

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner

v.

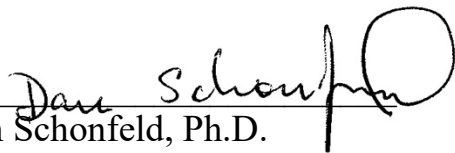
TELEFONAKTIEBOLAGET LM ERICSSON,
Patent Owner.

IPR2024-01008
U.S. Patent No. 9,641,841

DECLARATION OF DAN SCHONFELD, PH.D.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: June 13, 2024

By: 
Dan Schonfeld, Ph.D.

INTEL-1003
9,641,841

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	QUALIFICATIONS.....	2
III.	MATERIALS CONSIDERED	8
IV.	LEGAL STANDARDS.....	9
V.	TECHNOLOGY BACKGROUND.....	15
A.	Video Encoding and Decoding	15
B.	Deblocking Filters.....	20
1.	Pixel Values	22
2.	Filters	28
3.	Deblocking Filter Development.....	34
a.	H.263.....	34
b.	H.264.....	38
c.	Draft H.265 (B205).....	45
VI.	'841 PATENT	48
A.	Specification.....	48
B.	Prosecution History.....	53
VII.	PRIOR ART	55
A.	Bjontegaard (and Bjontegaard's Code) (Ex-1009-1015).....	55
B.	Aksu (Ex-1019).....	66
C.	Gonzalez (Ex-1020)	72
D.	H.241 (Ex-1017)	75
E.	B205/Applicant-Admitted Prior Art (Ex-1018).....	79
F.	Fu (Ex-1022)	80
G.	B118 (Ex-1023).....	81
VIII.	LEVEL OF ORDINARY SKILL.....	83
IX.	CLAIM CONSTRUCTION.....	84

TABLE OF CONTENTS
(continued)

	Page
X. GROUNDS	85
A. Ground 1A: Obviousness over Bjontegaard in view of Bjontegaard's Code.....	85
1. Claim 1	86
a. Element 1[pre]: A method for filtering control applicable to a block of multiple pixels in a video frame, each pixel having a respective pixel value, said method comprising:	86
b. Element 1[a]: calculating a first filter decision value for said block, by only using pixels p_{2_i} , p_{1_i} , and p_{0_i} , based on: $ p_{2_i}-2p_{1_i}+p_{0_i} $ wherein p_{0_i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1_i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2_i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	87
c. Element 1[b]: calculating a second filter decision value for said neighboring block, by only using pixels q_{2_i} , q_{1_i} , and q_{0_i} , based on: $ q_{2_i}-2q_{1_i}+q_{0_i} $ wherein q_{0_i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1_i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2_i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	92
d. Element 1[c]: determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and	93

TABLE OF CONTENTS
(continued)

	Page
e. Element 1[d] determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.....	96
2. Claim 3	100
a. Element 3[a]: The method according to claim 1, wherein determining how many pixels in said first line of pixels in said block to filter comprises:.....	100
b. Element 3[b]: comparing said first filter decision value to a first threshold value;	101
c. Element 3[c]: determining to filter two pixels in said first line of pixels in said block relative to said block boundary if said first filter decision value is below said first threshold value; and	101
d. Element 3[d]: determining to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value; and	101
e. Element 3[e]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises:	102
f. Element 3[f]: comparing said second filter decision value to a second threshold value;	102
g. Element 3[g]: determining to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value; and	102

TABLE OF CONTENTS
(continued)

	Page
h. Element 3[h]: determining to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second threshold value.....	102
3. Claim 5	103
a. Element 5[pre]: A filtering control device comprising:	103
b. Element 5[a]: a first decision value calculator configured to calculate a first filter decision value for a block of multiple pixels in a video frame, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $ p_{2i}-2p_{1i}+p_{0i} $ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;.....	104
c. Element 5[b]: a second decision value calculator configured to calculate a second filter decision value for a neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $ q_{2i}-2q_{1i}+q_{0i} $ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	105
d. Element 5[c]: a first pixel determiner configured to determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based	

TABLE OF CONTENTS
(continued)

	Page
on said first filter decision value calculated by said first decision value calculator; and.....	106
e. Element 5[d]: a second pixel determiner configured to determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator, wherein.....	107
f. Element 5[e]: further comprising a first comparator configured to compare said first filter decision value calculated by said first decision value calculator to a first threshold value; and	108
g. Element 5[f]: a second comparator configured to compare said second filter decision value calculated by said second decision value calculator to a second threshold value.....	109
4. Claim 7.....	109
a. Element 7[a]: The device according to claim 5, further comprising: said first pixel determiner is configured to i) determine to filter two pixels in said first line of pixels in said block relative to said block boundary if said first filter decision value is below said first threshold value as determined by said first comparator and	109
b. Element 7[b]: ii) determine to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value as determined by said first comparator; and.....	110
c. Element 7[c]: said second pixel determiner is configured to i) determine to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value	

TABLE OF CONTENTS
(continued)

	Page
is below said second threshold value as determined by said second comparator and.....	110
d. Element 7[d]: ii) determine to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second threshold value as determined by said second comparator.	110
5. Claim 9.....	111
a. Element 9[a]: The device according to claim 5, further comprising a threshold determiner configured to i) determine said first threshold value based on a quantization parameter associated with said block, and.....	111
b. Element 9[b]: ii) determine said second threshold value based on a quantization parameter associated with said neighboring block.	113
6. Claim 10.....	114
a. Element 10[a]: The device according to claim 5, further comprising: a first offset calculator configured to calculate a first offset Δ based on $(9x(q_{0j} - p_{0j}) - 3x(q_{1j} - p_{1j}))/16$ wherein p_{0j} denotes a pixel value of a pixel closest to, in said first line of pixels, said block boundary, p_{1j} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary, q_{0j} denotes a pixel value of a pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary and q_{1j} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary;	114
b. Element 10[b]: a first pixel modifier configured to modify said pixel value of said pixel closest to said	

TABLE OF CONTENTS
(continued)

Page

block boundary in said first line of pixels by adding said first offset calculated by said first offset calculator to said pixel value of said pixel closest to said block boundary in said first line of pixels;	133
c. Element 10[c]: a second pixel modifier configured to modify said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels by subtracting said first offset calculated by said first offset calculator from said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels;	134
d. Element 10[d]: a second offset calculator configured to calculate, if said first filter decision value is below said first threshold value as determined by said first comparator, a second offset based on $(p0_j + p2_j - 2p1_j + 2\Delta)/4$ wherein $p2_j$ denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	135
e. Element 10[e]: a third pixel modifier configured to modify, if said first filter decision value is below said first threshold as determined by said first comparator, said pixel value of said pixel next closest to said block boundary in said first line of pixels by adding said second offset calculated by said second offset calculator to said pixel value of said pixel next closest to said block boundary in said first line of pixels;	146
f. Element 10[f]: a third offset calculator configured to calculate, if said second filter decision value is below said second threshold value as determined by said second comparator, a third offset based on $(q0_j + q2_j - 2q1_j - 2\Delta)/4$ wherein $q2_j$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said	

TABLE OF CONTENTS
(continued)

	Page
corresponding first line of pixels, said block boundary; and.....	147
g. Element 10[g]: a fourth pixel modifier configured to modify, if said second filter decision value is below said second threshold as determined by said second comparator, said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels by adding said third offset calculated by said third offset calculator to said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels.	150
7. Claim 12.....	152
a. Element 12[a]: An encoder comprising a filtering control device according to claim 5.....	152
8. Claim 13.....	153
a. Element 13[pre]: A user equipment comprising:	153
b. Element 13[a]: a memory configured to store video frames; and.....	155
c. Element 13[b]: an encoder according to claim 12 configured to encode said video frames into encoded video frames, wherein said memory is further configured to store said encoded video frames.	156
9. Claim 14.....	157
a. Element 14[a]: A decoder comprising a filtering control device according to claim 5.	157
10. Claim 15.....	158
a. Element 15[pre]: A user equipment comprising:	158

TABLE OF CONTENTS
(continued)

	Page
b. Element 15[a]: a memory configured to store encoded video frames;.....	158
c. Element 15[b]: a decoder according to claim 14 configured to decode said encoded video frames into decoded video frames; and	159
d. Element 15[c]: a media player configured to render said decoded video frames into video data displayable on a display.....	160
11. Claim 16.....	161
a. Element 16 [pre]: A non-transitory computer-readable medium comprising, stored thereupon, a computer program for filtering control of a block of multiple pixels in a video frame each pixel having a respective pixel value, said computer program comprising code means which when run on a computer causes the computer to:	161
b. Element 16[a]: calculate a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $ p_{2i}-2p_{1i}+p_{0i} $ wherein p_{0i} , denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	162
c. Element 16[b]: calculate a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $ q_{2i}-2q_{1i}+q_{0i} $ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a	

TABLE OF CONTENTS
(continued)

	Page
pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	163
d. Element 16[c]: determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value; and	163
e. Element 16[d]: determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value, ...	163
f. Element 16[e]: wherein determine how many pixels in said first line of pixels in said block to filter comprises determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and	164
g. Element 16[f]: determine how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.....	164
B. Ground 1B: Obviousness over Bjontegaard in view of H.241 and B205	164
1. Claim 2	164
a. Element 2[a]: The method according to claim 1, wherein calculating said first filter decision value comprises calculating said first filter decision value as: $ p_{20}-2p_{10}+p_{00} + p_{23}-2p_{13}+p_{03} $ wherein p_{00} denotes said pixel	

TABLE OF CONTENTS
(continued)

	Page
value of said pixel closest to, in said first line of pixels, said block boundary, $p1_0$ denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, $p2_0$ denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, $p0_3$ denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, $p1_3$ denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and $p2_3$ denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and	165
b. Element 2[b]: calculating said second filter decision value comprises calculating said second filter decision value as: $ q2_0-2q1_0+q0_0 + q2_3-2q1_3+q0_3 $ wherein $q0_0$ denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, $q1_0$ denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, $q2_0$ denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary, $q0_3$ denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, $q1_3$ denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and $q2_3$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.	173
2. Claim 6.....	174
a. Element 6[a]: The device according to claim 5, wherein said first decision value calculator is configured to	

TABLE OF CONTENTS
(continued)

Page

calculate said first filter decision value as: $ p_{2_0}-2p_{1_0}+p_{0_0} + p_{2_3}-2p_{1_3}+p_{0_3} $ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{2_3} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and	175
b. Element 6[b]: said second decision value calculator is configured to calculate said second filter decision value as: $ q_{2_0}-2q_{1_0}+q_{0_0} + q_{2_3}-2q_{1_3}+q_{0_3} $ wherein q_{0_0} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{1_0} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{2_0} denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary, q_{0_3} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, q_{1_3} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and q_{2_3} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.....	175
3. Claim 11	176

TABLE OF CONTENTS
(continued)

	Page
a. Element 11[a]: The device according to claim 5, further comprising a third comparator configured to compare a sum of said first filter decision value and said second filter decision value to a threshold value, wherein	176
b. Element 11[b]: said first pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and	179
c. Element 11[c]: said second pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator.....	179
C. Ground 2A: Obviousness over Aksu in view of Gonzalez.....	180
1. Claim 1	180
a. Element 1[pre]: A method for filtering control applicable to a block of multiple pixels in a video frame, each pixel having a respective pixel value, said method comprising:	180
b. Element 1[a]: calculating a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $ p_{2i}-2p_{1i}+p_{0i} $ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a	

TABLE OF CONTENTS
(continued)

	Page
pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	181
c. Element 1[b]: calculating a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $ q_{2i}-2q_{1i}+q_{0i} $ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	194
d. Element 1[c]: determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and	196
e. Element 1[d]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.....	202
2. Claim 3	202
a. Element 3[a]: The method according to claim 1, wherein determining how many pixels in said first line of pixels in said block to filter comprises:.....	202
b. Element 3[b]: comparing said first filter decision value to a first threshold value;	203
c. Element 3[c]: determining to filter two pixels in said first line of pixels in said block relative to said block	

TABLE OF CONTENTS
(continued)

	Page
boundary if said first filter decision value is below said first threshold value; and	203
d. Element 3[d]: determining to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value; and	206
e. Element 3[e]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises:	206
f. Element 3[f]: comparing said second filter decision value to a second threshold value;	207
g. Element 3[g]: determining to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value; and	207
h. Element 3[h]: determining to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second threshold value	208
3. Claim 5	209
a. Element 5[pre]: A filtering control device comprising:	209
b. Element 5[a]: a first decision value calculator configured to calculate a first filter decision value for a block of multiple pixels in a video frame, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $ p_{2i}-2p_{1i}+p_{0i} $ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to,	

TABLE OF CONTENTS
(continued)

Page

in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;.....	209
c. Element 5[b]: a second decision value calculator configured to calculate a second filter decision value for a neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $ q_{2i}-2q_{1i}+q_{0i} $ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	210
d. Element 5[c]: a first pixel determiner configured to determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and.....	210
e. Element 5[d]: a second pixel determiner configured to determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator, wherein.....	211
f. Element 5[e]: further comprising a first comparator configured to compare said first filter decision value calculated by said first decision value calculator to a first threshold value; and	211
g. Element 5[f]: a second comparator configured to compare said second filter decision value calculated by	

TABLE OF CONTENTS
(continued)

	Page
said second decision value calculator to a second threshold value.....	212
4. Claim 7	212
a. Element 7[a]: The device according to claim 5, further comprising: said first pixel determiner is configured to i) determine to filter two pixels in said first line of pixels in said block relative to said block boundary if said first filter decision value is below said first threshold value as determined by said first comparator and	212
b. Element 7[b]: ii) determine to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value as determined by said first comparator; and.....	213
c. Element 7[c]: said second pixel determiner is configured to i) determine to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value as determined by said second comparator and.....	213
d. Element 7[d]: ii) determine to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second threshold value as determined by said second comparator.	213
5. Claim 9.....	214
a. Element 9[a]: The device according to claim 5, further comprising a threshold determiner configured to i) determine said first threshold value based on a quantization parameter associated with said block, and.....	214

TABLE OF CONTENTS
(continued)

	Page
b. Element 9[b]: ii) determine said second threshold value based on a quantization parameter associated with said neighboring block.	215
6. Claim 12.....	215
a. Element 12[a]: An encoder comprising a filtering control device according to claim 5.....	215
7. Claim 13.....	216
a. Element 13[pre]: A user equipment comprising:	216
b. Element 13[a]: a memory configured to store video frames; and.....	217
c. Element 13[b]: an encoder according to claim 12 configured to encode said video frames into encoded video frames, wherein said memory is further configured to store said encoded video frames.	217
8. Claim 14.....	219
a. Element 14[a]: A decoder comprising a filtering control device according to claim 5.....	219
9. Claim 15.....	219
a. Element 15[pre]: A user equipment comprising:	219
b. Element 15[a]: a memory configured to store encoded video frames;.....	219
c. Element 15[b]: a decoder according to claim 14 configured to decode said encoded video frames into decoded video frames; and	221
d. Element 15[c]: a media player configured to render said decoded video frames into video data displayable on a display.....	221

TABLE OF CONTENTS
(continued)

	Page
10. Claim 16.....	222
a. Element 16 [pre]: A non-transitory computer-readable medium comprising, stored thereupon, a computer program for filtering control of a block of multiple pixels in a video frame each pixel having a respective pixel value, said computer program comprising code means which when run on a computer causes the computer to:	222
b. Element 16[a]: calculate a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $ p_{2i}-2p_{1i}+p_{0i} $ wherein p_{0i} , denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	223
c. Element 16[b]: calculate a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $ q_{2i}-2q_{1i}+q_{0i} $ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;	223
d. Element 16[c]: determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value; and	224

TABLE OF CONTENTS
(continued)

	Page
e. Element 16[d]: determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value, ...	224
f. Element 16[e]: wherein determine how many pixels in said first line of pixels in said block to filter comprises determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and	224
g. Element 16[f]: determine how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.....	224
D. Ground 2B: Obviousness over Aksu in view of Gonzalez, H.241, and B205/AAPA	225
1. Claim 2.....	225
a. Element 2[a]: The method according to claim 1, wherein calculating said first filter decision value comprises calculating said first filter decision value as: $ p_{2_0} - 2p_{1_0} + p_{0_0} + p_{2_3} - 2p_{1_3} + p_{0_3} $ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block	

TABLE OF CONTENTS
(continued)

	Page
boundary and p_{23} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and	225
b. Element 2[b]: calculating said second filter decision value comprises calculating said second filter decision value as: $ q_{20}-2q_{10}+q_{00} + q_{23}-2q_{13}+q_{03} $ wherein q_{00} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{10} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{20} denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary, q_{03} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, q_{13} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and q_{23} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.....	227
2. Claim 6.....	228
a. Element 6[a]: The device according to claim 5, wherein said first decision value calculator is configured to calculate said first filter decision value as: $ p_{20}-2p_{10}+p_{00} + p_{23}-2p_{13}+p_{03} $ wherein p_{00} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{10} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{20} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{03} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said	

TABLE OF CONTENTS
(continued)

	Page
block boundary, p_{13} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{23} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and	229
b. Element 6[b]: said second decision value calculator is configured to calculate said second filter decision value as: $q_{20}-2q_{10}+q_{00} + q_{23}-2q_{13}+q_{03}$ wherein q_{00} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{10} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{20} denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary, q_{03} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, q_{13} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and q_{23} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.....	229
3. Claim 11	230
a. Element 11[a]: The device according to claim 5, further comprising a third comparator configured to compare a sum of said first filter decision value and said second filter decision value to a threshold value, wherein	230
b. Element 11[b]: said first pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator,	

TABLE OF CONTENTS
(continued)

	Page
how many pixels in said first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and	233
c. Element 11[c]: said second pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator.....	233
E. Ground 2C: Obviousness over Aksu in view of Gonzalez, Fu, and B118	234
1. Claim 10	234
a. Element 10[a]: The device according to claim 5, further comprising: a first offset calculator configured to calculate a first offset Δ based on $(9x(q_{0j} - p_{0j}) - 3x(q_{1j} - p_{1j}))/16$ wherein p_{0j} denotes a pixel value of a pixel closest to, in said first line of pixels, said block boundary, p_{1j} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary, q_{0j} denotes a pixel value of a pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary and q_{1j} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary;	234
b. Element 10[b]: a first pixel modifier configured to modify said pixel value of said pixel closest to said block boundary in said first line of pixels by adding said first offset calculated by said first offset calculator to	

TABLE OF CONTENTS
(continued)

	Page
said pixel value of said pixel closest to said block boundary in said first line of pixels;	238
c. Element 10[c]: a second pixel modifier configured to modify said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels by subtracting said first offset calculated by said first offset calculator from said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels;	239
d. Element 10[d]: a second offset calculator configured to calculate, if said first filter decision value is below said first threshold value as determined by said first comparator, a second offset based on $(p0_j + p2_j - 2p1_j + 2\Delta)/4$ wherein $p2_j$ denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;	240
e. Element 10[e]: a third pixel modifier configured to modify, if said first filter decision value is below said first threshold as determined by said first comparator, said pixel value of said pixel next closest to said block boundary in said first line of pixels by adding said second offset calculated by said second offset calculator to said pixel value of said pixel next closest to said block boundary in said first line of pixels;	245
f. Element 10[f]: a third offset calculator configured to calculate, if said second filter decision value is below said second threshold value as determined by said second comparator, a third offset based on $(q0_j + q2_j - 2q1_j - 2\Delta)/4$ wherein $q2_j$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary; and	246

TABLE OF CONTENTS
(continued)

	Page
g. Element 10[g]: a fourth pixel modifier configured to modify, if said second filter decision value is below said second threshold as determined by said second comparator, said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels by adding said third offset calculated by said third offset calculator to said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels.	249
XI. SUMMARY AND ADDITIONAL REMARKS.....	251

I, Dr. Dan Schonfeld, do hereby declare:

I. INTRODUCTION

1. My name is Dan Schonfeld. I am currently a Professor in the Department of Electrical and Computer Engineering at the University of Illinois at Chicago.

2. I have been retained by Petitioner Intel Corporation (“Intel” or “Petitioner”) to submit a Declaration in support of Intel’s Petition for *Inter Partes* Review of claims 1-3, 5-7, 9-16 (“Challenged Claims”) of U.S. Patent No. 9,641,841 (“’841 Patent”) (Ex-1001).

3. I have been asked to opine on whether the Challenged Claims of the ’841 Patent are rendered obvious by the prior art. My opinions are based on my years of education, research, and experience, as well as my investigation and study of relevant materials. The materials that I evaluated in support of this Declaration include all exhibits cited in this Declaration and in the Petition.

4. I may rely upon these materials, my knowledge and experience, and/or additional materials to rebut arguments raised by the Patent Owner. Further, I may also consider additional documents and information in forming any necessary opinions, including documents that may not yet have been provided to me.

5. My analysis of the materials produced in this matter is ongoing, and I will continue to review any new material as it is provided. This Declaration

represents only those opinions I have formed to date. I reserve the right to revise, supplement, and/or amend my opinions stated herein based on new information and on my continuing analysis of the materials already provided.

6. I am being compensated for my services at my standard hourly rate and reimbursed for reasonable expenses that I may incur while working on this matter. I have no financial interest in the outcome of this matter or any other matter that may exist between Petitioners and Telefonaktiebolaget LM Ericsson (“Ericsson” or “Patent Owner”). This compensation does not depend in any way on the conclusions I reach.

II. QUALIFICATIONS

7. My qualifications for forming the opinions set forth in this Declaration are summarized here and include my educational background, career history, publications, and other relevant qualifications. My full *curriculum vitae* is available at Ex-1004, and includes my detailed employment background, professional experience, and list of publications.

8. I received my B.S. degree in Electrical Engineering and Computer Science from the University of California, Berkeley, California, in 1986 with a concentration on Computer Engineering / Systems Engineering. I received my M.S. degree in Electrical and Computer Engineering from The Johns Hopkins University, Baltimore, Maryland, in 1988 with a concentration on Speech Processing /

Biomedical Signal Processing. I received my Ph.D. degree in Electrical and Computer Engineering from The Johns Hopkins University, Baltimore, Maryland, in 1990 with a concentration on Nonlinear Signal Processing / Image Analysis.

9. In August 1990, I joined the Department of Electrical Engineering and Computer Science at the University of Illinois, Chicago, Illinois, where I am a tenured Professor in the Departments of Electrical and Computer Engineering, Computer Science, and Bioengineering. Before I joined the University of Illinois at Chicago, I served as an instructor in the Department of Electrical and Computer Engineering at The Johns Hopkins University, Baltimore, Maryland.

10. At the University of Illinois at Chicago, I have served as the Director of the University-Industry Engineering Research Center (UIERC), formerly the Manufacturing Research Center (MRC). I am also Co-Director of the Multimedia Communications Laboratory (MCL) and a member of the Signal and Image Research Laboratory (SIRL).

11. Over the past few decades, I have also served as a visiting professor in (a) the Advanced Analytics Institute (AAI) at the University of Technology, Sydney, Australia, (b) the Department of Information Engineering and Computer Science (DISI) at the University of Trento, Italy, (c) the School of Computer Engineering at the Nanyang Technological University, Singapore, and (d) the Department of Electrical Engineering—Systems at Tel-Aviv University, Israel.

12. I have been elected Fellow of the Institute of Electrical and Electronics Engineers (IEEE) “for contributions to image and video analysis” as well as Fellow of the International Society for Optics and Photonics (SPIE) “for specific achievements in morphological image processing and video analysis.” I have also been elected University Scholar of the University of Illinois. A complete list of my publications, professional activities, and honors that I have received is fully set forth in my *curriculum vitae*, available at Ex-1004.

13. I have previously served as Editor-in-Chief and Deputy Editor-in-Chief of the IEEE Transactions on Circuits and Systems for Video Technology. I have also previously served as Area Editor for special issues of the IEEE Signal Processing Magazine. I have served as Associate Editor of the IEEE Transactions on Image Processing (on image and video storage, retrieval and analysis), Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology (on video analysis), Associate Editor of the IEEE Transactions on Signal Processing (on multidimensional signal processing and multimedia signal processing), and Associate Editor of the IEEE Transactions on Image Processing (on nonlinear filtering). I have also served on the editorial board of the IEEE Signal Processing Magazine, EURASIP Journal of Image and Video Processing, Research Letters in Signal Processing, and Bentham Science Publishers, Ltd.’s “Recent Patents on Computer Science” and “Recent Patents on Electrical Engineering” publications. I

have served as guest editor of numerous special issues in various journal publications in the area of multimedia systems.

14. I have previously served on the Conference Board of the IEEE Signal Processing Society. I have previously served as Technical Program Chair of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2018 as well as Program Chair of the IEEE Conference on Visual Communications and Image Processing (VCIP) 2015. I have also previously served as General Co-Chair of the Workshop on Big Data in 3D Computer Vision 2013 and the IEEE International Conference on Multimedia and Expo (ICME) 2012. I have served as Chair of the IEEE Workshop on Video Mining 2008 and the SPIE Conference on Visual Communications and Image Processing 2007. I have also served on the organizing committees of various conferences including the IEEE International Conference on Image Processing 1998, 2012, and 2020, IEEE/SPIE VCIP 2010, 2017, and IEEE Workshop on Nonlinear Signal and Image Processing (NSIP) 1997. I was an organizer of the Thematic Symposium on Multimedia Search and Retrieval at ICASSP 2009.

15. I have authored and co-authored over 250 technical papers for various journals and conferences. I was author of a book chapter, entitled: "Image and video communication networks," and later editions entitled: "Video communication networks." I was co-author (with Carlo Giulietti and Rashid Ansari) of a paper that

won the Best Paper Award at the ACM Multimedia Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking 2010. I was also co-author (with Junlan Yang) of a paper that won the Best Student Paper Award at the IEEE International Conference on Image Processing 2007. I was also co-author (with Wei Qu) of a paper that won the Best Student Paper Award at the IEEE International Conference on Image Processing 2006. I was also co-author (with Nidhal Bouaynaya) of a paper that won the Best Student Paper Award in Visual Communications and Image Processing 2006. In addition, many of my publications relate to the broad topic of multimedia systems, which includes audio, image, and video processing. My publications in the area of multimedia systems dates back to 1988. A list of my publications within the past ten years is included in Ex-1004.

16. I have been selected to be the keynote speaker at the IEEE International Conference on Computer, Electrical and Communication Engineering (ICCECE), Kolkata, India, in 2023. I was also the keynote speaker at the N-Brain Alliance Launch & Product Release Conference and the Think for Education Conference (TEC) AI+Education, both held in Beijing, China, in 2018. I was also the keynote speaker at the International Conference on Wireless Communications and Signal Processing (WCSP), Yangzhou, China, in 2016, and the International Conference on Intelligent Control and Information Processing (ICICIP) and International

Conference on Brain Inspired Cognitive Systems (BICS), Beijing, China, in 2013. Further, I was a plenary speaker at the IEEE/IET International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, in 2010, and at the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Genoa, Italy, in 2009. I was also a plenary speaker at the INPT/ASME International Conference on Communications, Signals, and Systems (ICCSS), Rabat Morocco, in 1995 and 2001.

17. I have served as Representative of Regions 1-6 (North America) on the Chapters Committee of the IEEE Signal Processing Society. I have also served as Chairman of the IEEE Signal Processing Chicago Chapter. I have also served on the IEEE Image, Video, and Multidimensional Signal Processing (IVMSP) Technical Committee, formerly the IEEE Image and Multidimensional Signal Processing (IMDSP) Technical Committee, Visual Signal Processing and Communications (VSPC) Technical Committee, IEEE Signal and Image Processing in Medicine Technical Committee, and the IEEE Multimedia Communications Technical Committee. I currently serve on the American National Standards Institute (ANSI)/Underwriters Laboratory (UL) Standards Technical Panel (STP) on Multimedia Systems.

18. I have also taught various courses that relate to multimedia systems. For example, since the late 1990s, I have introduced and taught an advanced

undergraduate-level/first-year graduate-level course on multimedia systems (originally called multimedia communication networks), which focuses on audio, image, and video processing and communications.

19. I have also served as a consultant in various engagements related to multimedia systems. For example, over the past decade, I have served as an expert witness in several cases related to multimedia systems. In 1997, I served as a consultant for Prairiecomm Corp. where, among other tasks, I developed and deployed multimedia systems. Since 2002, I have also served as a member of the American National Standards Institute (ANSI)/Underwriters Laboratory (UL) Standards Technical Panel (STP) on various standards related to multimedia systems.

20. Additional details of my education and work experience, professional activities, awards and honors, and publications that may be relevant to opinions I have formed are set forth in my *curriculum vitae*, which is available at Ex-1004. Additionally, I have consulted for several companies in the area of signal processing and multimedia systems.

III. MATERIALS CONSIDERED

21. The materials that I considered and relied upon in preparing my Declaration and forming my opinions include all exhibits cited in this Declaration

and the Petition, including the '841 Patent, the '841 file history, and all of the relevant prior art.

22. I also have relied on my academic and professional experience in reaching the opinions expressed in this Declaration.

IV. LEGAL STANDARDS

23. Counsel has informed me of the legal principles that apply in determining whether patent claims are unpatentable for obviousness. Paragraphs in this section beginning with “I understand” or “I have been informed” capture my understanding of the law as provided to me by counsel. The other paragraphs are my opinion.

24. I understand that a claim may be unpatentable as obvious under § 103(a) if the subject matter described in the claim as a whole would have been obvious to a hypothetical “person of ordinary skill in the art” (“POSITA”) in view of a prior-art reference, or combination of prior-art references, at the time the claimed invention was made.

25. I understand that “the time the claimed invention was made” refers to the “priority” or “critical” date of the invention.

26. I understand that, before considering the prior art, the claims are first construed to determine what subject matter the claims encompass. I understand that the claims are read from the perspective of POSITA at the time of the alleged

invention. I understand that the words of a claim are generally given their ordinary and customary meaning as understood by a POSITA in light of the entire patent specification in which they appear, as well as the prosecution history of the patent.

27. I understand, however, that the words of a claim may be given a meaning other than their plain and ordinary meaning if: (1) a patentee expressly and unambiguously defined a term in a way that is different from its plain meaning; or (2) a patentee “disclaims” the full scope of a claim term’s plain meaning, for instance, by making clear and unambiguous statements during prosecution that a claim term does or does not encompass certain subject matter that may otherwise be encompassed by the plain words of the claim.

28. I understand that, after construing the claims, the teachings of the prior art are considered. I understand that the prior art includes references, among other things, that were known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the date the invention was allegedly made by the inventor(s). For a printed publication to qualify as prior art, I understand that it must be demonstrated that the publication was disseminated or otherwise sufficiently accessible to the public. However, I also understand that a POSITA is assumed to be aware of all relevant prior art in the field of endeavor covered by the patent-at-issue and all analogous prior art.

29. I understand that although the ultimate question of the obviousness of a claimed invention is a legal determination, obviousness is based on several factual inquiries, including: the scope and content of the prior art, the differences between the claimed subject matter and the prior art, the level of ordinary skill in the art at the time of the invention, and any “objective indicia” or “secondary considerations” of non-obviousness, which must all be considered.

30. I understand that, in determining the scope and content of the prior art, a prior-art reference is considered relevant if it falls within the field of the inventor’s endeavor, or if it is reasonably pertinent to the particular problem that the inventor was trying to solve. A reference is reasonably pertinent if it logically would have commended itself to an inventor’s attention in considering the problem sought to be solved.

31. I understand that, to assess the differences between prior art and the claimed subject matter, the claimed invention must be considered as a whole. I understand that this involves showing that a POSITA, confronted by the same problems as the inventor and with no knowledge of the claimed invention, would have identified the elements in the prior art and combined them in the claimed manner.

32. I understand that there are several rationales for combining prior-art references. For instance, I understand that it is considered obvious to:

- combine prior-art elements according to known methods to yield predictable results;
- substitute one known element for another to obtain predictable results;
- use the prior-art elements in a predictable way according to their established functions;
- apply a known technique to a known device (method or product) ready for improvement to yield predictable results;
- choose from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- or if there is some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify a prior-art reference or combine prior-art teachings to arrive at the claimed invention.

I understand, however, that a POSITA does not need to have the same motivations or reasons to combine the prior art as the inventors, and can arrive at the claimed invention for completely different reasons than the inventors.

33. I also understand that a claim may be considered obvious in light of the prior art, even if no one prior-art reference provided its exact teachings, if the general conditions of the claimed subject matter are disclosed in the prior art, and the claimed invention is simply an optimization of the prior art that could be achieved through routine experimentation. I understand that, for a claim to be obvious on this basis, it

must be demonstrated that a POSITA would have recognized that a particular variable in the prior art was “result-effective”—meaning that modifying the variable was known to affect a particular result—and optimizing that variable would have been within the skill of a POSITA.

34. However, I understand that even if the general working conditions of the claimed subject matter were known, the claims may nonetheless be non-obvious if: (1) the claimed invention produces a new and unexpected result that is different “in kind” and not merely “in degree” from that of the prior art; (2) the prior art taught away from the claimed invention; (3) the claimed variable was not recognized as “result-effective”; or (4) the prior art disclosed a broad range that would not invite routine optimization. I also understand that a difference “in kind” means some transformative difference beyond what was recognized by the prior art, and not merely a change of percentages in, *e.g.*, efficiency.

35. I understand that, when considering the prior art, the prior art cannot be modified or combined using “hindsight.” I understand that hindsight refers to situations in which the patent-at-issue is used as a framework to pick and choose pieces from the prior art to arrive at the claims, or adopting rationales identified by the patent-at-issue that those in the art would not have recognized. Instead, I understand that the obviousness inquiry is evaluated from the perspective of a POSITA as of the critical date of the patent-at-issue, with all the prior art before

them, to determine whether the POSITA would have independently arrived at the claimed invention without the benefit of the patent-at-issue for guidance.

36. I understand that “objective indicia” or “secondary considerations” of non-obviousness must also be considered in the obviousness analysis, and include whether: (1) there was a long-felt need for the claimed invention; (2) the claimed invention has achieved commercial success; (3) there was copying of the claimed invention by others; (4) there were failed attempts by others to make the alleged invention; (5) there was praise of the invention in the field; and (6) there was skepticism the claimed invention could be achieved, among a few others.

37. As of the time that I prepared this Declaration, the Patent Owner had not identified any secondary considerations of non-obviousness. Therefore, I reserve the right to elaborate on my understanding of these secondary considerations and to address any secondary considerations that Patent Owner may subsequently identify.

38. Finally, I understand that, in an *inter partes* review proceeding, the obviousness of a claim must be demonstrated by “a preponderance of the evidence,” and that this burden falls on the Petitioner. I understand that the preponderance of the evidence standard means that a reasonable factfinder would find a material fact more probable than the nonexistence of that fact. It does not allow for speculation regarding specific facts and is instead focused on whether the evidence more likely than not demonstrates the existence or non-existence of specific material facts. I

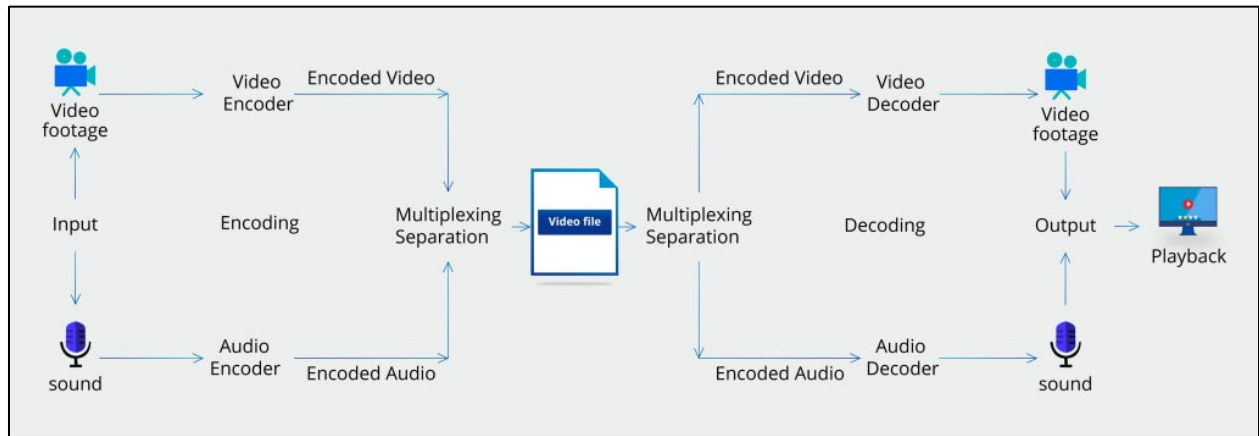
understand that “preponderance of the evidence” is a lower standard than “clear and convincing evidence” (which requires a fact to be substantially more likely to be true than untrue), or “beyond a reasonable doubt” (which is an exceedingly high standard that I understand is generally reserved for criminal matters).

39. I have applied the “preponderance of the evidence” standard throughout my Declaration.

V. TECHNOLOGY BACKGROUND

A. Video Encoding and Decoding

40. Digital videos must generally undergo an encoding process to convert the images (also called “pictures” or “frames”) that make up the video into a binary bitstream that can be processed and stored by computers. As part of the encoding process, videos are usually “compressed” to reduce the amount of data needed to represent the video. When the video is to be displayed on a computer screen, television, or some other display device, the binary bitstream undergoes a decoding process that essentially reverses the encoding process to reconstruct the video images from the bitstream.



Ex-1031, 1.¹

41. The encoder encodes video images by performing several steps. At a high level, the key steps involved in the encoding of a video image are provided below.

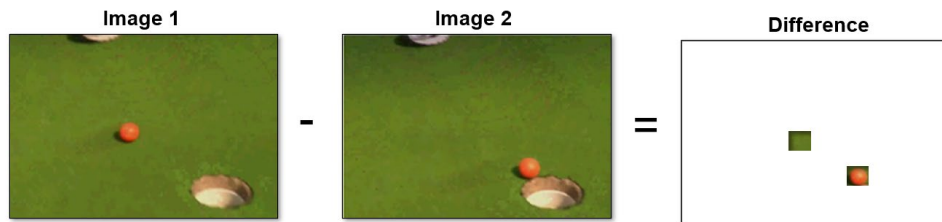
- a. **Blocking:** The encoder subdivides the image into blocks and processes each block individually to alleviate demands on processing resources. By way of example, an image may be divided into blocks of 16x16 pixels, 8x8 pixels, or even 4x4 pixels, although it was known in the art that different-sized blocks could be used.

¹ Although some of the images I identify in the technology background may post-date the earliest-possible priority date of the patent, they are provided only as useful demonstratives of certain concepts, and the aspects on which I am relying on them were known prior to the earliest-possible priority date of the patent.



Ex-1070, 4.

- b. **Prediction and Residuals:** The encoder achieves much of the compression by characterizing, when possible, only the difference between different images in the video.



The encoder characterizes the differences between images in two ways: (1) the motion of objects between two images; and (2) any remaining “residual” differences between the images. First, the encoder characterizes the motion of objects by taking a block in the current image and searching a previously encoded “reference” image to find a set of pixels with a similar arrangement of pixels as the current block. The encoder then characterizes how those pixels have moved between the two images in the form of a “motion vector.” The encoder will then apply the motion vector to identify

pixels in the reference image to create a “prediction” block of the current image. This “prediction” block is then subtracted from the current block to capture any remaining differences between the current block and prediction block, which is called “residual” data. If the prediction is accurate, then the difference is small, which significantly reduces the amount of data that is output by the encoder. At least two types of prediction were commonly used prior to the '841 Patent: (1) inter-picture prediction, which predicts a current block based on the corresponding block in a reference frame, and (2) intra-picture prediction, which predicts a current block based on adjacent blocks in the same frame.

- c. **Transforms:** The encoder then applies an orthogonal transform (Discrete Cosine Transform, or “DCT”) to the residual data to transform them to another domain in order to achieve further compression. Specifically, the data is transformed from the block’s spatial data to frequency data. The human eye is more sensitive to the information contained in low frequencies (corresponding to larger, less-variable features in an image) than to the information contained in high frequencies (corresponding to smaller, rapidly changing features). The DCT separates the more perceptually significant information from less perceptually significant information by evaluating the contribution of different frequency components to the residual data to produce a “DCT coefficient” characterizing each frequency component’s contribution. After the DCT, the compression algorithm encodes the low-frequency DCT coefficients with high precision, but uses fewer bits to encode the

high-frequency DCT coefficients as part of a “quantization” process.

- d. **Entropy encoding:** Finally, the encoder applies an entropy coding technique to convert the DCT coefficients into codewords to achieve further compression. An entropy coding technique maps frequently occurring DCT coefficients to shorter codewords and infrequently occurring DCT coefficients to longer code words to achieve compression. This is also known as “variable length coding.” One early example of this process is “Morse code,” in which more-often used letters (*e.g.*, “e” and “t”) are assigned the shortest codewords (one dot or one dash, respectively), and less-used letters have progressively longer codes. The decoder performs the reverse and employs the same mapping to convert the codewords into the DCT coefficients.

42. In sum, the output of the encoder is a bitstream that contains a complete representation of a video image, or when an image is based on a “prediction” of another fully encoded image, the motion vectors and residuals needed to reconstruct the predicted image. When the decoder receives the bitstream containing fully encoded images, or the motion vectors and residual data for predicted images, it will then reverse the encoding process. For instance, whereas the encoder calculates a motion vector by comparing pixels in the current block to a reference, the decoder will apply the motion vector to a reference image to create the prediction block. Then, the decoder will reverse the quantization and transform process performed by

the encoder—recreating the “residual data” from its compressed, encoded form in the frequency domain back into spatial pixel data—and *add* the residual data back to the prediction block to create a reconstructed version of the current block.

43. An exemplary overview of the encoding process consistent with the one I have described above can be found at Ex-1040, 28-35 (motion compensation), 42-43, 45-47 (transform), 50-52 (quantization), 58 (entropy encoding); Ex-1030.

B. Deblocking Filters

44. One drawback of block-based encoding and decoding is that reconstructed images may contain perceptible pixel-color changes at the boundaries of neighboring blocks, called “blocking artifacts,” because of the way each block is processed and quantized separately. For instance, because the transform and quantization process employed during encoding may discard higher-frequency DCT coefficients so that the encoded video requires less data than the original video, when the video image is subsequently decoded and reconstructed using predominantly lower-frequency DCT coefficients, the high-frequency details in the original image would not be present, which contributes to the generation of blocking artifacts. An example of a reconstructed image with noticeable “blocking artifacts” is shown below.



Ex-1052, 1372.

45. To reduce the apparent visibility of these blocking artifacts, reconstructed images can be processed using a “deblocking filter.” In essence, a deblocking filter will evaluate the boundaries between adjacent blocks and, when certain conditions are met, apply a filter to blend the pixels along the boundary between neighboring blocks. As shown in the example below, the lines that were visible between different blocks in the image above are less noticeable.

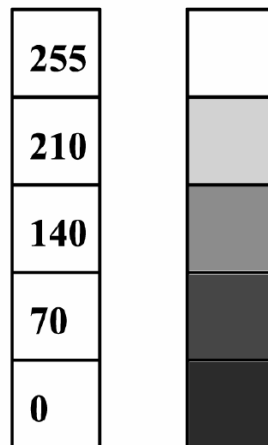


Ex-1052, 1372.

1. Pixel Values

46. Deblocking filters operate on the pixels that make up each video image. Digital video images are made up of pixels, which can be mapped to pixels on an output device (such as a computer display or television). For instance, each image in a 1080p video typically has 1920x1080 pixels, and each image in a 4K video typically has 3840x2160 pixels.

47. Each pixel in a video image can be represented by a pixel value. For instance, an 8-bit greyscale image means that each pixel can be represented using 8 bits of data (8 zeros or ones), which would result in 256 (or 2^8) possible pixel values. *See, e.g.,* Ex-1034, 2. Each pixel in an 8-bit greyscale image can therefore be assigned a value between 0 (pure black) to 255 (pure white), as shown below:



Id.

48. The pixels in a color image can be represented using one of several color-representation schemes. One such scheme is called the RGB scheme that relies

upon Red, Green, and Blue—the three primary colors. *Id.*, 6. With this scheme, each color component (red, green, and blue) can be assigned, for instance, its own 8-bit value—*each* with a value between 0 and 255—such that each pixel is represented by 24 bits in total. *Id.* This produces approximately 16.7 million (or 2^{24}) possible color combinations. *Id.*

R / G / B		
255/0/0		
0/255/0		
0/0/255		
0/0/0		
255/255/255		
128/128/128		

When a pixel is displayed, the three color components are combined to realize the final color. *See id.*, 7.

49. Alternately, the same information regarding the color of a pixel can be represented by Luminance or Luma (Y) and three color differences. Luminance (Y) is a weighted average of the R, G and B components of a pixel. Ex-1029, 3; Ex-1040, 13-14. The color differences (Chrominance or Chroma) represent the differences between the three color components and the luminance:

$$CR = R - Y$$

$$CB = B - Y$$

$$CG = G - Y$$

Ex-1029, 3; Ex-1040, 13-14.

50. Because the sum of CR, CB and CG is constant, only two of the three values are used. The third value can be derived from the other two. Typically, CR and CB are used along with Y to represent colors. Ex-1029, 4; Ex-1040, 14. In the Y:CR:CB scheme, only the Luminance/Luma (Y), the red chroma (CR) and blue chroma (CB) are used. The human eye is more sensitive to luminance than to color. Ex-1040, 13. As a result, the CR and the CB components can be represented with a lower resolution than the Luminance component without affecting the visual quality of the image. *Id.* Thus, the number of bits required to represent an image in the Y:CR:CB scheme is fewer than the bits needed in the RGB scheme. Many video codecs—such as the H.263 and H.264 Standards—describe the data contained in macroblocks and blocks using the Y:CR:CB scheme. Ex-1037, 4 (“As in previous video coding Recommendations and International Standards, a macroblock, consisting of a 16x16 block of luma samples and two corresponding blocks of chroma samples, is used as the basic processing unit of the video decoding process.”).

51. Because pixels are represented as numeric values, they can also be operated on, such as added, subtracted, multiplied, and divided. When digital images are edited on a computer, for instance, to change the brightness or contrast, or apply pre-determined “filters,” behind the scenes, mathematical equations modify the pixel values.

52. Regardless of the pixel-value scheme used, a deblocking filter implements mathematics to alter the pixel values of pixels near block boundaries in an attempt to smooth out the differences between neighboring blocks. A simplistic example of how a deblocking filter might operate is provided below, showing two neighboring blocks (a “P” block and “Q” block) that each have particular pixel values (100 or 175, respectively), that produce a perceptible change at the boundary.

P Block			Q Block		
p_2	p_1	p_0	q_0	q_1	q_2
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175

53. There are two primary ways to calculate new pixel values during deblocking. The first is to calculate the new pixel value directly using a set of

original pixel values. For instance, using an averaging filter, a new pixel value for the pixel closest to the boundary (p_0 , which I label p_0') can be calculated using the following example equation:

$$p_0' = \frac{p_1 + p_0 + q_0}{3} = \frac{100 + 100 + 175}{3} = 125$$
$$q_0' = \frac{q_1 + q_0 + p_0}{3} = \frac{175 + 175 + 100}{3} = 150$$

Alternatively, an intermediate “delta” or “offset” (Δ) value can be calculated, which is then *applied* to the original pixel values (*e.g.*, added to one side and subtracted from the other). For instance, one example offset equation could be:

$$\Delta = \frac{q_0 - p_0}{3} = \frac{175 - 100}{3} = 25$$
$$p_0' = p_0 + \Delta = 100 + 25 = 125$$
$$q_0' = q_0 - \Delta = 175 - 25 = 150$$

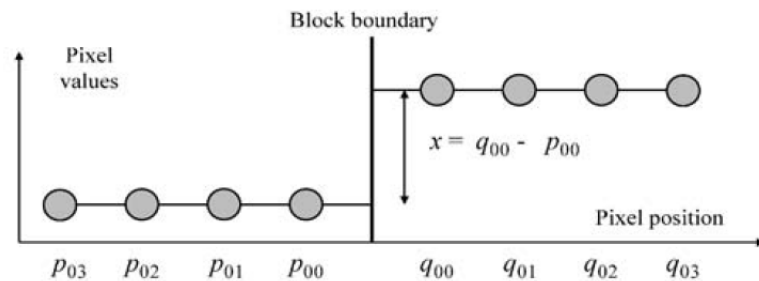
While these two different techniques produced the same result in this very simplified example, that is not always the case, because how much a pixel is modified depends on the specific equations and pixel values used. Regardless, modifying the pixel values in this example using these example equations produces a smoother transition between the two blocks:

		p_0	q_0		
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175
100	100	100	175	175	175

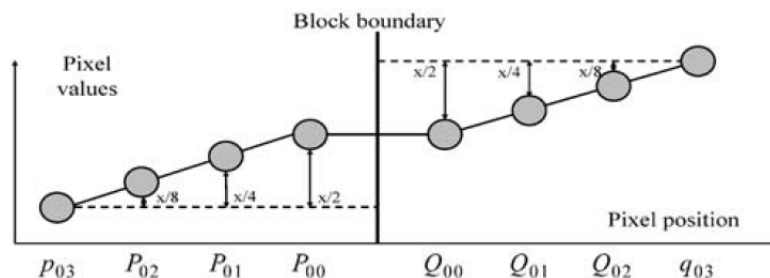
→

		p_0	q_0		
100	100	125	150	175	175
100	100	125	150	175	175
100	100	125	150	175	175
100	100	125	150	175	175
100	100	125	150	175	175

This can also be depicted graphically, as the image below shows how multiple pixels are filtered to produce a smooth transition over a block boundary by calculating the difference across the boundary (“x”) and dividing the difference by a larger integer (2, 4, 8) the further the pixel is from the block boundary to produce a smooth transition:



(a)



Ex-1039, 215.

2. Filters

54. Pixel values are often “filtered” based on the frequency characteristics of the image block, which refers to how pixel values fluctuate over a given space. For instance, below is a signal that consists of a set of pixels that are considered to have a relatively “low frequency”:

150	160	170	180	170	160	150	160
-----	-----	-----	-----	-----	-----	-----	-----

See Gonzalez, 258 (“low frequencies ... are related to slowly varying intensity components in an image”). As can be seen, the pixel values are not changing significantly over the 8-pixel space shown. Compare this, however, to a signal that consists of pixel values with a relatively “high frequency”:

0	150	255	150	0	255	0	150
---	-----	-----	-----	---	-----	---	-----

Id. (“high frequencies are caused by sharp transitions in intensity, such as edges and noise.”).

55. Although filters can serve many different purposes, at a high level, filters are often classified into one of three frequency-discriminator filters: “low-pass” filters, “high-pass” filters, and “band-pass” filters.

56. A low-pass filter means that the filter is intended to allow lower-frequency components to “pass” through from input to output, and filter (or suppress) high-frequency components. Gonzalez, 258. In image processing, a low-

pass filter reduces rapid changes in pixel values. One use of a low-pass filter is to remove spurious data such as random noise. A simple way to achieve this is by averaging filters, in which the filter takes multiple pixel values as inputs and produces an average value as its output, thereby reducing rapid changes in pixel values. An artifact of low-pass filtering, including simple averaging, is the introduction of “blurring” to the filtered image. *Id.* (explaining how a “lowpass filter ... would blur an image.”). For example, if we average each pixel value in the above low-frequency signal example based on the values of its two neighboring pixels (except for the left-most and right-most pixel, which have only one neighboring pixel), the following output would be provided:

Input (Low-Frequency Signal):

150	160	170	180	170	160	150	160
-----	-----	-----	-----	-----	-----	-----	-----

Output (Low-Pass Filtered, Low-Frequency Signal):

155	160	170	173	170	160	157	155
-----	-----	-----	-----	-----	-----	-----	-----

As can be seen, the pixel values of the low-pass filtered version of the low-frequency signal do not fluctuate much, and in fact, only a few pixels were modified at all. But if this same filter were applied to the high-frequency example signal, the following output would be provided:

Input (High-Frequency Signal):

0	150	255	150	0	255	0	150
---	-----	-----	-----	---	-----	---	-----

Output (Low-Pass Filtered, High-Frequency Signal):

75	135	185	135	135	128	135	75
----	-----	-----	-----	-----	-----	-----	----

A low-pass filter therefore has a much more significant impact on high-frequency signals than low-frequency ones.

57. A “high-pass” filter does the opposite of a low-pass filter: it allows high-frequency components to “pass” through the filter, and therefore *accentuates* changes in pixel value. One use of a high-pass filter is to perform a “sharpening” operation, in which the filter takes multiple pixel values as inputs and amplifies differences in pixel values to thereby increase rapid changes in pixel values. Gonzalez, 258 (explaining how a “highpass filter ... would enhance sharp details”). For instance, if we apply a high-pass filter (*e.g.*, [-1, 3, -1]) to the low-frequency signal (except for the left-most and right-most pixel, which have only one neighboring pixel), the following output would be provided:

Input (Low-Frequency Signal):

150	160	170	180	170	160	150	160
-----	-----	-----	-----	-----	-----	-----	-----

Output (High-Pass Filtered, Low-Frequency Signal):

130	160	170	200	170	160	130	180
-----	-----	-----	-----	-----	-----	-----	-----

As can be seen, the differences in pixel values are accentuated with the high-pass filter. Because the example high-frequency signal I provide at the beginning of this section is already extreme and would not be as helpful in demonstrating the changes to pixel values imparted by a high-pass filter, I have used a different, slightly lower high-frequency signal as the input example to demonstrate how a high-pass filter would further accentuate the high-frequencies in its output:

Input (High-Frequency Signal):

50	150	165	150	50	175	100	150
----	-----	-----	-----	----	-----	-----	-----

Output (High-Pass Filtered, High-Frequency Signal):

0	235	195	235	0	255	0	250
---	-----	-----	-----	---	-----	---	-----

58. Finally, a “band-pass” filter has an intermediate effect in that it removes both high-frequency and low-frequency components, while preserving the intermediary-frequency components of the signal. It is commonly used, for example, in radio and television communications systems where a message is transmitted over

a specific channel (*i.e.*, frequency range) and a band-pass filter is used to extract the message.

59. There are several ways to evaluate the frequency characteristics of a signal. One way to identify the fluctuations in pixel values is by using a high-pass filter, not to modify the pixel values as shown above, but simply to evaluate their frequency attributes. For instance, a simple difference operation—otherwise known as an approximation of the “first-order derivative”—between adjacent pixels can be performed. In physics terms, this is like measuring the slope or velocity between two pixels.

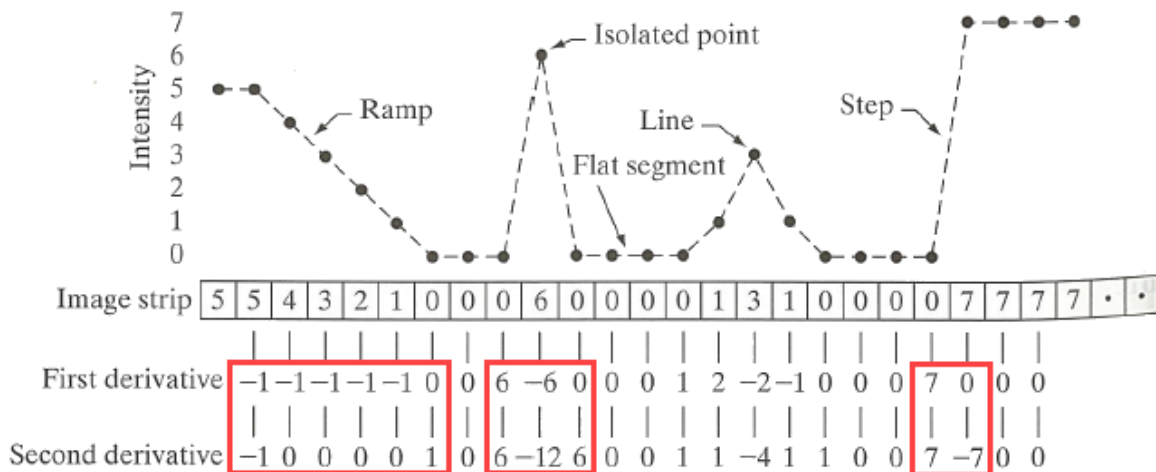
60. However, measuring the slope or velocity of change is not always the preferred approach to evaluating pixels, because it will respond to *any* changes in pixel values. For instance, if a series of pixels gradually change in value (like the low-frequency example signal above), the first-order derivative would indicate a change at every pixel. For this reason, in image processing, first-order derivatives were known to produce “thick” edges when evaluating an image because they would measure changes over a wider area. *See, e.g.*, Gonzalez, 694; Ex-1041, 154 (Fig. 5.8 noting first-order derivative will capture the entire range between points “a” and “b”).

61. Another preferred way to identify a change in pixel values is by calculating the “difference-of-difference”—otherwise known as an approximation

of the “second-order derivative”—in pixel values. Such a calculation requires a minimum of three pixels (*e.g.*, p_2 , p_1 , p_0), and, as the name suggests, takes the difference of the differences: $((p_0 - p_1) - (p_1 - p_2)) = (p_0 - 2p_1 + p_2)$. In physics terms, this is like measuring the acceleration between pixels.

62. Because a second-order derivative responds only to *changes* in the change of pixel-values, it would not respond to gradual, consistent changes like the first-order derivative would, but would respond to rapid changes in pixel values. *See, e.g.*, Gonzalez, 693-94; Ex-1041, 154. For this reason, second-order derivatives were preferred for assessing the location of fine details in an image. *See* Gonzalez, 693-95.

63. An example of the different responses to a first- and second-order derivative is shown below:



Gonzalez, 694 (annotations added).

64. Typically, systems use either a first- or second-order derivative to assess pixel changes for simplicity. Gonzalez, 692.

3. Deblocking Filter Development

65. Prior to the '841 Patent, deblocking filters had been routinely used in video coding systems. Each typically followed the same two-step approach: (1) analyze the pixels near a block boundary to determine whether they need to be filtered; and if it is determined that filtering should be performed, (2) filter/modify the pixel values. Within this framework, prior-art deblocking filters largely differed in three ways: (1) what conditions were used to decide whether deblocking filtering is performed; (2) how many pixels to filter as part of deblocking; and (3) the equations used to filter/modify the pixel values.

66. To illustrate, I have provided below a walkthrough of the International Telecommunication Union ("ITU") standards that have been released over the last two decades, but many other deblocking filters existed in the prior art.

a. H.263

67. In 1998, the International Telecommunication Union's Telecommunication Standardization Sector ("ITU-T") released a revised version of the H.263 standard that, for the first time for the ITU, provided an optional deblocking filter as part of its standardized video decoder. Ex-1035, 4 (§ 3.4.8). The standard called for the deblocking filter to be implemented in-loop, meaning that the

filter is applied during the image-reconstruction process so that the reference images used to predict current images would have already had the filter applied. *Id.*, 4, 81 (J.3); Ex-1039, 1.

68. Deblocking filters could also be implemented in “post-processing,” which means the deblocking filter is applied *after* all of the decoding operations have been performed but before the video image is displayed on screen. Ex-1033, 188; Ex-1039, 211. In other words, in post-processing deblocking, the reference images used to predict current images would maintain the blocking artifacts during prediction, and the artifacts would only be eliminated later. However, it has been preferred to keep deblocking filters “in-loop” because that produced more accurate predictions during encoding and decoding. Ex-1033, 188.

69. In H.263, a decision is first made whether to apply the deblocking filter at all by checking the type of macroblock to which the block belonged. Ex-1035, 82. If the correct conditions are met, then a deblocking filter would be applied. *Id.* Using A and B to represent pixels in one block, and C and D to represent pixels in a neighboring block, the H.263 standard provided the equations that would be applied to perform deblocking. *Id.*

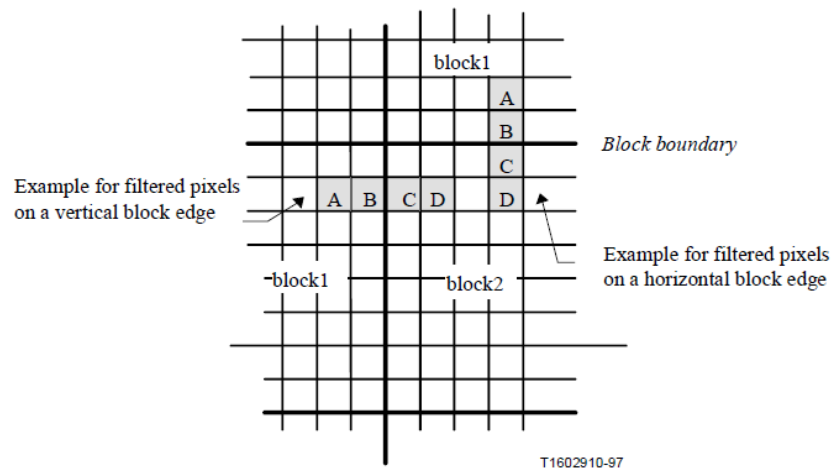


Figure J.1/H.263 – Examples of positions of filtered pixels

Id.

70. To perform deblocking, first, a preliminary “offset” value (d) is calculated by weighting the pixels near the block boundary using the equation:

$$d = \frac{A - 4B + 4C - D}{8}$$

Id. This preliminary offset value is then processed through a “ramp” function that linearly adjusts the offset depending on the intended “strength” of the ramp. *Id.* The “strength” of the ramp is, in turn, controlled by a “quantization parameter,” *id.*, 83, which characterizes how much the block was quantized (or compressed) during the encoding process. In other words, the more the block was quantized during encoding, the stronger the filter would be. The ramp function would then output a modified offset value “ $d1$.” *Id.*, 82.

71. This “d1” offset value would then be used to modify the pixel values of the pixels closest to the block boundary. Specifically, the d1 value would be added to the value of pixel B to produce a new pixel value B1 (or $B1=B+d1$), and subtracted from the value of pixel C to produce a new pixel value C1 (or $C1=C-d1$). *Id.* Applying the offset to pixels B and C blends the transition between the two blocks. However, to ensure the new pixel values for B1 and C1 would stay within the range of 0-255 (the only valid pixel values when using 8 bits), the output of the operation $B+d1$ and $C-d1$ calculations would be “clipped.” *Id.*, 81-82.

72. The clipping function serves as a limit on an output, defining the minimum and maximum values that can be output. For instance, because 8-bit pixels must have a value between 0 and 255, if $B+d1$ produced a value of 258, the clipping function would instead output the maximum value of 255. Or if $C-d1$ produced a pixel value of -4, the clipping function would output the minimum value of 0 instead. *See id.* The clipping function just ensures the final output stays within a specified window of possible pixel values.

73. Then, a second-offset value “d2” would be calculated for pixels A and D using the equation:

$$d2 = \frac{A - D}{4}$$

Id., 94. However, the value of $d2$ is also “clipped” to be, at most, one-half of $d1$. *Id.*; *see also id.*, 95 (“the function $\text{clipd1}(x, \text{lim})$ clips x to the range $\pm \text{abs}(\text{lim})$.”). The value of $d2$ would then be subtracted from A to produce a new pixel value $A1$ (or $A1=A-d2$), and added to D to produce new pixel value $D1$ (or $D1=D+d2$). Pixels A and D would therefore be modified *less* than pixels B and C , because they are further from the block boundary.

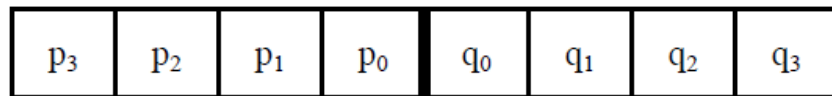
b. H.264

74. After the release of H.263, ITU-T formed a joint venture with the Motion Picture Expert Group (“MPEG”)—another standards-setting organization for multimedia coding—called the “Joint Video Team” (“JVT”) to develop its successor. When ITU-T/MPEG released the H.264 standard in 2003, the deblocking filter was no longer optional, and its operation was more complex.

75. First, the H.264 standard² outlines a process to evaluate the boundary strength (characterized as variable “bS”) between two neighboring blocks, and assign the boundary strength a number between 0 and 4. Ex-1037, 204-05. A bS of zero means no filtering is to be applied, and the higher the bS number, the more filtering that will be applied. *Id.*, 206 (Equation 8-468).

² I will refer to the March 2010 version of the H.264 standard, which is prior to the earliest-possible priority date of the ’841 Patent.

76. When the bS is greater than 0 but less than 4, the deblocking filter will evaluate additional conditions to determine whether to perform deblocking. Specifically, three calculations are performed— $|p_0 - q_0|$ (the pixel difference across the boundary), $|p_1 - p_0|$ (the pixel difference of the pixels nearest the boundary in block p), and $|q_1 - q_0|$ (the pixel difference of the pixels nearest the boundary in block q)—which are each compared to a specific “threshold” value that is based on the quantization parameters for the two blocks being evaluated. *Id.*, 205-06 (Equation 8-468).



Id. 202. In other words, the H.264 standard checks the difference in pixel values across the block boundary ($p_0 - q_0$) and the difference in pixel values of the pixels nearest to the block boundary on each respective side ($p_1 - p_0$ and $q_1 - q_0$), and compares each to a threshold to determine whether to perform deblocking.

77. Then, intermediate “threshold variables a_p and a_q are derived” using the following equations:

$$a_p = \text{Abs}(p_2 - p_0)$$

$$a_q = \text{Abs}(q_2 - q_0)$$

Id., 207 (equations 8-471, 8-472). In other words, H.264 calculates the difference between the pixel closest to the block boundary and third-closest pixel to the block boundary for each of the p and q blocks, separately.

78. The H.264 standard then provides the filtering that is to be applied. Specifically, the offset is calculated using the equation:

$$\Delta_{p0,q0} = \text{Clip3}(-tC, tC, (((q0 - p0) \ll 2) + (p1 - q1) + 4) \gg 3))$$

Id., 207.

79. This equation is written in what is informally called “bit-shift” form. This equation has two particular “bit-shift” operations: a left ($\ll$) shift, and a right ($\gg$) shift. These operations are approximations of the mathematical processes of multiplication and division, respectively. However, they are performed on the actual binary bits of data stored by the computer, which uses less computational resources than actually performing multiplication or division. And because the “bit-shift” operations operate on binary bits of data, they can be implemented in hardware configurations as well as software.

80. Bit-shift operations take advantage of the way numbers are represented and related to one another in binary form. Below is a table of example numerical values and their binary counterparts:

Number	Binary Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

81. What is notable about this representation is that every number representing a power of 2 (*e.g.*, $1=2^0$, $2=2^1$, $4=2^2$, $8=2^3$) has a single “1” in its binary form that shifts one binary unit to the left as the value increases, and the numbers between them follow a pattern of alternating between “0” and “1” to the right of the leading “1.” It is this relationship that bit-shift operations take advantage of.

82. A “left-shift” operation—which is effectively a multiplication operation—means that each binary value is shifted to the left, and a zero is appended to the right. Thus, a left shift of $\ll n$ would shift all bits “n” to the left and add “n” zeros to the right. This equates to multiplying the number by 2^n . Some examples are provided below.

Input Number	Input Binary	Shift Operation	Math Form	Output Binary	Output Number
1	0001	$1 \ll 1$	$1 * 2^1$	0010	2
1	0001	$1 \ll 2$	$1 * 2^2$	0100	4
1	0001	$1 \ll 3$	$1 * 2^3$	1000	8
2	0010	$2 \ll 1$	$2 * 2^1$	0100	4
2	0010	$2 \ll 2$	$2 * 2^2$	1000	8
3	0011	$3 \ll 1$	$3 * 2^1$	0110	6
4	0100	$4 \ll 1$	$4 * 2^1$	1000	8

83. A “right-shift” operation does something similar, but in the opposite direction, and represents an integer-approximation of division by 2^n . Some examples of right-shift operations are shown below:

Input Number	Input Binary	Shift Operation	Math Form	Output Binary	Output Number
8	1000	$8 \gg 3$	$8/2^3$	0001	1
8	1000	$8 \gg 2$	$8/2^2$	0010	2
8	1000	$8 \gg 1$	$8/2^1$	0100	4
7	0111	$7 \gg 2$	$7/2^2$	0001	1
7	0111	$7 \gg 1$	$7/2^1$	0011	3
6	0110	$6 \gg 1$	$6/2^1$	0011	3

There are a few additional noteworthy elements about the right-shift operation. First, the output of a right-shift operation is always an integer, even when division would produce a decimal or fractional remainder. Second, the output is always rounded down to the next-lowest integer value, not the integer nearest the decimal or

fractional remainder. For instance, when taking $7 \gg 2$, the output is 1; it is not 1.75 (the true value of $7/2^2$ or $7/4$), or 2 (1.75 rounded to its nearest integer).

84. To round to the nearest integer, a rounding value is added to the shift operation equivalent to half the value of the denominator (or $2^n/2$). For instance, because “ $\gg 3$ ” equates to dividing by eight, to ensure each output is rounded to the nearest integer, a value of 4 should be added to each input number. And because “ $\gg 2$ ” equates to dividing by 4, a value of 2 should be added to each input number. I have rearranged the examples above to show how this would apply in practice.

Right-Shift Operation	Input Number	Rounding Value	New Input Number	Input Binary	Output Binary	Output Number
$\gg 3$	8	+4	12	1100	0001	1
$\gg 2$	8	+2	10	1010	0010	2
$\gg 2$	7	+2	9	1001	0010	2
$\gg 1$	7	+1	8	1000	0100	4

Even when the rounding value is added to, *e.g.*, an input of “8,” the output remains the same as if the rounding value had not been added. That is because the extra right-facing bits that are added to the binary value of 8 (1000) for 12 (1100) and 10 (1010) are ultimately eliminated by the right-shift operation anyway. The output, however, *does* change when the rounding value is added to, *e.g.*, an input of “7” (0111) because there is a new left-facing bit that is not eliminated by the right-shift operation (*e.g.*, for 9, 1001; and 8, 1000) by the right-shift operation. Adding the

rounding value to odd numbers or even numbers not divisible by a power of 2 ensures the output number is rounded to the nearest integer value, rather than always rounded down.

85. Returning to the H.264 standard, then, the offset equation $((((q_0 - p_0) \ll 2) + (p_1 - q_1) + 4) \gg 3)$ can be rewritten in division form by applying each bit-shift operation and dropping the rounding value (with the understanding that the output of the division form would be rounded to the nearest integer as well):

$$\frac{4(q_0 - p_0) + (p_1 - q_1)}{8}$$

86. According to the H.264 standard, this offset (subject to clipping) is then added to the pixel value for p_0 and subtracted from the value q_0 to produce new pixel values for the pixels closest to the block boundary. *Id.*, 207.

87. After filtering the pixels closest to the block boundary, H.264 discloses that a decision is made whether to filter the pixel second closest to the block boundary in each block, separately. *Id.*, 207-208. Specifically, using the block-specific evaluations previously calculated ($a_p = \text{Abs}(p_2 - p_0)$; $a_q = \text{Abs}(q_2 - q_0)$), if a specific flag (chromaStyleFilteringFlag) is set, and a_p is less than a threshold (β), pixel p_1 is filtered; otherwise, pixel p_1 is not filtered. *Id.* And if a_q is less than a threshold (β) (and the same flag is set), pixel q_1 is filtered; otherwise, pixel q_1 is not

filtered. *Id.*, 208. In other words, H.264 discloses asymmetric filtering similar to the '841 Patent.

88. If a decision is made to filter the p1 and/or q1 pixels, p1 and q1 are modified using the following offset equations, which are added to p1 or q1, respectively:

$$\Delta_{p1} = \text{Clip3}(-tC_0, tC_0, (p2 + ((p0 + q0 + 1) >> 1) - (p1 << 1)) >> 1))$$

$$\Delta_{q1} = \text{Clip3}(-tC_0, tC_0, (q2 + ((p0 + q0 + 1) >> 1) - (q1 << 1)) >> 1))$$

Id., 208. Like $\Delta_{p0,q0}$, the offset equations can also be rewritten in division form (subject to rounding and clipping):

$$\Delta_{p1} = \frac{2(p2) - 4(p1) + p0 + q0}{4}$$

$$\Delta_{q1} = \frac{2(q2) - 4(q1) + q0 + p0}{4}$$

89. Finally, if the bS value is equal to 4, then a different set of offset equations are applied for deblocking, which can modify up to three pixels on each side of the boundary. *Id.*, 208-09.

c. Draft H.265 (B205)

90. In 2010, ITU and MPEG formed another joint-venture called the “Joint Collaborative Team on Video Coding” (“JCT-VC”) which was tasked with “develop[ing] a new generation video coding standard that will further reduce by

50% the data rate needed for high quality video coding.” Ex-1042. This new standard would be called the “High Efficiency Video Coding (HEVC)” standard. *Id.*

91. Prior to the earliest-possible effective filing date of the ’841 Patent (as the ’841 Patent explains), the JCT-VC released a document numbered JCTVC-B205, titled “Test Model under Consideration.” Ex-1018, 1. The draft was intended to capture “a coding efficiency close to the best performing proposals” and “a complexity point that is close to that of the lowest complexity submissions shown to provide substantial improvement of coding efficiency” that had been submitted to date. *Id.*

92. The test model at the time included a deblocking filter that worked a bit differently than the one in H.264. *See id.*, 127-32. Specifically, rather than calculate an initial “boundary strength” value like H.264, B205 proposed a deblocking filter that would check one of a series of possible conditions *plus* a filter-decision equation to determine whether deblocking should be performed. The filter-decision equation is provided below:

$$|p_{2_2}-2*p_{1_2}+p_{0_2}| + |q_{2_2}-2*q_{1_2}+q_{0_2}| + |p_{2_5}-2*p_{1_5}+p_{0_5}| + |q_{2_5}-2*q_{1_5}+q_{0_5}| < \beta$$

Id., 130. In simpler terms, this equation took four second-order “difference-of-difference” measurements—two from each side of the block boundary using two different lines of pixels—summed them and compared them to a threshold value (β) that was based on the quantization parameters for the subject blocks.

p3 ₀ p2 ₀ p1 ₀ p0 ₀	q0 ₀ q1 ₀ q2 ₀ q3 ₀
p3 ₁ p2 ₁ p1 ₁ p0 ₁	q0 ₁ q1 ₁ q2 ₁ q3 ₁
p3 ₂ p2 ₂ p1 ₂ p0 ₂	q0 ₂ q1 ₂ q2 ₂ q3 ₂
A p3 ₃ p2 ₃ p1 ₃ p0 ₃	q0 ₃ q1 ₃ q2 ₃ q3 ₃ B
P3 ₄ p2 ₄ p1 ₄ p0 ₄	q0 ₄ q1 ₄ q2 ₄ q3 ₄
P3 ₅ p2 ₅ p1 ₅ p0 ₅	q0 ₅ q1 ₅ q2 ₅ q3 ₅
P3 ₆ p2 ₆ p1 ₆ p0 ₆	q0 ₆ q1 ₆ q2 ₆ q3 ₆
P3 ₇ p2 ₇ p1 ₇ p0 ₇	q0 ₇ q1 ₇ q2 ₇ q3 ₇

Id. (annotated). If the conditions outlined above were met, it was then determined whether to apply a “strong” or “weak” filter depending on an additional set of conditions.

93. If weak filtering was applied, two pixels on each side of the boundary would be modified (p0/q0 and p1/q1), and B205 provided the offset equation (Δ) to calculate the offset, and then applied the full offset to pixels p0 and q0, and half the offset to pixels p1 and q1:

$$\begin{aligned}\Delta &= \text{Clip}(-t_c, t_c, (13*(q0_i - p0_i) + 4*(q1_i - p1_i) - 5*(q2_i - p2_i) + 16) \gg 5)) \\ p0_i &= \text{Clip}_{0-255}(p0_i + \Delta) \\ q0_i &= \text{Clip}_{0-255}(q0_i - \Delta) \\ p1_i &= \text{Clip}_{0-255}(p1_i + \Delta/2) \\ q1_i &= \text{Clip}_{0-255}(q1_i - \Delta/2)\end{aligned}$$

Id., 131.

94. If strong filtering was applied, three pixels on each side of the boundary would be modified (p0/q0, p1/q1, and p2/q2), and B205 provided those equations as well (calculating new pixel values instead of offsets):

$$\begin{aligned}p0_0 &= \text{Clip}_{0-255}((p2_i + 2*p1_i + 2*p0_i + 2*q0_i + q1_i + 4) \gg 3) \\ q0_0 &= \text{Clip}_{0-255}((p1_i + 2*p0_i + 2*q0_i + 2*q1_i + q2_i + 4) \gg 3)\end{aligned}$$

$$\begin{aligned} p1_0 &= \text{Clip}_{0-255}((p2_i + p1_i + p0_i + q0_i + 2) \gg 2) \\ q1_0 &= \text{Clip}_{0-255}((p0_i + q0_i + q1_i + q2_i + 2) \gg 2) \\ p2_0 &= \text{Clip}_{0-255}((2 * p3_i + 3 * p2_i + p1_i + p0_i + q0_i + 4) \gg 3) \\ q2_0 &= \text{Clip}_{0-255}((p0_i + q0_i + q1_i + 3 * q2_i + 2 * q3_i + 4) \gg 3) \end{aligned}$$

Id.

VI. '841 PATENT

95. The '841 Patent was filed on August 26, 2013. The application was a national-stage entry of international patent application PCT/SE2011/051526, which was filed on December 16, 2011, and claims priority to a provisional U.S. patent application No. 61/477,862, filed on March 1, 2011.

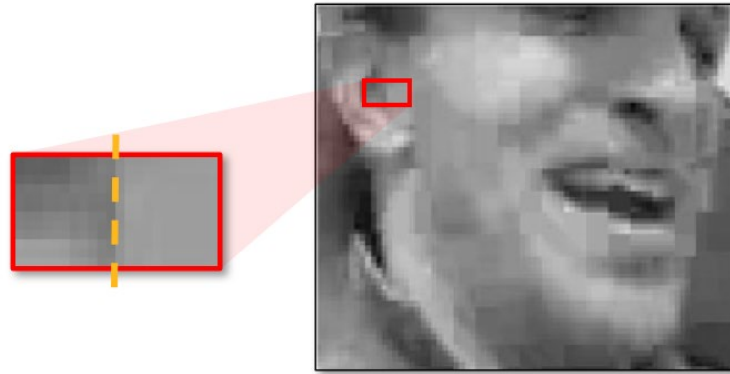
96. I therefore understand that, for purposes of comparing the '841 Patent to the prior art, the earliest-possible priority date is March 1, 2011.

97. I have reviewed the '841 Patent and provide the following summary.

A. Specification

98. The '841 Patent begins by providing an overview of the deblocking filter in both the H.264, as well as the proposed deblocking filter for the then-current draft of the HEVC standard, as outlined in B205. *See* Ex-1001, 1:19-3:5. However, according to the '841 Patent, “[d]eblocking filtering decisions according to HEVC can lead to inaccurate deblocking filtering over block boundaries for certain blocks,” and particularly those “having different levels of local structures” by over-filtering those “local structures.” *Id.*, 3:6-12. For instance, sometimes, a real detail in an

image may align with a block boundary, which could result in filtering of that detail, such as the example I have provided below:



If this boundary were extensively blurred, the crisp line between the outer ear and inner ear would be less apparent, resulting in a loss of visual quality.

99. To remedy this perceived problem, the '841 Patent proposes not only using an equation similar to that used in B205 (which uses pixels from two neighboring blocks) to decide *whether* to filter, but also splitting the equation for each block (each using pixels from just one block) to determine *how many* pixels to filter in each block. I have depicted the equations and what they are used for below:

$d = p2_i - 2p1_i + p0_i + q2_i - 2q1_i + q0_i $ $+ p2_j - 2p1_j + p0_j + q2_j - 2q1_j + q0_j < \beta$	If true, perform deblocking.
$d = p2_i - 2p1_i + p0_i < f(\beta)$	If deblocking, how many pixels to filter in block p .
$d = q2_i - 2q1_i + q0_i < f(\beta)$	If deblocking, how many pixels to filter in block q .

100. According to the '841 Patent, by also splitting the equation for each block, “[t]he asymmetric decisions means that the amount of filtering applied to one side of the block boundary can differ from the amount of filtering applied to the other side of the block boundary, thus providing additional adaptation to the local structure.” *Id.*, 4:62-5:2.

101. The '841 Patent explains that if the “p” or “q” decision values (called “ d_{pi} ” and “ d_{qi} ”, respectively) are less than their respective thresholds, two pixels will be modified on that side of the boundary. *Id.*, 9:35-59. But if they are greater than their respective thresholds, only one or no pixels on that side of the boundary will be modified. *Id.* The '841 Patent explains that “the closer the first or second filter decision value is to zero the more filtering is to be applied ... [because] a zero or low first or second filter decision value implies no or few structures but rather a fairly uniform area in the video frame.” *Id.*, 16:4-15. In other words, the '841 Patent proposes four different possible filtering configurations, outlined below, with the highlighted regions indicating which pixels may be modified, and dashed lines representing optional modifications (*e.g.*, either zero or one pixel may be modified).

$d_{pi} < \text{threshold 1}$ and $d_{qi} < \text{threshold 2}$

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃
p3 ₄	p2 ₄	p1 ₄	p0 ₄	q0 ₄	q1 ₄	q2 ₄	q3 ₄
p3 ₅	p2 ₅	p1 ₅	p0 ₅	q0 ₅	q1 ₅	q2 ₅	q3 ₅

$d_{pi} < \text{threshold 1}$ and $d_{qi} > \text{threshold 2}$

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃
p3 ₄	p2 ₄	p1 ₄	p0 ₄	q0 ₄	q1 ₄	q2 ₄	q3 ₄
p3 ₅	p2 ₅	p1 ₅	p0 ₅	q0 ₅	q1 ₅	q2 ₅	q3 ₅

$d_{pi} > \text{threshold 1}$ and $d_{qi} < \text{threshold 2}$

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃
p3 ₄	p2 ₄	p1 ₄	p0 ₄	q0 ₄	q1 ₄	q2 ₄	q3 ₄
p3 ₅	p2 ₅	p1 ₅	p0 ₅	q0 ₅	q1 ₅	q2 ₅	q3 ₅

$d_{pi} > \text{threshold 1}$ and $d_{qi} > \text{threshold 2}$

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃
p3 ₄	p2 ₄	p1 ₄	p0 ₄	q0 ₄	q1 ₄	q2 ₄	q3 ₄
p3 ₅	p2 ₅	p1 ₅	p0 ₅	q0 ₅	q1 ₅	q2 ₅	q3 ₅

102. But that is not the only change the '841 Patent proposes to the draft HEVC standard. The filtering decisions are decided for particular lines in a block. The '841 Patent recognizes that the filter decisions can be made in a number of ways, including: (1) by evaluating the pixels in each line separately, and making decisions on a line-by-line basis; or (2) using some subset of lines to make a representative decision for other lines (e.g., using pixels from line 0 to make a decision for lines 0–2, line 3 to make a decision for lines 3–5, etc.) or the entire block (e.g., using lines 2 and 5 to make a decision for lines 0–7, like the draft HEVC standard. *Compare* Ex-1001, 10:14-46, *with* B205, 130; *see also* Ex-1001, 8:53-9:25, 11:14-12:29. Furthermore, different “line-specific weights” could be assigned to each line of pixels as part of the filter decision. *Id.*, 12:46-58. The '841 Patent does not identify

any specific benefit to using certain lines of pixels instead of other lines, or explain exactly how any weights could be determined.

103. The '841 Patent explains that “[t]he actual filtering works by computing an offset (Δ), adding it to the original pixel value and clip the sum to a filtered output pixel value in the range of 0-255.” *Id.*, 2:44-46. The '841 Patent also provides offset equations—which, due to its nature, appears to correlate to the draft HEVC standard’s “weak” filter—that are different from those in the draft HEVC standard. Provided below is the offset equation used to calculate the offset for the p_0/q_0 pixels. The top line shows the bit-shift form of the equations provided by B205 and the '841 Patent, respectively, and the bottom line shows the equation rewritten in division form and normalized to have the same denominator. The division form would also be rounded to the nearest integer, for the reasons I discuss above. *See* Section V.B.3.b. The annotations in red show the differences between the two equations.

Draft HEVC Standard (B205) (<i>Id.</i> , 2:47-48)	'841 Patent (<i>Id.</i> , 17:5-6, 18:30-33)
$(13(q_0-p_0)+4(q_1-p_1)-5(q_2-p_2)+16)>>5$	$(9(q_0-p_0)-3(q_1-p_1)+8)>>4$
$\frac{7.5(q_0-p_0) + 2(q_1-p_1) - 2.5(q_2-p_2)}{16}$	$\frac{9(q_0-p_0) - 3(q_1-p_1)}{16}$

The '841 Patent similarly provides a different offset equation for the p_1/q_1 pixels, also shown below. For simplicity, I have provided the calculations for the modified p values, and not the associated q values. The offset equation for the p_1/q_1 pixels in

both the draft HEVC standard and '841 Patent incorporate the first offset calculated for the p_0/q_0 pixels, represented by “ Δ ”.

Draft HEVC Standard (B205) (<i>Id.</i> , 2:45-57)	'841 Patent (<i>Id.</i> , 17:31-56, 18:33-35)
$\Delta_{(p_0, q_0)} \ggg 1$	$((p_0 + p_2 + 1) \ggg 1) - p_1 + \Delta_{(p_0, q_0)} \ggg 1$
$\frac{\Delta}{2}$ (or $\frac{2\Delta}{4}$)	$\frac{(p_2 - 2p_1 + p_0) + 2\Delta}{4}$

The '841 Patent does not explain why these functions were chosen instead of those proposed by B205, but notes that “[d]ifferent such functions are possible and can be used” in place of those provided. *Id.*, 18:22-23.

B. Prosecution History

104. I have reviewed the '841 prosecution history and provide the following summary.

105. As I mentioned above, the '841 Patent is a national stage entry of an earlier-filed international application. *See* Ex-1002, 93. Before the U.S. submission, I understand that the original, internationally filed claims were subject to examination by an international search authority, which had found the claims not inventive. *Id.*, 8. Specifically, the international search authority had concluded that nearly all of the claim elements—which were directed to asymmetric filtering—were taught by U.S. Patent Application No. 2010/142844 (“Pereira”). *Id.*, 9-10. Although Pereira did not disclose the exact filter-decision equations $(p_2 - 2p_1 + p_0)$ and $(q_2 -$

2q1+q0), the international search authority found their disclosure in a JCT-VC proposal rendered the claims not inventive. *Id.* The Applicant disputed the non-inventiveness of the claims by arguing that “it is not disclosed in [Pereira] that the number of pixels of a line of pixels is determined from filter decision values based on the pixels in the same line of pixels,” and in the JCT-VC document “pixels on both side [sic] of the border are used, and the same number of pixels is filtered on both sides.” *Id.*, 14-17.

106. When the application was entered in the U.S., the Applicant submitted a series of preliminary claim amendments in what I understand was an attempt to bring them into U.S. claim standards. *Id.*, 267-314; *see also id.*, 94-107, 242-55. The Examiner, however, rejected two of the claims under § 101, and almost all of the claims in view of a combination of references (“Bjontegaard ’048” (Ex-1065) and “List” (Ex-1024)),³ but found one claim allowable—which ultimately became dependent claim 10 with the specific offset equations—if it were rewritten in independent form and other formalities addressed. *Id.*, 407-20.

107. In response, the Applicant did not amend the claims (except for a minor amendment to overcome the separate § 101 rejection, and cancellation of one claim). *Id.* 488-502. The Applicant then argued that the rejection should be withdrawn

³ This is a different “Bjontegaard” reference than the one I discuss in detail below.

because “the cited combination of references fails to disclose or suggest a technique that involves separately determining how many pixels to filter on [each] side of a boundary,” and that the cited art used pixels from both sides of the block boundary to decide *whether* to filter. *See id.*, 503-08.

108. The Examiner then allowed the claims. *Id.*, 516-19. The Examiner “adopt[ed] as his reasons for allowance the explanation in the submission filed by Applicant,” and specifically that “the decision to filter a determined number of pixels on one side of a block boundary has no mathematical basis in (no relationship with) the pixels on the other side of the block boundary,” as recited by the allowed claims. *Id.*, 518.

VII. PRIOR ART

A. Bjontegaard (and Bjontegaard’s Code) (Ex-1009-1015)

109. Bjontegaard (Ex-1009) is a working draft of a standards proposal submitted as part of the ITU Video Coding Experts Group (Question 15), Study Group 16, and was published on May 9, 2000. Ex-1082. Because Bjontegaard was published (May 9, 2000) more than one year before the earliest-possible priority date of the ’841 Patent (March 1, 2011), I understand Bjontegaard constitutes prior art

under at least pre-AIA §102(b).⁴ Bjontegaard was not cited during prosecution. Ex-1001, 1-2.

110. Moreover, alongside Bjontegaard's proposal disclosure, Bjontegaard provided a package of computer code in a series of computer files embodying the deblocking filter described in the proposal ("Bjontegaard's Code"). *See* Ex-1016 (Doc. Q15-J-27 stating "[t]he complete filter description is given in q15j28.doc"); Bjontegaard, 1 (labeled Q15-J-28); Ex-1010, 1 (printout of the computer code's file system with an introductory "readme.txt"); Ex-1015, 1 (readme.txt file stating "lcodec4.0 is an implementation of the Telenor H.26L proposal for the Osaka meeting.... The new proposed Telenor elements described in document Q15J27"); *see also* Ex-1011-1014 (printouts of individual files within the code). Because Bjontegaard's Code was also published at the same time as Bjontegaard's proposal (May 9, 2000), Ex-1082 (showing Q15-J-27, 28, and 29 all uploaded May 9, 2000), I understand the code also constitutes prior art under at least pre-AIA §102(b).

⁴ I have also reviewed the declaration of Dr. Cliff Reader, in which he explains that Bjontegaard and Bjontegaard's Code were publicly available and accessible before the earliest-possible priority date of the '841 Patent, Ex-1005, ¶¶51-70, which further supports my understanding that Bjontegaard and Bjontegaard's Code constitute prior art.

Moreover, because Bjontegaard's proposal and the associated code were published by the same contributor, at the same time, and expressly cross-reference each other, in my opinion, a POSITA would have read Bjontegaard in light of Bjontegaard's Code.

111. Bjontegaard was submitted as part of an early exploration to establish the next video-coding standard at the time, first under the title H.26L, which would ultimately become H.264. Ex-1038, 3 (noting H.26L was presumed to be a precursor to H.264 at the time).

112. Bjontegaard provides a proposal for a complete "coding model," which was presented "by mainly defining the decoder actions." Bjontegaard, 1, 4.

113. It is common for multimedia coding standards to be defined by the decoder's operations, rather than the encoder. That is because coding standards describe the syntax to ensure that an encoded bitstream complying with a standard will be decodable by any standard-compliant decoder, thus ensuring interoperability among standards-compliant devices. What this means is that vendors that design encoders can modify the operation of the encoder in some ways, as long as the output from the encoder conforms to the syntax specified for decoders. For certain elements in which the encoder and decoder are intended to perform identically—such as "in-loop" elements used for image prediction and reconstruction—however, descriptions of the decoder would directly implicate the encoder's operation as well.

See generally, Section V.A (explaining how motion vectors and residuals determined at the encoder must be reconstructed, preferably identically, at the decoder).

114. Bjontegaard discloses a deblocking filter that is “in loop”—and thus would be expected to be present in both the encoder and decoder—because Bjontegaard describes the deblocking filter in Section 4, dedicated to the “[t]ransform and inverse transform” process associated with “in-loop” prediction coding. *See* Bjontegaard, 16-20; Section V.A. This is confirmed by Bjontegaard’s Code, which includes the deblocking filter in the encoder file, *see* Ex-1014, 1 (encoder), 6-8, 10-12 (deblocking filter), and is titled “loopfilter_new,” further indicating the deblocking filter is “in-loop.”

115. In the proposal, as part of the decoder, Bjontegaard discloses the incorporation of a “deblocking filter process.” Bjontegaard, 18. Bjontegaard explains that, before any filtering is performed, a decision is made whether or not to perform deblocking at all. *Id.*, 18-19. To make that decision, a series of “Strength” variables are determined based on the properties of the blocks under consideration, including, *e.g.*, whether the block is “intra” coded, has “nonzero transform coefficients,” and the difference between motion vector components. *Id.*, 18-19.

116. Once it is decided to filter the boundary between two neighboring blocks, Bjontegaard describes the deblocking filter's operations with reference to an arrangement of pixels that span two neighboring blocks, using pixels a-h:



Id., 19 (“Assume that the pixels a b c d e f g h belong to two neighbouring blocks (1 and 2). There is a block boundary between pixels d and e.”).

117. Bjontegaard then performs a series of calculations that are subsequently used to determine various operations of the deblocking filter:

$$S0 = \mathbf{abs}(4(c-f)+12(e-d)) + \mathbf{abs}(9(c-d-e+f))$$

$$S0_1 = 12\mathbf{abs}(a-2b+c)$$

$$S1_1 = \mathbf{abs}(4(a-d)+12(c-b)) + \mathbf{abs}(9(a-b-c+d))$$

$$S0_2 = 12\mathbf{abs}(e-2f+g)$$

$$S1_2 = \mathbf{abs}(4(e-h)+12(g-f)) + \mathbf{abs}(9(e-f-g+h))$$

Bjontegaard, 19. Specifically, the variable “S0” represents a measure of activity *across* the block boundary, because it uses pixels from both blocks 1 and 2. The variables “S0₁” and “S1₁” then represent different measures of activity in block 1, with S0₁ being a discrete version of the second-order derivative (or difference-of-difference) of three pixels in block 1, and the variable “S1₁” represents a weighted measure of activity that includes an additional pixel in block 1. The same is true for

variables “S0₂” and “S1₂,” except these variables use pixels from block 2 instead of block 1.

118. After these five calculations are performed, Bjontegaard then uses these calculations to determine an overall measure of activity in each block *and* across the block boundary. Bjontegaard, 19. Specifically, Bjontegaard first determines the value of variable “Act_{overall}” equal to “3–i,” in which “i” is the smallest number such that $S0 < \text{Lim}(\text{qp}) * 2^i$. *Id.* As Bjontegaard explains, “Lim(qp)” is a “threshold value[] for the filtering process depending on qp,” otherwise known as the “quantization parameter.” *Id.* In fact, Bjontegaard provides a table to convert the quantization parameter (QP) to Lim(qp):

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Lim	7	8	9	10	11	12	14	15	17	20	22	25	28	31	35	39	44	49	55	62	69	78	88	98	110	124	139	156	175	197	221	248

Id. In other words, a POSITA would have understood that Act_{overall} is effectively an integer representation of the overall activity across the block boundary, and can take on a value between 0 and 3. *Id.* Moreover, because a lower value of S0 means less pixel activity across the boundary, which thereby means a lower value of “i” is required for the inequality $S0 < \text{Lim}(\text{qp}) * 2^i$ to be true, Act_{overall} will increase as activity decreases.

119. After the integer value for overall activity ($Act_{overall}$) is determined, Bjontegaard determines an integer value of the pixel activity in block 1 (Act_1) using the following conditions:

- Act_1 is initially set to 1
- If $S0_1 < Lim(qp)$, Act_1 is incremented by 1
- If $S1_1 < Lim(qp)$, Act_1 is incremented by 1

Bjontegaard, 19. In other words, Act_1 is incremented to take on a value between 1 and 3 depending on how much pixel activity is occurring in block 1. Because $S0_1$ and $S1_1$ will decrease with less activity, that will cause Act_1 to increase, meaning a higher value of “ Act_1 ” indicates that there is less activity in the block. But if $S0_1$ and $S1_1$ are large due to more activity in the block, Act_1 will not increment and will stay equal to 1.

120. Once the value of Act_1 has been determined, Bjontegaard explains that the “same procedure is made for Act_2 ” to determine the aggregate activity in block 2 (using $S0_2$ and $S1_2$). Bjontegaard, 19.

121. When each of $Act_{overall}$, Act_1 and Act_2 have been determined, a final comparison is made to determine variables Act_{11} and Act_{22} . Specifically, Bjontegaard explains that Act_{11} is set equal to the smaller of Act_1 and $Act_{overall}$, and the same for Act_{22} by comparing Act_2 to $Act_{overall}$. This, in effect, ensures that the amount of filtering that is subsequently applied to the pixels is determined based on

the overall activity across the boundary *or* the activity within each block, depending on which measure demonstrates more activity (because a smaller activity value indicates *more* activity in the region of interest).

122. Once the final activity levels (Act_{11} and Act_{22}) for each block have been determined, Bjontegaard describes a process to modify the values of pixels d and e—the two pixels closest to the block boundary. Bjontegaard, 20 (§4.5.5). Bjontegaard averages Act_{11} and Act_{22} to produce variable “ Act_{edge} ,” which is then used to determine which of two offset equations are applied to pixels d and e:

$$Act_{edge} = (Act_{11} + Act_{22})/2$$

$$\text{If } Act_{edge} \text{ equals 1 or 2:} \quad Diff = (3(c - f) - 8(d - e))/16$$

$$\text{If } Act_{edge} \text{ equals 3:} \quad Diff = (b + c - 8(d - e) - f - g)/16$$

$$d_{modified} = d + Diff \text{ (clipped to stay in the range 0-255)}$$

$$e_{modified} = e - Diff \text{ (clipped to stay in the range 0-255)}$$

Bjontegaard, 20. As shown here, the variable “Diff” represents how much the pixel value for pixels d and e should change. Once “Diff” is calculated (and clipped to a value whose absolute value does not exceed “ $Clip_{edge}$,” not shown), it is then added to pixel d and subtracted from pixel e so that their pixel values are closer to one another.

123. After filtering pixels d and e, Bjontegaard discloses that a decision is made whether to filter a second pixel (c) in block 1. Bjontegaard, 20 (§4.5.6.1).

Specifically, Bjontegaard initializes “Diff” or the “offset” to zero, meaning by default pixel c will not be modified. Only if Act_{11} equals 2 or 3 will an offset be calculated to modify pixel c :

If Act_{11} equals 2: $Diff = 5(b - 2c + d)/16$

If Act_{11} equals 3: $Diff = 3(a + b - 4c + d + e)/16$

$c_{modified} = c + Diff$ (clipped to stay in the range 0-255)

Bjontegaard, 20. Once “Diff” is calculated (and clipped to a value whose absolute value does not exceed “Clip(qp,Strength1),” not shown), it is then added to pixel c . By implication, that means if Act_{11} equals 0 or 1, pixel c will not be modified because the offset “Diff” will remain zero.

124. This same process is then repeated to decide whether to filter pixel f in block 2, but using Act_{22} and pixels from block 2. Bjontegaard, 20 (§4.5.6.2).

125. Bjontegaard’s Code then implements Bjontegaard—including Bjontegaard’s deblocking filter—in both a video encoder and decoder. *See, e.g.*, Ex-1012 (decoder); Ex-1014 (encoder). Bjontegaard’s Code uses different variables in the computer code than what appears in Bjontegaard, but they can be mapped to one another. For instance, the equations provided in Bjontegaard can be mapped to the equations provided in Bjontegaard’s Code as follows, with the image pointers “img->” removed from the Code and some equations restructured (but not in a way that alters the mathematics) for mapping and readability:

Bjontegaard (Bjontegaard, 19-20)	Bjontegaard's Code (Ex-1012, 10-11; Ex-1014, 10-12)
$S0 = \text{abs}(4(c-f)+12(e-d)) + \text{abs}(9(c-d-e+f))$	$i0=4*(li[2]-li[5])-12*(li[3]-li[4])$ $i1=9*(li[2]+li[5]-li[3]-li[4]);$ $iaa=\text{abs}(i0)+\text{abs}(i1)$ <i>thus, when combined:</i> $iaa=\text{abs}(4*(li[2]-li[5])+12*(li[4]-li[3])) + \text{abs}(9*(li[2]-li[3]-li[4]+li[5]))$
$S1_1 = \text{abs}(4(a-d)+12(c-b)) + \text{abs}(9(a-b-c+d))$	$i0=4*(li[0]-li[3])-12*(li[1]-li[2]);$ $i1=9*(li[0]+li[3]-li[1]-li[2]);$ $il4=\text{abs}(i0)+\text{abs}(i1);$ <i>thus, when combined:</i> $il4=\text{abs}(4*(li[0]-li[3])+12*(li[2]-li[1])) + \text{abs}(9*(li[0]-li[1]-li[2]+li[3]))$
$S0_1 = 12\text{abs}(a-2b+c)$	$il3=12*\text{abs}(li[1]-2*li[2]+li[3]);$
$S0_2 = 12\text{abs}(e-2f+g)$	$ir3=12*\text{abs}(li[4]-2*li[5]+li[6]);$
$S1_2 = \text{abs}(4(e-h)+12(g-f)) + \text{abs}(9(e-f-g+h))$	$i0=4*(li[4]-li[7])-12*(li[5]-li[6]);$ $i1=9*(li[4]+li[7]-li[5]-li[6]);$ $ir4=\text{abs}(i0)+\text{abs}(i1);$ <i>thus, when combined:</i> $ir4=\text{abs}(4*(li[4]-li[7])+12*(li[6]-li[5])) + \text{abs}(9*(li[4]-li[5]-li[6]+li[7]))$
$\text{Diff} = (3(c-f) - 8(d-e))/16$	$id=(3*(li[2]-li[5])-8*(li[3]-li[4]))/16;$
$\text{Diff} = (b+c-8(d-e)-f-g)/16$	$id=(li[1]+li[2]-8*(li[3]-li[4])-li[5]-li[6])/16;$
$\text{Diff} = 5(b-2c+d)/16$	$id=5*(li[1]-2*li[2]+li[3])/16;$
$\text{Diff} = 3(a+b-4c+d+e)/16$	$id=3*(li[0]+li[1]-4*li[2]+li[3]+li[4])/16;$
$\text{Diff} = 5(e-2f+g)/16$	$id=5*(li[4]-2*li[5]+li[6])/16;$
$\text{Diff} = 3(d+e-4f+g+h)/16$	$id=3*(li[3]+li[4]-4*li[5]+li[6]+li[7])/16;$

This mapping shows that the variable $li[n]$ in Bjontegaard's Code is mapped to pixels a-h in Bjontegaard as follows:

Bjontegaard	a	b	c	d	e	f	g	h
Bjontegaard's Code	$li[0]$	$li[1]$	$li[2]$	$li[3]$	$li[4]$	$li[5]$	$li[6]$	$li[7]$

This mapping, however, also reveals a typographical error in Bjontegaard's $S0_1$ equation (identified above in red), Bjontegaard, 19, because Bjontegaard's Code uses what should be pixels b, c, and d, rather than pixels a, b, and c, *see, e.g.*, Ex-1012, 10. I discuss this typographical error in more detail below.

126. Bjontegaard's Code also clarifies how second-pixels c and f are filtered compared to Bjontegaard. Specifically, in some equations, Bjontegaard uses the subscript "modified" to identify when a pixel has been filtered. *See* Bjontegaard, 20 (labeling filtered pixels c_{modified} , d_{modified} , e_{modified} , f_{modified}). However, when filtering pixels c and f, which is performed *after* pixels d and e have been modified, Bjontegaard does not use the "modified" subscript for pixels d and e in the filter equations for pixels c and f (*i.e.*, $5(b-2c+d_{\text{modified}})/16$ for pixel c; $5(e_{\text{modified}}-2f+g)/16$ for pixel f). Thus, Bjontegaard at least superficially suggests the original-pixel values for pixels d and e are used to filter pixels c and f, respectively. But, as I describe in more detail below, Bjontegaard's Code confirms that the *modified* values of pixels d and e (d_{modified} and e_{modified}) are used to filter pixels c and f, respectively, because Bjontegaard's Code redefines the values of $li[3]$ (pixel d) and $li[4]$ (pixel e)

before filtering pixels c and f. *See, e.g.*, Ex-1012, 11 (calculating the offset (Diff) for pixel d using equation “ $id=(3*(li[2]-li[5])-8*(li[3]-li[4]))/16$ ”, applying the offset to pixel d using equation “ $il1=li[3]+id$ ” ($d_{\text{modified}} = d + \text{Diff}$), and then redefining the value of pixel d using equation “ $li[3] = il1$ ” ($d = d_{\text{modified}}$) *before* using the redefined variable “ $li[3]$ ” in the filter equation for pixel c under heading “/* center left */”).

127. Thus, when Bjontegaard and Bjontegaard’s Code are read together, a complete and accurate picture of the operation of Bjontegaard’s deblocking filter is provided.

B. Aksu (Ex-1019)

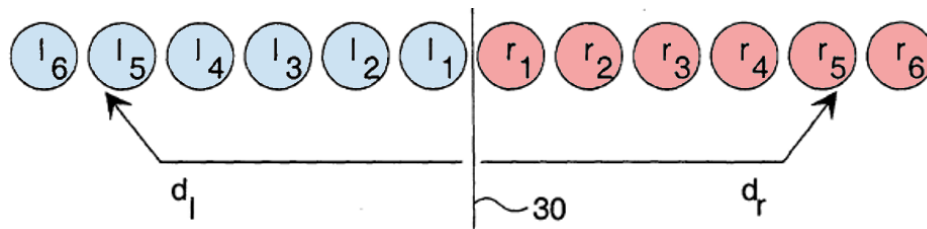
128. Aksu is an International Patent Application filed on January 22, 2001, and published on July 26, 2001. Because Aksu was published (July 26, 2001) more than one year before the earliest-possible priority date of the ’841 Patent (March 1, 2011), I understand that Aksu constitutes prior art under at least pre-AIA §102(b). Aksu was not cited during prosecution. Ex-1001, 1-2.

129. Aksu discloses a “method and associated device for filtering digital video images,” Aksu, 1:1, and particularly for “reducing blocking artefacts,” *id.*, 5:26-29.

130. Aksu acknowledges that deblocking filters had been proposed in the prior art before. Aksu, 5:1-14. However, Aksu recognized a problem with prior-art deblocking filters in that, on the one hand, they “tend to remove lines that belong to

real features of the image,” while on the other hand, sometimes “are not always capable of removing all blocking artefacts.” *Id.*, 5:22-24. Aksu therefore proposes a deblocking filter in which, according to Aksu, “only pixels containing blocking artefacts are selected for corrective filtering and that the quality of edges that are part of the image itself is not affected during filtering.” *Id.*, 7:10-14.

131. To accomplish this, Aksu first explains how the deblocking filter is intended to target lines of pixels from two neighboring blocks, which Aksu labels an “l” (“left”) block and an “r” (“right”) block:



Aksu, Fig. 2 (coloring added); *id.*, 9:4-6 (explaining “Figure 2 represents the location of pixels in relation to a block boundary”), 9:27-28 (same).

132. Aksu then discloses how, first, a decision must be made whether to perform deblocking at all. To make the decision, Aksu first determines how many pixels in each block to evaluate as part of the decision. *Id.*, 9:27-10:11. Aksu explains that a delta (Δ) value is calculated that represents “the difference of the pixel values [] across the block boundary.” *Id.* Aksu then uses delta (Δ) to determine a value for “n,” which is “the largest number of pixels to be examined from the block boundary in one direction,” according to the following chart:

$$n = \begin{cases} 0 & \Delta \geq 2.00\alpha \\ 1 & 1.50\alpha \leq \Delta < 2.00\alpha \\ 2 & 1.00\alpha \leq \Delta < 1.50\alpha \\ 3 & 0.66\alpha \leq \Delta < 1.00\alpha \\ 4 & 0.40\alpha \leq \Delta < 0.66\alpha \\ 5 & 0.25\alpha \leq \Delta < 0.40\alpha \\ 6 & 0 \leq \Delta < 0.25\alpha \end{cases}$$

Id. As Aksu explains, the value $\alpha = \text{QP} \cdot \log(\text{QP})$, in which “QP” is “the size of the quantization step [] of the coefficients received as the result of image coding.” *Id.* In other words, to determine whether (and how many) pixels to evaluate as part of the filter decision, Aksu compares the difference between pixel values on opposite sides of the block boundary to thresholds based on a “quantization parameter.” *Id.* As Aksu explains, “it is advantageous that [n’s] value is generally higher when the difference of the pixel values Δ across the block boundary is small in comparison to the size of the quantization step QP If the difference between the pixel values Δ is large, there is a high probability that there is a real image edge at the block boundary, and in this case the pixels are preferably not examined for filtering at all (n=0).” Aksu, 10:9-21.

133. Once the value of “n” has been determined (after considering a few additional conditions such as whether “n” should be “truncat[ed],” *see* Aksu, 10:23-12:4), the pixels identified for evaluation are processed to calculate a decision value

“d” for each block separately. Specifically, using the pixels from block “r” as exemplary, Aksu provides the following set of equations:

$$\begin{aligned} d_r &= 6, \text{ if } |r_1 - r_j| \leq \beta/j \text{ with all } j \in [1, 6], \\ \text{otherwise: } d_r &= i, \text{ where } i \text{ fulfils the conditions} \\ i &\in [1, n_{tr}], \quad (4) \\ |r_1 - r_{i+1}| &> \beta/i, \text{ and} \\ |r_1 - r_j| &\leq \beta/j \text{ with all } j \in [1, i]. \end{aligned}$$

Aksu at 13:15-19. This is an iterative calculation. Specifically, Aksu will take the pixel closest to the block boundary in the “r” block (“r₁”) and subtract adjacent pixels one-by-one: r₁−r₂, r₁−r₃, r₁−r₄, etc. This difference calculation will indicate how much pixels further from the boundary change compared to r₁. The change in pixel value is then compared to a threshold of β/j, in which “j” will match the position of the pixel being compared to r₁. Thus, Aksu’s equations can be written out as: (1) |r₁−r₂| ≤ β/2; (2) |r₁−r₃| ≤ β/3; (3) |r₁−r₄| ≤ β/4; and so on. If each inequality holds true, then the decision value for the “r” block (“d_r”) is set to 6. Otherwise, it is set to a lower value based on what number of pixels meet a similar inequality condition. Aksu, 13:15-19.

134. As Aksu explains, “the value[] of the parameter[] ... d_r ... represent[s] activity, or the differences of pixel values between pixels on one side of the block boundary.” Aksu, 13:10-13; *see also id.*, 14:1-5 (“the value of parameter d_r ... provides an indication of how many pixels counted from the block boundary have approximately the same value as the pixel at the block boundary.”). In other words,

the value of d_r indicates how many pixels from the block boundary are sufficiently similar, meaning there are likely no image details contained within them, and therefore, can be filtered without a loss of image-detail quality.

135. After calculating d_r , this same iterative process is then repeated for the “l” block, using the “l” pixel values to calculate a similar value (d_l). Aksu, 13:21-23. Aksu explains that “[i]t is advantageous that the values of the parameters d_r and d_l are calculated independent of one another, because the image information contained by the frame can be different on different sides of the block boundary.” *Id.*, 13:30-33. In other words, Aksu independently evaluates block “r” and block “l” to determine how much “activity” is occurring in each block near the boundary, but performs the evaluation separately for each block.

136. Once the values of d_r and d_l have been determined, those values are used to control how many pixels to filter on each side of the block boundary. Aksu, 15:17-16:4. Aksu explains that if the value of “d” is low for a given block, that indicates more activity in the block and thus fewer pixels will be filtered. *See* Aksu, 14:17-23; *see also id.*, 15:17-35 (showing how the value of “d” controls number of pixels filtered). If the value of “d” is high, that indicates less activity in the block and thus more pixels will be filtered. *See id.*

137. Aksu explains that “filtering is carried out using a suitable filter,” and that “[t]he invention does not limit the kind of filter that can be used[.]” Aksu, 15:1-

4. However, Aksu then discloses one example filter in which “a new pixel value is determined for a given pixel by calculating the mean of the pixel values that appear in the filtering window.” *Id.*, 15:4-8. Aksu then provides a chart to show how many pixels are filtered, and the size of the filtering window used to calculate the “mean” pixel value:

$d_r (d_l > 1)$	r_1	r_2	r_3
1	X	X	X
2	1	X	X
3	1	1*	X
4	2	2	X
5	2	2	2**
6	3 or 2***	3	3

Aksu, 15:17-35. This chart shows that if $d_r = 1$, no pixels are filtered (represented by “X”). If $d_r = 2$, then only pixel r_1 will be filtered by calculating a mean of r_1 , with one pixel on each side of r_1 (e.g., $(l_1 + r_1 + r_2)/3$). If $d_r = 3$, pixels r_1 and r_2 will be filtered, with each being calculated by taking the mean using one pixel on each side (e.g., $r_1' = (l_1 + r_1 + r_2)/3$; $r_2' = (r_1' + r_2 + r_3)/3$). It is also worth noting that Aksu states r_2 is calculated using the already-filtered version of r_1 (which I’ve labeled r_1'). *Id.*, 16:1. And, once again, after the requisite number of pixels in block “r” have been filtered, the same process is repeated for the pixels in block “l” depending on the value of d_l . *Id.*, 15:19-21.

C. Gonzalez (Ex-1020)

138. Gonzalez is a textbook titled “Digital Image Processing” which was authored by Rafael C. Gonzalez and Richard E. Woods, and published in 2008. Because Gonzalez was published (2008) more than one year before the earliest-possible priority date of the ’841 Patent (March 1, 2011), I understand Gonzalez constitutes prior art under at least pre-AIA §102(b).⁵ Gonzalez was not cited during prosecution. Ex-1001, 1-2.

139. Gonzalez is a general-reference textbook commonly used for undergraduate courses that describes many of the underlying concepts of image processing.

140. For instance, Gonzalez provides chapters on “Spatial Filtering” and “Filtering in the Frequency Domain” to modify pixel values by “accepting (passing) or rejecting certain frequency components” in the image. *See, e.g.*, Gonzalez, 104, 145, 199-200. For instance, Gonzalez describes how “linear filters” can take a series of pixel values as inputs, and then (among other things) average them equally or

⁵ I have also reviewed the declaration of June Munford, in which she explains that Gonzalez was publicly available and accessible before the earliest-possible priority date of the ’841 Patent, Ex-1007, ¶¶6-10, which supports my understanding that Gonzalez constitutes prior art.

weight pixels differently to calculate a new pixel value that eliminates unwanted image components, such as noise. *See generally* Gonzalez, 151-53.

141. In another chapter on “Image Segmentation,” Gonzalez explains how images can be analyzed not to output modified pixels, but for the “attributes extracted from those images,” such as identifying objects through edge detection. Gonzalez, 689. Gonzalez explains that “segmentation methods ... are based on detecting sharp, *local* changes in intensity.” *Id.*, 692. To detect these features, Gonzalez explains how “first- and second-order derivatives are particularly well suited for this purpose,” and that “[d]erivatives ... are defined in terms of differences.” *Id.*

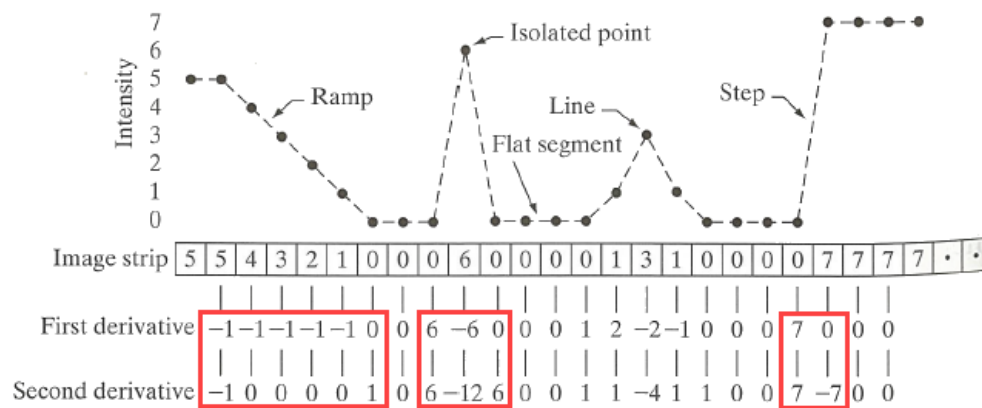
142. Gonzalez then provides a walkthrough of different algorithms that are used for image segmentation, beginning with the simplest one-dimensional “first derivative” that simply calculates the difference between two pixel values, Gonzalez, 693, as I discussed previously, *see* Section V.B.2 (overview of filters). Gonzalez then explains that the second-order derivative takes the *difference-of-differences*, producing an equation known as the Laplacian:

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x + 1) + f(x - 1) - 2f(x)$$

Gonzalez, 693; *see also id.*, 160 (noting this second-order derivative is called the “Laplacian”). When written as a one-dimensional mask (to apply to a set of three

pixels on a line), it would be written using the three-tap filter $[1, -2, 1]$, because those three coefficients would be applied to each pixel $[(x-1), (x), (x+1)]$, with “x” representing the current pixel location) within the mask. As Gonzalez explains, “[b]ecause the Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels.” *Id.*, 161.

143. Gonzalez then shows how a first-order difference calculation and a second-order Laplacian produce different outputs for purposes of image analysis:



Gonzalez, 694 (annotations added). Gonzalez explains that “first-order derivatives produce ‘thick’ edges and second-order derivatives much finer ones,” and that “a second-order derivative is much more aggressive than a first-order derivative in enhancing sharp changes. Thus, we can expect second-order derivatives to enhance fine detail (including noise) much more than first-order derivatives.” *Id.*, 694-95. Moreover, because the second-order derivative produces a characteristic “double-

edge” (also known as a “zero crossing,” Ex-1041, 149), this feature allows for more accurate identification of edges. Gonzalez, 695. For this reason, Gonzalez explains that the Laplacian is ideal for detecting “isolated points” in an image, lines, and edges. *See id.*, 695-97.

144. Moreover, Gonzalez explains how two, one-dimensional equations can be combined into a two-dimensional “mask” that assigns the following weights to a 3x3 set of pixel values taken as inputs:

0	1	0
1	-4	1
0	1	0

Id., 161.

D. H.241 (Ex-1017)

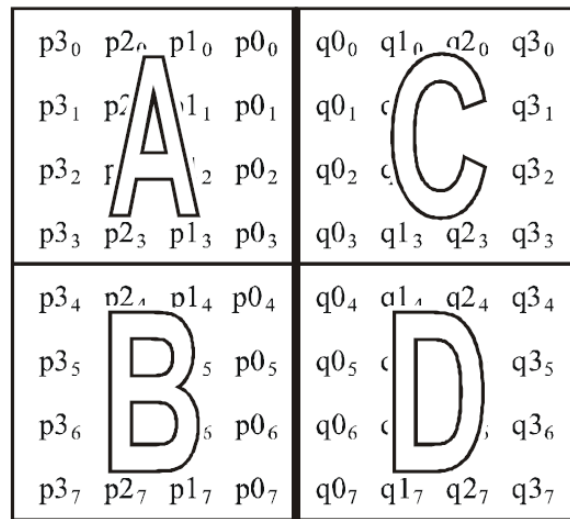
145. H.241 is an international telecommunications standard released by the ITU to “[e]xtend[] video procedures and control signals for H.300-series terminals,” dated May 2006, H.241, Cover, and published by at least December 2006, Ex-1077. Because H.241 was published (December 2006) more than a year before the earliest-possible priority date of the ’841 Patent (March 1, 2011), I understand that H.241

constitutes prior art under at least pre-AIA §102(b).⁶ H.241 was not cited during prosecution. Ex-1001, 1-2.

146. The H.241 standard specifies how to implement other ITU video-coding standards—such as H.264—in video-processing terminals. H.241, 1. As part of the standard, H.241 includes “a reduced-complexity decoding process to be applied to H.264 Baseline profile bitstreams when such use has been negotiated using ITU-T Rec. H.241.” H.241, 1. In particular, Annex B provides a deblocking filter for video images that is different (and relatively less complex) than the deblocking filter used in H.264. H.241, 27.

147. Specifically, H.241 discloses an arrangement of two adjacent blocks, a “p” block and a “q” block, each having 8x8 pixels, that can be broken into smaller 4x4 blocks:

⁶ I have also reviewed the declaration of Dr. Cliff Reader, in which he explains that H.241 was publicly available and accessible before the earliest-possible priority date of the '841 Patent, Ex-1005, ¶¶89-103, which supports my understanding that H.241 constitutes prior art.



a) Filtering across vertical edges

H.241, 28.

148. As part of the modified deblocking filter process, H.241 states that a value “d” must first be calculated using the following equation:

$$d = \left| \begin{array}{c} p2_2 - 2*p1_2 + p0_2 \\ q0_5 - 2*q1_5 + q2_5 \end{array} \right| + \left| \begin{array}{c} q0_2 - 2*q1_2 + q2_2 \\ p2_5 - 2*p1_5 + p0_5 \end{array} \right| +$$

Id., 29. Then, “[a] determination is made whether or not to apply filtering across a block edge.” *Id.*, 29. To make that determination, H.241 enumerates a series of conditions, in which if *any* “are true, no filtering across a block edge takes place.”

Id. One condition is if the value of $d \geq \beta$. *Id.*

149. H.241 explains that “ β ” is a function of an “index” as specified in H.264, and “[t]o obtain β , use index = indexB,” *id.*, 27, which is not defined in H.241 itself, but refers to the “indexB” calculated in H.264, *id.*, 27 (§ B.6.2); Ex-1036, 190-91.

Table B.2/H.241 – Value of t_c and β as a function of index

index	0-15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
t_c	0	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3
β	0	6	7	8	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28

To obtain t_c , use index = indexA. To obtain β , use index = indexB.

Table B.3/H.241 (concluded) – Value of t_c and β as a function of index

index	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
t_c	3	4	4	4	5	5	6	6	7	8	9	9	11	12	13	13	16	18
β	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64

To obtain t_c , use index = indexA. To obtain β , use index = indexB.

H.241, 28.

150. H.241 also describes other conditions that control whether or not any deblocking filtering is performed apart from the inequality relationship ($d \geq \beta$), including if a variable “filterBlockEdge” is set equal to zero. *Id.*, 29. As H.241 explains, the variable filterBlockEdge is set to zero unless one of the following conditions are true: (1) one or more of the blocks are “intra” coded; (2) one or more of the blocks have non-zero transform coefficients; (3) the absolute difference between the horizontal and vertical motion vector components exceeds a threshold, and other conditions, in which case the variable is set to 1. *Id.*

151. If deblocking filtering is to be performed based on one or more of the above conditions, H.241 then provides specific offset-filter equations to use to modify the pixel values:

$$\Delta = \text{Clip3}(-t_c, t_c, ((q0_i + ((p2_i + q1_i) \gg 1)) \gg 1) - ((p0_i + ((q2_i + p1_i) \gg 1)) \gg 1))$$

$$\begin{aligned}p0_i &= \text{Clip1}_Y(p0_i + \Delta) \\ q0_i &= \text{Clip1}_Y(q0_i - \Delta) \\ p1_i &= \text{Clip1}_Y(p1_i + \Delta/2) \\ q1_i &= \text{Clip1}_Y(q1_i - \Delta/2)\end{aligned}$$

H.241, 29.

E. B205/Applicant-Admitted Prior Art (Ex-1018)

152. B205 is a working-draft proposal for the High-Efficiency Video Coding standard, and was published on October 9, 2010. Ex-1094, 3. Because B205 was published (October 9, 2010) less than a year before the earliest-possible filing date of the '841 Patent (March 11, 2011), I understand that B205 constitutes prior art under at least pre-AIA §102(a).⁷ However, I also understand that the '841 Patent described B205 as Applicant-Admitted Prior Art. Ex-1001, 2:5-8.

153. As both B205 and the '841 Patent explain, B205 disclosed a deblocking filter that used the same filter-decision equation as H.241. B205, 130; Ex-1001, 2:5-27.

$$|p2_2 - 2*p1_2 + p0_2| + |q2_2 - 2*q1_2 + q0_2| + |p2_5 - 2*p1_5 + p0_5| + |q2_5 - 2*q1_5 + q0_5| < \beta$$

⁷ I have also reviewed the declaration of Dr. Cliff Reader, in which he explains that B205 was publicly available and accessible before the earliest-possible priority date of the '841 Patent, Ex-1005, ¶¶104-124, which supports my understanding that B205 constitutes prior art.

154. And like H.241, B205 discloses that one of the conditions to decide whether to perform deblocking is if a decision value (d) is less than a threshold (β). B205, 130; Ex-1001, 2:5-27.

155. However, unlike H.241, if a decision is made to perform deblocking filtering, B205 provides a different equation used to modify two pixels on each side of the block boundary:

$$\begin{aligned}\Delta &= \text{Clip}(-t_c, t_c, (13*(q_{0i} - p_{0i}) + 4*(q_{1i} - p_{1i}) - 5*(q_{2i} - p_{2i}) + 16) \gg 5)) \\ p_{0i} &= \text{Clip}_{0-255}(p_{0i} + \Delta) \\ q_{0i} &= \text{Clip}_{0-255}(q_{0i} - \Delta) \\ p_{1i} &= \text{Clip}_{0-255}(p_{1i} + \Delta/2) \\ q_{1i} &= \text{Clip}_{0-255}(q_{1i} - \Delta/2)\end{aligned}$$

B205, 131; Ex-1001, 2:44-49.

F. Fu (Ex-1022)

156. Fu is an International Patent Application filed on August 21, 2008 and published on March 19, 2009. Because Fu was published (March 19, 2009) more than one year before the earliest-possible priority date of the '841 Patent (March 11, 2011), I understand that Fu constitutes prior art under at least §102(b). Fu was not cited during prosecution. Ex-1001, 1-2.

157. Fu discloses a method and device for “reducing block distortion” using a filter. Fu, Cover, 1:15-38. After describing a high-level deblocking filter process

similar to those used throughout the prior art, Fu expressly provides a specific offset-filter equation for the pixels (p0, q0) closest to the block boundary:

$$\begin{aligned}\text{delta} &= \text{Clip3}(-C, C, (((q_0 - p_0) \times 3 + (p_1 - q_1) + 4) \gg 3)) \\ P_0 &= \text{Clip1}(p_0 + \text{delta}) \\ Q_0 &= \text{Clip1}(q_0 - \text{delta}),\end{aligned}$$

Fu, 8:5-9. Fu then separately provides specific offset-filter equations for the pixels (p1, q1) second closest to the block boundary on each respective side of the block boundary:

$$\begin{aligned}P_1 &= \text{Clip1}(p_1 + \text{Clip3}(-C, C, ((P_0 - p_1) \times 3 + (p_2 - Q_0) + 4) \gg 3)) \\ Q_1 &= \text{Clip1}(q_1 - \text{Clip3}(-C, C, ((q_1 - Q_0) \times 3 + (P_0 - q_2) + 4) \gg 3))\end{aligned}$$

Id.

G. B118 (Ex-1023)

158. B118 is a working draft of a standards proposal submitted for the Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, and was published on the ITU's website on March 15, 2002. Because B118 was published (March 15, 2002), more than one year before the earliest-possible priority date of the '841 Patent (March 1,

2011), I understand that B118 is prior art under pre-AIA §102(b).⁸ B118 was not cited during prosecution. Ex-1001, 1-2.

159. B118 discloses that “[a]fter the reconstruction of a macroblock a conditional filtering of this macroblock is taking place, that effects the boundaries of the 4x4 block structure. Filtering is done on a macroblock level.” B118, 46.

160. B118 defines “[t]he set of eight pixels across a 4x4 block horizontal or vertical boundary” as:

$$p_4, p_3, p_2, p_1 \mid q_1, q_2, q_3, q_4$$

“with the actual boundary between p_1 and q_1 .” *Id.*, 47. In other words, p_1 is the closest pixel to the block boundary in the p block of pixels, p_2 is the next closest pixel to the block boundary in the p block of pixels, etc. Similarly, q_1 is the closest pixel to the block boundary in the neighboring q block of pixels, p_2 is the next closest pixel to the block boundary in the q block of pixels, etc. *Id.*

161. B118 teaches that the filtering process depends on the boundary strength for the block boundary between two neighboring blocks of pixels. *Id.* B118

⁸ I have also reviewed the declaration of Dr. Cliff Reader, in which he explains that B118 was publicly available and accessible before the earliest-possible priority date of the '841 Patent, Ex-1005, ¶¶71-88, which supports my understanding that B118 constitutes prior art.

defines two types of filters: a default (weaker) filter; and a “stronger” filter. *Id.* The weaker filter is used for the pixels closest to the block boundary:

$$\begin{aligned}\Delta &= \text{Clip}(-C, C, ((q_1 - p_1) \ll 2 + (p_2 - q_2) + 4) \gg 3) \\ P_1 &= \text{Clip}(0, 255, (p_1 + \Delta)) \\ Q_1 &= \text{Clip}(0, 255, (q_1 - \Delta))\end{aligned}$$

B118, 48.

162. B118 then discloses that a decision is made whether to filter the pixels second closest from the block boundary (p2/q2) using “two intermediate threshold variables” “are used to decide whether p₂ and q₂ [i.e., p₁ and q₁] are filtered.”

$$\begin{aligned}a_p &= |p_3 - p_1| \\ a_q &= |q_3 - q_1|\end{aligned}$$

B118, 48. If $a_p < \beta$, pixel p₂ is filtered, and if $a_q < \beta$, pixel q₂ is filtered. *Id.* Thus, B118 discloses asymmetric filtering as well.

163. If the above conditions are met, B118 discloses that the following filter equations are used:

$$\begin{aligned}P_2 &= p_2 + \text{Clip}(-C0, C0, (p_3 + P_1 - p_2 \ll 1) \gg 1) \\ Q_2 &= q_2 + \text{Clip}(-C0, C0, (q_3 + Q_1 - q_2 \ll 1) \gg 1)\end{aligned}$$

Id.

VIII. LEVEL OF ORDINARY SKILL

164. I understand that when both interpreting the claims and assessing the prior art, I must do so from the perspective of a person of ordinary skill in the art

(“POSITA”) as of the relevant priority date. My understanding is that the earliest-possible priority date of the ’841 Patent is March 1, 2011.

165. In my opinion, at that time, a POSITA would have had a bachelor’s degree in electrical engineering, computer science, or a similar field and two to three years of experience in the design and development of relevant multimedia processing, such as video and/or image coding or compression. However, advanced graduate studies with coursework and/or research focusing on topics of video or image encoding/decoding and processing could substitute for working experience in the field.

166. I was at least a POSITA as of the earliest-possible priority date of the ’841 Patent, and I still am.

IX. CLAIM CONSTRUCTION

167. As I noted above, I understand that unless claim terms are provided an express construction, the terms must be given their plain and ordinary meaning.

168. I understand that Petitioner is not advancing any constructions at this time. I have therefore applied the plain and ordinary meaning of each term in the claims throughout this Declaration.

169. I also understand, however, that in separate proceedings involving the Patent Owner and another party (Lenovo), the parties dispute the constructions of several terms related to the ’841 Patent. I understand that the construction of one

term in particular may be of consequence here: whether “determining” how many pixels to filter in a given block may have any “mathematical basis” in the pixel values in a different block. I understand that Lenovo in that action is arguing that the claim should be construed such that the decision of how many pixels in one block to filter can have “no mathematical basis” in the pixel values of another block. Ex-1026, 6-7. I also understand that the Patent Owner has not offered a construction, but has instead argued that the “plain meaning” of the claim should apply, Ex-1026, 6-7, which does not preclude the determination having some mathematical relationship with pixels from another block, so long as a filter-decision value is calculated separately for each block using pixels from each respective block, Ex-1027, 16-19.

170. I do not offer an opinion on the correct construction, because in my opinion, every claim element was taught by the prior art under either of the proposed constructions, as I explain in further detail below.

X. GROUNDS

A. Ground 1A: Obviousness over Bjontegaard in view of Bjontegaard’s Code

171. If the number of pixels to be filtered can in some way depend on pixel values in a neighboring block, as I understand Patent Owner argues in the ITC claim

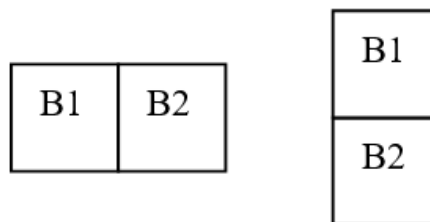
construction, then, in my opinion, Bjontegaard in view of other art (as discussed below) would teach or suggest all of the elements of the Challenged Claims.

1. Claim 1

- a. Element 1[pre]: A method for filtering control applicable to a block of multiple pixels in a video frame, each pixel having a respective pixel value, said method comprising:**

172. If the preamble is limiting, in my opinion, Bjontegaard discloses or suggests element 1[pre].

173. Bjontegaard discloses a process to decode video images. Bjontegaard, 1, 4 (noting use of well-known “CIF” and “QCIF” video frames, Ex-1040, 17-18). As part of that process, Bjontegaard discloses a method of “[f]iltering of pixels on both sides of a 4x4 block boundary,” Bjontegaard, 19, and provides the relationships of the blocks relative to one another:



Bjontegaard, 18. Bjontegaard explains that, within these blocks, “pixels [a-h] belong to two neighbouring blocks (1 and 2). There is a block boundary between pixels d [and] e.” Bjontegaard, 19. In other words, Bjontegaard describes a single row (or column) of pixels spanning these two blocks and arranged in the following way:

Block 1				Block 2			
a	b	c	d	e	f	g	h

Although a POSITA would have understood that these pixels are all assigned pixel values because that is how pixel information is conventionally stored in digital video systems, *see* Section V.B.1 (Pixel Values), it would have been apparent to a POSITA that Bjontegaard teaches multiple pixels in neighboring blocks that each have a respective pixel value because Bjontegaard provides the equations to operate on these pixel values. *Id.* (equations for $S0_n$ and $S1_n$ that incorporate pixels a-h). Accordingly, in my opinion, a POSITA would have understood that Bjontegaard describes a method for filtering blocks with pixels that each have pixel values that are then operated on by the filter.

- b. Element 1[a]: calculating a first filter decision value for said block, by only using pixels $p2_i$, $p1_i$, and $p0_i$, based on: $|p2_i - 2p1_i + p0_i|$ wherein $p0_i$ denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, $p1_i$ denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and $p2_i$ denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;**

174. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 1[a].

175. Specifically, Bjontegaard discloses calculating a first value: $S0_1 = 12\text{abs}(a - 2b + c)$. Bjontegaard, 19. A POSITA would have understood that this value “ $S0_1$ ” constitutes a “first filter decision value” because the value is used to make filter decisions, as I explained previously. *See* Section VII.A (Bjontegaard summary explaining how “ $S0_1$ ” is used to determine “ Act_1 ,” which is used to decide how many pixels to filter).

176. Moreover, a POSITA would have understood that the $S0_1$ equation is “based on” the equation “ $|p2_i - 2p1_i + p0_i|$,” as appears in the claims. First, a POSITA would have understood that the vertical bars represent the “absolute value” of a value, which is a measure of magnitude. In other words, if the calculation of the term between the bars produces a positive value (for instance, 3), the positive value is output (3). But if the calculation of the term between the bars produces a negative value (for instance, -3), the absolute value would output a positive value with the same magnitude (3). A POSITA would have further understood that Bjontegaard discloses the exact same “absolute value” using the “abs()” function. *See* Bjontegaard, 19 (“abs() is the absolute value function”). Bjontegaard also discloses that three pixel values (a, b, and c) are assigned the same coefficients as that of the claimed invention: $[1 \ -2 \ 1]$. Therefore, a POSITA would have understood that Bjontegaard’s equation is “based on” “ $|p2_i - 2p1_i + p0_i|$ ” by disclosing $12\text{abs}(a - 2b + c)$.

The fact that Bjontegaard multiplies the equation by “12” does not change my opinion because the claim uses the words “based on” and not “equals.”

177. In my opinion, a POSITA would have understood that Bjontegaard discloses using the three pixel values closest to the block boundary in the S_0 equation, even though Bjontegaard’s equation is written with respect to pixels a, b, and c that begin one pixel away from the block boundary:



178. A POSITA would have appreciated that this is a typographical error, and that the equation was intended to use the three pixels closest to the block boundary. A POSITA would have therefore read the equation as $S_0 = 12abs(b - 2c + d)$, even though it says $12abs(a - 2b + c)$. That is because, generally, a POSITA would have expected the deblocking filter to evaluate the pixel values closest to the block boundary, not pixel values beginning one pixel away from the block boundary. For instance, the prior art is filled with teachings that evaluate the pixels closest to the block boundary (with the closest pixels to the boundary identified below in red) to make filter decisions, without the one-pixel gap Bjontegaard appears to mistakenly disclose. *See, e.g.*, Ex-1037, 206-07 (H.264 evaluating $|p_0 - q_0|$, $|p_1 - p_0|$ and $|q_1 - q_0|$ to decide whether to filter (Eq. 8-468), and $|p_2 - p_0|$ and $|q_2 - p_0|$ to determine how many pixels to filter (Eqs. 8-471, 8-472)); B118, 47 (same); Fu, 16 (“obtain the

difference between the pair of pixels **R0** and **L0** on both sides of the block boundary ... and two pairs of pixels **L0** and **L1**, **R0** and **R1**”); H.241, 29 (evaluating **p0**, **p1**, and **p2**, and **q0**, **q1**, and **q2** to determine whether to filter); B205, 130 (same); Aksu, 13:15-19 (all difference calculations involving **r₁** and **l₁** closest to the block boundary).

179. This understanding is confirmed by Bjontegaard’s description of the equations for block 2, which has counterpart equations to those provided for block 1. For instance, in block 2, Bjontegaard discloses that S_{0_2} —the block 2 counterpart to S_{0_1} —uses the three pixels closest to the block boundary, $S_{0_2}=12\text{abs}(\text{e}-2\text{f}+\text{g})$, Bjontegaard, 19, as shown below:



S_{0_2} is therefore consistent with what a POSITA would have expected, unlike S_{0_1} .

180. Relatedly, when Bjontegaard subsequently calculates an offset to modify pixel “f” in block 2, Bjontegaard again uses the three pixels closest to the block boundary— $\text{Diff}_{[2]}=5(\text{e}-2\text{f}+\text{g})/16$ —which centers on pixel f. Bjontegaard, 20.⁹

⁹ I have added subscripts to the different “Diff” variables to indicate which block they are related to, because Bjontegaard uses the same “Diff” label for different blocks.

Consistently, when the same equation is used to modify pixel c in block 1, Bjontegaard uses the three pixels closest to the block boundary— $\text{Diff}_{[1]}=5(\text{b}-2\text{c}+\text{d})$ —centered on pixel c. Bjontegaard, 20. In block 2, therefore $S0_2$ and $\text{Diff}_{[2]}$ use the same three pixels (e, f, and g), but in block 1, $S0_1$ and $\text{Diff}_{[1]}$ use different sets of pixels (a, b, c versus b, c, d). A POSITA would have appreciated, however, that $S0_1$ and $\text{Diff}_{[1]}$ should be using the same pixels (like $S0_2$ and $\text{Diff}_{[2]}$) because $S0_1$ is used to *decide* whether to filter pixel c.

181. Finally, the typographical error is confirmed by Bjontegaard’s Code, which, as I mentioned previously, *see* Section VII.A (Bjontegaard summary), would have been understood by a POSITA as being read in conjunction with Bjontegaard. Specifically, in Bjontegaard’s Code, $S0_1$ is calculated using the following equation (omitting the image pointers “img->” for each pixel): $\text{il3}=12*\text{abs}(\text{li}[1]+\text{li}[3]-2*\text{li}[2])$. Separately, $S0_2$ is calculated using the following equation (again omitting the pointers): $\text{ir3}=12*\text{abs}(\text{li}[4]+\text{li}[6]-2*\text{li}[5])$. Ex-1012, 10. This is a contiguous set of pixels $\text{li}[1]–\text{li}[6]$. I have reproduced the mapping I provide above, Section VII.A (Bjontegaard summary), between Bjontegaard’s pixels (a–h) and the code ($\text{li}[0]–[7]$) below.

Bjontegaard	a	b	c	d	e	f	g	h
Bjontegaard’s Code	$\text{li}[0]$	$\text{li}[1]$	$\text{li}[2]$	$\text{li}[3]$	$\text{li}[4]$	$\text{li}[5]$	$\text{li}[6]$	$\text{li}[7]$

182. There is no gap between the $S0_1$ and $S0_2$ equations that would be expected if pixels a, b, and c were used instead of pixels b, c, and d. Moreover, Bjontegaard's Code consistently maps $li[0]$ to pixel a, $li[1]$ to pixel b, $li[2]$ to pixel c, and $li[3]$ to pixel d, throughout all of the equations *except* the $S0_1$ equation. See Section VII.A (Bjontegaard summary, mapping the relevant equations between Bjontegaard and Bjontegaard's Code). This would have confirmed to a POSITA that Bjontegaard should read: $S0_1=12abs(b-2c+d)$.

183. In sum, a POSITA would have understood that Bjontegaard discloses calculating a filter-decision value " $S0_1$ " using the three pixels closest to the block boundary, which matches the location of pixels $p2_i$, $p1_i$, and $p0_i$ in the Challenged Claims.

- c. **Element 1[b]: calculating a second filter decision value for said neighboring block, by only using pixels $q2_i$, $q1_i$, and $q0_i$, based on: $|q2_i-2q1_i+q0_i|$ wherein $q0_i$ denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, $q1_i$ denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and $q2_i$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;**

184. In my opinion, Bjontegaard discloses or suggests element 1[b].

185. Specifically, Bjontegaard discloses calculating $S0_2=12abs(e-2f+g)$.
Bjontegaard, 19. A POSITA would have understood that this value " $S0_2$ " constitutes

a “second filter decision value” because the value is then used to make filter decisions, as I explained previously. *See* Section VII.A (Bjontegaard summary, explaining how “S0₂” is used to determine “Act₂,” which is used to decide how many pixels to filter). Bjontegaard also discloses that three pixel values (e, f, and g) are assigned the same coefficients as that of the claimed invention: [1 -2 1]. And Bjontegaard discloses that these three pixels are the ones closest to the block boundary in a neighboring block, just like pixels q2_i, q1_i, and q0_i.



- d. Element 1[c]: determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and**

186. In my opinion, Bjontegaard discloses or suggests element 1[c].

187. Specifically, Bjontegaard discloses that if Act_{overall}=0, no pixels will be filtered. Bjontegaard further discloses that if Act_{overall}=1, 2, or 3 (true at least for all blocks whose strength is 3), at least pixel d in block 1—the pixel closest to the block boundary in block 1—is filtered. Bjontegaard, 20 (§4.5.5). Afterwards, a separate decision is made whether to also filter an additional pixel c located one pixel away from the block boundary in block 1. Bjontegaard, 20 (§4.5.6.1). Bjontegaard, therefore, discloses deciding whether to filter zero, one, or two pixels in block 1.

188. Pixel c in block 1 is filtered only if Act_{11} equals 2 or 3, which in turn depends on a comparison of $S0_1$ to a threshold value $Lim(qp)$, as explained further below:

4.5.6.1 Modification if c in block 1

Initially set $Diff = 0$

If Act_{11} equals 2: $Diff = 5(b - 2c + d)/16$

If Act_{11} equals 3: $Diff = 3(a + b - 4c + d + e)/16$

$Diff = \max(-Clip(qp, Strength_1), \min(Clip(qp, Strength_1), Diff))$

$c_{modified} = c + Diff$ (clipped to stay in the range 0-255)

Bjontegaard, 20. According to Bjontegaard, pixel c is not filtered by default, because the value of “Diff” (the offset; $c_{modified}=c+Diff$) is “initially set” to zero each time, and can only become non-zero to modify pixel c if Act_{11} equals 2 or 3. Bjontegaard, 20 (§4.5.6.1).

189. Bjontegaard explains that the value of Act_{11} is determined by evaluating several conditions. Bjontegaard, 19-20 (§4.5.4). To calculate Act_{11} , Bjontegaard explains that a preliminary variable “ Act_1 ” is first set equal to 1. Bjontegaard, 19. Then, two conditions are checked:

- If $S0_1 < Lim(qp)$, Act_1 is incremented by 1
- If $S1_1 < Lim(qp)$, Act_1 is incremented by 1

Bjontegaard, 19. For each condition that is met, the value of Act_1 will be incremented by 1. And as Bjontegaard explains, one of the conditions is whether the value of $S0_1$

is less than a first-threshold value ($\text{Lim}(\text{qp})$). Bjontegaard, 19. As a result, Act_1 will always be greater than 1 if either condition is met.

190. After the two conditions above are checked and the final value of Act_1 is determined (*e.g.*, Act_1 takes on a value between 1 and 3), Bjontegaard provides a final check:

$$\text{Act}_{11} = \min(\text{Act}_{\text{overall}}, \text{Act}_1),$$

Bjontegaard, 19. This “min()” function means that Act_{11} is set equal to the smaller of Act_1 and $\text{Act}_{\text{overall}}$. Bjontegaard, 19. Any time Act_1 is less than or equal to $\text{Act}_{\text{overall}}$, Act_{11} would therefore equal Act_1 . Bjontegaard, 19.

191. Any time Act_1 is less than or equal to $\text{Act}_{\text{overall}}$, the value of Act_1 alone will determine whether a second pixel will be filtered if *either* of the two conditions identified above is met. Therefore, when Act_1 is less than or equal to $\text{Act}_{\text{overall}}$, if $S0_1$ is less than the threshold $\text{Lim}(\text{qp})$, Act_1 will always be greater than 1, and a second pixel *c* will be filtered; but if $S0_1$ is greater than or equal to $\text{Lim}(\text{qp})$ —and the second condition is not met as well—then Act_1 will equal 1, Act_{11} will not be greater than 1, and only one pixel *d* will be filtered. Bjontegaard, 19. Thus, the number of pixels that will be filtered in block 1 is determined based on the value of Act_{11} , which in turn is determined based on a comparison of $S0_1$ to a threshold value $\text{Lim}(\text{qp})$.

192. In my opinion, a POSITA would have understood that because Act_{11} is set to be the value of Act_1 , any time Act_1 is less than or equal to $\text{Act}_{\text{overall}}$, and the

value of Act_1 depends on comparing the first filter decision value $S0_1$ to a threshold $Lim(qp)$, Bjontegaard discloses that the number of pixels to filter in block 1 (either zero, one, or two pixels) is determined based on a comparison of the first filter decision value $S0_1$ to a first threshold $Lim(qp)$. Bjontegaard, 19-20.

- e. **Element 1[d] determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.**

193. In my opinion, Bjontegaard discloses or suggests element 1[d].

194. Bjontegaard explains that after a decision is made whether to filter pixel c in block 1, the same procedure is followed to determine whether to filter a second pixel f in block 2. *See* Bjontegaard, 19 (§4.5.4 stating “[t]he same procedure” of incrementing Act_1 based on $S0_1$ and $S1_1$ “is made for Act_2 ”), 20 (§4.5.6.2 providing the same filter conditions for pixel f as pixel c, except using variable “ Act_{22} ”). A POSITA therefore would have understood that Act_{22} is calculated in the same way as Act_{11} , except that variables $S0_2$ and $S1_2$ are used to increment Act_2 in block 2 instead of $S0_1$ and $S1_1$ used for Act_1 in block 1.

195. Because Act_{22} is calculated in the same way as Act_{11} , except using the variables associated with block 2 ($S0_2$ and $S1_2$), for the same reasons I discussed above, *see* Section X.A.1.d (element 1[c]), Bjontegaard discloses making a decision as to how many pixels to filter (zero, one, or two pixels) based on a comparison of

the second filter-decision value $S0_2$ to a second-threshold $\text{Lim}(qp)$. Bjontegaard, 19-20.

196. A POSITA would have been motivated to modify Bjontegaard to use a different threshold for each block, and thus Bjontegaard renders this element obvious even if the claim requires using a *different* threshold for each of the two blocks. Specifically, a POSITA would have known that each block has its own quantization parameter (“qp”) value representing how much that block was quantized (or compressed). Bjontegaard, 19 (“As pixels a to h belong to two blocks, qp of the two blocks may differ.”); *see also* Aksu, 10:9-12 (noting “QP [may have] a different value in blocks on different sides of the block boundary....”). Bjontegaard, however, teaches using a single qp value corresponding to the maximum qp value of the two blocks. Bjontegaard, 19 (“In this context qp means the maximum qp for the two blocks.”).

197. Because Bjontegaard measures activity in each block separately, using pixels from each respective block ($S0_1$ uses pixels b-d, and $S0_2$ uses pixels e-g), a POSITA would have been motivated to use the qp value associated with each block to calculate thresholds for block 1 and block 2, respectively (effectively threshold 1 (block 1) = $\text{Lim}(qp_1)$ and threshold 2 (block 2) = $\text{Lim}(qp_2)$). A POSITA would have appreciated that using the qp assigned to each block would be more representative

of the quantization of that block, and thus would be a better threshold to determine whether the activity in those pixels exceeds that threshold.

198. In particular, using a block-specific qp value for each block would ensure that, when the qp values for the two blocks differ, the threshold used to make filter decisions would not be changed based on the qp value of another block. With Bjontegaard, for instance, if the qp value of block 1 (qp_1) is smaller than the qp value of block 2 (qp_2), that means the threshold used to control how many pixels are filtered in block 1 is controlled by a larger qp_2 value (because Bjontegaard discloses using the maximum qp of the two blocks, and Bjontegaard provides a table showing the relation between $Lim(qp)$ and qp, Bjontegaard, 19). If (the smaller) qp_1 were instead used to determine the threshold $Lim(qp_1)$, the two conditions ($S0_1 < Lim(qp_1)$ and $S1_1 < Lim(qp_1)$) are less likely to be met, the value of Act_1 (and consequently of Act_{11}) will be the same or lower, and therefore the likelihood of a determination that the second pixel c will be filtered is lower. A POSITA would have known that such an approach is beneficial because a block that undergoes less-severe quantization (compression) during encoding would be expected to require less deblocking filtering because it better reflects the pixel values in the original image (*e.g.*, it is less likely to have a blocking artifact). Bjontegaard's selection of a threshold based on the *maximum* qp of the two blocks, therefore, may result in *over*-filtering of less-quantized blocks.

199. In my opinion, Bjontegaard discloses selecting the “maximum qp” of the two blocks as a basis for threshold $\text{Lim}(\text{qp})$ for efficiency. In fact, Aksu discloses making the opposite selection as Bjontegaard: that when selecting a “QP” value to use as a threshold, “if QP has a different value in blocks on different sides of the block boundary, the smaller value of QP is used in the calculations.” Aksu, 10:9-12. Similar to Bjontegaard, a POSITA would have known that Aksu’s selection of a threshold based on the *minimum* qp of the two blocks may result in *under*-filtering of more-quantized blocks, because it is less likely the threshold would be met and, therefore, less filtering would be performed in blocks with larger qp values.

200. Although both Bjontegaard and Aksu select a single qp value to serve as the basis of the threshold for simplicity, their different approaches simply reflect a design decision as to how likely the threshold will be met, and therefore how often the functions which depend on the comparisons to the threshold are triggered as a result of the lower or higher qp value used. The more accurate, if *slightly* more computationally taxing choice would be to use the qp value of each block when a threshold is used to independently evaluate conditions in each block.

201. In fact, others in the art (such as Kwon) had used the qp specific to each block when comparing their activity measures to a threshold. *See, e.g.*, Ex-1064, Fig. 17 (comparing “Diff” values for block 1 to QP1 and “Diff” values for block 2 to QP2, separately), ¶¶ 95, 96. In sum, a POSITA would have been motivated to use a

threshold ($\text{Lim}(qp_1)$) for block 1 that is determined based only on the QP of block 1, and a threshold ($\text{Lim}(qp_2)$) for block 2 that is determined based only on the QP of block 2.

202. A POSITA would have had a reasonable expectation of success in making this modification because the qp values for both blocks would have already been available for use (as they are determined during encoding), *see* Section V.A (Video Encoding and Decoding summary, discussing quantization process), and thus this would have involved nothing more than minor modification of computer code to ensure the block 1 calculations were compared to a different $\text{Lim}(qp)$ than the block 2 calculations. A POSITA would have also understood that using block-specific qp values as thresholds works. Ex-1064, Fig. 17, ¶¶ 95, 96.

2. Claim 3

- a. Element 3[a]: The method according to claim 1, wherein determining how many pixels in said first line of pixels in said block to filter comprises:**

203. For the reasons provided in more detail below, in my opinion, Bjontegaard discloses or suggests claim 3. And as I discussed previously, Bjontegaard discloses the “determining” step of claim 1. *See* Section X.A.1.d (element 1[c]).

b. Element 3[b]: comparing said first filter decision value to a first threshold value;

204. In my opinion, Bjontegaard discloses or suggests comparing a first filter decision value ($S0_1$) to a first threshold ($\text{Lim}(qp)$ or $\text{Lim}(qp_1)$). *See* Sections X.A.1.d (element 1[c]), X.A.1.e (element 1[d], discussing first threshold).

c. Element 3[c]: determining to filter two pixels in said first line of pixels in said block relative to said block boundary if said first filter decision value is below said first threshold value; and

205. In my opinion, Bjontegaard discloses or suggests making a determination to filter two pixels (c, d) in block 1 if the first filter decision value ($S0_1$) is below a first threshold ($\text{Lim}(qp)$ or $\text{Lim}(qp_1)$), because if that condition is met, that will increment Act_1 and thus cause pixel c to be filtered as long as Act_1 is also less than or equal to $\text{Act}_{\text{overall}}$. *See* Section X.A.1.d (element 1[c]).

d. Element 3[d]: determining to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value; and

206. In my opinion, Bjontegaard discloses or suggests making a determination to filter one pixel (d) in block 1 if the first filter decision value ($S0_1$) is equal to or exceeds a first threshold ($\text{Lim}(qp)$) or $\text{Lim}(qp_1)$, because in such circumstances Act_1 will not be incremented as long as $S1_1$ is also less than the first threshold. *See* Section X.A.1.d (element 1[c]). Bjontegaard therefore discloses a circumstance in which $S0_1$ alone controls whether a second pixel (c) is filtered.

- e. Element 3[e]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises:**

207. For the reasons provided in more detail below, in my opinion, Bjontegaard discloses or suggests element 3[e]. And as I discussed previously, Bjontegaard discloses the “determining” step of claim 1. *See* Section X.A.1.e (element 1[d]).

- f. Element 3[f]: comparing said second filter decision value to a second threshold value;**

208. In my opinion, Bjontegaard discloses or suggests comparing a second filter decision value ($S0_2$) to a second threshold ($\text{Lim}(qp)$ or $\text{Lim}(qp_2)$). *See* Section X.A.1.e (element 1[d]).

- g. Element 3[g]: determining to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value; and**

209. In my opinion, Bjontegaard discloses or suggests making a determination to filter two pixels (e, f) in block 2 if the second filter decision value ($S0_2$) is below a second threshold ($\text{Lim}(qp)$ or $\text{Lim}(qp_2)$) because that condition will increment Act_2 and thus cause pixel f to be filtered as long as Act_2 is also less than or equal to $\text{Act}_{\text{overall}}$. *See* Sections X.A.1.d (element 1[c]), X.A.1.e (element 1[d]).

- h. Element 3[h]: determining to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second**

filter decision value is equal to or exceeds said second threshold value.

210. In my opinion, Bjontegaard discloses or suggests making a determination to filter one pixel (e) if the second filter decision value ($S0_2$) is equal to or exceeds a second threshold ($\text{Lim}(\text{qp})$ or $\text{Lim}(\text{qp}_2)$) because in such circumstances Act_2 will not be incremented as long as $S1_2$ is also less than the second threshold. *See* Sections X.A.1.d (element 1[c]), X.A.1.e (element 1[d]). Bjontegaard therefore discloses a circumstance in which $S0_2$ alone controls whether a second pixel (f) is filtered.

3. Claim 5

a. Element 5[pre]: A filtering control device comprising:

211. If the preamble is limiting, in my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[pre].

212. Specifically, Bjontegaard discloses a "coding model" for video images. Bjontegaard, 4.

213. A POSITA would have also understood that Bjontegaard's Code was intended to operate the coding model on a computer because the code included executable computer files (.exe) as well as the underlying source code files (.c) to operate on a computer. *See, e.g.,* Ex-1010. Bjontegaard's Code also includes instructions for how to operate the code on a computer. *See* Ex-1015, 1 (describing

usage of encoder and decoder by running “DOS command prompt[s]”). A POSITA would have, therefore, understood that Bjontegaard would be run on a “device.”

214. Moreover, video codecs with deblocking filters are conventionally run on devices such as computers and other digital processing devices (*e.g.*, smartphones, physical-media players, etc.). *See, e.g.*, Ex-1040, 1-5 (describing importance of video codecs, and particularly H.264, to various consumer electronics); Ex-1061, 9:49-57 (describing “[a]n electronic device for deblocking and edge enhancement”), 10:9-13 (“Those of ordinary skill would further appreciate that the various illustrative logical blocks, modules, and algorithm steps ... may be implemented as electronic hardware, computer software, or combinations of both.”); Ex-1062, 18:17-18 (“a deblocking filter is utilized for loop filtering”), 22:23-24 (providing “a computer system with which some embodiments of the invention is implemented.”).

215. For this reason, even if Bjontegaard does not expressly disclose a “filtering control device,” a POSITA would have known that Bjontegaard was intended to be implemented in a device (*e.g.*, a computer), and would have been motivated to implement Bjontegaard accordingly.

- b. Element 5[a]: a first decision value calculator configured to calculate a first filter decision value for a block of multiple pixels in a video frame, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $|p_{2i}-2p_{1i}+p_{0i}|$ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video**

frame, p_{li} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;

216. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[a]. *See* Section X.A.1.b (element 1[a]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies calculations that would require a processor serving as a calculator, Ex-1012, 10 (providing code to calculate (without pointers) first filter-decision value $il3=12*\text{abs}(li[1]+li[3]-2*li[2])$); Ex-1001, 23:13-21 (noting the calculator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "a first decision value calculator" to perform the $S0_1$ calculation Bjontegaard discloses. *See* Section X.A.1.b (element 1[a]).

- c. Element 5[b]: a second decision value calculator configured to calculate a second filter decision value for a neighboring block, by only using pixels q_{2i} , q_{li} , and q_{0i} , based on: $|q_{2i}-2q_{1i}+q_{0i}|$ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{li} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel**

in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;

217. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[b]. *See* Section X.A.1.c (element 1[b]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies calculations that would require a processor serving as a calculator, Ex-1012, 10 (providing code to calculate (without pointers) second filter-decision value $ir3=12*\text{abs}(li[4]+li[6]-2*li[5])$); Ex-1001, 23:13-21 (noting the calculator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "a second decision value calculator" to perform the $S0_2$ calculation Bjontegaard discloses. *See* Section X.A.1.c (element 1[b]).

- d. **Element 5[c]: a first pixel determiner configured to determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and**

218. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[c]. *See* Section X.A.1.d (element 1[c]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]),

and Bjontegaard's Code expressly specifies determinations that would require a processor serving as a determiner, Ex-1012, 11 (providing code to determine "if(idl==2)" or "if(idl==3)," *e.g.*, if Act₁₁ equals 2 or 3, to determine whether to filter pixel c); Ex-1001, 23:13-21 (noting the determiner can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "a first pixel determiner" to determine how many pixels to filter in block 1, as Bjontegaard discloses. *See* Section X.A.1.d (element 1[c]).

- e. **Element 5[d]: a second pixel determiner configured to determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator, wherein**

219. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[d]. *See* Section X.A.1.e (element 1[d]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies determinations that would require a processor serving as a determiner, Ex-1012, 11 (providing code to determine "if(idr==2)" or "if(idr==3)," *e.g.*, if Act₂₂ equals 2 or 3, to determine whether to filter

pixel f); Ex-1001, 23:13-21 (noting the determiner can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "a second pixel determiner" to determine how many pixels to filter in block 1, as Bjontegaard discloses. *See* Section X.A.1.d (element 1[c]).

- f. Element 5[e]: further comprising a first comparator configured to compare said first filter decision value calculated by said first decision value calculator to a first threshold value; and**

220. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 5[e]. *See* Section X.A.1.d (element 1[c]), X.A.2.b (element 3[b]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies comparisons between a first filter-decision value and a first threshold that would require a processor serving as a comparator, Ex-1012, 10 (providing code to determine "if(il3<lim)" *e.g.*, if $S0_1 < \text{Lim}(\text{qp})$); Ex-1001, 23:13-21 (noting the comparator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "a first

comparator” to compare the first filter decision value to a first threshold, as Bjontegaard discloses. *See* Section X.A.1.d (element 1[c]), X.A.2.b (element 3[b]).

- g. Element 5[f]: a second comparator configured to compare said second filter decision value calculated by said second decision value calculator to a second threshold value.**

221. In my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 5[f]. *See* Section X.A.1.e (element 1[d]), X.A.2.f (element 3[f]). Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard’s Code expressly specifies comparisons between a second filter-decision value and a second threshold that would require a processor serving as a comparator, Ex-1012, 10-11 (providing code to determine “if(ir3<lim)” *e.g.*, if $S0_2 < \text{Lim}(\text{qp})$); Ex-1001, 23:13-21 (noting the comparator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard’s coding method would have had a general processor programmed or a dedicated processor with “a second comparator” to compare the second filter decision value to a second threshold, as Bjontegaard discloses. *See* Section X.A.1.e (element 1[d]), X.A.2.f (element 3[f]).

4. Claim 7

- a. Element 7[a]: The device according to claim 5, further comprising: said first pixel determiner is configured to i) determine to filter two pixels in said first line of pixels**

in said block relative to said block boundary if said first filter decision value is below said first threshold value as determined by said first comparator and

222. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 7[a]. *See* Sections X.A.2.c (element 3[c], two pixels), X.A.3.d (element 5[c], first pixel determiner), X.A.3.f (element 5[e], first comparator).

- b. Element 7[b]: ii) determine to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value as determined by said first comparator; and**

223. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 7[b]. *See* Section X.A.2.d (element 3[d], one pixel), X.A.3.d (element 5[c], first pixel determiner), X.A.3.f (element 5[e], first comparator).

- c. Element 7[c]: said second pixel determiner is configured to i) determine to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value as determined by said second comparator and**

224. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 7[c]. *See* Section X.A.2.g (element 3[g], two pixels), X.A.3.e (element 5[d], second pixel determiner), X.A.3.g (element 5[f], second comparator).

- d. Element 7[d]: ii) determine to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second**

threshold value as determined by said second comparator.

225. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 7[d]. *See* Section X.A.2.h (element 3[h], one pixel), X.A.3.e (element 5[d], second pixel determiner), X.A.3.g (element 5[f], second comparator).

5. Claim 9

- a. Element 9[a]: The device according to claim 5, further comprising a threshold determiner configured to i) determine said first threshold value based on a quantization parameter associated with said block, and**

226. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 9[a].

227. Specifically, Bjontegaard discloses that threshold $\text{Lim}(\text{qp})$ comes from “a table defining threshold values for the filtering process depending on qp” (or “quantization parameter”). Bjontegaard, 19; *see also id.*, 9 (“QP is a pointer to the actual quantization parameter”). Bjontegaard's table to convert the “qp” of the block to $\text{Lim}(\text{qp})$ used as the threshold is provided below:

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Lim	7	8	9	10	11	12	14	15	17	20	22	25	28	31	35	39	44	49	55	62	69	78	88	98	110	124	139	156	175	197	221	248

Bjontegaard, 19. The use of similar lookup tables to determine thresholds based on quantization parameters of the blocks was well known in the art. *See, e.g.*, H.241, 27-28; B205, 131; Ex-1037, 206. Moreover, Bjontegaard discloses that, to determine

Lim(qp), the “maximum” qp of the two blocks should be used for the lookup table, Bjontegaard, 19, and thus the qp of block 1 (qp_1) would be selected when it is larger than the qp of neighboring block 2 (qp_2). In my opinion, therefore, Bjontegaard discloses determining a first threshold (Lim(qp)) based on a “quantization parameter” (qp) associated with block 1.

228. However, as I discussed above, a POSITA would have also been motivated to determine a first threshold specifically using the qp of block 1 (not just because it is the “maximum” qp of the two blocks). *See* Section X.A.1.e (element 1[d]).

229. A POSITA would have also understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard’s Code expressly specifies determining the first threshold that would require a processor serving as a threshold determiner, Ex-1012, 10 (providing code to determine first threshold “lim=LIMqp”); Ex-1001, 23:13-21 (noting the threshold determiner can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard’s coding method would have had a general processor programmed or a dedicated processor with a “threshold determiner” to determine the threshold (Lim(qp)) using the lookup table, as Bjontegaard discloses.

- b. Element 9[b]: ii) determine said second threshold value based on a quantization parameter associated with said neighboring block.**

230. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 9[b].

231. Bjontegaard discloses that the same $\text{Lim}(\text{qp})$ value serves as both the first threshold (when compared to $S0_1$) and the second threshold (when compared to $S0_2$), Bjontegaard, 19, and therefore Bjontegaard in view of Bjontegaard's Code teaches a threshold determiner that determines the second threshold, Section X.A.5.a (element 9[a]) (explaining how the threshold $\text{Lim}(\text{qp})$ is determined by a threshold determiner and "based on" a quantization parameter qp). In my opinion, Bjontegaard also discloses the first and second thresholds $\text{Lim}(\text{qp})$ are each "based on" the qp of both block 1 and neighboring block 2 because Bjontegaard evaluates both to select the "maximum" qp to serve as first and second threshold for $S0_1$ and $S0_2$, respectively. Bjontegaard, 19.

232. However, as I discussed above, a POSITA would have also been motivated to determine a second threshold specifically using the qp of neighboring block 2, *see* Section X.A.1.e (element 1[d]), and would have known that, like block 1, the second threshold would be determined by the threshold determiner, Section X.A.5.a (element 9[a]).

6. Claim 10

- a. Element 10[a]: The device according to claim 5, further comprising: a first offset calculator configured to calculate a first offset Δ based on $(9x(q_{0j} - p_{0j}) - 3x(q_{1j} - p_{1j}))/16$ wherein p_{0j} denotes a pixel value of a pixel closest to, in said first line of pixels, said block boundary, p_{1j} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary, q_{0j} denotes a pixel value of a pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary and q_{1j} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary;**

233. In my opinion, Bjontegaard in view of Bjontegaard's Code and a POSITA's knowledge of the art teaches or suggests element 10[a].

234. Specifically, Bjontegaard discloses the following first-offset equation is used when $Act_{edge}=1$ or 2: $Diff=(3(c-f)-8(d-e))/16$. Bjontegaard, 20. This equation can be rewritten using "p" and "q" for a line of pixels ("j") as follows:

$$\Delta = \frac{3(p_{1j} - q_{1j}) - 8(p_{0j} - q_{0j})}{16} = \frac{8(q_{0j} - p_{0j}) - 3(q_{1j} - p_{1j})}{16}$$

The only difference between Bjontegaard's equation and the one found in the claim is that the "8" was changed to a "9" for the coefficient applied to the difference between pixel q_0 and p_0 . A POSITA would have known that these coefficients provide one variable that controls how much blocking artifacts will be filtered. And because Bjontegaard's coefficients (8 and 3) are very close to those claimed (9 and

3), a POSITA would have expected them to have very similar deblocking filtering properties.

235. Specifically, a POSITA would have appreciated that the art was filled with extremely similar offset equations for deblocking filters that varied only as to two variables: (1) the weights assigned to each difference in pixel values: $X(q_0 - p_0)$; $Y(q_1 - p_1)$; and/or $Z(q_2 - p_2)$; and (2) the number of pixel differences— $(q_0 - p_0)$, $(q_1 - p_1)$, $(q_2 - p_2)$, etc.—incorporated into the equation. These two variables were known to control the frequency response of the filter, or, in other words, how much particular frequencies would be filtered. *See, e.g.*, Gonzales, 153, 273. By modifying these two variables, therefore, a POSITA could produce the desired balance between how much blocking artifacts would be filtered and maintaining overall image quality (by not over-blurring block boundaries). And because the prior art taught numerous similar offset equations that differed only as to these two variables, in my opinion, a POSITA would have considered these two variables points of routine experimentation, and therefore would have been motivated to experiment with these variables in Bjontegaard's offset equation.

236. For instance, B118 discloses a deblocking filter that incorporates a specific offset equation for the two pixels (p_0/q_0 , which B118 calls p_1/q_1) closest to the block boundary:

$$\Delta = \text{Clip}(-C, C, ((q_1 - p_1) \ll 2 + (p_2 - q_2) + 4) \gg 3)$$

B118, 48. The “delta” or offset equation could be re-written as follows (using p0/q0 convention, ignoring the clipping function for now, and assuming rounding to the nearest integer):

$$\Delta = \frac{4(q_0 - p_0) + (p_1 - q_1)}{8} = \frac{4(q_0 - p_0) - (q_1 - p_1)}{8}$$

See generally Section V.B.3.b (describing conversion of bit-shift form to division form).

237. Fu similarly discloses a deblocking filter that uses the following offset equation for the two pixels (p0/q0) closest to the block boundary:

$$\text{delta} = \text{Clip3}(-C, C, (((q_0 - p_0) \times 3 + (p_1 - q_1) + 4) \gg 3))$$

Fu, 8. The “delta” or offset equation could be re-written as follows (again, without clipping and assuming rounding):

$$\Delta = \frac{3(q_0 - p_0) + (p_1 - q_1)}{8} = \frac{3(q_0 - p_0) - (q_1 - p_1)}{8}$$

238. And as I mentioned previously, *see* Section V.B.3.b, the H.264 standard uses the following offset equation:

$$\Delta = \text{Clip3}(-t_c, t_c, (((q_0 - p_0) \ll 2) + (p_1 - q_1) + 4) \gg 3))$$

Ex-1037, 207. This equation can be re-written in the following form (again, without clipping and assuming rounding):

$$\Delta = \frac{4(q_0 - p_0) + (p_1 - q_1)}{8} = \frac{4(q_0 - p_0) - (q_1 - p_1)}{8}$$

239. Similarly, as part of the Video Coding Experts Group, a proposal (M21) was provided that offered the following offset equation:

$$\text{Difference} = \text{clip}(32 * (E - D) + 10 * (C - F) + 32) \gg 6 ;$$

Ex-1069, 2. This equation can be re-written in the following form (again, without clipping and assuming rounding):

$$\Delta = \frac{32(q_0 - p_0) + 10(p_1 - q_1)}{64} = \frac{32(q_0 - p_0) - 10(q_1 - p_1)}{64}$$

240. Although each of these offset-filter equations used four pixels as inputs, other prior-art offset-filter equations expanded the number to six pixels.

241. For instance, the H.241 standard uses the following offset equation:

$$\Delta = \text{Clip3}(-t_c, t_c, ((q_0 + (p_2 + q_1) \gg 1)) \gg 1 - ((p_0 + ((q_2 + p_1) \gg 1)) \gg 1))$$

H.241, 29. This equation can be re-written in the following form:

$$\Delta = \frac{2(q_0 - p_0) + (q_1 - p_1) - (q_2 - p_2)}{4}$$

This same equation also appears in Bjontegaard '048. Ex-1065, ¶ 47.

242. Similarly, B205 proposed using the following offset equation:

$$\Delta = \text{Clip}(-t_c, t_c, (13 * (q_0 - p_0) + 4 * (q_1 - p_1) - 5 * (q_2 - p_2) + 16) \gg 5))$$

B205, 131. This equation can be re-written in the following form:

$$\Delta = \frac{13(q_0 - p_0) + 4(q_1 - p_1) - 5(q_2 - p_2)}{32}$$

243. For ease of readability and comparison, I have summarized each of the above offset equations, which I have normalized below so that all equations are being divided by 16, as well as the offset equation identified in the '841 Patent, claim 10. The differences between the equations are identified in red:

Table 1: Condensed Offset Equation	
'841 Patent	$\frac{9(q_0 - p_0) - 3(q_1 - p_1)}{16}$
Bjontegaard	$\frac{8(q_0 - p_0) - 3(q_1 - p_1)}{16}$
B-118	$\frac{8(q_0 - p_0) - 2(q_1 - p_1)}{16}$
Fu	$\frac{6(q_0 - p_0) - 2(q_1 - p_1)}{16}$
H.264, List	$\frac{8(q_0 - p_0) - 2(q_1 - p_1)}{16}$
VCEG-M21	$\frac{8(q_0 - p_0) - 2.5(q_1 - p_1)}{16}$
H.241 Bjontegaard '048	$\frac{8(q_0 - p_0) + 4(q_1 - p_1) - 4(q_2 - p_2)}{16}$
B205	$\frac{7.5(q_0 - p_0) + 2(q_1 - p_1) - 2.5(q_2 - p_2)}{16}$

These equations can also be expanded, applying each coefficient to each pixel, as follows:

Table 2: Expanded Offset Equation	
'659 Patent	$\frac{9q_0 - 9p_0 - 3q_1 + 3p_1}{16}$
Bjontegaard	$\frac{8q_0 - 8p_0 - 3q_1 + 3p_1}{16}$
B-118	$\frac{8q_0 - 8p_0 - 2q_1 + 2p_1}{16}$
Fu	$\frac{6q_0 - 6p_0 - 2q_1 + 2p_1}{16}$
H.264, List	$\frac{8q_0 - 8p_0 - 2q_1 + 2p_1}{16}$
VCEG-M21	$\frac{8q_0 - 8p_0 - 2.5q_1 + 2.5p_1}{16}$
H.241 Bjontegaard '048	$\frac{8q_0 - 8p_0 + 4q_1 - 4p_1 - 4q_2 + 4p_2}{16}$
B205	$\frac{7.5q_0 - 7.5p_0 + 2q_1 - 2p_1 - 2.5q_2 + 2.5p_2}{16}$

244. Because these are just the offset equations that are used to calculate *how much* to modify pixels p0 and q0, however, I have also provided below the full filter equations ($p_0' = p_0 + \Delta$) that would be used to calculate the modified value of p0 itself by adding the offsets above to p0 (or, more specifically, $\frac{16p_0}{16}$):

Table 3: Full Filter Equation (adding 16p0/16)	
'659 Patent	$\frac{9q0 + 7p0 - 3q1 + 3p1}{16}$
Bjontegaard	$\frac{8q0 + 8p0 - 3q1 + 3p1}{16}$
B-118	$\frac{8q0 + 8p0 - 2q1 + 2p1}{16}$
Fu	$\frac{6q0 + 10p0 - 2q1 + 2p1}{16}$
H.264, List	$\frac{8q0 + 8p0 - 2q1 + 2p1}{16}$
VCEG-M21	$\frac{8q0 + 8p0 - 2.5q1 + 2.5p1}{16}$
H.241 Bjontegaard '048	$\frac{8q0 + 8p0 + 4q1 - 4p1 - 4q2 + 4p2}{16}$
B205	$\frac{7.5q0 + 8.5p0 + 2q1 - 2p1 - 2.5q2 + 2.5p2}{16}$

As shown, only the coefficient applied to p0 changes compared to the equations in Table 2.

245. As demonstrated above, and particularly as shown in Table 1, each of these offset equations followed similar patterns that differed only as to the two variables I identified above, and demonstrate the workable ranges for each of these variables. When experimenting with these two variables to design an offset equation, however, a POSITA would have also observed and been guided by several additional design constraints apparent from the above equations.

246. **First**, as demonstrated in Table 1, the magnitude of the coefficient applied to (q0–p0) would generally be 2-4 times larger than the coefficient applied to (q1–p1) and (q2–p2), if included. In other words, the difference between the pixel

values closest to the block boundary would be given more weight when calculating the offset. The ratio between these coefficients was also known to control the “width” of the filter, *e.g.*, how wide the filtering curve is in the frequency domain. *See* Gonzales, 153, 273. A higher ratio between these coefficients introduces higher frequency components, which leads to a “wider” frequency response.¹⁰ Ex-1072, 7 (“large value coefficients carry more weight in the construction of output samples,” and providing examples in which weighting the center coefficients differently relative to more-distant coefficients changes the width of the filter). The “wider” frequency-response filter will consequently attenuate sharp transitions in the image less, and thus better preserve edges in the image.

247. **Second**, summing the coefficients (in Table 1) in the numerator and dividing by the denominator would produce a fraction between $\frac{1}{4}$ and $\frac{1}{2}$. This meant that the total amount the p0 and q0 pixel values would be modified would be approximately $\frac{1}{4}$ to $\frac{1}{2}$ of the differences in pixel values on each side of the boundary, depending on the amount of filtering desired. This ratio ensures that pixels would not be over-filtered. For instance, if the total amount of filtering exceeded $\frac{1}{2}$ the

¹⁰ For example, a unit impulse (*i.e.*, a signal which has a value of one at the origin and zero everywhere else) has a flat frequency response (*i.e.*, the frequency response is constant for all frequencies), and vice versa.

difference between p_0 and q_0 , then if p_0 started with a value lower than q_0 , it would end up producing a value higher than q_0 after filtering. The expectation of deblocking filtering, however, is that it will cause p_0 and q_0 to be closer together, and not to have their relationship invert. *See, e.g.*, Ex-1039, 215.

248. **Third**, the “window” used to calculate the offset (Tables 1 and 2) was typically either four (q_0-p_0 ; q_1-p_1) or six (q_0-p_0 ; q_1-p_1 ; q_2-p_2) pixels long. The prior-art equations typically used six pixels or less not only because more pixels increased the computing resources required, but because longer filters tended to introduce a “ringing” effect into the filtered image. This is a known aspect of filter design, and can occur when a low-pass filter first overshoots its target output and subsequently overcompensates and thus undershoots its target. *See, e.g.*, Ex-1073, 3 (“The longer the impulse response length is, the wider the ringing noise spreads”). Over time, those in the art had found empirically that using six pixels or less in an offset equation tended to avoid significant ringing.

249. **Fourth**, it was known that the “DC” gain of the offset should equal 1. *See, e.g.*, Gonzalez, 153-54 (noting “[t]he denominator ... is simply the sum of the mask coefficients”); Ex-1072, 7 (explaining that “[s]etting the coefficients [to] sum to 1 makes a filter with unity gain at DC”). This can be conceptualized more easily in audio-signal processing: audio signals are often filtered to eliminate specific frequency components to boost the *apparent* loudness of other frequencies, without

changing the overall loudness (or gain) of *all* frequencies. In image processing, gain controls the “brightness” of the pixels (how light or dark they appear), and when performing deblocking filtering, it was understood that the goal is to blur the blocking artifacts by eliminating the target frequencies *without* altering the overall brightness of the pixels. Keeping a “DC” gain of 1 means that if a “flat” signal of 1 is input into to the offset equations, the offset produces a value of zero. This is accomplished by ensuring the coefficients in the numerator of the offset equation (Table 2) sum to zero. Gonzalez, 696, 699. Relatedly, when a “flat” signal of 1 is input into the full filter equation (*e.g.*, $p1' = p1 + \text{offset}$), the output should stay 1. This is accomplished by ensuring the coefficients in the filter equation (Table 3), when summed, equal the denominator so that they divide into 1. Gonzalez, 153-54. As shown in Table 3, any coefficient applied to pixel $p1$ and, if used (*e.g.*, B205 and H.241), pixel $p2$ is cancelled out by the corresponding coefficient for pixels $q1$ and $q2$, respectively, and then summing the coefficients of $p0$ and $q0$ produces a value of 16, which is then divided by 16 to produce a final ratio of 1. Although this design consideration is not of particular concern when weights are applied to pixel differences (*e.g.*, $X(q0-p0)$) because the weight applied to one pixel will always be canceled by the weight applied to the other pixel (*e.g.*, $Xq0-Xp0$), it is still a relevant design consideration for non-difference-formatted equations, such as for $p1/q1$ offset equations I address below.

250. **Fifth**, while there are exceptions, the value of the right-shift was generally between 3 and 6 (meaning the offset is divided by a power of 2^n between 8 and 64). A higher right-shift value would allow for more-precise filter design because the coefficients and their ratios could be controlled to a greater degree. For instance, VCEG-M21 relied on one extreme and used a right-shift of 6, thus providing more control over the ratio between its coefficients. On the other extreme is H.241 (that specified that it was intended to be a low-complexity deblocking filter, H.241, 1), which provided a right shift of 1, thus providing little room to experiment with different coefficient ratios. However, right shifts greater than 6 were rare because that required extra bits to represent the coefficients.

251. **Finally**, integer coefficients (not decimals) are used to minimize the computational complexity of the calculations. That is also why right shifts are used for offset equations instead of using decimal coefficients applied to a pixel (or pixel differences, like the first-offset equations).

252. In my opinion, these prior-art first-offset equations demonstrate that those in the art were routinely experimenting with the coefficients and number of pixels used to control the frequency response of their filters. Even the '841 Patent recognized that the equation it proposes is not essential, as it explains that “[d]ifferent such functions are possible and can be used.” Ex-1001, 18:1-24; *see also id.*, 17:1-8.

253. For this reason, in my opinion, a POSITA would have experimented with Bjontegaard's equation (as those in the art had routinely experimented with the coefficients and filter length, usually through an empirical trial-and-error process), and specifically with other coefficient values that were close to those in Bjontegaard, to achieve a desired amount of filtering of blocking artifacts without also degrading the overall image quality. A POSITA would have experimented with different coefficient values within a workable range Bjontegaard and others had found throughout the prior art. A POSITA would be driven to experiment in view of competing demands for filtering video, such as filter quality versus computational complexity or performance of the filter across a range of different scenarios. And, in my opinion, a POSITA would have found the claimed offset equation obvious because the coefficients are so close—the claimed weights (9 and 3) can be obtained by increasing Bjontegaard's (8 and 3) by one value—that a POSITA would have expected only slight differences between their filtering performance, as shown below through some example applications¹¹ of the two equations:

¹¹ I have included the p2 pixel in these examples, even though it is not used in the equations, because I later use the same sets of pixels to evaluate the differences between Bjontegaard and the claimed p1 offset equations.

p2	p1	p0	q0	q1	Type	$\frac{9(q0 - p0) - 3(q0 - p0)}{16}$	$\frac{8(q0 - p0) - 3(q0 - p0)}{16}$
200	150	150	150	150	Step (p1)	0	0
200	200	150	150	150	Step (p0)	9	9
200	200	200	150	150	Step (q0)	-19	-16
150	160	170	180	190	Ramp (10)	0	-1
125	150	175	200	225	Ramp (25)	0	-2
200	250	200	250	200	Alternating	38	34
255	50	50	255	255	Line	77	64
255	255	50	50	255	Line	0	0
175	200	150	150	175	Random	5	5
50	125	100	150	125	Random	28	25
75	100	150	75	125	Random	-47	-42

A POSITA would have appreciated that coefficients 9 and 3 place slightly more emphasis on the difference between the p0 and q0 pixels relative to the difference between the p1 and q1 pixels in defining the offset (and therefore tend to apply *slightly* more filtering as the difference between these pixels increases), while still meeting all of the other offset-equation considerations: (1) it maintained a ratio of 3 that is between 2 and 4 for **X**(q0-p0); **Y**(q1-p1); (2) it would produce an offset of approximately $\frac{3}{8}$, just slightly more than Bjontegaard's original offset equation of

¼, and still between ¼ and ½; (3) it used four pixels; and (4) it maintained a DC gain of 1 (as shown in Tables 2 and 3).

254. Finally, in my opinion, a POSITA would have understood that Bjontegaard was intended to be run on a computer processor, as confirmed by Bjontegaard's Code, *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies first-offset calculations that would require a processor serving as a first-offset calculator, Ex-1012, 11 (providing code for first-offset calculator using "id=(3*(li[2]-li[5])-8*(li[3]-li[4]))/16" (omitting pointers)); Ex-1001, 23:13-21 (noting the first-offset calculator can be implemented on a processor). Accordingly, a POSITA would have understood that the processor would be operating a "first offset calculator" to calculate the "first offset."

255. To the extent the Patent Owner argues that selecting the numerator coefficients 9 and 3, and denominator coefficient of 16 would have been non-obvious because it produced unexpected results, I disagree.

256. I understand that the claimed filter equation of $\frac{9(q_0-p_0)-3(q_1-p_1)}{16}$ as well as the specific offset equations for the p1 and q1 pixels addressed in elements 10[d] and 10[f], below—had been proposed by the inventors for inclusion in the HEVC standard. Ex-1043, 2. I also understand that others had suggested using different offset-filter equations for the deblocking filter. I have summarized some of the alternative offset equations other standards participants had proposed below, all

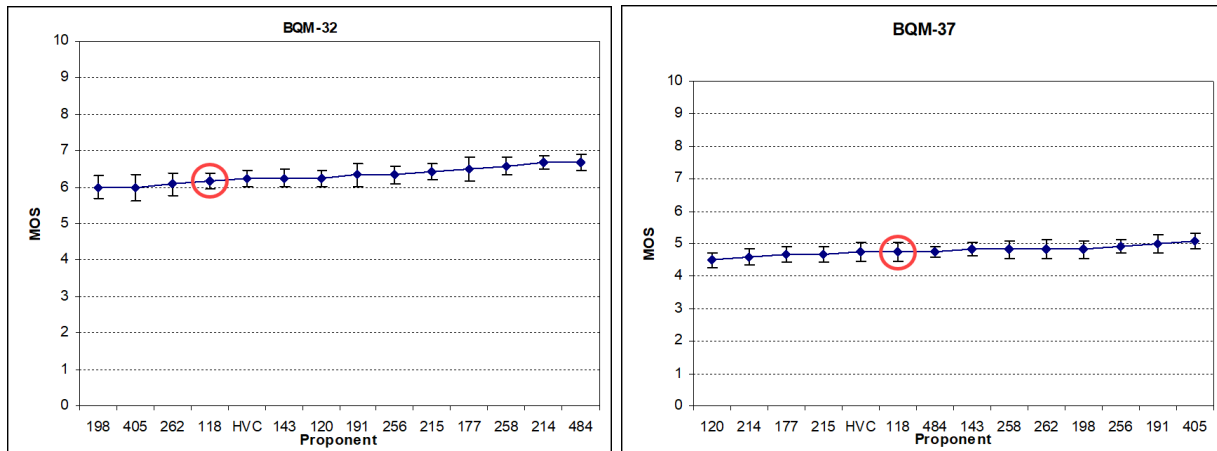
of which I have normalized to be divided by 16 and written in standard convention of $q_j - p_j$, with the differences between the equations identified in red.

JCT-VC Documents	Filter Equations (Normalized)
'659 Patent	$\frac{9(q_0 - p_0) - 3(q_1 - p_1)}{16}$
F120	$\frac{6(q_0 - p_0) - 1.5(q_1 - p_1)}{16}$
F256	$\frac{6.5(q_0 - p_0) + 2(q_1 - p_1) - 2(q_2 - p_2)}{16}$
F258	$\frac{8(q_0 - p_0) + 2(q_1 - p_1)}{16}$

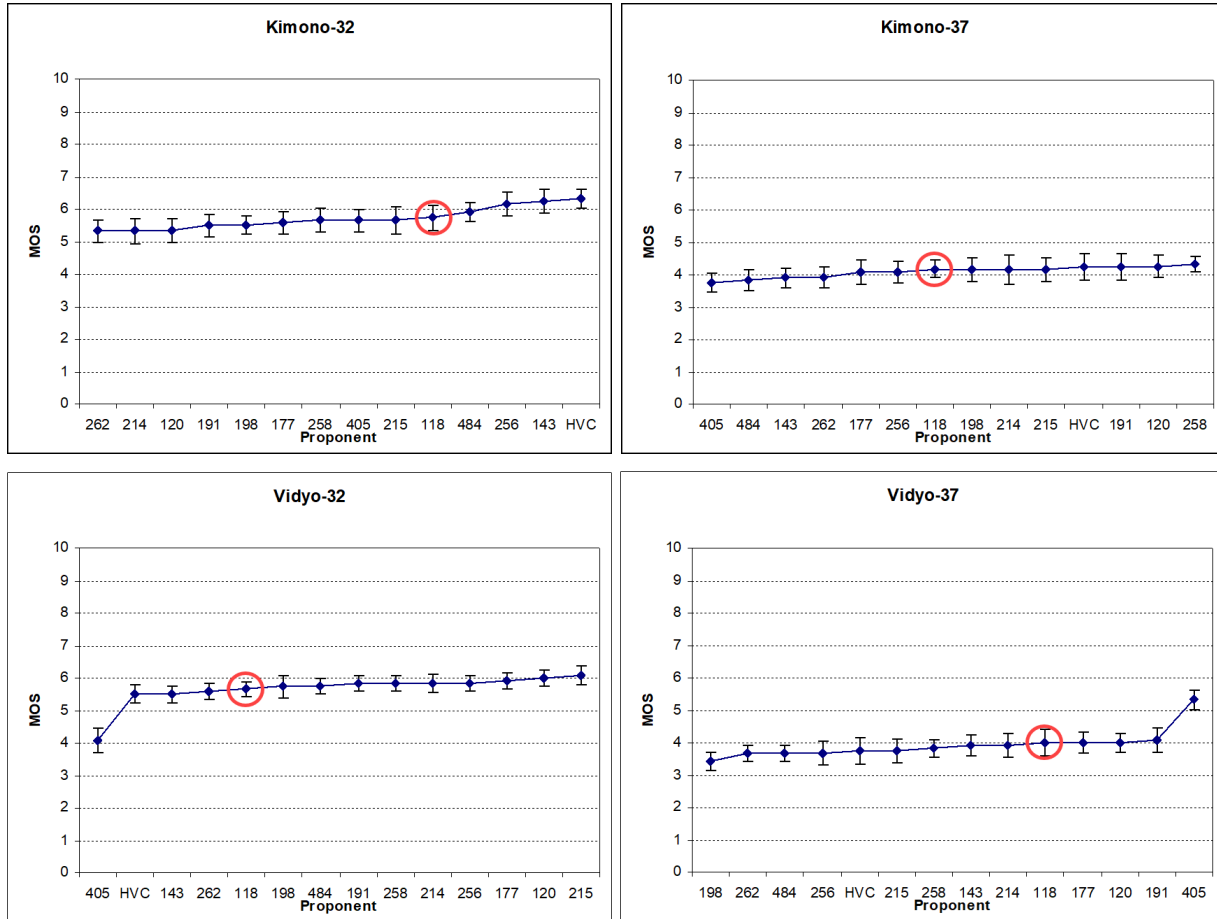
Ex-1044, 2; Ex-1045, 6; Ex-1046, 5.¹²

¹² It appears as if the filter equation proposed in F-258 contains a typographical error, and the second coefficient for $Y(q_1 - p_1)$ should be “-2” instead of “+2.” Specifically, F-258 states that “a normal filter of MPEG-4 Part.10 AVC/H.264 is used for weak filtering as follows: $\Delta = \text{Clip3}(-tc, tc, ((q_0 - p_0) \ll 2) + (q_1 - p_1) + 4) \gg 3$,” Ex-1046, 5. But H.264 recites the equation: $\Delta = \text{Clip3}(-tc, tc, ((q_0 - p_0) \ll 2) + (p_1 - q_1) + 4) \gg 3$. Ex-1037, 207. By mistakenly switching the order of operation for p_1 and q_1 , that switches the polarity of the coefficient. As I noted previously, H.264’s equation written in division form is: $\frac{4(q_0 - p_0) - (q_1 - p_1)}{8}$, or $\frac{8(q_0 - p_0) - 2(q_1 - p_1)}{16}$. Moreover, using “+” instead of “-” would mean that the p_0/q_0 would be at risk of over-filtering

257. Those involved in the standardization process ultimately performed subjective-quality tests of thirteen different deblocking-filter proposals (compared to one “anchor,” used as a baseline) with a variety of different offset equations. Ex-1047, 1. Below are the results of those comparisons, which involved three different test-video sequences (called BQM, Kimono, and Vidyo3), using two different encoding conditions for each proposal. *Id.*, 1. The deblocking filter that incorporated the claimed offset equations is circled in red.



because the total offset would be approximately 10/16ths of the difference in pixels over the boundary.



Ex-1047, 3-8 (red circles added).

258. The test demonstrated that “for most of the cases under test the performance of the proposals was equal to that of the anchor,” and only in very limited circumstances did a few proposals materially deviate. Ex-1047, 8. Ultimately, the group recommended that the “luma part” of the deblocking filter proposed by Ericsson should be adopted because “luma filter design and separate on/off decision for P/Q edge side” produced a “1.2% gain [in deblocking efficiency] without subjective quality loss.” Ex-1048, 176. In other words, the exact offset equation used had no material impact on the perceptual quality of the deblocking

filter, but the inventor's proposal was adopted because it resulted in a 1.2% increase in deblocking efficiency, which had largely been attributed to its asymmetric filtering decision that existed in the prior art.

259. Even if the 1.2% deblocking efficiency gain could be attributed to the offset equations themselves, I consider that a difference in degree and not kind because, in my opinion, a POSITA would have considered such small efficiency gains relatively minor, and one that would be expected when optimizing the parameters of prior-art filter equations. In my opinion, those involved in the standardization process did not recognize any unexpected, material transformation provided by those equations compared to what existed in the prior art.

260. To the extent Patent Owner argues that the prior art taught away from the claimed invention, I have not seen any such evidence. I reserve the right to respond if Patent Owner identifies such alleged evidence.

261. To the extent Patent Owner argues that the coefficients for the offset equations or number of pixels used were not recognized as result-effective variables, I disagree with such an argument. Based on my understanding of the art, these equations were routinely experimented with based on trial-and-error testing to produce a result—the amount block boundaries were filtered without degrading overall image quality—that most satisfied the designer. Indeed, the fact that those in the video-coding community proposed various equations with slightly different

coefficients or number of pixels incorporated into the equations demonstrates that those in the art had recognized these variables as result-effective; otherwise, they would have all used the same equation and experimented only with other aspects of the deblocking filter.

262. Finally, to the extent Patent Owner argues that there were too many options to choose from that would not invite routine experimentation, I would disagree with that argument. As I had mentioned above, those involved in deblocking-filter design understood the general parameters within which a deblocking filter should operate, which greatly limited the available options from which to choose (*e.g.*, maintaining a ratio between coefficients to ensure proper filter performance; not using too many pixels for efficiency reasons and to avoid introducing ringing artifacts; not using large coefficients for efficiency, etc.). Moreover, the fact that those in the art were routinely experimenting with and proposing different (but similar) coefficients reinforces that the options were not so numerous so as to discourage routine experimentation. Instead, those skilled in the art were actively engaged in such experimentation at the relevant time.

263. For these reasons, in my opinion, the offset equation ultimately selected by the inventors would have been a result of routine experimentation and optimization by modifying known result-effective variables, and therefore the recited equation would have been obvious to a POSITA at the time of the invention.

- b. Element 10[b]: a first pixel modifier configured to modify said pixel value of said pixel closest to said block boundary in said first line of pixels by adding said first offset calculated by said first offset calculator to said pixel value of said pixel closest to said block boundary in said first line of pixels;**

264. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 10[b].

265. Specifically, Bjontegaard discloses " $d_{\text{modified}} = d + \text{Diff}$," meaning that the first-pixel value d is modified by adding the first-offset Diff , and pixel d is the pixel closest to the block boundary. Bjontegaard, 20; *see also id.*, 19 (noting "[t]here is a block boundary between pixels d and e ").

266. Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies first-pixel modifications that would require a processor serving as a first-pixel modifier, Ex-1012, 11 (providing code for pixel modification of $li[3]$ (pixel d) using " $il1 = li[3] + id$ " ($d_{\text{modified}} = d + \text{Diff}$) and " $li[3] = il1$ " ($d = d_{\text{modified}}$) for subsequent calculations, before outputting the final modified pixel value using " $lu[3] = \max(0, \min(255, il1)$ " to clip the output modified pixel value between 0 and 255 (omitting pointers)); Ex-1001, 23:13-21 (noting the pixel modifier can be implemented on a processor). Accordingly, a POSITA would have understood that

a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with "first pixel modifier" to modify the value of pixel d closest to the boundary in block 1.

- c. **Element 10[c]: a second pixel modifier configured to modify said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels by subtracting said first offset calculated by said first offset calculator from said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels;**

267. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 10[c].

268. Specifically, Bjontegaard discloses " $e_{\text{modified}} = e - \text{Diff}$," meaning that the first-pixel value e is modified by subtracting the first-offset Diff, and pixel e is the pixel closest to the block boundary in the neighboring block. Bjontegaard, 20.

269. Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (e.g., a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies second-pixel modifications that would require a processor serving as a second-pixel modifier, Ex-1012, 11 (providing code for pixel modification of $li[4]$ (pixel e) using " $ir1 = li[4] - id$ " ($e_{\text{modified}} = e - \text{Diff}$), and " $li[4] = ir1$ " ($e = e_{\text{modified}}$) for subsequent calculations, before outputting the final modified pixel value using

“ $lu[4]=\max(0,\min(255,ir1))$ ” to clip the output modified pixel value between 0 and 255 (omitting pointers)); Ex-1001, 23:13-21 (noting the pixel modifier can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard’s coding method would have had a general processor programmed or a dedicated processor with “second pixel modifier” to modify the value of pixel e closest to the boundary in neighboring block 2.

- d. Element 10[d]: a second offset calculator configured to calculate, if said first filter decision value is below said first threshold value as determined by said first comparator, a second offset based on $(p0_j + p2_j - 2p1_j + 2\Delta)/4$ wherein $p2_j$ denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;**

270. In my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 10[d].

271. For pixel c, which is the second-closest pixel to the block boundary, when $Act_{11}=2$, Bjontegaard discloses a second-offset equation: $Diff=5(b-2c+d)/16$. Bjontegaard, 20.

272. However, in my opinion, a POSITA would have understood that Bjontegaard’s equation would be read as: $Diff_{[1]}=5(b-2c+d_{\text{modified}})/16$, because Bjontegaard provides a logical walkthrough of the deblocking filter’s operation, in which each section logically follows those that precede it. The reason that Bjontegaard re-initializes the variable “Diff” to equal zero in Sections §§ 4.5.6.1 and

4.5.6.2 is because Bjontegaard uses the same variable “Diff” (for efficiency) for multiple calculations. Without re-initializing “Diff” at the start of each section, the value of “Diff” previously stored in memory from calculations in a preceding section would be used.

273. The same reasoning therefore applies to the value of pixels d and e calculated in Section § 4.5.5. Although Bjontegaard labels the modified value of d as d_{modified} , logically it is just a redefined value of d (*i.e.*, $d = d + \text{Diff}$) because d_{modified} will replace d in the image.

274. Bjontegaard’s Code would have confirmed this understanding, because Bjontegaard discloses modifying pixel d *before* then filtering pixel c . *See* Ex-1012, 11. Specifically, Bjontegaard’s Code labels the first offset using variable “ id ” and pixel d using variable “ $li[3]$.” *Id.* The offset “ id ” is then applied to “ $li[3]$ ” using equations (and omitting the pointers): “ $il1=li[3]+id$ ” (to calculate the new pixel value) and “ $li[3] = il1$ ” (to apply the new pixel value to $li[3]$ (pixel d)). Only then does filtering proceed to pixel c ($li[2]$) using the new value of pixel d ($li[3]$). *Id.* (identified by “/* center left */” section of code). In my opinion, therefore, Bjontegaard discloses using the *modified* value of d in the equation $\text{Diff}_{[1]}=5(b-2c+d_{\text{modified}})/16$.

275. Even if Bjontegaard did not expressly disclose this, in my opinion, it would have been obvious to modify Bjontegaard to use d_{modified} instead of d . That is because, as was common in the art, filtering pixels further from the boundary should

depend on how much pixels closer to the boundary are filtered. *See, e.g.*, H.241, 29 (modifying p_0 using Δ , and p_1 using $\Delta/2$); B205, 131 (same); Ex-1039, 215 (applying $\Delta/2$, $\Delta/4$, $\Delta/8$ when moving further from boundary). If the offset applied to pixels further from the boundary did not depend on the offset applied to pixels closer to the boundary, that could result in new, unwanted discontinuities in the vicinity of the boundary as a result of the deblocking filter, which would be contrary to the intended purpose of deblocking to eliminate such discontinuities. By using d_{modified} instead of d , therefore, Bjontegaard would ensure the new pixel value for pixel c is determined based on the modified pixel d_{modified} and thus has some relationship to the offset applied to pixel d .

276. Applying this reasoning, Bjontegaard's equation either was or would obviously be re-written as follows:

$$\text{Diff}_{[c]} = \frac{5(b-2c+d_{\text{modified}})}{16}$$

$$\text{where } d_{\text{modified}} = d + \text{Diff}_{[d]}:$$

$$= \frac{5(b-2c+d+\text{Diff}_{[d]})}{16}$$

$$= \frac{5(b-2c+d)}{16} + \frac{5*\text{Diff}_{[d]}}{16}$$

And re-writing this equation with delta and p , consistent with the '841 Patent:

$$\Delta_{p1} = \frac{5(p2 - 2p1 + p0)}{16} + \frac{5\Delta_{p0}}{16}$$

277. In my opinion, a POSITA would have therefore recognized that the second-offset equation Bjontegaard discloses has two distinct components: (1) a high-pass filter component ($\frac{5(p_2 - 2p_1 + p_0)}{16}$), evident from the Laplacian identified in blue; and (2) a first-offset component ($\frac{5\Delta}{16}$) identified by the first offset (Δ) also in blue. The only differences between Bjontegaard's second-offset equation and the one found in 10[d] are the weights applied to each of these two components:

Claimed Equation	Bjontegaard's Equation
$\Delta = \frac{1(p_2 - 2p_1 + p_0)}{4} + \frac{1\Delta}{2}$	$\Delta = \frac{5(p_2 - 2p_1 + p_0)}{16} + \frac{5\Delta}{16}$

278. In my opinion, for the same reasons I discussed above with respect to the coefficients applied to each pixel difference in the first-offset equations, *see* Section X.A.6.a (element 10[a]), a POSITA would have also recognized that the coefficients applied to each of these components in the second-offset equation are variables¹³ subject to experimentation and optimization because they too control the frequency response of the second-offset filter. In fact, several other prior-art

¹³ As I discuss in further detail, another variable subject to routine experimentation was the exact high-pass filter equation used for the high-pass component.

references combined these same two components, but applied different weights to each of them to achieve the desired level of filtering.

279. Fu, for instance, discloses the following equation to modify the p1 pixel:

$$P_1 = \text{Clip1} (p_1 + \text{Clip3} (-C, C ((P_0 - p_1) \times 3 + (p_2 - Q_0) + 4) >> 3))$$

Fu, 8:17. This equation can be re-written (without clipping, for simplicity) as follows:

$$P_1 = p_1 + \{[3(\text{P0} - p_1) + (p_2 - \text{Q0}) + 4] >> 3\}$$

where $\text{P0} = p_0 + \Delta$ (the first offset)

$\text{Q0} = q_0 - \Delta$, and

$$= p_1 + \{[3(p_0 + \Delta - p_1) + (p_2 - q_0 + \Delta) + 4] >> 3\}$$

$$= p_1 + (p_2 - 3p_1 + 3p_0 - q_0 + 4\Delta + 4) >> 3$$

$$= p_1 + \frac{p_2 - 3p_1 + 3p_0 - q_0 + 4\Delta}{8}$$

$$= p_1 + \frac{p_2 - 3p_1 + 3p_0 - q_0}{8} + \frac{\Delta}{2}$$

See generally Section V.B.3.b (describing conversion of bit-shift form to division form). Fu therefore discloses a second-offset equation of: $\frac{p_2 - 3p_1 + 3p_0 - q_0}{8} + \frac{\Delta}{2}$.

280. The only difference between Bjontegaard and Fu is which high-pass filter equation—the other variable subject to experimentation that I noted above—is used to characterize additional high frequencies for filtering ($p_2 - 3p_1 + 3p_0 - q_0$ for Fu, compared to $p_2 - 2p_1 + p_0$ for Bjontegaard), and how much weight is applied to each

component (1/8 for Fu's high-pass component and 1/2 for Fu's first-offset component, compared to 5/16 for both of Bjontegaard's components).

281. Similarly, during the Joint Video Team ("JVT") proceedings to develop the H.264 standard, in a document labeled "B118," the JVT proposed as part of the draft standard using a p1 offset equation that combined a weighted Laplacian high-pass filter component with a weighted first-offset component. B118, 48. The JVT presented the following equation, using "p1" as a notation for the pixel closest to the block boundary, which I have converted to p0 notation for consistency:

$$P1 = p1 + (p2 + P0 - p1 \ll 1) \gg 1)$$

$$\text{where } P0 = p0 + \Delta \text{ (the first offset)}$$

$$= p1 + (p2 + p0 + \Delta - p1 \ll 1) \gg 1)$$

$$= p1 + \frac{p2 - 2p1 + p0 + \Delta}{2}$$

$$= p1 + \frac{p2 - 2p1 + p0}{2} + \frac{\Delta}{2}$$

B118 therefore discloses a second-offset equation of $\frac{p2 - 2p1 + p0}{2} + \frac{\Delta}{2}$.

282. The only difference between Bjontegaard and B118 is the coefficient that is applied to each component (1/2 for B118, compared to 5/16 for Bjontegaard).

283. Because other prior art disclosed second-offset equations that included the same two components as Bjontegaard, with different coefficients applied to each, in my opinion, a POSITA would have been motivated to experiment and optimize

Bjontegaard's coefficients to achieve a desired balance between filtering blocking artifacts and maintaining overall image quality, just like with the first-offset equation. And, in my opinion, a POSITA would have found the claimed second-offset equation obvious because the coefficients are close enough—the claimed weights ($\frac{1}{4}$ and $\frac{1}{2}$) can be obtained by adjusting each of Bjontegaard's component weights ($\frac{5}{16}$) by a few fractions—that a POSITA would have expected only slight differences between their filtering performance, as shown below through some example applications of the two equations (using each of the first-offset values calculated above, Section X.A.6.a (element 10[a]), for each respective second offset):

p2	p1	p0	q0	q1	Type	$\frac{1(p2 - 2p1 + p0)}{4} + \frac{1\Delta}{2}$	$\frac{5(p2 - 2p1 + p0)}{16} + \frac{5\Delta}{16}$
200	150	150	150	150	Step (p1)	13	16
200	200	150	150	150	Step (p0)	-8	-13
200	200	200	150	150	Step (q0)	-10	-5
150	160	170	180	190	Ramp (10)	0	0
125	150	175	200	225	Ramp (25)	0	-1
200	250	200	250	200	Alternating	-6	-21
255	50	50	255	255	Line	90	84
255	255	50	50	255	Line	-51	-64
175	200	150	150	175	Random	-16	-22
50	125	100	150	125	Random	-11	-23
75	100	150	75	125	Random	-17	-5

284. Specifically, a POSITA would have understood that the high-pass filter component is intended to characterize high-frequency pixel changes present near the block boundary for additional filtering, and adjusting the weight of this component would control how much those high frequencies would be filtered. A POSITA would have therefore expected that modifying the coefficient from 5/16 to 1/4—which is just a difference of 1/16th—would have meant that *slightly* less additional high-frequency filtering would be applied by the second offset, and therefore *slightly* less blurring would occur (at least due to this component).

285. A POSITA would have also understood that the first-offset component is intended to apply some fraction of the same offset applied to pixels closer to the block boundary to pixels further from the boundary to create a smoother transition, and adjusting the weight of this component would control how much of the first offset is carried over to the second offset to further blur the blocking artifact. *See* Ex-1039, 215 (applying $\Delta/2$, $\Delta/4$, $\Delta/8$ when moving further from boundary). A POSITA would have therefore expected that modifying the weight from $5/16$ to $1/2$ —which is just a difference of $3/16$ ths—would have meant that *slightly* more of the first offset would be applied as part of the second offset, and therefore *slightly* more blurring would occur (at least due to this component). In sum, weighing each of these components relative to one another controls the *overall* amount of filtering performed by the second offset.

286. For these reasons, in my opinion, a POSITA would have been motivated to experiment with the coefficients applied to each of the high-pass component and first-offset component—as Bjontegaard, Fu, and B118 demonstrate to achieve different amounts of filtering—knowing that they controlled how much each component contributed to the second offset. And a POSITA would have found applying a weight of $1/4$ to the high-pass Laplacian component, and $1/2$ to the first-offset component obvious in light of Bjontegaard, based on the different coefficients applied to these components throughout the art (*e.g.*, Fu and B118).

287. Moreover, a POSITA would have recognized that the second (and third, addressed in further detail below) offset equations claimed do not produce any unexpected results, for the reasons I discussed previously. *See* Section X.A.6.a (element 10[a]). In fact, not only did the '841 Patent state that “[d]ifferent such functions are possible and can be used,” Ex-1001, 18:1-24, but another patent sought by the named inventors for these same equations recognized that the coefficients for the components in the second (and third) offset(s) can be changed, depending on how much filtering is desired, Ex-1063, 15:42-45 (“one can use a stronger low-pass filter in combination with the first offset. In this case, the second offset is calculated based on, such as equal to, $(p_2 + p_0 - 2xp_1 + \Delta)/2$,” instead of $(p_2 + p_0 - 2xp_1 + 2\Delta)/4$, as claimed).¹⁴ In my opinion, this reinforces what would have been already apparent to a POSITA: that the weights applied to each component simply control how much

¹⁴ The high-pass component $(p_2 - 2p_1 + p_0)$ is referred to in the patent as a “low-pass” filter—despite being a high-pass equation—because once the high-pass filter in the offset is *applied* to the original pixel value in the filter equation $(p_1' = p_1 + \Delta_{p1})$, it becomes a low-pass filter by eliminating the (higher) frequencies the high-pass component has characterized (*e.g.*, $p_1' = p_1 + (p_2 - 2p_1 + p_0)/4 = (p_2 + 2p_1 + p_0)/4$, which is a low-pass filter).

the pixel is ultimately modified, balancing the desired contributions of the high-pass and first-offset components.

288. In my opinion, a POSITA also would have had a reasonable expectation of success in designing a filter equation for the p1 (and q1) pixels because the prior art was replete with examples, the design considerations were known, and it was largely understood to be an empirical exercise of testing different filter coefficients to arrive at a desired outcome (*e.g.*, overall image quality and deblocking performance, which depends on the amount of the offset).

289. Finally, a POSITA would have also understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies second-offset calculations that would require a processor serving as a second-offset calculator, Ex-1012, 11 (providing code for a second-offset calculator using " $id=(5*(li[1]-2*li[2]+li[3])/16$ " (omitting pointers)); Ex-1001, 23:13-21 (noting the second-offset calculator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or dedicated processor with a "second offset calculator" to calculate the second offset, as Bjontegaard discloses.

- e. **Element 10[e]: a third pixel modifier configured to modify, if said first filter decision value is below said first threshold as determined by said first comparator, said pixel value of said pixel next closest to said block boundary in said first line of pixels by adding said second offset calculated by said second offset calculator to said pixel value of said pixel next closest to said block boundary in said first line of pixels;**

290. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 10[e].

291. Specifically, Bjontegaard discloses filtering a third pixel c , which is the second-closest pixel from the block boundary, when the decision value is less than a threshold. *See* Sections X.A.2.b-X.A.2.d (elements 3[b]-[d]).

292. Bjontegaard also discloses " $c_{\text{modified}} = c + \text{Diff}$," meaning that the third pixel c is modified by adding the second offset (Diff). Bjontegaard, 20.

293. Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (e.g., a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies third-pixel modifications that would require a processor serving as a third-pixel modifier, Ex-1012, 11 (providing code for pixel modifications of $li[2]$ (pixel c) using " $il2 = li[2] + id$ " ($c_{\text{modified}} = c + \text{Diff}$), before outputting the final modified pixel value using " $lu[2] = \max(0, \min(255, il2))$ " to clip the output modified pixel value between 0 and 255 (omitting pointers)); Ex-1001, 23:13-21 (noting the pixel

modifier can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with a "third pixel modifier" to modify the value of pixel c second closest to the boundary in block 1.

- f. Element 10[f]: a third offset calculator configured to calculate, if said second filter decision value is below said second threshold value as determined by said second comparator, a third offset based on $(q0_j + q2_j - 2q1_j - 2\Delta)/4$ wherein $q2_j$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary; and**

294. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 10[f]. *See* Section X.A.6.d (element 10[d]).

295. Specifically, Bjontegaard discloses calculating a third offset for the second closest pixel f in neighboring block 2 using: $\text{Diff} = 5(e - 2f + g)/16$. Bjontegaard, 20.

296. As I discussed previously, however, a POSITA would have known or found it obvious that Bjontegaard's equation uses pixel e_{modified} . *See* Section X.A.6.d (element 10[d]). Like with pixel d, Bjontegaard's Code would have confirmed this understanding, because Bjontegaard discloses modifying pixel e *before* then filtering pixel f. *See* Ex-1012, 11. Specifically, Bjontegaard's Code labels the first offset

using variable “id” and pixel e using variable “li[4].” *Id.* The offset “id” is then applied to “li[4]” using equations: “ir1=img->li[4]-id” (to calculate the new pixel value) and “img->li[4] = ir1” (to apply the new pixel value to li[4] (pixel e)). Only then does filtering proceed to pixel f (li[5]) using the new value of pixel e (li[4]). *Id.* (identified by “/* center left */” section of code). Applying this reasoning, Bjontegaard’s equation either was or would obviously be re-written as follows:

$$\begin{aligned} \text{Diff}_{[f]} &= \frac{5(\text{e}_{\text{modified}} - 2f + g)}{16} \\ \text{where } \text{e}_{\text{modified}} &= e - \text{Diff}_{[e]}: \\ &= \frac{5(e - \text{Diff}_{[e]} - 2f + g)}{16} \\ &= \frac{5(e - 2f + g)}{16} - \frac{5 * \text{Diff}_{[e]}}{16} \end{aligned}$$

And re-writing this equation with delta and q, consistent with the ’841 Patent:

$$\Delta_{q1} = \frac{5(q2 - 2q1 + q0)}{16} - \frac{5\Delta_{q0}}{16}$$

297. However, like the second offset, similar equations had been used in the prior art, which differed only as to: (1) the specific high-pass filter component used; and (2) the weights assigned to each component.

298. For instance, Fu discloses the following filter equation incorporating a third offset:

$$Q_1 = \text{Clip1} (q_1 - \text{Clip3} (-C, C ((q_1 - Q_0) \times 3 + (P_0 - q_2) + 4) >> 3))$$

Fu, 8:18. Just like with the P1 equation, the Q1 equation can be written as:

$$\begin{aligned}
 Q_1 &= q_1 - \{[3(q_1 - Q_0) + (P_0 - q_2) + 4] \gg 3\} \\
 &\text{where } P_0 = p_0 + \Delta \text{ (the first offset)} \\
 Q_0 &= q_0 - \Delta \\
 &= q_1 - \{[3(q_1 - q_0 + \Delta) + (p_0 + \Delta - q_2) + 4] \gg 3\} \\
 &= q_1 - (p_0 - 3q_0 + 3q_1 - q_2 + 4\Delta + 4) \gg 3 \\
 &= q_1 - \frac{p_0 - 3q_0 + 3q_1 - q_2 + 4\Delta}{8} \\
 &= q_1 - \frac{p_0 - 3q_0 + 3q_1 - q_2}{8} + \frac{\Delta}{2} \\
 &= q_1 + \frac{q_2 - 3q_1 + 3q_0 - p_0}{8} - \frac{\Delta}{2}
 \end{aligned}$$

299. Similarly, the JVT proposed a third-offset equation, B118, 48 (“ $Q_1 = q_1 + (q_2 + Q_0 - q_1 \ll 1) \gg 1$ ”), that combined a scaled Laplacian high-pass filter into the offset with “ $\Delta/2$ ”:

$$\begin{aligned}
 Q_1 &= q_1 + (q_2 + Q_0 - q_1 \ll 1) \gg 1 \\
 &\text{where } Q_0 = q_0 - \Delta \text{ (the first offset)} \\
 &= q_1 + (q_2 + q_0 - \Delta - q_1 \ll 1) \gg 1 \\
 &= q_1 + \frac{q_2 - 2q_1 + q_0 - \Delta}{2} \\
 &= q_1 + \frac{q_2 - 2q_1 + q_0}{2} - \frac{\Delta}{2}
 \end{aligned}$$

300. As I discussed previously, in my opinion, a POSITA would have been motivated to experiment with the weights applied to each of the high-pass

component and first-offset component knowing that they controlled how much each component contributed to the second offset, and would have found applying a weight of $\frac{1}{4}$ to the high-pass Laplacian component, and $\frac{1}{2}$ to the first-offset component obvious. *See* Section X.A.6.d (element 10[d]).

301. A POSITA would have also understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies third-offset calculations that would require a processor serving as a third-offset calculator, Ex-1012, 11 (providing code for the third-offset calculator using "id=(5*(li[4]-2*li[5]+li[6])/16" (omitting pointers)); Ex-1001, 23:13-21 (noting the third-offset calculator can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a general processor programmed or a dedicated processor with a "third offset calculator" to calculate the third offset for pixel f in neighboring block 2.

- g. Element 10[g]: a fourth pixel modifier configured to modify, if said second filter decision value is below said second threshold as determined by said second comparator, said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels by adding said third offset calculated by said third offset calculator to said pixel value of said pixel in said neighboring block**

next closest to said block boundary in said corresponding first line of pixels.

302. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 10[g].

303. Specifically, Bjontegaard discloses filtering a fourth pixel f , which is the second-closest pixel from the block boundary in a neighboring block 2, when the decision value is less than a threshold. *See* Sections X.A.2.f-X.A.2.h (element 3[f]-[h]).

304. Bjontegaard also discloses " $f_{\text{modified}}=f+\text{Diff}$," meaning that the second pixel value f is modified by adding the third offset Diff . Bjontegaard, 20.

305. A POSITA would have also understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]), and Bjontegaard's Code expressly specifies fourth-pixel modifications that would require a processor serving as a fourth-pixel modifier, Ex-1012, 11 (providing code for pixel modifications of $li[5]$ (pixel (f)) using " $ir2=li[5]+id$ " ($f_{\text{modified}} = f + \text{Diff}$), before outputting the final modified pixel value using " $lu[5]=\max(0,\min(255,ir2))$ " to clip the output modified pixel value between 0 and 255 (omitting pointers)); Ex-1001, 23:13-21 (noting the pixel modifier can be implemented on a processor). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard's coding method would have had a

general processor programmed or a dedicated processor with a “fourth pixel modifier” to modify the value of pixel f second closest to the boundary in neighboring block 2.

7. Claim 12

a. Element 12[a]: An encoder comprising a filtering control device according to claim 5.

306. In my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 12[a].

307. Specifically, Bjontegaard discloses a “coding method by mainly defining the decoder actions.” Bjontegaard, 4. However, Bjontegaard’s Code also provided the deblocking filter in an “H.26L encoder.” Ex-1014, 1, 6-8, 10-13. Thus, a POSITA would have understood that Bjontegaard’s deblocking filter was intended to be implemented in an encoder. Bjontegaard also discloses a “filtering control device” (*e.g.*, deblocking filter) as recited in claim 5, for the reasons discussed above. *See* Section X.A.3.

308. A POSITA would have also understood that it was extremely old and well known in the art to build deblocking filters into an encoder to eliminate blocking artifacts during the image-prediction-encoding process. *See, e.g.*, Ex-1040, 171 (“[t]he deblocking filter is applied after the inverse transform *in the encoder* before reconstructing and storing the macroblock for future predictions.”) (emphasis added); Ex-1061 at 3:20-31 (“deblocking component 125 can perform low pass filtering

and/or edge enhancement in encoder device 105.”), 18:14-18 (“To eliminate these artifacts, the encoder applies a loop filter to a reference frame before that frame is utilized in motion estimation operations. In some embodiments, a deblocking filter is utilized for loop filtering.”); Ex-1062, 18:21-23 (“either the decoder or both the decoder and encoder may use deblocking filters.”). In other words, because an encoder itself will reconstruct (decode) already-encoded blocks to use them as predictions to encode subsequent blocks, if blocking artifacts are present in the reconstructed blocks, that may result in less-accurate predictions for subsequent blocks. That is why those in the art would remove blocking artifacts in the encoder, to improve the accuracy of the predictive encoding.

309. For these reasons, even if Bjontegaard does not disclose a deblocking filter in an encoder, in my opinion, a POSITA would have been motivated to make this conventional modification and build Bjontegaard’s deblocking filter into an encoder to improve the accuracy of the encoder.

8. Claim 13

a. Element 13[pre]: A user equipment comprising:

310. To the extent the preamble is limiting, in my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 13[pre].

311. Specifically, Bjontegaard discloses a “coding method by mainly defining the decoder actions.” Bjontegaard, 4. However, Bjontegaard’s Code was

intended to operate on a computer, Ex-1010-1015, which the '841 Patent considers “user equipment.” *See, e.g.*, Ex-1001, 24:1-2 (“[t]he computer 70 of FIG. 12 can be a user equipment or be present in a user equipment.”); *see also id.*, 25:27-30 (“The user equipment 80 can be any device having media decoding functions.”).

312. Moreover, a POSITA would have also understood that user equipment was an extremely old, well-known, and conventional mechanism by which deblocking filters were implemented, because it is user equipment that is often used to record and play back video. *See, e.g.*, Ex-1040, 1-5 (noting video encoding is essential to digital television broadcasting, Internet streaming, mobile video streaming, physical media such as DVDs, and video calling, and applicable to relevant devices such as computers, mobile phones, and televisions), 247 (noting file formats to store encoded video on a computer); Ex-1061, 2:56-62 (describing encoder built into “a general communications system”), 9:49-57 (“[a]n electronic device for deblocking and edge enhancement”), 10:9-13 (“Those of ordinary skill would further appreciate that the various illustrative logical blocks, modules, and algorithm steps ... may be implemented as electronic hardware, computer software, or combinations of both.”); Ex-1062, 18:17-18 (“a deblocking filter is utilized for loop filtering), 22:23-24 (providing “a computer system with which some embodiments of the invention is implemented.”).

313. For this reason, even if Bjontegaard does not expressly disclose a deblocking filter in user equipment, a POSITA would have known that Bjontegaard was intended to be implemented in user equipment, and would have been motivated to implement Bjontegaard accordingly.

b. Element 13[a]: a memory configured to store video frames; and

314. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 13[a].

315. Specifically, Bjontegaard discloses a "coding method by mainly defining the decoder actions." Bjontegaard, 4. Bjontegaard's Code also not only requires memory to operate (to store the encoder's files), but also to store video frames. Ex-1013, 2 (noting how the encoder reads an "Input YUV file," a well-known video format for representation of video frames, *see, e.g.*, Ex-1040, 257-59, and outputs a "H.26L bitstream").

316. A POSITA would have also understood that encoding and decoding video frames requires memory. *See, e.g.*, Ex-1040, 84 (showing prediction reference frames stored in a "Picture Buffer," which is a type of memory); Ex-1061, 7:33-44; Ex-1062, 10:56-59 ("The encoder encodes any frames that it receives and then stores them on the storage, which could be computer memory, hard disk, DVD, or similar media.")). This is because, in encoders, an encoded video frame is decoded (or

reconstructed) and stored in memory to serve as a reference frame to help encode subsequent frames. *See* Section V.A (describing encoding process, generally); *see also* Bjontegaard, 13 (§3.5 noting the use of “[r]eference frame” for prediction); Ex-1040, 84.

317. For this reason, even if Bjontegaard does not expressly disclose memory to store video frames, a POSITA would have known that Bjontegaard was intended to be implemented in user equipment which would require memory to perform any encoding, and would have been motivated to provide memory accordingly.

c. Element 13[b]: an encoder according to claim 12 configured to encode said video frames into encoded video frames, wherein said memory is further configured to store said encoded video frames.

318. In my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 13[b]. *See* Sections X.A.6 (claim 12), X.A.8.b (element 13[a]).

319. Specifically, Bjontegaard discloses a “coding method by mainly defining the decoder actions.” Bjontegaard, 4. Bjontegaard’s Code also outputs the encoded video frames for storage in memory. Ex-1013, 2 (noting how the encoder reads an “Input YUV file” and outputs a “H.26L bitstream”).

320. A POSITA would have also known that it was extremely conventional to store encoded video frames locally in the encoding device. *See, e.g.*, Ex-1040, 3

(describing encoding, transmission, and decoding operations), 247 (“It is common for single compressed video sequences to be stored in files.”); Ex-1062, 10:56-59 (“The encoder encodes any frames that it receives and then stores them on the storage, which could be computer memory, hard disk, DVD, or similar media.”); Ex-1066, ¶ 30 (“The output video bitstream is then stored or transmitted through a communication channel.”). In fact, video is conventionally stored in memory in its encoded form in order to significantly reduce the space needed to store the video.

321. For this reason, even if Bjontegaard did not expressly disclose storing encoded video frames in memory, in my opinion, a POSITA would have been motivated to configure Bjontegaard to store encoded video frames in memory because that was conventionally done in the art.

9. Claim 14

a. Element 14[a]: A decoder comprising a filtering control device according to claim 5.

322. In my opinion, Bjontegaard teaches or suggests element 14[a].

323. Specifically, Bjontegaard discloses a “coding method by mainly defining the decoder actions.” Bjontegaard, 4. As part of the “decoder actions,” Bjontegaard discloses a filtering control device (*e.g.*, a deblocking filter). Bjontegaard, 18 (§4.5). Bjontegaard’s Code also includes an implementation of Bjontegaard’s deblocking filter in a decoder. *See, e.g.*, Ex-1011, 1 (“Main program for H.26L decoder”).

324. As I explained above, Bjontegaard also discloses the filtering control device of claim 5. *See* Section X.A.3 (claim 5).

10. Claim 15

a. Element 15[pre]: A user equipment comprising:

325. If the preamble is limiting, in my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 15[pre]. *See* Section X.A.8.a (element 13[pre]). A POSITA would have understood generally that Bjontegaard was intended to be implemented on user equipment (*e.g.*, a computer), *see id.*, and Bjontegaard's Code was intended to operate on user equipment (*e.g.*, a computer), *see, e.g.*, Ex-1010; Ex-1015, 1.

b. Element 15[a]: a memory configured to store encoded video frames;

326. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 15[a]. *See* Section X.A.8.b (element 13[a]).

327. Specifically, Bjontegaard discloses a “coding method by mainly defining the decoder actions.” Bjontegaard, 4. Bjontegaard's Code also included an “H.26L decoder” that instructed the decoder to receive as “input [an] H.26L bitstream file”—or encoded video frames—stored in memory. Ex-1011, 1, 3.

328. A POSITA would have known that it was extremely conventional to store encoded video frames locally in the decoding device prior to decoding. *See, e.g.*, Ex-1040, 3 (describing encoding, transmission, and decoding operations), 247

(“It is common for single compressed video sequences to be stored in files.”); Ex-1062, 10:56-59 (“The encoder encodes any frames that it receives and then stores them on the storage, which could be computer memory, hard disk, DVD, or similar media.”); Ex-1066, ¶ 30 (“The output video bitstream is then stored or transmitted through a communication channel ... [and] contains enough information for a reconstruction of the input video frame”). In fact, video is conventionally stored in memory in its encoded form in order to significantly reduce the space needed to store the video prior to decoding. Ex-1040, 2 (“video encoding is the process of reducing the amount of data required to represent a digital video, prior to transmission or storage. The complementary operation, decompression or decoding, recovers a digital video signal from a compressed representation....”).

329. A POSITA would have appreciated that Bjontegaard is either storing the encoded bitstream locally (to be subsequently used by a local decoder), or transmitting the encoded bitstream, which is then stored by a separate decoder device, as this was conventional. *See* Ex-1040, 1-5.

c. Element 15[b]: a decoder according to claim 14 configured to decode said encoded video frames into decoded video frames; and

330. In my opinion, Bjontegaard teaches or suggests element 15[b]. *See* Section X.A.9 (claim 14). Bjontegaard generally provides a process to decode encoded video frames into decoded video frames. *See generally* Bjontegaard.

Bjontegaard's Code also decoded encoded video frames into decoded video frames. *See* Ex-1011, 1 (noting "input H.26L bitstream file" and "output YUV file").

d. Element 15[c]: a media player configured to render said decoded video frames into video data displayable on a display.

331. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 15[c].

332. As I previously explained, in my opinion, a POSITA would have understood that Bjontegaard's coding method would have been built into video-capable devices, such as user equipment, in order to encode and decode video. *See* Section X.A.8.a (element 13[pre]).

333. Bjontegaard's Code also included code to decode video-image data. Ex-1011, 1 (noting "input H.26L bitstream file" and "output YUV file"); *see generally* Ex-1012. For this reason, in my opinion, Bjontegaard teaches or suggests a "media player" because Bjontegaard provides computer code to decode video images to display on a computer.

334. However, a POSITA would have also known that the use of a media player to implement video decoders was extremely old and well known in the art. For instance, video decoders are essential for dedicated media players such as DVD players and television set-top boxes. *See* Ex-1040, 1-5. Moreover, software-based media players are also essential to multi-purpose computing devices. At the time of

the '841 Patent, Windows Media Player had already existed for decades, which incorporated “various video codecs ... including MPEG1 ... [and] an unofficial variation of MPEG4.” Ex-1055. Another well-known media player that also dated back decades is “VLC,” which supported many different video codecs. Ex-1071.

335. For this reason, even if Bjontegaard does not expressly disclose a “media player” separate from the decoder, a POSITA would have known to build Bjontegaard’s decoder into a media player to enable playback of the decoded video on a display. Moreover, a POSITA would have been motivated to modify Bjontegaard to build the decoder Bjontegaard describes into a media player to support playback of videos decoded by either Bjontegaard’s method alongside other decoding codecs.

11. Claim 16

- a. **Element 16 [pre]: A non-transitory computer-readable medium comprising, stored thereupon, a computer program for filtering control of a block of multiple pixels in a video frame each pixel having a respective pixel value, said computer program comprising code means which when run on a computer causes the computer to:**

336. If the preamble is limiting, in my opinion, Bjontegaard in view of Bjontegaard’s Code teaches or suggests element 16[pre].

337. Specifically, Bjontegaard discloses that it is describing a “coding method by mainly defining the decoder actions,” Bjontegaard, 4, which includes a

method for filtering control, *see, e.g.*, Bjontegaard, 19. Bjontegaard's Code contained computer files (including the code and execution files) by which to operate the coding model on a computer, which needs to be stored on a computer-readable medium in order to operate. Ex-1010.

338. A POSITA would have also understood that a coding system such as Bjontegaard's would be stored on "non-transitory computer-readable medium" with a program to operate it because video codecs are conventionally run on such devices. *See, e.g.*, Ex-1061, 9:58-66 ("A computer-readable medium embodying means for causing a computer to execute a method for deblocking and edge enhancement of block based digital video...."); Ex-1062, cl. 26 ("A computer readable medium storing a computer program for encoding an image ...").

339. For this reason, a POSITA would have understood that Bjontegaard was intended to be implemented in such a system, and would have been motivated to implement Bjontegaard accordingly.

- b. Element 16[a]: calculate a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $|p_{2i}-2p_{1i}+p_{0i}|$ wherein p_{0i} , denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel**

second next closest to, in said first line of pixels, said block boundary;

340. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[a]. *See* Section X.A.1.b (element 1[a]).

- c. **Element 16[b]: calculate a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $|q_{2i}-2q_{1i}+q_{0i}|$ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;**

341. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[b]. *See* Section X.A.1.c (element 1[b]).

- d. **Element 16[c]: determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value; and**

342. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[c]. *See* Section X.C.1.d (element 1[c]).

- e. **Element 16[d]: determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value,**

343. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[d]. *See* Section X.A.1.d (element 1[c]).

- f. Element 16[e]: wherein determine how many pixels in said first line of pixels in said block to filter comprises determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and**

344. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[e]. *See* Sections X.A.1.d (element 1[c]), X.A.2.b (element 3[b]).

- g. Element 16[f]: determine how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.**

345. In my opinion, Bjontegaard in view of Bjontegaard's Code teaches or suggests element 16[f]. *See* Sections X.A.1.e (element 1[d]), X.A.2.f (element 3[f]).

B. Ground 1B: Obviousness over Bjontegaard in view of H.241 and B205

1. Claim 2

346. I understand that in the co-pending ITC proceedings, the respondents contend that there may be a contradiction between claims 1 and 2, because claim 1 requires using “only” three pixels in one “first” line of pixels to make a filter decision, but claim 2 requires six pixels from two lines (three pixels each) to make a filter decision.

347. I do not provide any opinion on the definiteness of this claim, but I have instead assumed that claim 1 does not foreclose claiming multiple lines of pixels when applying the prior art.

- a. **Element 2[a]: The method according to claim 1, wherein calculating said first filter decision value comprises calculating said first filter decision value as: $|p_{2_0}-2p_{1_0}+p_{0_0}|+|p_{2_3}-2p_{1_3}+p_{0_3}|$ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{2_3} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and**

348. In my opinion, Bjontegaard in view of Bjontegaard's Code, H.241 and B205 teaches or suggests element 2[a].

349. Specifically, Bjontegaard discloses calculating a filter-decision value using pixels from a single line in the block. Bjontegaard, 19. Bjontegaard also expressly discloses performing deblocking "on a 4x4 block structure." *Id.*, 18. Because Bjontegaard does not specify that the filter-decision values are calculated once, and represent the entire block, in my opinion, Bjontegaard discloses calculating filter-decision value S_{0_1} line-by-line because if Bjontegaard had used

specific lines in a block to calculate $S0_1$, a POSITA would have expected Bjontegaard to have specified that.

350. **First**, in my opinion, a POSITA would have appreciated that rather than calculate filter decisions line-by-line for an entire block, it is much more efficient to filter an entire block based on a decision made using some representative number of lines from the block. Both H.241 and B205 follow this approach, using multiple lines (2 and 5) from a block as part of the filter-decision calculation to make a decision that applies to all lines in the block:

$$d = \left| \begin{array}{c} p_{2_2} - 2*p_{1_2} + p_{0_2} \\ q_{0_5} - 2*q_{1_5} + q_{2_5} \end{array} \right| + \left| \begin{array}{c} q_{0_2} - 2*q_{1_2} + q_{2_2} \\ p_{2_5} - 2*p_{1_5} + p_{0_5} \end{array} \right| +$$

H.241, 29; B205, 130; Ex-1001, 2:5-27 (same). In fact, a POSITA would have recognized that making filter decisions using a representative set of lines significantly increases computational efficiency, Ex-1053, 27-28 (calculating boundary strength measurement for a deblocking filter using line 0 to represent lines 0-3, line 4 to represent lines 4-7, and so on, thus “decreas[ing] nearly $\frac{3}{4}$ complexity” in these computations “without any video quality loss.”); Ex-1056, 3 (alternating lines to reduce computational demands), and using two lines for a block can serve as a sufficiently representative sample for the entire block, *see, e.g.*, Ex-1059, 1693 (“To limit the number of operations, the measure d is only calculated for two of the eight rows/columns that are orthogonal to the edge ... as opposed to calculating the

measure for each edge line separately.”); Ex-1065 ¶¶44 (same). As Ugur explains, using two lines as representative of the block “is motivated by the fact that the activity between edge lines varies less for high resolution sequences than for lower resolutions. Consequently, significant complexity reduction is achieved with no observable visual degradation at high resolutions.” See Ex-1059, 1693. Selecting either line-by-line analysis, or some smaller number of lines to represent the block were both well-understood choices to a POSITA, with the choice being a balance of accuracy (line-by-line) and computational efficiency (representative lines). See, e.g., Ex-1065 ¶¶41-44 (describing calculating filter decisions line-by-line or two lines as representative as alternative embodiments). Moreover, a POSITA would have also appreciated that using two lines as representative instead of just one reduces the risk of erroneous filter decisions, because a single line may contain errant noise or a feature not representative of the entire block.

351. A POSITA would have further appreciated that making filter decisions, including *whether* (like H.241/B205) and *how many* pixels to filter (like Bjontegaard) by using certain lines as representative of others, would provide efficiency gains. The considerations are the same: balancing the computational complexity of making different filter decisions to control filtering (including *whether* to filter and *how many* pixels to filter) versus the accuracy of when and how much the deblocking filter should be applied.

352. For this reason, in my opinion, a POSITA would have been motivated to modify Bjontegaard's line-by-line calculation of $S0_1$ to instead use two lines to calculate $S0_1$ for block 1 (and $S0_2$ for neighboring block 2), and use that value as representative of the entire block to reduce the computational burdens of calculating these decision values. In my opinion, this modification simply applies a known technique (using a smaller number of lines, *e.g.*, two, to represent a block) to a known device (deblocking filter) ready for improvement (computational efficiency) to yield predictable results (a more efficient deblocking decision for an entire block). *See* Ex-1053, 27-28; Ex-1056, 3; Ex-1059, 1693.

353. A POSITA would have understood that the weight applied to $S0_1$ is intended to bring the Laplacian used for $S0_1$ to the same order of magnitude as the threshold $\text{Lim}(qp)$. Bjontegaard does this so that several different equations ($S0$, $S0_1$, $S1_1$, $S0_2$, and $S1_2$)—which apply different coefficients to the pixel values in each calculation—can all be compared to the same threshold $\text{Lim}(qp)$. *See, e.g.*, Bjontegaard, 19 (comparing both $12\text{abs}(a-2b+c)$ and $\text{abs}(4(a-d)+12(c-b))+\text{abs}(9(a-b-c+d))$ to the same threshold $\text{Lim}(qp)$). However, comparing $12(a-2b+c) < \text{Lim}(qp)$ is mathematically the same as $(a-2b+c) < \text{Lim}(qp)/12$. There is no mathematical difference, and it is just a choice whether to scale the calculation or the threshold. In fact, both H.241 and B205 just use a Laplacian with no additional weight, and compare it to a lower set of thresholds. *See, e.g.*, H.241, 29; B205, 130.

354. Specifically, Bjontegaard uses the following table to convert the quantization parameter (which can range from 0-31 for Bjontegaard) to $\text{Lim}(\text{qp})$. The lowest threshold (at $\text{QP}=0$) begins at 7, and increases up to the highest threshold (at $\text{QP}=31$) of 248:

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Lim	7	8	9	10	11	12	14	15	17	20	22	25	28	31	35	39	44	49	55	62	69	78	88	98	110	124	139	156	175	197	221	248

H.241 similarly uses the following table, but has up to 51 “index” values based on QP. The lowest threshold (at $\text{Index}=0$) begins at 0, and increases up to the highest threshold (at $\text{Index}=51$) of 64. H.241, 27-28 (“The definitions of ... indexB are as specified in Equations ... 8-464/H.264”); Ex-1036, 190 (H.264 equation 8-464 for $\text{Index B} = \text{Clip3}(0, 51, \text{qP}_{\text{av}} + \text{FilterOffsetB})$):

index	0-15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
t_c	0	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3
β	0	6	7	8	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28

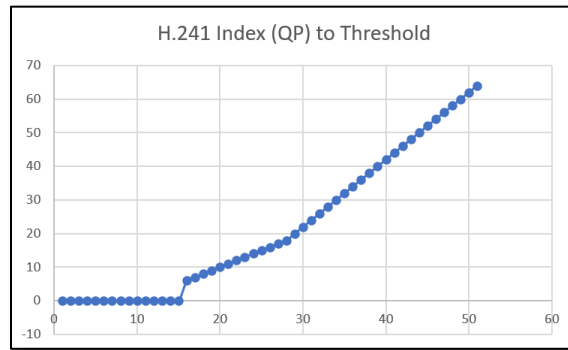
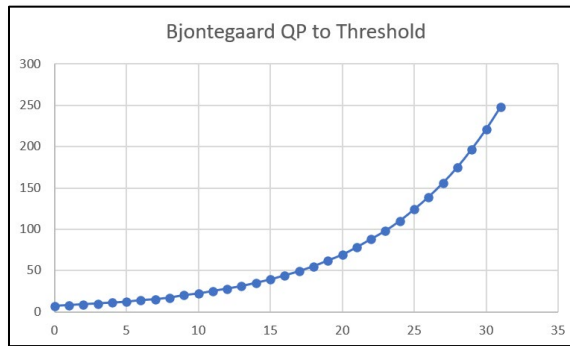
To obtain t_c , use $\text{index} = \text{indexA}$. To obtain β , use $\text{index} = \text{indexB}$.

Table B.3/H.241 (concluded) – Value of t_c and β as a function of index

index	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
t_c	3	4	4	4	5	5	6	6	7	8	9	9	11	12	13	13	16	18
β	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64

To obtain t_c , use $\text{index} = \text{indexA}$. To obtain β , use $\text{index} = \text{indexB}$.

When these are plotted, the two follow similar trends:



The biggest difference between the two is that—for blocks with moderate quantization applied—a slightly higher relative threshold is used by H.241 than Bjontegaard. B205 uses the same mapping as H.241, but uses the qp value directly rather than the intermediate “IndexB.” B205, 131.

355. A POSITA would have also appreciated that, when using pixels from multiple lines in a block to make filter decisions, it would be slightly more computationally efficient to divide the threshold once $(\text{Lim}(\text{qp})/12)$ rather than multiply each separate Laplacian calculation for each line $(12*(a_1-2b_1+c_1)$ and $12*(a_2-2b_2+c_2))$. This is consistent with H.241 and B205, which both use lower pre-defined thresholds when comparing to an unweighted Laplacian calculation. H.241, 29; B205, 130. The '841 Patent recognized that whether the Laplacian is weighted or not is not critical, as either could be used. *See* Ex-1001, 12: 30-58 (disclosing one embodiment without weights, and one with weights, applied to the Laplacian). In my opinion, a POSITA therefore would have preferred not to apply a weight to $S0_1$ when using multiple lines of pixels to make filter-decisions.

356. **Second**, in my opinion, a POSITA would have known that the selection of *which* lines to choose as representative is merely a design choice. H.241 and B205 appeared to select lines 2 and 5 because they provide equidistant spacing in an 8x8 block:

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃
p3 ₄	p2 ₄	p1 ₄	p0 ₄	q0 ₄	q1 ₄	q2 ₄	q3 ₄
p3 ₅	p2 ₅	p1 ₅	p0 ₅	q0 ₅	q1 ₅	q2 ₅	q3 ₅
p3 ₆	p2 ₆	p1 ₆	p0 ₆	q0 ₆	q1 ₆	q2 ₆	q3 ₆
p3 ₇	p2 ₇	p1 ₇	p0 ₇	q0 ₇	q1 ₇	q2 ₇	q3 ₇

But others in the art used other lines as representative, *see, e.g.*, Ex-1053, 27-28 (using lines 0, 4, 8 and 12 as representative, depending on block size); Ex-1056, 3 (alternating lines), and even the '841 Patent suggests the selection of *which* lines is a design choice. *See* Ex-1001, 10:14-16 (lines 2 and 5), 11:14-17 (lines 3 and 4), 11:61-64 (lines 0 and 3), 12:5-11 (lines 4 and 7). What is common throughout the prior art, however, is that the selection of lines maintained spacing between those that were selected. Ex-1053, 27-28; Ex-1056, 3; Ex-1065, ¶44. As Ugur explains, especially with higher-resolution images, spatially closer pixels are expected to be similar, and thus a POSITA would not want to select two neighboring lines to represent an entire block. Ex-1059, 1693. But, in my opinion, the exact lines that are

selected makes little, if any, difference, because blocks generally have different image features in different locations within each block; selecting specific lines requires assuming that those lines will be sufficiently representative of any block.

357. Because Bjontegaard discloses that deblocking is performed on 4x4 blocks, a POSITA would have known that lines 2 and 5 could not be selected (like H.241 and B205) because there would be no line 5. Instead, there were only three two-line combinations to choose from that maintained spacing between the lines selected: 0-2, 0-3, or 1-3, as shown below:

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀	p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁	p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂	p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃	p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃

p3 ₀	p2 ₀	p1 ₀	p0 ₀	q0 ₀	q1 ₀	q2 ₀	q3 ₀
p3 ₁	p2 ₁	p1 ₁	p0 ₁	q0 ₁	q1 ₁	q2 ₁	q3 ₁
p3 ₂	p2 ₂	p1 ₂	p0 ₂	q0 ₂	q1 ₂	q2 ₂	q3 ₂
p3 ₃	p2 ₃	p1 ₃	p0 ₃	q0 ₃	q1 ₃	q2 ₃	q3 ₃

For these reasons, a POSITA would have considered choosing lines 0 and 3 an obvious design choice because it maximized the distance between the lines used as representative of the block (and maintained a two-line spacing commonly used for 8x8 blocks, as taught by H.241 and B205). But, for the reasons I discussed above, a POSITA would have generally been motivated to experiment with any of these

configurations for other-sized blocks to empirically test which provided the most preferred deblocking results.

358. In my opinion, therefore, a POSITA would have been motivated not only to modify Bjontegaard to calculate $S0_1$ (and $S0_2$) using two lines from a block to make a filter decision for the entire block, but also to select lines 0 and 3 for the calculation as a design choice because it was one of a very limited number of options to choose from.

359. A POSITA would have had a reasonable expectation of success of using a Laplacian on two lines to calculate $S0_1$ (and $S0_2$) because the prior art had already established that using two lines as representative of a block would work (Ex-1053, 27-28; Ex-1056, 3; Ex-1059, 1693), including when using a Laplacian to measure activity (*e.g.*, H.241 and B205). And a POSITA would have also had a reasonable expectation of success in choosing lines 0 and 3 for the filter decision because they maintain the same two-line spacing as lines 2 and 5 do for 8x8 blocks used in prior-art deblocking filters, and would require minimal modification to some computer code to incorporate pixels from these lines into the d_r (and d_l) equation(s).

- b. Element 2[b]: calculating said second filter decision value comprises calculating said second filter decision value as: $|q2_0 - 2q1_0 + q0_0| + |q2_3 - 2q1_3 + q0_3|$ wherein $q0_0$ denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, $q1_0$ denotes said pixel value of said pixel of said neighboring block**

next closest to, in said corresponding first line of pixels, said block boundary, q_{2_0} denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary, q_{0_3} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, q_{1_3} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and q_{2_3} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.

360. In my opinion, Bjontegaard in view of Bjontegaard's Code, H.241, and B205 teaches or suggests element 2[b]. Specifically, selecting two lines, as well as lines 0 and 3, for evaluation as representative of an entire block would have been an obvious design choice, *see* Section X.B.1.a (element 2[a]), and the rationale for choosing these lines for block 1 apply equally to block 2.

2. Claim 6

361. I understand that in the co-pending ITC proceedings, the respondents contend that there may be a contradiction between claims 5 and 6, because claim 5 requires using "only" three pixels in one "first" line of pixels to make a filter decision, but claim 6 requires six pixels from two lines (three pixels each) to make a filter decision.

362. I do not provide any opinion on the definiteness of this claim, but I have instead assumed that claim 1 does not foreclose claiming multiple lines of pixels when applying the prior art.

- a. **Element 6[a]: The device according to claim 5, wherein said first decision value calculator is configured to calculate said first filter decision value as: $|p_{2_0}-2p_{1_0}+p_{0_0}|+|p_{2_3}-2p_{1_3}+p_{0_3}|$ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{2_3} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and**

363. In my opinion, Bjontegaard in view of Bjontegaard's Code, H.241, and B205 teaches or suggests element 6[a]. *See* Sections X.B.1.a (element 2[a]), X.A.3.b (element 5[a], first calculator).

- b. **Element 6[b]: said second decision value calculator is configured to calculate said second filter decision value as: $|q_{2_0}-2q_{1_0}+q_{0_0}|+|q_{2_3}-2q_{1_3}+q_{0_3}|$ wherein q_{0_0} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{1_0} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{2_0} denotes said pixel value of said pixel in said neighboring block second next closest to, in said**

corresponding first line of pixels, said block boundary, $q0_3$ denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, $q1_3$ denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and $q2_3$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.

364. In my opinion, Bjontegaard in view of H.241 and B205 teaches or suggests element 6[b]. *See* Sections X.B.1.a-X.B.1.b (element 2[a]-[b]), X.A.3.c (element 5[b], second calculator).

3. Claim 11

- a. Element 11[a]: The device according to claim 5, further comprising a third comparator configured to compare a sum of said first filter decision value and said second filter decision value to a threshold value, wherein**

365. Bjontegaard in view of Bjontegaard's Code, H.241, and B205, teaches or suggests element 11[a].

366. Specifically, as part of the deblocking-filter design, Bjontegaard first decides whether to perform deblocking or not. Bjontegaard, 19. Bjontegaard makes this decision by evaluating a series of "Strength" conditions based on specific properties of the block, but does not evaluate the pixel values in the block itself for this decision. *Id.* (describing procedure to determine "Strength," and that "[f]iltering takes place only if [$Strength_1$ and $Strength_2$] are both different from 0"). As such,

Bjontegaard does not disclose comparing the sum of first and second filter-decision values to a threshold value to determine whether or not to filter any pixels (as required by subsequent claim elements, discussed below).

367. In my opinion, however, a POSITA would have been motivated to sum the first and second filter decisions (S_{01} and S_{02}), and compare this sum to a threshold to decide whether to perform deblocking filtering or not. A POSITA would have recognized that H.241 and B205 disclose summing a series of Laplacian calculations—just like into Bjontegaard’s S_{01} and S_{02} calculations—to decide whether or not to perform deblocking filtering:

$$d = \left| p_{22} - 2*p_{12} + p_{02} \right| + \left| q_{02} - 2*q_{12} + q_{22} \right| + \left| p_{25} - 2*p_{15} + p_{05} \right| + \left| q_{05} - 2*q_{15} + q_{25} \right|$$

H.241, 29; B205, 130; Ex-1001, 2:5-27 (same). In fact, H.241 and B205 disclose the same “Strength” conditions Bjontegaard discloses to decide *whether* to filter a block, but *also* includes the equation above as an independent condition for their deblocking decision. *Compare* H.241, 29 (determining whether filtering is performed depending on whether a block “is intra coded,” “contain[] non-zero transform coefficients,” and “absolute difference between ... motion vectors” by setting a “filterBlockEdge” flag to 0 or 1); B205, 130 (same), *with* Bjontegaard, 19 (same); *see also* H.241, 29 (“[if] *one or more* of the following conditions are true, no filtering across the block edge takes place $d \geq \beta$... [*or*] filterBlockEdge =

0.”); B205, 130 (setting forth same conditions using “or,” but the calculation of comparison of $d < \beta$ using “and”).

368. For this reason, in my opinion, a POSITA would have been motivated to sum Bjontegaard’s S_{01} and S_{02} , and compare that sum to a threshold (*e.g.*, $\text{Lim}(\text{qp})$) as an additional condition when deciding whether to perform deblocking filtering or not, as both H.241 and B205 teach, because that would enable filter decisions to be made based not only on the “Strength” conditions that do not consider the pixel values near the block boundary themselves, but also on the activity of the pixels near the block boundary.

369. In my opinion, a POSITA would have had a reasonable expectation of success in making this modification because both H.241 and B205 had already demonstrated that including an independent decision condition based on the value of “d” would work, and implementing it would have simply involved some computer coding to incorporate the additional equation.

370. Moreover, a POSITA would have understood generally that Bjontegaard was intended to be run on a processing device (*e.g.*, a computer processor), *see* Section X.A.3.a (element 5[pre]). Accordingly, a POSITA would have understood that a processing device implementing Bjontegaard’s coding method, as modified by H.241 and B205, would have had a general processor programmed or a dedicated processor with “a third comparator” to compare a sum

of said first filter decision value and said second filter decision value to a threshold value, as H.241 and B205 teach.

- b. Element 11[b]: said first pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and**

371. In my opinion, Bjontegaard in view of Bjontegaard's Code, H.241, and B205, teaches or suggests element 11[b]. *See* Sections X.A.1.d (element 1[c], determine how many pixels in block 1), X.A.3.b (element 5[a], first decision-value calculator), X.A.3.d (element 5[c], first pixel determiner), X.B.3.a (element 11[a], comparing sum using third comparator).

- c. Element 11[c]: said second pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator.**

372. In my opinion, Bjontegaard in view of Bjontegaard's Code, H.241, and B205, teaches or suggests element 11[c]. *See* Sections X.A.1.e (element 1[d], determine how many pixels in neighboring block 2), X.A.3.c (element 5[b], second

decision-value calculator), X.A.3.e (element 5[d], second pixel determiner), X.B.3.a (element 11[a], comparing sum using third comparator).

C. Ground 2A: Obviousness over Aksu in view of Gonzalez

373. If Lenovo's construction is correct, and the determination for how many pixels to filter in a given block can have "no mathematical basis" in the pixel values of any other block, then in my opinion, Aksu in view Gonzalez teaches every element of the claims.

1. Claim 1

- a. Element 1[pre]: A method for filtering control applicable to a block of multiple pixels in a video frame, each pixel having a respective pixel value, said method comprising:**

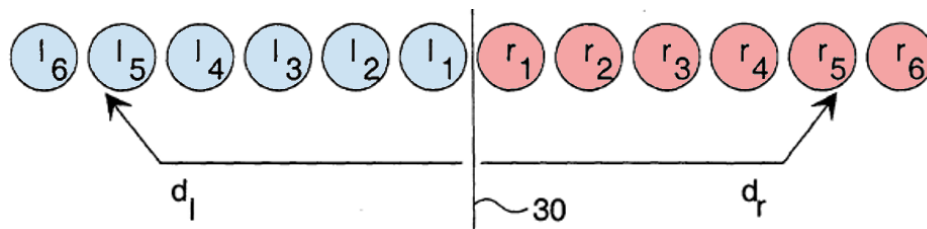
374. If the preamble is limiting, in my opinion, Aksu discloses or suggests element 1[pre].

375. Aksu discloses "a method for reducing visual artefacts in a frame of a digital video ... in which filtering is performed to reduce visual artefacts due to a block boundary." Aksu, 1:3-7; *see also id.*, Abstract (noting the method is applied "in a frame of a digital video signal, which is coded by blocks and then decoded").

376. Aksu discloses that "[t]he frame is divided into adjacent blocks having a size of *e.g.* 8 x 8 pixels." This means that each block has "multiple pixels" (*i.e.*, 64 in total).

377. Aksu also explains that each pixel is assigned a pixel value. *See, e.g.*, Aksu, 4:29-32 (explaining filtering is needed because “there is a discontinuity of pixel values at the boundary between two adjacent blocks”), 5:16-20 (explaining filtering depends on “the difference of the pixel values on different sides of the pixel being processed); 15:4-8 (“Filtering is used to determine a new value for the pixels selected for filtering.”); *see also id.*, 7:16-20, 8:1-6, 8:11-13, 13:10-13. But a POSITA would have also known generally that each of Aksu’s pixels has a pixel value because that is how pixels in digital video are stored as digital data and are used for processing pixel values. *See, e.g.*, Section V.B.1 (overview of pixel values).

378. Aksu also shows a line of pixels spanning two neighboring blocks (l_{1-6} and r_{1-6}):



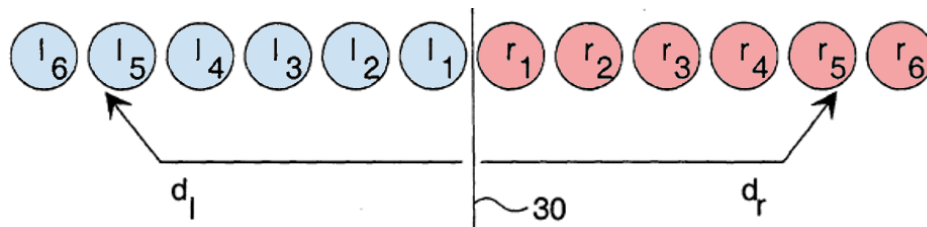
Aksu, Fig. 2 (coloring added).

- b. **Element 1[a]: calculating a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $|p_{2i}-2p_{1i}+p_{0i}|$ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel**

second next closest to, in said first line of pixels, said block boundary;

379. In my opinion, Aksu in view of Gonzalez teaches or suggests element 1[a].

380. Aksu discloses calculating a first filter-decision value “ d_r ” which represent[s] activity, or the difference of pixel values between pixels on one side of the block boundary.” Aksu, 13:10-19. As I mentioned in the overview of Aksu, *see* Section VII.B, to calculate “ d_r ,” Aksu discloses performing multiple difference calculations between “ r_1 ”—which is the pixel closest to the block boundary—and up to five additional pixels (r_{2-6}) further from the block boundary, and then comparing each calculation to a “threshold value” based on β by dividing β by an integer associated with the position of the pixel (*e.g.*, (1) $|r_1 - r_2| \leq \beta/2$; (2) $|r_1 - r_3| \leq \beta/3$; (3) $|r_1 - r_4| \leq \beta/4$). Aksu, 13:10-19. Aksu also shows, d_r is calculated using a first line of pixels in a first (r) block, which is adjacent to a neighboring (l) block of multiple pixels:



Aksu, Fig. 2 (coloring added).

381. Because Aksu implements a pre-decision process to determine how many pixels (between one and six) to evaluate in a given block as part of the filter-decision process to calculate d_r , Aksu discloses a condition in which Aksu will evaluate three pixels (r_1 , r_2 , and r_3 , corresponding to pixel values p_{0i} , p_{1i} , and p_{2i} in the claims) to calculate the value of d_r . *Id.*, 9:31-10:21. However, Aksu does not disclose that d_r is calculated “based on” the equation $|p_{2i}-2p_{1i}+p_{0i}|$.

382. Instead of using a series of “first-order” difference calculations to measure activity like Aksu discloses, in my opinion, a POSITA would have known that a “second-order” algorithm would provide a better approach to calculate the activity in a line of pixels, as taught by Gonzalez. Specifically, as Gonzalez recognized, “second-order” algorithms were superior to first-order difference calculations at detecting finer details, including points, lines, and edges, and with greater precision. Gonzalez, 161, 694-97. And, as Gonzalez (and many others in the art) had recognized, the Laplacian algorithm—which provides a “second-order” difference operator and used the coefficients $[1 \ -2 \ 1]$ applied to a set of three pixels—would be an ideal candidate to evaluate changes in pixel values in a region. Gonzalez, 693.

383. In fact, beyond just Gonzalez, the Laplacian was widely used in the image-processing arts to detect edges in an image because it was able to identify “regions of rapid intensity change.” *See, e.g.*, Ex-1049, 616; *see also* Ex-1028, 54,

57-58 (noting the Laplacian is useful for detecting intensity changes); Ex-1041, 149-54. Although edge-detection schemes often used a two-dimensional, 3x3 Laplacian kernel to create two-dimensional edge maps of images, Ex-1041, 155; Ex-1049, 616, as Gonzalez demonstrates, a one-dimensional Laplacian was known to identify regions of rapid change in pixel values outside the context of edge detectors. *See, e.g.,* Gonzalez, 693-97, Ex-1028, 58, Fig. 2-11(c), (d) (disclosing one-dimensional Laplacian); Ex-1041, 154 (Equation 5.24) (same). In my opinion, a one-dimensional form would have been preferred when the goal is simply to measure activity in a line of pixels (which exist in one dimension), as Aksu discloses is the goal, rather than create a two-dimensional edge map. It was well known to a POSITA that it is much more efficient in terms of computational resources to implement a two-dimensional separable operator (such as the Laplacian) using the composition of two one-dimensional operators in orthogonal directions (*e.g.,* horizontal and vertical). *See, e.g.,* Ex-1058, 6:14-26 (“[A]pplying ... a vertical one-dimensional filter and afterwards a second horizontal one-dimensional filter to the same region ... provides a two-dimensional filter effect, however, with a reduced complexity compared to an application of a two-dimensional filter.”); Ex-1049, 614-15 (noting use of one-dimensional derivatives for feature detection).

384. In my opinion, a POSITA would have been motivated to replace Aksu's iterative difference calculations of d_r with a one-dimensional Laplacian taught by Gonzalez for several reasons.

385. **First**, a POSITA would have understood that the one-dimensional Laplacian will identify *rapid* pixel-value changes with greater precision than a first-order difference calculation, while also not responding to gradual pixel value changes that are not considered true “details” in an image and can therefore be filtered without sacrificing image quality. *See* Gonzalez, 693-97 (explaining increased detection and accuracy of second-order calculations compared to first-order difference calculations); Section V.B.2 (Filters, explaining first- and second-order derivatives).

386. **Second**, Aksu discloses an exemplary pre-decision process to determine how many pixels—between one and six—to evaluate when determining the values of d_r and d_l . Aksu, 9:31-10:21. Aksu therefore discloses using three pixels as a special case of the pre-decision process used to determine the values of d_r and d_l . *Id.* As Aksu explains, this pre-decision process is not critical and can be replaced by other processes. *Id.*, 10:12-14. In my opinion, a POSITA would have recognized that, by replacing Aksu's multiple difference calculations with a one-dimensional Laplacian as taught by Gonzalez, this pre-decision process can be eliminated because the Laplacian will always use three pixels as inputs; there would be no need

to separately determine how many pixels to evaluate because the number would always be set at three pixels. This would, in turn, reduce the complexity of the deblocking filter, which was critical, especially for in-loop deblocking filters¹⁵ because of the immense computational burden they imposed by having to filter every reconstruction of a reference frame during the predictive-encoding process in both the encoder and decoder. Ex-1032, 2017 (noting in-loop deblocking comes “at the cost of a large amount of computation and memory read/write operations resulting in a reduced speed of both encoding and decoding processes”); Ex-1064, ¶ 22 (“The main drawback of these iterative de-blocking methods is their high computational complexity, which prevents them from being used in a real-time video decoder.”).

387. By eliminating the pre-decision process, a POSITA would have understood that Aksu’s filter decision would no longer have any “mathematical basis” with pixel values from a neighboring block. Specifically, Aksu’s pre-decision

¹⁵ Aksu does not expressly specify whether the deblocking filter was provided “in-loop” or as “post-processing.” Because of Aksu’s silence, in my opinion, a POSITA would have understood that Aksu was intended to be applicable to either, as the functioning of the deblocking filter does not change, but the only change is where in the decoder’s signal path it is placed (e.g., during image reconstruction, or after decoding is complete).

process calculated a value of Δ —representing the “difference of pixel values ... across the block boundary”—and comparing Δ to thresholds to determine how many pixels to evaluate to calculate d_r . Aksu, 9:27-10:8. But by using a Laplacian that always takes three pixels as inputs, there would be no need to calculate Δ or determine how many pixels to evaluate based on Δ as part of the pre-decision process.

388. *Third*, a POSITA also would have recognized that by replacing up to five difference calculations that required up to six pixels as inputs, as required by Aksu, with a single calculation that used three inputs for the Laplacian would further reduce computational complexity and resource demands. The combination would simplify Aksu’s calculation of d_r as follows:

<u>Aksu</u>	<u>Aksu + Gonzalez</u>
d_r determined based on, <i>e.g.</i> : $ r_1 - r_2 $ $ r_1 - r_3 $ $ r_1 - r_4 $ $ r_1 - r_5 $ $ r_1 - r_6 $	$d_r = r_3 - 2r_2 + r_1 $

A POSITA would have known that the Laplacian would be centered on pixel r_2 because the intention is to measure activity of the pixels closest to the boundary *within* a given block (here, block r). *See* Aksu, 13:10-13 (noting purpose of the calculation is to “represent activity ... between pixels on one side of the block boundary.”). Centering the Laplacian on r_1 would not be measuring activity only in block r , as it would incorporate pixel l_1 from the neighboring block, thus providing

a measure of activity *across* the boundary ($|r_2 - 2r_1 + l_1|$). Centering the Laplacian on r_3 would also be disfavored because that would omit the pixel closest to the boundary from the activity measure. That is why the prior art that used Laplacians as activity measures centered the Laplacian on the second-closest pixel from the block boundary. *See, e.g.*, H.241, 29; B205, 130.

389. A POSITA would have been especially motivated by the fact that similar modifications had been expressly proposed for Aksu's teachings during subsequent prior-art standards developments, confirming that Aksu's pre-decision process could be eliminated, and Aksu's series of difference calculations could be replaced by a single calculation looking at pixels closest to the block boundary.

390. Specifically, about a year after Aksu's provisional application was filed (and around the same time Aksu's non-provisional application was filed), one of the named inventors of Aksu (Marta Karczewicz) submitted a proposal (Doc. No. VCEG-L25) to the ITU Study Group 16 for a "Reduced Complexity Loop Filter." Ex-1067, 1. That proposal included: (1) a simplified version of Aksu's pre-decision process; (2) the same iterative first-order difference calculations as Aksu to calculate block-specific filter decision values (labeled a_r and a_l instead of d_r and d_l); (3) asymmetric filtering like Aksu of up to *two* pixels (instead of Aksu's three pixels) in each block based on the values of a_r and a_l , respectively; and (4) a simplified

averaging filter similar to Aksu (but always using a 3-pixel window instead of Aksu's variable-pixel window). *Id.*, 1-2.

391. This proposal was subsequently adopted into the next standard-test model that ITU was developing at the time. Ex-1068, 1 (“This d0 version of [test model] TML-6 includes the following changes from TML-5: ... Reduced complexity loopfilter (VCEG-L25).”).

392. After being adopted, these standards developments were then transitioned to the “Joint Video Team” (JVT) development effort to develop the next iteration of the H.26X standard (which was tentatively called H.26L and ultimately became H.264). Ex-1050, 1 (“This document presents Joint Model Number 1 of the [JVT] formed by ITU-T SG16 Q.6 (VCEG)”). The proposed deblocking filter included many of the features first proposed in VCEG-L25, including: (1) an even more simplified pre-decision process; (2) the same iterative first-order difference calculations to calculate block-specific filter decision values (now labeled a_p and a_q); and (3) the same asymmetric filtering of up to *two* pixels in each block based on the values of a_p and a_q , respectively. *Id.*, 33-34. Critically (at least for claim 10, discussed below), unlike Aksu and the VCEG-L25 proposal, the JVT draft standard calculated new pixel values using weighted-offset equations, instead of an averaging filter. *Id.*, 34.

393. At the following meeting, one of the participants then proposed (Doc. No. JVT-B011) modifying the Aksu-based deblocking filter to eliminate the pre-decision process entirely, and calculate the block-specific filter-decision values (a_p and a_q) using a *single* first-order difference calculation that looked to the difference between pixels p_0 and p_2 , and q_0 and q_2 (the first and third pixels closest to the block boundary), respectively. Ex-1051, 3-4. This proposal was then adopted by the JVT. Ex-1023, 1 (“The following changes are included relative to [working draft] WD-1: Deblocking filter JVT-B011,” noting its “general usefulness”), 48 (incorporating proposed modifications).

394. In other words, after Aksu’s disclosure, those in the art made nearly all of the same modifications to those proposed here: (1) replacing Aksu’s iterative difference calculations with a single evaluation of (two of) the three pixels closest to the block boundary; and (2) eliminating Aksu’s pre-decision process. The only difference between the modification here and those adopted during the standards-development process is the further obvious modification to use a second-order Laplacian to evaluate activity in the three pixels closest to the block boundary, rather than a single first-order difference calculation. But, as I discussed above, a POSITA would have known of the benefits of using a second-order Laplacian over a first-order difference calculation, and other subsequent standards used Laplacians for that exact same purpose. *See* H.241, 29; B205, 130, as I discuss in more detail below.

Accordingly, a POSITA would have been motivated to make the same types of modifications to Aksu that those in the art actually made, and would have had assurances that those modifications would work because standards proposals are not adopted absent scrutiny from other standards participants.

395. A POSITA would have also appreciated that using a Laplacian (as taught by Gonzalez) had been previously used or proposed for low-complexity measures of activity near a block boundary. Specifically, the H.241 standard was intended to be a “reduced complexity” deblocking filter compared to H.264, and proposed using a sum of Laplacians to calculate a “decision value” for the deblocking filter:

$$d = \left| p_{2_2} - 2 * p_{1_2} + p_{0_2} \right| + \left| q_{0_2} - 2 * q_{1_2} + q_{2_2} \right| + \left| p_{2_5} - 2 * p_{1_5} + p_{0_5} \right| + \left| q_{0_5} - 2 * q_{1_5} + q_{2_5} \right|$$

H.241, 1, 29. Moreover, a POSITA would have been aware that this same equation had been proposed for what was intended to be the next “high efficiency” video-codec standard (HEVC or H.265) just prior to the '841 Patent. *See* Ex-1001, 2:5-27; *see also* B205, 130.

396. A POSITA would have appreciated that H.241 and B205 were measuring *overall* activity near the block boundary for an entire block because they used two lines of pixels (2 and 5) from each block, and pixels from two adjacent blocks (p and q) in one equation. But a POSITA also would have appreciated that

Aksu's calculations of d_r and d_l were not to measure *overall* activity, but rather the activity in a particular line in each block separately. *See* Aksu, 13:30-33. For this reason, in my opinion, a POSITA would have known to maintain these separate calculations of d_r and d_l by making separate Laplacian calculations for a particular line for each block (e.g., $d = |p2_i - 2p1_i + p0_i|$).

397. A POSITA would have also understood that, when using the Laplacian to measure activity as part of Aksu's filter-decision calculation, the Laplacian would evaluate the three pixels closest to the block boundary (r_1, r_2, r_3). *See, e.g.,* Aksu, 12:29-13:4 (explaining how pixels "closest to the block boundary" are evaluated); 13:10-20 (calculating all pixel differences relative to " r_1 " closest to the boundary). Those in the art would have understood that the pixels closest to the block boundary are evaluated because that is where the blocking artifact exists. That is why Aksu discloses iteratively measuring activity beginning with the pixels closest to the boundary and moving further away. *See* Aksu, 12:29-13:4, 13:10-20 (all difference calculations involving r_1 and l_1 closest to the block boundary). That is also why other prior-art deblocking filters also measured activity using pixels nearest the block boundary. Ex-1037, 206, 207 (H.264 evaluating $|p0-q0|$, $|p1-p0|$ and $|q1-q0|$ to decide whether to filter, and $|p2-p0|$ and $|q2-p0|$ to determine how many pixels to filter); B118, 47 (same); Fu, 16 ("obtain the difference between the pair of pixels $R0$ and $L0$ on both sides of the block boundary ... and two pairs of pixels $L0$ and $L1$,

R0 and R1”); H.241, 29 (evaluating p0, p1, and p2, and q0, q1, and q2 to determine whether to filter); B205, 130 (same).

398. A POSITA also would have understood to take the absolute value—represented using vertical bars around the equation—of the Laplacian (*i.e.*, $|r_1 - 2r_2 + r_3|$) to calculate d_r (and d_l). The absolute value represents the *magnitude* of a calculation, and not its polarity (*i.e.*, positive or negative). For instance, the absolute value of 1 is 1, and the absolute value of -1 is also 1, because they both have a *magnitude* of 1, but in different directions (*i.e.*, $|-1| = |1| = 1$). To provide an example relevant for image processing, using the two sets of three pixels below as an example, both show a rapid change in pixel value due to the middle pixel:



If the values on the left were input into a one-dimensional Laplacian, that would produce a value of -150 ($125 - 2 \cdot 200 + 125$). If the values on the right were input into a one-dimensional Laplacian, that would produce a value of 150 ($125 - 2 \cdot 50 + 125$). But a POSITA would have understood that the deblocking filter should be equally influenced by both of these types of changes; it does not matter the *direction* in which the value of the pixels is changing, only by *how much* (*i.e.*, the *activity* of the pixels). That is why Aksu computes the absolute value of its difference calculations, and why others in the art did as well (including of the Laplacian). Aksu, 13:15-19 (each

difference calculation finding an absolute value $|x|$); *see also* H.241, 29 (same), B205, 130 (same); Ex-1037, 206-07 (H.264 same).

399. More generally, in my opinion, a POSITA would have been motivated to modify Aksu, and would have had a reasonable expectation of making the modification because modifying Aksu to use a Laplacian would have been nothing more than a simple substitution of one known element (a first-order activity measure using difference calculations) with another (a second-order activity measure using a Laplacian) to obtain predictable results (a measure of pixel activity near the block boundary). Furthermore, the modification applies a known technique (using a Laplacian to measure activity) to a known device (deblocking filter) ready for improvement (in computational efficiency) to yield predictable results (a lower-complexity and improved measure of activity), as demonstrated by H.241 and B205.

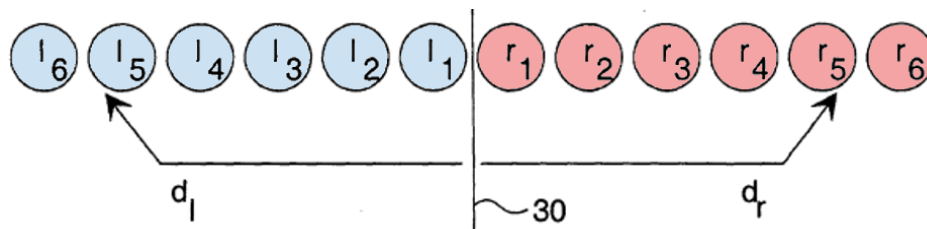
400. In my opinion, a POSITA would have had a reasonable expectation of success in replacing Aksu's multiple, first-order difference calculations for d_r with a second-order Laplacian. Making this modification would have involved basic computer programming to replace the calculations (and delete the pre-calculation determination of how many pixels to evaluate as part of the decision), which would have been within the skill of a POSITA.

- c. **Element 1[b]: calculating a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $|q_{2i}-2q_{1i}+q_{0i}|$ wherein q_{0i} denotes a**

pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_1 denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_2 denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;

401. In my opinion, Aksu in view of Gonzalez teaches or suggests element 1[b].

402. After calculating d_r , Aksu discloses calculating a value of “ d_l ... which represent[s] activity, or the difference of pixel values between pixels on one side of the block boundary.” Aksu, 13:10-19. To calculate d_l , Aksu performs the same series of difference calculations that were used for d_r , “except that all r ’s are replaced by l ’s” (e.g., r_1, r_2, r_3 , etc. are replaced with l_1, l_2, l_3 , etc. when calculating d_l). Aksu, 13:21-23.



Aksu, Fig. 2 (coloring added).

403. As I explained previously, however, a POSITA would have been motivated to replace Aksu’s difference calculations with a Laplacian, as taught by Gonzalez (and, more specifically, the absolute value of the Laplacian). See Section

X.C.1.b (element 1[a]). A POSITA would have understood that if the Laplacian were used to calculate d_r , it would also be used to calculate d_l , as Aksu acknowledges that “[t]he value of parameter d_l is determined similarly [to d_r], except that all r ’s are replaced by l ’s[.]” Aksu, 13:21-23. Upon making this modification, a POSITA would have known that the Laplacian would use only pixels l_1 , l_2 , and l_3 , which are the three pixels closest to the block boundary, just like pixel q_{0i} , q_{1i} , and q_{2i} in the ’841 Patent, ultimately resulting in the equation $|q_{2i}-2q_{1i}+q_{0i}|$, as claimed.

d. Element 1[c]: determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and

404. In my opinion, Aksu in view of Gonzalez teaches or suggests element 1[c].

405. Aksu discloses determining how many pixels in a line of pixels in block r to filter by comparing the first filter-decision value (d_r) to a threshold. Specifically, Aksu discloses that the value of d_r is determined by comparing its first-order difference calculations to a threshold based on β . Aksu, 13:15-19. Aksu, 15:15-36 (explaining $d_r = i$ where $|r_1 - r_{i+1}| > \beta/i$, , and $|r_1 - r_j| \leq \beta/j$, where j includes all values between 1 and i). The result of these comparisons, in turn, produces a value of d_r that controls how many pixels are filtered in the line of pixels in block r :

$d_r (d_l > 1)$	r_1	r_2	r_3
1	X	X	X
2	1	X	X
3	1	1*	X
4	2	2	X
5	2	2	2**
6	3 or 2***	3	3

Id., 15:24-36; *see also id.*, 15:17-19. As Aksu explains, the number associated with each pixel in the table represents the size of the “filtering window,” which “includes a number of pixels shown by the number from each side of the pixel being examined.” *Id.*, 15:17-26. In other words, when $d_r = 1$, no pixels in the “r” block are filtered. *See Id.*, 15:21-22 (“‘X’ means that the pixel in question is not filtered at all”). When $d_r = 2$, only pixel r_1 will be filtered, and its new value would be the mean of r_1 and one pixel on each side of r_1 : $(l_1 + r_1 + r_2)/3$. When $d_r = 3$, two pixels (r_1 and r_2) will be filtered, where r_1 again equals $(l_1 + r_1 + r_2)/3$ and r_2 equals $(r_1 + r_2 + r_3)/3$. Aksu also explains that the “1*” means the *filtered* value of r_1 is used for r_2 , not the original r_1 pixel value (so the equation would be $(r_1' + r_2 + r_3)/3$). *Id.*, 16:1-3.

406. A POSITA would have understood that Aksu sets d_r equal to an integer value between 1 and 6 as a way to track when the multiple difference calculations that are being performed exceed a threshold in order to determine activity in a block. Specifically, the set of equations Aksu provides effectively serves as a way to find the location at which the difference between two pixels exceeds a threshold, indicating that pixel location is the cutoff point for filtering.

$$\begin{aligned} d_r &= 6, \text{ if } |r_1 - r_j| \leq \beta/j \text{ with all } j \in [1,6], \\ \text{otherwise: } d_r &= i, \text{ where } i \text{ fulfils the conditions} \\ i &\in [1, n_{tr}], \quad (4) \\ |r_1 - r_{i+1}| &> \beta/i, \text{ and} \\ |r_1 - r_j| &\leq \beta/j \text{ with all } j \in [1,i]. \end{aligned}$$

Aksu, 13:15-19. In the following, I will explain these equations in greater detail.

407. At the start, Aksu provides: $d_r = 6$ when $|r_1 - r_j| \leq \beta/j$ with all $j \in [1,6]$.

This means that the only way d_r will equal 6 is if, for every value of j between 1 and 6, each of the following equations is true:

$$\begin{aligned} |r_1 - r_2| &\leq \beta/2 \\ |r_1 - r_3| &\leq \beta/3 \\ |r_1 - r_4| &\leq \beta/4 \\ |r_1 - r_5| &\leq \beta/5 \\ |r_1 - r_6| &\leq \beta/6 \end{aligned}$$

In other words, if every pixel difference is less than the prescribed threshold, that means pixels 1-6 are considered to have low activity, and thus more filtering can be performed. Otherwise, d_r will equal “i,” where “i” is between 1 and a (truncated) value of “n” (the number of pixels to evaluate as part of the filter decision that was determined by the pre-decision process, Aksu, 11:5-12:5), based on the provided equations:

$$\begin{aligned} |r_1 - r_{i+1}| &> \beta/i, \text{ and} \\ |r_1 - r_j| &\leq \beta/j \text{ with all } j \in [1,i]. \end{aligned}$$

Basically, what these two equations seek is the point at which the difference between r_1 and another pixel (r_{2-6}) (or its first-order activity) transitions from being less than

or equal to a threshold to being greater than a threshold. For instance, if $|r_1 - r_2| \leq \beta/2$, but $|r_1 - r_3| > \beta/3$, d_r will be set equal to 2 because a substantial-enough difference exists between the r_1 and r_3 pixels, meaning filtering should be more limited. But if $|r_1 - r_2| \leq \beta/2$, $|r_1 - r_3| \leq \beta/3$, and only $|r_1 - r_4| > \beta/4$, d_r will be set equal to 3. In sum, Aksu discloses that as the difference in pixel values increases, less filtering should be performed because a larger difference indicates activity in the block. Aksu, 13:33-14:5 (“it is advantageous that these parameters are used to limit the blocking artefact processing relatively close to the block boundary, if there is a real image edge near the block boundary.... [T]he value of parameter d_r provides an indication of how many pixels counted from the block boundary have approximately the same value as the pixel at the block boundary.”).

408. Aksu uses integers 1 through 6 for d_r because it relies on a *series* of difference calculations compared to a threshold (β/j) across several pixels to perform a measure of activity, and therefore must track the activity measure for the line of pixels while basing it on multiple calculations. But if these *series* of calculations were replaced with a single Laplacian as taught by Gonzalez, *see* Section X.C.1.b (element 1[a]), there would be no need to keep track of these different calculations through variable d_r . Instead, d_r can simply be defined as the output of the Laplacian ($d_r = |r_3 - 2r_2 + r_1|$), and d_r could be *directly* compared to a threshold based on β .

409. Not only is a direct comparison between the filter-decision value (d) to the threshold consistent with H.241 and B205—which both made such a comparison, H.241, 29 (no filtering when “ $d \geq \beta$ ”); B205, 130 (filtering when $d < \beta$); Ex-1001, 2:5-27 (recognizing same for B205)—but also when Aksu’s teachings were modified during subsequent standards developments to replace the series of difference calculations with a single difference calculation to calculate a filter-decision value for each block (a_p and a_q), the output of that calculation was compared directly to the threshold (β). Ex-1051, 4 (proposal); B118, 1 (adoption of proposal). Specifically, after filtering the two pixels closest to the block boundary (labeled $p1$ and $q1$ in the proposal), a filter-decision value (a_p) was calculated using the equation $a_p = |p3 - p1|$ (with $p1$ being the pixel closest to the boundary), and if $a_p < \beta$, an additional pixel ($p2$) (with $p2$ being the second closest from the block boundary) would be filtered. Otherwise, only one pixel ($p1$) would be filtered. Ex-1051, 4; B118, 48. This same asymmetric filtering based on a comparison of $a_p < \beta$ was ultimately adopted in H.264. Ex-1036, 191-92.

410. Thus, as post-Aksu developments would have confirmed, if d_r is defined by a Laplacian and is below a threshold, that indicates there is no significant activity in the three pixels near the block boundary in that block, and more filtering can be performed (just like Aksu and post-Aksu developments), but if d_r is defined by a Laplacian and is above a threshold, that indicates there is more activity in the

pixels near the block boundary in that block, and less or no filtering should be performed (again, just like Aksu and post-Aksu developments).

411. Finally, I understand that the parties dispute the construction of the “determining” step in this claim element, and specifically whether it permits the determination of how many pixels to filter to have any mathematical basis in pixels from a neighboring block. In my opinion, both Aksu’s difference calculations *and* a Laplacian in view of the Aksu-Gonzalez combination disclose that determining how many pixels to filter in a block has no mathematical relationship with pixels in another block because d_r determines how many pixels are filtered, and uses only pixels from block r .

412. To the extent the threshold must also be based only on block r , Aksu also discloses this relationship. Specifically, Aksu discloses that the threshold is based on the minimum “QP” value of block r and block l . Aksu, 10:9-12. Thus, Aksu discloses a circumstance in which β (the basis for the thresholds) depends only on the QP of block r (when $QP_r < QP_l$). I understand that a claim can be obvious if it is taught by the prior art under some specific conditions, even if the prior art may perform differently under other, different conditions (*e.g.*, if $QP_r > QP_l$).

413. In my opinion, however, a POSITA would have also been motivated to select block-specific thresholds (*i.e.*, β_r for block r , and β_l for block l), for the same reasons I discussed above. *See* Section X.A.1.e (Ground 1, element 1[d]).

- e. **Element 1[d]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.**

414. In my opinion, Aksu in view of Gonzalez teaches or suggests element 1[d].

415. Aksu discloses that “[t]he [modified] values of the pixels l_1 , l_2 and l_3 are determined in the same manner according to the value of the parameter d_l .” Aksu, 15:19-21. For this reason, in my opinion, a POSITA would have known to use d_l to determine how many pixels in the “l” block to filter based on a comparison to a threshold based on β . *See* Section X.C.1.d (element 1[c]). However, to the extent the second threshold for block l must be different than the threshold for block r, selecting block-specific thresholds (*i.e.*, β_r and β_l) based on the QP of each block would have been obvious to a POSITA, for the reasons I discussed above. *See* Sections X.C.1.d (element 1[c]), X.A.1.e (Ground 1, element 1[d]).

2. Claim 3

- a. **Element 3[a]: The method according to claim 1, wherein determining how many pixels in said first line of pixels in said block to filter comprises:**

416. For the reasons provided in more detail below, in my opinion, Aksu in view of Gonzalez teaches or suggests claim 3. And as I discussed previously, Aksu

and Gonzalez teach or suggest the “determining” step of claim 1. *See* Section X.C.1.d (element 1[c]).

b. Element 3[b]: comparing said first filter decision value to a first threshold value;

417. As I explained previously, Aksu in view of Gonzalez teaches or suggests comparing a first filter-decision value (d_r) to a first threshold value (β or β_r). *See* Section X.C.1.d (element 1[c]).

c. Element 3[c]: determining to filter two pixels in said first line of pixels in said block relative to said block boundary if said first filter decision value is below said first threshold value; and

418. In my opinion, Aksu in view of Gonzalez teaches or suggests element 3[c].

419. Specifically, Aksu discloses a d_r value ($d_r=3$) in which two pixels are filtered. Aksu, 15:24-36. For d_r to equal 3, that means that two of Aksu’s first-order difference calculations ($|r_1-r_2|$, $|r_1-r_3|$) are less than a threshold based on β . *Id.*, 13:15-19 (explaining $d_r = i$ where $|r-r_{i+1}|$ exceeds the threshold). Aksu also discloses that the value of d_r depends on how many of Aksu’s difference calculations are less than a threshold based on β , in which case more pixels are filtered. Aksu, 13:15-19; *see* Section X.C.1.d (element 1[c]).

420. As I discussed previously, in my opinion, a POSITA would have known, when using a Laplacian (as taught by Gonzalez), to compare the output directly to

the threshold (as suggested by Aksu) to determine how many pixels to filter. *See* Section X.C.1.d (element 1[c]). In my opinion, a POSITA would have been further motivated to establish a set of relationships between d_r and the threshold (β or β_r) to preserve Aksu's different filtering operations (*e.g.*, how many pixels to filter and/or how much to filter each pixel) based on the value of d_r .

421. For instance, a POSITA would have been motivated to determine different inequalities based on empirical testing to control the number of pixels to filter in each block. I have provided below an example of such relationships for demonstrative purposes only:

$d_r \geq \beta$	No pixels
$\beta > d_r \geq \beta/2$	Filter r_1
$\beta/2 > d_r \geq \beta/3$	Filter r_1 and r_2
$\beta/3 > d_r \geq \beta/4$	Filter r_1, r_2 , and r_3

See Section X.C.1.d (element 1[c]) (explaining how Aksu filters more pixels when more difference calculations are less than a threshold, and vice versa). Determining the thresholds, however, would have involved nothing more than optimizing the thresholds until a desired amount of deblocking is achieved. Providing asymmetric filtering of two adjacent blocks was the entire point of Aksu's invention. *See, e.g.*, Aksu, 8:15-17, 13:30-14:5 (“[I]t is advantageous that these [d] parameters are used

to limit the blocking artefact processing relatively close to the block boundary, if there is a real image edge near the block boundary.”).

422. Alternatively, a POSITA would have been motivated to simplify Aksu’s asymmetric filter, and use a single threshold to control how many pixels would be filtered. As I mentioned previously, *see* Section X.C.1.d (element 1[c]), a POSITA would have known that during subsequent standards developments, Aksu’s teachings were modified to provide a simpler deblocking filter that filtered two pixels if the p-block filter-decision value (a_p) was less than a threshold, or otherwise filter one pixel. While this modification would provide slightly less control over *how many* pixels would be filtered (because only 0, 1, or 2 pixels would be filtered, without the option for 3 pixels as Aksu discloses), the efficiency gains by selecting a single threshold for each block (rather than a series of thresholds for each block) would have been apparent to a POSITA.

423. In my opinion, therefore, a POSITA would have been motivated to preserve this asymmetric filtering while enabling more-efficient computation of the activity level in each block using a Laplacian, for the reasons I have explained with respect to claim 1. *See* Sections X.C.1.b (element 1[a]), X.C.1.d (element 1[c]). And a POSITA would have had a reasonable expectation of success in achieving this because it is largely a routine empirical exercise of testing different threshold values to map those thresholds to filtering different numbers of pixels.

- d. Element 3[d]: determining to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value; and**

424. In my opinion, Aksu in view of Gonzalez teaches or suggests element 3[d].

425. As I explained previously, Aksu discloses a d_r value ($d_r = 2$) in which one pixel is filtered. Aksu, 15:24-36. For d_r to equal 2, that means that only one of Aksu's first-order difference calculations ($|r_1 - r_2|$) is less than a threshold based on β . *Id.*, 13:15-19 (explaining $d_r = i$ where $|r - r_{i+1}|$ exceeds the threshold). Aksu also discloses that the value of d_r depends on the point at which one of Aksu's first-order difference calculations equal or exceed a threshold based on β , at which point fewer pixels are filtered. Aksu, 13:15-19; Section X.C.1.d (element 1[c]).

426. As I also explained previously, in my opinion, a POSITA would have been motivated to: (1) compare d_r directly to threshold (β or β_r); and (2) determine a threshold (β or β_r) in which one pixel would be filtered in block "r" when d_r was equal to or exceeded that threshold. *See* Sections X.C.1.d (element 1[c]), X.C.2.c (element 3[c]).

- e. Element 3[e]: determining how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises:**

427. For the reasons provided in more detail below, in my opinion, Aksu in view of Gonzalez teaches or suggests element 3[e]. And as I discussed previously,

Aksu and Gonzalez teach or suggest the “determining” step of claim 1. *See* Section X.C.1.e (element 1[d]).

f. Element 3[f]: comparing said second filter decision value to a second threshold value;

428. As I explained previously, Aksu in view of Gonzalez teaches or suggests comparing a second filter-decision value (d_l) to a second threshold value (β or β_l). *See* Sections X.C.1.d (element 1[c]), X.C.1.e (element 1[d]).

g. Element 3[g]: determining to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value; and

429. In my opinion, Aksu in view of Gonzalez teaches or suggests element 3[g].

430. Specifically, Aksu discloses filtering the “l” following the same procedure as the “r” block, but the decisions whether to filter pixels in each block are made separately and filters are applied separately. *See* Sections X.C.1.d (element 1[c]), X.C.1.e (element 1[d]). And for the same reasons I explained with respect to block “r,” in my opinion, a POSITA would have been motivated and had a reasonable expectation of success in modifying Aksu to: (1) compare d_l directly to threshold (β or β_l); and (2) establish a relationship between d_l and threshold (β or β_l) in which two pixels would be filtered in block “l” when d_l was below that second

threshold. *See* Sections X.C.1.d (element 1[c]), X.C.1.e (element 1[d]), X.C.2.c (element 3[c]).

- h. Element 3[h]: determining to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second threshold value.**

431. In my opinion, Aksu in view of Gonzalez teaches or suggests element 3[h].

432. Specifically, Aksu discloses that the deblocking filter is intended to operate the same in the “r” block and the “l” block, but the decisions whether to filter pixels in each block are made separately and filters are applied separately. *See* Sections X.C.1.d (element 1[c]), X.C.1.e (element 1[d]). And for the same reasons I explained with respect to block “r,” in my opinion, a POSITA would have been motivated and had a reasonable expectation of success in modifying Aksu to: (1) compare d_l directly to threshold (β or β_l); and (2) establish a relationship between d_l and threshold (β or β_l) in which one pixel would be filtered in block “l” when d_l was equal to or exceeded that threshold. *See* Sections X.C.1.d (element 1[c]), X.C.1.e (element 1[d]), X.C.2.c (element 3[c]), X.A.2.d (element 3[d]).

3. Claim 5

a. Element 5[pre]: A filtering control device comprising:

433. If the preamble is limiting, in my opinion, Aksu discloses or suggests element 5[pre].

434. Specifically, Aksu discloses “[a] method and associated device for filtering digital video images.” Aksu, 1:1. Aksu also discloses that the device “implement[s] the method” of filtering. *Id.*, 6:12-16. Section X.C.1 (method). For this reason, in my opinion, Aksu discloses or suggests a filtering control device, as claimed.

b. Element 5[a]: a first decision value calculator configured to calculate a first filter decision value for a block of multiple pixels in a video frame, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $|p_{2i}-2p_{1i}+p_{0i}|$ wherein p_{0i} denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;

435. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[a]. *See* Section X.C.1.b (element 1[a]). Moreover, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “first

decision value calculator” to calculate the “first decision value” addressed previously. *See* Section X.C.1.b (element 1[a]).

- c. **Element 5[b]: a second decision value calculator configured to calculate a second filter decision value for a neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $|q_{2i}-2q_{1i}+q_{0i}|$ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;**

436. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[b]. *See* Section X.C.1.c (element 1[b]). Moreover, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “second decision value calculator” to calculate the “second decision value” addressed previously. *See* Section X.C.1.c (element 1[b]).

- d. **Element 5[c]: a first pixel determiner configured to determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and**

437. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[c]. *See* Section X.C.1.d (element 1[c]). Moreover, Aksu discloses a “digital signal

processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “first pixel determiner” to determine how many pixels to filter in the line of pixels in block r, as addressed previously. *See* Section X.C.1.d (element 1[c]).

- e. **Element 5[d]: a second pixel determiner configured to determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator, wherein**

438. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[d]. *See* Section X.C.1.e (element 1[d]). Moreover, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “second pixel determiner” to determine how many pixels to filter in the line of pixels in neighboring block l, as addressed previously. *See* Section X.C.1.e (element 1[d]).

- f. **Element 5[e]: further comprising a first comparator configured to compare said first filter decision value calculated by said first decision value calculator to a first threshold value; and**

439. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[e]. *See* Sections X.C.1.d (element 1[c]), X.C.2.b (element 3[b]). Moreover, Aksu

discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “first comparator” to compare the first filter decision value and the threshold, as addressed previously. *See* Sections X.C.1.d (element 1[c]), X.C.2.b (element 3[b]).

- g. Element 5[f]: a second comparator configured to compare said second filter decision value calculated by said second decision value calculator to a second threshold value.**

440. In my opinion, Aksu in view of Gonzalez teaches or suggests element 5[f]. *See* Sections X.C.1.e (element 1[d]), X.C.2.f (element 3[f]). Moreover, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “second comparator” to compare the second filter decision value and the threshold, as addressed previously. *See* Sections X.C.1.e (element 1[d]), X.C.2.f (element 3[f]).

4. Claim 7

- a. Element 7[a]: The device according to claim 5, further comprising: said first pixel determiner is configured to i) determine to filter two pixels in said first line of pixels in said block relative to said block boundary if said**

first filter decision value is below said first threshold value as determined by said first comparator and

441. In my opinion, Aksu in view of Gonzalez teaches or suggests element 7[a]. *See* Sections X.C.2.c (element 3[c], two pixels); X.C.3.d (element 5[c], first determiner); X.C.3.f (element 5[e], first comparator).

- b. Element 7[b]: ii) determine to filter one pixel in said first line of pixels in said block relative to said block boundary if said first filter decision value is equal to or exceeds said first threshold value as determined by said first comparator; and**

442. In my opinion, Aksu in view of Gonzalez teaches or suggests element 7[b]. *See* Sections X.C.2.d (element 3[d], one pixel); X.C.3.d (element 5[c], first determiner); X.C.3.f (element 5[e], first comparator).

- c. Element 7[c]: said second pixel determiner is configured to i) determine to filter two pixels in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is below said second threshold value as determined by said second comparator and**

443. In my opinion, Aksu in view of Gonzalez teaches or suggests element 7[c]. *See* Sections X.C.2.g (element 3[g], two pixels); X.C.3.e (element 5[d], second determiner); X.C.3.g (element 4[f], second comparator).

- d. Element 7[d]: ii) determine to filter one pixel in said corresponding first line of pixels in said neighboring block relative to said block boundary if said second filter decision value is equal to or exceeds said second**

threshold value as determined by said second comparator.

444. In my opinion, Aksu in view of Gonzalez teaches or suggests element 7[d]. *See* Sections X.C.2.h (element 3[h], one pixel), X.C.3.e (element 5[d], second determiner), X.C.3.g (element 5[f], second comparator).

5. Claim 9

- a. Element 9[a]: The device according to claim 5, further comprising a threshold determiner configured to i) determine said first threshold value based on a quantization parameter associated with said block, and**

445. In my opinion, Aksu in view of Gonzalez teaches or suggests element 9[a].

446. Specifically, as I explained previously, when combining Aksu with Gonzalez, the Laplacian (taught by Gonzalez) would be compared to a threshold based on β (as disclosed by Aksu). *See* Section X.C.1.d (element 1[c], including a discussion of selecting block-specific thresholds). Aksu also discloses performing a calculation to determine threshold β (“ $\beta = 4\log(QP)$ ”). Aksu, 13:21; *see also id.*, 9:31-35 (defining “QP” as the “quantization step”), and that threshold β can be selected based on the quantization parameter of block r , *id.*, 10:9-12 (*e.g.*, selecting QP_r if $QP_r < QP_l$).

447. However, for the reasons I discussed above, in my opinion, a POSITA would have been motivated to select a block-specific threshold, which is based on the quantization parameter of block *r*. *See* Section X.A.1.e (Ground 1, element 1[d]).

448. Finally, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that the processor is therefore programmed or designed to have a “threshold determiner” that determines threshold (β or β_r).

b. Element 9[b]: ii) determine said second threshold value based on a quantization parameter associated with said neighboring block.

449. In my opinion, Aksu in view of Gonzalez teaches or suggests element 9[b]. *See* Sections X.C.1.d (element 1[c], Aksu’s thresholds, generally), X.C.1.e (element 1[d], threshold for neighboring block); X.C.5.a (element 9[a], determining the threshold); Section X.A.1.e (Ground 1, element 1[d], block-specific thresholds).

6. Claim 12

a. Element 12[a]: An encoder comprising a filtering control device according to claim 5.

450. Aksu in view of Gonzalez teaches or suggests element 12[a].

451. Specifically, Aksu discloses that “the present invention relates to a video encoder comprising means for coding and decoding a digital video signal by blocks ... which encoder comprises a filter for reducing visual artefacts due to a

block boundary.” Aksu, 1:13-18. Aksu therefore teaches including a filtering control device in an encoder. And for the reasons I provide above, Aksu in view of Gonzalez also teach or suggest the filtering control device of claim 5. *See* Section X.C.3 (claim 5).

7. Claim 13

a. Element 13[pre]: A user equipment comprising:

452. If the preamble is limiting, in my opinion, Aksu discloses or suggests element 13[pre].

453. Specifically, Aksu discloses that “the invention is implemented in a digital signal processor or corresponding device suited for processing a digital signal.” Aksu, 17:19-23. Aksu also discloses “a mobile terminal 46 intended for use as a portable video telecommunication device and applying the method according to the invention.” Aksu, 18:31-19:3, Fig. 6.

454. Based on this disclosure, a POSITA would have understood that a “mobile terminal” constitutes “user equipment” as recited in the claim because user equipment is known to be a generic term for a user-facing device that implements an encoder and/or decoder. The ’841 Patent explains that “user equipment” can be a generic “computer.” *See, e.g.*, Ex-1001, 24:1-2 (“[t]he computer 70 of FIG. 12 can be a user equipment or be present in a user equipment.”); *see also id.*, 25:27-30 (“The user equipment 80 can be any device having media decoding functions.”).

b. Element 13[a]: a memory configured to store video frames; and

455. In my opinion, Aksu discloses or suggests element 13[a].

456. Aksu begins by describing the conventional process typically followed by video encoders, like the one I provide above, *see* Section V.A (encoding/decoding process overview). As part of this process, it was known that encoders typically will include a decoder “in-loop”—which undoes the actions of the encoder—to reconstruct the encoded frame as a “reference frame” that is then used to predict and encode the current frames. *See id*; Aksu, 3:5-33. As part of that in-loop decoder (within the encoder), Aksu explains that “[t]he decoded frame is saved in a frame memory 16. To code the next frame, the frame saved in the frame memory 16 is read as a reference frame ... and is transformed into a new prediction frame.” Aksu, 3:5-21.

457. Based on this disclosure, a POSITA would have understood that Aksu discloses user equipment (*e.g.*, the device or computer with the encoder) for encoding video frames that includes memory to store video frames as part of the encoding process. Without memory to store video frames, an encoder would not be able to reconstruct and store already-encoded video frames to serve as reference frames for subsequent encoding.

c. Element 13[b]: an encoder according to claim 12 configured to encode said video frames into encoded

video frames, wherein said memory is further configured to store said encoded video frames.

458. In my opinion, Aksu in view of Gonzalez teaches or suggests element 13[b]. *See* Sections X.C.6 (claim 12), X.C.7.b (element 13[a]). Aksu also discloses that the encoded video frames are transmitted to a decoder, Aksu, 2:5-7.

459. A POSITA would have known that it was conventional to store encoded video frames locally in the encoding device and prior to transmission. Aksu, 2:5-8; *see also, e.g.*, Ex-1040, 3 (describing encoding, transmission, and decoding operations), 247 (“It is common for single compressed video sequences to be stored in files.”); Ex-1062, 10:56-59 (“The encoder encodes any frames that it receives and then stores them on the storage, which could be computer memory, hard disk, DVD, or similar media.”); Ex-1066, ¶ 30 (“The output video bitstream is then stored or transmitted through a communication channel.”). In fact, video is conventionally stored in memory in its encoded form in order to significantly reduce the space needed to store the video.

460. For this reason, even if Aksu did not expressly disclose storing encoded video frames in memory (such as prior to transmission), in my opinion, a POSITA would have been motivated to configure Aksu to store encoded video frames in memory because that was conventionally done in the art. *See, e.g.*, Ex-1040, 3; Ex-1062, 10:56-59; Ex-1066, ¶ 30.

8. Claim 14

a. Element 14[a]: A decoder comprising a filtering control device according to claim 5.

461. In my opinion, Aksu in view of Gonzalez teaches or suggests element 14[a].

462. Specifically, Aksu discloses that “[t]he present invention also relates to a video decoder comprising means for reducing visual artefacts in a frame of a digital video signal ... which video decoder comprises a filter for reducing visual artefacts due to a block boundary.” Aksu, 1:13-18. Aksu therefore teaches building a filtering control device into a decoder. And for the reasons I provide above, Aksu in view of Gonzalez also teach or suggest the filtering control device of claim 5. *See* Section X.C.3 (claim 5).

9. Claim 15

a. Element 15[pre]: A user equipment comprising:

463. If the preamble is limiting, in my opinion, Aksu discloses or suggests element 15[pre]. *See* Section X.C.7.a (element 13[pre]).

b. Element 15[a]: a memory configured to store encoded video frames;

464. In my opinion, Aksu discloses or suggests element 15[a].

465. Aksu begins by describing the conventional process typically followed by video decoders, like the one I provide above, *see* Section V.A (encoder/decoder

process overview); Aksu, 4:7-10. Although Aksu discloses that *decoded* video frames are “saved in frame memory” to serve as decoding predictions for subsequent frames, Aksu, 4:13-18, Aksu does not expressly disclose a memory to store *encoded* video frames.

466. However, a POSITA would have known that a decoder would store the encoded video frames in memory after being encoded by an encoder, prior to decoding. As I explained previously, video was conventionally stored in its encoded form to reduce space before being retrieved for decoding and displaying on a screen. *See* Section X.C.7.c (element 13[b]); *see also, e.g.*, Ex-1040, 3 (describing encoding, transmission, and decoding operations), 247 (“It is common for single compressed video sequences to be stored in files.”); Ex-1062, 10:56-59 (“The encoder encodes any frames that it receives and then stores them on the storage, which could be computer memory, hard disk, DVD, or similar media.”); Ex-1066, ¶ 30 (“The output video bitstream is then *stored* or transmitted through a communication channel.”). Aksu likewise discloses that the encoded bitstream is intended to be transmitted from an encoder to a decoder, Aksu, 3:32-33, and thus a POSITA would have appreciated that Aksu is either storing the encoded bitstream locally (to be subsequently used by a local decoder), or that the encoded bitstream is transmitted by an encoder and then stored by the “user equipment” with a decoder prior to decoding.

- c. Element 15[b]: a decoder according to claim 14 configured to decode said encoded video frames into decoded video frames; and**

467. In my opinion, Aksu in view of Gonzalez teaches or suggests element 15[b]. *See* Sections X.C.8, X.C.9.b.

- d. Element 15[c]: a media player configured to render said decoded video frames into video data displayable on a display.**

468. In my opinion, Aksu in view of Gonzalez teaches or suggests element 15[c].

469. In my opinion, Aksu discloses a media player because Aksu discloses that “the mobile terminal 46 comprises at least display means 47 for displaying images” and is intended as “a portable video telecommunication device.” Aksu, 18:31-19:3. In other words, Aksu envisions the decoded video being displayed on a screen. Ex-1001, 25:25-35 (noting a media player can “include mobile telephones and other portable media players”).

470. However, a POSITA also would have known that the use of a media player to implement video decoders was extremely old and well known in the art at the relevant time. For instance, video decoders are essential for dedicated media players such as DVD players and television set-top boxes. *See* Ex-1040, 1-5. Moreover, software-based media players are also essential to multi-purpose computing devices. At the time of the '841 Patent, Windows Media Player had

already existed for decades, which incorporated “various video codecs ... including MPEG1 ... [and] an unofficial variation of MPEG4.” Ex-1055. Another well-known media player that also dated back decades is “VLC,” which supported many different video codecs. Ex-1071.

471. For this reason, to the extent the claim requires, and Aksu does not disclose, a dedicated “media player” separate from the decoder, a POSITA would have known to build Aksu’s user equipment with a media player to enable playback of the decoded video on a display.

10. Claim 16

- a. Element 16 [pre]: A non-transitory computer-readable medium comprising, stored thereupon, a computer program for filtering control of a block of multiple pixels in a video frame each pixel having a respective pixel value, said computer program comprising code means which when run on a computer causes the computer to:**

472. In my opinion, Aksu discloses or suggests element 16[pre].

473. Specifically, Aksu discloses “[a] storage medium can be used for storing a software program comprising machine executable steps for performing the method according to the invention,” and that “the software program is read from the storage medium to a device comprising programmable means, *e.g.* a processor, for performing the method of the invention.” Aksu, 18:10-15. In other words, by disclosing a “storage medium ... for storing a software program,” Aksu discloses a

“non-transitory computer-readable medium” with a “computer program.”

Furthermore, as discussed previously, the “method” Aksu discloses to be performed using the storage medium and processor provides is for filtering control. *See* Section X.C.1.a (element 1[pre]).

- b. Element 16[a]: calculate a first filter decision value for said block, by only using pixels p_{2i} , p_{1i} , and p_{0i} , based on: $|p_{2i}-2p_{1i}+p_{0i}|$ wherein p_{0i} , denotes a pixel value of a pixel closest to, in a first line of pixels in said block, a block boundary to a neighboring block of multiple pixels in said video frame, p_{1i} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary and p_{2i} denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;**

474. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[a]. *See* Section X.C.1.b (element 1[a]).

- c. Element 16[b]: calculate a second filter decision value for said neighboring block, by only using pixels q_{2i} , q_{1i} , and q_{0i} , based on: $|q_{2i}-2q_{1i}+q_{0i}|$ wherein q_{0i} denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding first line of pixels in said neighboring block, said block boundary, q_{1i} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary and q_{2i} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary;**

475. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[b]. *See* Section X.C.1.c (element 1[b]).

- d. Element 16[c]: determine how many pixels in a first line of pixels in said block to filter relative to said block boundary based on said first filter decision value; and**

476. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[c]. *See* Section X.C.1.d (element 1[c]).

- e. Element 16[d]: determine how many pixels in a corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value,**

477. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[d]. *See* Section X.C.1.e (element 1[d]).

- f. Element 16[e]: wherein determine how many pixels in said first line of pixels in said block to filter comprises determining how many pixels in said first line of pixels in said block to filter relative to said block boundary based on a comparison of said first filter decision value to a first threshold value; and**

478. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[e]. *See* Sections X.C.1.d (element 1[c]), X.C.2.b (element 3[b]).

- g. Element 16[f]: determine how many pixels in said corresponding first line of pixels in said neighboring block to filter comprises determining how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on a comparison of said second filter decision value to a second threshold value.**

479. In my opinion, Aksu in view of Gonzalez teaches or suggests element 16[f]. *See* Sections X.C.1.e (element 1[d]), X.C.2.f (element 3[f]).

D. Ground 2B: Obviousness over Aksu in view of Gonzalez, H.241, and B205/AAPA

1. Claim 2

480. I understand that in the co-pending ITC proceedings, the respondents contend that there may be a contradiction between claims 1 and 2, because claim 1 requires using “only” three pixels in one “first” line of pixels to make a filter decision, but claim 2 requires six pixels from two lines (three pixels each) to make a filter decision.

481. I do not provide any opinion on the definiteness of this claim, but I have instead assumed that claim 1 does not foreclose claiming multiple lines of pixels when applying the prior art.

- a. **Element 2[a]: The method according to claim 1, wherein calculating said first filter decision value comprises calculating said first filter decision value as: $|p_{2_0} - 2p_{1_0} + p_{0_0}| + |p_{2_3} - 2p_{1_3} + p_{0_3}|$ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{2_3} denotes a pixel**

value of a pixel second next closest to, in said second line of pixels, said block boundary; and

482. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 2[a].

483. Aksu discloses calculating a filter-decision value (d_r) using pixels from a single line in the block, Aksu, 13:10-19, and then repeating the filter-decision calculation for every line in the block, *id.* at 16:15-18. Aksu does not expressly disclose calculating a filter-decision value using pixels from two lines in the block, or that those lines should be 0 and 3.

484. For the same reasons I discussed in Ground 1, however, a POSITA would have been motivated to use two lines to make a filter decision for the entire block, as taught by H.241 and B205 (and many others), to reduce the computational complexity of the filter-decision calculations. *See* Section X.B.1.a (Ground 1, element 2[a]). And as I also explained previously, the selection of any particular lines would have been a design choice. *See* Section X.B.1.a (Ground 1, element 2[a]).

485. Although Aksu discloses performing deblocking on an 8x8 block as one example, Aksu, 10:3-5, even for an 8x8 block, there are 21, two-line options to choose from that maintain spacing between lines (0-2; 0-3; 0-4; 0-5; 0-6; 0-7; 1-3; 1-4; 1-5; 1-6; 1-7; 2-4; 2-5; 2-6; 2-7; 3-5; 3-6; 3-7; 4-6; 4-7; 5-7). However, in my opinion, a POSITA would have known that Aksu's teaching of asymmetric filtering

is not limited to 8x8 blocks, and filtering 4x4 blocks was also common. *See, e.g.*, Ex-1037, 197 (“The conditional filtering process is applied to all $N \times N$ (*where* $N = 4$ or $N = 8$ for luma ...”); Ex-1053, 26 (“Considering that deblocking filter is applied to all of the vertical and horizontal edges of 4x4 blocks”); Ex-1056, 3 (“Considering that each basic deblocking filtering unit is a 4x4 block...”); Ex-1040, 139-41 (noting use of 4x4 blocks, and choice of block size “tends to be a trade-off between (i) prediction efficiency and (ii) cost of signalling the prediction mode.”), 149, 158 (showing breakdown of block sizes).

486. In my opinion, therefore, a POSITA would have been motivated not only to modify Aksu to calculate d_r (and d_l) using two lines from a block to make a filter decision for the entire block, but also to select lines 0 and 3 for the calculation as a design choice, especially when performing deblocking on 4x4 blocks, and would have known that such modifications would work. Section X.B.1.a (Ground 1, element 2[a]).

- b. Element 2[b]: calculating said second filter decision value comprises calculating said second filter decision value as: $|q_{20}-2q_{10}+q_{00}|+|q_{23}-2q_{13}+q_{03}|$ wherein q_{00} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{10} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{20} denotes said pixel value of said pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block**

boundary, $q0_3$ denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, $q1_3$ denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and $q2_3$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.

487. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 2[b]. Specifically, for the reasons discussed in Ground 1, a POSITA would have found it obvious two lines in a block as representative to make a filter decision calculation for the entire block not just for block “r” (d_r), but also block “l” (d_l). *See* Sections X.B.1.a (Ground 1, element 2[a]), X.D.1.a (Ground 2, element 2[a]). And a POSITA would have been further motivated to select lines 0 and 3 as an obvious design choice. *See* Sections X.B.1.a (Ground 1, element 2[a]), X.D.1.a (Ground 2, element 2[a]).

2. Claim 6

488. I understand that in the co-pending ITC proceedings, the respondents contend that there may be a contradiction between claims 5 and 6, because claim 5 requires using “only” three pixels in one “first” line of pixels to make a filter decision, but claim 6 requires six pixels from two lines (three pixels each) to make a filter decision.

489. I do not provide any opinion on the definiteness of this claim, but I have instead assumed that claim 5 does not foreclose claiming multiple lines of pixels when applying the prior art.

- a. **Element 6[a]: The device according to claim 5, wherein said first decision value calculator is configured to calculate said first filter decision value as: $|p_{2_0}-2p_{1_0}+p_{0_0}|+|p_{2_3}-2p_{1_3}+p_{0_3}|$ wherein p_{0_0} denotes said pixel value of said pixel closest to, in said first line of pixels, said block boundary, p_{1_0} denotes said pixel value of said pixel next closest to, in said first line of pixels, said block boundary, p_{2_0} denotes said pixel value of said pixel second next closest to, in said first line of pixels, said block boundary, p_{0_3} denotes a pixel value of a pixel closest to, in a second line of pixels in said block, said block boundary, p_{1_3} denotes a pixel value of a pixel next closest to, in said second line of pixels, said block boundary and p_{2_3} denotes a pixel value of a pixel second next closest to, in said second line of pixels, said block boundary; and**

490. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 6[a]. See Sections X.D.1.a (element 2[a]); X.C.3.b (element 5[a]); X.B.1.a (Ground 1, element 2[a]).

- b. **Element 6[b]: said second decision value calculator is configured to calculate said second filter decision value as: $|q_{2_0}-2q_{1_0}+q_{0_0}|+|q_{2_3}-2q_{1_3}+q_{0_3}|$ wherein q_{0_0} denotes said pixel value of said pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary, q_{1_0} denotes said pixel value of said pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary, q_{2_0} denotes said pixel value of said pixel in said neighboring block second next closest to, in said**

corresponding first line of pixels, said block boundary, $q0_3$ denotes a pixel value of a pixel in said neighboring block closest to, in a corresponding second line of pixels in said neighboring block, said block boundary, $q1_3$ denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding second line of pixels, said block boundary and $q2_3$ denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding second line of pixels, said block boundary.

491. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 6[b]. *See* Sections X.D.1.b (element 2[b]); X.C.3.c (element 5[b]); X.B.1.a (Ground 1, element 2[a]).

3. Claim 11

- a. Element 11[a]: The device according to claim 5, further comprising a third comparator configured to compare a sum of said first filter decision value and said second filter decision value to a threshold value, wherein**

492. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 11[a].

493. Specifically, Aksu discloses that “for filtering to be applied to any pixel, parameters d_r and d_l must both have a value greater than 1.” Aksu, 15:24-25. In other words, *after* both filter-decision values d_r and d_l are calculated, Aksu suggests using those values to make a global decision whether to perform deblocking filtering or not. Aksu suggests providing this final check because if the “d” value of one block is 1, that indicates a lot of activity in that block. However, by providing this final

check, that means a high amount of activity in one block can preclude filtering of the neighboring block with low activity. For instance, if $d_r = 1$ but $d_l = 6$, no filtering of block “l” would occur, even though there may be a benefit to providing *some* filtering of block l to provide a smoother transition into the high-activity areas of block r.

494. Upon replacing Aksu’s series of difference calculations with a Laplacian (as taught in Gonzalez), Sections X.C.1.b-X.C.1.e (elements 1[a]-[d]), a POSITA would have been motivated to not only use each Laplacian calculation to control *how many* pixels to filter (using d_r and d_l), but also to sum those decision values and compare them to a threshold based on β to decide *whether* to filter or not at all (similar to Aksu’s global decision based on evaluating d_r and d_l separately). This is exactly what H.241 and B205 do. Specifically, H.241 and B205 both provide a single equation that sums the Laplacians taken on each side of the block boundary to decide whether or not to filter any pixels:

$$d = \left| \begin{array}{c} p_{22} - 2*p_{12} + p_{02} \\ q_{05} - 2*q_{15} + q_{25} \end{array} \right| + \left| \begin{array}{c} q_{02} - 2*q_{12} + q_{22} \\ p_{25} - 2*p_{15} + p_{05} \end{array} \right| +$$

H.241, 29; B205, 130; Ex-1001, 2:5-27. A POSITA would have understood that H.241 and B205 use this cross-border equation as a measure of the overall activity near the block boundary, and proceed to filter only if $d < \beta$. *See* H.241, 29; B205, 130; Ex-1001, 2:5-27 (AAPA same). Once the overall decision to filter is then made,

further decisions would be made as to *how many* pixels to filter based on the values of d_r and d_l (calculated using a Laplacian), consistent with Aksu's disclosure of asymmetric filtering (in combination with Gonzalez). *See* Section X.C.1.b (element 1[a]).

495. In my opinion, a POSITA would have understood that modifying Aksu to use a summation of the Laplacians used for d_r and d_l would have been nothing more than a substitution of one known element (a separate comparison of d_r and d_l to a threshold) with another (a sum of d_r and d_l compared to a threshold) to obtain predictable results (a yes/no filter decision). This modification would have also been understood to be the application of a known technique (summing of Laplacians representing overall activity across the boundary) to a known device (deblocking filter) ready for improvement to yield predictable results (a yes/no filter decision).

496. A POSITA would have had a reasonable expectation of success in making this modification because it was known in the prior art that the Laplacian measures activity in pixel values, and that combining different sets of Laplacians from neighboring blocks would provide measures of either overall activity (like H.241 and B205), or block-specific activity (like Aksu). *See* Section X.C.1.b (element 1[a]). Using the values of d_r and d_l to control both *whether* and *how many* pixels to filter would have simply involved modifying the code to sum the values of d_r and d_l calculated consistent with Aksu and Gonzalez.

- b. Element 11[b]: said first pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said first line of pixels in said block to filter relative to said block boundary based on said first filter decision value calculated by said first decision value calculator; and**

497. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 11[b]. *See* Sections X.C.1.d (element 1[c], determine how many pixels in block r), X.A.3.b (element 5[a], first calculator), X.C.3.d (element 5[c], first pixel determiner), X.D.3.a (element 11[a], sum of filter decisions).

- c. Element 11[c]: said second pixel determiner is configured to determine, if said sum of said first filter decision value and said second filter decision value is below said threshold value as determined by said third comparator, how many pixels in said corresponding first line of pixels in said neighboring block to filter relative to said block boundary based on said second filter decision value calculated by said second decision value calculator.**

498. In my opinion, Aksu in view of Gonzalez, H.241, and B205 teaches or suggests element 11[c]. *See* Sections X.C.1.e (element 1[d], determine how many pixels in neighboring block 1), X.C.3.c (element 5[b], second calculator), X.C.3.e (element 5[d], second determiner), X.D.3.a (element 11[a], sum of filter decisions).

E. Ground 2C: Obviousness over Aksu in view of Gonzalez, Fu, and B118

1. Claim 10

- a. Element 10[a]: The device according to claim 5, further comprising: a first offset calculator configured to calculate a first offset Δ based on $(9x(q_{0j} - p_{0j}) - 3x(q_{1j} - p_{1j}))/16$ wherein p_{0j} denotes a pixel value of a pixel closest to, in said first line of pixels, said block boundary, p_{1j} denotes a pixel value of a pixel next closest to, in said first line of pixels, said block boundary, q_{0j} denotes a pixel value of a pixel in said neighboring block closest to, in said corresponding first line of pixels, said block boundary and q_{1j} denotes a pixel value of a pixel of said neighboring block next closest to, in said corresponding first line of pixels, said block boundary;**

499. In my opinion, Aksu in view of Gonzalez, Fu, B118, and a POSITA's general knowledge of the art teaches or suggests element 10[a].

500. Specifically, Aksu discloses that once the filter-decision values (d_r and d_l) have been determined, "filtering is carried out using a suitable filter. The invention does not limit the kind of filter that can be used." Aksu, 15:1-8. As one preferred example, Aksu describes a simple averaging filter that calculates the mean of multiple pixel values within a "filtering window" of a specified length.

501. However, a POSITA would have known that an averaging filter like Aksu's would provide strong low-pass filtering. *See* Gonzalez, 152. Using $d_r = 1$ as an example, the new value for $r_1 = (l_1 + r_1 + r_2)/3$, such that pixels l_1 , r_1 , r_2 would all be assigned the same weight. Aksu, 15:24-36. That produces what is known as a

“box” or “averaging” filter. Gonzalez, 153. I also previously provided an example of an averaging filter, which shows how it substantially suppresses high-frequency components. *See* Section V.B.2.

502. As Gonzalez explains, however, “averaging filters have the undesirable side effect that they blur edges,” “which almost always are desirable features of an image.” Gonzalez, 152. Rather than have a filter that applies equal weights to all pixel values, a better approach is to use a “weighted average,” which “giv[es] more importance (weight) to some pixels at the expense of others.” *Id.*, 153. A POSITA would have known that weighted filters maintain simplicity and efficiency sufficient for a deblocking filter, but provide far more control of the filter’s frequency response, meaning weighted-average filters provide greater control over how different frequencies are filtered. *See, e.g., id.*, 153-54 (discussing control afforded with weighted filters).

503. To that end, a POSITA would have known that the prior art was filled with deblocking filters that had coalesced around using weighted-offset equations. *See* Section X.A.6.a (Ground 1, element 10[a], walking through prior-art offset equations). Because Aksu suggests incorporating any filter equation—and, in fact, subsequent standards developments implementing Aksu’s asymmetric filter decisions (including what ultimately resulted in B118) incorporated weighted-offset equations instead of Aksu’s exemplary averaging filter, Ex-1050, 34—in my opinion,

a POSITA would have been motivated to experiment with different weighted-offset equations like those that predominated throughout the art (such as Fu, B118, or any similar equations discussed above), adjusting the coefficients applied to each pixel difference and number of pixel differences used, while guided by the general design considerations that I previously outlined. *See* Section X.A.6.a (Ground 1, element 10[a]). A POSITA would have known that these variables control the frequency response of the filter, and thus would have known how much and in what circumstance a pixel is filtered relative to its neighboring pixel values. *See id.*

504. In my opinion, a POSITA engaging in routine, empirical experimentation with these types of prior-art equations (which is how these filters were typically designed) would have recognized that only using pixel differences ($q_0 - p_0$) and ($q_1 - p_1$) would be a preferred starting point (like Fu and B118) because these equations are simpler and more computationally efficient than also including ($q_2 - p_2$).

505. When starting with these sets of pixels, the coefficients 9 and 3, divided by 16, were one obvious, possible option to choose to weight those pixel differences. First, the coefficients (9 and 3) were nothing but a scaled version of Fu's coefficients (6 and 2) by 50%. That means that the filter would be *slightly* stronger than Fu's, as Fu would apply approximately $\frac{1}{4}$ offset to the p_0/q_0 pixels, whereas the coefficients 9 and 3 would apply approximately $\frac{3}{8}$ offset. Moreover, a POSITA would have

recognized that the coefficients 9 and 3 are just one or two values different than those used by Bjontegaard (8 and 3) and B118 (8 and 2). A POSITA would have appreciated that these different coefficients produce *slight* changes in the amount of filtering applied to the p0 and q0 pixels, and thus would have been motivated to select a preferred amount of filtering:

p2	p1	p0	q0	q1	Type	9 and 3 (claims)	6 and 2 (Fu)	8 and 3 (Bjont.)	8 and 2 (B118)
200	150	150	150	150	Step (p1)	0	0	0	0
200	200	150	150	150	Step (p0)	9	6	9	6
200	200	200	150	150	Step (q0)	-19	-13	-16	-19
150	160	170	180	190	Ramp (10)	0	0	-1	1
125	150	175	200	225	Ramp (25)	0	0	-2	3
200	250	200	250	200	Alternating	38	25	34	31
255	50	50	255	255	Line	77	51	64	77
255	255	50	50	255	Line	0	0	0	0
175	200	150	150	175	Random	5	3	5	3
50	125	100	150	125	Random	28	19	25	25
75	100	150	75	125	Random	-47	-31	-42	-41

506. And a POSITA would have appreciated that the coefficients 9 and 3 met every general design requirement (discussed above) for such offset filters. *See* Section X.A.6.a (Ground 1, element 10[a]).

507. Finally, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a first-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “first offset calculator” to calculate the “first offset.”

- b. Element 10[b]: a first pixel modifier configured to modify said pixel value of said pixel closest to said block boundary in said first line of pixels by adding said first offset calculated by said first offset calculator to said pixel value of said pixel closest to said block boundary in said first line of pixels;**

508. In my opinion, Aksu in view of Gonzalez, Fu, B118, and a POSITA’s general knowledge of the art teaches or suggests element 10[b].

509. In my opinion, upon calculating the first offset, a POSITA would have followed the conventional approach of adding that offset to the pixel value for r_1 (or p_0). *See, e.g.*, Fu, 8:3-10 (calculating “delta,” then stating “ $P_0 = \text{Clip1}(p_0 + \text{delta})$ ”); B205, 131 (same); H.241, 29 (same); Bjontegaard, 20 (same); B118, 48 (same)¹⁶; Ex-1037, 207 (same); Ex-1069, 2 (same).

¹⁶ B118 labels the pixels closest to the boundary “ p_1 ” and “ q_1 ,” followed by the pixels second closest to the boundary “ p_2 ” and “ q_2 ,” and so on. For ease of

510. Aksu also discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a first-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “first pixel modifier” to modify pixel r_1 (*i.e.*, p_0).

- c. **Element 10[c]: a second pixel modifier configured to modify said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels by subtracting said first offset calculated by said first offset calculator from said pixel value of said pixel in said neighboring block closest to said block boundary in said corresponding first line of pixels;**

511. In my opinion, Aksu in view of Gonzalez, Fu, B118, and a POSITA’s general knowledge of the art teaches or suggests element 10[c].

512. In my opinion, upon calculating the first offset, a POSITA would have followed the conventional approach of subtracting that offset from the pixel value for l_1 (or q_0). *See, e.g.*, Fu, 8:3-10 (calculating “delta,” then stating “ $Q_0 = \text{Clip1}(q_0 -$

comparison throughout the analysis below, I re-label B118’s pixels, beginning with “ p_0 ” and “ q_0 ” as the pixels closest to the boundary.

delta)"); B205, 131 (same); H.241, 29 (same); Bjontegaard, 20 (same); B118, 48 (same); Ex-1037, 207 (same); Ex-1069, 2 (same).

513. Aksu also discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a first-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “second pixel modifier” to modify pixel l_1 (*i.e.*, q_0).

- d. Element 10[d]: a second offset calculator configured to calculate, if said first filter decision value is below said first threshold value as determined by said first comparator, a second offset based on $(p_0 + p_2 - 2p_1 + 2\Delta)/4$ wherein p_2 denotes a pixel value of a pixel second next closest to, in said first line of pixels, said block boundary;**

514. In my opinion, Aksu in view of Gonzalez, Fu, and B118 teaches or suggests element 10[d].

515. As with the first offset, Aksu discloses that “filtering is carried out using a suitable filter. The invention does not limit the kind of filter that can be used,” but again describes an exemplary “mean” filter applied to all pixels. Aksu, 15:1-8. Accordingly, for the same reasons I discussed above, because the selection of a filter was left to a POSITA, in my opinion, a POSITA would have been motivated to

experiment with different second-offset equations already known in the art. *See* Section X.E.1.a (element 10[a]).

516. A POSITA would have known that second-offset equations differed in their complexity, depending on how much control over the filter was desired. For instance, some simply divided the first offset by two, providing relatively low complexity, but also little control over the frequency response or accounting for the values of the pixels near p_1 . *See, e.g.*, H.241, 29 ($\Delta/2$); B205, 131 ($\Delta/2$). Others, however, used slightly more complex offsets, with the added benefit that they provided more-refined control over the second offset's frequency response and accounted for the pixel values near p_1 . Specifically, some prior-art second-offset equations would *combine* two different elements: (1) a high-pass filter component (identified in blue); and (2) a first-offset component (identified in red). *See* Fu, 8:17 (disclosing equivalent of $\Delta_{p1} = \frac{1(p2-3p1+3p0-q0)}{8} + \frac{1\Delta}{2}$); B118, 48 (disclosing equivalent of $\Delta_{p1} = \frac{1(p2-2p1+p0)}{2} + \frac{1\Delta}{2}$); *see also* Bjontegaard, 20, (disclosing equivalent of $\Delta_{p1} = \frac{5(p2-2p1+p0)}{16} + \frac{5\Delta_{p0}}{16}$); Section X.A.6.d (Ground 1, element 10[d]).

517. As I explained previously, providing these two components in a single offset equation allowed for greater control over the amount of high-frequency filtering, in addition to the filtering applied based on the first offset, by assigning

each component its own weight and thus providing more control over the filter's frequency response. *See* Section X.A.6.d (Ground 1, element 10[d]). Moreover, these prior-art, second-offset filters differed only by: (1) which high-pass filter equation was used as the high-pass filter component (*e.g.*, Fu's $p^2-3p^1+3p^0-q^0$, or B118 and Bjontegaard's $p^2-2p^1+p^0$ Laplacians); and (2) the weights assigned to each of the high-pass and first-offset components, both of which were known variables that control the filter's frequency response. *See id.*

518. If, for instance, it was desired to match Fu's level of additional filtering of the high frequencies (with its high-pass component divided by 8), the high-pass Laplacian component should be divided by 4 (or $\frac{p^2-2p^1+p^0}{4}$) because that would produce similar outputs. Or, if more of the high-pass filtering component was desired, the high-pass component could be divided by 2 (or $\frac{p^2-2p^1+p^0}{2}$), as B118 did. To demonstrate how the weight assigned to the high-pass component impacts the amount of high-frequency filtering, I have provided several example pixel configurations to show how Fu's high-pass filter component and a Laplacian divided by 4 would produce similar outputs to modify pixel p^1 , but dividing the Laplacian by 2 (like B118) would result in a larger contribution to the high-pass component:

p2	p1	p0	q0	q1	Type	Fu	Laplacian/4	Laplacian/2
200	150	150	150	150	Step (p1)	6	13	25
200	200	150	150	150	Step (p0)	-13	-13	-25
200	200	200	150	150	Step (q0)	6	0	0
150	160	170	180	190	Ramp (10)	0	0	0
125	150	175	200	225	Ramp (25)	0	0	0
200	250	200	250	200	Alternating	-25	-25	-50
255	50	50	255	255	Line	0	51	103
255	255	50	50	255	Line	-51	-51	-103
175	200	150	150	175	Random	-16	-19	-38
50	125	100	150	125	Random	-22	-25	-50
75	100	150	75	125	Random	19	6	13

519. While these high-pass components do not perform identically in every instance (particularly because the Laplacian does not use q0 as part of its calculation like Fu), these high-pass components operate extremely similarly across various pixel conditions. The only difference is precisely how much the high-frequency component is contributed to the offset (and therefore how much additional high-frequency filtering is applied in the deblocked image). As a comparison of the two Laplacians with different weights shows, by increasing the weight (from $\frac{1}{4}$ to $\frac{1}{2}$),

the p1 pixel will be filtered more, bringing it closer to its neighbors and thus filtering the higher frequencies to a greater extent than using a lower weight.

520. For the same reasons I discussed above, therefore, a POSITA would have been motivated to experiment with these two-component second-offset equations. *See* Section X.A.6.d (Ground 1, element 10[d]). In my opinion, a POSITA would have been motivated to select a Laplacian ($p_2 - 2p_1 + p_0$) as the high-frequency component because it is among the simplest second-order high-pass filters, Gonzalez, 693-97, and had already been used in the prior art as the high-pass component in second-offset equations, B118, 48. Moreover, as I discussed previously, a POSITA would have selected the weights of $\frac{1}{4}$ for the high-pass Laplacian component if it was desired to maintain similar high-frequency filtering as Fu, and $\frac{1}{2}$ for the first-offset component because that was a commonly used weight assigned to this component. *See* Fu, 8:17 (weighing $\Delta/2$); B118, 48 (same); H.241, 29 (same); B205, 131 (same); Section X.A.6.d (Ground 1, element 10[d]). Accordingly, in my opinion, a POSITA would have arrived at the claimed second-offset equation ($\Delta_{p1} = \frac{1(p_2 - 2p_1 + p_0)}{4} + \frac{1\Delta}{2}$) through routine experimentation with these two variables.

521. Finally, Aksu discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon

combining Aksu and Gonzalez, and implementing a second-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “second offset calculator” to calculate the “second offset.”

- e. **Element 10[e]: a third pixel modifier configured to modify, if said first filter decision value is below said first threshold as determined by said first comparator, said pixel value of said pixel next closest to said block boundary in said first line of pixels by adding said second offset calculated by said second offset calculator to said pixel value of said pixel next closest to said block boundary in said first line of pixels;**

522. In my opinion, Aksu in view of Gonzalez, Fu, and B118 teaches or suggests element 10[e].

523. Specifically, the Aksu-Gonzalez combination discloses filtering a second pixel r_2 when the filter-decision value d_r is less than a threshold based on β . *See* Sections X.C.2.b-X.C.2.d (element 3[b]-[d]).

524. In my opinion, upon calculating the second offset, a POSITA would have followed the conventional approach of adding that offset to the pixel value for r_2 (or p_1). *See, e.g.,* Fu at 8:15-17 (calculating “ $P_1 = \text{Clip1}(\text{p1} + \text{Clip3}(-C, C((P_0 - p_1) \times 3 + (p_2 - Q_0) + 4) >> 3))$ ”); H.241, 20 ($p_{1i} = \text{Clip1}_Y(\text{p1}_i + \Delta/2)$); B205, 131 (same); Ex-1001, 2:44-49 (same); Ex-1037, 207 (H.264 specifying “ $p'_1 = p_1 + [\text{offset}]$ ”). A POSITA would have understood the offset would be applied to the same p_1 pixel regardless of which high-pass component was included in the equation.

525. Aksu also discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a second-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “third pixel modifier” to modify pixel r_2 (*i.e.*, p_1).

- f. Element 10[f]: a third offset calculator configured to calculate, if said second filter decision value is below said second threshold value as determined by said second comparator, a third offset based on $(q_{0j} + q_{2j} - 2q_{1j} - 2\Delta)/4$ wherein q_{2j} denotes a pixel value of a pixel in said neighboring block second next closest to, in said corresponding first line of pixels, said block boundary; and**

526. In my opinion, Aksu in view of Gonzalez, Fu, and B118 teach or suggest element 10[f]. *See* Section X.E.1.d.

527. For the same reasons discussed above, because Aksu leaves the selection of a particular filter up to a POSITA, a POSITA would have been motivated to experiment with third-offset equations that existed in the art, and to select one that combined a high-pass component and first-offset component for better control of the frequency response of the third offset. *See* Sections X.A.6.f (Ground 1, element 10[f]), X.E.1.d (Ground 2, element 10[d]); B118, 48 (disclosing equivalent of $\Delta_{q1} =$

$$\frac{1(q2-2q1+q0)}{2} - \frac{1\Delta}{2}); \text{ Bjontegaard, 20 (disclosing equivalent of } \Delta_{q1} = \frac{5(q2-2q1+q0)}{16} - \frac{5\Delta_{q0}}{16}).$$

528. Although Fu discloses the equation $\Delta_{q1} = \frac{1(p0-3q0+3q1-q2)}{8} + \frac{1\Delta}{2}$, which *adds* the first-offset to Fu’s high-pass component rather than subtract it, Fu, 8:18, Fu does so because the third offset is subsequently *subtracted* from pixel q1, *id.*, 8:20 (“Q1=Clip1(q1-[Δ_{q1}])”). However, a POSITA would have understood that Fu’s equation can be re-written in its negative form as $\Delta_{q1} = \frac{1(q2-3q1+3q0-p0)}{8} - \frac{1\Delta}{2}$, which *subtracts* the first offset from the high-pass component (like element 10[f] does), as long as the third offset is then *added* to the value of q1 rather than subtracted (*e.g.*, “Q1=Clip1(q1+[Δ_{q1}])”) (like element 10[g] does). The negative form of Fu’s third-offset equation aligns with how B118 and Bjontegaard (as well as the ’841 Patent, elements 10[f] and [g]) treat the third offset, making it the more appropriate form for comparison between these equations:

	Third Offset	Application of Third Offset
Fu (Original)	$\frac{1(p0 - 3q0 + 3q1 - q2)}{8} + \frac{1\Delta}{2}$	Q1=q1-[offset]
Fu (Negative)	$\frac{1(q2 - 3q1 + 3q0 - p0)}{8} - \frac{1\Delta}{2}$	Q1=q1+[offset]
B118	$\frac{1(q2 - 2q1 + q0)}{2} - \frac{1\Delta}{2}$	Q1=q1+[offset]

Bjontegaard	$\frac{5(q_2 - 2q_1 + q_0)}{16} - \frac{5\Delta}{16}$	$Q_1 = q_1 + [\text{offset}]$
'841 Patent	$\frac{1(q_2 - 2q_1 + q_0)}{4} - \frac{1\Delta}{2}$	$Q_1 = q_1 + [\text{offset}]$

And mathematically, Fu's original and Fu's negative third-offset equations produce the same modified value of q1. A comparison of these two offset equations is provided below, in which I take the example signals from above, extend the examples to include q2 pixel values, and use Fu's first-offset values for each example signal calculated in element 10[a]:

p2	p1	p0	q0	q1	q2	Type	$\frac{1(p_0 - 3q_0 + 3q_1 - q_2)}{8} + \frac{1\Delta}{2}$	$\frac{1(q_2 - 3q_1 + 3q_0 - p_0)}{8} - \frac{1\Delta}{2}$
200	150	150	150	150	150	Step (p1)	0	0
200	200	150	150	150	150	Step (p0)	3	-3
200	200	200	150	150	150	Step (q0)	0	0
150	160	170	180	190	200	Ramp (10)	0	0
125	150	175	200	225	250	Ramp (25)	0	0
200	250	200	250	200	250	Alternating	-13	13
255	50	50	255	255	255	Line	0	0
255	255	50	50	255	255	Line	51	-51
175	200	150	150	175	200	Random	5	-5
50	125	100	150	125	100	Random	0	0
75	100	150	75	125	75	Random	13	-13

Subtracting Fu's third offset calculated by $\frac{1(p_0-3q_0+3q_1-q_2)}{8} + \frac{1\Delta}{2}$ would therefore produce the same modified pixel value for pixel q1 as *adding* the third offset calculated by $\frac{1(q_2-3q_1+3q_0-p_0)}{8} - \frac{1\Delta}{2}$.

529. For the same reasons I discussed above, it would have been obvious to select a Laplacian (but for the q block: $q_2-2q_1+q_0$) as the high-pass filter component, and apply the weights $\frac{1}{4}$ to that component, and $\frac{1}{2}$ to the first-offset component, because those would have already been selected for the "r" block, and a POSITA would have known to use a corresponding equation for the "l" block. *See* Sections X.A.6.f (Ground 1, element 10[f]), X.E.1.d (Ground 2, element 10[d]).

530. Finally, Aksu discloses a "digital signal processor" programmed or designed "to apply predetermined processing functions" to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a third-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a "third offset calculator" to calculate the "third offset."

- g. Element 10[g]: a fourth pixel modifier configured to modify, if said second filter decision value is below said second threshold as determined by said second comparator, said pixel value of said pixel in said neighboring block next closest to said block boundary in said corresponding first line of pixels by adding said third offset calculated by said third offset calculator to said pixel value of said pixel in said neighboring block**

next closest to said block boundary in said corresponding first line of pixels.

531. In my opinion, Aksu in view of Gonzalez, Fu, and B118 teaches or suggests element 10[e].

532. In my opinion, upon calculating the third offset, a POSITA would have followed the conventional approach of adding that offset to the pixel value for l_2 (or q_1). *See, e.g.*, Fu, 8:15-17 (“ $Q1 = \text{Clip1}(q_1 - [\text{offset}])$ ”); B118, 48 (“ $Q2 = q_2 + [\text{offset}]$ ”); Bjontegaard, 20 (“ $f_{\text{modified}} = f + \text{Diff}$ ”); Ex-1037, 208 (H.264 specifying “ $q'_1 = q_1 + [\text{offset}]$ ”). Although some prior art (like Fu) *subtracted* the third offset, this is the mathematical equivalent of adding its negative, which simply depends on how the equation is written but does not change the result. *See, e.g.*, Sections X.A.6.f (Ground 1, element 10[f]), X.E.1.f (Ground 2, element 10[f]) (noting equivalence of

$$Q1 = q1 - \frac{p0 - 3q0 + 3q1 - q2}{8} + \frac{\Delta}{2} = q1 + \frac{q2 - 3q1 + 3q0 - p0}{8} - \frac{\Delta}{2}.$$

533. Aksu also discloses a “digital signal processor” programmed or designed “to apply predetermined processing functions” to implement the method. Aksu, 17:19-23. In my opinion, a POSITA would have understood that upon combining Aksu and Gonzalez, and implementing a third-offset equation like those that existed in the prior art, the processor would be programmed or designed to have a “fourth pixel modifier” to modify pixel l_2 (*i.e.*, q_1).

XI. SUMMARY AND ADDITIONAL REMARKS

534. It is my opinion that Grounds 1(A–B) and 2(A–C) render the Challenged Claims unpatentable as obvious, as I explained in the paragraphs preceding.

535. I currently hold the opinions expressed in this Declaration based on the information available to me at this time. I may, however, form additional or new opinions should additional information or insights be made available, or upon further consideration.