US012389028B2

(12) **United States Patent**
Hashimoto et al.

(10) **Patent No.: US 12,389,028 B2**
(45) **Date of Patent: Aug. 12, 2025**

(54) **ENCODER, DECODER, ENCODING METHOD, AND DECODING METHOD**

(71) Applicant: **Panasonic Intellectual Property Corporation of America**, Torrance, CA (US)

(72) Inventors: **Takashi Hashimoto**, Hyogo (JP); **Kiyofumi Abe**, Osaka (JP); **Tadamasa Toma**, Osaka (JP); **Takahiro Nishi**, Nara (JP); **Ryuichi Kanoh**, Osaka (JP)

(73) Assignee: **PANASONIC INTELLECTUAL PROPERTY CORPORATION OF AMERICA**, Torrance, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **18/748,771**

(22) Filed: **Jun. 20, 2024**

(65) **Prior Publication Data**

US 2024/0340441 A1 Oct. 10, 2024

Related U.S. Application Data

(63) Continuation of application No. 17/550,210, filed on Dec. 14, 2021, now Pat. No. 12,058,366, which is a continuation of application No. 17/081,267, filed on Oct. 27, 2020, now Pat. No. 11,240,530, which is a continuation of application No. 16/816,733, filed on Mar. 12, 2020, now Pat. No. 10,856,005, which is a

(Continued)

(51) **Int. Cl.**

H04N 19/52	(2014.01)
H04N 19/159	(2014.01)
H04N 19/182	(2014.01)

(52) **U.S. Cl.**

CPC ***H04N 19/52*** (2014.11); ***H04N 19/159*** (2014.11); ***H04N 19/182*** (2014.11)

(58) **Field of Classification Search**

CPC H04N 19/52; H04N 19/159; H04N 19/182; H04N 19/54; H04N 19/57; H04N 19/117; H04N 19/82

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

11,006,106 B2	5/2021	Liu		
2001/0024525 A1 *	9/2001	Hata	.....	H04N 19/107 382/238

(Continued)

FOREIGN PATENT DOCUMENTS

CN	109391814	2/2019
JP	2000-350216	12/2000
WO	2018/230493	12/2018

OTHER PUBLICATIONS

International Search Report issued Jan. 15, 2019 in International (PCT) Application No. PCT/JP2018/039740.

(Continued)

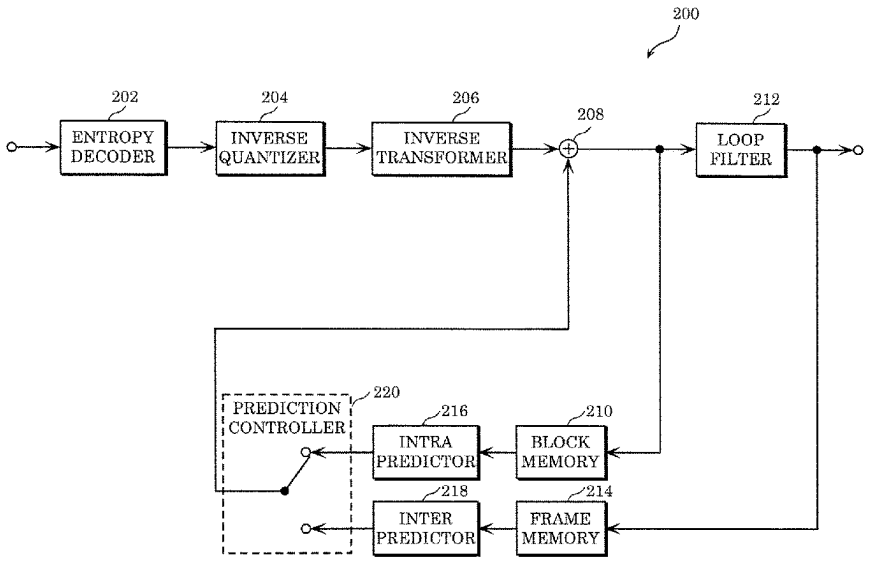
Primary Examiner — Tracy Y. Li

(74) *Attorney, Agent, or Firm* — Wenderoth, Lind & Ponack, L.L.P.

(57) **ABSTRACT**

An encoder includes: circuitry; and memory, in which using the memory, the circuitry, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block.

3 Claims, 26 Drawing Sheets



Related U.S. Application Data

continuation of application No. PCT/JP2018/039740, filed on Oct. 25, 2018.

(60) Provisional application No. 62/577,955, filed on Oct. 27, 2017.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0012619	A1*	1/2004	Glass	H04N 21/4725 375/E7.008
2004/0032908	A1*	2/2004	Hagai	H04N 19/527 375/E7.193
2011/0090960	A1	4/2011	Leontaris	
2013/0163668	A1	6/2013	Chen	
2013/0301946	A1	11/2013	Rossato	
2015/0092851	A1	4/2015	Yoshikawa et al.	
2017/0134742	A1*	5/2017	Deshpande	H04N 19/184
2017/0332095	A1	11/2017	Zou	
2018/0192069	A1	7/2018	Chen	
2018/0359483	A1	12/2018	Chen	
2019/0110064	A1	4/2019	Zhang	
2019/0273943	A1	9/2019	Zhao	
2019/0364284	A1	11/2019	Moon	
2019/0385276	A1	12/2019	Kondo	
2020/0021837	A1	1/2020	Ikai	
2020/0045329	A1	2/2020	Hashimoto	
2020/0177877	A1*	6/2020	Chen	H04N 19/54
2020/0221122	A1	7/2020	Ye	

OTHER PUBLICATIONS

H.265 (ISO/IEC 23008-2 HEVC) / HEVC (High Efficiency Video Coding), Dec. 1, 2013.

Feng Zou et al., "Improved affine motion prediction", Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-C0062_v2, 3rd Meeting: Geneva, CH, May 26-Jun. 1, 2016, pp. 1-5.

Jianle Chen et al., "Algorithm Description of Joint Exploration Test Model 7(JEM 7)", Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-G1001-v1, 7th Meeting: Torino, IT, Jul. 13-21, 2017, pp. i, 18-20.

* cited by examiner

FIG. 1

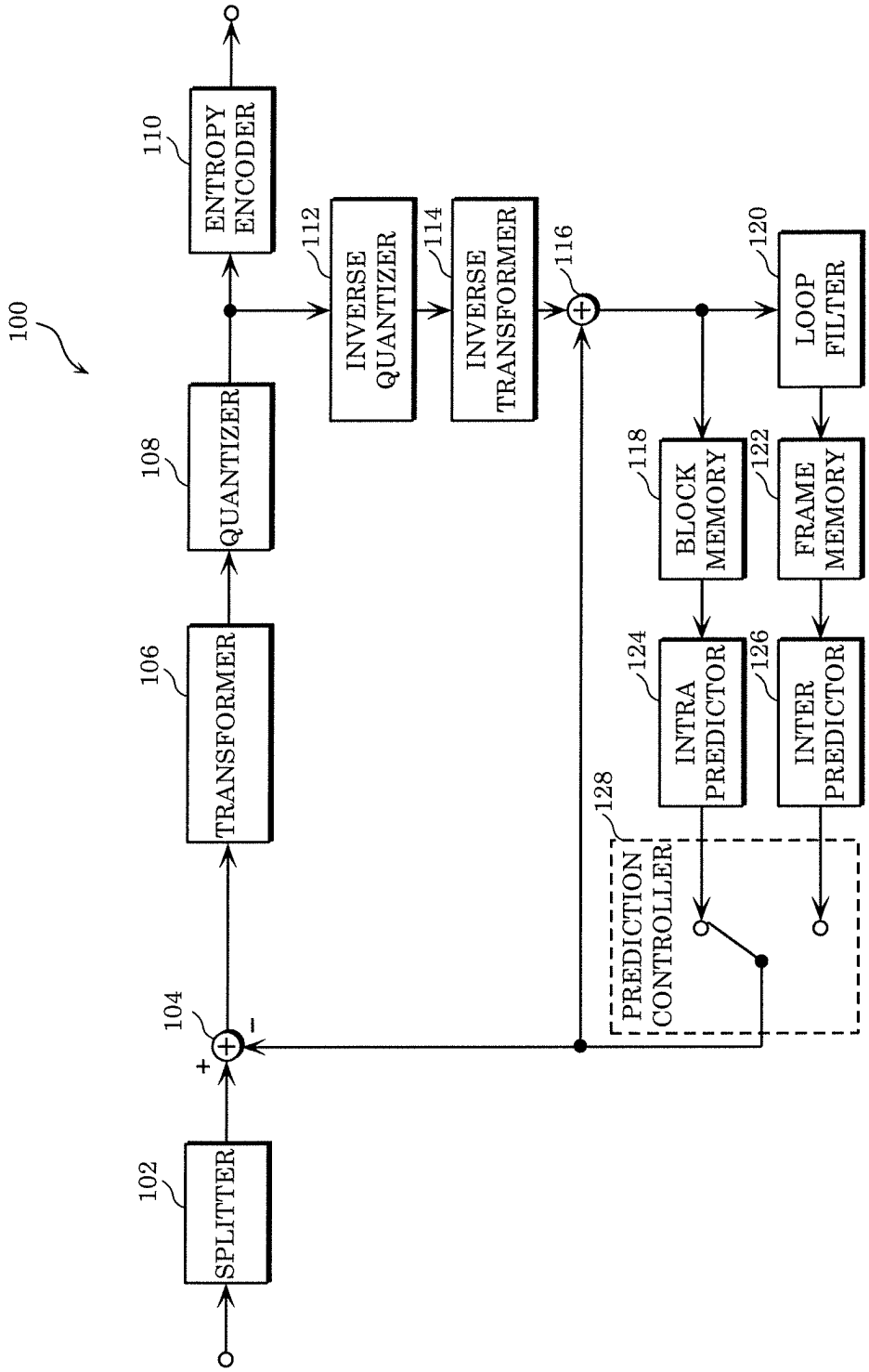

100
102
SPLITTER
104
+
-
106
TRANSFORMER
108
QUANTIZER
110
ENTROPY ENCODER
112
INVERSE QUANTIZER
114
INVERSE TRANSFORMER
116
118
BLOCK MEMORY
120
LOOP FILTER
122
FRAME MEMORY
124
INTRA PREDICTOR
126
INTER PREDICTOR
128
PREDICTION CONTROLLER

FIG. 2

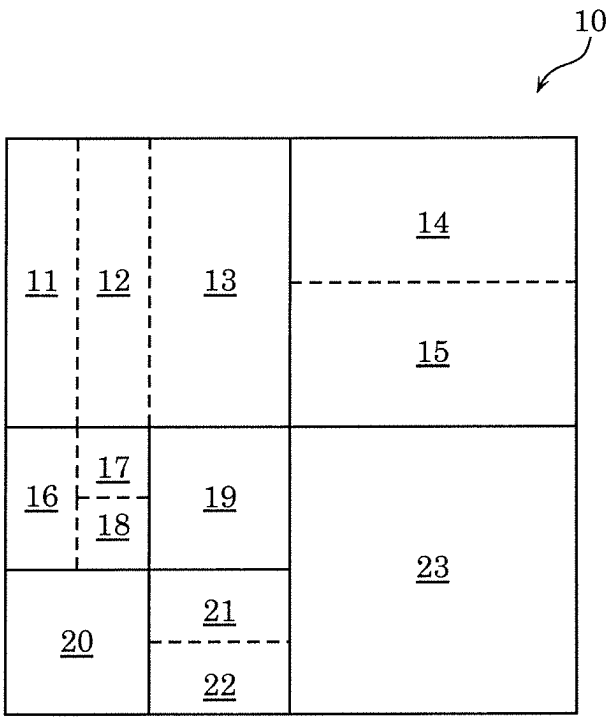


FIG. 3

TRANSFORM TYPE	BASIS FUNCTION $T_i(j)$, i, j = 0, 1, ..., N - 1
DCT - II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j+1)}{2N}\right)$ WHERE $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$
DCT - V	$T_i(j) = \omega_0 \cdot \omega_1 \cdot \sqrt{\frac{2}{2N-1}} \cdot \cos\left(\frac{2\pi \cdot i \cdot j}{2N-1}\right)$ WHERE $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$, $\omega_1 = \begin{cases} \sqrt{\frac{2}{N}} & j = 0 \\ 1 & j \neq 0 \end{cases}$
DCT - VIII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \cos\left(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2}\right)$
DST - I	$T_i(j) = \sqrt{\frac{2}{N+1}} \cdot \sin\left(\frac{\pi \cdot (i+1) \cdot (j+1)}{N+1}\right)$
DST - VII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin\left(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1}\right)$

FIG. 4A

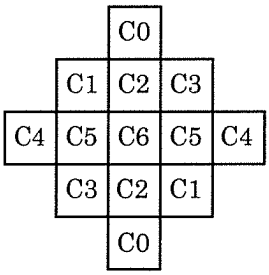


FIG. 4B

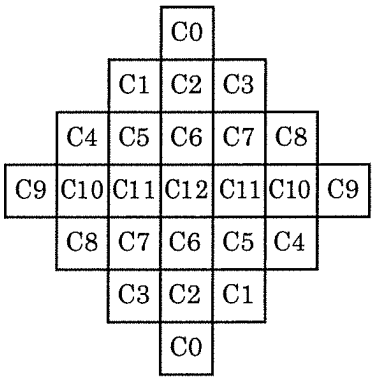


FIG. 4C

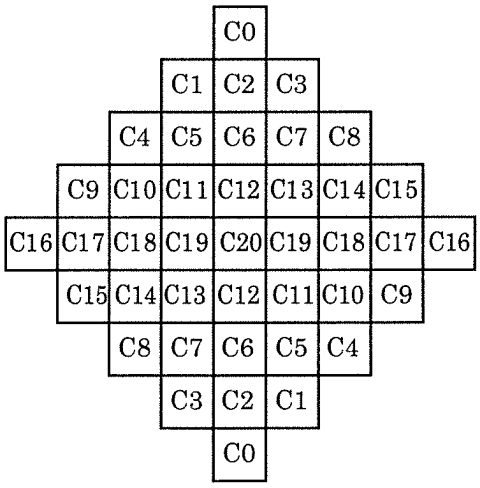


FIG. 5A

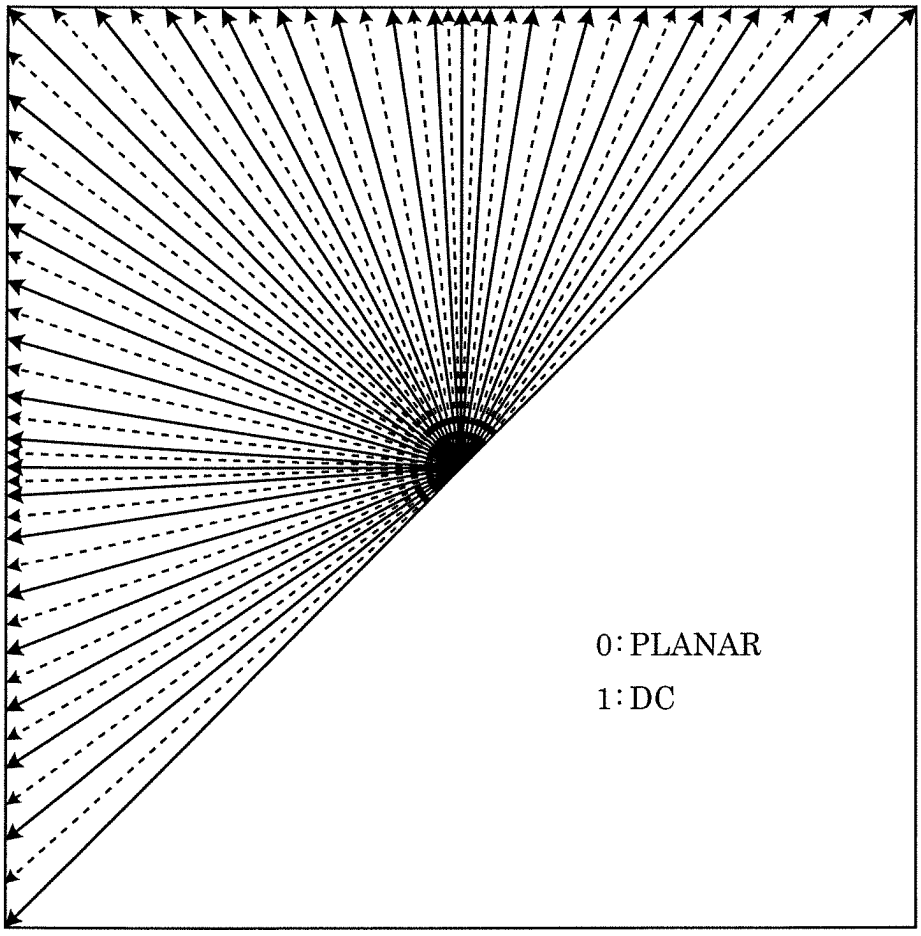


FIG. 5B

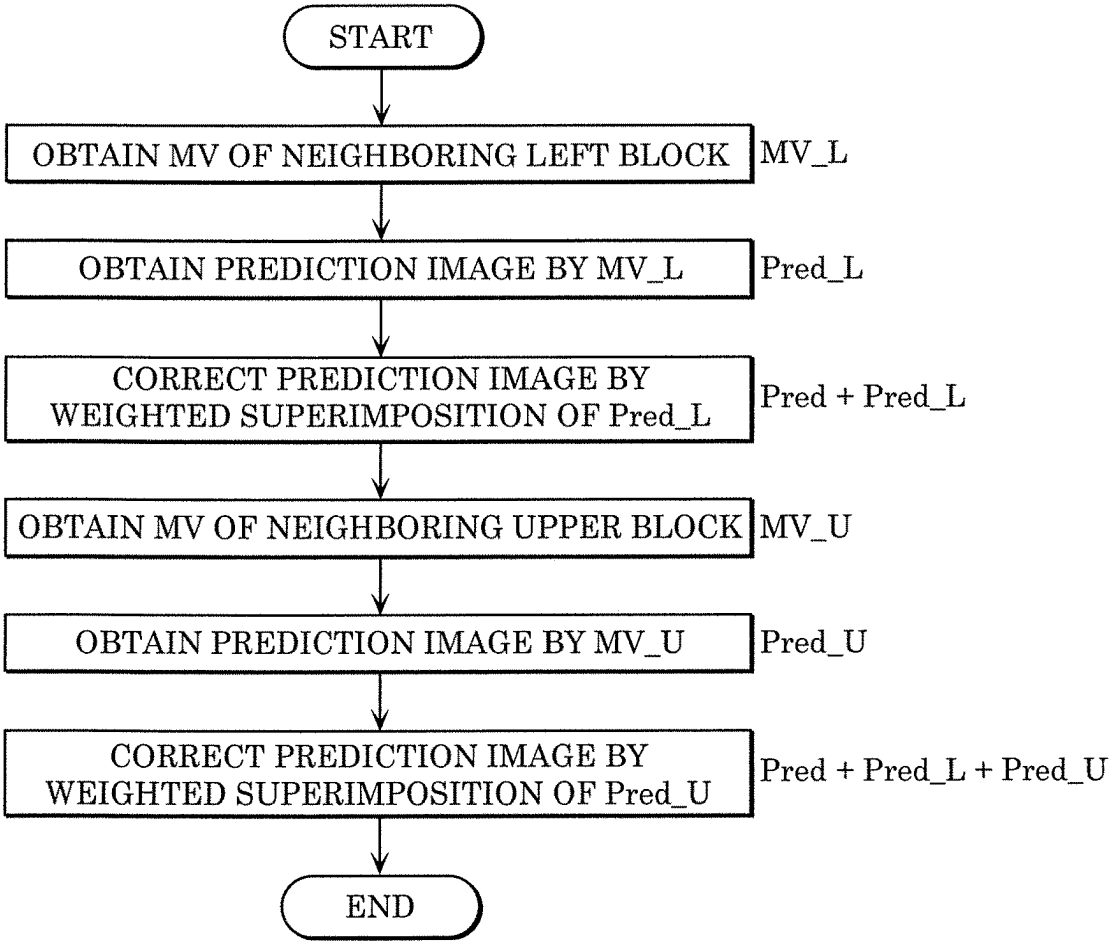

START
OBTAIN MV OF NEIGHBORING LEFT BLOCK
MV_L
OBTAIN PREDICTION IMAGE BY MV_L
Pred_L
CORRECT PREDICTION IMAGE BY WEIGHTED SUPERIMPOSITION OF Pred_L
Pred + Pred_L
OBTAIN MV OF NEIGHBORING UPPER BLOCK
MV_U
OBTAIN PREDICTION IMAGE BY MV_U
Pred_U
CORRECT PREDICTION IMAGE BY WEIGHTED SUPERIMPOSITION OF Pred_U
Pred + Pred_L + Pred_U
END

FIG. 5C

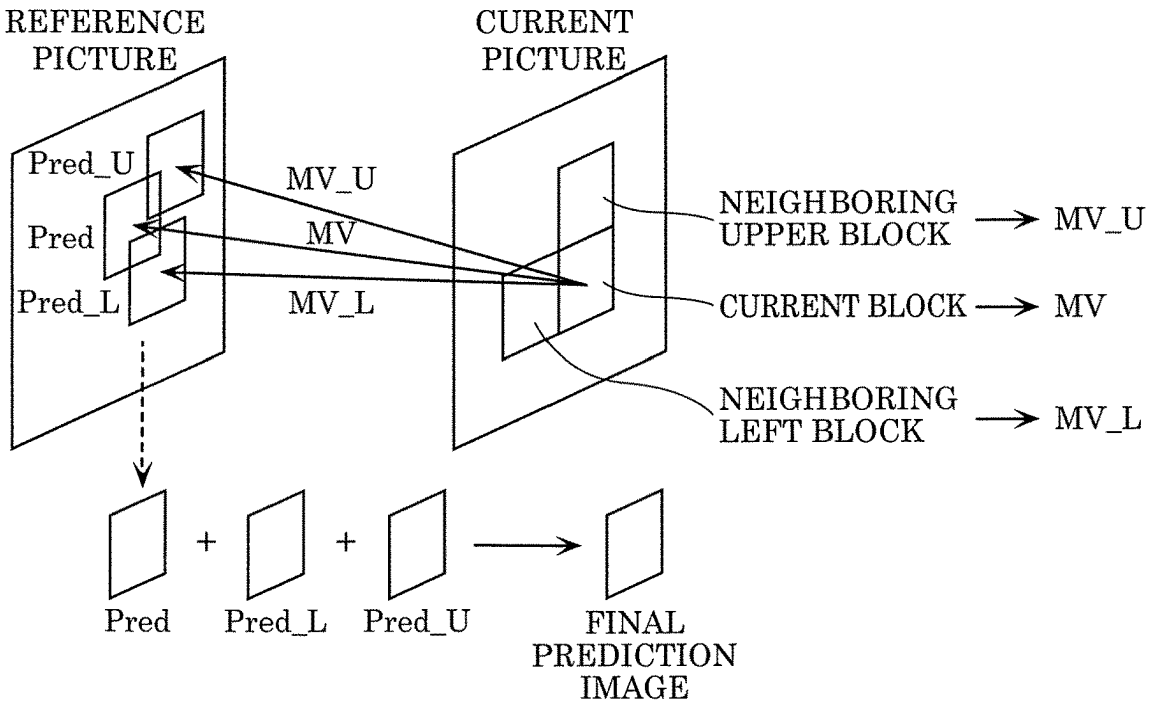

REFERENCE PICTURE
CURRENT PICTURE
Pred_U
Pred
Pred_L
MV_U
MV
MV_L
NEIGHBORING UPPER BLOCK
MV_U
CURRENT BLOCK
MV
NEIGHBORING LEFT BLOCK
MV_L
Pred
Pred_L
Pred_U
FINAL PREDICTION IMAGE

FIG. 5D

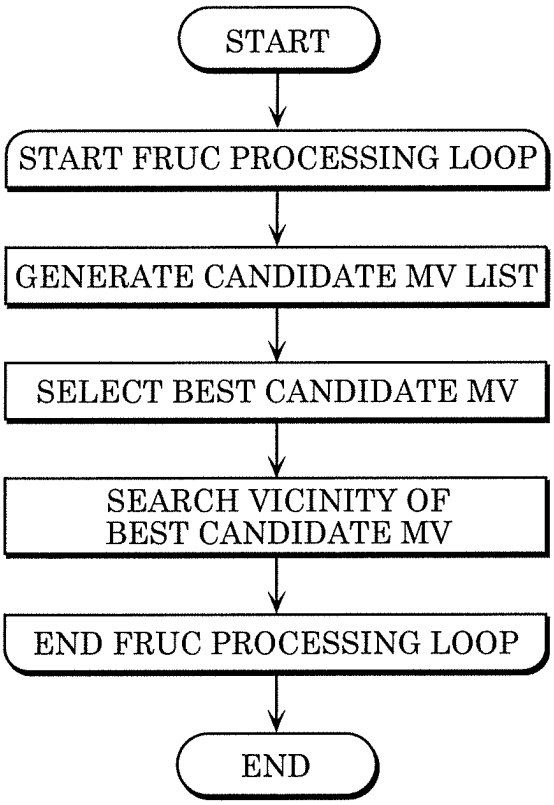

START
START FRUC PROCESSING LOOP
GENERATE CANDIDATE MV LIST
SELECT BEST CANDIDATE MV
SEARCH VICINITY OF BEST CANDIDATE MV
END FRUC PROCESSING LOOP
END

FIG. 6

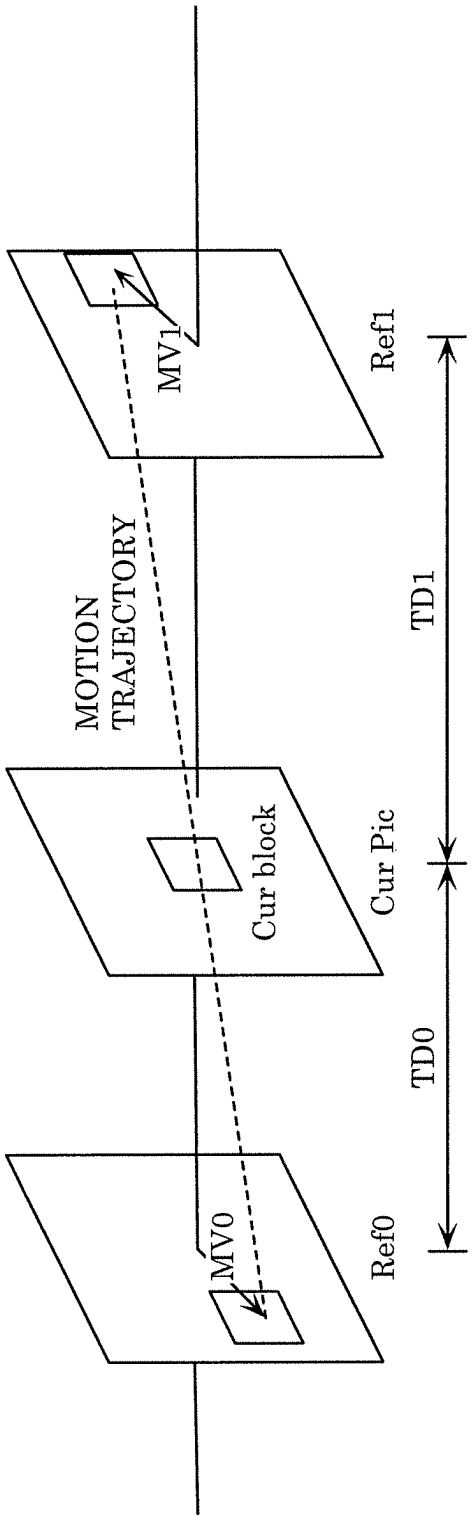

Ref1
MV1
MOTION TRAJECTORY
TD1
Cur Pic
Cur block
TD0
MV0
Ref0

FIG. 7

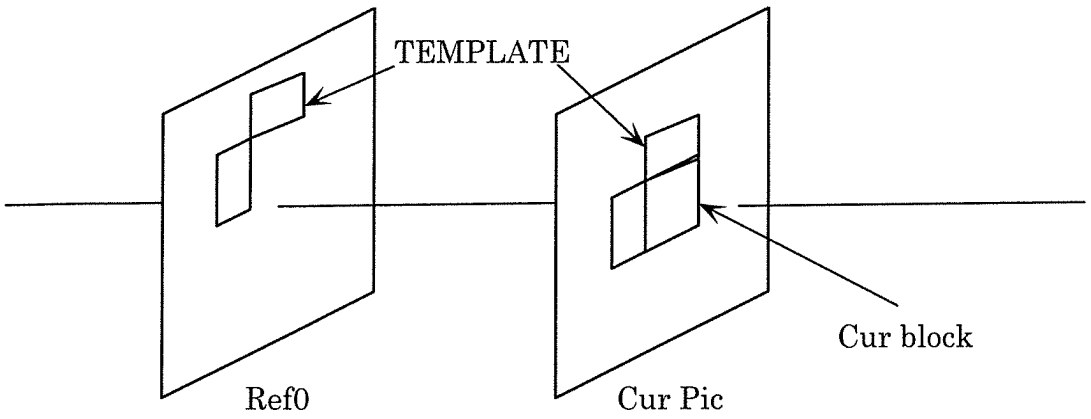

TEMPLATE
Cur block
Ref0
Cur Pic

FIG. 8

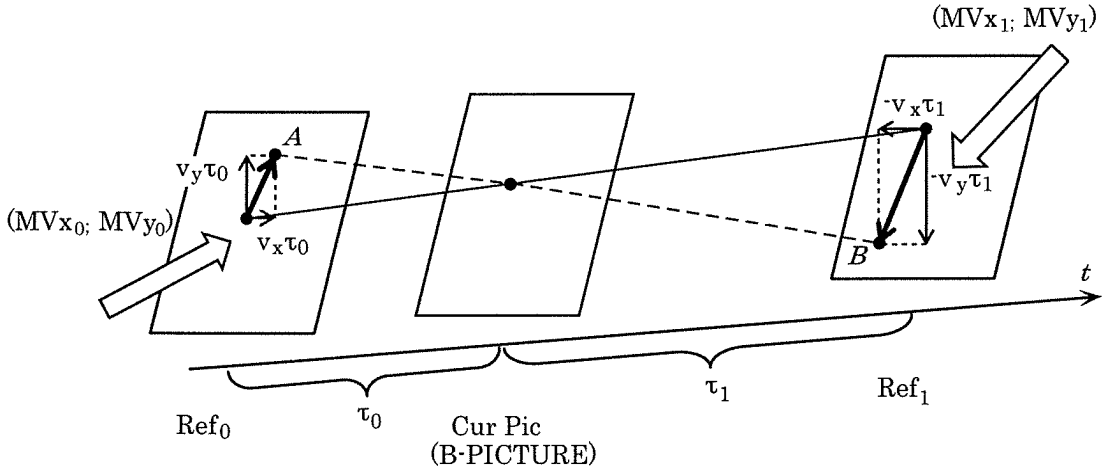

(MVx1; MVy1)
-vxτ1
-vyτ1
A
vyτ0
(MVx0; MVy0)
vxτ0
B
t
τ1
Ref1
Ref0
τ0
Cur Pic
(B-PICTURE)

FIG. 9A

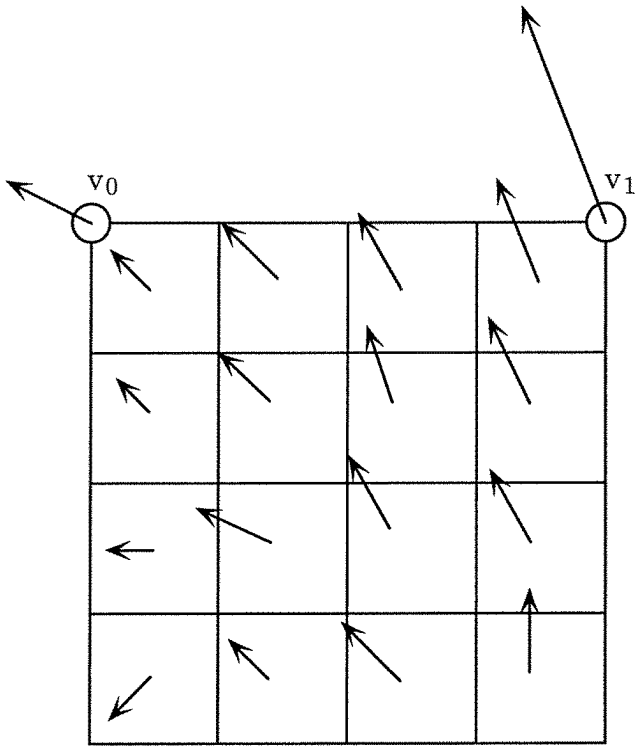

v0
v1

FIG. 9B

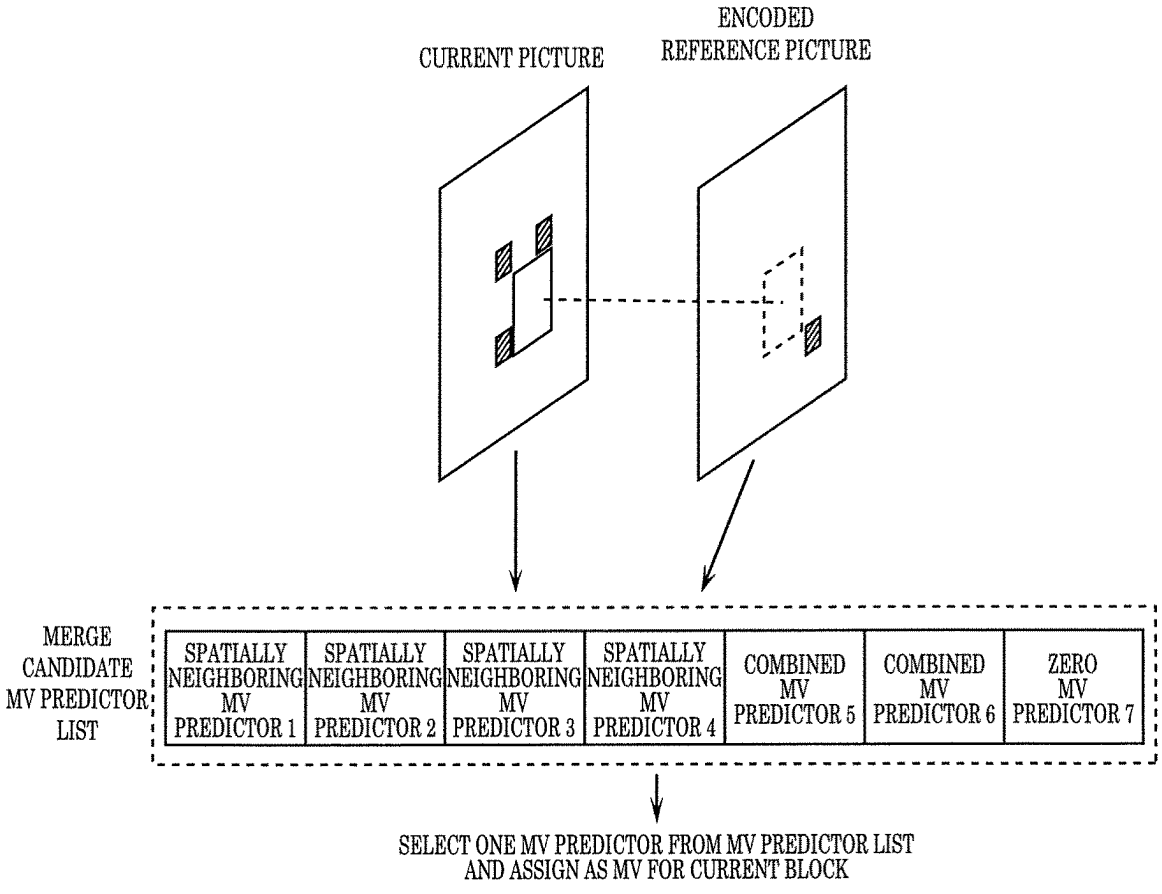

CURRENT PICTURE
ENCODED REFERENCE PICTURE
MERGE CANDIDATE MV PREDICTOR LIST
SPATIALLY NEIGHBORING MV PREDICTOR 1
SPATIALLY NEIGHBORING MV PREDICTOR 2
SPATIALLY NEIGHBORING MV PREDICTOR 3
SPATIALLY NEIGHBORING MV PREDICTOR 4
COMBINED MV PREDICTOR 5
COMBINED MV PREDICTOR 6
ZERO MV PREDICTOR 7
SELECT ONE MV PREDICTOR FROM MV PREDICTOR LIST AND ASSIGN AS MV FOR CURRENT BLOCK

FIG. 9C

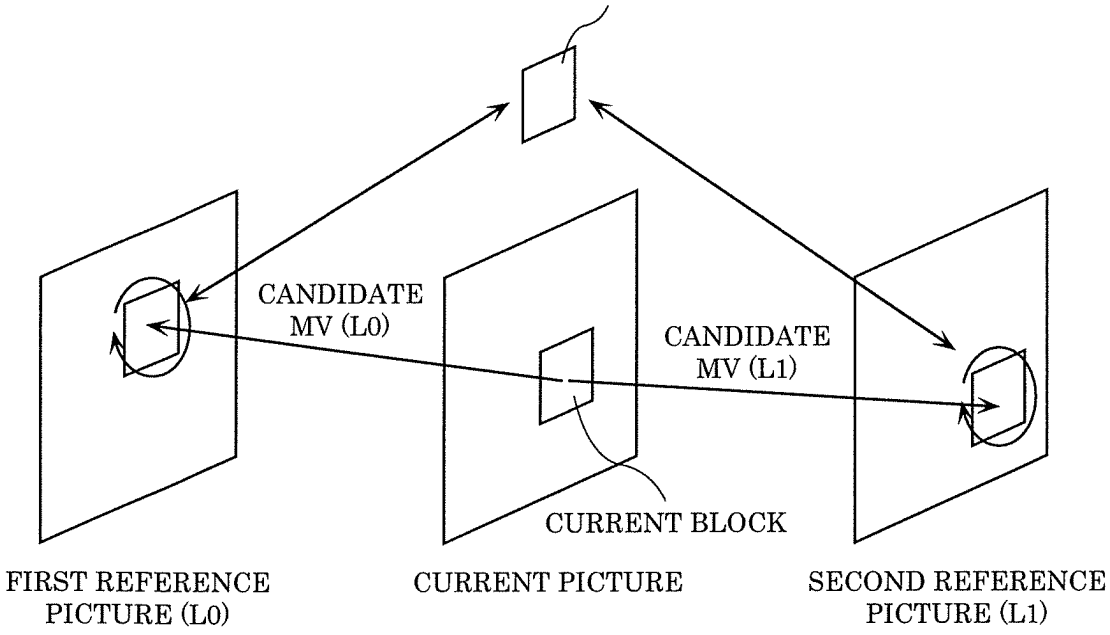

TEMPLATE GENERATED BASED ON REFERENCE PIXEL OF CANDIDATE MV (L0) AND REFERENCE PIXEL OF CANDIDATE MV (L1)
CANDIDATE MV (L0)
CANDIDATE MV (L1)
CURRENT BLOCK
FIRST REFERENCE PICTURE (L0)
CURRENT PICTURE
SECOND REFERENCE PICTURE (L1)

FIG. 9D

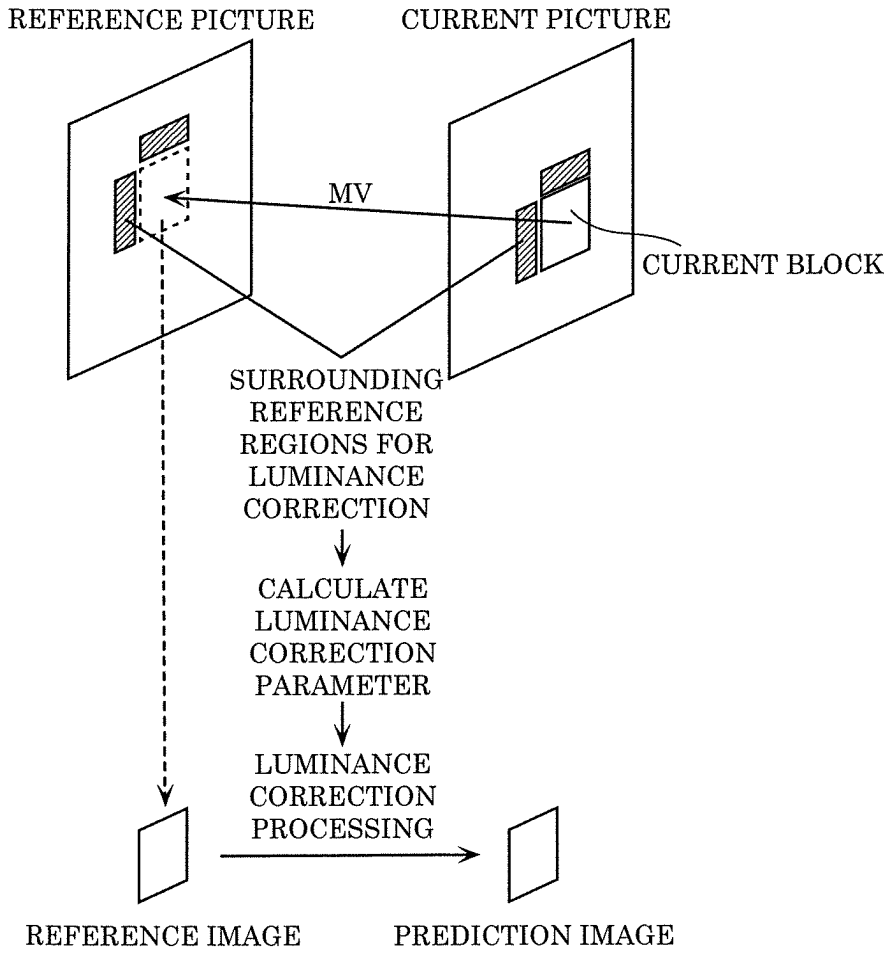

REFERENCE PICTURE
CURRENT PICTURE
MV
CURRENT BLOCK
SURROUNDING REFERENCE REGIONS FOR LUMINANCE CORRECTION
CALCULATE LUMINANCE CORRECTION PARAMETER
LUMINANCE CORRECTION PROCESSING
REFERENCE IMAGE
PREDICTION IMAGE

FIG. 10

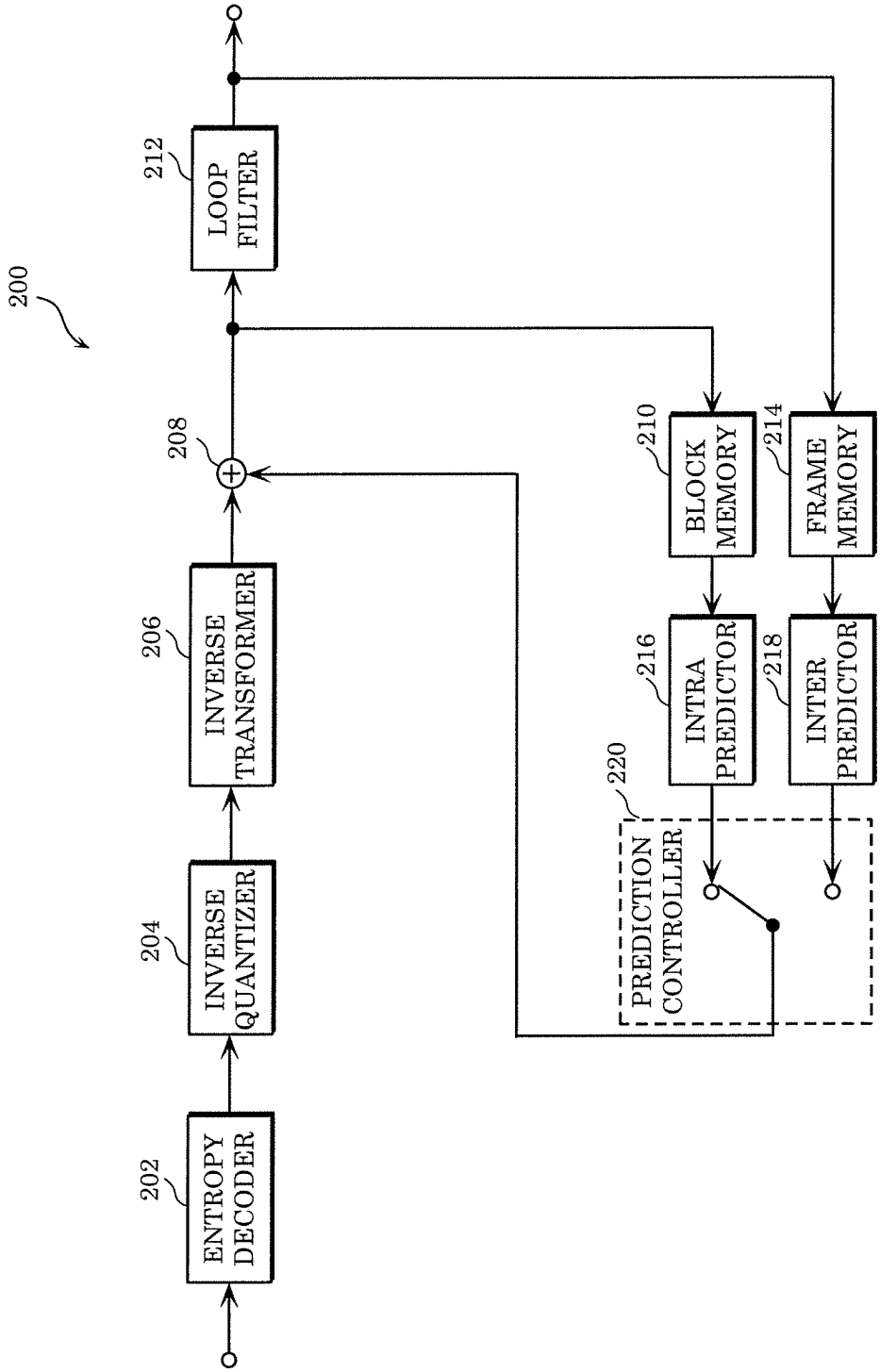

200
202
ENTROPY DECODER
204
INVERSE QUANTIZER
206
INVERSE TRANSFORMER
208
212
LOOP FILTER
210
BLOCK MEMORY
214
FRAME MEMORY
216
INTRA PREDICTOR
218
INTER PREDICTOR
220
PREDICTION CONTROLLER

FIG. 11

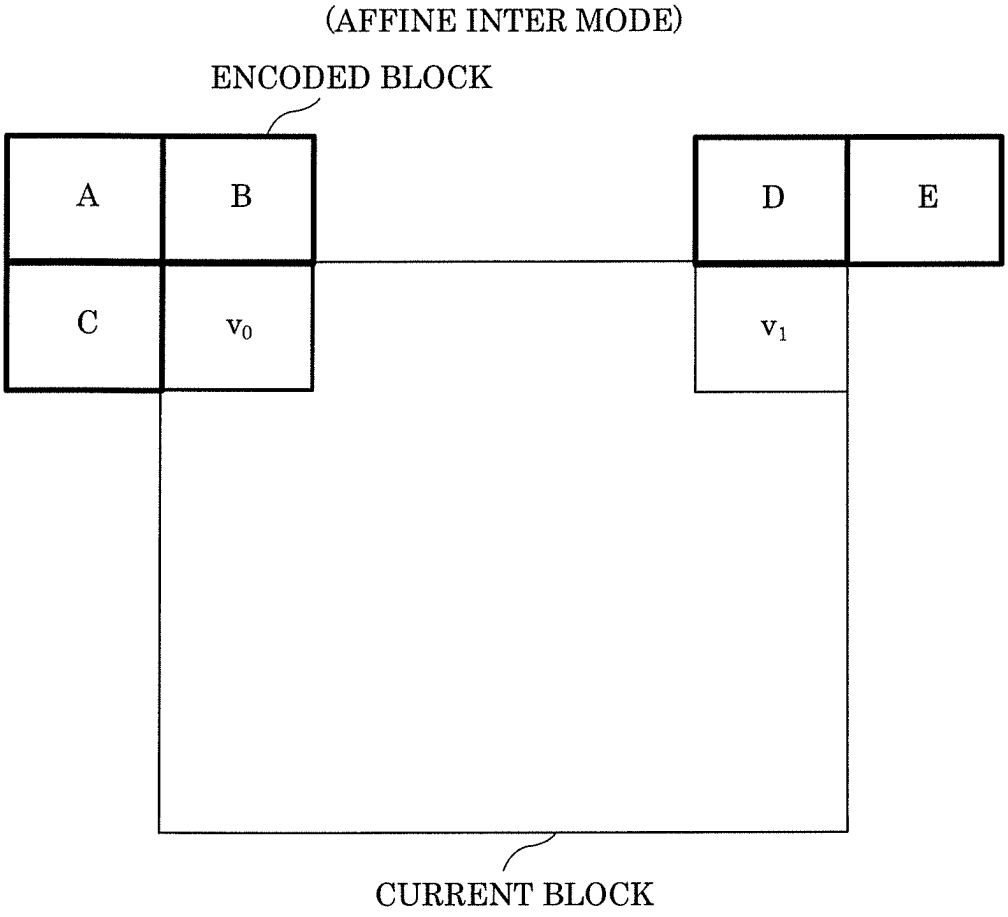


FIG. 12A

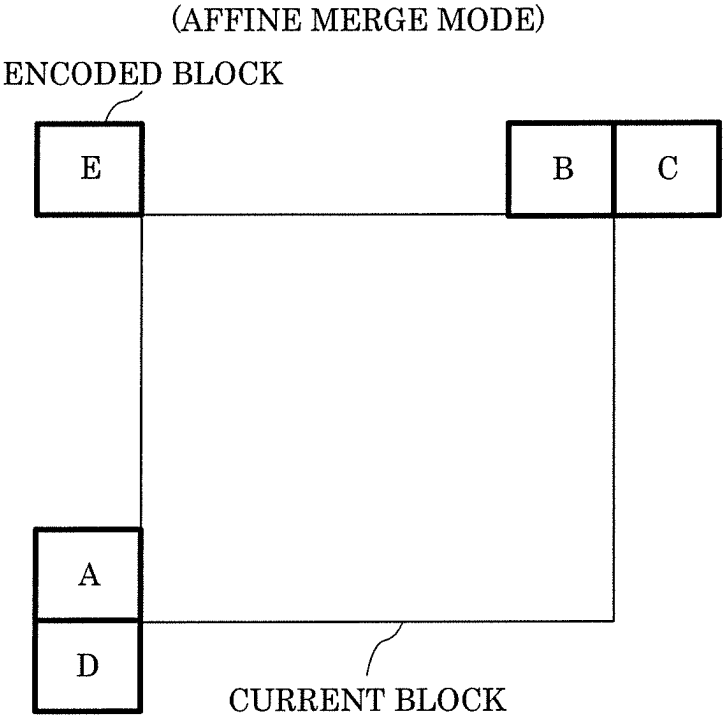


FIG. 12B

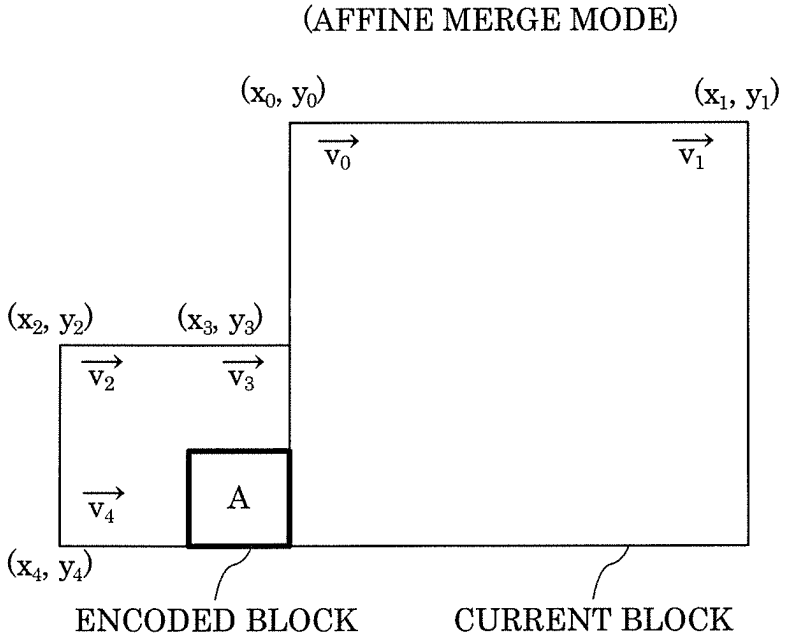


FIG. 13

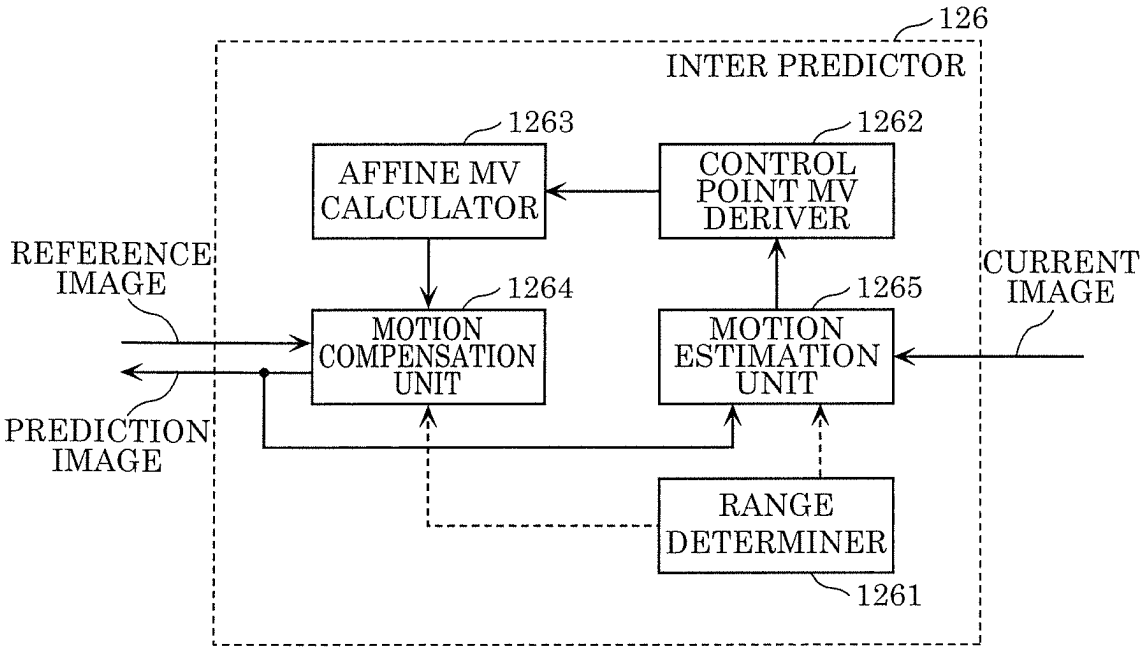

126
INTER PREDICTOR
1263
AFFINE MV CALCULATOR
1262
CONTROL POINT MV DERIVER
REFERENCE IMAGE
1264
MOTION COMPENSATION UNIT
1265
MOTION ESTIMATION UNIT
CURRENT IMAGE
PREDICTION IMAGE
RANGE DETERMINER
1261

FIG. 14

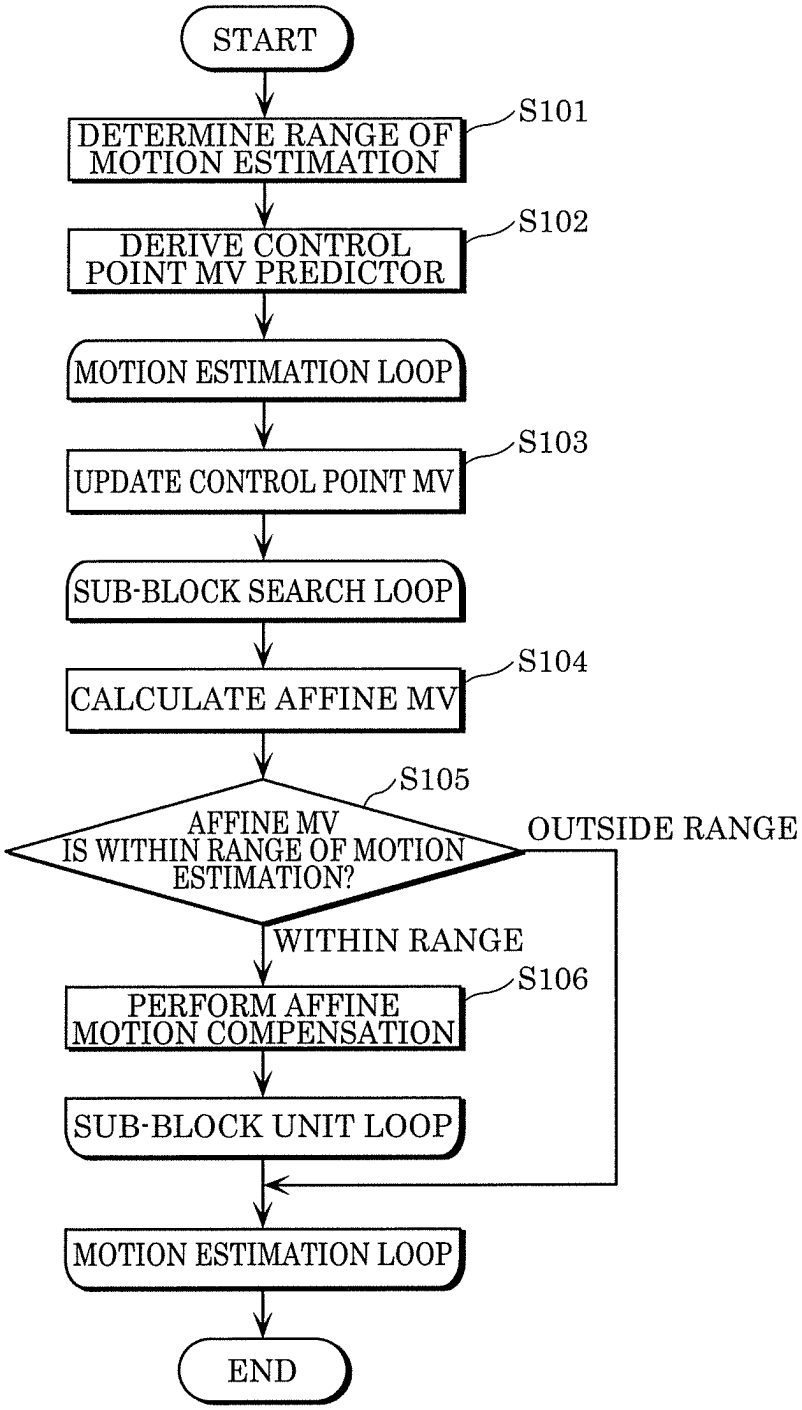

START
DETERMINE RANGE OF MOTION ESTIMATION
S101
DERIVE CONTROL POINT MV PREDICTOR
S102
MOTION ESTIMATION LOOP
UPDATE CONTROL POINT MV
S103
SUB-BLOCK SEARCH LOOP
CALCULATE AFFINE MV
S104
AFFINE MV IS WITHIN RANGE OF MOTION ESTIMATION?
S105
OUTSIDE RANGE
WITHIN RANGE
PERFORM AFFINE MOTION COMPENSATION
S106
SUB-BLOCK UNIT LOOP
MOTION ESTIMATION LOOP
END

FIG. 15

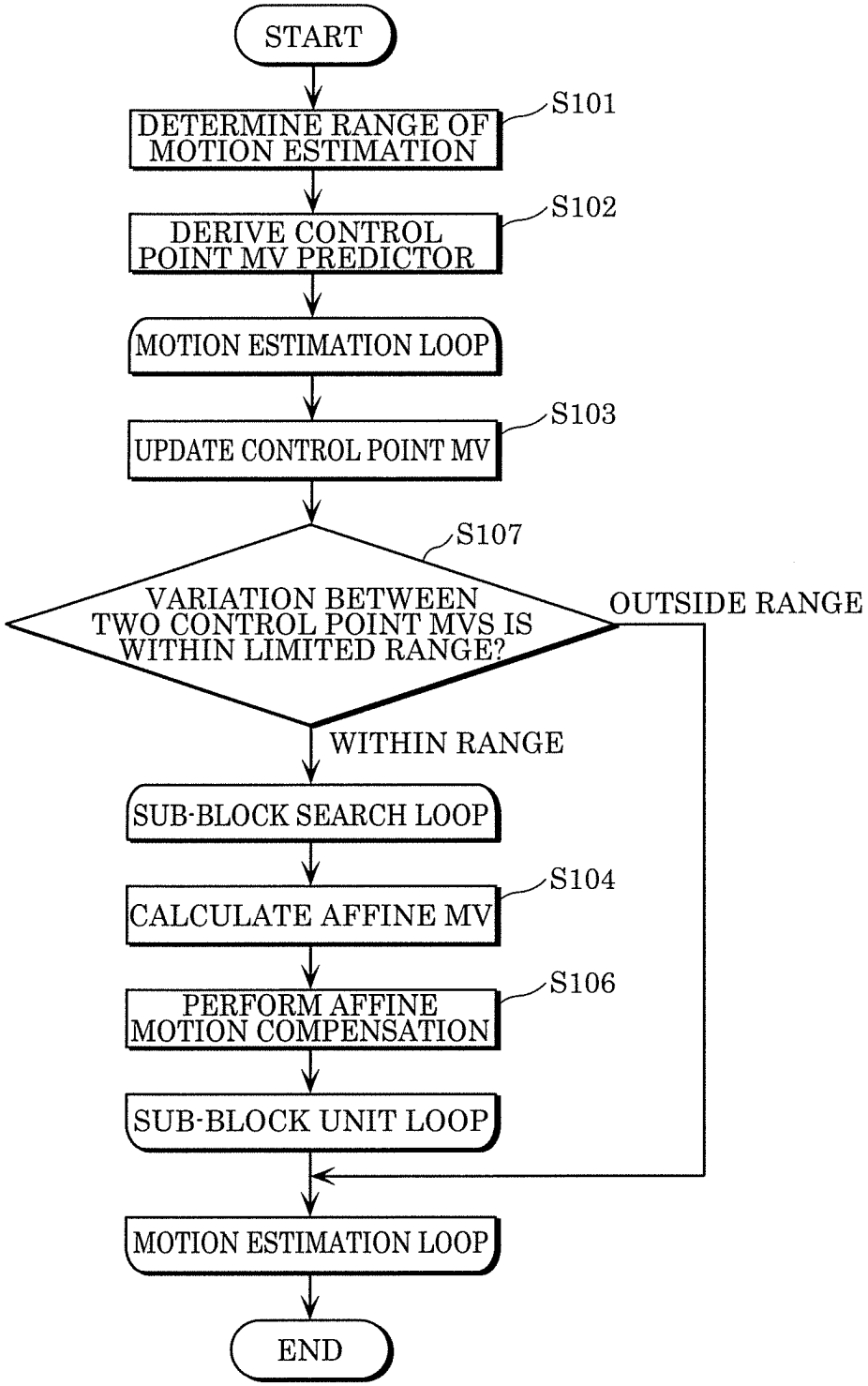

START
DETERMINE RANGE OF MOTION ESTIMATION
S101
DERIVE CONTROL POINT MV PREDICTOR
S102
MOTION ESTIMATION LOOP
UPDATE CONTROL POINT MV
S103
S107
VARIATION BETWEEN TWO CONTROL POINT MVS IS WITHIN LIMITED RANGE?
OUTSIDE RANGE
WITHIN RANGE
SUB-BLOCK SEARCH LOOP
CALCULATE AFFINE MV
S104
PERFORM AFFINE MOTION COMPENSATION
S106
SUB-BLOCK UNIT LOOP
MOTION ESTIMATION LOOP
END

FIG. 16

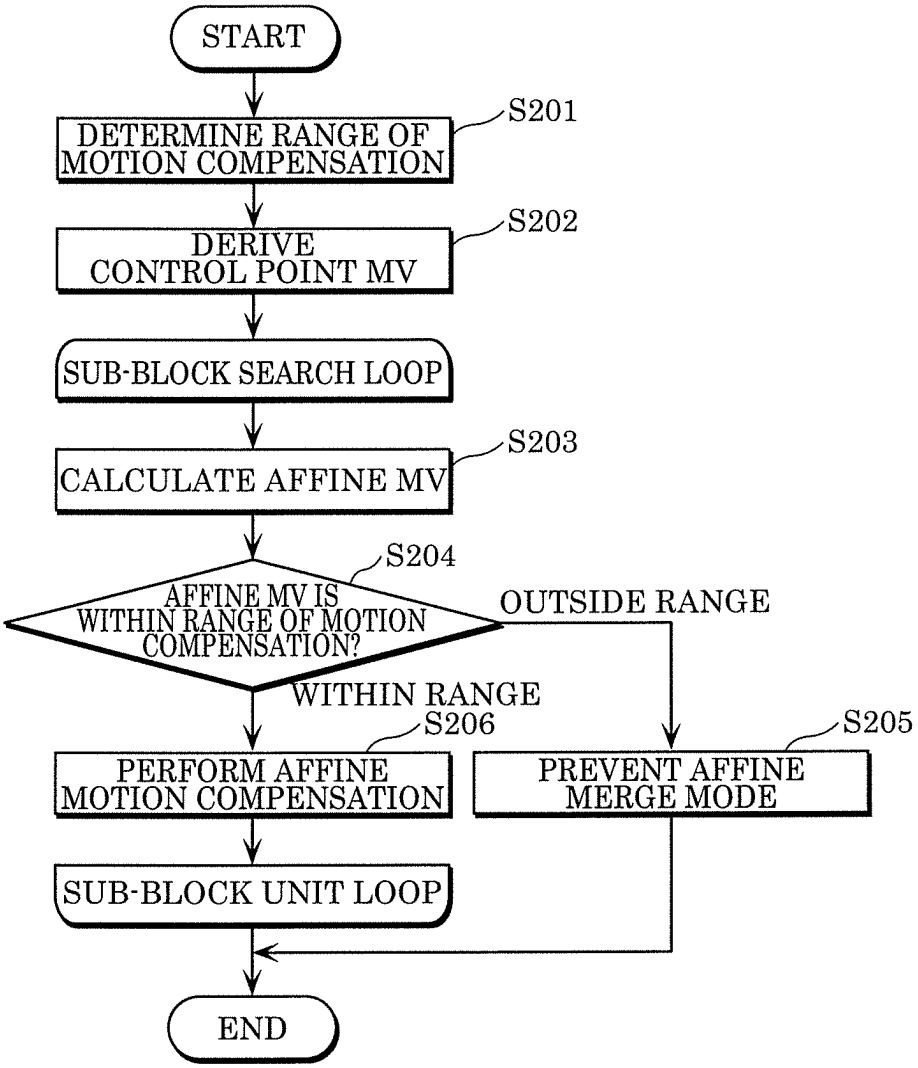

START
DETERMINE RANGE OF MOTION COMPENSATION
S201
DERIVE CONTROL POINT MV
S202
SUB-BLOCK SEARCH LOOP
CALCULATE AFFINE MV
S203
AFFINE MV IS WITHIN RANGE OF MOTION COMPENSATION?
S204
OUTSIDE RANGE
WITHIN RANGE
PERFORM AFFINE MOTION COMPENSATION
S206
PREVENT AFFINE MERGE MODE
S205
SUB-BLOCK UNIT LOOP
END

FIG. 17

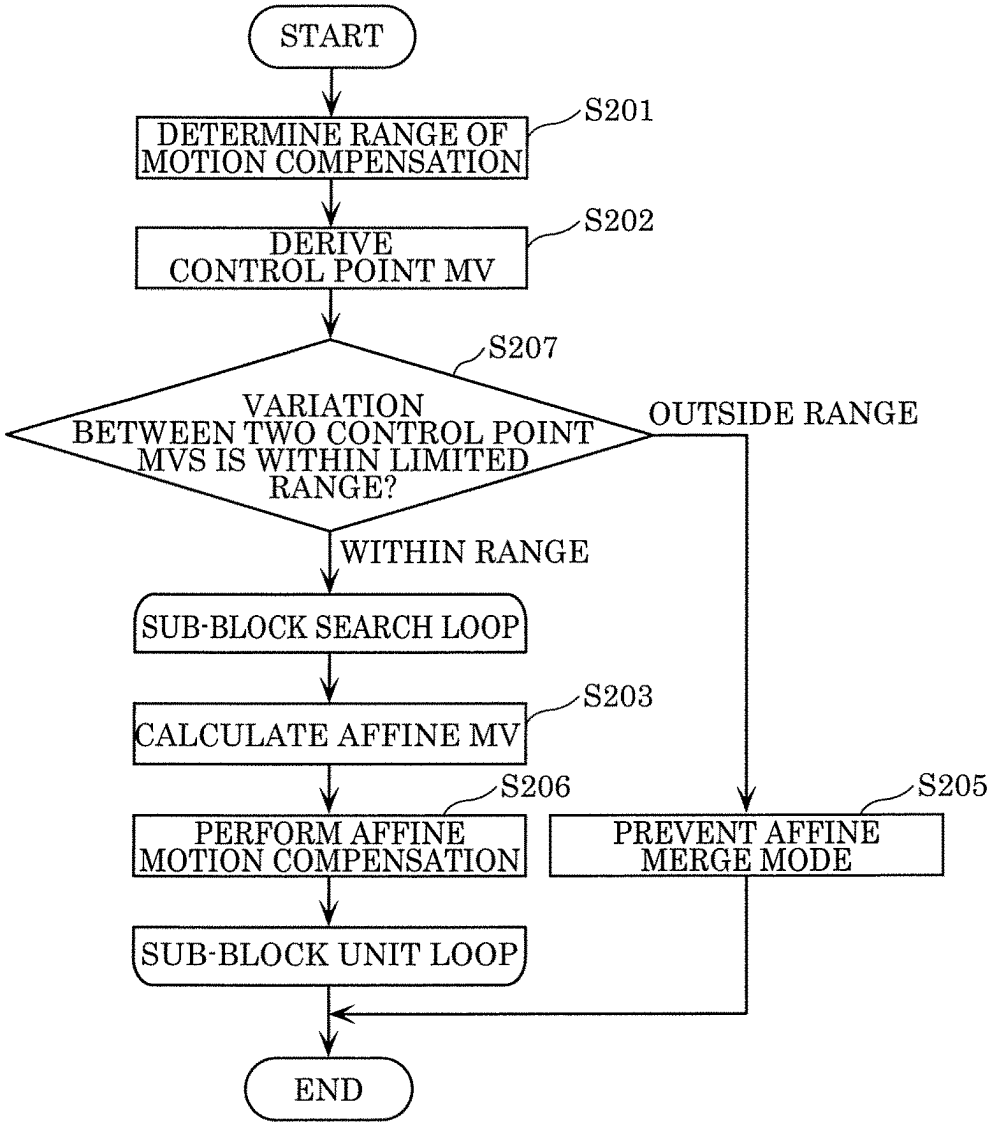

START
DETERMINE RANGE OF MOTION COMPENSATION
S201
DERIVE CONTROL POINT MV
S202
VARIATION BETWEEN TWO CONTROL POINT MVS IS WITHIN LIMITED RANGE?
S207
OUTSIDE RANGE
WITHIN RANGE
SUB-BLOCK SEARCH LOOP
CALCULATE AFFINE MV
S203
PERFORM AFFINE MOTION COMPENSATION
S206
PREVENT AFFINE MERGE MODE
S205
SUB-BLOCK UNIT LOOP
END

FIG. 18

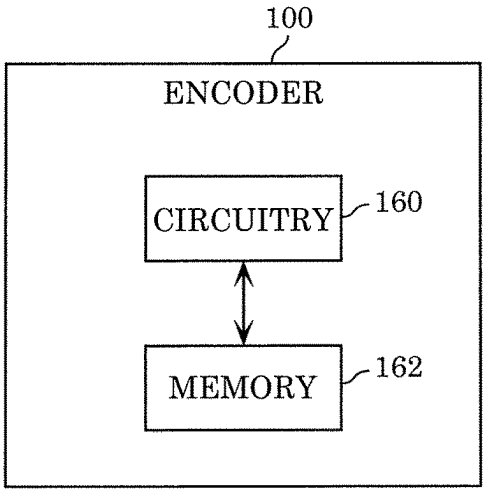

100
ENCODER
CIRCUITRY
160
MEMORY
162

FIG. 19

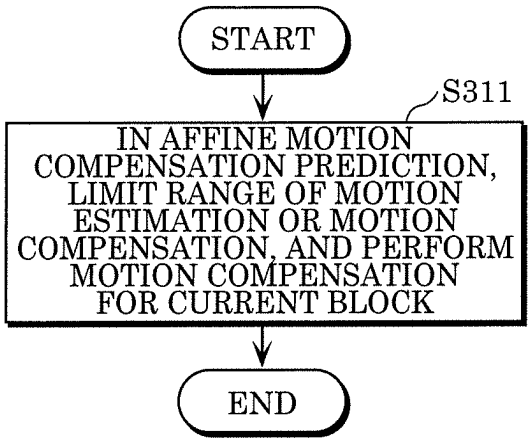


FIG. 20

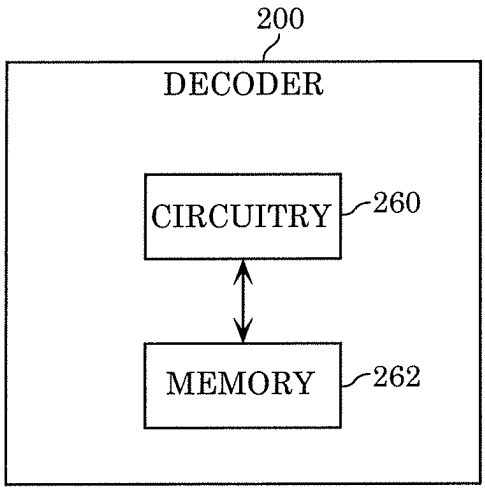


FIG. 21

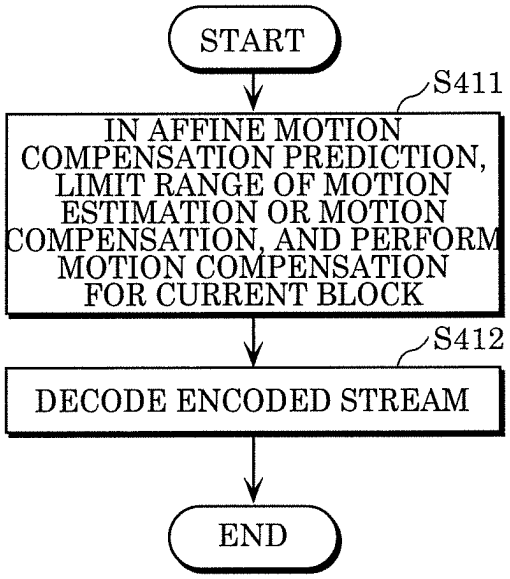


FIG. 22

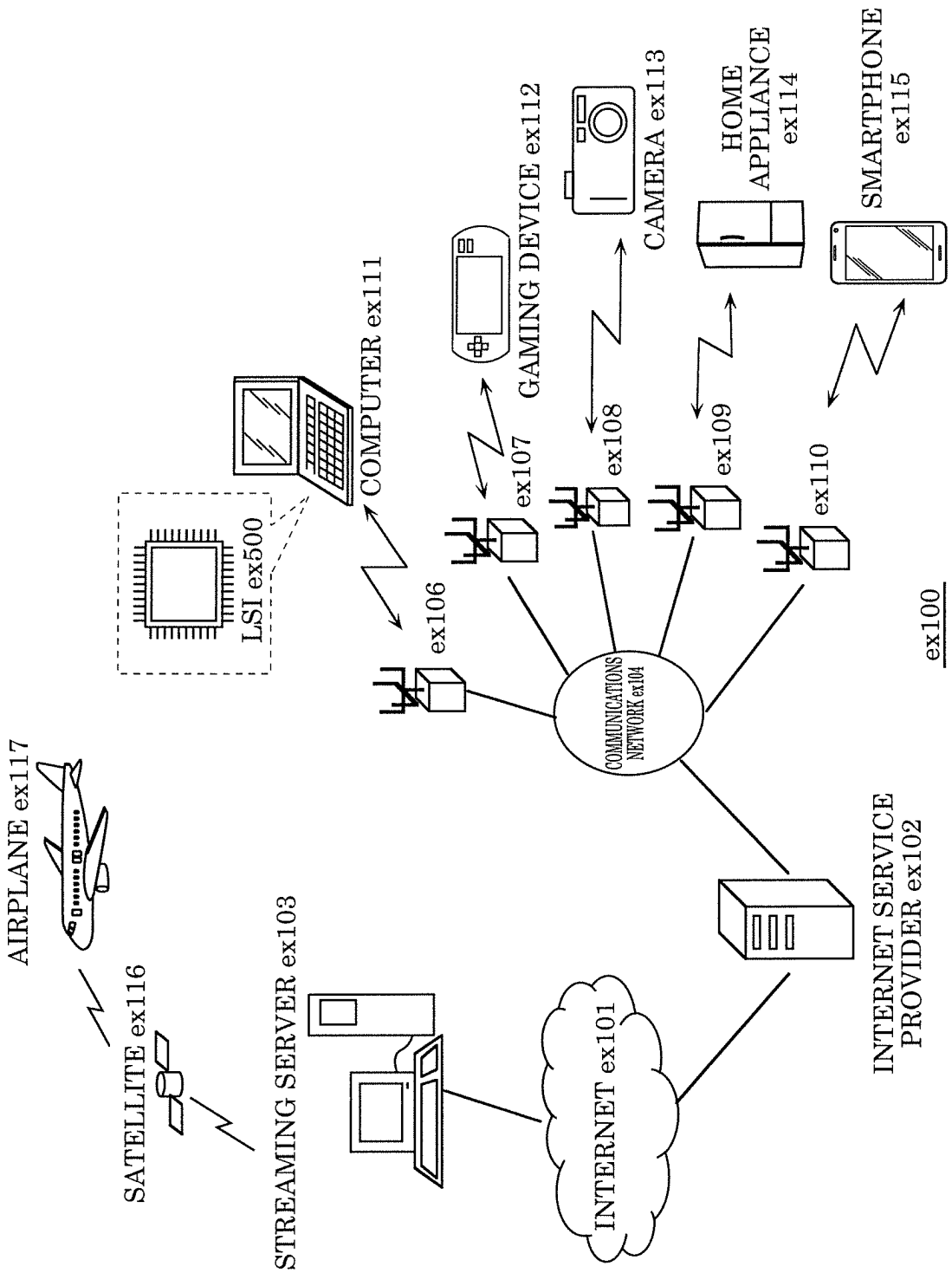

AIRPLANE ex117
SATELLITE ex116
STREAMING SERVER ex103
INTERNET ex101
INTERNET SERVICE PROVIDER ex102
COMMUNICATIONS NETWORK ex104
ex100
ex106
ex107
ex108
ex109
ex110
LSI ex500
COMPUTER ex111
GAMING DEVICE ex112
CAMERA ex113
HOME APPLIANCE ex114
SMARTPHONE ex115

FIG. 23

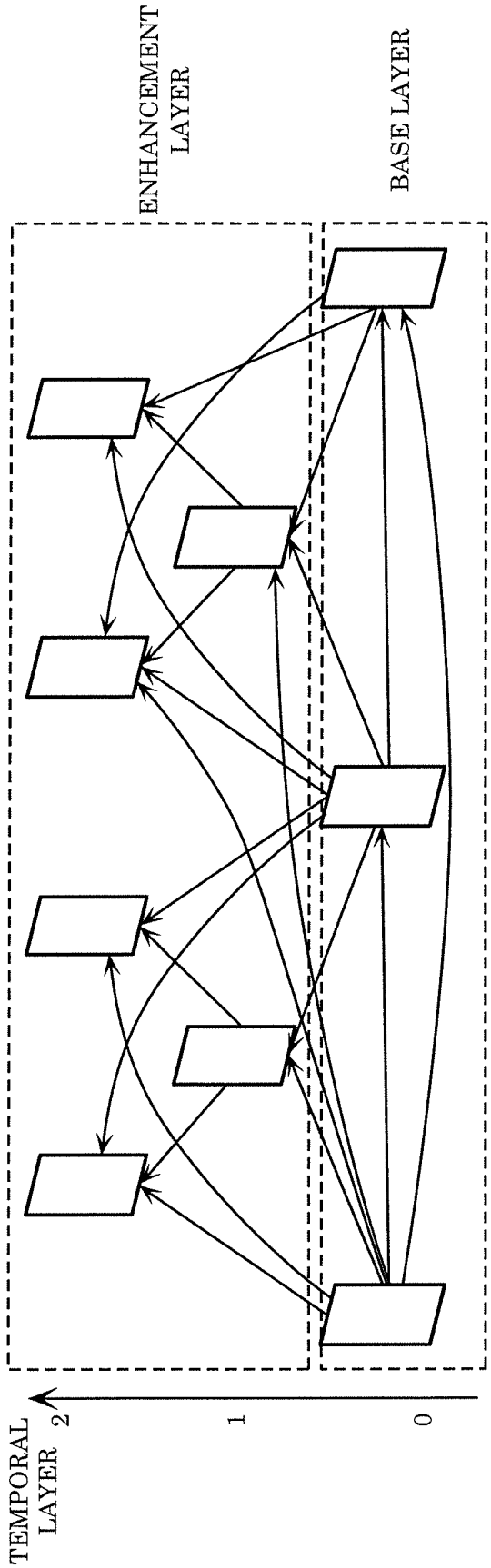
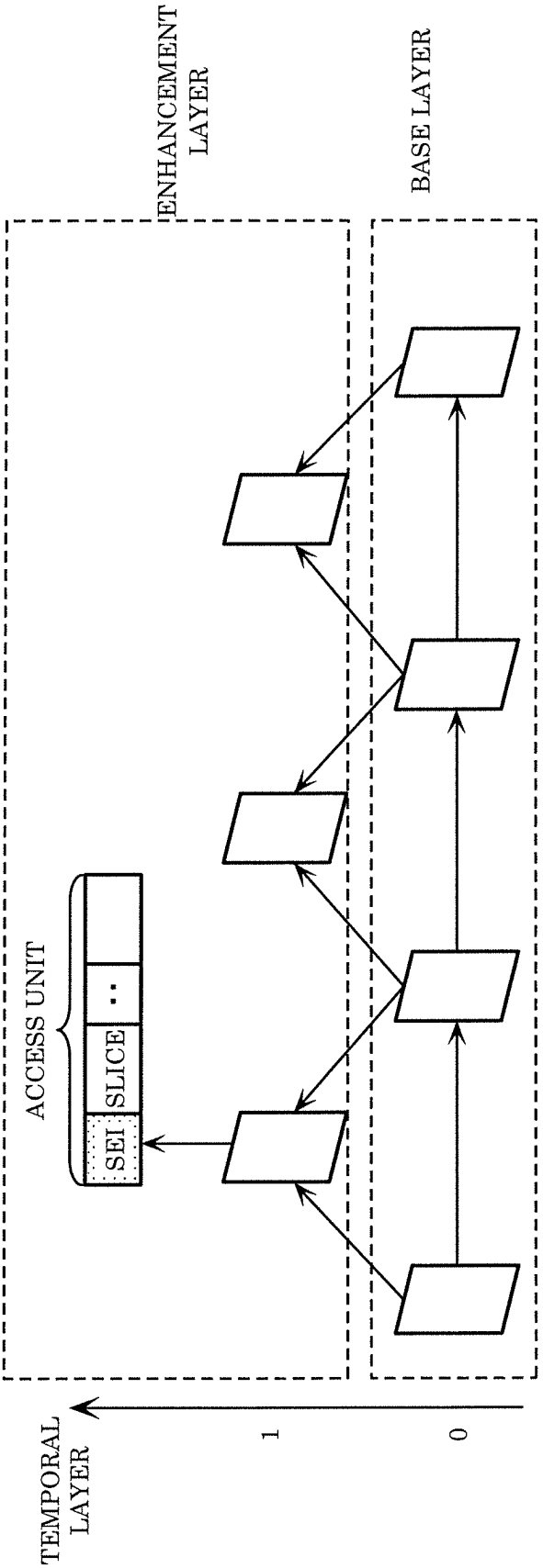


FIG. 24


ENHANCEMENT LAYER
BASE LAYER
ACCESS UNIT
SEI
SLICE
..
TEMPORAL LAYER
1
0

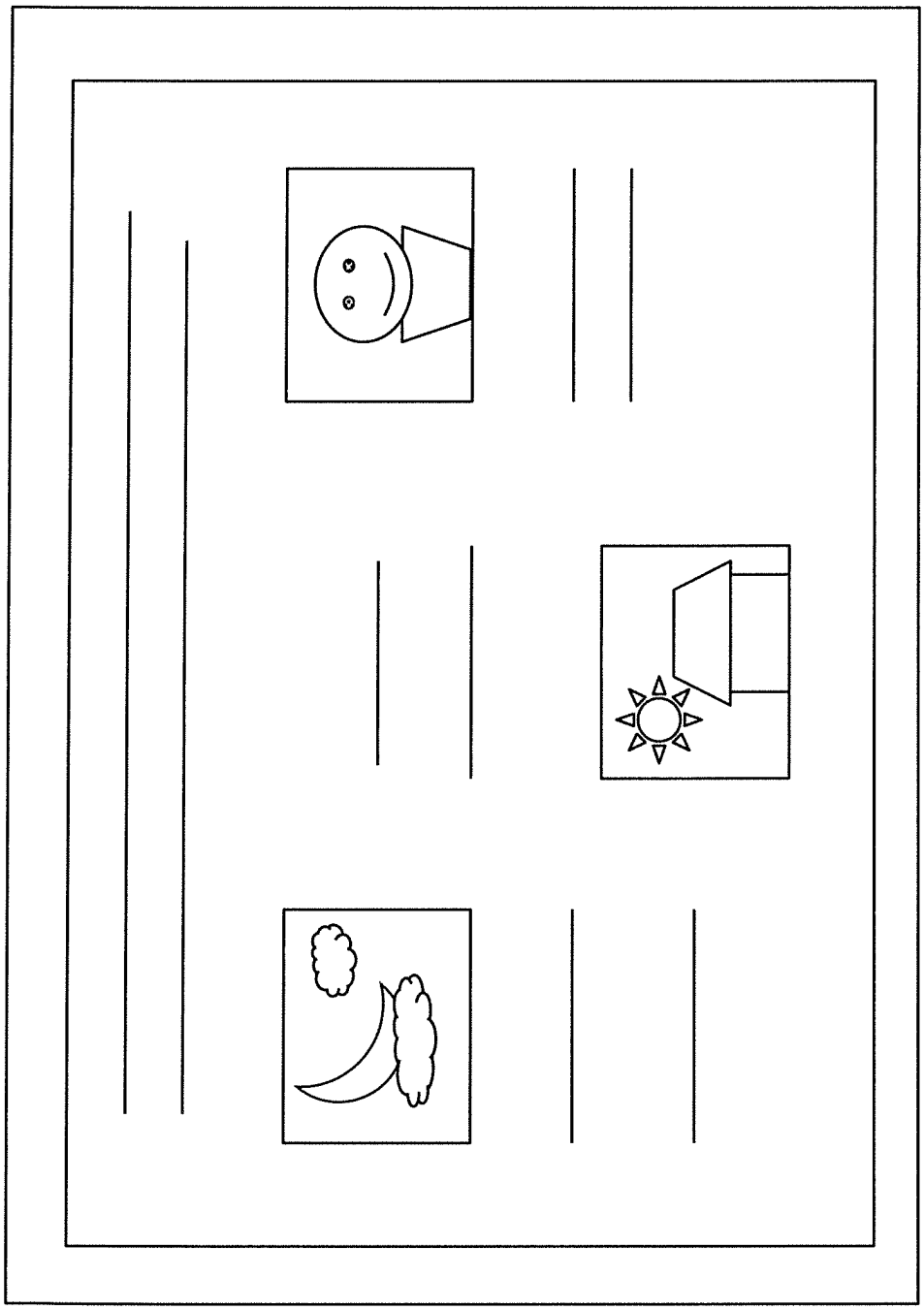


FIG. 25

FIG. 26

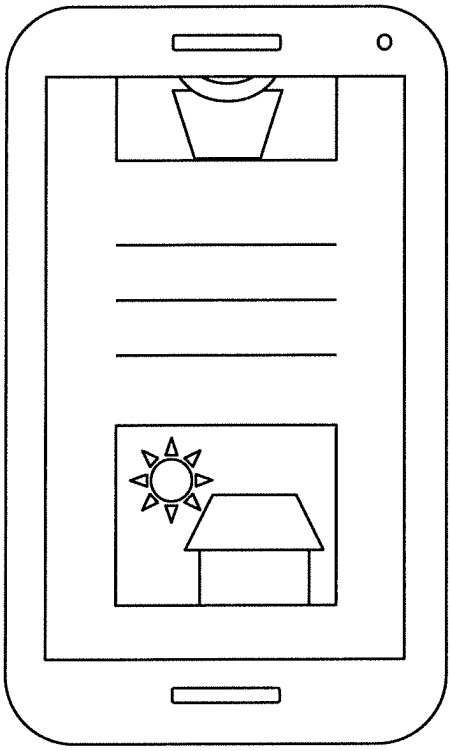


FIG. 27

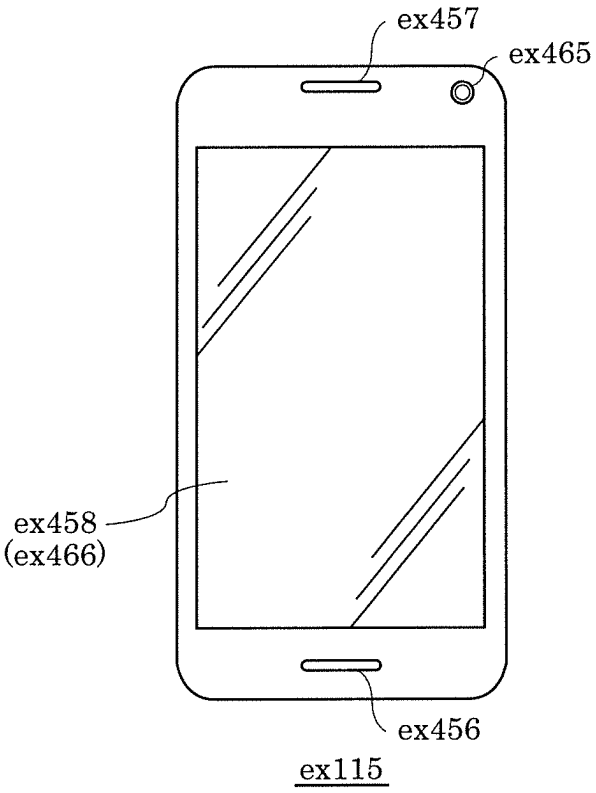
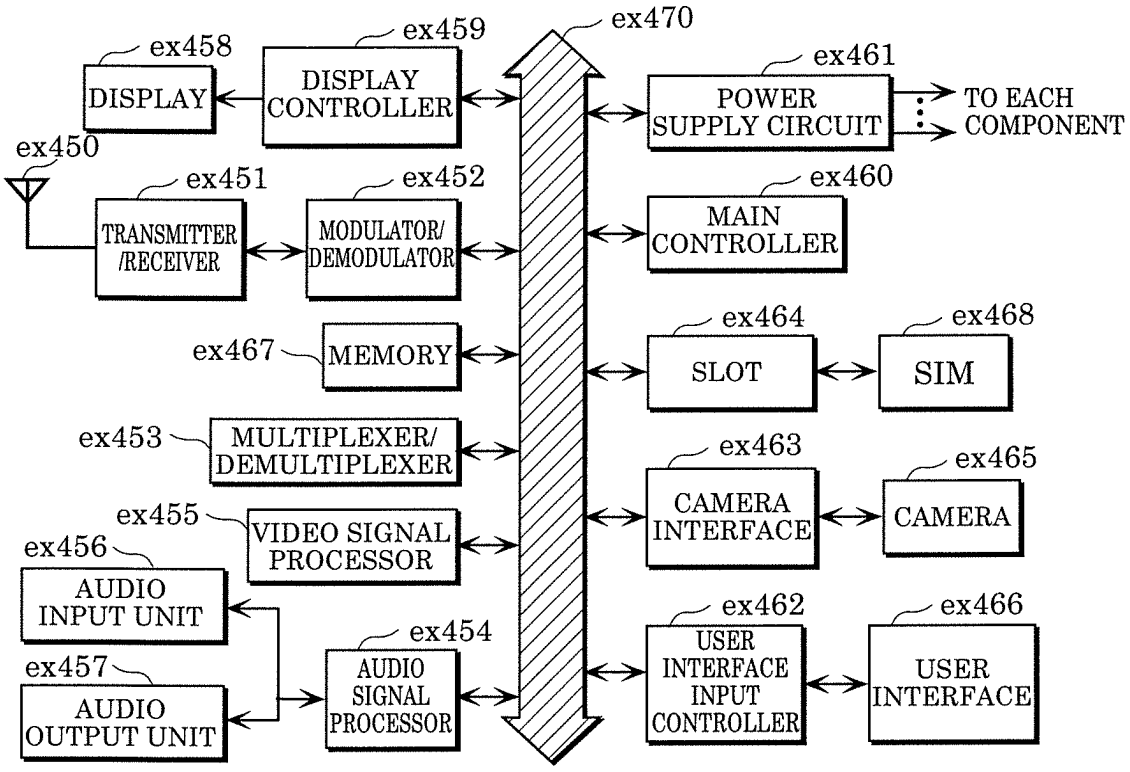


FIG. 28


ex470
ex458
DISPLAY
ex459
DISPLAY CONTROLLER
ex461
POWER SUPPLY CIRCUIT
TO EACH COMPONENT
ex450
ex451
TRANSMITTER /RECEIVER
ex452
MODULATOR/ DEMODULATOR
ex460
MAIN CONTROLLER
ex467
MEMORY
ex464
SLOT
ex468
SIM
ex453
MULTIPLEXER/ DEMULTIPLEXER
ex463
CAMERA INTERFACE
ex465
CAMERA
ex455
VIDEO SIGNAL PROCESSOR
ex456
AUDIO INPUT UNIT
ex462
USER INTERFACE INPUT CONTROLLER
ex466
USER INTERFACE
ex454
AUDIO SIGNAL PROCESSOR
ex457
AUDIO OUTPUT UNIT

ENCODER, DECODER, ENCODING METHOD, AND DECODING METHOD

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 17/550,210 filed on Dec. 14, 2021, which is a continuation of U.S. application Ser. No. 17/081,267, now U.S. Pat. No. 11,240,530, filed on Oct. 27, 2020, which is a continuation of U.S. application Ser. No. 16/816,733, now U.S. Pat. No. 10,856,005, filed on Mar. 12, 2020, which is a continuation application of PCT International Patent Application Number PCT/JP2018/039740 filed on Oct. 25, 2018, claiming the benefit of priority of U.S. Provisional Application No. 62/577,955 filed on Oct. 27, 2017. The entire disclosures of the above-identified applications, including the specifications, drawings, and claims are incorporated herein by reference in their entirety.

BACKGROUND

1. Technical Field

The present disclosure relates to an encoder, a decoder, an encoding method, and a decoding method.

2. Description of the Related Art

There has been H.265 as the video coding standard. H.265 is also referred to as High-Efficiency Video Coding (HEVC).

SUMMARY

An encoder according to an aspect of the present disclosure includes circuitry and memory. Using the memory, the circuitry, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block.

Note that these general and specific aspects may be implemented using a system, a device, a method, an integrated circuit, a computer program, a computer-readable recording medium such as a CD-ROM, or any combination of systems, devices, methods, integrated circuits, computer programs or recording media.

BRIEF DESCRIPTION OF DRAWINGS

These and other objects, advantages and features of the disclosure will become apparent from the following description thereof taken in conjunction with the accompanying drawings that illustrate a specific embodiment of the present disclosure.

FIG. **1** is a block diagram illustrating a functional configuration of an encoder according to Embodiment 1;

FIG. **2** illustrates one example of block splitting according to Embodiment 1;

FIG. **3** is a chart indicating transform basis functions for each transform type;

FIG. **4**A illustrates one example of a filter shape used in ALF;

FIG. **4**B illustrates another example of a filter shape used in ALF;

FIG. **4**C illustrates another example of a filter shape used in ALF;

FIG. **5**A illustrates 67 intra prediction modes used in intra prediction;

FIG. **5**B is a flow chart for illustrating an outline of a prediction image correction process performed via OBMC processing;

FIG. **5**C is a conceptual diagram for illustrating an outline of a prediction image correction process performed via OBMC processing;

FIG. **5**D illustrates one example of FRUC;

FIG. **6** is for illustrating pattern matching (bilateral matching) between two blocks along a motion trajectory;

FIG. **7** is for illustrating pattern matching (template matching) between a template in the current picture and a block in a reference picture;

FIG. **8** is for illustrating a model assuming uniform linear motion;

FIG. **9**A is for illustrating deriving a motion vector of each sub-block based on motion vectors of neighboring blocks;

FIG. **9**B is for illustrating an outline of a process for deriving a motion vector via merge mode;

FIG. **9**C is a conceptual diagram for illustrating an outline of DMVR processing;

FIG. **9**D is for illustrating an outline of a prediction image generation method using a luminance correction process performed via LIC processing;

FIG. **10** is a block diagram illustrating a functional configuration of a decoder according to Embodiment 1;

FIG. **11** is a conceptual diagram for illustrating affine inter mode of affine motion compensation prediction;

FIG. **12**A is a conceptual diagram for illustrating affine merge mode of affine motion compensation prediction;

FIG. **12**B is a conceptual diagram for illustrating affine merge mode of affine motion compensation prediction;

FIG. **13** is a block diagram illustrating an internal configuration for affine motion compensation prediction in the inter predictor included in the encoder according to Embodiment 1;

FIG. **14** is a flow chart for illustrating the first processing procedure for affine inter mode of affine motion compensation performed by the inter predictor in the encoder according to Embodiment 1;

FIG. **15** is a flow chart for illustrating the second processing procedure for affine inter mode of affine motion compensation performed by the inter predictor in the encoder according to Embodiment 1;

FIG. **16** is a flow chart for illustrating the first processing procedure for affine merge mode of affine motion compensation performed by the inter predictor in the encoder according to Embodiment 1;

FIG. **17** is a flow chart for illustrating the second processing procedure for affine merge mode of affine motion compensation performed by the inter predictor in the encoder according to Embodiment 1;

FIG. **18** is a block diagram illustrating an implementation example of the encoder according to Embodiment 1;

FIG. **19** is a flow chart for illustrating an operation example of the encoder according to Embodiment 1;

FIG. **20** is a block diagram illustrating an implementation example of the decoder according to Embodiment 1;

FIG. **21** is a flow chart for illustrating an operation example of the decoder according to Embodiment 1;

FIG. **22** illustrates an overall configuration of a content providing system for implementing a content distribution service;

FIG. **23** illustrates one example of an encoding structure in scalable encoding;

FIG. **24** illustrates one example of an encoding structure in scalable encoding;

FIG. **25** illustrates an example of a display screen of a web page;

FIG. **26** illustrates an example of a display screen of a web page;

FIG. **27** illustrates one example of a smartphone; and

FIG. **28** is a block diagram illustrating a configuration example of a smartphone.

DETAILED DESCRIPTION OF THE EMBODIMENTS

For example, an encoder according to an aspect of the present disclosure includes circuitry and memory. Using the memory, the circuitry, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block.

With this, the encoder can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

Here, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is placed so that a variation between a control point motion vector at a top left corner and a control point motion vector at a top right corner of the current block in the affine motion compensation prediction falls within a predetermined range.

With this, the encoder can reduce the variation between the control point motion vectors in affine motion compensation prediction, and thus the affine motion compensation prediction is more likely to be selected in the inter perdition. Accordingly, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is newly determined for each current picture, or an appropriate set is selected for each current picture from among a plurality of predetermined sets of motion estimation ranges or a plurality of predetermined sets of motion compensation ranges.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be placed at predetermined intervals or using the predetermined sets, and thus it is possible to reduce the amount of processing, i.e. the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is changed in accordance with a type of a reference picture.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be placed at predetermined intervals or using the predetermined sets, and thus it is possible to reduce the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is determined for each predetermined profile and level.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be determined for each predetermined profile and level, and thus it is possible to reduce the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is determined in accordance with motion-estimation processing power depending on computational processing power on an encoder side or memory bandwidth.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be determined in accordance with the estimation processing power, and thus it is possible not only to reduce the amount of processing in accordance with the estimation processing power but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed includes a limit on reference pictures in addition to a limit on a range of referable pixels.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, information about the limit on the range within which the motion estimation or the motion compensation is performed is included in header information of a video parameter set (VPS), a sequence parameter set (SPS), or a picture parameter set (PPS) of an encoded bitstream.

With this, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, for example, in the affine motion compensation prediction, the information about the limit on the range within which the motion estimation or the motion compensation is performed includes information for limiting reference pictures in addition to information for limiting a range of referable pixels.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in the motion estimation for affine inter mode, when a region referred to by a motion vector derived from a current control point motion vector to be evaluated is outside the range within which the motion estimation is performed, the current control point motion vector is eliminated from candidates on which the motion estimation is performed.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in affine inter mode, when a region referred to by a motion vector derived from a control point

motion vector predictor obtained from surrounding encoded blocks neighboring the current block is outside the range within which the motion estimation is performed, encoding is prevented from being performed in the affine inter mode.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in affine merge mode, when a region referred to by a motion vector derived from a control point motion vector obtained from surrounding encoded blocks neighboring the current block is outside the range within which the motion compensation is performed, encoding is prevented from being performed in the affine merge mode.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, a decoder according to an aspect of the present disclosure includes circuitry and memory. Using the memory, the circuitry, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block to decode an encoded stream.

With this, the decoder can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation.

Here, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is placed so that a variation between a control point motion vector at a top left corner and a control point motion vector at a top right corner of the current block in the affine motion compensation prediction falls within a predetermined range.

With this, the decoder can reduce the variation between the control point motion vectors in affine motion compensation prediction, and thus the affine motion compensation prediction is more likely to be selected in the inter perdition. Accordingly, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is newly determined for each current picture, or an appropriate set is selected for each current picture from among a plurality of predetermined sets of motion estimation ranges or a plurality of predetermined sets of motion compensation ranges.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be placed at predetermined intervals or using the predetermined sets, and thus it is possible to reduce the amount of processing, i.e. the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is changed in accordance with a type of a reference picture.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be placed at predetermined intervals or using the predetermined sets, and thus it is possible to reduce the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is determined for each predetermined profile and level.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be determined for each predetermined profile and level, and thus it is possible to reduce the memory bandwidth required for the external memory.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed is determined in accordance with motion-estimation processing power depending on computational processing power on an encoder side or memory bandwidth.

With this, the limit on the range within which the motion estimation or the motion compensation is performed can be determined in accordance with the estimation processing power, and thus it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, the limit on the range within which the motion estimation or the motion compensation is performed includes a limit on reference pictures in addition to a limit on a range of referable pixels.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, information about the limit on the range within which the motion estimation or the motion compensation is performed is included in header information of a video parameter set (VPS), a sequence parameter set (SPS), or a picture parameter set (PPS) of an encoded bitstream.

With this, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, for example, in the affine motion compensation prediction, the information about the limit on the range within which the motion estimation or the motion compensation is performed includes information for limiting reference pictures in addition to information for limiting a range of referable pixels.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in the motion estimation for affine inter mode, when a region referred to by a motion vector derived from a current control point motion vector to be evaluated is outside the range within which the motion estimation is performed, the current control point motion vector is eliminated from candidates on which the motion estimation is performed.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in affine inter mode, when a region referred to by a motion vector derived from a control point motion vector predictor obtained from surrounding decoded blocks neighboring the current block is outside the range within which the motion estimation is performed, encoding is prevented from being performed in the affine inter mode.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, in the affine motion compensation prediction, in affine merge mode, when a region referred to by a motion vector derived from a control point motion vector obtained from surrounding decoded blocks neighboring the current block is outside the range within which the motion compensation is performed, encoding is prevented from being performed in the affine merge mode.

With this, it is possible not only to reduce the memory bandwidth required for the external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, for example, an encoding method according to an aspect of the present disclosure includes in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block.

With this, a device and others using the encoding method can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

Furthermore, for example, a decoding method according to an aspect of the present disclosure includes, in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block to decode an encoded stream.

With this, a device and others using the decoding method can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

Furthermore, these general and specific aspects may be implemented using a system, a device, a method, an integrated circuit, a computer program, a non-transitory computer-readable recording medium such as a CD-ROM, or any combination of systems, devices, methods, integrated circuits, computer programs or recording media.

Hereinafter, embodiment(s) will be described with reference to the drawings.

Note that the embodiment(s) described below each show a general or specific example. The numerical values, shapes, materials, components, the arrangement and connection of the components, steps, order of the steps, etc., indicated in the following embodiment(s) are mere examples, and therefore are not intended to limit the scope of the claims. Therefore, among the components in the following embodiment(s), those not recited in any of the independent claims defining the broadest inventive concepts are described as optional components.

Embodiment 1

First, an outline of Embodiment 1 will be presented. Embodiment 1 is one example of an encoder and a decoder to which the processes and/or configurations presented in subsequent description of aspects of the present disclosure are applicable. Note that Embodiment 1 is merely one example of an encoder and a decoder to which the processes and/or configurations presented in the description of aspects of the present disclosure are applicable. The processes and/or configurations presented in the description of aspects of the present disclosure can also be implemented in an encoder and a decoder different from those according to Embodiment 1.

When the processes and/or configurations presented in the description of aspects of the present disclosure are applied to Embodiment 1, for example, any of the following may be performed.

(1) regarding the encoder or the decoder according to Embodiment 1, among components included in the encoder or the decoder according to Embodiment 1, substituting a component corresponding to a component presented in the description of aspects of the present disclosure with a component presented in the description of aspects of the present disclosure;

(2) regarding the encoder or the decoder according to Embodiment 1, implementing discretionary changes to functions or implemented processes performed by one or more components included in the encoder or the decoder according to Embodiment 1, such as addition, substitution, or removal, etc., of such functions or implemented processes, then substituting a component corresponding to a component presented in the description of aspects of the present disclosure with a component presented in the description of aspects of the present disclosure;

(3) regarding the method implemented by the encoder or the decoder according to Embodiment 1, implementing discretionary changes such as addition of processes and/or substitution, removal of one or more of the processes included in the method, and then substituting a processes corresponding to a process presented in the description of aspects of the present disclosure with a process presented in the description of aspects of the present disclosure;

(4) combining one or more components included in the encoder or the decoder according to Embodiment 1 with a component presented in the description of aspects of the present disclosure, a component including one or more functions included in a component presented in the description of aspects of the present disclosure, or a component that implements one or more processes implemented by a component presented in the description of aspects of the present disclosure;

(5) combining a component including one or more functions included in one or more components included in the encoder or the decoder according to Embodiment 1, or a component that implements one or more processes implemented by one or more components included in the encoder or the decoder according to Embodiment 1 with a component presented in the description of aspects of the present disclosure, a component including one or more functions included in a component presented in the description of aspects of the present disclosure, or a component that implements one or more processes implemented by a component presented in the description of aspects of the present disclosure;

(6) regarding the method implemented by the encoder or the decoder according to Embodiment 1, among processes included in the method, substituting a process corresponding to a process presented in the description of aspects of the present disclosure with a process presented in the description of aspects of the present disclosure; and

(7) combining one or more processes included in the method implemented by the encoder or the decoder according to Embodiment 1 with a process presented in the description of aspects of the present disclosure.

Note that the implementation of the processes and/or configurations presented in the description of aspects of the present disclosure is not limited to the above examples. For example, the processes and/or configurations presented in the description of aspects of the present disclosure may be implemented in a device used for a purpose different from the moving picture/picture encoder or the moving picture/picture decoder disclosed in Embodiment 1. Moreover, the processes and/or configurations presented in the description of aspects of the present disclosure may be independently implemented. Moreover, processes and/or configurations described in different aspects may be combined.

Encoder Outline

First, the encoder according to Embodiment 1 will be outlined. FIG. **1** is a block diagram illustrating a functional configuration of encoder **100** according to Embodiment 1. Encoder **100** is a moving picture/picture encoder that encodes a moving picture/picture block by block.

As illustrated in FIG. **1**, encoder **100** is a device that encodes a picture block by block, and includes splitter **102**, subtractor **104**, transformer **106**, quantizer **108**, entropy encoder **110**, inverse quantizer **112**, inverse transformer **114**, adder **116**, block memory **118**, loop filter **120**, frame memory **122**, intra predictor **124**, inter predictor **126**, and prediction controller **128**.

Encoder **100** is realized as, for example, a generic processor and memory. In this case, when a software program stored in the memory is executed by the processor, the processor functions as splitter **102**, subtractor **104**, transformer **106**, quantizer **108**, entropy encoder **110**, inverse quantizer **112**, inverse transformer **114**, adder **116**, loop filter **120**, intra predictor **124**, inter predictor **126**, and prediction controller **128**. Alternatively, encoder **100** may be realized as one or more dedicated electronic circuits corresponding to splitter **102**, subtractor **104**, transformer **106**, quantizer **108**, entropy encoder **110**, inverse quantizer **112**, inverse transformer **114**, adder **116**, loop filter **120**, intra predictor **124**, inter predictor **126**, and prediction controller **128**.

Hereinafter, each component included in encoder **100** will be described.

Splitter

Splitter **102** splits each picture included in an input moving picture into blocks, and outputs each block to subtractor **104**. For example, splitter **102** first splits a picture into blocks of a fixed size (for example, 128×128). The fixed size block is also referred to as coding tree unit (CTU). Splitter **102** then splits each fixed size block into blocks of variable sizes (for example, 64×64 or smaller), based on recursive quadtree and/or binary tree block splitting. The variable size block is also referred to as a coding unit (CU), a prediction unit (PU), or a transform unit (TU). Note that in this embodiment, there is no need to differentiate between CU, PU, and TU; all or some of the blocks in a picture may be processed per CU, PU, or TU.

FIG. **2** illustrates one example of block splitting according to Embodiment 1. In FIG. **2**, the solid lines represent block boundaries of blocks split by quadtree block splitting, and the dashed lines represent block boundaries of blocks split by binary tree block splitting.

Here, block **10** is a square 128×128 pixel block (128×128 block). This 128×128 block **10** is first split into four square 64×64 blocks (quadtree block splitting).

The top left 64×64 block is further vertically split into two rectangle 32×64 blocks, and the left 32×64 block is further vertically split into two rectangle 16×64 blocks (binary tree block splitting). As a result, the top left 64×64 block is split into two 16×64 blocks **11** and **12** and one 32×64 block **13**.

The top right 64×64 block is horizontally split into two rectangle 64×32 blocks **14** and **15** (binary tree block splitting).

The bottom left 64×64 block is first split into four square 32×32 blocks (quadtree block splitting). The top left block and the bottom right block among the four 32×32 blocks are further split. The top left 32×32 block is vertically split into two rectangle 16×32 blocks, and the right 16×32 block is further horizontally split into two 16×16 blocks (binary tree block splitting). The bottom right 32×32 block is horizontally split into two 32×16 blocks (binary tree block splitting). As a result, the bottom left 64×64 block is split into 16×32 block **16**, two 16×16 blocks **17** and **18**, two 32×32 blocks **19** and **20**, and two 32×16 blocks **21** and **22**.

The bottom right 64×64 block **23** is not split.

As described above, in FIG. **2**, block **10** is split into 13 variable size blocks **11** through **23** based on recursive quadtree and binary tree block splitting. This type of splitting is also referred to as quadtree plus binary tree (QTBT) splitting.

Note that in FIG. **2**, one block is split into four or two blocks (quadtree or binary tree block splitting), but splitting is not limited to this example. For example, one block may be split into three blocks (ternary block splitting). Splitting including such ternary block splitting is also referred to as multi-type tree (MBT) splitting.

Subtractor

Subtractor **104** subtracts a prediction signal (prediction sample) from an original signal (original sample) per block

split by splitter **102**. In other words, subtractor **104** calculates prediction errors (also referred to as residuals) of a block to be encoded (hereinafter referred to as a current block). Subtractor **104** then outputs the calculated prediction errors to transformer **106**.

The original signal is a signal input into encoder **100**, and is a signal representing an image for each picture included in a moving picture (for example, a luma signal and two chroma signals). Hereinafter, a signal representing an image is also referred to as a sample.

Transformer

Transformer **106** transforms spatial domain prediction errors into frequency domain transform coefficients, and outputs the transform coefficients to quantizer **108**. More specifically, transformer **106** applies, for example, a predefined discrete cosine transform (DCT) or discrete sine transform (DST) to spatial domain prediction errors.

Note that transformer **106** may adaptively select a transform type from among a plurality of transform types, and transform prediction errors into transform coefficients by using a transform basis function corresponding to the selected transform type. This sort of transform is also referred to as explicit multiple core transform (EMT) or adaptive multiple transform (AMT).

The transform types include, for example, DCT-II, DCT-V, DCT-VIII, DST-I, and DST-VII. FIG. **3** is a chart indicating transform basis functions for each transform type. In FIG. **3**, N indicates the number of input pixels. For example, selection of a transform type from among the plurality of transform types may depend on the prediction type (intra prediction and inter prediction), and may depend on intra prediction mode.

Information indicating whether to apply such EMT or AMT (referred to as, for example, an AMT flag) and information indicating the selected transform type is signalled at the CU level. Note that the signaling of such information need not be performed at the CU level, and may be performed at another level (for example, at the sequence level, picture level, slice level, tile level, or CTU level).

Moreover, transformer **106** may apply a secondary transform to the transform coefficients (transform result). Such a secondary transform is also referred to as adaptive secondary transform (AST) or non-separable secondary transform (NSST). For example, transformer **106** applies a secondary transform to each sub-block (for example, each 4×4 sub-block) included in the block of the transform coefficients corresponding to the intra prediction errors. Information indicating whether to apply NSST and information related to the transform matrix used in NSST are signalled at the CU level. Note that the signaling of such information need not be performed at the CU level, and may be performed at another level (for example, at the sequence level, picture level, slice level, tile level, or CTU level).

Here, a separable transform is a method in which a transform is performed a plurality of times by separately performing a transform for each direction according to the number of dimensions input. A non-separable transform is a method of performing a collective transform in which two or more dimensions in a multidimensional input are collectively regarded as a single dimension.

In one example of a non-separable transform, when the input is a 4×4 block, the 4×4 block is regarded as a single array including 16 components, and the transform applies a 16×16 transform matrix to the array.

Moreover, similar to above, after an input 4×4 block is regarded as a single array including 16 components, a transform that performs a plurality of Givens rotations on the array (i.e., a Hypercube-Givens Transform) is also one example of a non-separable transform.

Quantizer

Quantizer **108** quantizes the transform coefficients output from transformer **106**. More specifically, quantizer **108** scans, in a predetermined scanning order, the transform coefficients of the current block, and quantizes the scanned transform coefficients based on quantization parameters (QP) corresponding to the transform coefficients. Quantizer **108** then outputs the quantized transform coefficients (hereinafter referred to as quantized coefficients) of the current block to entropy encoder **110** and inverse quantizer **112**.

A predetermined order is an order for quantizing/inverse quantizing transform coefficients. For example, a predetermined scanning order is defined as ascending order of frequency (from low to high frequency) or descending order of frequency (from high to low frequency).

A quantization parameter is a parameter defining a quantization step size (quantization width). For example, if the value of the quantization parameter increases, the quantization step size also increases. In other words, if the value of the quantization parameter increases, the quantization error increases.

Entropy Encoder

Entropy encoder **110** generates an encoded signal (encoded bitstream) by variable length encoding quantized coefficients, which are inputs from quantizer **108**. More specifically, entropy encoder **110**, for example, binarizes quantized coefficients and arithmetic encodes the binary signal.

Inverse Quantizer

Inverse quantizer **112** inverse quantizes quantized coefficients, which are inputs from quantizer **108**. More specifically, inverse quantizer **112** inverse quantizes, in a predetermined scanning order, quantized coefficients of the current block. Inverse quantizer **112** then outputs the inverse quantized transform coefficients of the current block to inverse transformer **114**.

Inverse Transformer

Inverse transformer **114** restores prediction errors by inverse transforming transform coefficients, which are inputs from inverse quantizer **112**. More specifically, inverse transformer **114** restores the prediction errors of the current block by applying an inverse transform corresponding to the transform applied by transformer **106** on the transform coefficients. Inverse transformer **114** then outputs the restored prediction errors to adder **116**.

Note that since information is lost in quantization, the restored prediction errors do not match the prediction errors calculated by subtractor **104**. In other words, the restored prediction errors include quantization errors.

Adder

Adder **116** reconstructs the current block by summing prediction errors, which are inputs from inverse transformer

114, and prediction samples, which are inputs from prediction controller 128. Adder 116 then outputs the reconstructed block to block memory 118 and loop filter 120. A reconstructed block is also referred to as a local decoded block.

Block Memory

Block memory 118 is storage for storing blocks in a picture to be encoded (hereinafter referred to as a current picture) for reference in intra prediction. More specifically, block memory 118 stores reconstructed blocks output from adder 116.

Loop Filter

Loop filter 120 applies a loop filter to blocks reconstructed by adder 116, and outputs the filtered reconstructed blocks to frame memory 122. A loop filter is a filter used in an encoding loop (in-loop filter), and includes, for example, a deblocking filter (DF), a sample adaptive offset (SAO), and an adaptive loop filter (ALF).

In ALF, a least square error filter for removing compression artifacts is applied. For example, one filter from among a plurality of filters is selected for each 2×2 sub-block in the current block based on direction and activity of local gradients, and is applied.

More specifically, first, each sub-block (for example, each 2×2 sub-block) is categorized into one out of a plurality of classes (for example, 15 or 25 classes). The classification of the sub-block is based on gradient directionality and activity. For example, classification index C is derived based on gradient directionality D (for example, 0 to 2 or 0 to 4) and gradient activity A (for example, 0 to 4) (for example, C=5 D+A). Then, based on classification index C, each sub-block is categorized into one out of a plurality of classes (for example, 15 or 25 classes).

For example, gradient directionality D is calculated by comparing gradients of a plurality of directions (for example, the horizontal, vertical, and two diagonal directions). Moreover, for example, gradient activity A is calculated by summing gradients of a plurality of directions and quantizing the sum.

The filter to be used for each sub-block is determined from among the plurality of filters based on the result of such categorization.

The filter shape to be used in ALF is, for example, a circular symmetric filter shape. FIG. 4A through FIG. 4C illustrate examples of filter shapes used in ALF. FIG. 4A illustrates a 5×5 diamond shape filter, FIG. 4B illustrates a 7×7 diamond shape filter, and FIG. 4C illustrates a 9×9 diamond shape filter. Information indicating the filter shape is signalled at the picture level. Note that the signaling of information indicating the filter shape need not be performed at the picture level, and may be performed at another level (for example, at the sequence level, slice level, tile level, CTU level, or CU level).

The enabling or disabling of ALF is determined at the picture level or CU level. For example, for luma, the decision to apply ALF or not is done at the CU level, and for chroma, the decision to apply ALF or not is done at the picture level. Information indicating whether ALF is enabled or disabled is signalled at the picture level or CU level. Note that the signaling of information indicating whether ALF is enabled or disabled need not be performed at the picture level or CU level, and may be performed at another level (for example, at the sequence level, slice level, tile level, or CTU level).

The coefficients set for the plurality of selectable filters (for example, 15 or 25 filters) is signalled at the picture level. Note that the signaling of the coefficients set need not be performed at the picture level, and may be performed at another level (for example, at the sequence level, slice level, tile level, CTU level, CU level, or sub-block level).

Frame Memory

Frame memory 122 is storage for storing reference pictures used in inter prediction, and is also referred to as a frame buffer. More specifically, frame memory 122 stores reconstructed blocks filtered by loop filter 120.

Intra Predictor

Intra predictor 124 generates a prediction signal (intra prediction signal) by intra predicting the current block with reference to a block or blocks in the current picture and stored in block memory 118 (also referred to as intra frame prediction). More specifically, intra predictor 124 generates an intra prediction signal by intra prediction with reference to samples (for example, luma and/or chroma values) of a block or blocks neighboring the current block, and then outputs the intra prediction signal to prediction controller 128.

For example, intra predictor 124 performs intra prediction by using one mode from among a plurality of predefined intra prediction modes. The intra prediction modes include one or more non-directional prediction modes and a plurality of directional prediction modes.

The one or more non-directional prediction modes include, for example, planar prediction mode and DC prediction mode defined in the H.265/high-efficiency video coding (HEVC) standard (see H.265 (ISO/IEC 23008-2 HEVC (High Efficiency Video Coding))).

The plurality of directional prediction modes include, for example, the 33 directional prediction modes defined in the H.265/HEVC standard. Note that the plurality of directional prediction modes may further include 32 directional prediction modes in addition to the 33 directional prediction modes (for a total of 65 directional prediction modes). FIG. 5A illustrates 67 intra prediction modes used in intra prediction (two non-directional prediction modes and 65 directional prediction modes). The solid arrows represent the 33 directions defined in the H.265/HEVC standard, and the dashed arrows represent the additional 32 directions.

Note that a luma block may be referenced in chroma block intra prediction. In other words, a chroma component of the current block may be predicted based on a luma component of the current block. Such intra prediction is also referred to as cross-component linear model (CCLM) prediction. Such a chroma block intra prediction mode that references a luma block (referred to as, for example, CCLM mode) may be added as one of the chroma block intra prediction modes.

Intra predictor 124 may correct post-intra-prediction pixel values based on horizontal/vertical reference pixel gradients. Intra prediction accompanied by this sort of correcting is also referred to as position dependent intra prediction combination (PDPC). Information indicating whether to apply PDPC or not (referred to as, for example, a PDPC flag) is, for example, signalled at the CU level. Note that the signaling of this information need not be performed at the CU level, and may be performed at another level (for example, on the sequence level, picture level, slice level, tile level, or CTU level).

Inter Predictor

Inter predictor **126** generates a prediction signal (inter prediction signal) by inter predicting the current block with reference to a block or blocks in a reference picture, which is different from the current picture and is stored in frame memory **122** (also referred to as inter frame prediction). Inter prediction is performed per current block or per sub-block (for example, per 4×4 block) in the current block. For example, inter predictor **126** performs motion estimation in a reference picture for the current block or sub-block. Inter predictor **126** then generates an inter prediction signal of the current block or sub-block by motion compensation by using motion information (for example, a motion vector) obtained from motion estimation. Inter predictor **126** then outputs the generated inter prediction signal to prediction controller **128**.

The motion information used in motion compensation is signalled. A motion vector predictor may be used for the signaling of the motion vector. In other words, the difference between the motion vector and the motion vector predictor may be signalled.

Note that the inter prediction signal may be generated using motion information for a neighboring block in addition to motion information for the current block obtained from motion estimation. More specifically, the inter prediction signal may be generated per sub-block in the current block by calculating a weighted sum of a prediction signal based on motion information obtained from motion estimation and a prediction signal based on motion information for a neighboring block. Such inter prediction (motion compensation) is also referred to as overlapped block motion compensation (OBMC).

In such an OBMC mode, information indicating sub-block size for OBMC (referred to as, for example, OBMC block size) is signalled at the sequence level. Moreover, information indicating whether to apply the OBMC mode or not (referred to as, for example, an OBMC flag) is signalled at the CU level. Note that the signaling of such information need not be performed at the sequence level and CU level, and may be performed at another level (for example, at the picture level, slice level, tile level, CTU level, or sub-block level).

Hereinafter, the OBMC mode will be described in further detail. FIG. **5**B is a flowchart and FIG. **5**C is a conceptual diagram for illustrating an outline of a prediction image correction process performed via OBMC processing.

First, a prediction image (Pred) is obtained through typical motion compensation using a motion vector (MV) assigned to the current block.

Next, a prediction image (Pred_L) is obtained by applying a motion vector (MV_L) of the encoded neighboring left block to the current block, and a first pass of the correction of the prediction image is made by superimposing the prediction image and Pred_L.

Similarly, a prediction image (Pred_U) is obtained by applying a motion vector (MV_U) of the encoded neighboring upper block to the current block, and a second pass of the correction of the prediction image is made by superimposing the prediction image resulting from the first pass and Pred_U. The result of the second pass is the final prediction image.

Note that the above example is of a two-pass correction method using the neighboring left and upper blocks, but the method may be a three-pass or higher correction method that also uses the neighboring right and/or lower block.

Note that the region subject to superimposition may be the entire pixel region of the block, and, alternatively, may be a partial block boundary region.

Note that here, the prediction image correction process is described as being based on a single reference picture, but the same applies when a prediction image is corrected based on a plurality of reference pictures. In such a case, after corrected prediction images resulting from performing correction based on each of the reference pictures are obtained, the obtained corrected prediction images are further superimposed to obtain the final prediction image.

Note that the unit of the current block may be a prediction block and, alternatively, may be a sub-block obtained by further dividing the prediction block.

One example of a method for determining whether to implement OBMC processing is by using an obmc_flag, which is a signal that indicates whether to implement OBMC processing. As one specific example, the encoder determines whether the current block belongs to a region including complicated motion. The encoder sets the obmc_flag to a value of "1" when the block belongs to a region including complicated motion and implements OBMC processing when encoding, and sets the obmc_flag to a value of "0" when the block does not belong to a region including complication motion and encodes without implementing OBMC processing. The decoder switches between implementing OBMC processing or not by decoding the obmc_flag written in the stream and performing the decoding in accordance with the flag value.

Note that the motion information may be derived on the decoder side without being signalled. For example, a merge mode defined in the H.265/HEVC standard may be used. Moreover, for example, the motion information may be derived by performing motion estimation on the decoder side. In this case, motion estimation is performed without using the pixel values of the current block.

Here, a mode for performing motion estimation on the decoder side will be described. A mode for performing motion estimation on the decoder side is also referred to as pattern matched motion vector derivation (PMMVD) mode or frame rate up-conversion (FRUC) mode.

One example of FRUC processing is illustrated in FIG. **5**D. First, a candidate list (a candidate list may be a merge list) of candidates each including a motion vector predictor is generated with reference to motion vectors of encoded blocks that spatially or temporally neighbor the current block. Next, the best candidate MV is selected from among a plurality of candidate MVs registered in the candidate list. For example, evaluation values for the candidates included in the candidate list are calculated and one candidate is selected based on the calculated evaluation values.

Next, a motion vector for the current block is derived from the motion vector of the selected candidate. More specifically, for example, the motion vector for the current block is calculated as the motion vector of the selected candidate (best candidate MV), as-is. Alternatively, the motion vector for the current block may be derived by pattern matching performed in the vicinity of a position in a reference picture corresponding to the motion vector of the selected candidate. In other words, when the vicinity of the best candidate MV is searched via the same method and an MV having a better evaluation value is found, the best candidate MV may be updated to the MV having the better evaluation value, and the MV having the better evaluation value may be used as the final MV for the current block. Note that a configuration in which this processing is not implemented is also acceptable.

The same processes may be performed in cases in which the processing is performed in units of sub-blocks.

Note that an evaluation value is calculated by calculating the difference in the reconstructed image by pattern matching performed between a region in a reference picture corresponding to a motion vector and a predetermined region. Note that the evaluation value may be calculated by using some other information in addition to the difference.

The pattern matching used is either first pattern matching or second pattern matching. First pattern matching and second pattern matching are also referred to as bilateral matching and template matching, respectively.

In the first pattern matching, pattern matching is performed between two blocks along the motion trajectory of the current block in two different reference pictures. Therefore, in the first pattern matching, a region in another reference picture conforming to the motion trajectory of the current block is used as the predetermined region for the above-described calculation of the candidate evaluation value.

FIG. **6** is for illustrating one example of pattern matching (bilateral matching) between two blocks along a motion trajectory. As illustrated in FIG. **6**, in the first pattern matching, two motion vectors (MV**0**, MV**1**) are derived by finding the best match between two blocks along the motion trajectory of the current block (Cur block) in two different reference pictures (Ref**0**, Ref**1**). More specifically, a difference between (i) a reconstructed image in a specified position in a first encoded reference picture (Ref**0**) specified by a candidate MV and (ii) a reconstructed picture in a specified position in a second encoded reference picture (Ref**1**) specified by a symmetrical MV scaled at a display time interval of the candidate MV may be derived, and the evaluation value for the current block may be calculated by using the derived difference. The candidate MV having the best evaluation value among the plurality of candidate MVs may be selected as the final MV.

Under the assumption of continuous motion trajectory, the motion vectors (MV**0**, MV**1**) pointing to the two reference blocks shall be proportional to the temporal distances (TD**0**, TD**1**) between the current picture (Cur Pic) and the two reference pictures (Ref**0**, Ref**1**). For example, when the current picture is temporally between the two reference pictures and the temporal distance from the current picture to the two reference pictures is the same, the first pattern matching derives a mirror based bi-directional motion vector.

In the second pattern matching, pattern matching is performed between a template in the current picture (blocks neighboring the current block in the current picture (for example, the top and/or left neighboring blocks)) and a block in a reference picture. Therefore, in the second pattern matching, a block neighboring the current block in the current picture is used as the predetermined region for the above-described calculation of the candidate evaluation value.

FIG. **7** is for illustrating one example of pattern matching (template matching) between a template in the current picture and a block in a reference picture. As illustrated in FIG. **7**, in the second pattern matching, a motion vector of the current block is derived by searching a reference picture (Ref**0**) to find the block that best matches neighboring blocks of the current block (Cur block) in the current picture (Cur Pic). More specifically, a difference between (i) a reconstructed image of an encoded region that is both or one of the neighboring left and neighboring upper region and (ii) a reconstructed picture in the same position in an encoded reference picture (Ref**0**) specified by a candidate MV may be derived, and the evaluation value for the current block may be calculated by using the derived difference. The candidate MV having the best evaluation value among the plurality of candidate MVs may be selected as the best candidate MV.

Information indicating whether to apply the FRUC mode or not (referred to as, for example, a FRUC flag) is signalled at the CU level. Moreover, when the FRUC mode is applied (for example, when the FRUC flag is set to true), information indicating the pattern matching method (first pattern matching or second pattern matching) is signalled at the CU level. Note that the signaling of such information need not be performed at the CU level, and may be performed at another level (for example, at the sequence level, picture level, slice level, tile level, CTU level, or sub-block level).

Here, a mode for deriving a motion vector based on a model assuming uniform linear motion will be described. This mode is also referred to as a bi-directional optical flow (BIO) mode.

FIG. **8** is for illustrating a model assuming uniform linear motion. In FIG. **8**, (v_x, v_y) denotes a velocity vector, and τ_0 and τ_1 denote temporal distances between the current picture (Cur Pic) and two reference pictures (Ref_0, Ref_1). (MVx_0, MVy_0) denotes a motion vector corresponding to reference picture Ref_0, and (MVx_1, MVy_1) denotes a motion vector corresponding to reference picture Ref_1.

Here, under the assumption of uniform linear motion exhibited by velocity vector (v_x, v_y), (MVx_0, MVy_0) and (MVx_1, MVy_1) are represented as $(v_x\tau_0, v_y\tau_0)$ and $(-v_x\tau_1, -v_y\tau_1)$, respectively, and the following optical flow equation is given.

MATH. 1

$$\partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad (1)$$

Here, $I^{(k)}$ denotes a luma value from reference picture k (k=0, 1) after motion compensation. This optical flow equation shows that the sum of (i) the time derivative of the luma value, (ii) the product of the horizontal velocity and the horizontal component of the spatial gradient of a reference picture, and (iii) the product of the vertical velocity and the vertical component of the spatial gradient of a reference picture is equal to zero. A motion vector of each block obtained from, for example, a merge list is corrected pixel by pixel based on a combination of the optical flow equation and Hermite interpolation.

Note that a motion vector may be derived on the decoder side using a method other than deriving a motion vector based on a model assuming uniform linear motion. For example, a motion vector may be derived for each sub-block based on motion vectors of neighboring blocks.

Here, a mode in which a motion vector is derived for each sub-block based on motion vectors of neighboring blocks will be described. This mode is also referred to as affine motion compensation prediction mode.

FIG. **9**A is for illustrating deriving a motion vector of each sub-block based on motion vectors of neighboring blocks. In FIG. **9**A, the current block includes 16 4×4 sub-blocks. Here, motion vector v_0 of the top left corner control point in the current block is derived based on motion vectors of neighboring sub-blocks, and motion vector v_1 of the top right corner control point in the current block is derived based on motion vectors of neighboring blocks. Then, using

the two motion vectors v_0 and v_1, the motion vector (v_x, v_y) of each sub-block in the current block is derived using Equation 2 below.

MATH. 2

$$\begin{cases} v_x = \frac{(v_{1x} - v_{0x})}{w}x - \frac{(v_{1y} - v_{0y})}{w}y + v_{0x} \\ v_y = \frac{(v_{1y} - v_{0y})}{w}x + \frac{(v_{1x} - v_{0x})}{w}y + v_{0y} \end{cases} \quad (2)$$

Here, x and y are the horizontal and vertical positions of the sub-block, respectively, and w is a predetermined weighted coefficient.

Such an affine motion compensation prediction mode may include a number of modes of different methods of deriving the motion vectors of the top left and top right corner control points. Information indicating such an affine motion compensation prediction mode (referred to as, for example, an affine flag) is signalled at the CU level. Note that the signaling of information indicating the affine motion compensation prediction mode need not be performed at the CU level, and may be performed at another level (for example, at the sequence level, picture level, slice level, tile level, CTU level, or sub-block level).

Prediction Controller

Prediction controller **128** selects either the intra prediction signal or the inter prediction signal, and outputs the selected prediction signal to subtractor **104** and adder **116**.

Here, an example of deriving a motion vector via merge mode in a current picture will be given. FIG. **9**B is for illustrating an outline of a process for deriving a motion vector via merge mode.

First, an MV predictor list in which candidate MV predictors are registered is generated. Examples of candidate MV predictors include: spatially neighboring MV predictors, which are MVs of encoded blocks positioned in the spatial vicinity of the current block; a temporally neighboring MV predictor, which is an MV of a block in an encoded reference picture that neighbors a block in the same location as the current block; a combined MV predictor, which is an MV generated by combining the MV values of the spatially neighboring MV predictor and the temporally neighboring MV predictor; and a zero MV predictor, which is an MV whose value is zero.

Next, the MV of the current block is determined by selecting one MV predictor from among the plurality of MV predictors registered in the MV predictor list.

Furthermore, in the variable-length encoder, a merge_idx, which is a signal indicating which MV predictor is selected, is written and encoded into the stream.

Note that the MV predictors registered in the MV predictor list illustrated in FIG. **9**B constitute one example. The number of MV predictors registered in the MV predictor list may be different from the number illustrated in FIG. **9**B, the MV predictors registered in the MV predictor list may omit one or more of the types of MV predictors given in the example in FIG. **9**B, and the MV predictors registered in the MV predictor list may include one or more types of MV predictors in addition to and different from the types given in the example in FIG. **9**B.

Note that the final MV may be determined by performing DMVR processing (to be described later) by using the MV of the current block derived via merge mode.

Here, an example of determining an MV by using DMVR processing will be given.

FIG. **9**C is a conceptual diagram for illustrating an outline of DMVR processing.

First, the most appropriate MVP set for the current block is considered to be the candidate MV, reference pixels are obtained from a first reference picture, which is a picture processed in the L0 direction in accordance with the candidate MV, and a second reference picture, which is a picture processed in the L1 direction in accordance with the candidate MV, and a template is generated by calculating the average of the reference pixels.

Next, using the template, the surrounding regions of the candidate MVs of the first and second reference pictures are searched, and the MV with the lowest cost is determined to be the final MV. Note that the cost value is calculated using, for example, the difference between each pixel value in the template and each pixel value in the regions searched, as well as the MV value.

Note that the outlines of the processes described here are fundamentally the same in both the encoder and the decoder.

Note that processing other than the processing exactly as described above may be used, so long as the processing is capable of deriving the final MV by searching the surroundings of the candidate MV.

Here, an example of a mode that generates a prediction image by using LIC processing will be given.

FIG. **9**D is for illustrating an outline of a prediction image generation method using a luminance correction process performed via LIC processing.

First, an MV is extracted for obtaining, from an encoded reference picture, a reference image corresponding to the current block.

Next, information indicating how the luminance value changed between the reference picture and the current picture is extracted and a luminance correction parameter is calculated by using the luminance pixel values for the encoded left neighboring reference region and the encoded upper neighboring reference region, and the luminance pixel value in the same location in the reference picture specified by the MV.

The prediction image for the current block is generated by performing a luminance correction process by using the luminance correction parameter on the reference image in the reference picture specified by the MV.

Note that the shape of the surrounding reference region illustrated in FIG. **9**D is just one example; the surrounding reference region may have a different shape.

Moreover, although a prediction image is generated from a single reference picture in this example, in cases in which a prediction image is generated from a plurality of reference pictures as well, the prediction image is generated after performing a luminance correction process, via the same method, on the reference images obtained from the reference pictures.

One example of a method for determining whether to implement LIC processing is by using anlic_flag, which is a signal that indicates whether to implement LIC processing. As one specific example, the encoder determines whether the current block belongs to a region of luminance change. The encoder sets the lic_flag to a value of "1" when the block belongs to a region of luminance change and implements LIC processing when encoding, and sets the lic_flag to a value of "0" when the block does not belong to a region of luminance change and encodes without implementing LIC processing. The decoder switches between implement-

ing LIC processing or not by decoding the lic_flag written in the stream and performing the decoding in accordance with the flag value.

One example of a different method of determining whether to implement LIC processing is determining so in accordance with whether LIC processing was determined to be implemented for a surrounding block. In one specific example, when merge mode is used on the current block, whether LIC processing was applied in the encoding of the surrounding encoded block selected upon deriving the MV in the merge mode processing may be determined, and whether to implement LIC processing or not can be switched based on the result of the determination. Note that in this example, the same applies to the processing performed on the decoder side.

Decoder Outline

Next, a decoder capable of decoding an encoded signal (encoded bitstream) output from encoder **100** will be described. FIG. **10** is a block diagram illustrating a functional configuration of decoder **200** according to Embodiment 1. Decoder **200** is a moving picture/picture decoder that decodes a moving picture/picture block by block.

As illustrated in FIG. **10**, decoder **200** includes entropy decoder **202**, inverse quantizer **204**, inverse transformer **206**, adder **208**, block memory **210**, loop filter **212**, frame memory **214**, intra predictor **216**, inter predictor **218**, and prediction controller **220**.

Decoder **200** is realized as, for example, a generic processor and memory. In this case, when a software program stored in the memory is executed by the processor, the processor functions as entropy decoder **202**, inverse quantizer **204**, inverse transformer **206**, adder **208**, loop filter **212**, intra predictor **216**, inter predictor **218**, and prediction controller **220**. Alternatively, decoder **200** may be realized as one or more dedicated electronic circuits corresponding to entropy decoder **202**, inverse quantizer **204**, inverse transformer **206**, adder **208**, loop filter **212**, intra predictor **216**, inter predictor **218**, and prediction controller **220**.

Hereinafter, each component included in decoder **200** will be described.

Entropy Decoder

Entropy decoder **202** entropy decodes an encoded bitstream. More specifically, for example, entropy decoder **202** arithmetic decodes an encoded bitstream into a binary signal. Entropy decoder **202** then debinarizes the binary signal. With this, entropy decoder **202** outputs quantized coefficients of each block to inverse quantizer **204**.

Inverse Quantizer

Inverse quantizer **204** inverse quantizes quantized coefficients of a block to be decoded (hereinafter referred to as a current block), which are inputs from entropy decoder **202**. More specifically, inverse quantizer **204** inverse quantizes quantized coefficients of the current block based on quantization parameters corresponding to the quantized coefficients. Inverse quantizer **204** then outputs the inverse quantized coefficients (i.e., transform coefficients) of the current block to inverse transformer **206**.

Inverse Transformer

Inverse transformer **206** restores prediction errors by inverse transforming transform coefficients, which are inputs from inverse quantizer **204**.

For example, when information parsed from an encoded bitstream indicates application of EMT or AMT (for example, when the AMT flag is set to true), inverse transformer **206** inverse transforms the transform coefficients of the current block based on information indicating the parsed transform type.

Moreover, for example, when information parsed from an encoded bitstream indicates application of NSST, inverse transformer **206** applies a secondary inverse transform to the transform coefficients.

Adder

Adder **208** reconstructs the current block by summing prediction errors, which are inputs from inverse transformer **206**, and prediction samples, which is an input from prediction controller **220**. Adder **208** then outputs the reconstructed block to block memory **210** and loop filter **212**.

Block Memory

Block memory **210** is storage for storing blocks in a picture to be decoded (hereinafter referred to as a current picture) for reference in intra prediction. More specifically, block memory **210** stores reconstructed blocks output from adder **208**.

Loop Filter

Loop filter **212** applies a loop filter to blocks reconstructed by adder **208**, and outputs the filtered reconstructed blocks to frame memory **214** and, for example, a display device.

When information indicating the enabling or disabling of ALF parsed from an encoded bitstream indicates enabled, one filter from among a plurality of filters is selected based on direction and activity of local gradients, and the selected filter is applied to the reconstructed block.

Frame Memory

Frame memory **214** is storage for storing reference pictures used in inter prediction, and is also referred to as a frame buffer. More specifically, frame memory **214** stores reconstructed blocks filtered by loop filter **212**.

Intra Predictor

Intra predictor **216** generates a prediction signal (intra prediction signal) by intra prediction with reference to a block or blocks in the current picture and stored in block memory **210**. More specifically, intra predictor **216** generates an intra prediction signal by intra prediction with reference to samples (for example, luma and/or chroma values) of a block or blocks neighboring the current block, and then outputs the intra prediction signal to prediction controller **220**.

Note that when an intra prediction mode in which a chroma block is intra predicted from a luma block is selected, intra predictor **216** may predict the chroma component of the current block based on the luma component of the current block.

Moreover, when information indicating the application of PDPC is parsed from an encoded bitstream, intra predictor **216** corrects post-intra-prediction pixel values based on horizontal/vertical reference pixel gradients.

Inter Predictor

Inter predictor **218** predicts the current block with reference to a reference picture stored in frame memory **214**. Inter prediction is performed per current block or per sub-block (for example, per 4×4 block) in the current block. For example, inter predictor **218** generates an inter prediction signal of the current block or sub-block by motion compensation by using motion information (for example, a motion vector) parsed from an encoded bitstream, and outputs the inter prediction signal to prediction controller **220**.

Note that when the information parsed from the encoded bitstream indicates application of OBMC mode, inter predictor **218** generates the inter prediction signal using motion information for a neighboring block in addition to motion information for the current block obtained from motion estimation.

Moreover, when the information parsed from the encoded bitstream indicates application of FRUC mode, inter predictor **218** derives motion information by performing motion estimation in accordance with the pattern matching method (bilateral matching or template matching) parsed from the encoded bitstream. Inter predictor **218** then performs motion compensation using the derived motion information.

Moreover, when BIO mode is to be applied, inter predictor **218** derives a motion vector based on a model assuming uniform linear motion. Moreover, when the information parsed from the encoded bitstream indicates that affine motion compensation prediction mode is to be applied, inter predictor **218** derives a motion vector of each sub-block based on motion vectors of neighboring blocks.

Prediction Controller

Prediction controller **220** selects either the intra prediction signal or the inter prediction signal, and outputs the selected prediction signal to adder **208**.

Description of Affine Inter Mode of Affine Motion Compensation Prediction

As described above, a mode for deriving a motion vector of each sub-block based on motion vectors of neighboring blocks is referred to as affine motion compensation prediction mode. The affine motion compensation prediction mode includes two modes: affine inter mode; and affine merge mode. Hereinafter, the two modes, affine inter mode and affine merge mode, will be described.

FIG. **11** is a conceptual diagram for illustrating affine inter mode of affine motion compensation prediction. FIG. **11** shows a current block, motion vector predictor v_0 of the top left corner control point of the current block, and motion vector predictor v_1 of the top right corner control point of the current block.

In affine inter mode, as shown in FIG. **11**, motion vector predictor v_0 of the top left corner control point is selected from among motion vectors of blocks A, B, and C which are surrounding encoded blocks neighboring the current block. In the same manner, motion vector predictor v_1 of the top right corner control point is selected from among motion vectors of blocks D and E which are surrounding encoded blocks neighboring the current block.

In the encoding, based on the surrounding encoded blocks neighboring the current block, it is determined, using cost evaluation, etc., which motion vector of an encoded block is selected as the motion vector predictor of the control point in affine motion compensation prediction for the current block. Subsequently, a flag indicating which motion vector of an encoded block is selected is written in the stream as the motion vector predictor of the control point.

Furthermore, in the encoding, after the motion vector predictor of the control point in the current block is determined, motion estimation is performed to estimate the motion vector of the control point. Using the estimated motion vector of the control point, an affine motion vector of each sub-block in the current block is calculated from Equation 2 as described above, and motion compensation is performed. While the motion compensation is performed, a difference between the estimated motion vector of the control point and the motion vector predictor of the control point is written in the stream.

Note that the foregoing describes the operation of the encoding, but the same is true of the operation of the decoding.

Description of Affine Merge Mode of Affine Motion Compensation Prediction

FIGS. **12**A and **12**B are each a conceptual diagram for illustrating affine merge mode of affine motion compensation prediction. FIG. **12**A shows a current block, and blocks A to D which are surrounding encoded blocks neighboring the current block. FIG. **12**B shows a processing example of affine merge mode of affine motion compensation prediction. FIG. **12**B also shows the current block, motion vector predictor v_0 of the top left corner control point, motion vector predictor v_1 of the top right corner control point, and block A which is an encoded block.

In affine merge mode, as shown in FIG. **12**A, the surrounding encoded blocks neighboring the current block are checked in the following order: block A (left), block B (top), block C (top right), block D (bottom left), and block E (top left). Subsequently, the first effective encoded block which is encoded using affine motion compensation prediction is identified from among the surrounding encoded blocks A through D.

For example, referring to FIG. **12**B, when encoded block A neighboring the current block to the left is encoded using affine motion compensation prediction, motion vector v_2 at the top left corner, motion vector v_3 at the top right corner, and motion vector v_4 at the bottom left corner of the encoded block including block A are derived. Next, motion vector v_0 of the top left corner control point of the current block is derived from derived motion vectors v_2, v_3, and v_4. In the same manner, motion vector v_1 of the top right corner control point of the current block is derived.

Next, using derived motion vector v_0 of the top left corner control point and motion vector v_1 of the top right corner control point of the current block, an affine motion vector of each sub-block in the current block is calculated from Equation 2, and motion compensation is performed.

Note that the foregoing describes the operation of the encoding, but the same is true of the operation of the decoding.

Internal Configuration of Affine Motion Compensation Prediction in Inter Predictor in Encoder

FIG. **13** is a block diagram illustrating an internal configuration for affine motion compensation prediction in inter predictor **126** included in encoder **100** according to Embodiment 1. Note that the following focuses on the operation of

inter predictor **126** included in encoder **100**, but the same is true of the operation of inter predictor **218** included in decoder **200**.

As shown in FIG. **13**, inter predictor **126** according to Embodiment 1 includes range determiner **1261**, control point MV deriver **1262**, affine MV calculator **1263**, motion compensation unit **1264**, and motion estimation unit **1265**.

Range determiner **1261** determines a range of motion estimation or motion compensation in a reference picture based on range limit information of motion estimation or motion compensation indicating a partial region in the reference picture within which motion estimation or motion compensation is allowed. The determined range of motion estimation or motion compensation limits a region in the reference picture within which motion estimation or motion compensation in affine motion compensation prediction is allowed. Thus, the possible range of the motion vector (MV) selected in affine motion compensation prediction is limited by placing a limit on the region in the reference picture within which motion estimation or motion compensation is allowed.

In affine inter mode or affine merge mode of affine motion compensation prediction, control point MV deriver **1262** derives the motion vector of the control point (hereinafter, referred to as control point MV) using the MVs of the surrounding encoded blocks neighboring the current block.

Affine MV calculator **1263** calculates an affine motion vector (hereinafter, referred to as affine MV) for each sub-block from Equation 2 using the control point MVs derived by control point MV deriver **1262**.

Motion compensation unit **1264** generates a prediction image by performing motion compensation using the affine MV calculated for each sub-block by affine MV calculator **1263**.

In affine inter mode, motion estimation unit **1265** performs motion estimation through cost evaluation using a current image and the prediction image provided from motion compensation unit **1264**. In doing so, when the affine MV is outside the range of motion estimation and motion compensation in the reference picture, affine inter mode and affine merge mode using the derived control point MVs are prevented.

First Processing Procedure for Affine Inter Mode

FIG. **14** is a flow chart for illustrating the first processing procedure for affine inter mode of affine motion compensation performed by inter predictor **126** in encoder **100** according to Embodiment 1.

As shown in FIG. **14**, firstly, inter-picture predictor **126** determines a range of motion estimation within which motion estimation is allowed, based on the range limit information of motion estimation indicating a partial region in the reference picture within which motion estimation is allowed (S**101**).

Next, inter-picture predictor **126** obtains the motion vectors (MVs) of the surrounding encoded blocks neighboring the current block, and derives the motion vector predictor of the control point (hereinafter, referred to as control point MV predictor) using the obtained MVs (S**102**). In affine inter mode, as described with reference to FIG. **11**, the motion vectors (MVs) of the surrounding encoded blocks neighboring the current block are obtained, and the control point MV predictor is derived using the obtained MVs. Here, a flag indicating which MV is used from among the MVs is written in the stream.

Next, while updating the motion vector of the control point (hereinafter, referred to as control point MV) (S**103**), inter-picture predictor **126** calculates an affine MV for each sub-block from Equation 2 (S**104**).

In doing so, inter-picture predictor **126** determines whether the affine MV calculated at step S**104** is within the range of motion estimation (S**105**). When it is determined to be outside the range (outside the range at S**105**), inter-picture predictor **126** eliminates a current control point MV to be evaluated from search candidates.

On the other hand, when it is determined to be within the range (within the range at S**105**), inter-picture predictor **126** performs affine motion compensation (S**106**), and searches the control point MV having a minimum cost value (for example, a minimum difference value between the current image and the prediction image). Here, a difference value between the control point MV obtained through the search and the control point MV updated at step S**103** is written in the stream. For example, the obtained control point MV is set to the control point MV at the start of the search, and thereby a value closer to the optimal value can be set as an initial value of the search. Accordingly, the accuracy of the search can be improved.

Note that when surrounding pixels of the current block are used in a process following the affine motion compensation prediction (for example, OBMC) and a region including the surrounding pixels used in the process following the affine motion compensation prediction is outside the range of motion estimation, the derived control point MV may be eliminated from the search candidates.

Furthermore, inter predictor **218** in decoder **200** performs decoding based on information of the bitstream generated by encoder **100** in the foregoing manner, and thereby the affine motion compensation can be performed in affine inter mode within the limited range of motion estimation.

Second Processing Procedure for Affine Inter Mode

FIG. **15** is a flow chart for illustrating the second processing procedure for affine inter mode of affine motion compensation performed by inter predictor **126** in encoder **100** according to Embodiment 1. The same components as those in FIG. **14** are assigned with the same reference signs, and redundant descriptions will be omitted. The second processing procedure shown in FIG. **15** differs from the first processing procedure shown in FIG. **14** in that it is determined based on not the affine MV but the control point MV whether the current control point MV to be evaluated is eliminated from the search candidates.

More specifically, at step S**107**, inter-picture predictor **126** determines whether a variation in value between two control point MVs to be evaluated, i.e. the top left corner control point MV and the top right corner control point MV, is within the limited range. When it is determined to be outside the range (outside the range at S**107**), inter-picture predictor **126** eliminates the current control point MV to be evaluated from the search candidates. Here, for example, the variation in value between the two control point MVs means a difference in size or direction, temporal distance to a reference picture, or a difference value between the two control point MVs. For example, the position in the reference picture referred to by the two control point MVs, i.e. the top left corner control point MV and the top right corner control point MV, is within the limited range.

The affine MV for each sub-block is calculated from Equation 2 using the control point MVs at step S**104**, and thus a large variation between the two control point MVs

leads to a large variation between the calculated affine MVs, thereby exceeding the range of motion estimation. In contrast, when the variation between the two control point MVs is within a specified range, the calculated affine MV falls within the range of motion estimation. Accordingly, the specified range is determined in advance as the limited range at step S**101**, and thereby upon updating the control point MV, the derived control point MV can be eliminated from the search candidates to reduce the amount of processing.

Note that when a variation between control point MV predictors is determined to have already exceeded the limited range in the deriving of the control point MV predictor at step S**102**, inter-picture predictor **126** may prevent encoding from being performed in affine inter mode.

Furthermore, when surrounding pixels of the current block are used in a process following the affine motion compensation prediction (for example, OBMC) and a region including the surrounding pixels used in the process following the affine motion compensation prediction is outside the range of motion estimation, the derived control point MV may be eliminated from the search candidates.

Furthermore, inter predictor **218** in decoder **200** performs decoding based on information of the bitstream generated by encoder **100** in the foregoing manner, and thereby the affine motion compensation can be performed in affine inter mode within the limited range of motion estimation.

First Processing Procedure for Affine Merge Mode

FIG. **16** is a flow chart for illustrating the first processing procedure for affine merge mode of affine motion compensation performed by inter predictor **126** in encoder **100** according to Embodiment 1.

As shown in FIG. **16**, firstly, inter-picture predictor **126** determines a range of motion compensation within which motion compensation is allowed, based on the range limit information of motion compensation indicating a partial region in the reference picture within which motion compensation is allowed (S**201**).

Next, inter-picture predictor **126** checks surrounding encoded blocks neighboring the current block in a predetermined order. Subsequently, inter-picture predictor **126** selects the MV of the first effective encoded block which is encoded using affine motion compensation prediction, and derives the control point MV using the selected MV (S**202**). In affine merge mode, as described with reference to FIG. **12**A, the surrounding encoded blocks neighboring the current block are checked in the following order: block A (left), block B (top), block C (top right), block D (bottom left), and block E (top left) which are the surrounding encoded blocks neighboring the current block. Subsequently, the MV of the first effective encoded block which is encoded using affine motion compensation prediction is selected from the surrounding encoded blocks neighboring the current block, and the control point MV is derived using the selected MV.

Next, inter-picture predictor **126** calculates an affine MV for each sub-block from Equation 2 using the control point MVs derived at step S**202** (S**203**).

In doing so, inter-picture predictor **126** determines whether the affine MV calculated at step S**203** is within the range of motion compensation (S**204**). When it is determined to be outside the range (outside the range at S**204**), inter-picture predictor **126** prevents encoding from being performed in affine merge mode (S**205**).

On the other hand, when it is determined to be within the range (within the range at S**204**), inter-picture predictor **126** performs affine motion compensation (S**206**) to obtain the prediction image.

Note that when surrounding pixels of the current block are used in a process following the affine motion compensation prediction (for example, OBMC) and a region including the surrounding pixels used in the process following the affine motion compensation prediction is outside the range of motion compensation, encoding may be prevented from being performed in affine merge mode.

Furthermore, inter predictor **218** in decoder **200** performs decoding based on information of the bitstream generated by encoder **100** in the foregoing manner, and thereby the affine motion compensation can be performed in affine merge mode within the limited range of motion compensation.

Second Processing Procedure for Affine Merge Mode

FIG. **17** is a flow chart for illustrating the second processing procedure for affine merge mode of affine motion compensation performed by inter predictor **126** in encoder **100** according to Embodiment 1. The same components as those in FIG. **16** are assigned with the same reference signs, and redundant descriptions will be omitted. The second processing procedure shown in FIG. **17** differs from the first processing procedure shown in FIG. **16** in that it is determined based on not the affine MV but the control point MV whether a current control point MV to be evaluated is eliminated from the search candidates.

More specifically, at step S**207**, inter-picture predictor **126** determines whether a variation in value between two control point MVs to be evaluated, i.e. the top left corner control point MV and the top right corner control point MV, is within the limited range. When it is determined to be outside the range (outside the range at S**207**), the processing proceeds to step S**205** and the encoding is prevented from being performed in affine merge mode. Here, for example, the variation in value between the two control point MVs means a difference in size or direction, temporal distance to a reference picture, or a difference value between the two control point MVs. For example, the position in the reference picture referred to by the two control point MVs, i.e. the top left corner control point MV and the top right corner control point MV, is within the limited range.

The affine MV for each sub-block is calculated from Equation 2 using the control point MVs at step S**203**, and thus a large variation between the two control point MVs leads to a large variation between the calculated affine MVs, thereby exceeding the range of motion estimation. In contrast, when the variation between the two control point MVs is within a specified range, the calculated affine MV falls within the range of motion estimation. Accordingly, the specified range is determined in advance as the limited range at step S**201**, and thereby upon deriving the control point MV, the encoding can be prevented from being performed in affine merge mode to reduce the amount of processing.

Note that when surrounding pixels of the current block are used in a process following the affine motion compensation prediction (for example, OBMC) and a region including the surrounding pixels used in the process following the affine motion compensation prediction is outside the range of motion compensation, encoding may be prevented from being performed in affine merge mode.

Furthermore, inter predictor **218** in decoder **200** performs decoding based on information of the bitstream generated by

encoder **100** in the foregoing manner, and thereby the affine motion compensation can be performed in affine merge mode within the limited range of motion compensation.

Advantageous Effects of Embodiment 1

According to Embodiment 1, a limit is placed on the range of motion vectors, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

For example, assuming that the range of motion estimation can be unlimitedly and broadly defined, the size and direction of the motion vector for each block in a picture may vary in a wide range. Furthermore, the affine motion compensation is intended for parallel motion and linear transformation such as scaling, shearing, or rotation of an object in the picture, and thus when the affine motion compensation prediction is selected, it is expected that two control point motion vectors, i.e. the top left corner control point MV and the top right corner control point MV, are more likely to have the same direction. However, when the size and direction of the motion vector for each block in the picture vary in a wide range, the motion vector selected from surrounding encoded blocks neighboring the current block may also vary. This leads to the variation in the size and direction between the affine motion vectors, and thus the affine motion compensation prediction would be less likely to be selected. Furthermore, when the size and direction of the motion vector for each block in the picture vary in a wide range, the region in the picture to be obtained as a reference image is also expanded. Accordingly, the memory bandwidth required for an external memory, i.e. a frame memory is more likely to increase.

Note that in the affine motion compensation prediction performed by inter predictor **126** in encoder **100**, the following is achieved by placing a limit on the possible range of MVs. In other words, also in inter predictor **218** in decoder **200** that decodes the encoded bitstream generated by encoder **100**, the possible range of motion vectors in the affine motion compensation prediction is limited in the same manner as encoder **100**.

Furthermore, all the components described in Embodiment 1 are not necessarily required. Only some of the components in Embodiment 1 may be included. Furthermore, the processing of all the components described in Embodiment 1 is not limited to this. Components other than the components in Embodiment 1 may be included to perform the processing.

Variation 1

Note that in the affine motion compensation prediction, the motion estimation range or the motion compensation range may be newly determined for each current picture, or an appropriate set may be selected for each current picture from among a plurality of predetermined sets of motion estimation ranges or a plurality of predetermined sets of motion compensation ranges. With this, the variation between the control point motion vectors in affine motion compensation prediction can be reduced, and thus the affine motion compensation prediction is more likely to be selected in the inter perdition. Accordingly, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, in the affine motion compensation prediction, the motion estimation range or the motion compensation range may be changed in accordance with a type of a reference picture. For example, in referring to P picture, the motion estimation range or the motion compensation range may be larger than that in referring to B picture. Furthermore, in the affine motion compensation prediction, the motion estimation range or the motion compensation range may be determined for each predetermined profile and level. With this, the limit on the range within which the motion estimation or the motion compensation is performed is appropriately determined, and thus it is possible to reduce the memory bandwidth required for an external memory.

Furthermore, in the affine motion compensation prediction, the motion estimation range or the motion compensation range may be determined in accordance with motion-estimation processing power depending on computational processing power on an encoder side or memory bandwidth. With this, the limit on the range within which the motion estimation or the motion compensation is performed can be determined in accordance with the estimation processing power, and thus it is possible not only to reduce the amount of processing in accordance with the estimation processing power but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, in the affine motion compensation prediction, the limit on the motion estimation range or the motion compensation range may include a limit on reference pictures in addition to a limit on a range of referable pixels. With this, it is possible not only to reduce the memory bandwidth required for an external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, in the affine motion compensation prediction, information about the limit on the motion estimation range or the motion compensation range may be included in header information of a video parameter set (VPS), a sequence parameter set (SPS), or a picture parameter set (PPS) of an encoded bitstream. With this, motion compensation using affine motion compensation can be efficiently performed.

Furthermore, in the affine motion compensation prediction, the information about the limit on the motion estimation range or the motion compensation range may include information for limiting reference pictures in addition to information for limiting a range of referable pixels. With this, it is possible not only to reduce the memory bandwidth required for an external memory but also to efficiently perform motion compensation using affine motion compensation.

Furthermore, in the affine motion compensation prediction, the information about the limit on the motion estimation range or the motion compensation range may include only information on whether a limit is placed on motion estimation range and the motion compensation range.

Note that in the upper level system, when the information about the limit on the motion estimation range or the motion compensation range is predetermined or exchanged between a transmitter and a receiver, this information need not be included in header information of the encoded bitstream such as VPS, SPS, PPS, etc.

Furthermore, in the limiting of the reference pictures, a total number of reference pictures is specified, and the

reference pictures may be limited to only the specified reference pictures selected in order of increasing reference index.

Variation 2

Note that in the limiting of the reference pictures, when temporal scalable encoding/decoding is performed, a total number of reference pictures is specified for reference pictures in a layer lower than or equal to that of a current picture to be encoded and/or decoded indicated by the time identifier, and the reference pictures may be limited to only the specified reference pictures selected in order of increasing reference index.

Furthermore, in the limiting of the reference pictures, a total number of reference pictures is specified, and the reference pictures may be limited to only the specified reference pictures selected in order of picture order count (POC) indicating the order of output of pictures from a reference picture close to the current picture to be encoded.

Furthermore, in the limiting of the reference pictures, when temporal scalable encoding is performed, a total number of reference pictures is specified for reference pictures in a layer lower than or equal to that of a current picture to be encoded indicated by the time identifier, and the reference pictures may be limited to only the specified reference pictures selected in order of picture order count (POC) indicating the order of output of pictures from a reference picture close to the current picture to be encoded/decoded.

Implementation Example of Encoder

FIG. **18** is a block diagram illustrating an implementation example of encoder **100** according to Embodiment 1. Encoder **100** includes circuitry **160** and memory **162**. For example, the components in encoder **100** shown in FIGS. **1** and **13** are implemented as circuitry **160** and memory **162** shown in FIG. **18**.

Circuitry **160** performs information processing, and is accessible to memory **162**. For example, circuitry **160** is a dedicated or general-purpose electronic circuit for encoding a video. Circuitry **160** may be a processor such as a CPU. Circuitry **160** also may be an assembly of electronic circuits. Furthermore, for example, circuitry **160** may serve as components other than components for storing information among the components in encoder **100** shown in FIG. **1**, etc.

Memory **162** is a dedicated or general-purpose memory that stores information for encoding a video in circuitry **160**. Memory **162** may be an electronic circuit, and be connected to circuitry **160**. Memory **162** also may be included in circuitry **160**. Memory **162** also may be an assembly of electronic circuits. Memory **162** also may be a magnetic disk, an optical disk, etc., and be referred to as a storage, a recording medium, etc. Memory **162** also may be a non-volatile memory or a volatile memory.

For example, memory **162** may store a video to be encoded, or a bitstream corresponding to the encoded video. Memory **162** also may store a program for encoding a video in circuitry **160**.

Furthermore, for example, memory **162** may serve as components for storing information among the components in encoder **100** shown in FIG. **1**, etc. In particular, memory **162** may serve as block memory **118** and frame memory **122** shown in FIG. **1**. More specifically, memory **162** may store a reconstructed block, a reconstructed picture, etc.

Note that in encoder **100**, all the components shown in FIG. **1**, etc., need not be implemented, or all the foregoing processes need not be performed. Some of the components shown in FIG. **1**, etc., may be included in another device, or some of the foregoing processes may be performed by another device. Then, in encoder **100**, some of the components shown in FIG. **1**, etc., are implemented and some of the forging processes are performed, and thereby the motion compensation is effectively performed.

Hereinafter, an operation example of encoder **100** shown in FIG. **18** will be described. In the following operation example, affine motion compensation prediction is a process for deriving a motion vector for each sub-block based on motion vectors of neighboring blocks in the inter prediction for a current block.

FIG. **19** is a flow chart for illustrating an operation example of encoder **100** shown in FIG. **18**. For example, in encoding a video using motion compensation, encoder **100** shown in FIG. **18** performs the operation shown in FIG. **19**.

In particular, using memory **162**, circuitry **160** in encoder **100**, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block (S**311**).

With this, encoder **100** can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

Implementation Example of Decoder

FIG. **20** is a block diagram illustrating an implementation example of decoder **200** according to Embodiment 1. Decoder **200** includes circuitry **260** and memory **262**. For example, the components in decoder **200** shown in FIG. **10** are implemented as circuitry **260** and memory **262** shown in FIG. **20**.

Circuitry **260** performs information processing, and is accessible to memory **262**. For example, circuitry **260** is a dedicated or general-purpose electronic circuit for decoding a video. Circuitry **260** may be a processor such as a CPU. Circuitry **260** also may be an assembly of electronic circuits. Furthermore, for example, circuitry **260** may serve as components other than components for storing information among the components in decoder **200** shown in FIG. **10**, etc.

Memory **262** is a dedicated or general-purpose memory that stores information for decoding a video in circuitry **260**. Memory **262** may be an electronic circuit, and be connected to circuitry **260**. Memory **262** also may be included in circuitry **260**. Memory **262** also may be an assembly of electronic circuits. Memory **262** also may be a magnetic disk, an optical disk, etc., and be referred to as a storage, a recording medium, etc. Memory **262** also may be a non-volatile memory or a volatile memory.

For example, memory **262** may store a bitstream corresponding to an encoded video, or a video corresponding to a decoded bitstream. Memory **262** also may store a program for decoding a video in circuitry **260**.

Furthermore, for example, memory **262** may serve as components for storing information among the components in decoder **200** shown in FIG. **10**, etc. In particular, memory **262** may serve as block memory **210** and frame memory **214** shown in FIG. **10**. More specifically, memory **262** may store a reconstructed block, a reconstructed picture, etc.

Note that in decoder **200**, all the components shown in FIG. **10**, etc., need not be implemented, or all the foregoing processes need not be performed. Some of the components shown in FIG. **10**, etc., may be included in another device, or some of the foregoing processes may be performed by another device. Then, in decoder **200**, some of the components shown in FIG. **10**, etc., are implemented and some of the forging processes are performed, and thereby the motion compensation is effectively performed.

Hereinafter, an operation example of decoder **200** shown in FIG. **20** will be described. In the following operation example, affine motion compensation prediction is a process for deriving a motion vector for each sub-block based on motion vectors of neighboring blocks in the inter prediction for a current block.

FIG. **21** is a flow chart for illustrating an operation example of decoder **200** shown in FIG. **20**. For example, in decoding a video using motion compensation, decoder **200** shown in FIG. **20** performs the operation shown in FIG. **21**.

In particular, using memory **262**, circuitry **260** in decoder **200**, in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block (S**411**). Subsequently, the encoded stream is decoded (S**412**).

With this, decoder **200** can efficiently perform motion compensation using affine motion compensation. More specifically, a limit is placed on a range within which motion estimation or motion compensation is performed in affine motion compensation prediction, thereby allowing reduction in the variation between control point motion vectors in affine motion compensation prediction. Accordingly, the affine motion compensation prediction is more likely to be selected in the inter perdition, and thus the motion compensation can be efficiently performed using affine motion compensation. It is also possible to place a limit on the region of a reference picture to be obtained, thereby having the possibility of reducing the memory bandwidth required for an external memory, i.e. a frame memory.

Supplementary

Furthermore, encoder **100** and decoder **200** according to the present embodiment may be used as an image encoder and an image decoder, or may be used as a video encoder or a video decoder, respectively. Alternatively, encoder **100** and decoder **200** are applicable as an inter prediction device (an inter-picture prediction device).

In other words, encoder **100** and decoder **200** may correspond to only inter predictor (inter-picture predictor) **126** and inter predictor (inter-picture predictor) **218**, respectively. The remaining components such as transformer **106**, inverse transformer **206**, etc., may be included in another device.

Furthermore, in the present embodiment, each component may be configured by a dedicated hardware, or may be implemented by executing a software program suitable for each component. Each component may be implemented by causing a program executer such as a CPU or a processor to read out and execute a software program stored on a recording medium such as a hard disk or a semiconductor memory.

In particular, encoder **100** and decoder **200** may each include processing circuitry and a storage which is electrically connected to the processing circuitry and is accessible from the processing circuitry. For example, the processing circuitry corresponds to circuitry **160** or **260**, and the storage corresponds to memory **162** or **262**.

The processing circuitry includes at least one of a dedicated hardware and a program executer, and performs processing using the storage. Furthermore, when the processing circuitry includes the program executer, the storage stores a software program to be executed by the program executer.

Here, a software for implementing encoder **100**, decoder **200**, etc., according to the present embodiment is a program as follows.

In other words, the program may cause a computer to execute an encoding method including, in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block.

Alternatively, the program may cause a computer to execute a decoding method including, in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block to decode an encoded stream.

Furthermore, as described above, each component may be a circuit. The circuits may be integrated into a single circuit as a whole, or may be separated from each other. Furthermore, each component may be implemented as a general-purpose processor, or as a dedicated processor.

Furthermore, a process performed by a specific component may be performed by another component. Furthermore, the order of processes may be changed, or multiple processes may be performed in parallel. Furthermore, an encoding and decoding device may include encoder **100** and decoder **200**.

The ordinal numbers used in the illustration such as first and second may be renumbered as needed. Furthermore, the ordinal number may be newly assigned to a component, etc., or may be deleted from a component, etc.

As described above, the aspects of encoder **100** and decoder **200** have been described based on the embodiment, but the aspects of encoder **100** and decoder **200** are not limited to this embodiment. Various modifications to the embodiment that can be conceived by those skilled in the art, and forms configured by combining components in different embodiments without departing from the spirit of the present invention may be included in the scope of the aspects of encoder **100** and decoder **200**.

This aspect may be implemented in combination with one or more of the other aspects according to the present disclosure. In addition, part of the processes in the flowcharts, part of the constituent elements of the apparatuses, and part of the syntax described in this aspect may be implemented in combination with other aspects.

Embodiment 2

As described in each of the above embodiment, each functional block can typically be realized as an MPU and

memory, for example. Moreover, processes performed by each of the functional blocks are typically realized by a program execution unit, such as a processor, reading and executing software (a program) recorded on a recording medium such as ROM. The software may be distributed via, for example, downloading, and may be recorded on a recording medium such as semiconductor memory and distributed. Note that each functional block can, of course, also be realized as hardware (dedicated circuit).

Moreover, the processing described in each of the embodiments may be realized via integrated processing using a single apparatus (system), and, alternatively, may be realized via decentralized processing using a plurality of apparatuses. Moreover, the processor that executes the above-described program may be a single processor or a plurality of processors. In other words, integrated processing may be performed, and, alternatively, decentralized processing may be performed.

Embodiments of the present disclosure are not limited to the above exemplary embodiments; various modifications may be made to the exemplary embodiments, the results of which are also included within the scope of the embodiments of the present disclosure.

Next, application examples of the moving picture encoding method (image encoding method) and the moving picture decoding method (image decoding method) described in each of the above embodiments and a system that employs the same will be described. The system is characterized as including an image encoder that employs the image encoding method, an image decoder that employs the image decoding method, and an image encoder/decoder that includes both the image encoder and the image decoder. Other configurations included in the system may be modified on a case-by-case basis.

Usage Examples

FIG. **22** illustrates an overall configuration of content providing system ex**100** for implementing a content distribution service. The area in which the communication service is provided is divided into cells of desired sizes, and base stations ex**106**, ex**107**, ex**108**, ex**109**, and ex**110**, which are fixed wireless stations, are located in respective cells.

In content providing system ex**100**, devices including computer ex**111**, gaming device ex**112**, camera ex**113**, home appliance ex**114**, and smartphone ex**115** are connected to internet ex**101** via internet service provider ex**102** or communications network ex**104** and base stations ex**106** through ex**110**. Content providing system ex**100** may combine and connect any combination of the above elements. The devices may be directly or indirectly connected together via a telephone network or near field communication rather than via base stations ex**106** through ex**110**, which are fixed wireless stations. Moreover, streaming server ex**103** is connected to devices including computer ex**111**, gaming device ex**112**, camera ex**113**, home appliance ex**114**, and smartphone ex**115** via, for example, internet ex**101**. Streaming server ex**103** is also connected to, for example, a terminal in a hotspot in airplane ex**117** via satellite ex**116**.

Note that instead of base stations ex**106** through ex**110**, wireless access points or hotspots may be used. Streaming server ex**103** may be connected to communications network ex**104** directly instead of via internet ex**101** or internet service provider ex**102**, and may be connected to airplane ex**117** directly instead of via satellite ex**116**.

Camera ex**113** is a device capable of capturing still images and video, such as a digital camera. Smartphone ex**115** is a smartphone device, cellular phone, or personal handyphone system (PHS) phone that can operate under the mobile communications system standards of the typical 2G, 3G, 3.9G, and 4G systems, as well as the next-generation 5G system.

Home appliance ex**118** is, for example, a refrigerator or a device included in a home fuel cell cogeneration system.

In content providing system ex**100**, a terminal including an image and/or video capturing function is capable of, for example, live streaming by connecting to streaming server ex**103** via, for example, base station ex**106**. When live streaming, a terminal (e.g., computer ex**111**, gaming device ex**112**, camera ex**113**, home appliance ex**114**, smartphone ex**115**, or airplane ex**117**) performs the encoding processing described in the above embodiments on still-image or video content captured by a user via the terminal, multiplexes video data obtained via the encoding and audio data obtained by encoding audio corresponding to the video, and transmits the obtained data to streaming server ex**103**. In other words, the terminal functions as the image encoder according to one aspect of the present disclosure.

Streaming server ex**103** streams transmitted content data to clients that request the stream. Client examples include computer ex**111**, gaming device ex**112**, camera ex**113**, home appliance ex**114**, smartphone ex**115**, and terminals inside airplane ex**117**, which are capable of decoding the above-described encoded data. Devices that receive the streamed data decode and reproduce the received data. In other words, the devices each function as the image decoder according to one aspect of the present disclosure.

Decentralized Processing

Streaming server ex**103** may be realized as a plurality of servers or computers between which tasks such as the processing, recording, and streaming of data are divided. For example, streaming server ex**103** may be realized as a content delivery network (CDN) that streams content via a network connecting multiple edge servers located throughout the world. In a CDN, an edge server physically near the client is dynamically assigned to the client. Content is cached and streamed to the edge server to reduce load times. In the event of, for example, some kind of an error or a change in connectivity due to, for example, a spike in traffic, it is possible to stream data stably at high speeds since it is possible to avoid affected parts of the network by, for example, dividing the processing between a plurality of edge servers or switching the streaming duties to a different edge server, and continuing streaming.

Decentralization is not limited to just the division of processing for streaming; the encoding of the captured data may be divided between and performed by the terminals, on the server side, or both. In one example, in typical encoding, the processing is performed in two loops. The first loop is for detecting how complicated the image is on a frame-by-frame or scene-by-scene basis, or detecting the encoding load. The second loop is for processing that maintains image quality and improves encoding efficiency. For example, it is possible to reduce the processing load of the terminals and improve the quality and encoding efficiency of the content by having the terminals perform the first loop of the encoding and having the server side that received the content perform the second loop of the encoding. In such a case, upon receipt of a decoding request, it is possible for the encoded data resulting from the first loop performed by one

terminal to be received and reproduced on another terminal in approximately real time. This makes it possible to realize smooth, real-time streaming.

In another example, camera ex**113** or the like extracts a feature amount from an image, compresses data related to the feature amount as metadata, and transmits the compressed metadata to a server. For example, the server determines the significance of an object based on the feature amount and changes the quantization accuracy accordingly to perform compression suitable for the meaning of the image. Feature amount data is particularly effective in improving the precision and efficiency of motion vector prediction during the second compression pass performed by the server. Moreover, encoding that has a relatively low processing load, such as variable length coding (VLC), may be handled by the terminal, and encoding that has a relatively high processing load, such as context-adaptive binary arithmetic coding (CABAC), may be handled by the server.

In yet another example, there are instances in which a plurality of videos of approximately the same scene are captured by a plurality of terminals in, for example, a stadium, shopping mall, or factory. In such a case, for example, the encoding may be decentralized by dividing processing tasks between the plurality of terminals that captured the videos and, if necessary, other terminals that did not capture the videos and the server, on a per-unit basis. The units may be, for example, groups of pictures (GOP), pictures, or tiles resulting from dividing a picture. This makes it possible to reduce load times and achieve streaming that is closer to real-time.

Moreover, since the videos are of approximately the same scene, management and/or instruction may be carried out by the server so that the videos captured by the terminals can be cross-referenced. Moreover, the server may receive encoded data from the terminals, change reference relationship between items of data or correct or replace pictures themselves, and then perform the encoding. This makes it possible to generate a stream with increased quality and efficiency for the individual items of data.

Moreover, the server may stream video data after performing transcoding to convert the encoding format of the video data. For example, the server may convert the encoding format from MPEG to VP, and may convert H.264 to H.265.

In this way, encoding can be performed by a terminal or one or more servers. Accordingly, although the device that performs the encoding is referred to as a "server" or "terminal" in the following description, some or all of the processes performed by the server may be performed by the terminal, and likewise some or all of the processes performed by the terminal may be performed by the server. This also applies to decoding processes.

3D, Multi-Angle

In recent years, usage of images or videos combined from images or videos of different scenes concurrently captured or the same scene captured from different angles by a plurality of terminals such as camera ex**113** and/or smartphone ex**115** has increased. Videos captured by the terminals are combined based on, for example, the separately-obtained relative positional relationship between the terminals, or regions in a video having matching feature points.

In addition to the encoding of two-dimensional moving pictures, the server may encode a still image based on scene analysis of a moving picture either automatically or at a point in time specified by the user, and transmit the encoded still image to a reception terminal. Furthermore, when the server can obtain the relative positional relationship between the video capturing terminals, in addition to two-dimensional moving pictures, the server can generate three-dimensional geometry of a scene based on video of the same scene captured from different angles. Note that the server may separately encode three-dimensional data generated from, for example, a point cloud, and may, based on a result of recognizing or tracking a person or object using three-dimensional data, select or reconstruct and generate a video to be transmitted to a reception terminal from videos captured by a plurality of terminals.

This allows the user to enjoy a scene by freely selecting videos corresponding to the video capturing terminals, and allows the user to enjoy obtained by extracting, from three-dimensional data the content reconstructed from a plurality of images or videos, a video from a selected viewpoint. Furthermore, similar to with video, sound may be recorded from relatively different angles, and the server may multiplex, with the video, audio from a specific angle or space in accordance with the video, and transmit the result.

In recent years, content that is a composite of the real world and a virtual world, such as virtual reality (VR) and augmented reality (AR) content, has also become popular. In the case of VR images, the server may create images from the viewpoints of both the left and right eyes and perform encoding that tolerates reference between the two viewpoint images, such as multi-view coding (MVC), and, alternatively, may encode the images as separate streams without referencing. When the images are decoded as separate streams, the streams may be synchronized when reproduced so as to recreate a virtual three-dimensional space in accordance with the viewpoint of the user.

In the case of AR images, the server superimposes virtual object information existing in a virtual space onto camera information representing a real-world space, based on a three-dimensional position or movement from the perspective of the user. The decoder may obtain or store virtual object information and three-dimensional data, generate two-dimensional images based on movement from the perspective of the user, and then generate superimposed data by seamlessly connecting the images. Alternatively, the decoder may transmit, to the server, motion from the perspective of the user in addition to a request for virtual object information, and the server may generate superimposed data based on three-dimensional data stored in the server in accordance with the received motion, and encode and stream the generated superimposed data to the decoder. Note that superimposed data includes, in addition to RGB values, an a value indicating transparency, and the server sets the a value for sections other than the object generated from three-dimensional data to, for example, 0, and may perform the encoding while those sections are transparent. Alternatively, the server may set the background to a predetermined RGB value, such as a chroma key, and generate data in which areas other than the object are set as the background.

Decoding of similarly streamed data may be performed by the client (i.e., the terminals), on the server side, or divided therebetween. In one example, one terminal may transmit a reception request to a server, the requested content may be received and decoded by another terminal, and a decoded signal may be transmitted to a device having a display. It is possible to reproduce high image quality data by decentralizing processing and appropriately selecting content regardless of the processing ability of the communications terminal itself. In yet another example, while a TV, for example, is receiving image data that is large in size, a region of a

picture, such as a tile obtained by dividing the picture, may be decoded and displayed on a personal terminal or terminals of a viewer or viewers of the TV. This makes it possible for the viewers to share a big-picture view as well as for each viewer to check his or her assigned area or inspect a region in further detail up close.

In the future, both indoors and outdoors, in situations in which a plurality of wireless connections are possible over near, mid, and far distances, it is expected to be able to seamlessly receive content even when switching to data appropriate for the current connection, using a streaming system standard such as MPEG-DASH. With this, the user can switch between data in real time while freely selecting a decoder or display apparatus including not only his or her own terminal, but also, for example, displays disposed indoors or outdoors. Moreover, based on, for example, information on the position of the user, decoding can be performed while switching which terminal handles decoding and which terminal handles the displaying of content. This makes it possible to, while in route to a destination, display, on the wall of a nearby building in which a device capable of displaying content is embedded or on part of the ground, map information while on the move. Moreover, it is also possible to switch the bit rate of the received data based on the accessibility to the encoded data on a network, such as when encoded data is cached on a server quickly accessible from the reception terminal or when encoded data is copied to an edge server in a content delivery service.

Scalable Encoding

The switching of content will be described with reference to a scalable stream, illustrated in FIG. **23**, that is compression coded via implementation of the moving picture encoding method described in the above embodiments. The server may have a configuration in which content is switched while making use of the temporal and/or spatial scalability of a stream, which is achieved by division into and encoding of layers, as illustrated in FIG. **23**. Note that there may be a plurality of individual streams that are of the same content but different quality. In other words, by determining which layer to decode up to based on internal factors, such as the processing ability on the decoder side, and external factors, such as communication bandwidth, the decoder side can freely switch between low resolution content and high resolution content while decoding. For example, in a case in which the user wants to continue watching, at home on a device such as a TV connected to the internet, a video that he or she had been previously watching on smartphone ex**115** while on the move, the device can simply decode the same stream up to a different layer, which reduces server side load.

Furthermore, in addition to the configuration described above in which scalability is achieved as a result of the pictures being encoded per layer and the enhancement layer is above the base layer, the enhancement layer may include metadata based on, for example, statistical information on the image, and the decoder side may generate high image quality content by performing super-resolution imaging on a picture in the base layer based on the metadata. Super-resolution imaging may be improving the SN ratio while maintaining resolution and/or increasing resolution. Metadata includes information for identifying a linear or a non-linear filter coefficient used in super-resolution processing, or information identifying a parameter value in filter processing, machine learning, or least squares method used in super-resolution processing.

Alternatively, a configuration in which a picture is divided into, for example, tiles in accordance with the meaning of, for example, an object in the image, and on the decoder side, only a partial region is decoded by selecting a tile to decode, is also acceptable. Moreover, by storing an attribute about the object (person, car, ball, etc.) and a position of the object in the video (coordinates in identical images) as metadata, the decoder side can identify the position of a desired object based on the metadata and determine which tile or tiles include that object. For example, as illustrated in FIG. **24**, metadata is stored using a data storage structure different from pixel data such as an SEI message in HEVC. This metadata indicates, for example, the position, size, or color of the main object.

Moreover, metadata may be stored in units of a plurality of pictures, such as stream, sequence, or random access units. With this, the decoder side can obtain, for example, the time at which a specific person appears in the video, and by fitting that with picture unit information, can identify a picture in which the object is present and the position of the object in the picture.

Web Page Optimization

FIG. **25** illustrates an example of a display screen of a web page on, for example, computer ex**111**. FIG. **26** illustrates an example of a display screen of a web page on, for example, smartphone ex**115**. As illustrated in FIG. **25** and FIG. **26**, a web page may include a plurality of image links which are links to image content, and the appearance of the web page differs depending on the device used to view the web page. When a plurality of image links are viewable on the screen, until the user explicitly selects an image link, or until the image link is in the approximate center of the screen or the entire image link fits in the screen, the display apparatus (decoder) displays, as the image links, still images included in the content or I pictures, displays video such as an animated gif using a plurality of still images or I pictures, for example, or receives only the base layer and decodes and displays the video.

When an image link is selected by the user, the display apparatus decodes giving the highest priority to the base layer. Note that if there is information in the HTML code of the web page indicating that the content is scalable, the display apparatus may decode up to the enhancement layer. Moreover, in order to guarantee real time reproduction, before a selection is made or when the bandwidth is severely limited, the display apparatus can reduce delay between the point in time at which the leading picture is decoded and the point in time at which the decoded picture is displayed (that is, the delay between the start of the decoding of the content to the displaying of the content) by decoding and displaying only forward reference pictures (I picture, P picture, forward reference B picture). Moreover, the display apparatus may purposely ignore the reference relationship between pictures and coarsely decode all B and P pictures as forward reference pictures, and then perform normal decoding as the number of pictures received over time increases.

Autonomous Driving

When transmitting and receiving still image or video data such two- or three-dimensional map information for autonomous driving or assisted driving of an automobile, the reception terminal may receive, in addition to image data belonging to one or more layers, information on, for example, the weather or road construction as metadata, and

associate the metadata with the image data upon decoding. Note that metadata may be assigned per layer and, alternatively, may simply be multiplexed with the image data.

In such a case, since the automobile, drone, airplane, etc., including the reception terminal is mobile, the reception terminal can seamlessly receive and decode while switching between base stations among base stations ex**106** through ex**110** by transmitting information indicating the position of the reception terminal upon reception request. Moreover, in accordance with the selection made by the user, the situation of the user, or the bandwidth of the connection, the reception terminal can dynamically select to what extent the metadata is received or to what extent the map information, for example, is updated.

With this, in content providing system ex**100**, the client can receive, decode, and reproduce, in real time, encoded information transmitted by the user.

Streaming of Individual Content

In content providing system ex**100**, in addition to high image quality, long content distributed by a video distribution entity, unicast or multicast streaming of low image quality, short content from an individual is also possible. Moreover, such content from individuals is likely to further increase in popularity. The server may first perform editing processing on the content before the encoding processing in order to refine the individual content. This may be achieved with, for example, the following configuration.

In real-time while capturing video or image content or after the content has been captured and accumulated, the server performs recognition processing based on the raw or encoded data, such as capture error processing, scene search processing, meaning analysis, and/or object detection processing. Then, based on the result of the recognition processing, the server—either when prompted or automatically—edits the content, examples of which include: correction such as focus and/or motion blur correction; removing low-priority scenes such as scenes that are low in brightness compared to other pictures or out of focus; object edge adjustment; and color tone adjustment. The server encodes the edited data based on the result of the editing. It is known that excessively long videos tend to receive fewer views. Accordingly, in order to keep the content within a specific length that scales with the length of the original video, the server may, in addition to the low-priority scenes described above, automatically clip out scenes with low movement based on an image processing result. Alternatively, the server may generate and encode a video digest based on a result of an analysis of the meaning of a scene.

Note that there are instances in which individual content may include content that infringes a copyright, moral right, portrait rights, etc. Such an instance may lead to an unfavorable situation for the creator, such as when content is shared beyond the scope intended by the creator. Accordingly, before encoding, the server may, for example, edit images so as to blur faces of people in the periphery of the screen or blur the inside of a house, for example. Moreover, the server may be configured to recognize the faces of people other than a registered person in images to be encoded, and when such faces appear in an image, for example, apply a mosaic filter to the face of the person. Alternatively, as pre- or post-processing for encoding, the user may specify, for copyright reasons, a region of an image including a person or a region of the background be processed, and the server may process the specified region by, for example, replacing the region with a different image or blurring the region. If the region includes a person, the person may be tracked in the moving picture, and the head region may be replaced with another image as the person moves.

Moreover, since there is a demand for real-time viewing of content produced by individuals, which tends to be small in data size, the decoder first receives the base layer as the highest priority and performs decoding and reproduction, although this may differ depending on bandwidth. When the content is reproduced two or more times, such as when the decoder receives the enhancement layer during decoding and reproduction of the base layer and loops the reproduction, the decoder may reproduce a high image quality video including the enhancement layer. If the stream is encoded using such scalable encoding, the video may be low quality when in an unselected state or at the start of the video, but it can offer an experience in which the image quality of the stream progressively increases in an intelligent manner. This is not limited to just scalable encoding; the same experience can be offered by configuring a single stream from a low quality stream reproduced for the first time and a second stream encoded using the first stream as a reference.

Other Usage Examples

The encoding and decoding may be performed by LSI ex**500**, which is typically included in each terminal. LSI ex**500** may be configured of a single chip or a plurality of chips. Software for encoding and decoding moving pictures may be integrated into some type of a recording medium (such as a CD-ROM, a flexible disk, or a hard disk) that is readable by, for example, computer ex**111**, and the encoding and decoding may be performed using the software. Furthermore, when smartphone ex**115** is equipped with a camera, the video data obtained by the camera may be transmitted. In this case, the video data is coded by LSI ex**500** included in smartphone ex**115**.

Note that LSI ex**500** may be configured to download and activate an application. In such a case, the terminal first determines whether it is compatible with the scheme used to encode the content or whether it is capable of executing a specific service. When the terminal is not compatible with the encoding scheme of the content or when the terminal is not capable of executing a specific service, the terminal first downloads a codec or application software then obtains and reproduces the content.

Aside from the example of content providing system ex**100** that uses internet ex**101**, at least the moving picture encoder (image encoder) or the moving picture decoder (image decoder) described in the above embodiments may be implemented in a digital broadcasting system. The same encoding processing and decoding processing may be applied to transmit and receive broadcast radio waves superimposed with multiplexed audio and video data using, for example, a satellite, even though this is geared toward multicast whereas unicast is easier with content providing system ex**100**.

Hardware Configuration

FIG. **27** illustrates smartphone ex**115**. FIG. **28** illustrates a configuration example of smartphone ex**115**. Smartphone ex**115** includes antenna ex**450** for transmitting and receiving radio waves to and from base station ex**110**, camera ex**465** capable of capturing video and still images, and display ex**458** that displays decoded data, such as video captured by camera ex**465** and video received by antenna ex**450**. Smart-

phone ex115 further includes user interface ex466 such as a touch panel, audio output unit ex457 such as a speaker for outputting speech or other audio, audio input unit ex456 such as a microphone for audio input, memory ex467 capable of storing decoded data such as captured video or still images, recorded audio, received video or still images, and mail, as well as decoded data, and slot ex464 which is an interface for SIM ex468 for authorizing access to a network and various data. Note that external memory may be used instead of memory ex467.

Moreover, main controller ex460 which comprehensively controls display ex458 and user interface ex466, power supply circuit ex461, user interface input controller ex462, video signal processor ex455, camera interface ex463, display controller ex459, modulator/demodulator ex452, multiplexer/demultiplexer ex453, audio signal processor ex454, slot ex464, and memory ex467 are connected via bus ex470.

When the user turns the power button of power supply circuit ex461 on, smartphone ex115 is powered on into an operable state by each component being supplied with power from a battery pack.

Smartphone ex115 performs processing for, for example, calling and data transmission, based on control performed by main controller ex460, which includes a CPU, ROM, and RAM. When making calls, an audio signal recorded by audio input unit ex456 is converted into a digital audio signal by audio signal processor ex454, and this is applied with spread spectrum processing by modulator/demodulator ex452 and digital-analog conversion and frequency conversion processing by transmitter/receiver ex451, and then transmitted via antenna ex450. The received data is amplified, frequency converted, and analog-digital converted, inverse spread spectrum processed by modulator/demodulator ex452, converted into an analog audio signal by audio signal processor ex454, and then output from audio output unit ex457. In data transmission mode, text, still-image, or video data is transmitted by main controller ex460 via user interface input controller ex462 as a result of operation of, for example, user interface ex466 of the main body, and similar transmission and reception processing is performed. In data transmission mode, when sending a video, still image, or video and audio, video signal processor ex455 compression encodes, via the moving picture encoding method described in the above embodiments, a video signal stored in memory ex467 or a video signal input from camera ex465, and transmits the encoded video data to multiplexer/demultiplexer ex453. Moreover, audio signal processor ex454 encodes an audio signal recorded by audio input unit ex456 while camera ex465 is capturing, for example, a video or still image, and transmits the encoded audio data to multiplexer/demultiplexer ex453. Multiplexer/demultiplexer ex453 multiplexes the encoded video data and encoded audio data using a predetermined scheme, modulates and converts the data using modulator/demodulator (modulator/demodulator circuit) ex452 and transmitter/receiver ex451, and transmits the result via antenna ex450.

When video appended in an email or a chat, or a video linked from a web page, for example, is received, in order to decode the multiplexed data received via antenna ex450, multiplexer/demultiplexer ex453 demultiplexes the multiplexed data to divide the multiplexed data into a bitstream of video data and a bitstream of audio data, supplies the encoded video data to video signal processor ex455 via synchronous bus ex470, and supplies the encoded audio data to audio signal processor ex454 via synchronous bus ex470. Video signal processor ex455 decodes the video signal using a moving picture decoding method corresponding to the moving picture encoding method described in the above embodiments, and video or a still image included in the linked moving picture file is displayed on display ex458 via display controller ex459. Moreover, audio signal processor ex454 decodes the audio signal and outputs audio from audio output unit ex457. Note that since real-time streaming is becoming more and more popular, there are instances in which reproduction of the audio may be socially inappropriate depending on the user's environment. Accordingly, as an initial value, a configuration in which only video data is reproduced, i.e., the audio signal is not reproduced, is preferable. Audio may be synchronized and reproduced only when an input, such as when the user clicks video data, is received.

Although smartphone ex115 was used in the above example, three implementations are conceivable: a transceiver terminal including both an encoder and a decoder; a transmitter terminal including only an encoder; and a receiver terminal including only a decoder. Further, in the description of the digital broadcasting system, an example is given in which multiplexed data obtained as a result of video data being multiplexed with, for example, audio data, is received or transmitted, but the multiplexed data may be video data multiplexed with data other than audio data, such as text data related to the video. Moreover, the video data itself rather than multiplexed data maybe received or transmitted.

Although main controller ex460 including a CPU is described as controlling the encoding or decoding processes, terminals often include GPUs. Accordingly, a configuration is acceptable in which a large area is processed at once by making use of the performance ability of the GPU via memory shared by the CPU and GPU or memory including an address that is managed so as to allow common usage by the CPU and GPU. This makes it possible to shorten encoding time, maintain the real-time nature of the stream, and reduce delay. In particular, processing relating to motion estimation, deblocking filtering, sample adaptive offset (SAO), and transformation/quantization can be effectively carried out by the GPU instead of the CPU in units of, for example pictures, all at once.

Although only some exemplary embodiments of the present disclosure have been described in detail above, those skilled in the art will readily appreciate that many modifications are possible in the exemplary embodiments without materially departing from the novel teachings and advantages of the present disclosure. Accordingly, all such modifications are intended to be included within the scope of the present disclosure.

INDUSTRIAL APPLICABILITY

The present invention is applicable to a television receiver, a digital video recorder, a car navigation system, a mobile phone, a digital camera, a digital video camera, a teleconference system, an electronic mirror, etc.

What is claimed is:

1. A decoder comprising:

circuitry; and

memory, wherein

using the memory, the circuitry,

in affine motion compensation prediction in inter prediction for a current block, places a limit on a range within which motion estimation or motion compensation is performed, and performs the motion compensation for the current block to decode an encoded stream,

the limit on the range within which the motion estimation or the motion compensation is performed is changed in accordance with a type of a reference picture, and

the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a P picture is different from the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a B picture.

2. A decoding method comprising:

in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block to decode an encoded stream, wherein

the limit on the range within which the motion estimation or the motion compensation is performed is changed in accordance with a type of a reference picture, and

the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a P picture is different from the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a B picture.

3. A non-transitory computer readable medium storing a bitstream, the bitstream including an encoded signal and syntax information according to which a decoder performs a method comprising:

in affine motion compensation prediction in inter prediction for a current block, placing a limit on a range within which motion estimation or motion compensation is performed, and performing the motion compensation for the current block to decode an encoded stream, wherein

the limit on the range within which the motion estimation or the motion compensation is performed is changed in accordance with a type of a reference picture, and

the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a P picture is different from the range within which the motion estimation or the motion compensation is performed when the type of the reference picture is a B picture.

* * * * *