ПРИЛОЖЕНИЕ 1

ARITHM\_MINUS.m

function output\_image = ARITHM\_MINUS(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = image\_1\_(i, j, g) - image\_2\_(i, j, g);

if output\_image\_(i, j, g) < 0

output\_image\_(i, j, g) = 0;

end

end

end

end

output\_image = uint8(output\_image\_);

else

output\_image = image\_1\_;

end

ПРИЛОЖЕНИЕ 2

ARITHM\_MULT.m

function output\_image = ARITHM\_MULT(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = image\_1\_(i, j, g) \* image\_2\_(i, j, g);

if output\_image\_(i, j, g) > 255

output\_image\_(i, j, g) = 255;

end

end

end

end

output\_image = uint8(output\_image\_);

else

output\_image = image\_1\_;

end

ПРИЛОЖЕНИЕ 3

ARITHM\_PLUS.m

function output\_image = ARITHM\_PLUS(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = image\_1\_(i, j, g) + image\_2\_(i, j, g);

if output\_image\_(i, j, g) > 255

output\_image\_(i, j, g) = 255;

end

end

end

end

output\_image = uint8(output\_image\_);

else

output\_image = image\_1\_;

end

ПРИЛОЖЕНИЕ 4

IMG\_CNORM.m

function out = IMG\_CNORM(image\_1\_)

[x1, y1, z] = size(image\_1\_);

max = -1;

for i = 1:x1

for j = 1:y1

for g = 1:z

if image\_1\_(i, j, g) > max

max = image\_1\_(i, j, g);

end

end

end

end

out = max;

ПРИЛОЖЕНИЕ 5

IMG\_L1NORM.m

function out = IMG\_L1NORM(image\_1\_)

[x1, y1, z] = size(image\_1\_);

sum = 0;

for i = 1:x1

for j = 1:y1

for g = 1:z

sum = sum + double(image\_1\_(i, j, g));

end

end

end

out = sum;

ПРИЛОЖЕНИЕ 6

IMG\_L2NORM.m

function out = IMG\_L1NORM(image\_1\_)

[x1, y1, z] = size(image\_1\_);

sum = 0;

for i = 1:x1

for j = 1:y1

for g = 1:z

sum = sum + (double(image\_1\_(i, j, g)) ^ 2);

end

end

end

out = sum ^ 0.5;

ПРИЛОЖЕНИЕ 7

LOGIC\_AND.m

function output\_image = LOGIC\_AND(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

%C10TO2(image\_1\_(i, j, g))

output\_image\_(i, j, g) = bitand(image\_1\_(i, j, g), image\_2\_(i, j, g));

end

end

end

if IMG\_CNORM(image\_1\_) == 1

output\_image = output\_image\_;

else

output\_image = uint8(output\_image\_);

end

else

output\_image = output\_image\_;

end

ПРИЛОЖЕНИЕ 8

LOGIC\_EQUAL.m

function output\_image = LOGIC\_EQUAL(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

%C10TO2(image\_1\_(i, j, g))

if image\_1\_(i, j, g) == image\_2\_(i, j, g)

output\_image\_(i, j, g) = 1;

end

end

end

end

output\_image = output\_image\_;

else

output\_image = output\_image\_;

end

ПРИЛОЖЕНИЕ 9

LOGIC\_LESS.m

function output\_image = LOGIC\_LESS(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

%C10TO2(image\_1\_(i, j, g))

if image\_1\_(i, j, g) < image\_2\_(i, j, g)

output\_image\_(i, j, g) = 1;

end

end

end

end

output\_image = output\_image\_;

else

output\_image = output\_image\_;

end

ПРИЛОЖЕНИЕ 10

LOGIC\_MORE.m

function output\_image = LOGIC\_MORE(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

%C10TO2(image\_1\_(i, j, g))

if image\_1\_(i, j, g) > image\_2\_(i, j, g)

output\_image\_(i, j, g) = 1;

end

end

end

end

output\_image = output\_image\_;

else

output\_image = image\_1\_;

end

ПРИЛОЖЕНИЕ 11

LOGIC\_NOT.m

function output\_image = LOGIC\_NOT(image\_1\_)

[x1, y1, z] = size(image\_1\_);

if IMG\_CNORM(image\_1\_) == 1

MAX = 1; %БИНАРНОЕ ИЗОБРАЖЕНИЕ

else

MAX = 255;

end

output\_image\_ = zeros(x1, y1, z);

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = MAX - image\_1\_(i, j, g);

end

end

end

output\_image = im2uint8(output\_image\_ ./ MAX);

ПРИЛОЖЕНИЕ 12

LOGIC\_OR.m

function output\_image = LOGIC\_OR(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = bitor(image\_1\_(i, j, g), image\_2\_(i, j, g));

end

end

end

if IMG\_CNORM(image\_1\_) == 1

output\_image = output\_image\_;

else

output\_image = uint8(output\_image\_);

end

else

output\_image = output\_image\_;

end

ПРИЛОЖЕНИЕ 13

LOGIC\_XOR.m

function output\_image = LOGIC\_XOR(image\_1\_, image\_2\_)

[x1, y1, z] = size(image\_1\_);

if string(class(image\_2\_)) == 'double'

image\_2\_ = zeros(x1, y1, z) + image\_2\_;

end

[x2, y2, z] = size(image\_2\_);

output\_image\_ = zeros(x1, y1, z);

if (x1 == x2) && (y1 == y2)

for i = 1:x1

for j = 1:y1

for g = 1:z

output\_image\_(i, j, g) = bitxor(image\_1\_(i, j, g), image\_2\_(i, j, g));

end

end

end

if IMG\_CNORM(image\_1\_) == 1

output\_image = output\_image\_;

else

output\_image = uint8(output\_image\_);

end

else

output\_image = output\_image\_;

end