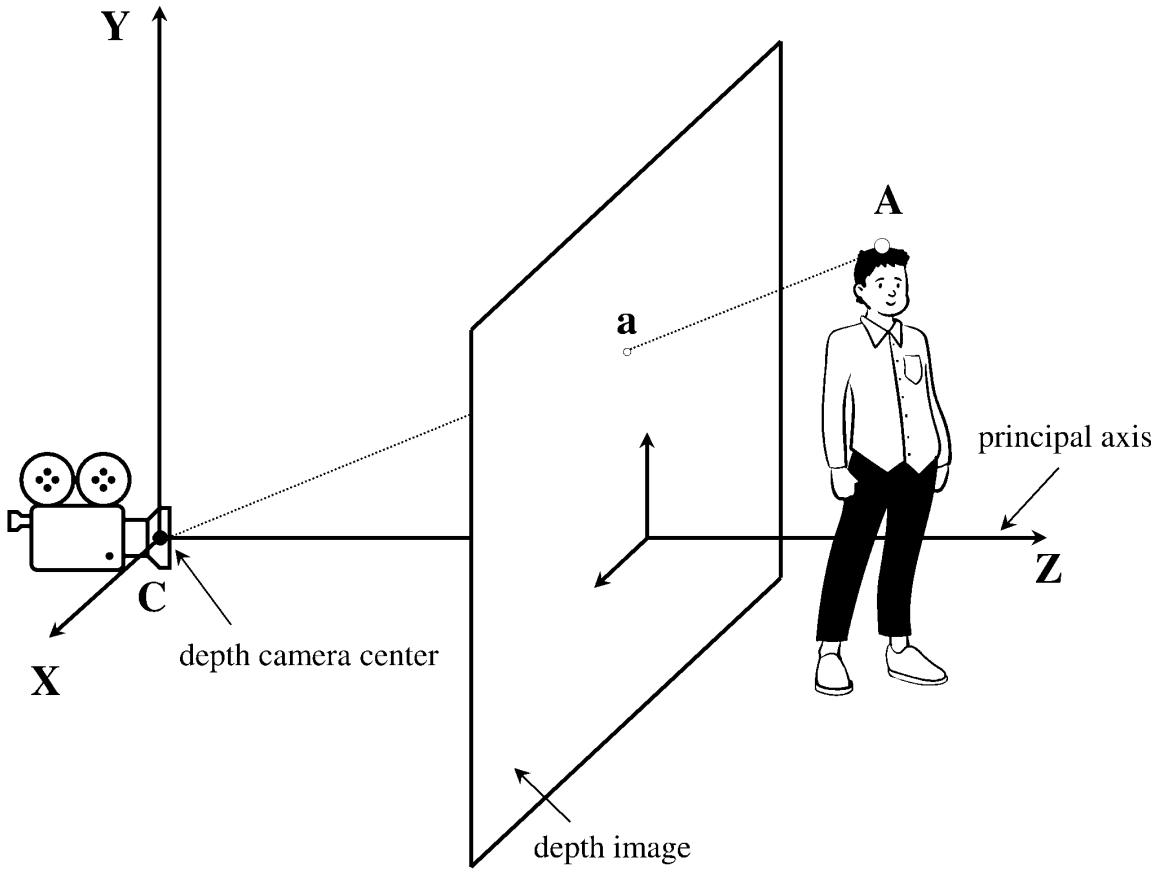
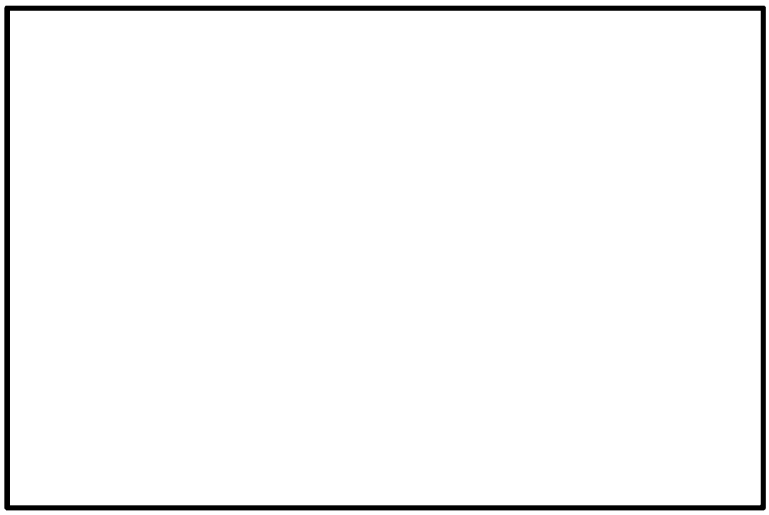


Fig. 1



Depth image



Color image

Fig. 2

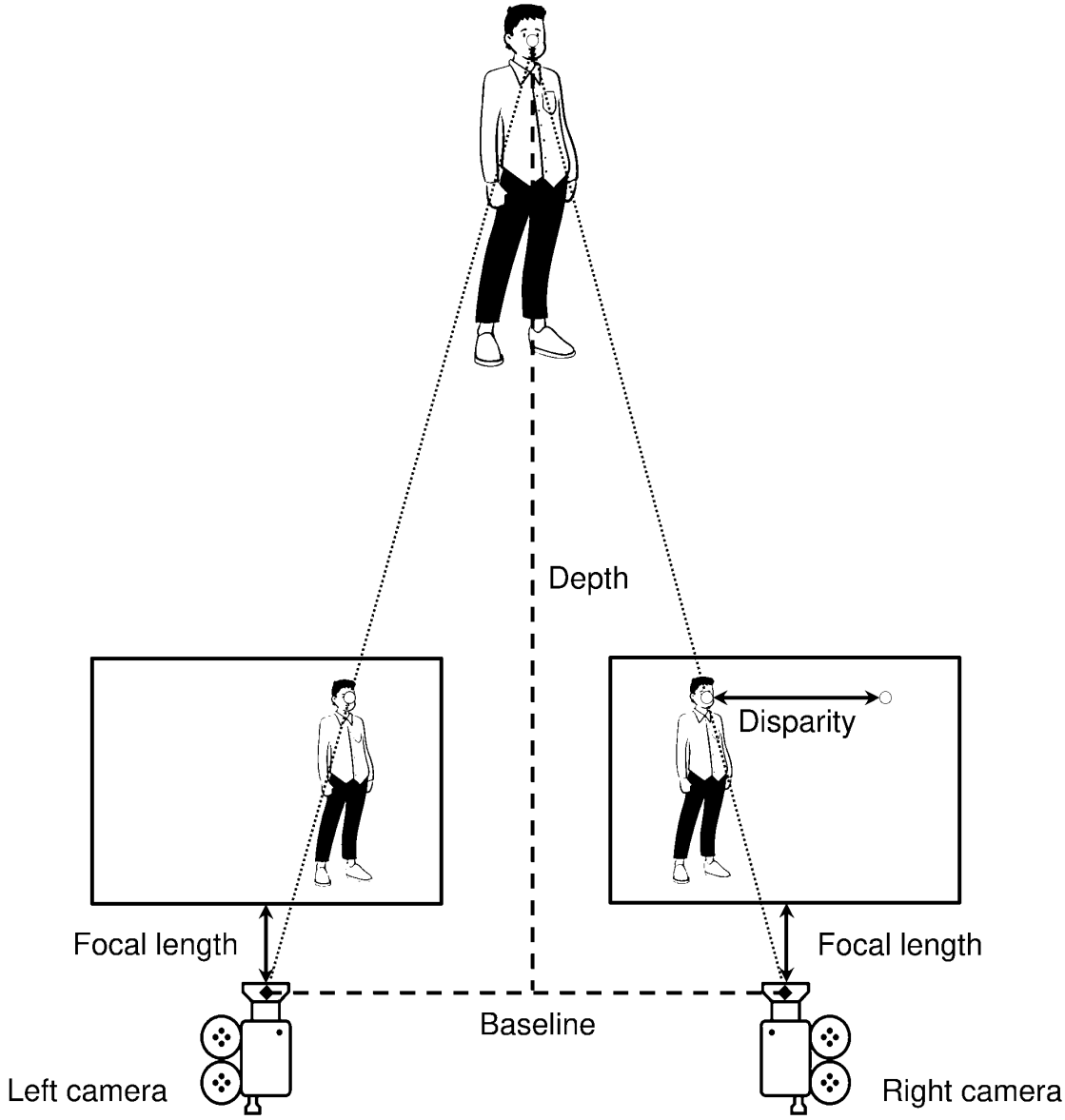

Depth
Disparity
Focal length
Focal length
Baseline
Left camera
Right camera

Fig. 3

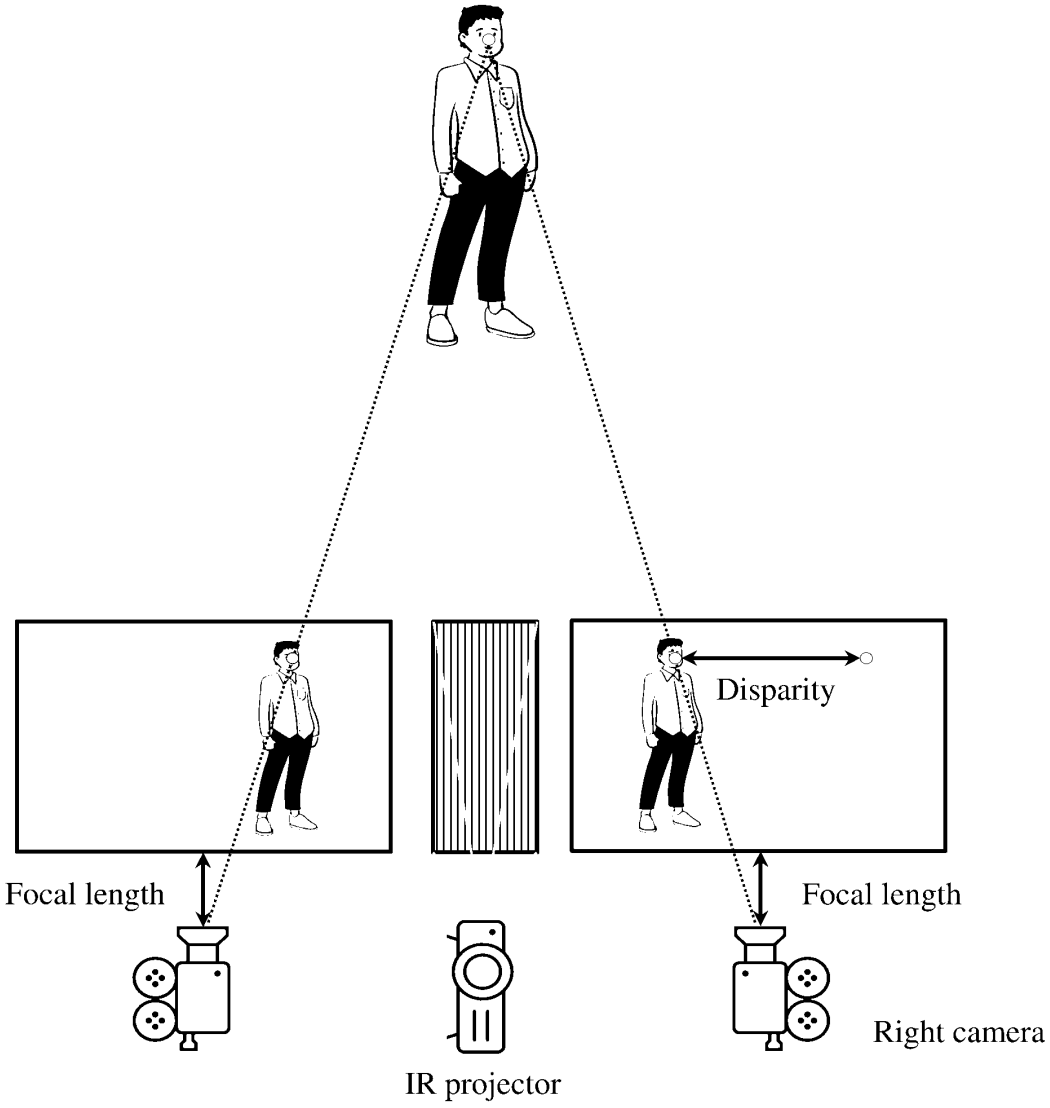

Disparity
Focal length
Focal length
Right camera
IR projector

Fig. 4

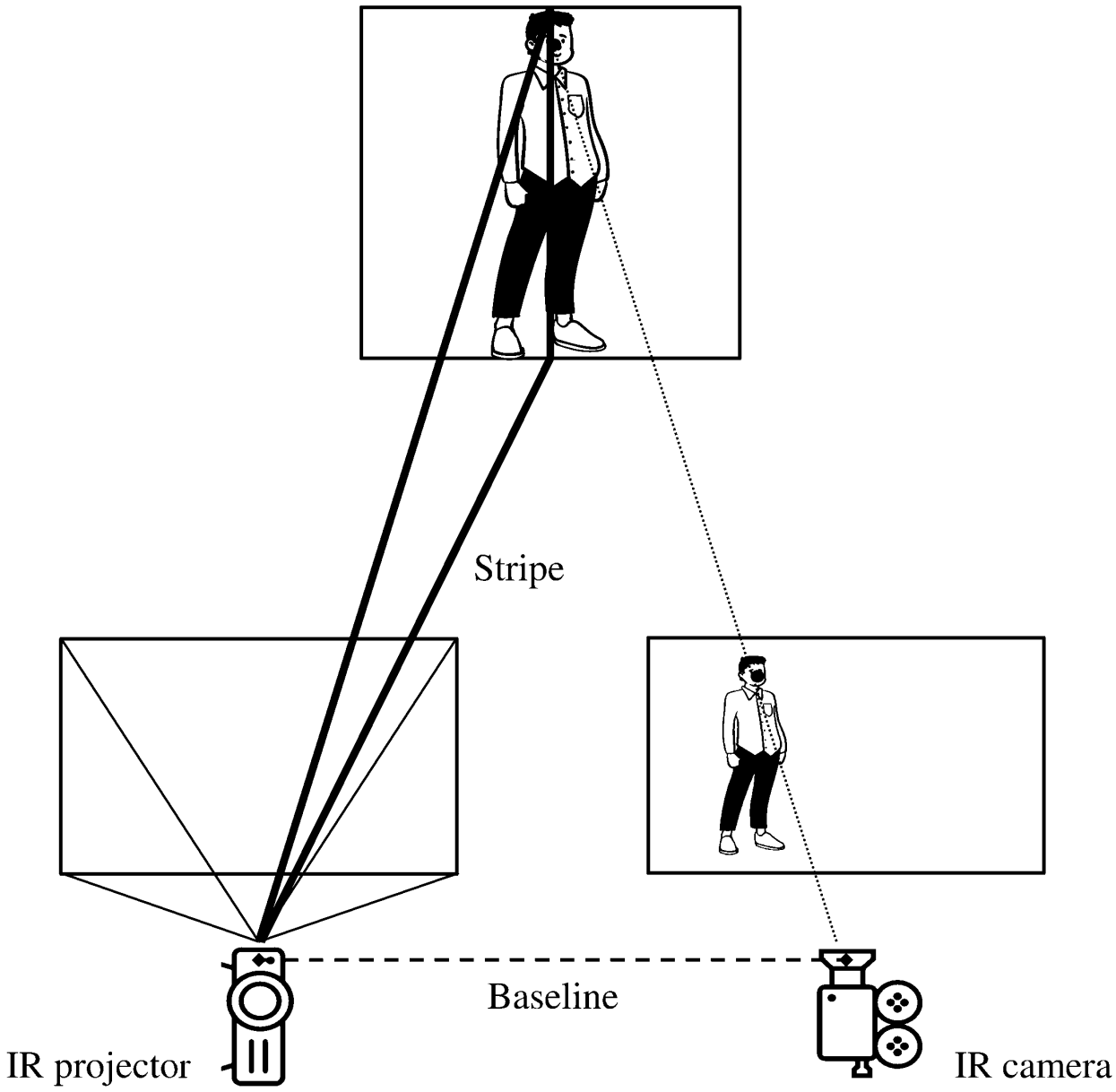


Fig. 5

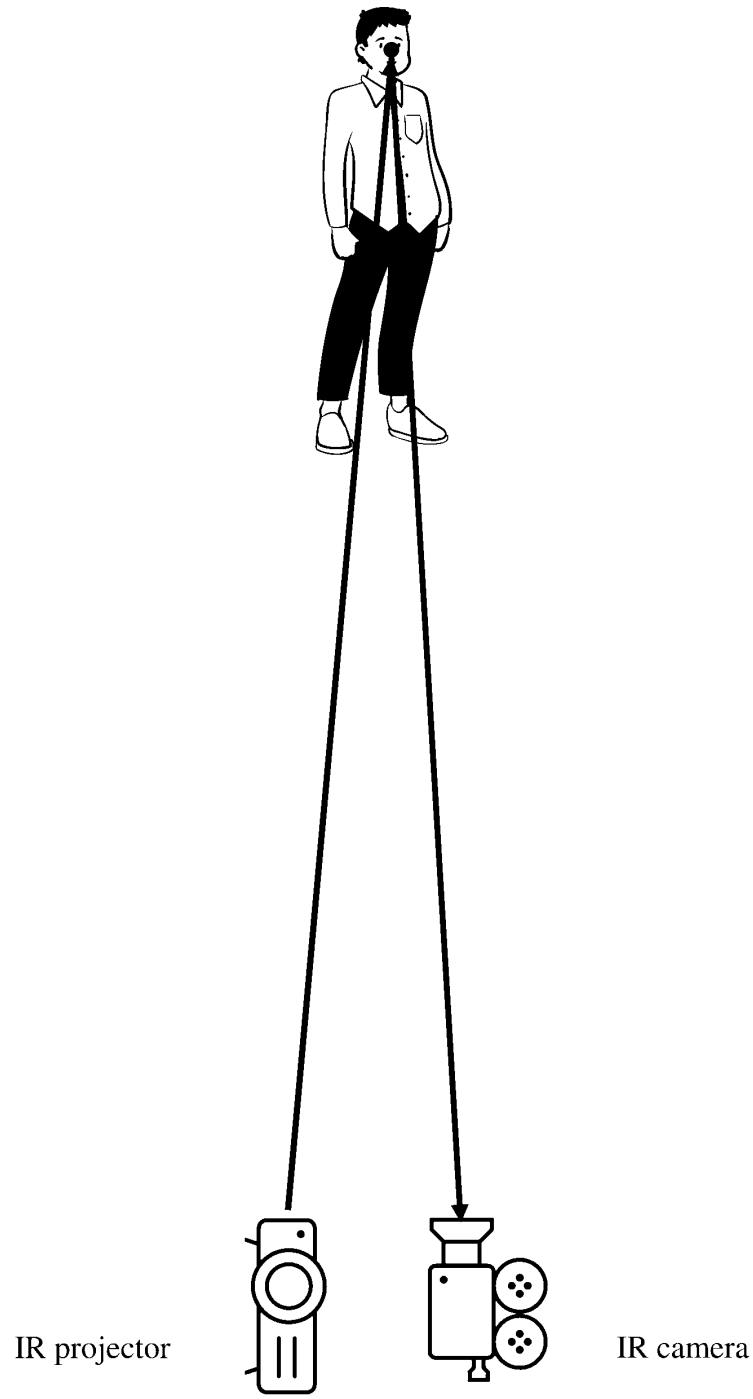


Fig. 6

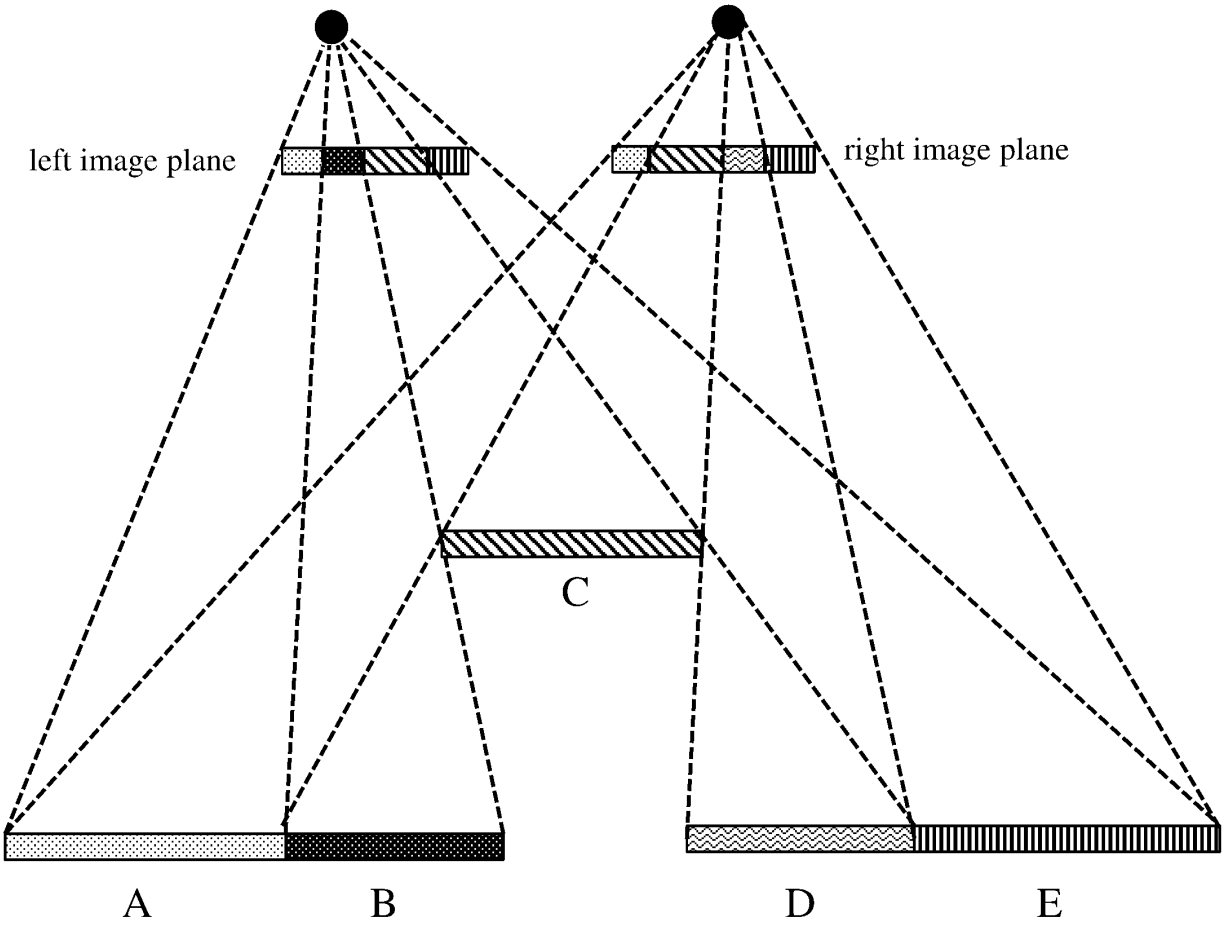


Fig. 7

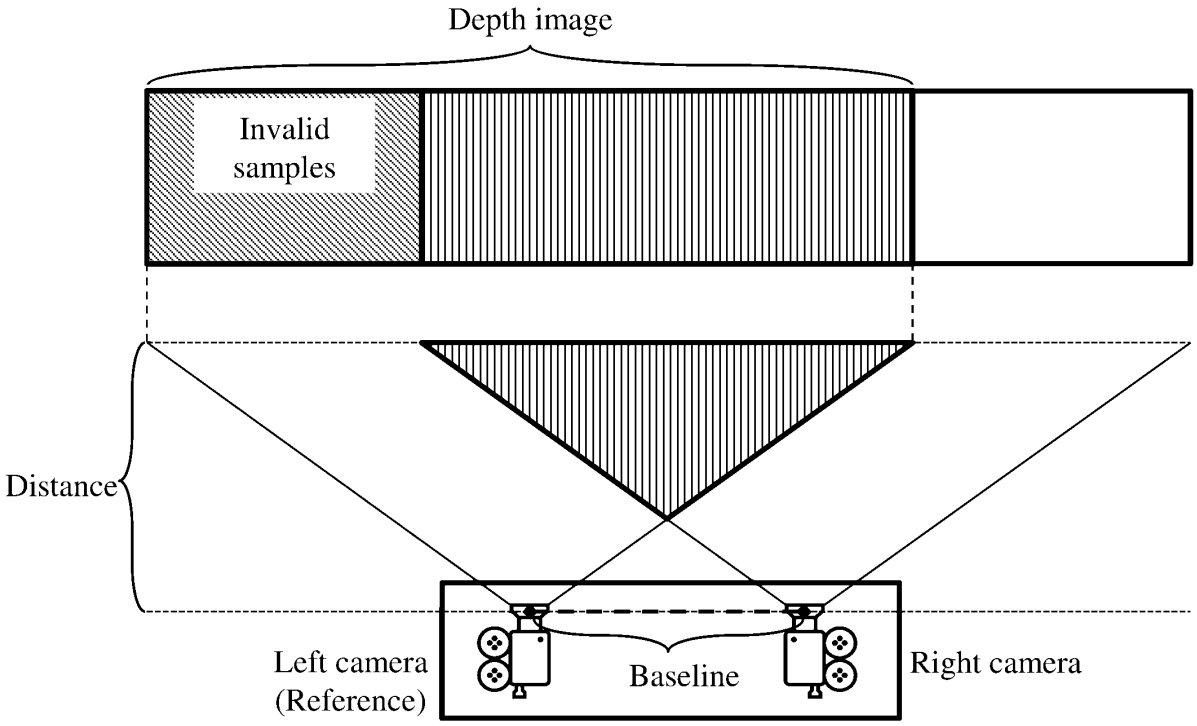


Fig. 8

0	0	0	13
0	0	13	13
0	13	13	13
0	0	13	13

			13
		13	13
	13	13	13
		13	13

Fig. 9

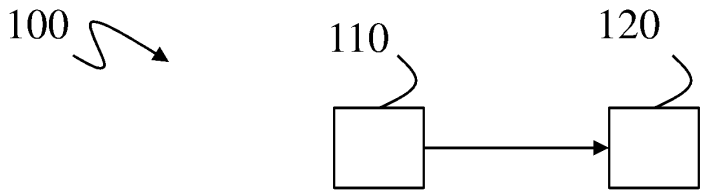


Fig. 10

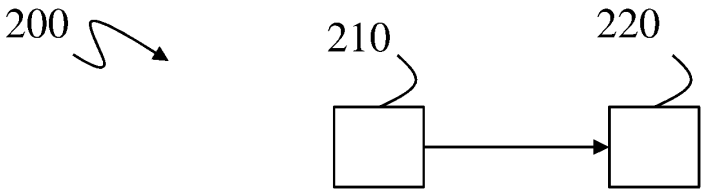


Fig. 11

nal_unit(NumBytesInNalUnit) {	Descriptor
nal_unit_header( )	
NumBytesInRbsp = 0	
for(i = 2; i < NumBytesInNalUnit; i++)	
if(i + 2 < NumBytesInNalUnit && next_bits(24) = = 0x000003) {	
rbsp_byte[NumBytesInRbsp++]	b(8)
rbsp_byte[NumBytesInRbsp++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRbsp++]	b(8)
}	

Fig. 12

sei_payload(payloadType, payloadSize) {	Descriptor
if(nal_unit_type = = PREFIX_SEI_NUT)	
if(payloadType = = 0)	
buffering_period(payloadSize)	
else if(payloadType = = 1)	
pic_timing(payloadSize)	
…	
else if(payloadType = = 133)	
scalable_nesting(payloadSize)	
else if(payloadType = = 134)	
invalid_depth_samples (payloadSize)	
else /* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 */	
reserved_message(payloadSize)	
…	
}	

Fig. 13

invalid_depth_samples_corner_band (payloadSize) {	Descriptor
idscb_cancel_flag	u(1)
if(! idscb_cancel_flag) {	
idscb_band_side	u(2)
idscb_band_length	ue(v)
}	
}	

Fig. 14

invalid_depth_samples_bands (payloadSize) {	Descriptor
idsb_cancel_flag	u(1)
if(! idsb_cancel_flag) {	
idsb_number_of_bands	u(4)
for(i=0; i < idsb_number_of_bands; i++) {	
idsb_band_orientation[i]	u(1)
idsb_band_coordinate[i]	ue(v)
idsb_band_length[i]	ue(v)
}	
}	
}	

Fig. 15

invalid_depth_samples_binary_mask(payloadSize) {	Descriptor
idsbm_cancel_flag	u(1)
if(! idsbm_cancel_flag) {	
idsbm_mask_height	ue(v)
idsbm_mask_width	ue(v)
for(i=0; i < idsbm_mask_height; i++)	
for(j=0; j < idsbm_mask_width; j++)	
idsbm_mask_value[i][j]	u(1)
}	
}	

Fig. 16

invalid_depth_samples_binary_mask(payloadSize) {	Descriptor
idsbm_cancel_flag	u(1)
if(! idsbm_ cancel_flag) {	
for(i=0; i < pps_pic_height_in_luma_samples; i++)	
for(j=0; j < pps_pic_width_in_luma_samples; j++)	
idsbm_ value[i][j]	u(1)
}	
}	

Fig. 17

scalability_dimension_info(payloadSize) {	Descriptor
sdi_max_layers_minus1	u(6)
sdi_multiview_info_flag	u(1)
sdi_auxiliary_info_flag	u(1)
if(sdi_multiview_info_flag || sdi_auxiliary_info_flag) {	
if(sdi_multiview_info_flag)	
sdi_view_id_len_minus1	u(4)
for(i = 0; i <= sdi_max_layers_minus1; i++) {	
sdi_layer_id[i]	u(6)
if(sdi_multiview_info_flag)	
sdi_view_id_val[i]	u(v)
if(sdi_auxiliary_info_flag)	
sdi_aux_id[i]	u(8)
if(sdi_aux_id[i] > 0) {	
sdi_num_associated_primary_layers_minus1[i]	u(6)
for(j = 0; j <= sdi_num_associated_primary_layers_minus1[i]; j++)	
sdi_associated_primary_layer_idx[i][j]	u(6)
}	
}	
}	
}	
}	

Fig. 18

invalid_depth_samples_shape_mask(payloadSize) {	Descriptor
idssm_cancel_flag	u(1)
if(! idssm_cancel_flag) {	
idssm_number_of_shapes	u(4)
for(i=0; i < idssm_number_of_shapes; i++) {	
idssm_shape_type[i]	u(3)
idssm_shape_mask_value[i]	u(1)
idssm_shape_center_x[i]	ue(v)
idssm_shape_center_y[i]	ue(v)
idssm_shape_width[i]	ue(v)
idssm_shape_height[i]	ue(v)
}	
}	

Fig. 19

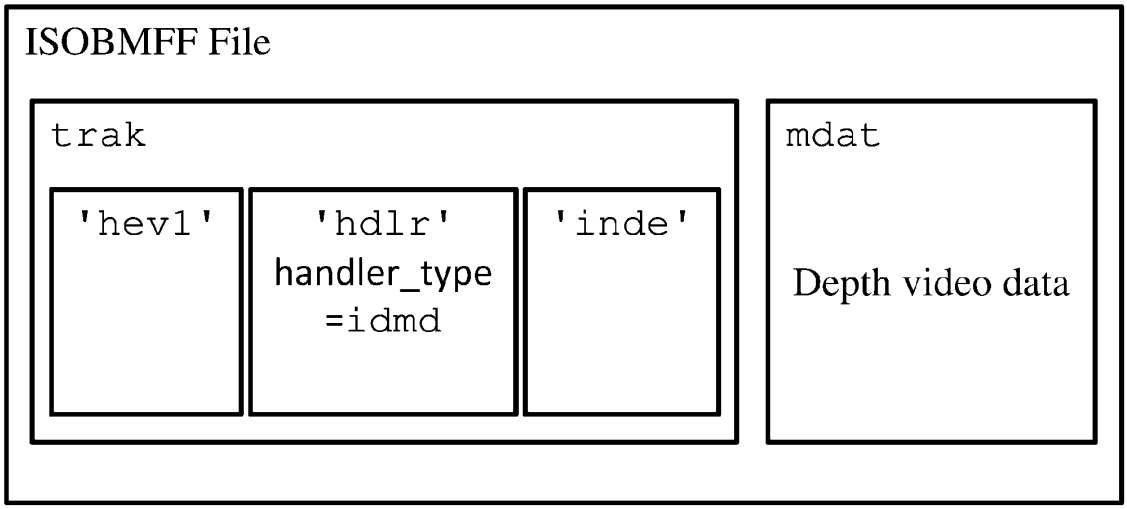


Fig. 20

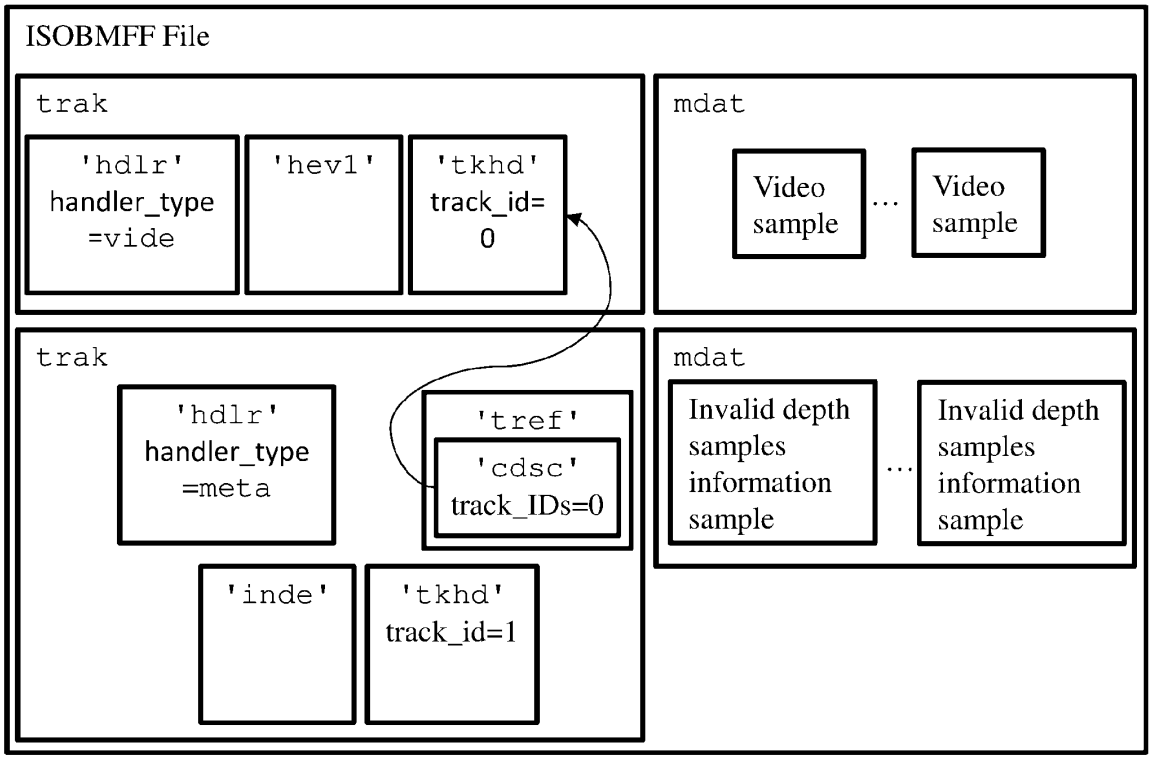

ISOBMFF File
trak
'hdlr'
handler_type
=vide
'hev1'
'tkhd'
track_id=
0
mdat
Video
sample
...
Video
sample
trak
'hdlr'
handler_type
=meta
'tref'
'cdsc'
track_IDs=0
'inde'
'tkhd'
track_id=1
mdat
Invalid depth
samples
information
sample
...
Invalid depth
samples
information
sample

Fig. 21

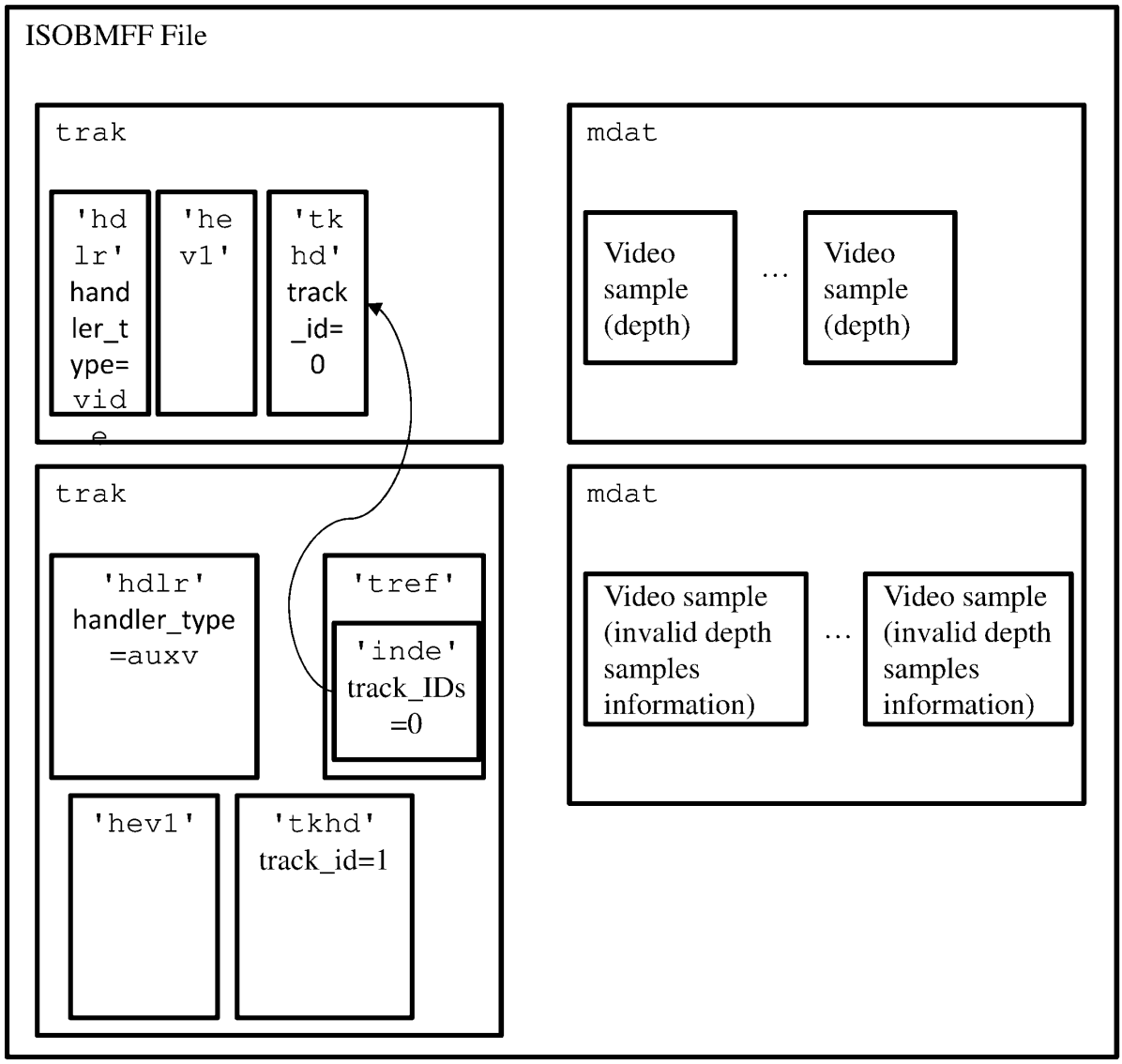

ISOBMFF File
trak
'hdlr' handler_type=vide
'hev1'
'tkhd' track_id=0
mdat
Video sample (depth)
...
Video sample (depth)
trak
'hdlr' handler_type =auxv
'tref'
'inde' track_IDs =0
'hev1'
'tkhd' track_id=1
mdat
Video sample (invalid depth samples information)
...
Video sample (invalid depth samples information)

Fig. 22

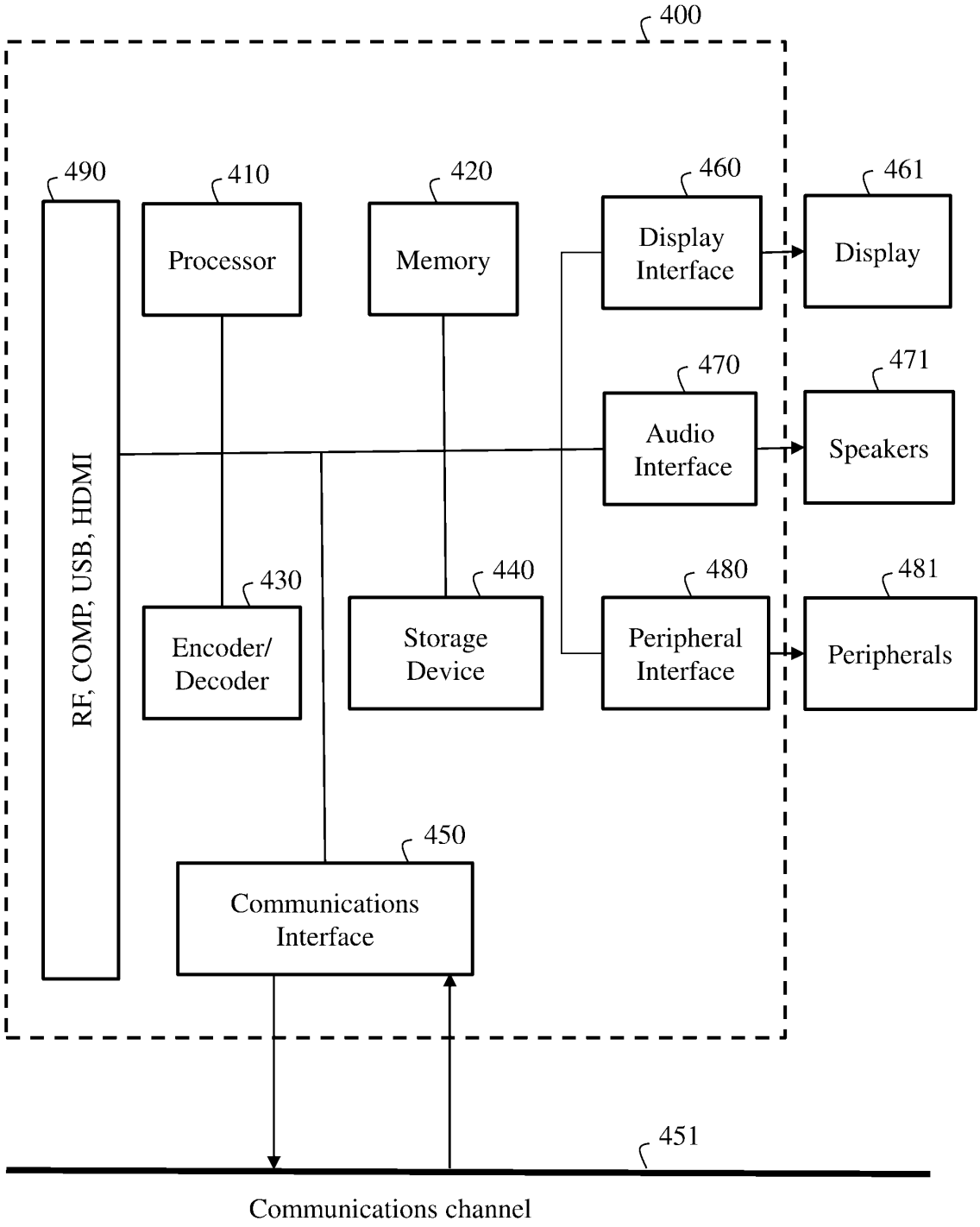

400
490
410
420
460
461
Processor
Memory
Display Interface
Display
470
471
Audio Interface
Speakers
RF, COMP, USB, HDMI
430
440
480
481
Encoder/ Decoder
Storage Device
Peripheral Interface
Peripherals
450
Communications Interface
451
Communications channel

Fig. 23