Image Prediction

1. Prediction

The algorithm will be applied on bmp grayscale images 256x256.Let’s assume we have part of a picture for which we know the values of the pixels A, B and C. The idea is to predict the value of pixel marked with with “?”. Then we will use the prediction to encode the information.

	C	**B**	
	A	**?**	

We’re going to use the following predictors

```
0:  128
1:  A
2:  B
3:  C
4:  A + B - C
5:  A + (B - C) / 2
6:  B + (A - C) / 2
7:  (A + B) / 2
8:  jpegLS
```

Where **jpegLS**

```
min(A,B)  if C >= max(A,B)
max(A,B)  if C <= min(A,B)
A+B-C     otherwise
```

The prediction will be done the following:

- If it is the first pixel => it will always be predicted with 128.
- If the selected prediction is 128, than all the values predicted will be 128
- For the rest of the predictions:
    - If we have to predict the value from the first line we’re going to use the A predictor
    - If we have to predict the value from the first column we’re going to use the B predictor
    - Use the selected prediction

The value 128 – is taken because it is the half distance [0..255].

Example:

We have the following sample image 3x3.

The notation convetion is Matrix [Row , Column] . Starting index is 0. First element is at [0,0]

The maximum pixel value is 15. We use the A+B-C prediction:

We have to compute 2 matrices. The Prediction matrix and error prediction matrix.

Error Prediction Matrix = Original – Prediction.

I. We first predict 8 and compute Original[0,0] – Prediction[0,0] = -4

4	6	3
5	3	12
9	3	5

8		

-4		

Table 1 Original - Prediction – Error (Computed)

II. Then we continue on the first raw. We predict the previous value from the **ORIGINAL MATRIX.** We predict 4 and compute Original[0,1] – Prediction[0,1] = -4

4	6	3
5	3	12
9	3	5

8	4	

-4	2	

III. We continue with the first row.

4	6	3
5	3	12
9	3	5

8	4	6

-4	2	-3

IV. We continue next with the first column.

4	6	3
5	3	12
9	3	5

8	4	6
4		

-4	2	-3
1		

V. We continue with the last element from the first column.

4	6	3
5	3	12
9	3	5

8	4	6
4		
5		

-4	2	-3
1		
4		

VI. We continue next with the element from the Original matrix at [1,1]. Here we apply the prediction A+B-C. Where the values are always **ALWAYS** read from the original image

A = Original[1,0] = 5
B = Original[0, 1] = 6
C = Original[0,0] = 4

4	6	3
5	3	12
9	3	5

8	4	6
4	7	
5		

-4	2	-3
1	-4	
4		

VII. We continue next with the element from the Original matrix at [1,2]. Here we apply the prediction A+B-C. Where the values are always **ALWAYS** read from the original image

A = Original[1,1] = 3
B = Original[0, 2] = 3
C = Original[0,1] = 6

4	6	3
5	3	12
9	3	5

8	4	6
4	7	0
5		

-4	2	-3
1	-4	12
4		

VIII. We continue next with the element from the Original matrix at [2,1]. Here we apply the prediction A+B-C. Where the values are always **ALWAYS** read from the original image

A = Original[2,0] = 9
B = Original[1,1] = 3
C = Original[1,0] = 5

4	6	3
5	3	12
9	3	5

8	4	6
4	7	0
5	7	

-4	2	-3
1	-4	12
4	-4	

IX. Last item from the Original matrix at [2,2]. Here we apply the prediction A+B-C. Where the values are always **ALWAYS** read from the original image

A = Original[2,1] = 3
B = Original[1,2] = 12
C = Original[1,1] = 3

4	6	3
5	3	12
9	3	5

8	4	6
4	7	0
5	7	12

-4	2	-3
1	-4	12
4	-4	-7

2. Encoding

For the encoding part we're going to store the Error Matrix inside the output file. Additionally we have to store what predictor we've used. On the first 4 bits which represents what prediction method we've used. Than we're going to write 256x256 values (error matrix) inside the output file.
For writing the error matrix we have 2 options.

2.1. 9 bits per value

Each value from the error matrix will be stored on 9 bits.
The error matrix can have values between -256 and + 256

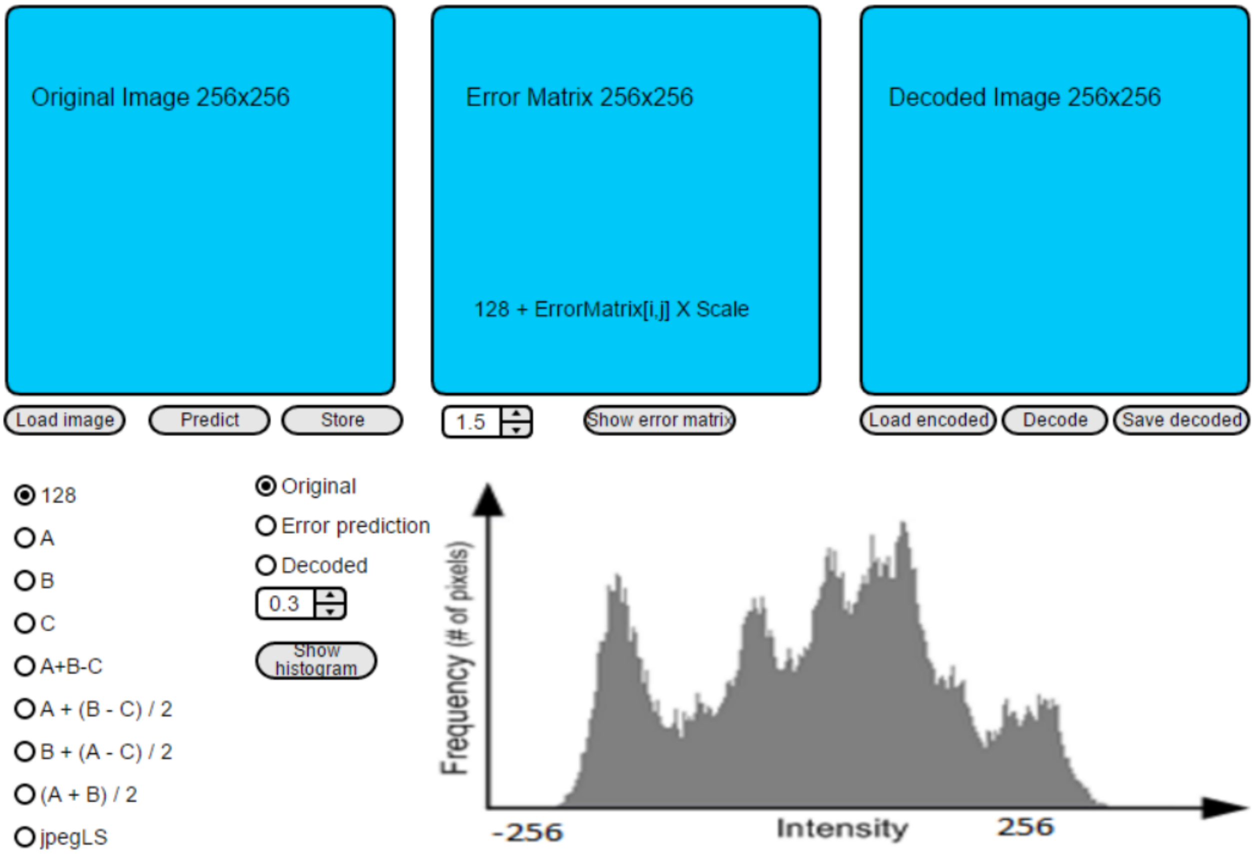
2.2. Use the following table below.

	Unary line code	Index position on line											
		0	1	2	3	4	5	6	7	...	...	...	...
0	0	0											
1	10	-1	1										
2	110	-3	-2	2	3								
3	1110	-7	-6	-5	-4	4	5	6	7				
4	11110	-15	-14	-13	...	-9	-8	8	9	...	13	14	15
5	111110	-31	-30	-29	...	-17	-16	16	17	...	29	30	31
6	1111110	-63	-62	-61	...	-33	-32	32	33	...	61	62	63
7	11111110	-127	-126	-125	...	-65	-64	64	65	...	125	126	127
8	111111110	-255	-254	-253	...	-129	-128	128	129	...	253	254	255

For e.g. .
If the value inside the error matrix is 0 than will write 0 bit to output
If the value inside the error matrix is -1 than will write 100 bit to output.
If the value inside the error matrix is 1 than will write 101 bit to output.
If the value inside the error matrix is -3 than will write 11000 bit to output.
If the value inside the error matrix is -2 than will write 11001 bit to output.
If the value inside the error matrix is 2 than will write 11010 bit to output.
If the value inside the error matrix is 3 than will write 11011 bit to output.
....

For decoding you'll have to read until the first 0 bit is encoutered. The number of 1 read before will indicate the line inside the table and also will indicate how many more bits need to be read after the 0 bit.

3. Implementation details



● 128
○ A
○ B
○ C
○ A+B-C
○ A + (B - C) / 2
○ B + (A - C) / 2
○ (A + B) / 2
○ jpegLS

● Original
○ Error prediction
○ Decoded
0.3

Load image - will load an grayscale bmp 256x256.
Predict – will compute prediction matrix and the error matrix
Store will do the following:

- Copy the first 1078 bytes from the original bmp file.
- Write another 4 BITS with the value meaning what prediction was selected
- Write the error matrix using one of the 2 options
- The output filename will be **filename.bmp[predictionNumber].pre**

Show histogram – displays the histogram for the selected matrix
Show error matrix – displays the error matrix
Load encoded – load an file with the pre extension
Decode – decodes the file and display the image on the decoded image panel
Save encoded

- will write to a file the first 1078 bytes from the encoded file.
- It will write than the pixels.
- the output filename shall be **filename.bmp[predictionNumber].pre.decoded**