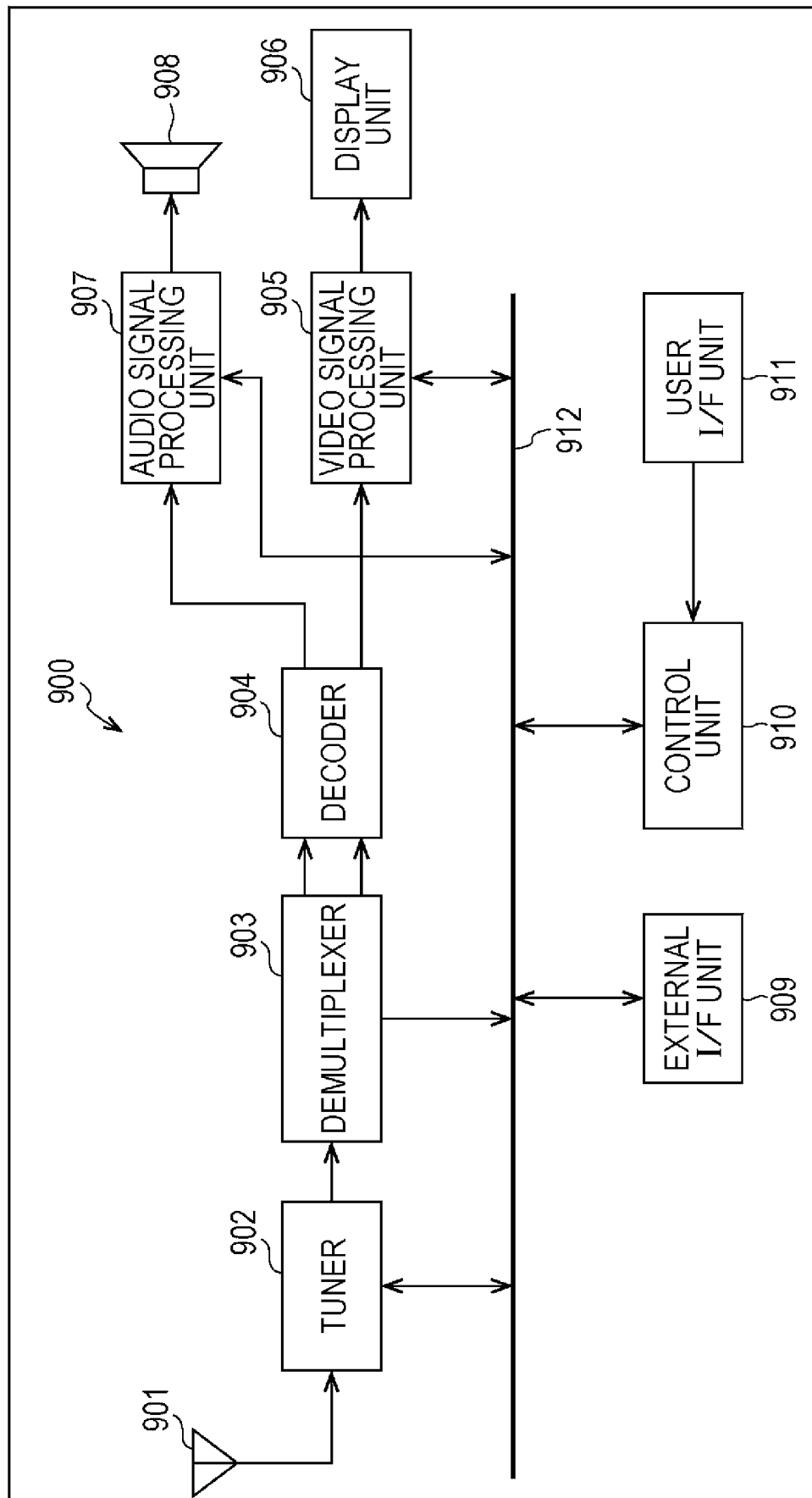
[Fig. 83]

FIG. 83



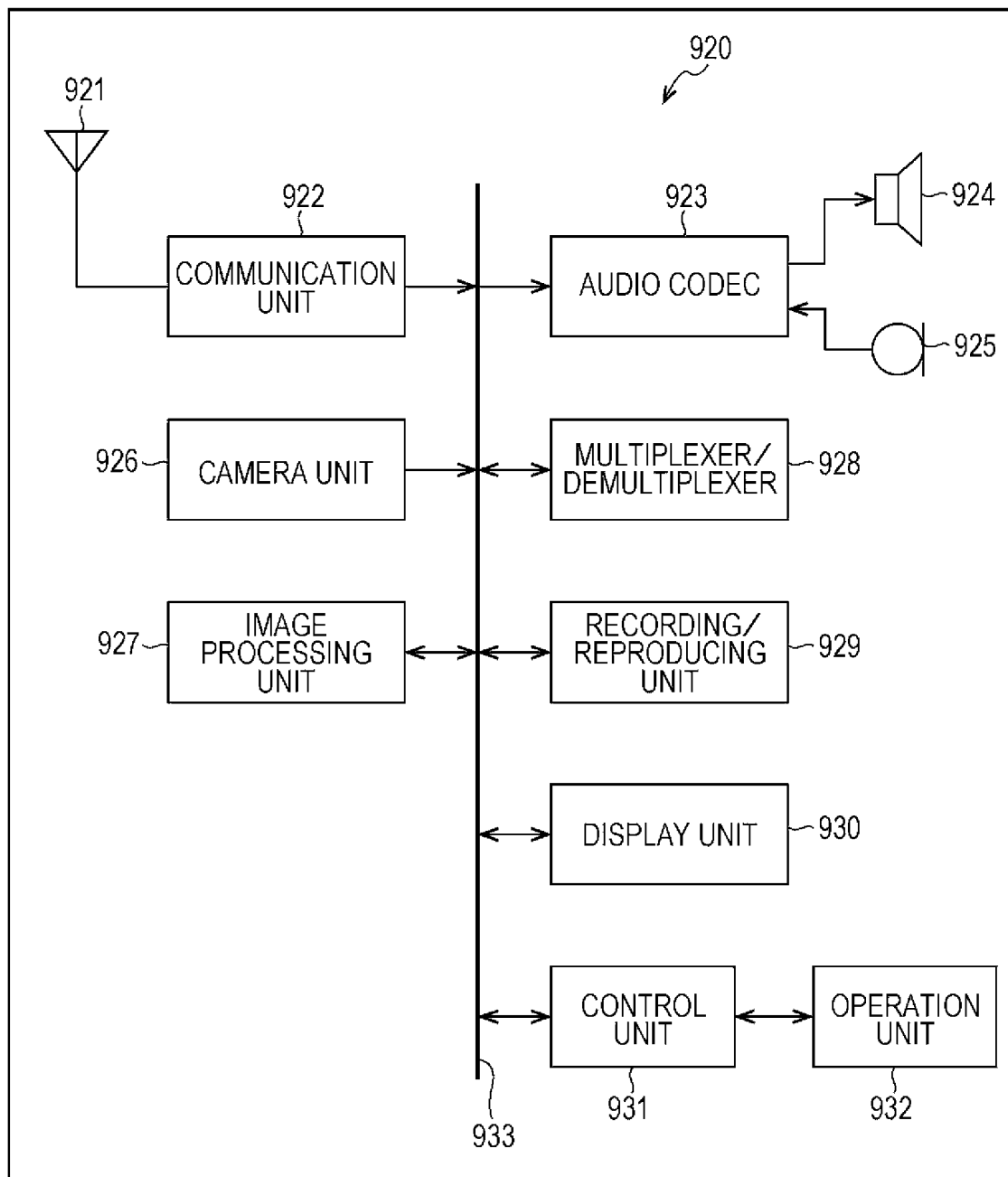
[Fig. 84]

FIG. 84



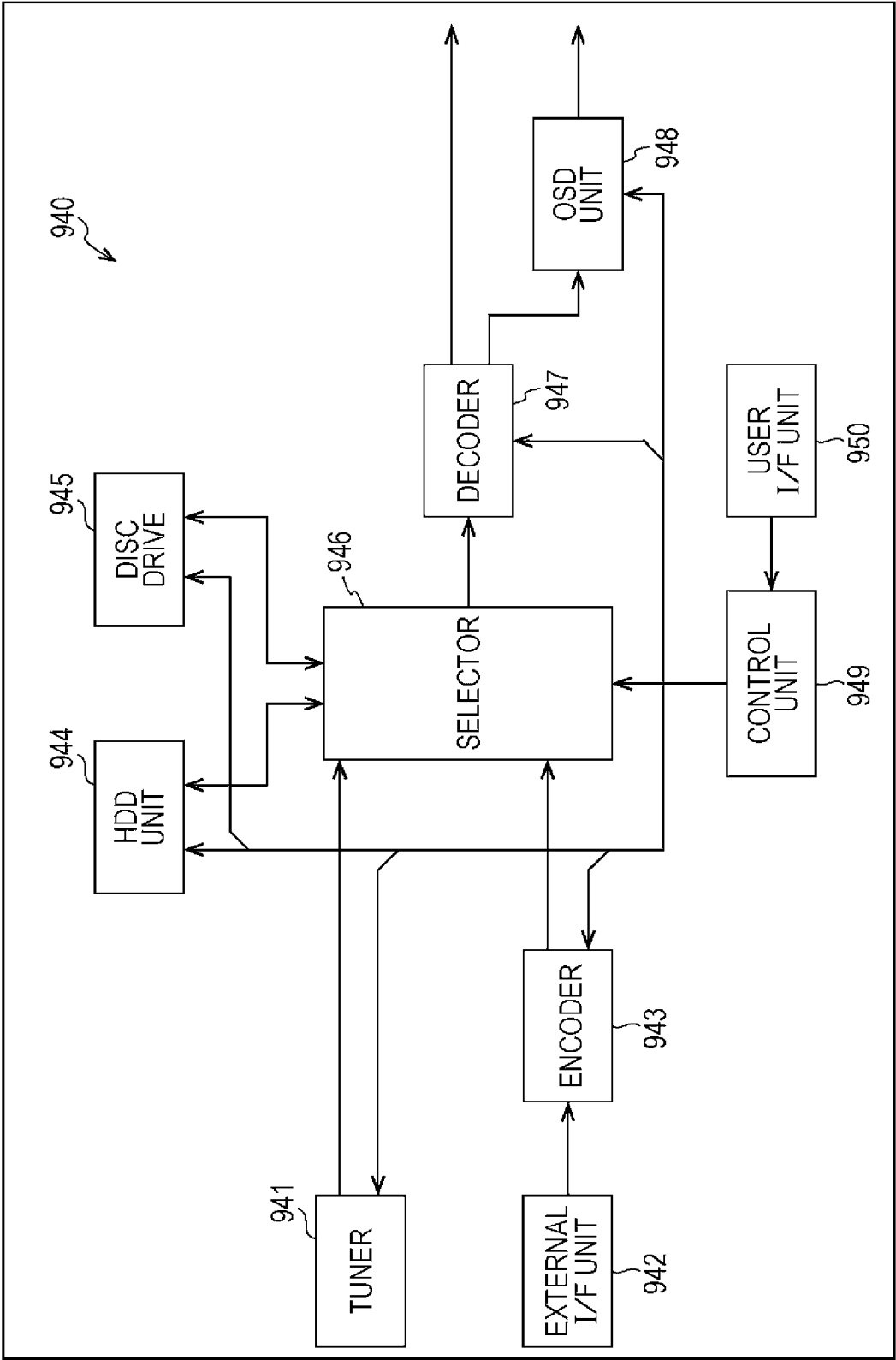
[Fig. 85]

FIG. 85



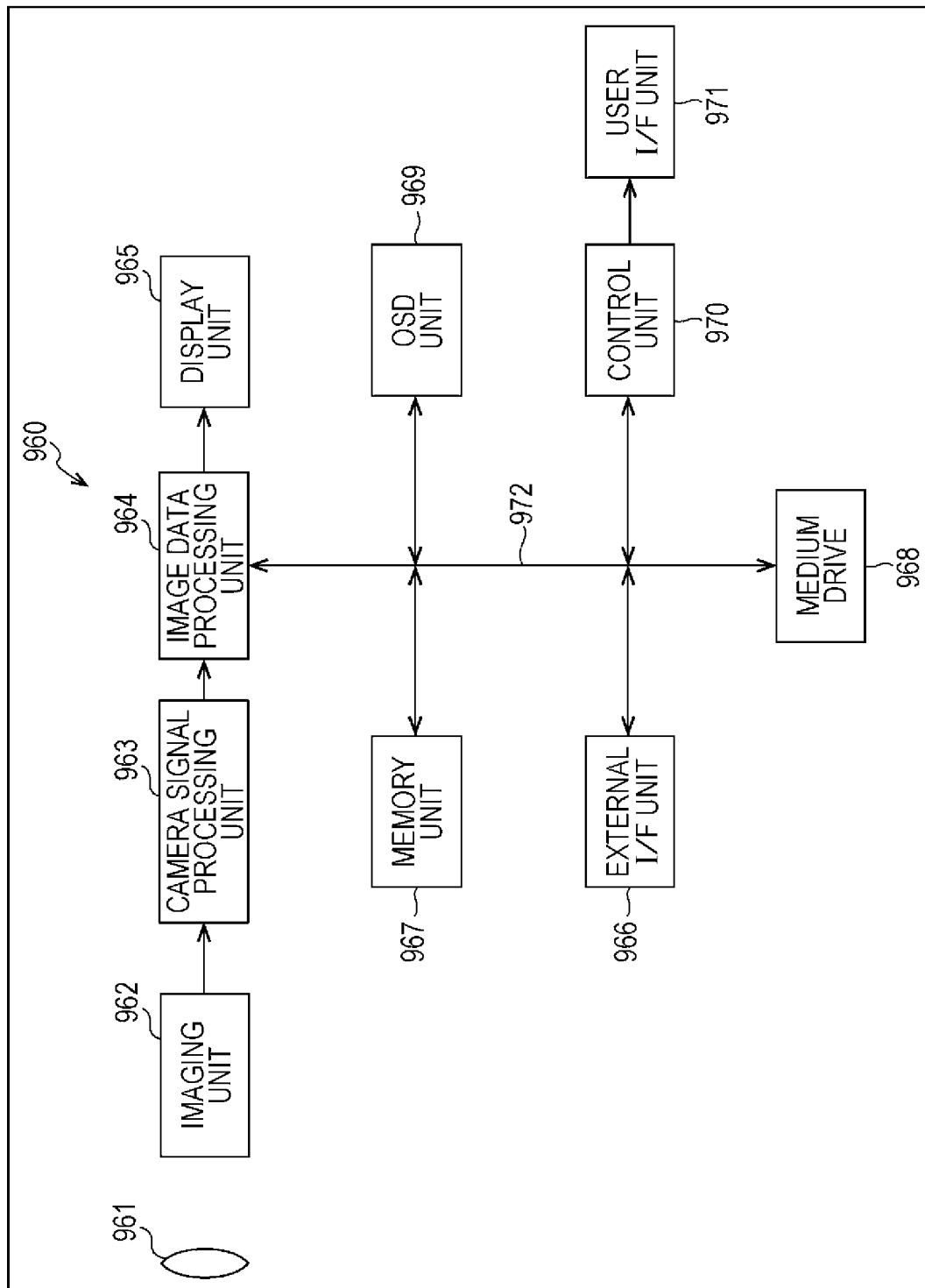
[Fig. 86]

FIG. 86

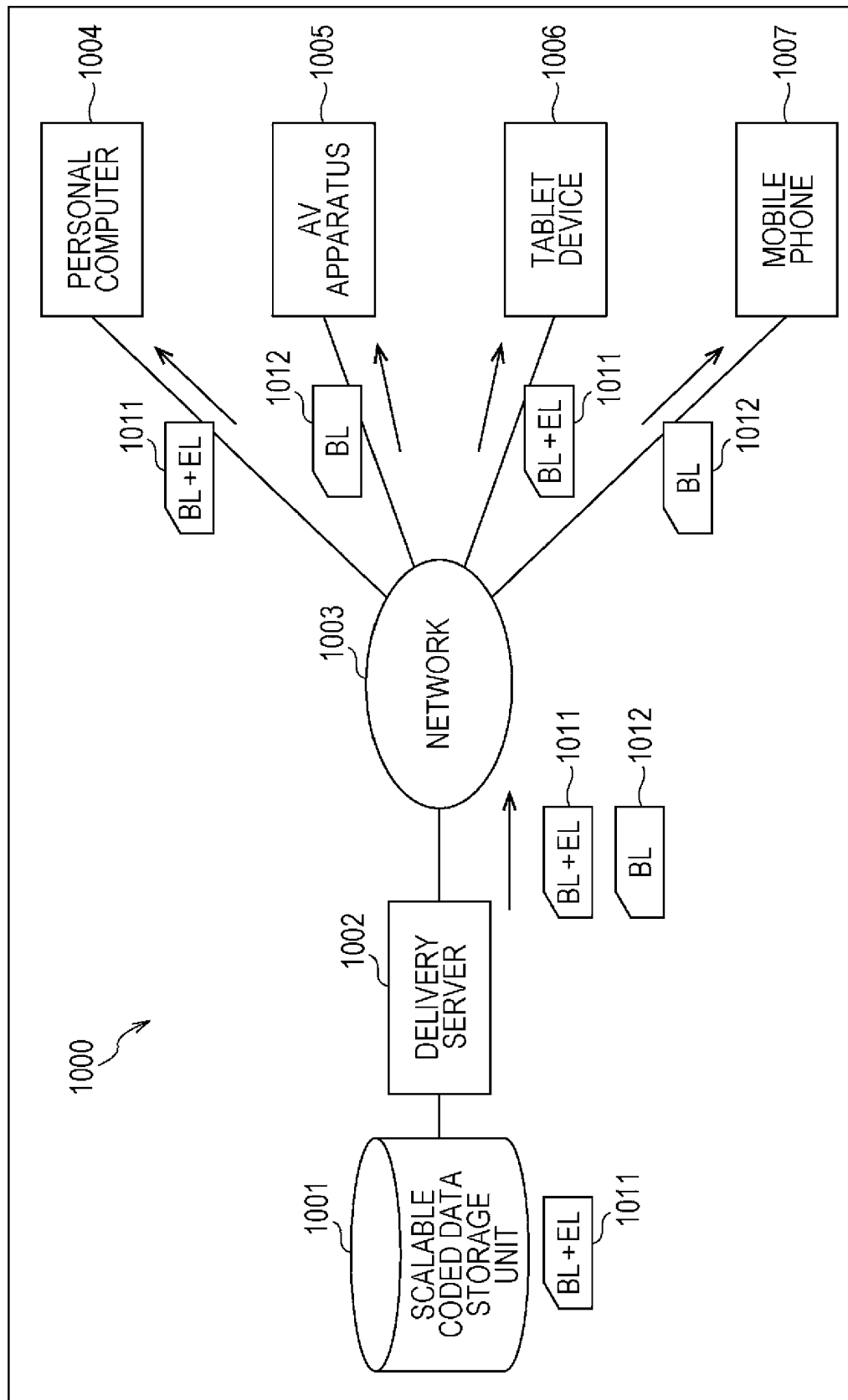


[Fig. 87]

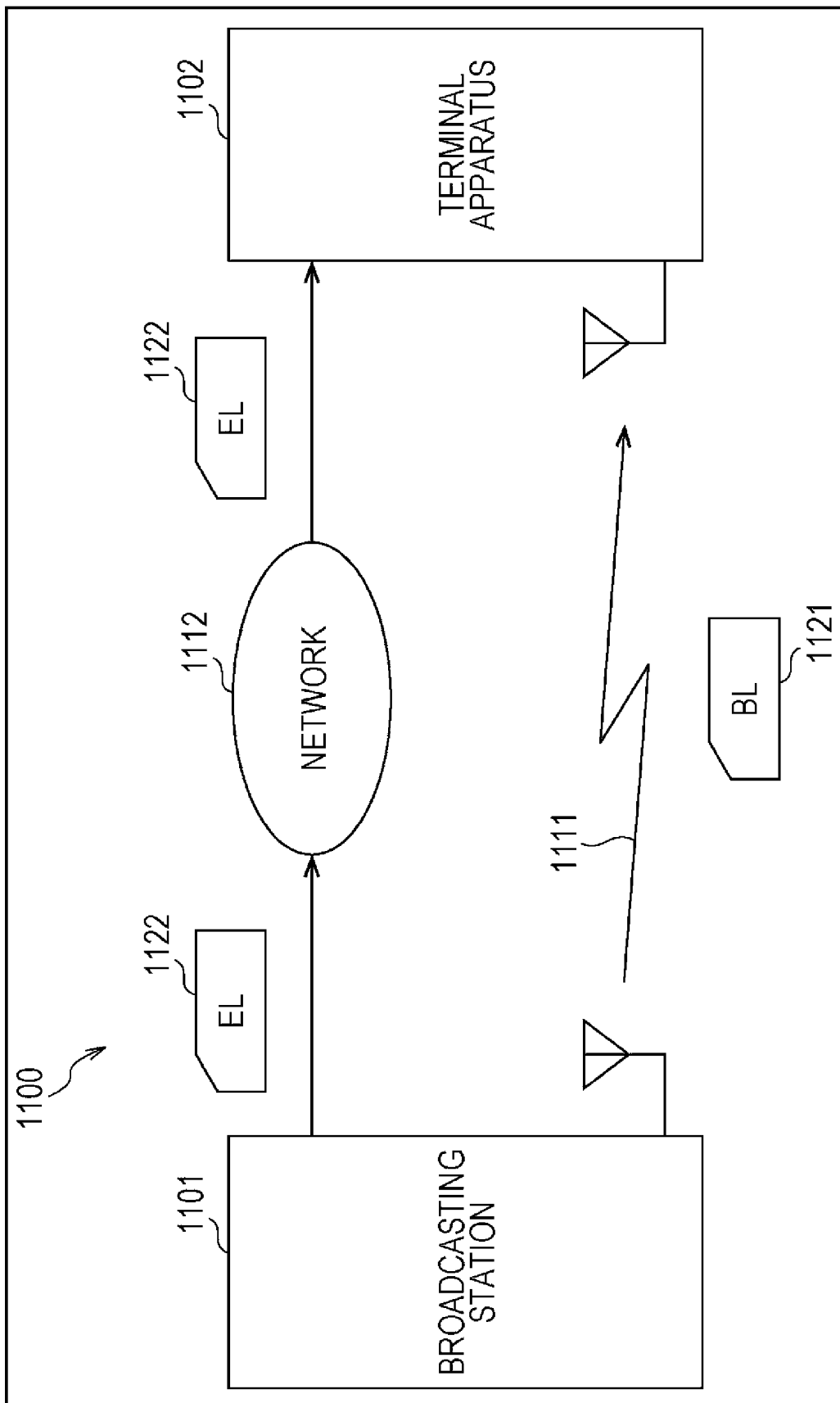
FIG. 87



[Fig. 88]

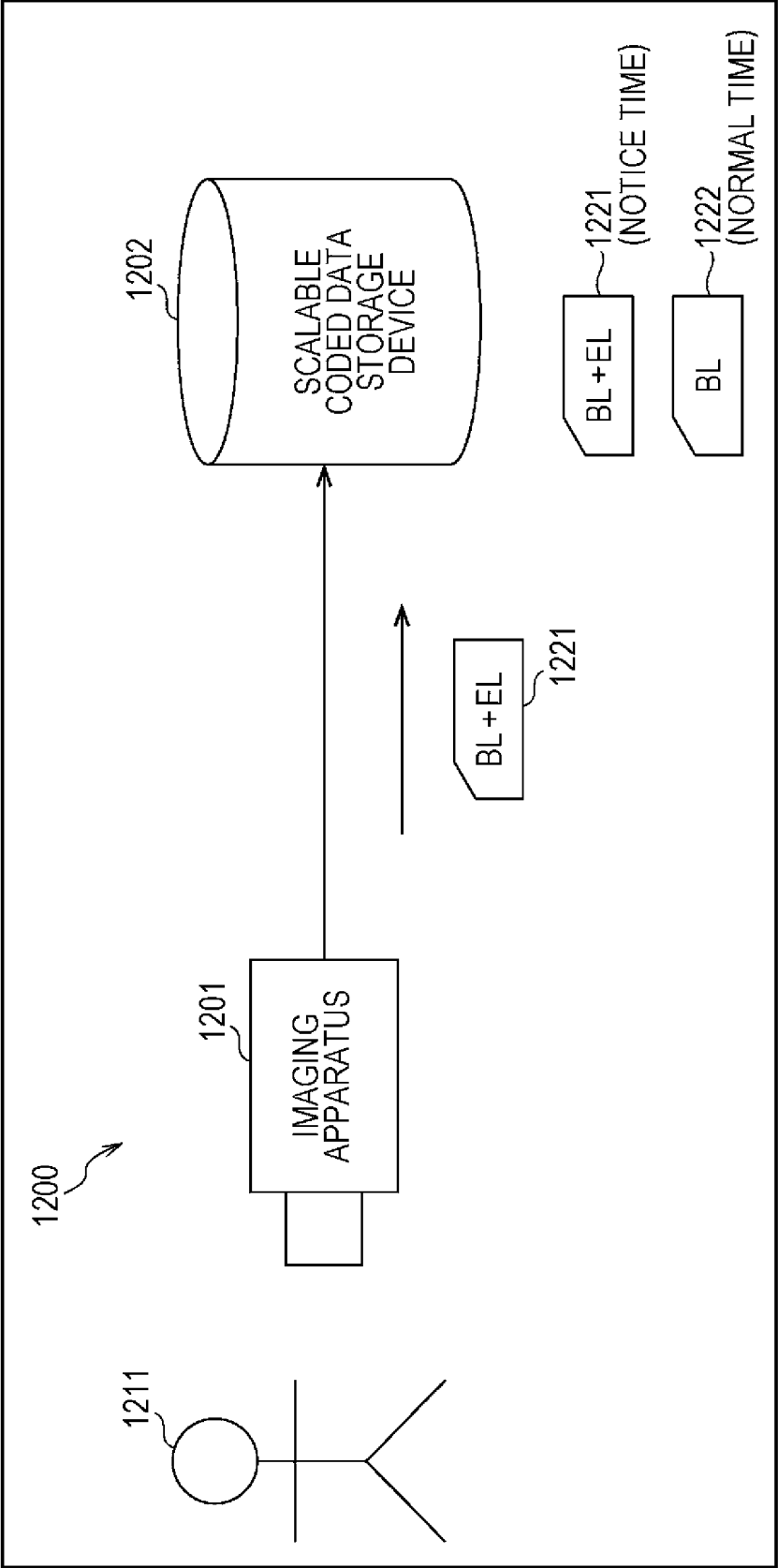


[Fig. 89]



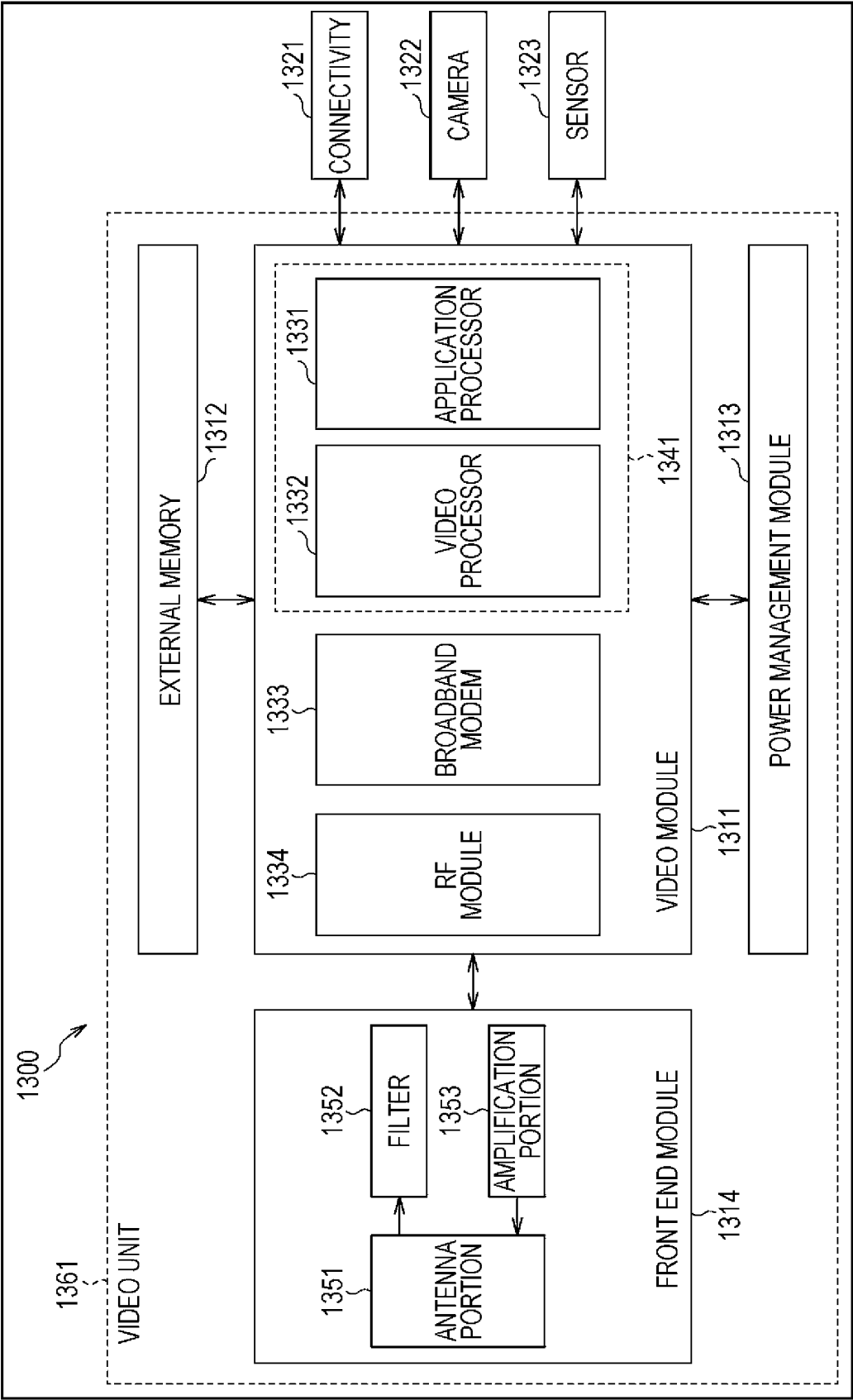
[Fig. 90]

FIG. 90

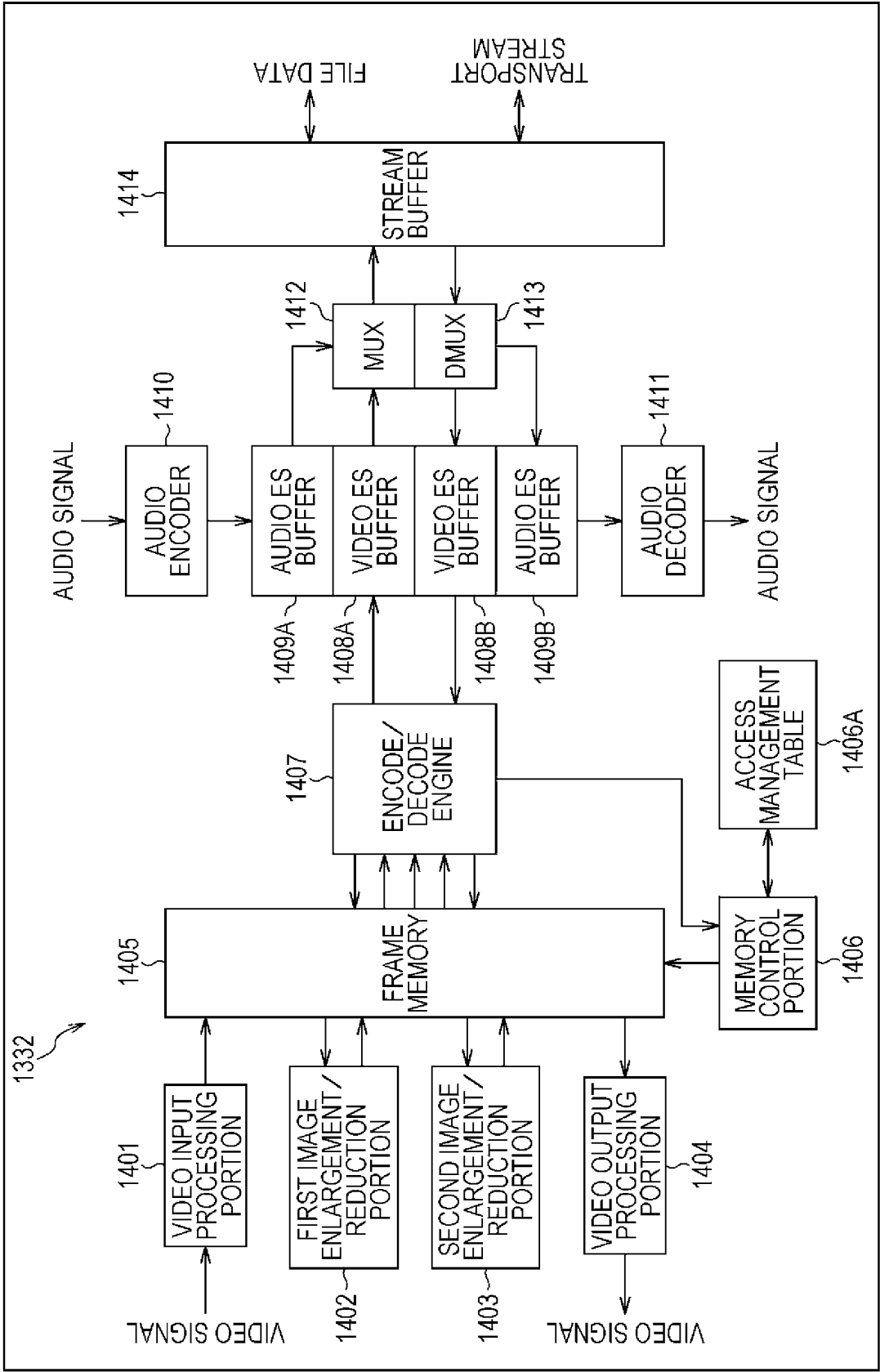


[Fig. 91]

FIG. 91



[Fig. 92]



[Fig. 93]

FIG. 93

