GLIC - GLitch Image Codec

	**Document contains all codec internals with all of its glitch sweet spots. If you feel that it's against glitch ideas (like "glitch is a matter of chance"), read till "Encoding process" section and skip the rest of the file. You can look through to see example images.** **You were warned :)**

GLIC is a image codec designed especially for artistic databending purposes. It includes steps and techniques used in most modern codecs like h264/h265(hevc) or VP9. Therefore all of known glitch artifacts are easy to reveal plus plenty of others. Codec is designed to give more control during databending and a possibility to easy turn on and off features and encoding steps.

There are three main issues which are problematic in widely used codecs:

1. Structure and data tangled together
2. Entropy (binary coding) - responsible for final compression
3. Integrity checks

That's why you can encounter problems with decompression after databend (errors, noise, artifacts only, etc.).

Test image source used in this document: https://stocksnap.io/photo/8MX9Y2ESWB

Source code: https://github.com/GlitchCodec/GLIC
Put content of the repository in 'glic' folder.

Codec is written in Processing (works well with Processing 2 and 3) + ControlP5 library (Sketch -> Import Library... -> Add Library… -> ControlP5)
Don't forget to add more available RAM to your Processing (File -> Preferences -> Increase…). Set it to half of your RAM.

Design goals

1. Metadata, structure and image data separation (6 different blocks of data)
2. Reveal artifacts without losing image context / structure
3. Separation of encoding steps (with option to turn on/off):
    a. Color space conversion
    b. Adaptive segmentation
    c. Predictions and residuals
    d. Quantization

    e. Wavelet transformations with compression
    f. Entropy coding / binary packing
4. Binary format format prepared for hexediting - fixed position in header
5. Plenty of possibilities for each step:
    a. 16 color spaces
    b. Quadtree segmentation based on image structure with variable block sizes - from 2 to 512 pixels
    c. 13 simple prediction modes + two configurable meta predictions: angle and reference
    d. 2 ways to encode prediction residuals
    e. 67 wavelets + 2 types of transformations -> 2 * 67 variants
    f. currently no entropy coding (but 2 planned: packed RAW and RLE)
6. Extensible format - possibility to add another kind of encoding for each step
7. Color space channels separation
8. (Almost) no errors during decoding invalid data - you can always get some result
9. No integrity checks
10. Lossy and lossless
11. GUI
12. Batch processing

Additional notes:

1. This is not format to compress your images, sorry.
2. No alpha channel (can be added if necessary)
3. Sometimes you can get instant glitch or artifacts on right side and bottom - if you don't want them, image dimension should be power of 2 (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, etc…)
4. It's slow and can eat a lot of memory.
5. No deblocking / in-loop filter - sharp block edges

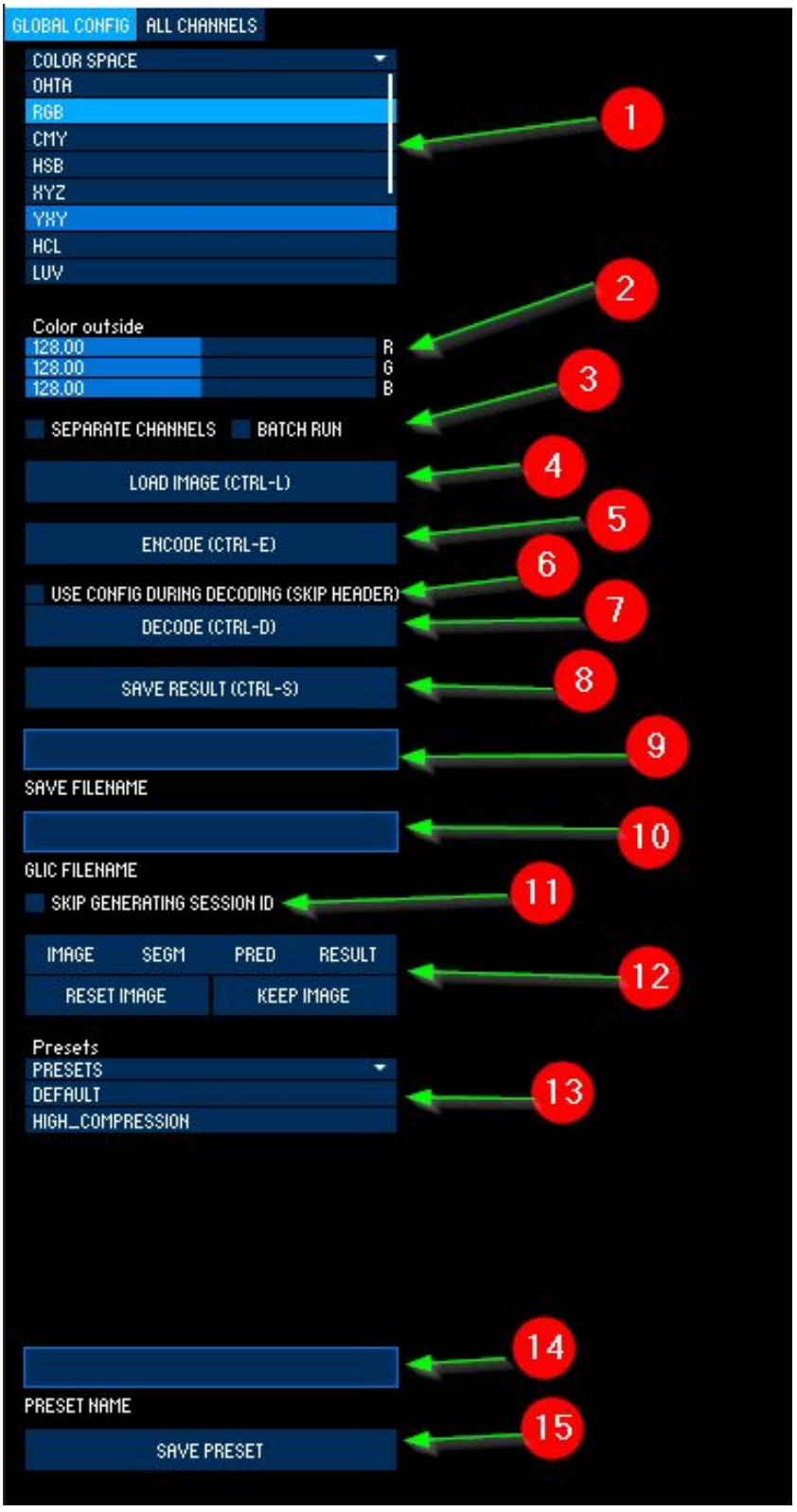
Comparison

So what is used in known and existing codecs?

1. As mentioned - segmentation, metadata, image data are usually packed together and it's hard to find sweet spot in the data during hexediting.
2. Only three color spaces are mainly used: (s)RGB (png), YUV (video codecs, webp), YCbCr (jpeg)
3. Fixed segmentation sizes for coded blocks: 1x1 pixels (png), 8x8 pixels (jpeg, webp chroma), 4x4/16x16 (webp luma)
4. Limited number of prediction modes: 5 (png), 5+1 (webp), 0 (jpeg), 2+1 (hevc)
5. Usually one transformation like DCT (jpeg, hevc) or some wavelet (dirac, snow)

GUI

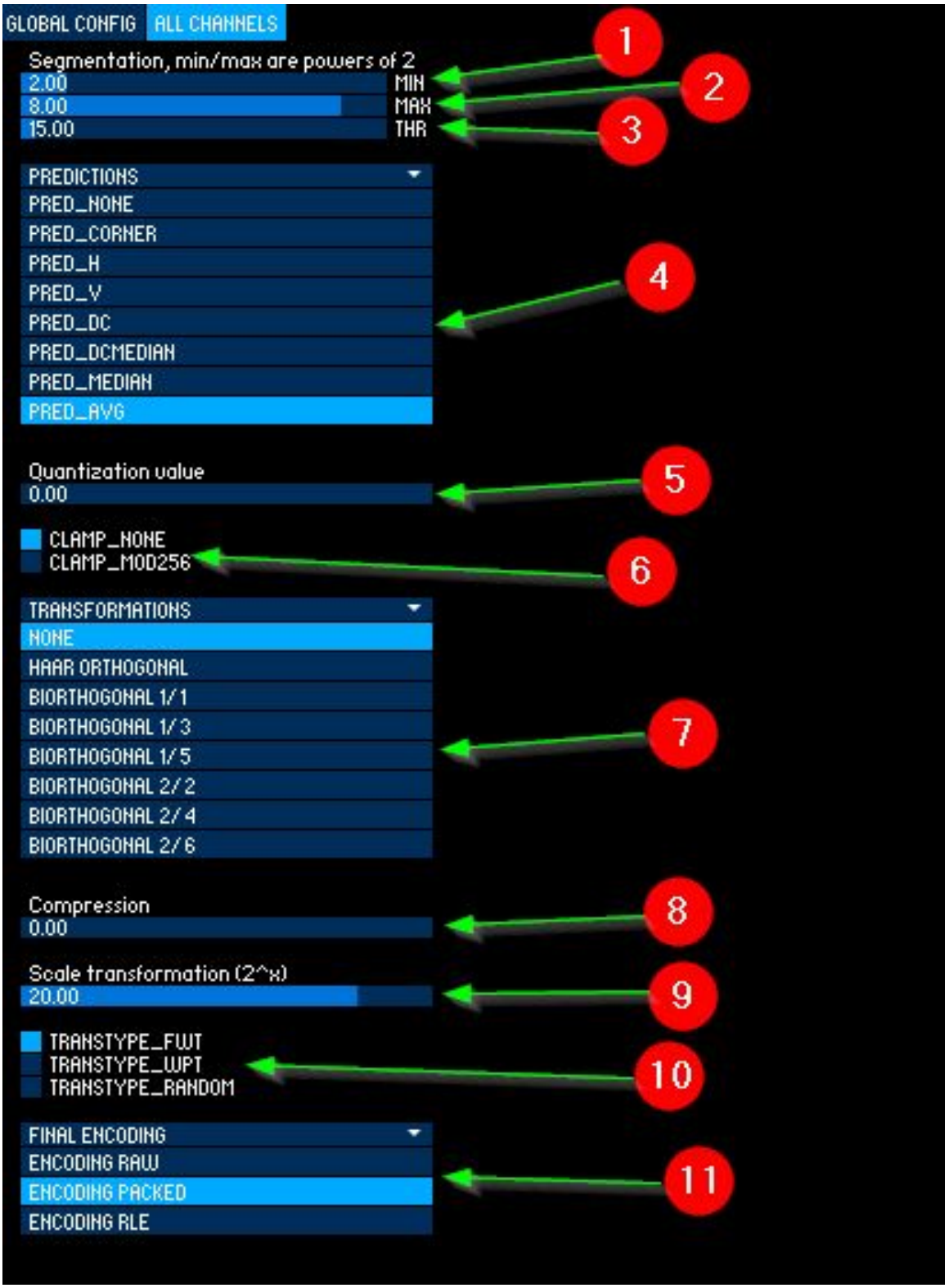
Global configuration



1. Select color space
2. Select RGB values for color outside the image (impact to prediction phase).
3. Two checkboxes:
    a. Separate channels - when checked, you can configure each color space channel separately; unchecked apply the same config to all channels
    b. Batch run - batch encoding / decoding
4. Load image from file:
    a. PNG or JPEG image
    b. GLIC file - in this case GLIC is automatically decoded and displayed
5. Encode image and save GLIC binary file with name from (10) and folder the same as original image
6. When decoding you can use configuration set in GUI instead of configuration stored in file. This is equivalent to change some header values.
7. Decode GLIC file with name from (10). Image should be encoded first!
8. Save displayed image as PNG under name from (9) and folder constructed from original folder and file name plus session_id (if used).
9. File name for results, each save (8) increments last 6 digits.
10. GLIC file name, used to save encoded image and to decode
11. When you load an image session_id (8 hex number) is created. It's used to generate filenames for resulting image and glic file. You can skip this and don't use session id in names by checking this parameter.
12. Click:
    a. Image - to see image you currently work with. It may be:
        i. image you loaded from the file
        ii. image you set via "keep image"
    b. Segm - to see segmentation after encoding/decoding
    c. Pred - to see predictions after encoding/decoding
    d. Result - to see result after encoding/decoding
    e. Reset image - use original loaded image
    f. Keep image - set image you want to work with to currently visible image
13. Encoding configuration presets list (stored in 'presets' folder)
14. Set preset name…
15. … and save to the file (it will appear in list (13))

Channel configuration

Channel configuration for encoding / decoding. Everything is explained in “Encoding process” section



1. Minimum block size (power of two, eg. value 4 -> 2^4 = 16 pixels for block side)
2. Maximum block size
3. Threshold
4. Prediction name
5. Quantization value
6. Clamping method
7. Wavelet used
8. Compression value
9. Scaling factor
10. Transformation type
11. Encoding method

How to use? Workflows.

Simple way

- Load image
- Select preset
- Encode
- Repeat: select, encode
- Save

Complicated way

- Load image
- Select preset or change something in “All channels” tab
- Encode
- Check “Use config during decoding”
- One or both:
    - Select different preset
    - Switch to “All channels” tab and change something
- Decode
- Repeat: select, encode, change something, decode
- Save

Databending

- Load image
- Select preset or change something in “All channels” tab
- Encode
- Open *.glic* file in hexeditor and edit something (see details and examples in following sections)
- Decode
- Repeat: edit in hexeditor, decode

Encoding process

Here I describe what are steps in detail and how to configure them during encoding. I mark also points where you can get instant glitch or low quality artifacts.

Color space

Image pixels are stored as a data about color. Color is just: 3 numbers which values are between 0 and 255 (sometimes there is another number which describes opacity - alpha, or blackness in CMYK). Which triple is which color is defined as color space. You probably know plenty of them like (s)RGB, CMY, HSB, YUV, YCbCr, etc. The same triple is different color in each color space.

In GLIC you can select one from 16 color spaces.

Color space	Value decimal	Value hex	Notes
OHTA	0	00	
RGB	1	01	
CMY	2	02	
HSB	3	03	
XYZ	4	04	
YXY	5	05	
HCL	6	06	
LUV	7	07	
LAB	8	08	buggy
HWB	9	09	Hue Whiteness Blackness
RGGBG	10	0A	R-G, G, B-G
YPbPr	11	0B	
YCbCr	12	0C	used in JPG
YDbDr	13	0D	
GS	14	0E	Grayscale, all channels have the same value (luma)
YUV	15	0F	

We have 3 types of colors paces:

- color based, each channels is based on some hue : RGB, CMY, XYZ(?)
- luma based, one channel describes luma (brightness), other two hue: OHTA, YXY, XYZ, LUV, LAB, RGGBG, YPbPr, YCbCr, YDbDr, GS (only luma), YUV
- hue based, one channel describes hue, the rest is intensity: HSB, HCL, HWB

Channels

Next step is channel separation. All color space channels are separated and is treated independently. This gives you opportunity to apply different rules for every channel.

Segmentation

Each channel is then cut into squares of pixels. And what is most important this structure keeps structure of the image. The idea is to use small squares for dense areas and larger squares for flat/similar color areas.
Algorithm chooses randomly 10% of block points and calculates standard deviation if it's higher than threshold parameter, block is subdivided into 4 smaller squares.

Visualization of segmentation in YXY color space:

or in HWB color space

Configuration of segmentation is based on 3 variables:

1. MIN - minimum block size (power of 2)
2. MAX - maximum block size (power of 2)
3. THR - values from 5 to 50, lower number - better precision, higher - worse.

When you use transformations you can get artifacts at bottom or right side of the image. The reasons are:

1. *Image size is not divisible by 2^MIN (example, MIN = 4, your image dimensions should be divisible by 2^4=16). Adjust image size or MIN value.*
2. *THR is too big. Lower it (15-20 is optimal)*
3. *Color outside (see below) is similar to color at boundaries. Set color outside to contrasting values 0,0,0 or 255,255,255*

Predictions

And we are at the heart of modern codecs. Prediction is a technique used to guess pixel values from surrounding (already known) pixels. In real codecs main reason is to get as much zeroes as possible. Zeroes compress best.

Process goes like that:

1. Select **prediction** algorithm
2. Guess pixels in the block using pixels from top and left side (they are already guessed)

3. Subtract guessed pixels from original pixels (this is named **residual**) and keep for further processing.

When you choose wisely you can get plenty of zeroes.

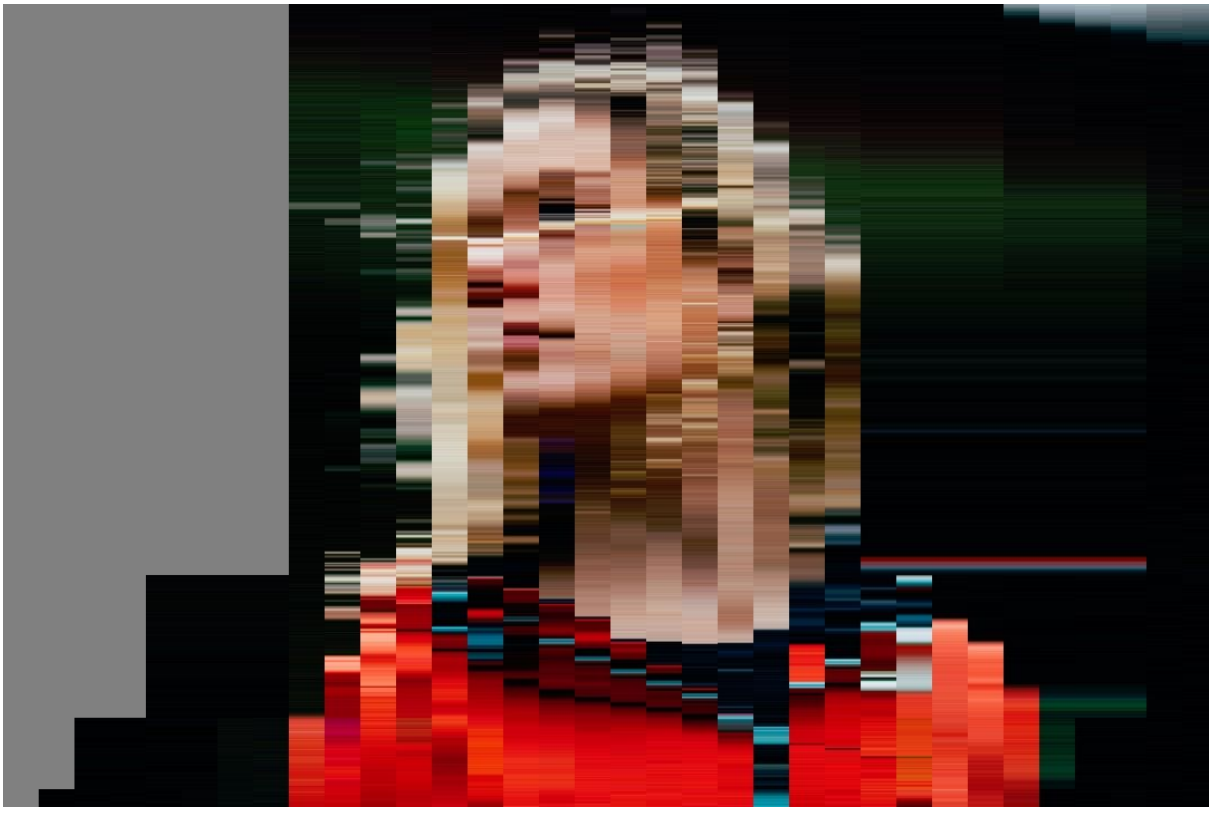
In GLIC I've implemented 15 algorithms including two special (angle based and reference based).

Name	Value decimal	Value hex	Notes
PRED_NONE	0	00	no prediction (turned off)
PRED_CORNER	1	01	
PRED_H	2	02	
PRED_V	3	03	
PRED_DC	4	04	
PRED_DCMEDIAN	5	05	
PRED_MEDIAN	6	06	
PRED_AVG	7	07	
PRED_TRUEMOTION	8	08	used in HEVC
PRED_PAETH	9	09	used in PNG
PRED_LDIAG	10	0A	
PRED_HV	11	0B	
PRED_JPEGLS	12	0C	used in JPEG-LS
PRED_DIFF	13	0D	
PRED_REF	14	0E	use other part of image as prediction
PRED_ANGLE	15	0F	
PRED_SAD	-1	--	find best for given block
PRED_BSAD	-2	--	find worst for given block
PRED_RANDOM	-3	--	random prediction

I visualize them so you can see what to expect.
Color space = RGB
Minimum block size = 5
Maximum block size = 8

PRED_CORNER (0x01)

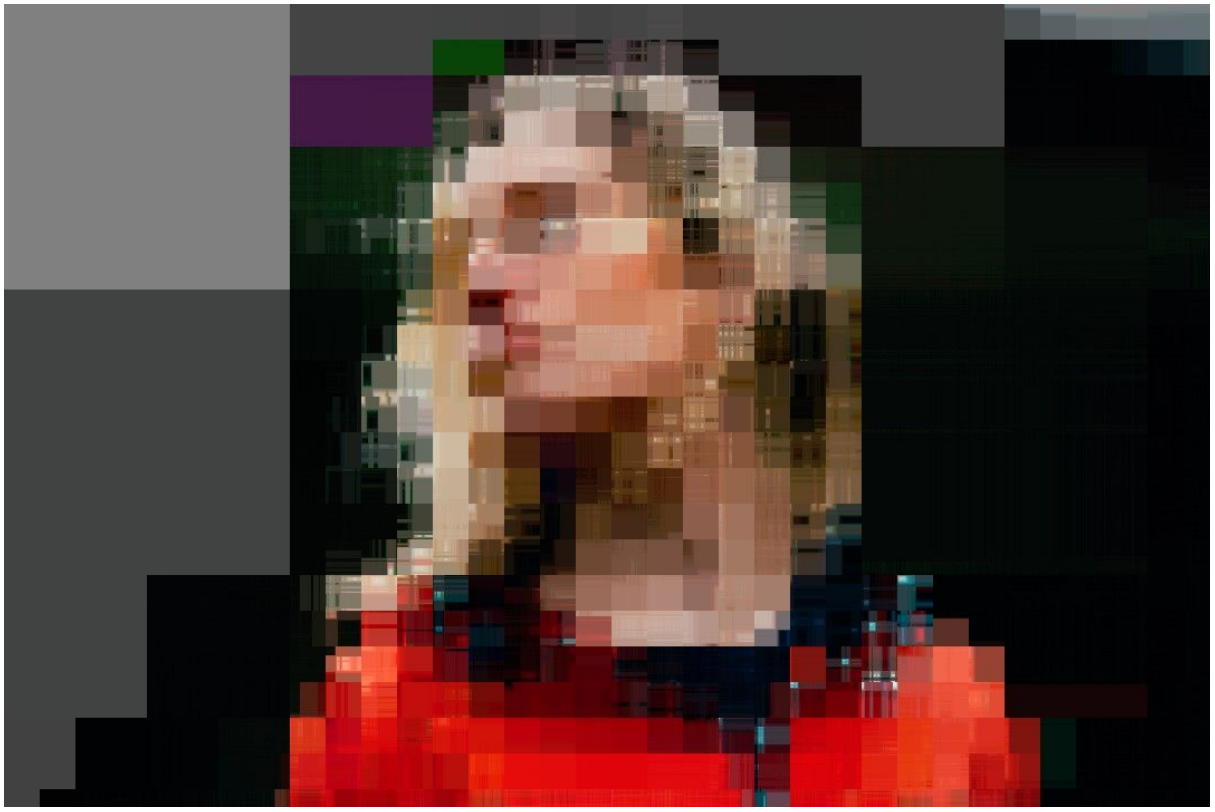
PRED_H (0x02)



PRED_V (0x03)

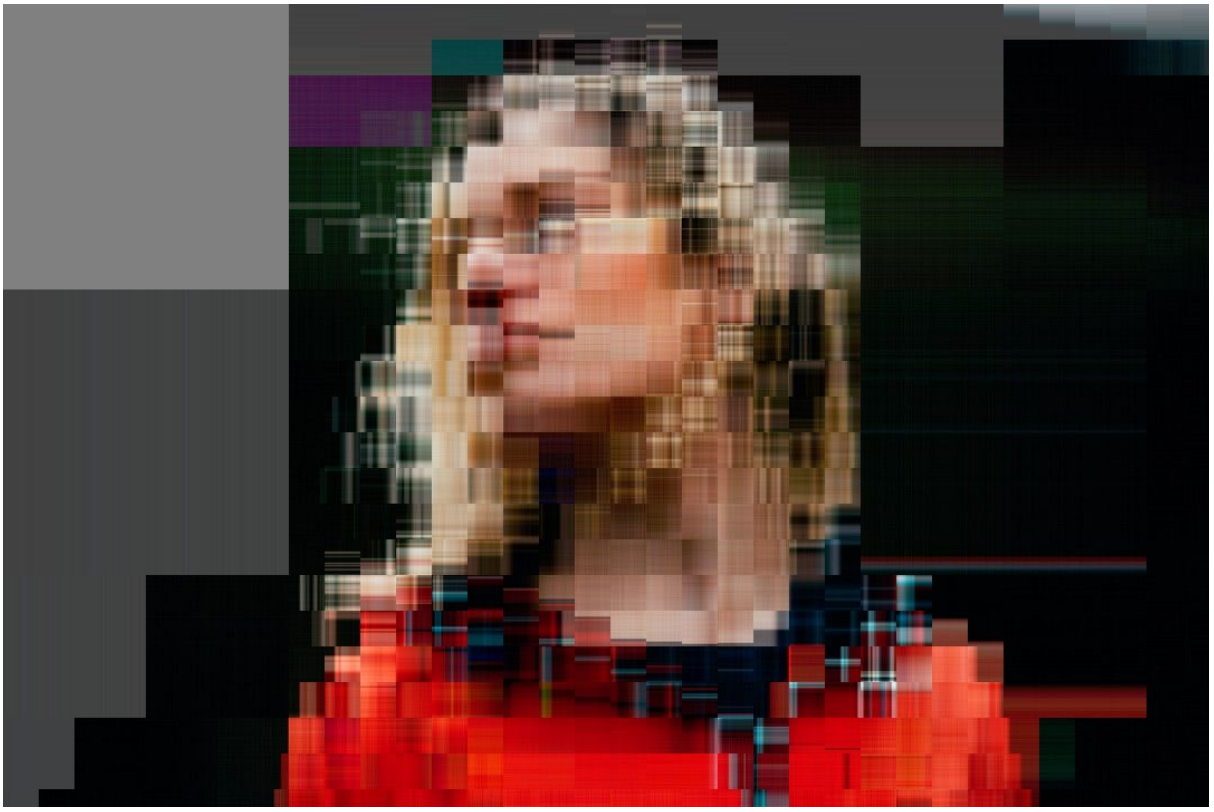
PRED_DC (0x04)

PRED_DCMEDIAN (0x05)

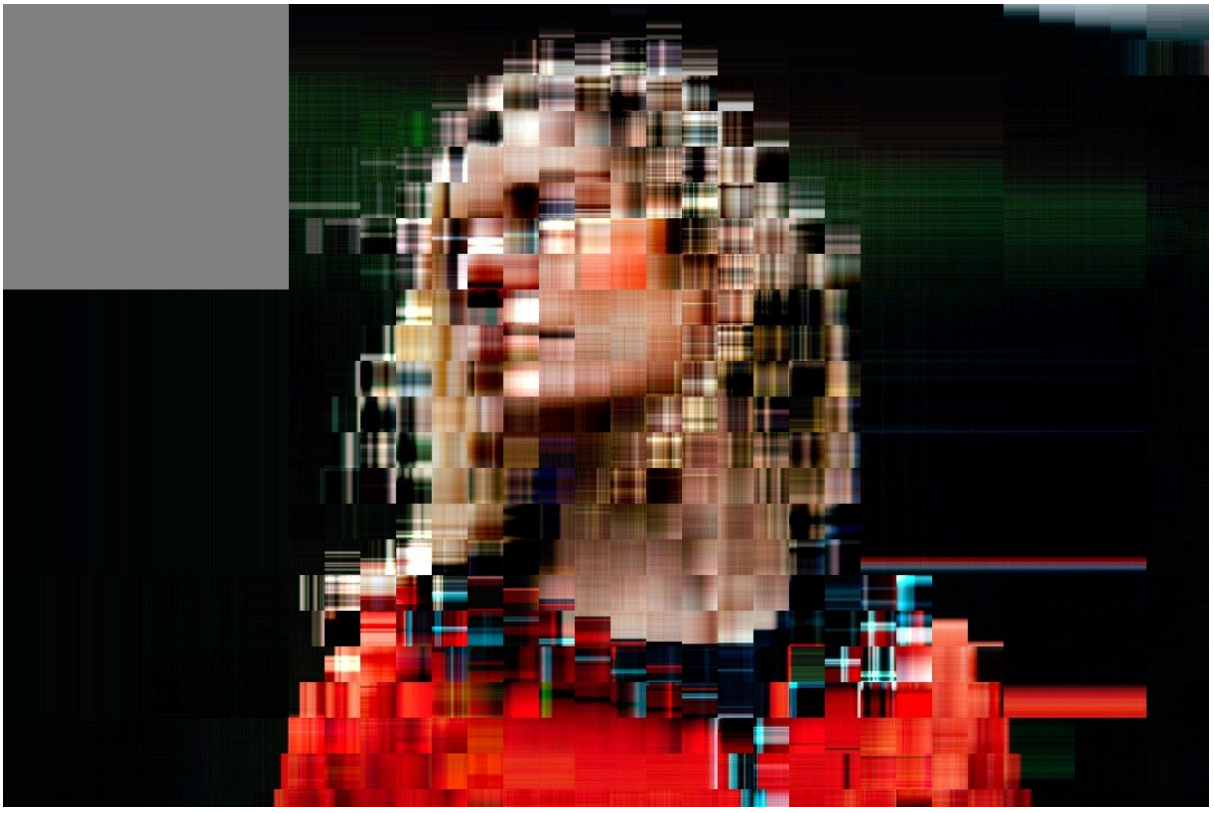


PRED_MEDIAN (0x06)

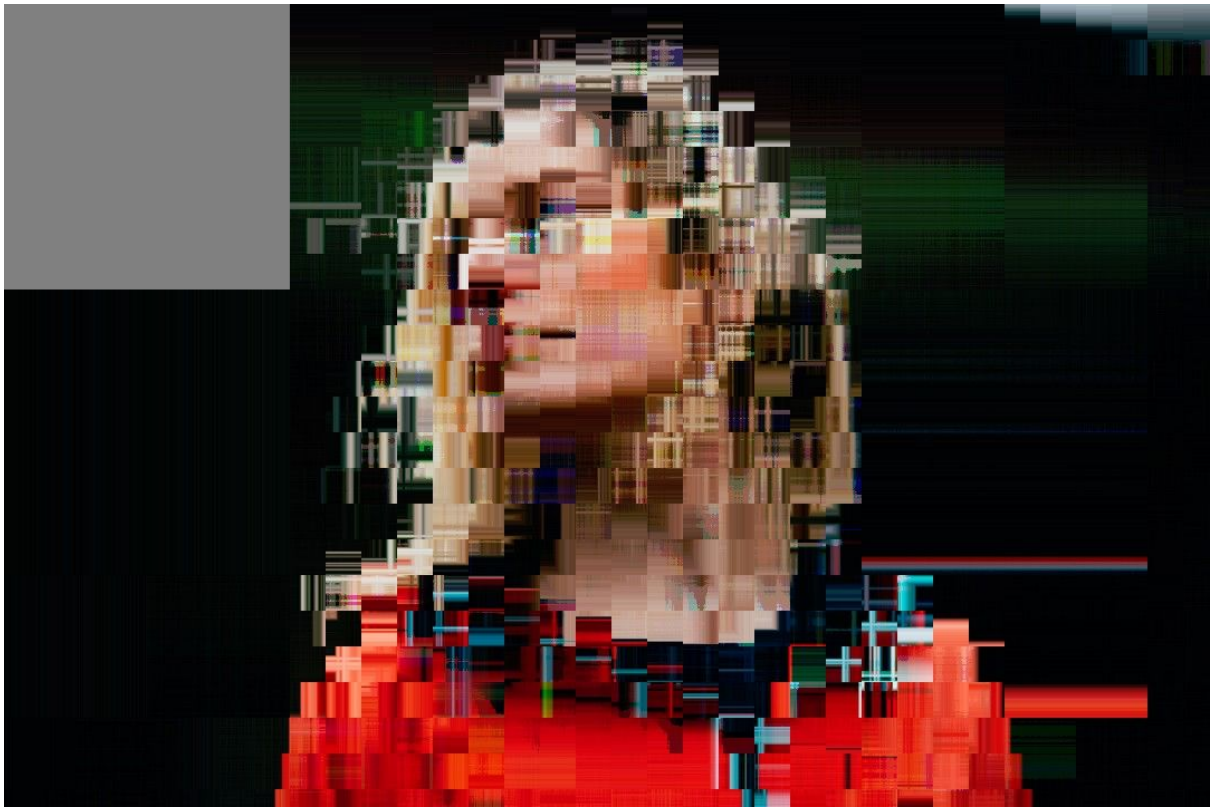
PRED_AVG (0x07)



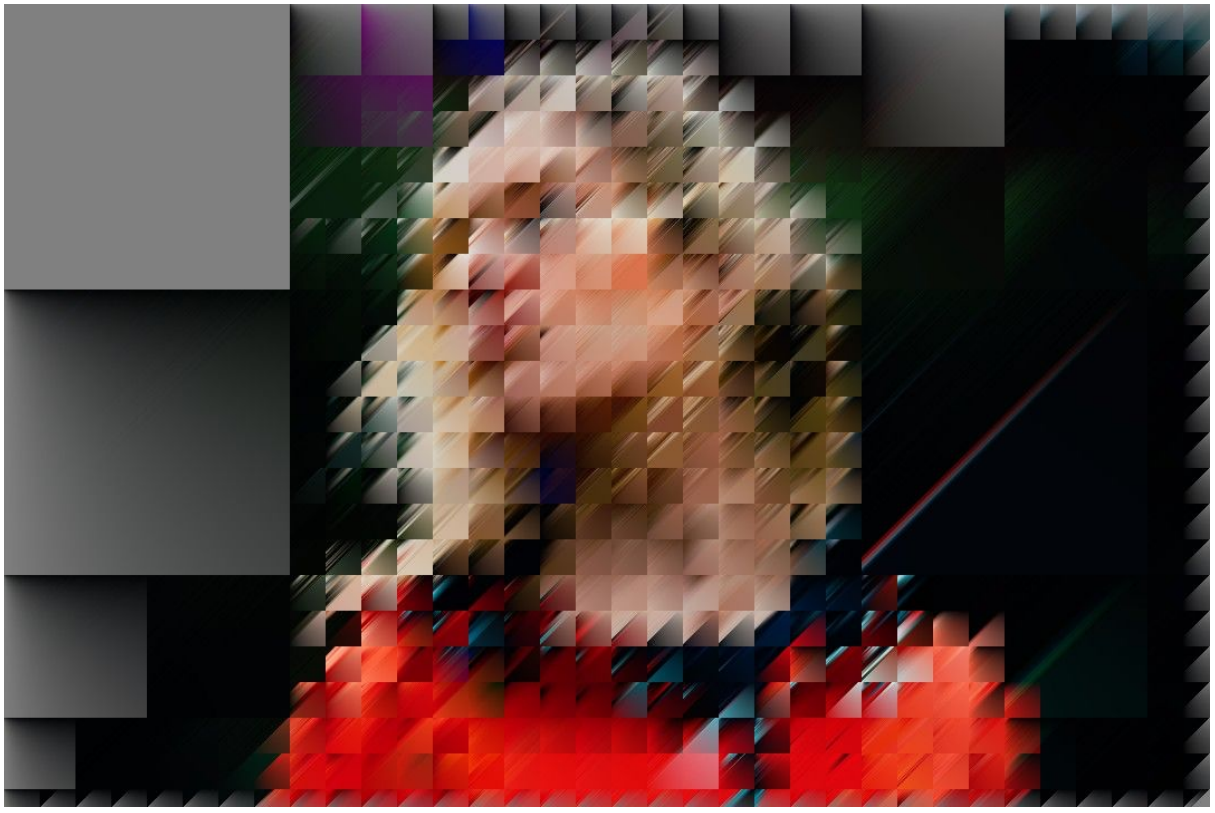
PRED_TRUEMOTION (0x08)



PRED_PAETH (0x09)

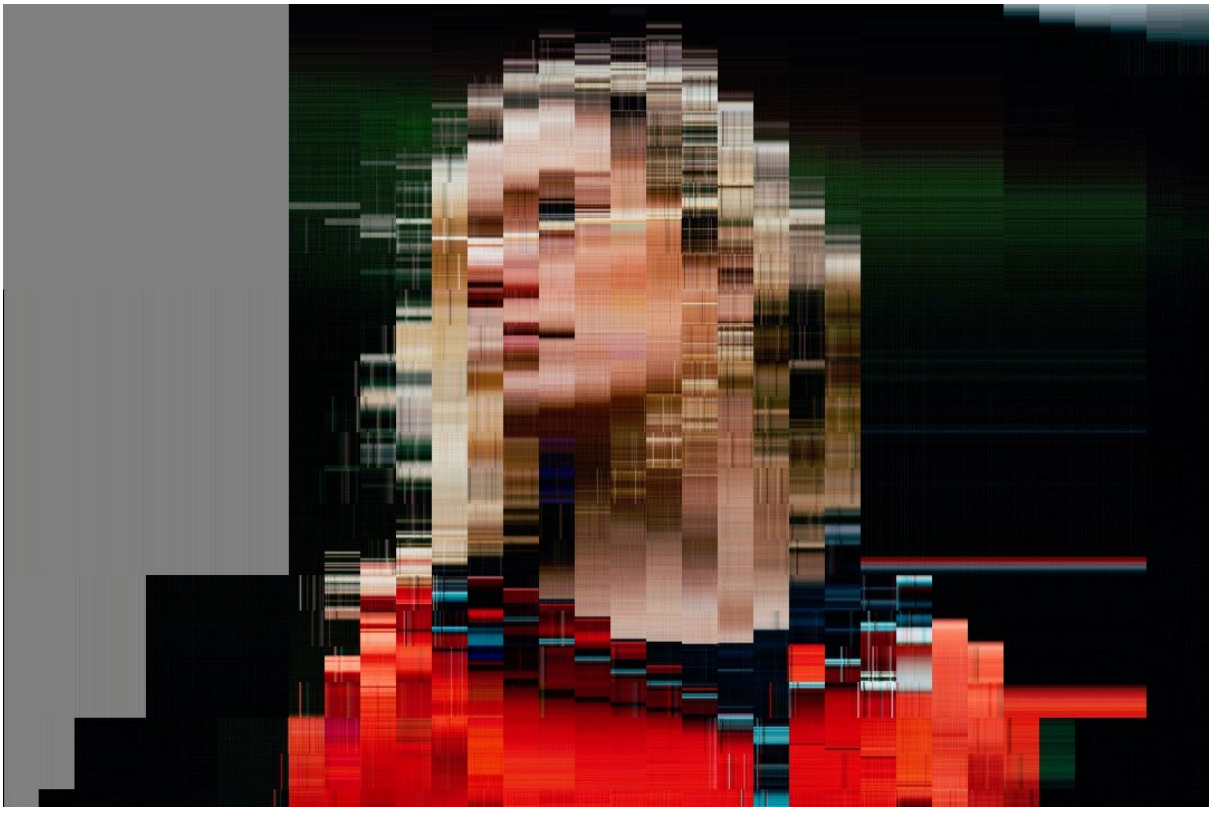


PRED_LDIAG (0x0A)

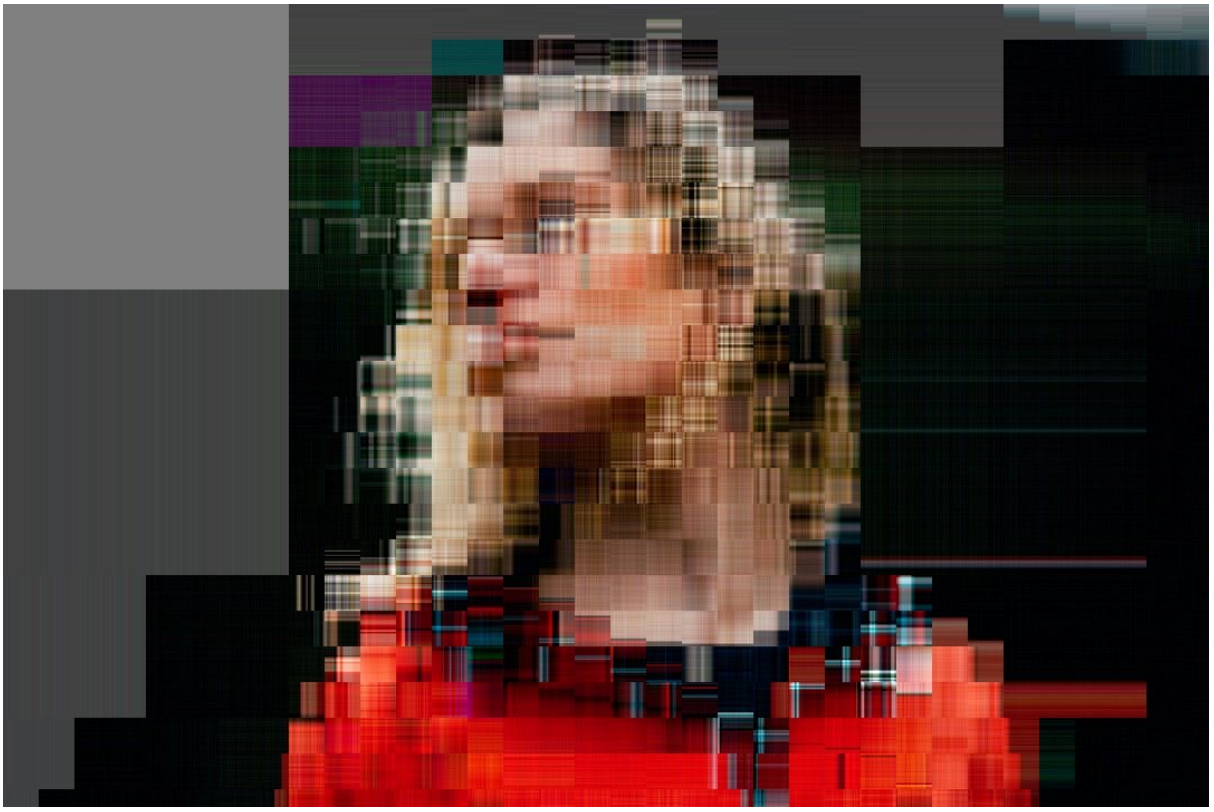


PRED_HV (0x0B)

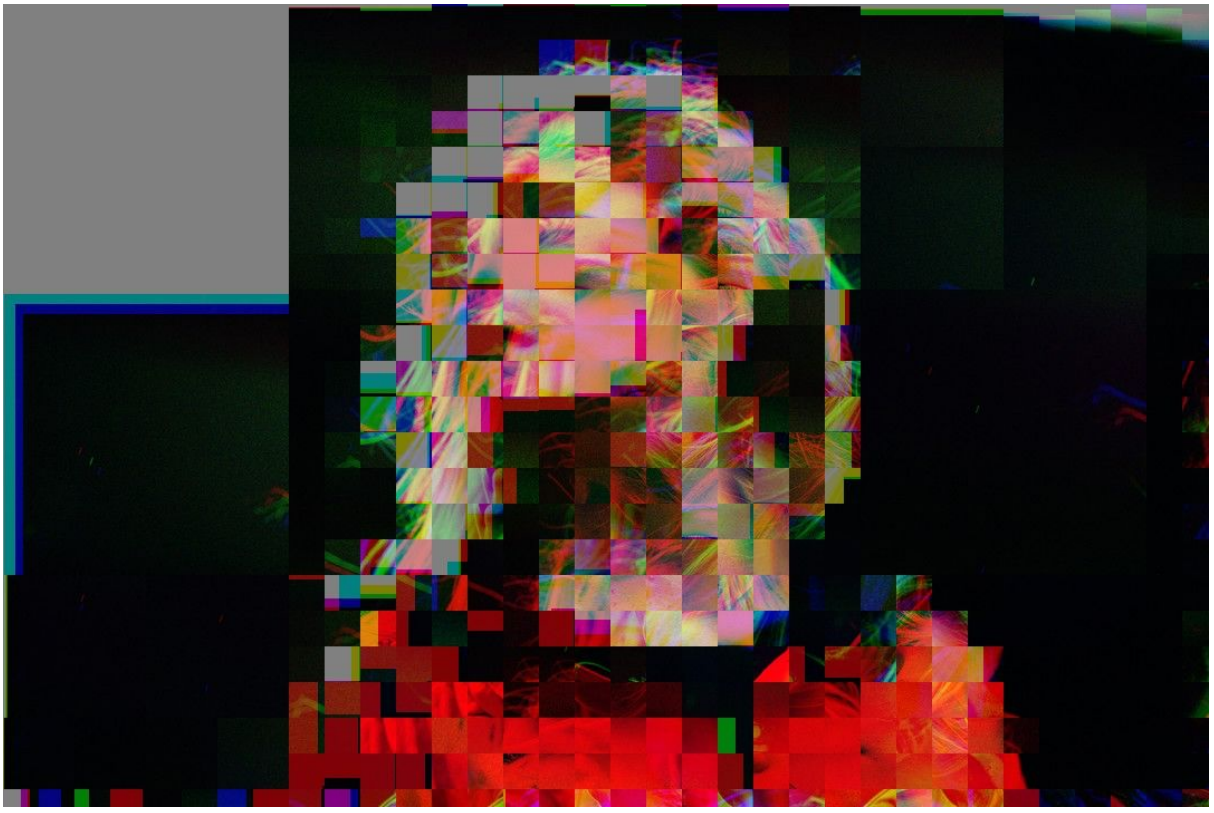
PRED_JPEGLS (0x0C)



PRED_DIFF (0x0D)



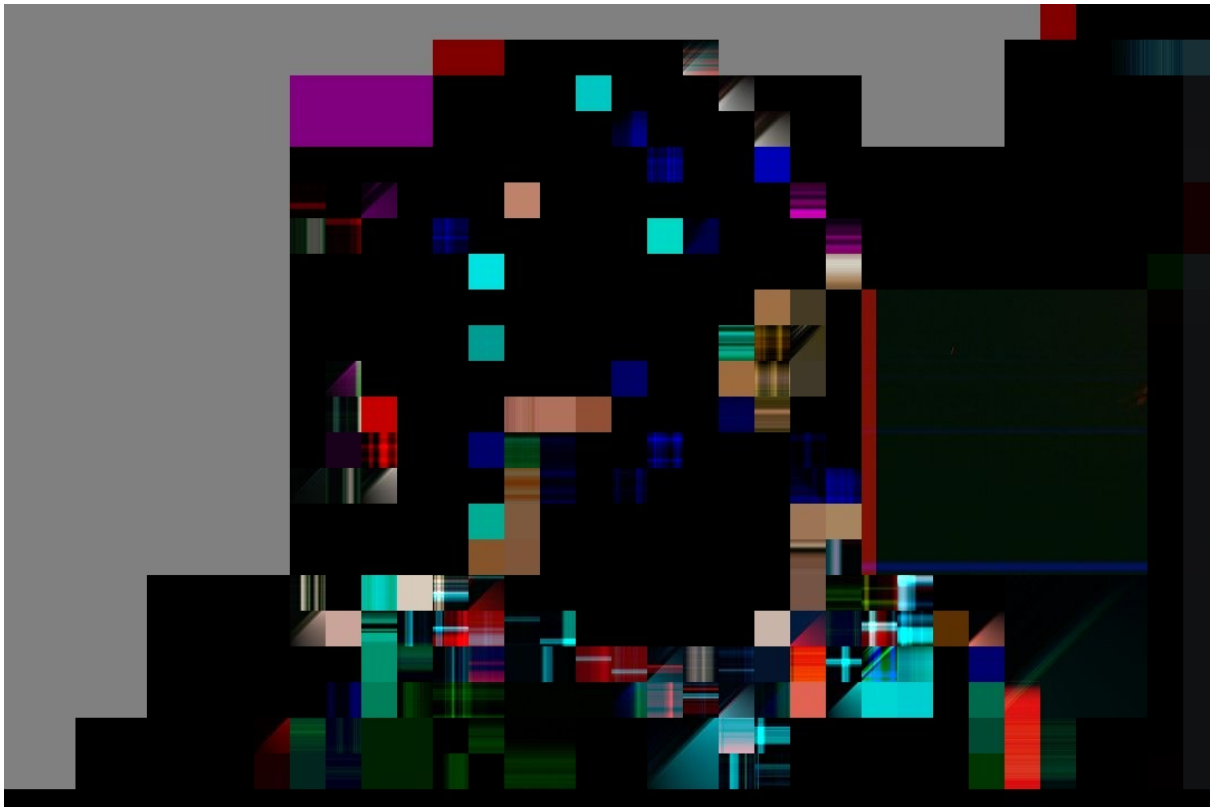
PRED_REF (0x0E)



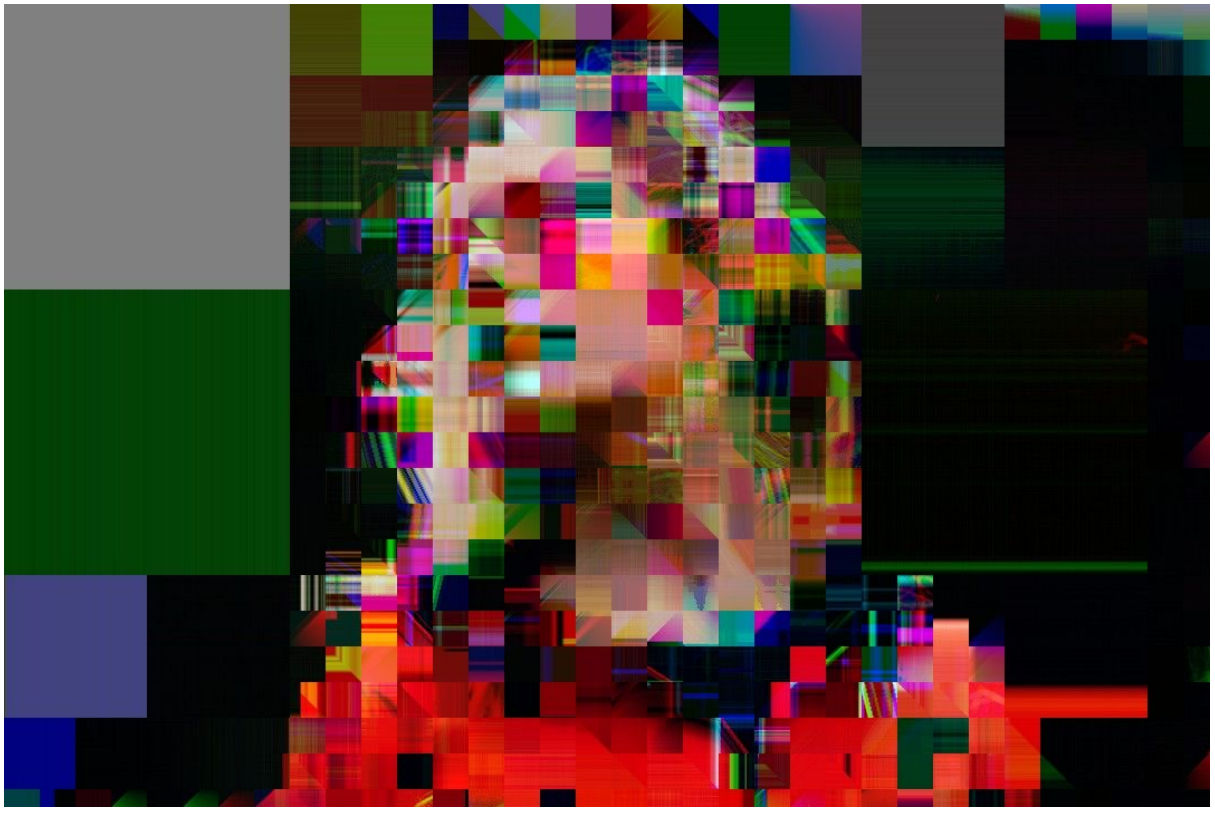
PRED_ANGLE (0x0F)

PRED_SAD

PRED_BSAD



PRED_RANDOM



Color around the image

One issue related to prediction is what happens on image boundaries. You have to define what color is outside to have prediction work better.

It’s visible on almost all above images as gray squares on the left (top) sides of the image.

Clamp method

When residuals from previous step are calculated values are from -255 to 255 and they can be left like that or encoded. Currently there are 2 options:

Clamp method	Value decimal	Value hex	Notes
CLAMP_NONE	0	00	do nothing
CLAMP_MOD256	1	01	encode to values between 0 and 255

You can see that it has huge impact for the next step.

(De)Quantization

At this stage data are quantized. It’s made by simple integer division by a number you set in “quantization value” parameter’. 0 means - no quantization

Setting high value gives instant glitch (low quality). Maximum value is 255.
During decoding data are dequantized by multiplication.

Example 1

No prediction, quantization_value = 200, color space OHTA.

Example 2

Prediction: PRED_LDIAG, color space: OHTA, clamp: CLAMP_NONE, quantization = 200

Change clamp to CLAMP_MOD256

Example 3

Prediction: PRED_DCMEDIAN, color space: YXY, clamp: CLAMP_NONE, quantization = 150

CLAMP_MOD256

Transformation

Next step is to transform block and possibly compress using wavelet transformations. There are plenty of them.

Transformation and wavelet is chosen for channel globally.
Additionally you can compress transformation which gives quality loss. Compression value is from 0 (no compression) to 255 (highest compression).

Another compression factor is scale.
Transformations give values from -1 to 1, to save it as integer I have to multiply it by some value. Lower value gives worse quality.

Transformation types

Transformation name	Value decimal	Value hex	Notes
TRANSTYPE_FWT	0	00	Fast Wavelet Transform
TRANSTYPE_WPT	1	01	Wavelet Packet Transform
TRANSTYPE_RANDOM	-1	--	random transformation

Wavelets

Below you can find list of possible wavelets, some of them are buggy and can cause instant glitch

Wavelet name	Value decimal	Value hex	Notes
WAVELET_NONE	0	00	no transformation
HAARORTHOGONAL	1	01	slight glitch
BIORTHOGONAL11	2	02	
BIORTHOGONAL13	3	03	
BIORTHOGONAL15	4	04	
BIORTHOGONAL22	5	05	glitch
BIORTHOGONAL24	6	06	glitch
BIORTHOGONAL26	7	07	glitch
BIORTHOGONAL28	8	08	glitch
BIORTHOGONAL31	9	09	
BIORTHOGONAL33	10	0A	
BIORTHOGONAL35	11	0B	
BIORTHOGONAL37	12	0C	
BIORTHOGONAL39	13	0D	
BIORTHOGONAL44	14	0E	glitch
BIORTHOGONAL55	15	0F	glitch
BIORTHOGONAL68	16	10	glitch
COIFLET1	17	11	
COIFLET2	18	12	
COIFLET3	19	13	
COIFLET4	20	14	
COIFLET5	21	15	
SYMLET2	22	16	
SYMLET3	23	17	
SYMLET4	24	18	
SYMLET5	25	19	
SYMLET6	26	1A	
SYMLET7	27	1B	
SYMLET8	28	1C	
SYMLET9	29	1D	
SYMLET10	30	1E	
SYMLET11	31	1F	
SYMLET12	32	20	
SYMLET13	33	21	
SYMLET14	34	22	
SYMLET15	35	23	
SYMLET16	36	24	
SYMLET17	37	25	
SYMLET18	38	26	

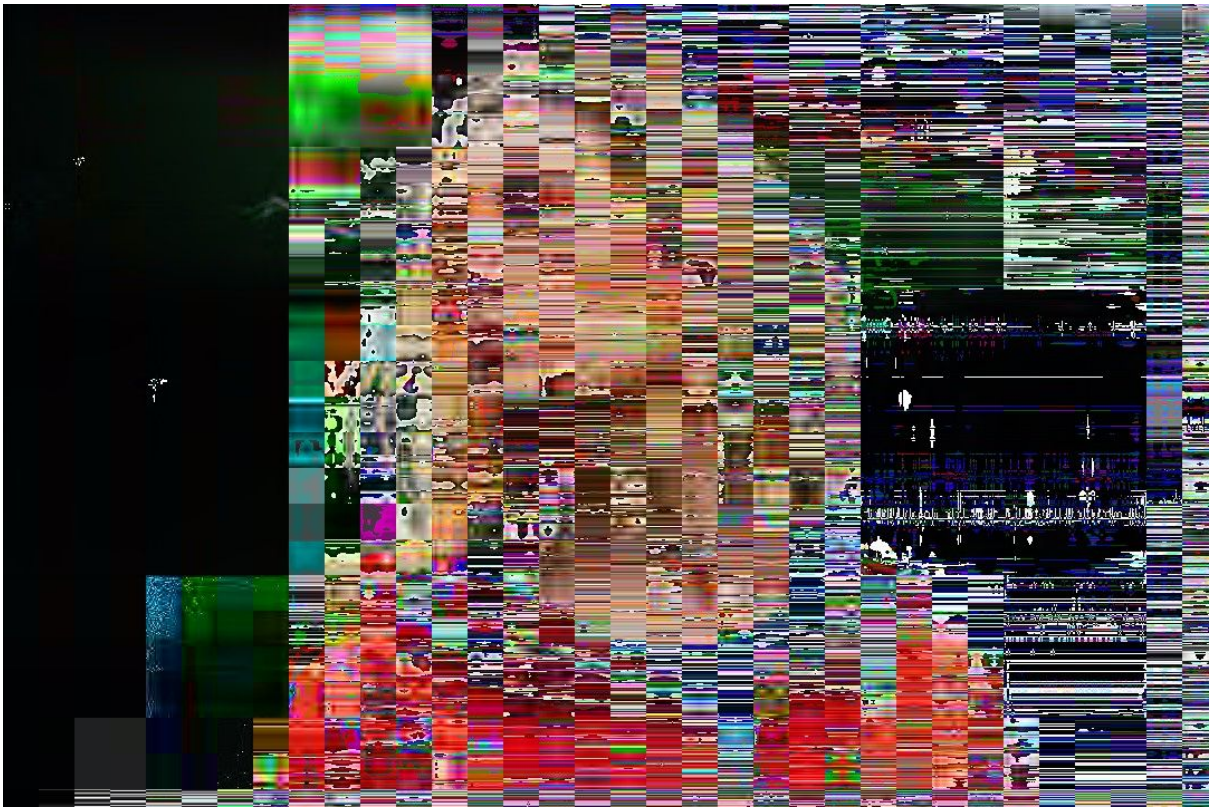
SYMLET19 SYMLET20	39 40	27 28	
LEGENDRE1 LEGENDRE2 LEGENDRE3	41 42 43	29 2A 2B	 glitch glitch
DAUBECHIES2 DAUBECHIES3 DAUBECHIES4 DAUBECHIES5 DAUBECHIES6 DAUBECHIES7 DAUBECHIES8 DAUBECHIES9 DAUBECHIES10 DAUBECHIES11 DAUBECHIES12 DAUBECHIES13 DAUBECHIES14 DAUBECHIES15 DAUBECHIES16 DAUBECHIES17 DAUBECHIES18 DAUBECHIES19 DAUBECHIES20	44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62	2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E	
BATTLE23	63	3F	doesn’t work well
CDF53 CDF97	64 65	40 41	glitch glitch
DISCRETEMAYER	66	42	
HAAR	67	43	
WAVELET_RANDOM	-1	--	random wavelet

Wavelet and compression examples

HAAR, TRANSTYPE_FWT, compression=100, no prediction, segmentation=5/8/15

SYMLET10, TRANSTYPE_WPT, compression=120, no prediction

DAUBECHIES3, TRANSTYPE_FWT, compression=100, color space=HCL, CLAMP_MOD256, PRED_H



as above but CLAMP_NONE and PRED_ANGLE

as above but quantization=150 and TRANSTYPE_WPT

HWB, Segmentation=7/8/15, PRED_REF, quantization=150, CLAMP_MOD256, TRANSTYPE_WPT, DAUBECHIES3, compress=100

Final encoding

Information about pixels is eventually encoded binary or compressed.

Name	Value decimal	Value hex	Notes
ENCODING_RAW	0	0x00	huge files
ENCODING_PACKED	1	0x01	binary packed, slightly compressed
ENCODING_RLE	2	0x02	RLE compression

Internals

ENCODING_RAW	Each value is stored as 4 byte integer (bigendian)
ENCODING_PACKED	if transformation is not used: - 8 bits for CLAMP_MOD256 (values 0-255) - 9 bits for CLAMP_NONE (values -255 - 255) if transformation is use, number of bits determined by scaling value (+1 for CLAMP_NONE)
ENCODING_RLE	First bit determines if value is repeated or not: - repeated: ‘1’ + 7 bits (up to 129 repetitions) followed by value the same as in ENCODING_PACKED - singular: ‘0’ + value

GLIC File Format

All encoded data and configuration is stored in the glic file. Structure of the file goes as below. See details and examples in ‘Decoding and databending’ section.

Headers

Below you have described meaning of GLIC file viewed in hex editor.
There are 6 sections:

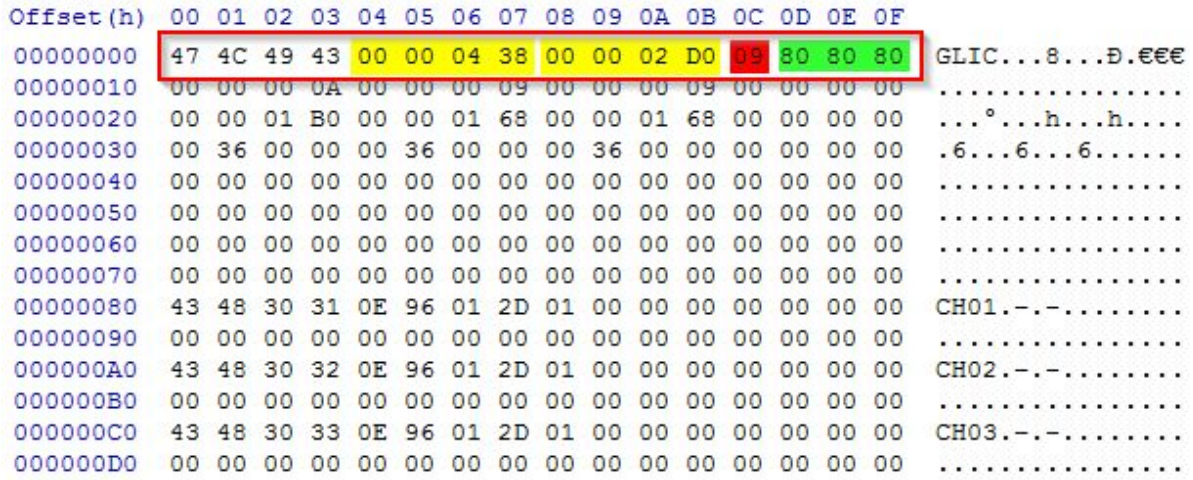
1. First header - image size, color space and color outside image (RGB)
2. Data sizes - sizes of 3 data blocks for separate for each channel:
    a. Segmentation data
    b. Prediction additional info (angle and ref)
    c. Image data
3. Second header, for each channel configuration:
    a. Global prediction method (if 0x00, prediction additional info is used)
    b. Quantization value

c. Clamp method
d. Wavelet
e. Transformation type
f. Transformation scaling value
g. Final encoding method

4. Data block with information about segmentations (3 channels)
5. Data block with information about local prediction method (prediction method, angle for PRED_ANGLE and reference data for PRED_REF) (3 channels)
6. Data block for image data (3 channels)

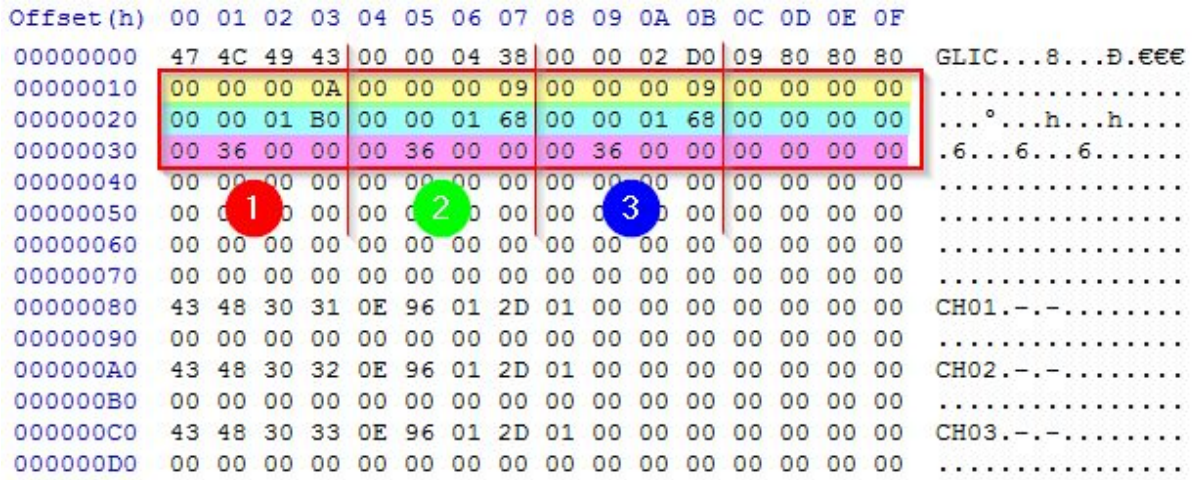
If section stores channel data, each channel section is marked with ‘CH01’, ‘CH02’ or ‘CH03’ (used in section 3-6).

First header



Yellow left - image width
Yellow right - image height
Red - color space (can be changed from GUI)
Green - color outside (r,g,b) (can be changed from GUI)

Data sizes

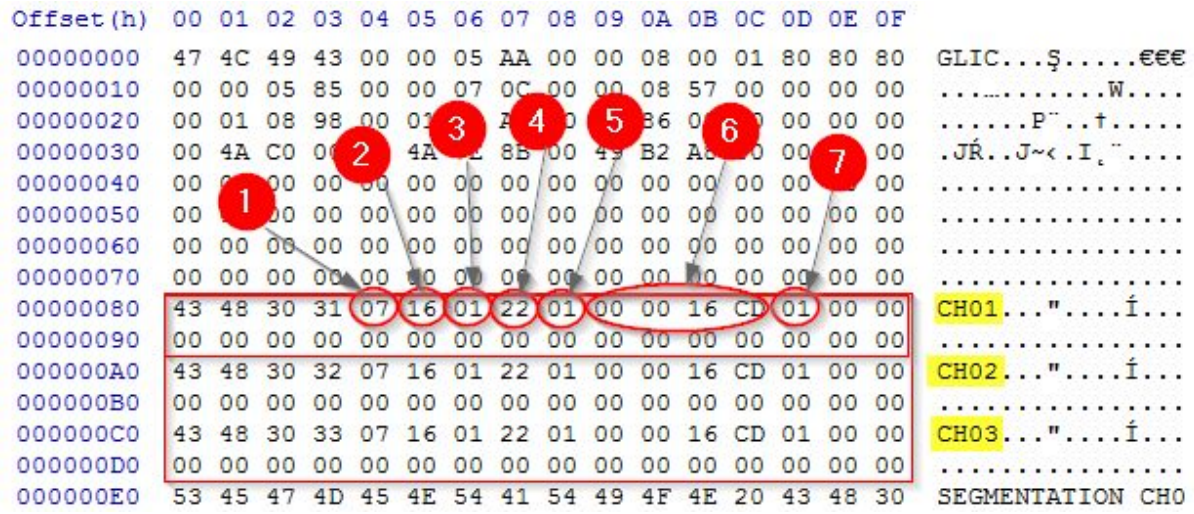


Yellow line - size of segmentation data

Blue line - size of prediction additional information data
Pink line - size of image data

Column 1 (red dot) - Channel 1
Column 2 (green dot) - Channel 2
Column 3 (blue dot) - Channel 3

Second header



For each channel marked as ‘CH01’, ‘CH02’ or ‘CH03’ you can set following configuration:

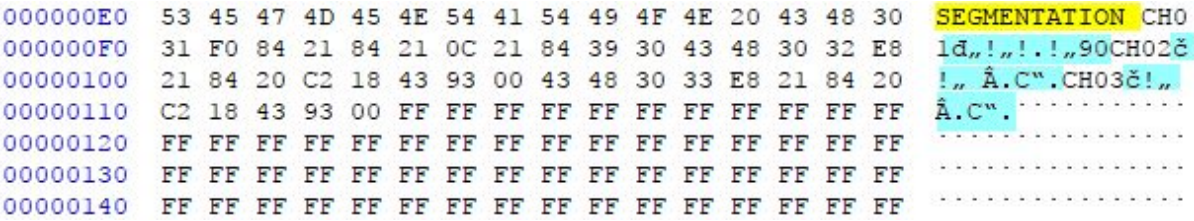
1. Global prediction method
2. Dequantization value
3. Clamp method
4. Wavelet
5. Transformation type
6. Scaling value
7. Final encoding method

All can be changed from GUI

Data

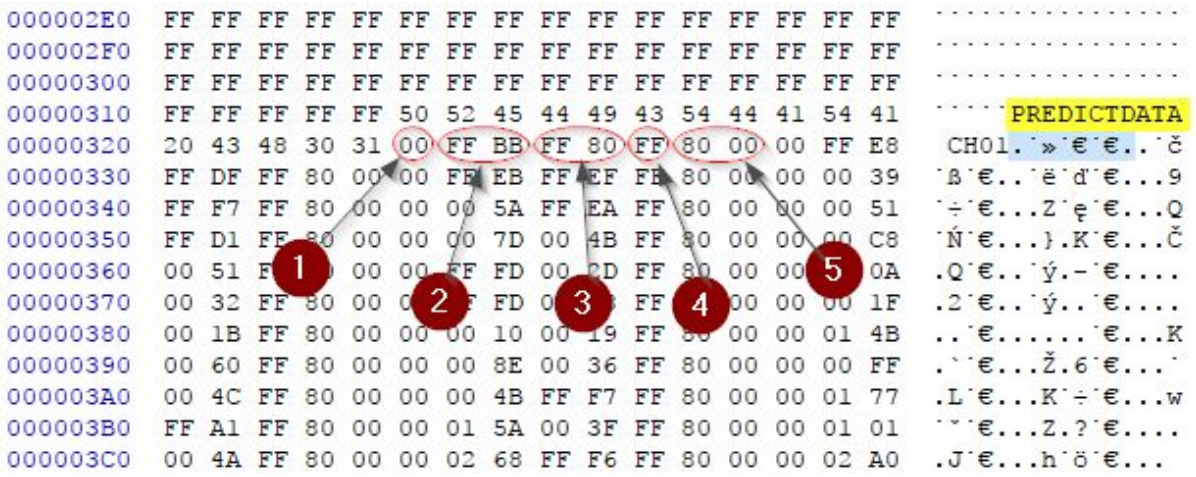
The rest of the file stores data:

Segmentation info (quadtree), marked with word: “SEGMENTATION”



Followed by 512 FF

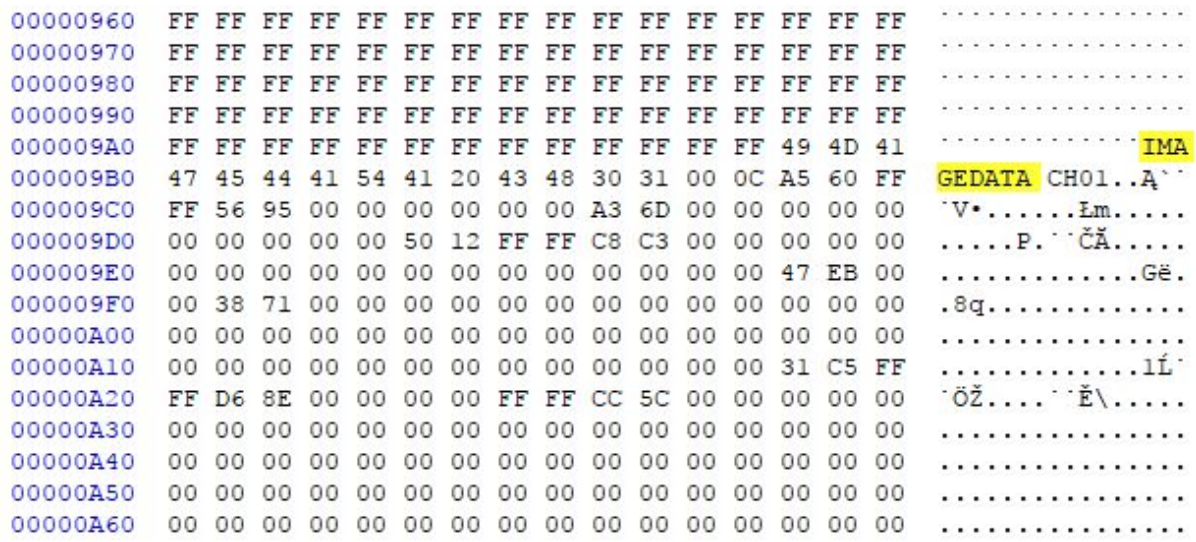
Prediction data, each marked block (5 numbers) describes one segment. Marked with word “PREDICTDATA”



1. Local prediction method (here PRED_NONE). If set to any value, replaces global prediction method set in second header
2. X position for PRED_REF
3. Y position for PRED_REF
4. Quarter for PRED_ANGLE
5. Angle for PRED_ANGLE

Block is followed by 512 FF

Finally image data encoded by encode method.



Decoding and databending

Ok, let’s start final and most interesting section. GLIC databending process. I present 3 examples here and show what results can be achieved.

Examples are:

1. image encoded without predictions and transformations
2. image encoded with predictions but without transformations
3. image encoded with predictions and transformations

No predictions, no transformations

LUV, Segmentation=2/256/15, PRED_NONE, CLAMP_NONE, WAVELET_NONE, other values set to 0.

Headers

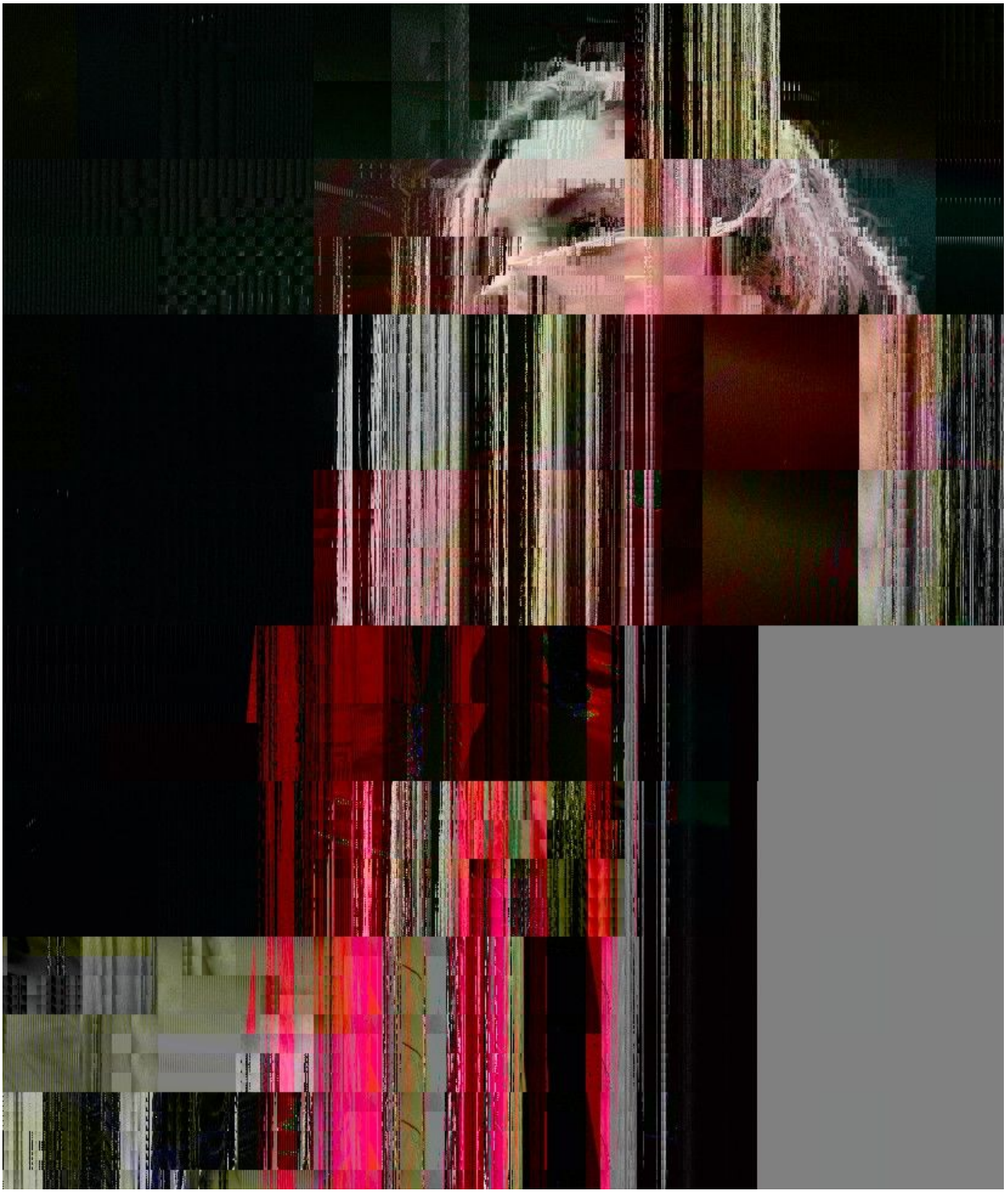
Here is original one:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   47 4C 49 43 00 00 04 38 00 00 02 D0 07 80 80 80  GLIC...8...Đ.€€€
00000010   00 00 0C 4D 00 00 01 56 00 00 00 17 00 00 00 00  ...M...V........
00000020   00 02 4C F0 00 00 3F 70 00 00 04 20 00 00 00 00  ..Lđ..?p... ....
00000030   00 30 48 00 00 39 F8 00 00 3C 00 00 00 00 00 00  .0H..9ř..<......
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080   43 48 30 31 00 00 00 00 00 00 00 00 00 00 00 00  CH01............
00000090   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0   43 48 30 32 00 00 00 00 00 00 00 00 00 00 00 00  CH02............
000000B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0   43 48 30 33 00 00 00 00 00 00 00 00 00 00 00 00  CH03............
000000D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0   53 45 47 4D 45 4E 54 41 54 49 4F 4E 20 43 48 30  SEGMENTATION CH0
```

Image sizes

```
00000000  47 4C 49 43 00 00 03 38 00 00 03 D0 07 80 80 80  GLIC...8...Đ.€€€
```



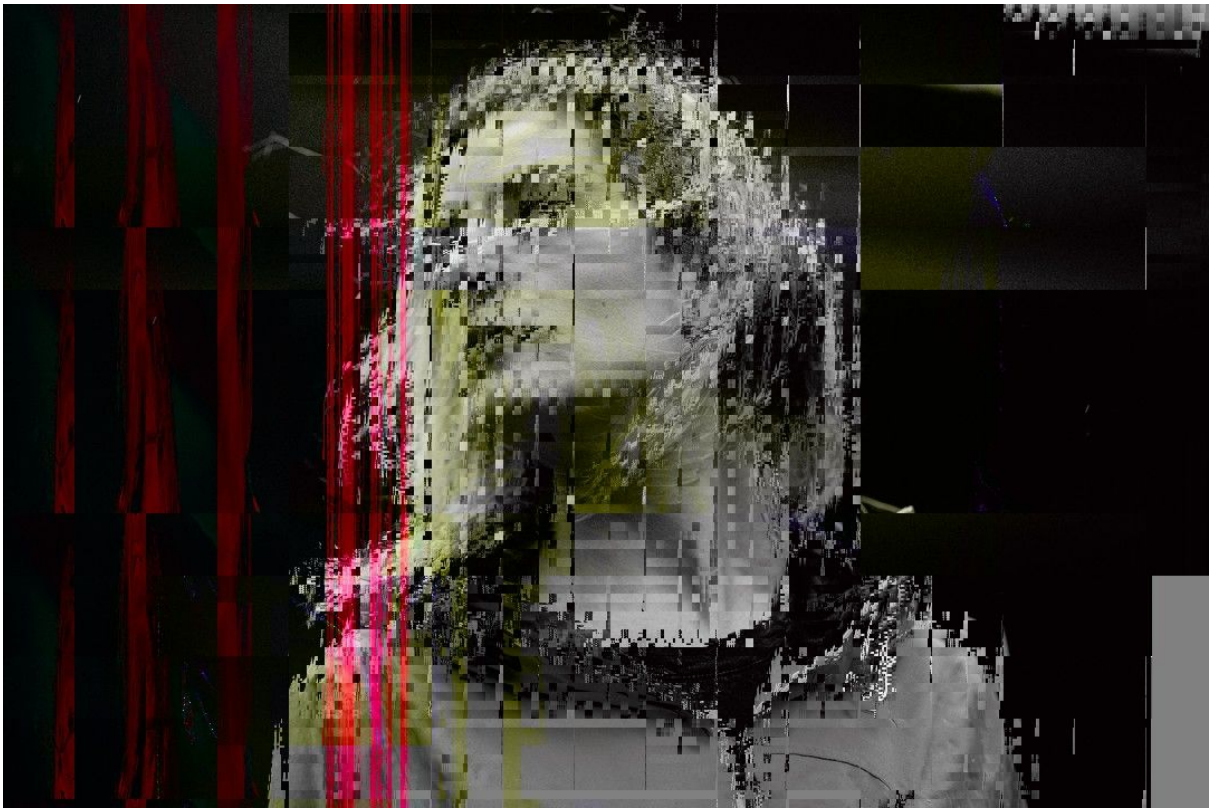
Colorspace

LUV -> XYZ

```
00000000  47 4C 49 43 00 00 04 38 00 00 02 D0 04 80 80 80  GLIC...8...Đ.€€€
```

Segmentation data sizes

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	47	4C	49	43	00	00	04	38	00	00	02	D0	07	80	80	80	GLIC...8...Đ.€€€
00000010	00	00	0C	3D	00	00	02	56	00	00	00	07	00	00	00	00	...=...V........
00000020	00	02	4C	F0	00	00	3F	70	00	00	04	20	00	00	00	00	..Ld..?p... ....
00000030	00	30	48	00	00	39	F8	00	00	3C	00	00	00	00	00	00	.0H..9ř..<......



Prediction data sizes

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	47	4C	49	43	00	00	04	38	00	00	02	D0	07	80	80	80	GLIC...8...Đ.€€€
00000010	00	00	0C	4D	00	00	01	56	00	00	00	17	00	00	00	00	...M...V........
00000020	00	02	4C	00	00	00	8F	70	00	00	04	20	00	00	00	00	..L...Źp... ....
00000030	00	30	48	00	00	39	F8	00	00	3C	00	00	00	00	00	00	.0H..9ř..<......

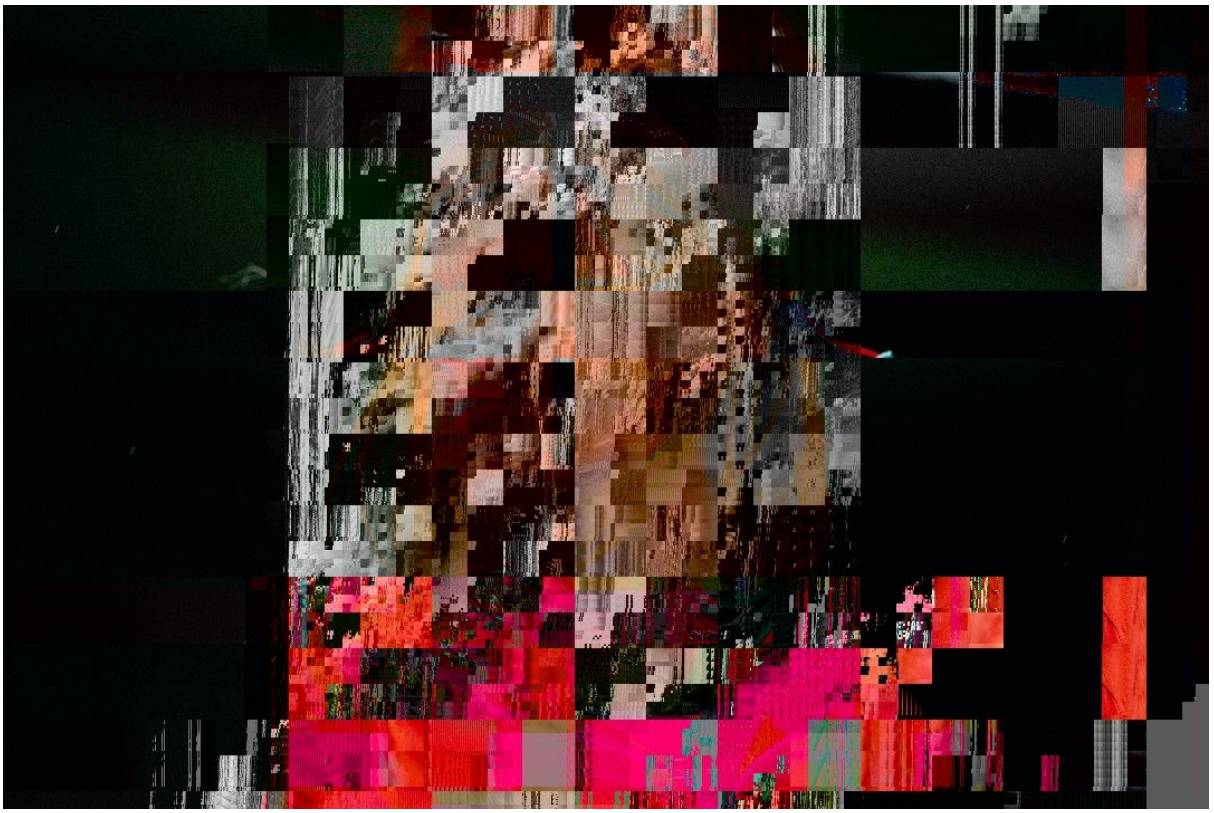


Image data sizes

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	47	4C	49	43	00	00	04	38	00	00	02	D0	07	80	80	80	GLIC...8...Đ.€€€
00000010	00	00	0C	4D	00	00	01	56	00	00	00	17	00	00	00	00	...M...V........
00000020	00	02	4C	F0	00	00	3F	70	00	00	04	20	00	00	00	00	..Ld..?p... ....
00000030	00	30	00	00	00	40	00	00	00	3C	00	00	00	00	00	00	.0...@...<......

Second header

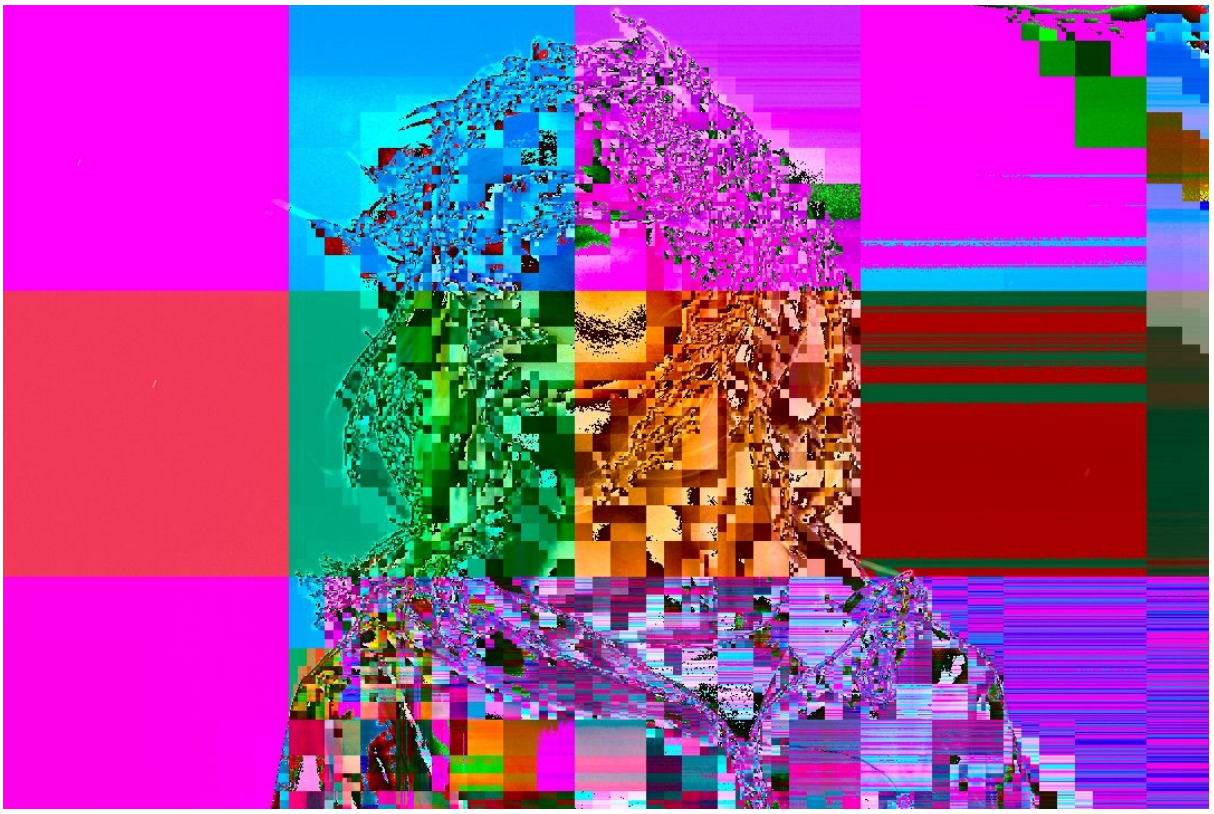
Here we can turn on predictions to see what can happen.

First let’s change global predictions to PRED_CORNER (channel 1), PRED_H (channel 2) and PRED_V (channel 3).

```
00000080  43 48 30 31 01 00 00 00 00 00 00 00 00 00 00 00  CH01............
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  43 48 30 32 02 00 00 00 00 00 00 00 00 00 00 00  CH02............
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  43 48 30 33 03 00 00 00 00 00 00 00 00 00 00 00  CH03............
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Now, let’s change clamp method to CLAMP_MOD256

```
00000080  43 48 30 31 01 00 01 00 00 00 00 00 00 00 00 00  CH01............
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  43 48 30 32 02 00 01 00 00 00 00 00 00 00 00 00  CH02............
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  43 48 30 33 03 00 01 00 00 00 00 00 00 00 00 00  CH03............
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```



and set quantization

```
00000080  43 48 30 31 01 00 01 00 00 00 00 00 00 00 00 00  CH01............
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  43 48 30 32 02 40 01 00 00 00 00 00 00 00 00 00  CH02.@..........
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  43 48 30 33 03 40 01 00 00 00 00 00 00 00 00 00  CH03.@..........
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Unfortunately you can’t do anything with transformations

Segmentation data databend

Let’s just change some of data in “SEGMENTATION” section for channels 1 and 2

Prediction data databend

Plenty of changes, nothing special here. Works better for bigger segmentation.

Image data databend

It’s raw image (without predictions and transformations).

Predictions set, no transformations

HCL, PRED_SAD, quantization = 10, rest as previously

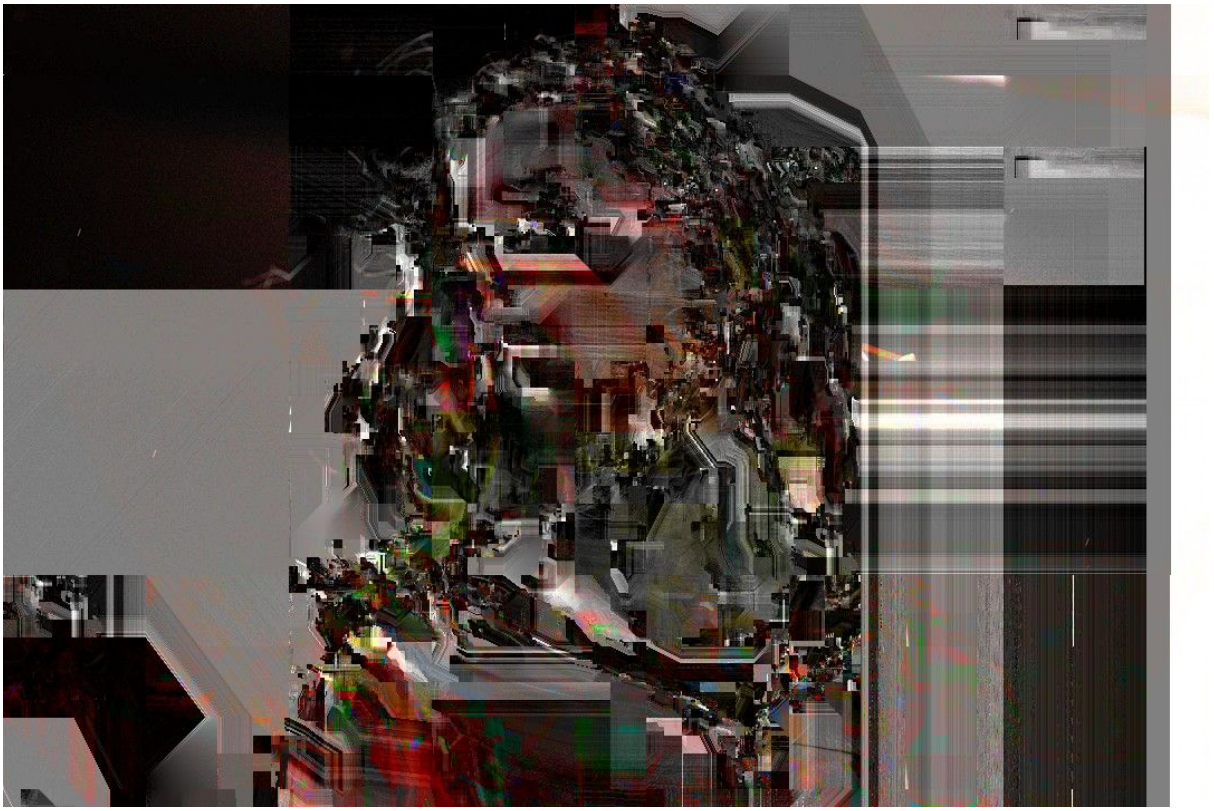
Headers

Image sizes

Colorspace

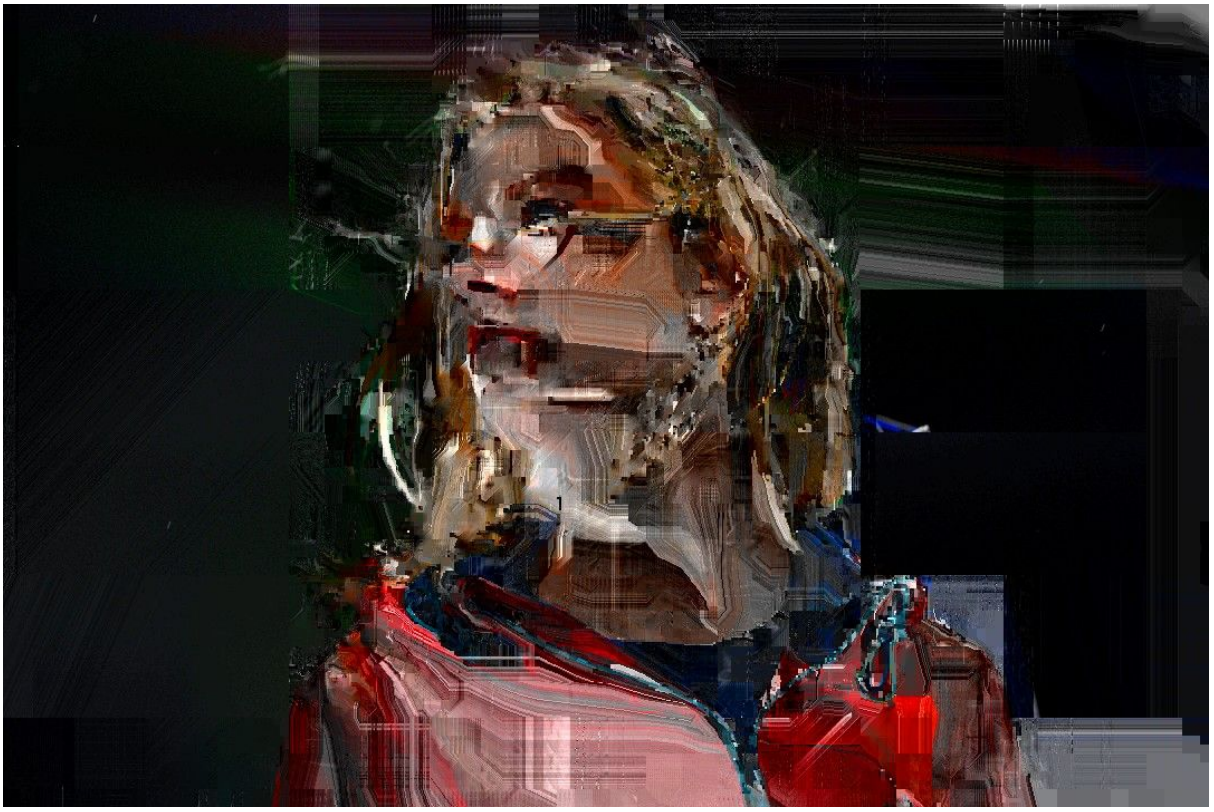
HCL -> HWB

Segmentation data sizes



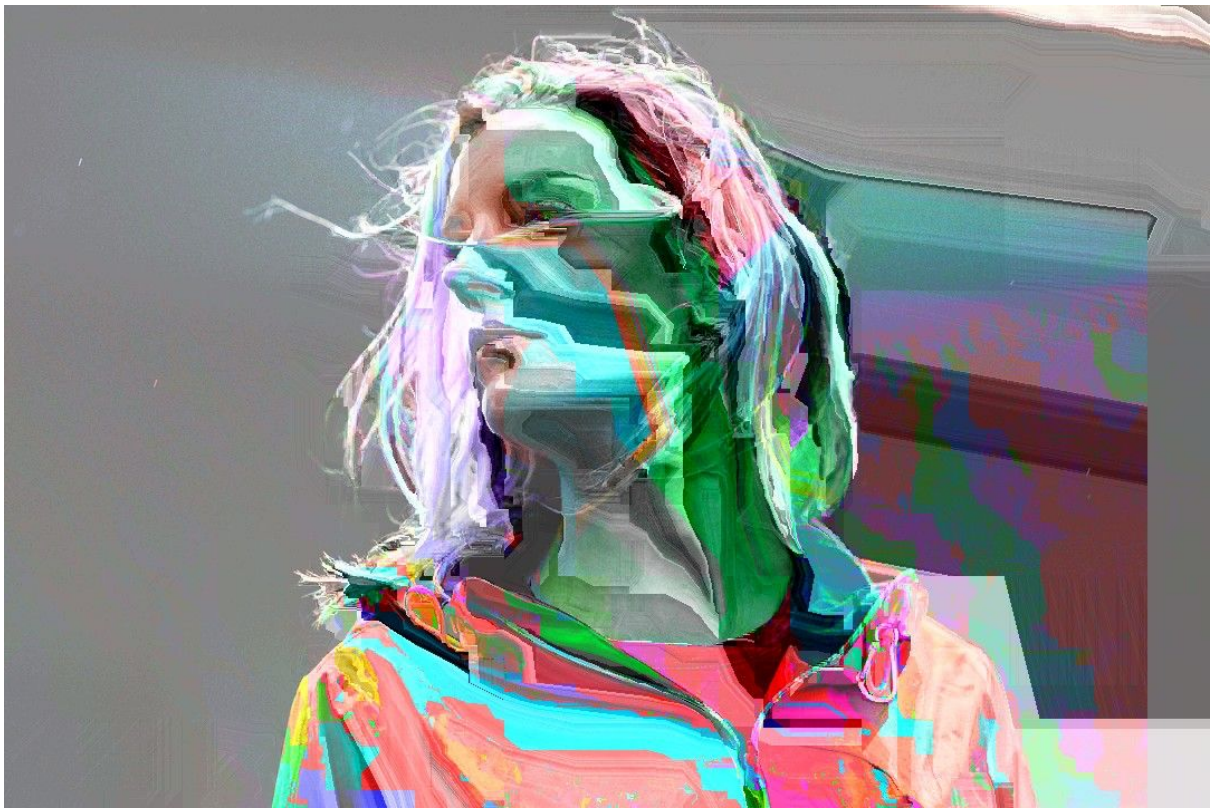
Prediction data sizes

Image data sizes

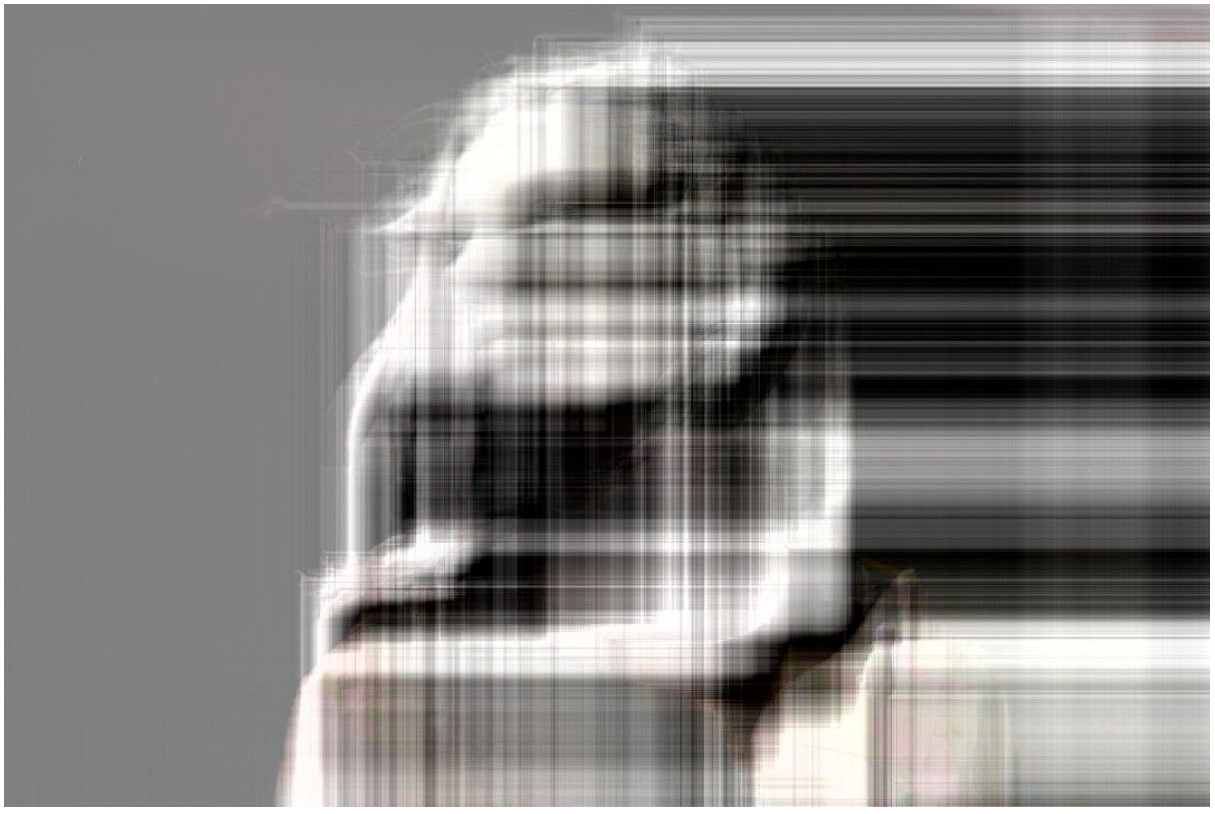


Second header

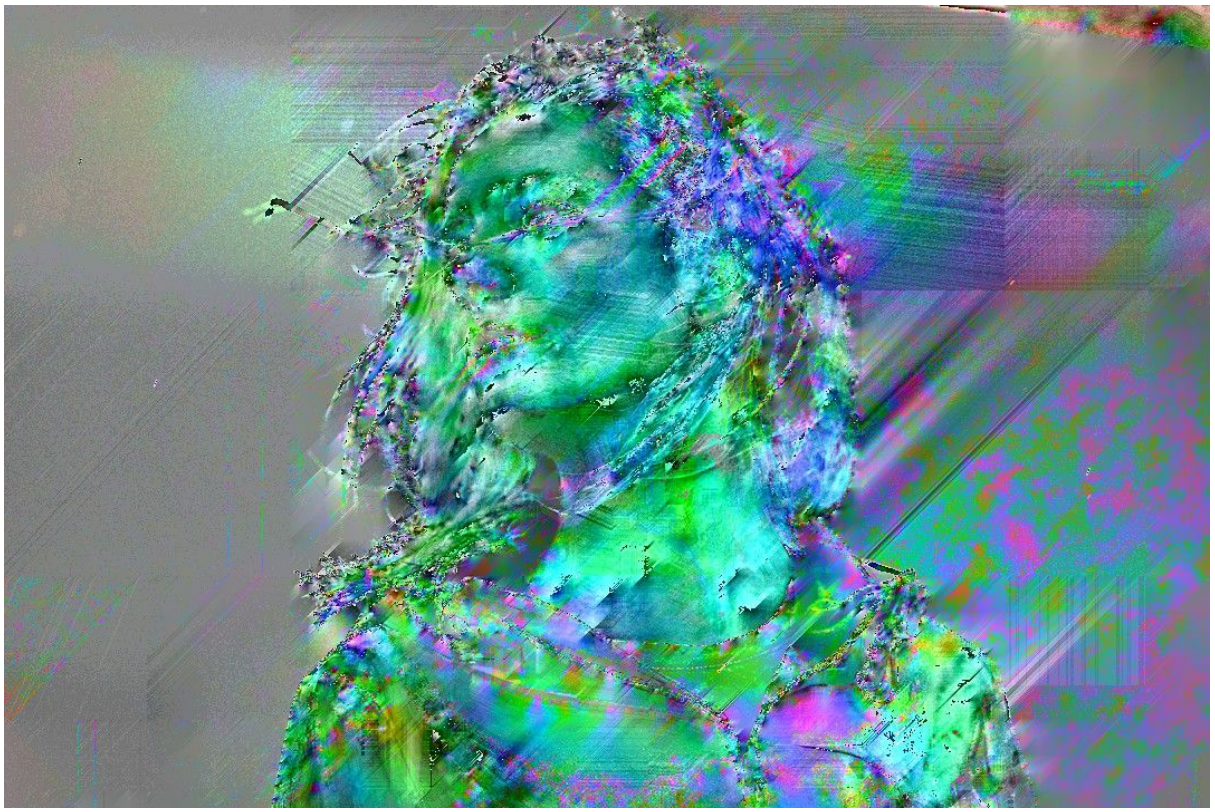
Force global prediction to PRED_ANGLE



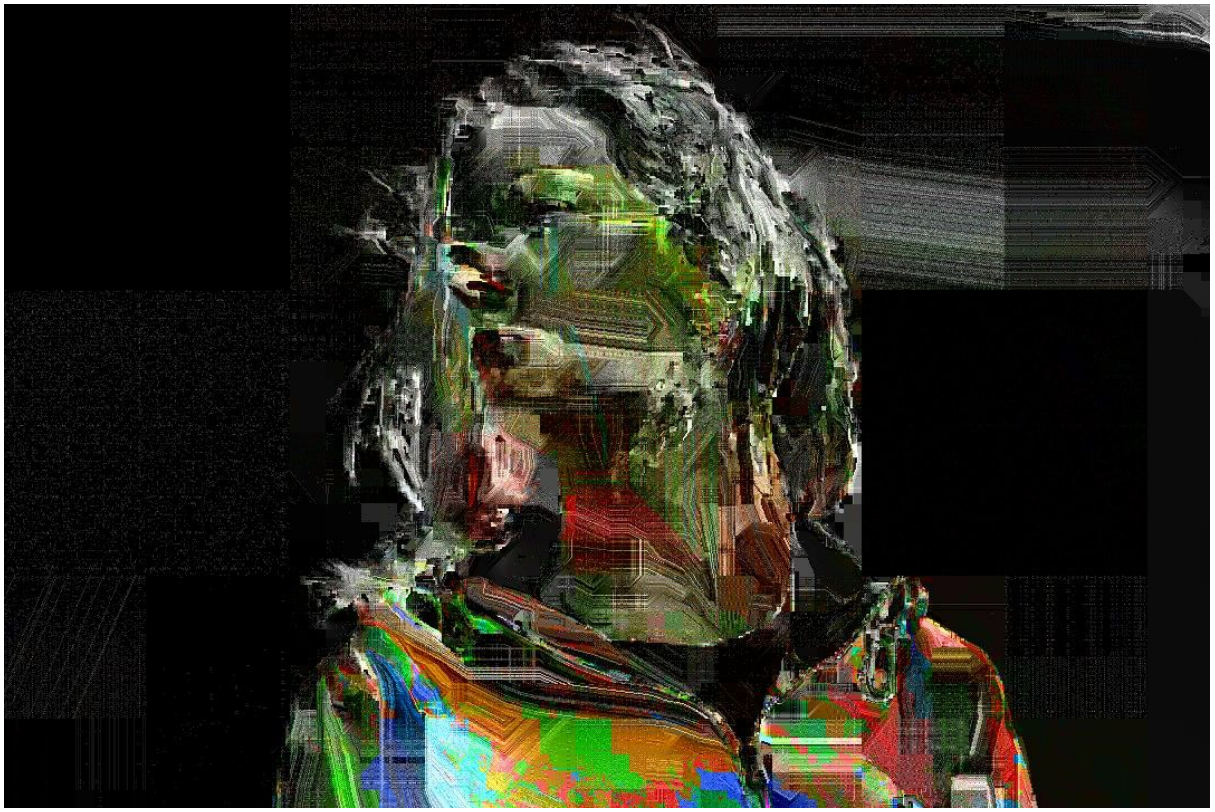
or PRED_MEDIAN (ch01), PRED_AVG (ch02), PRED_TRUEMOTION (ch03) and quant=0



or PRED_LDIAG all channels + quantization = 1A, and CLAMP_MOD256



quantization=4A, wavelet = COIFLET2 (0x12)

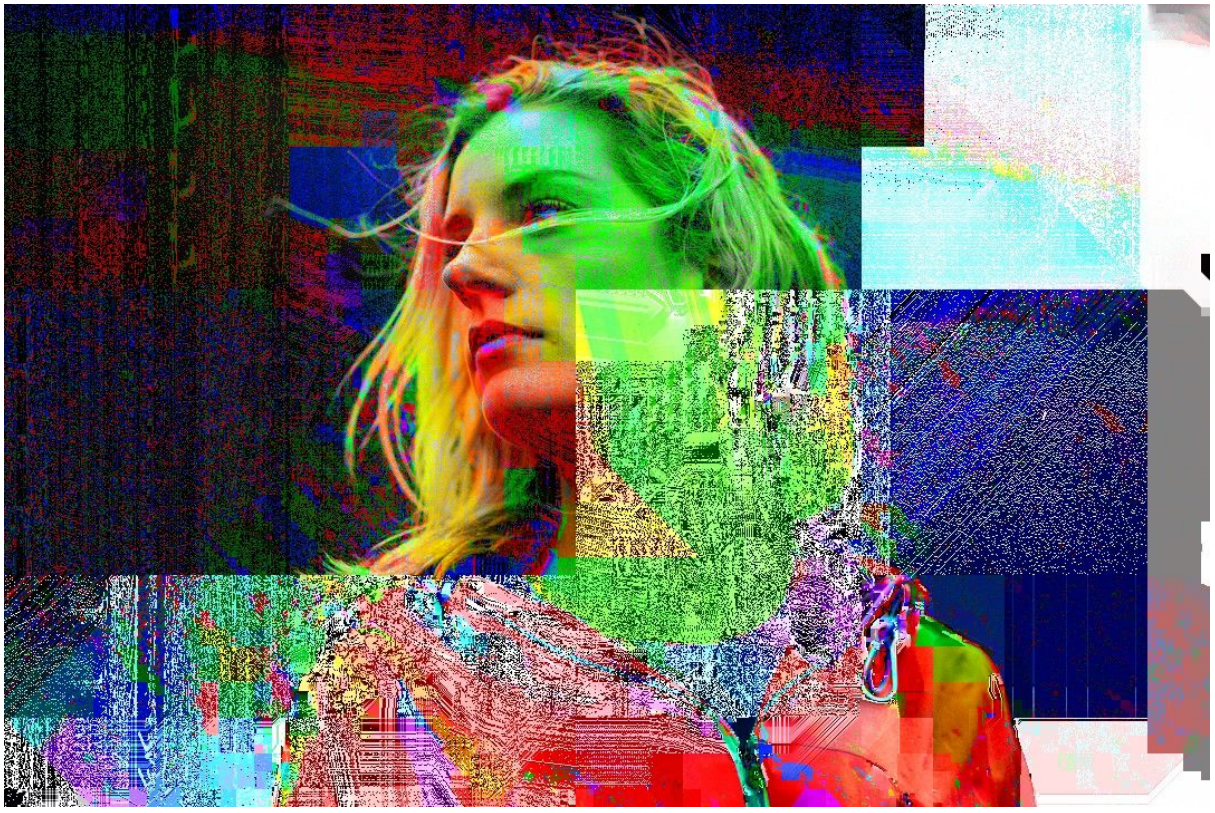


Segmentation data databend

Prediction data databend

Image data databend

Some huge copy&paste



Predictions and transformations set

YXY, PRED_DIAG, quantization=0, TRANS_FWT, COIFLET2/SYMLET4/DAUBECHIES6, compression=10

Headers

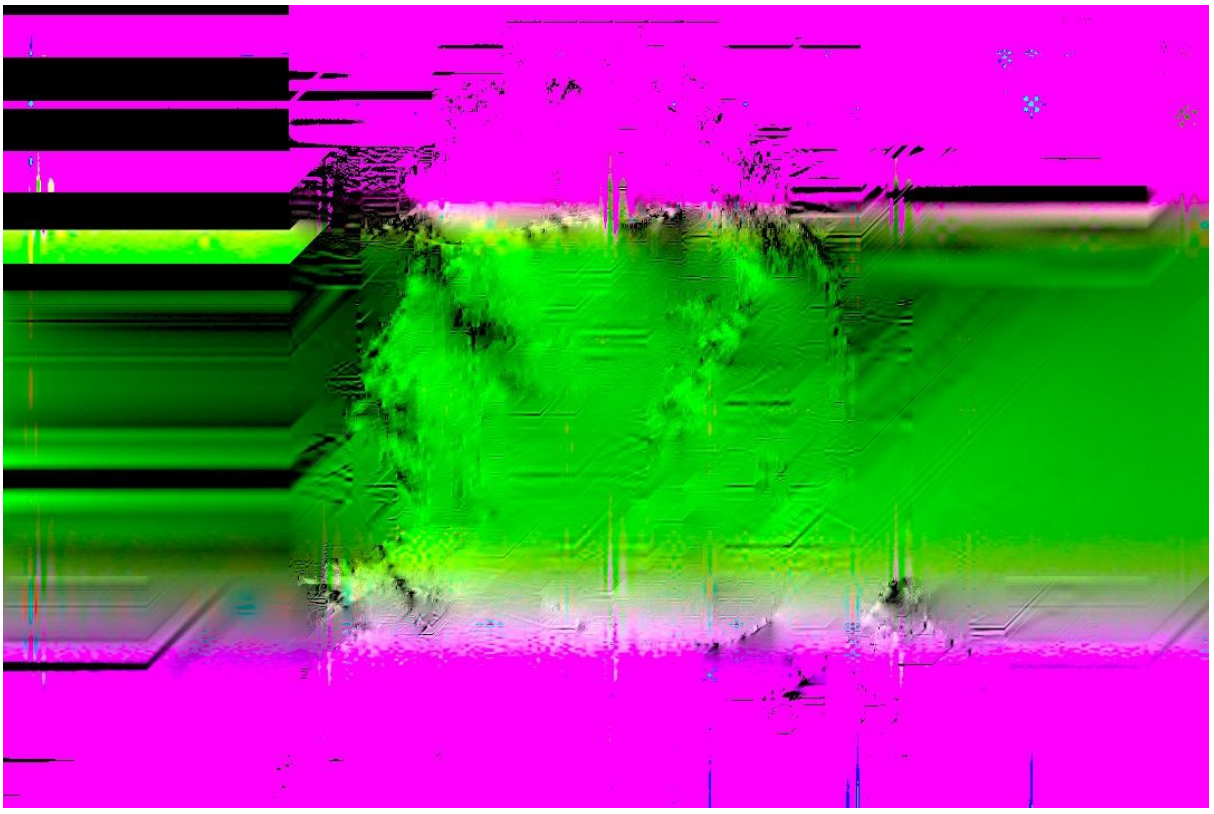
Image sizes

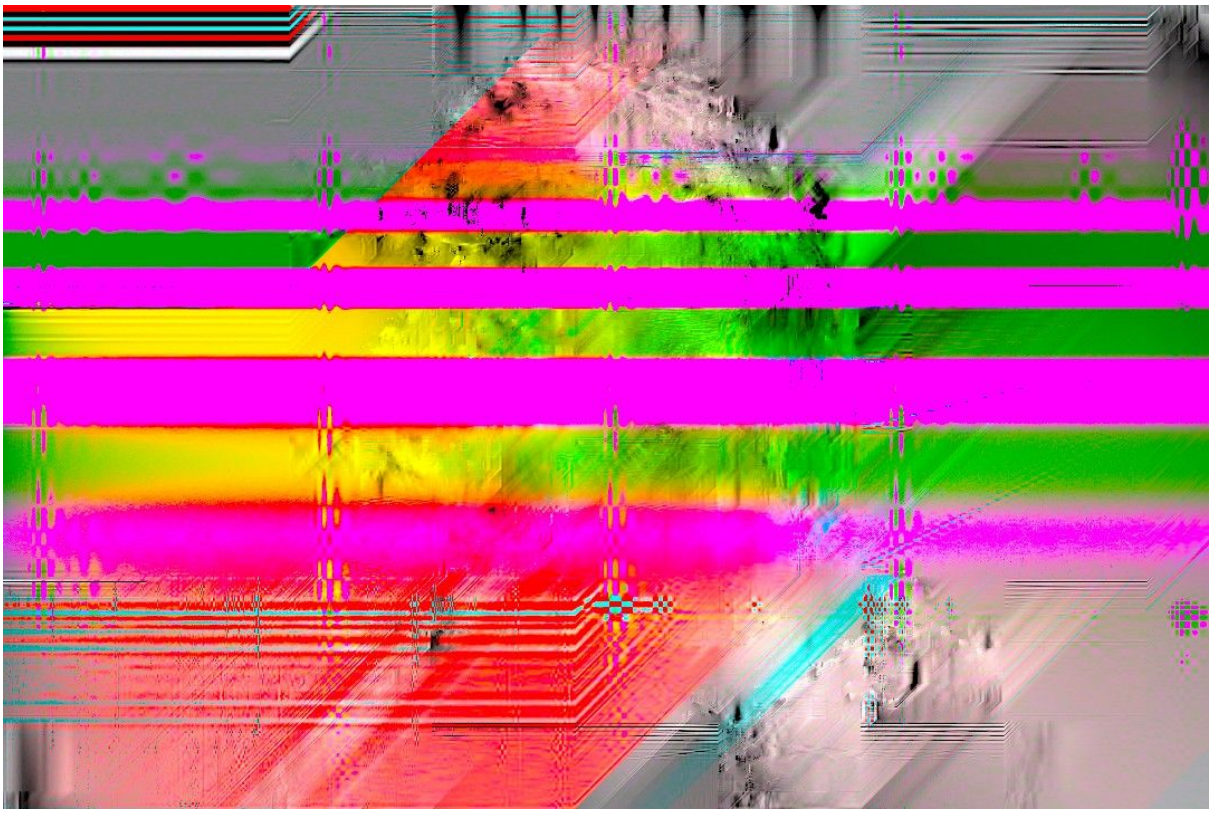
Colorspace

YXY->YDbDr

Color outside

Segmentation data sizes





Prediction data sizes

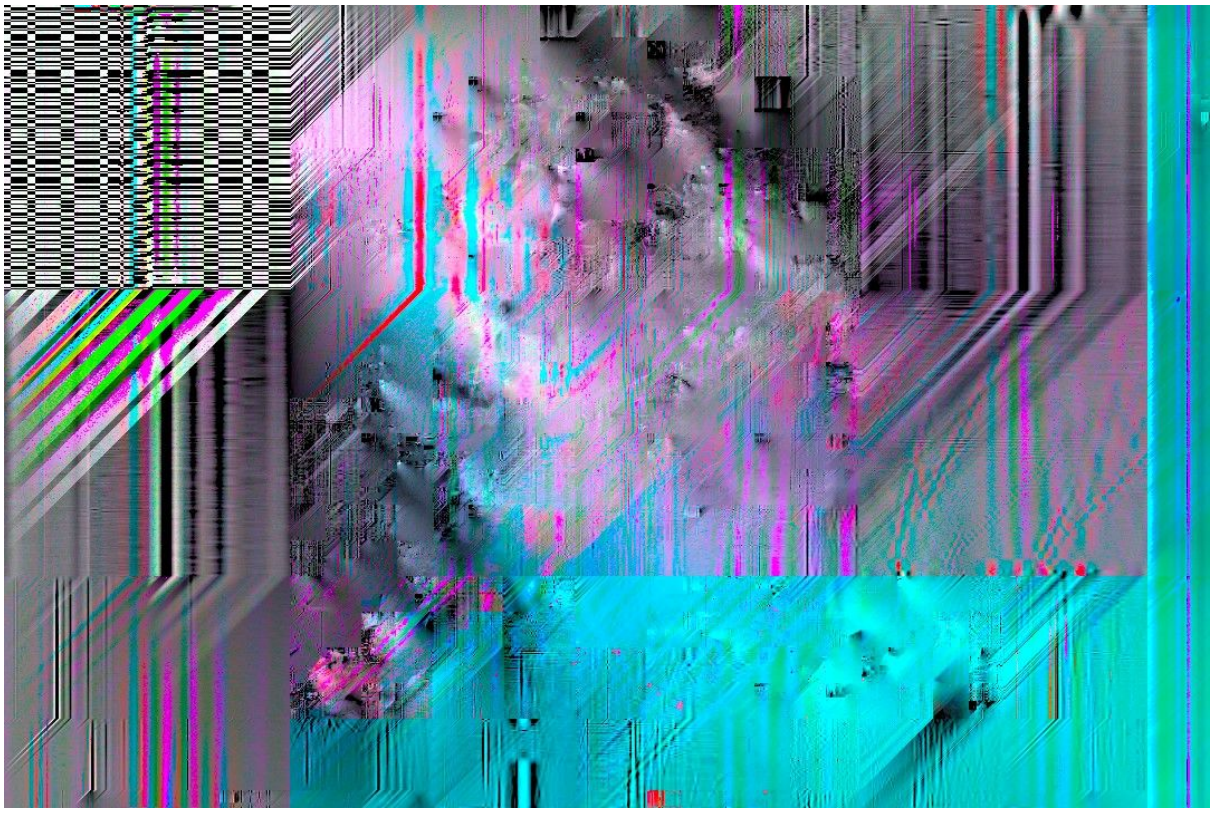
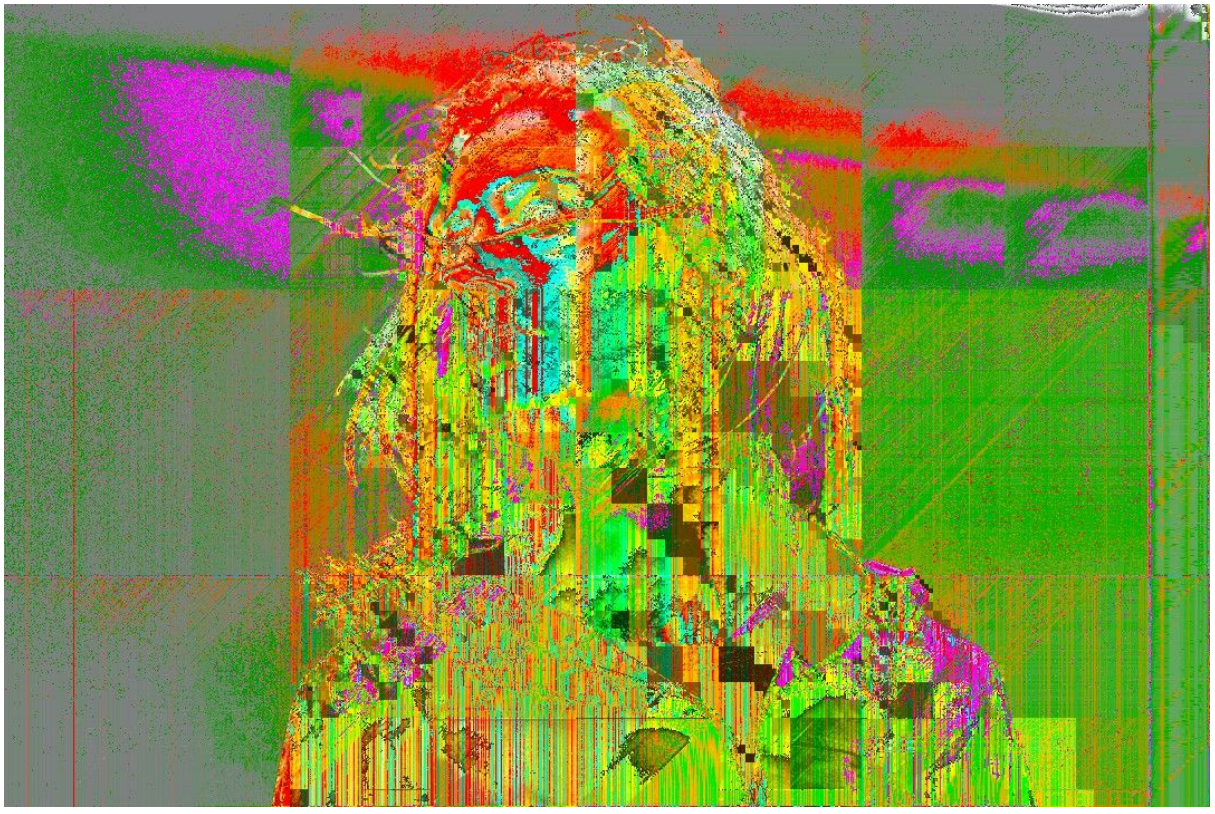


Image data sizes

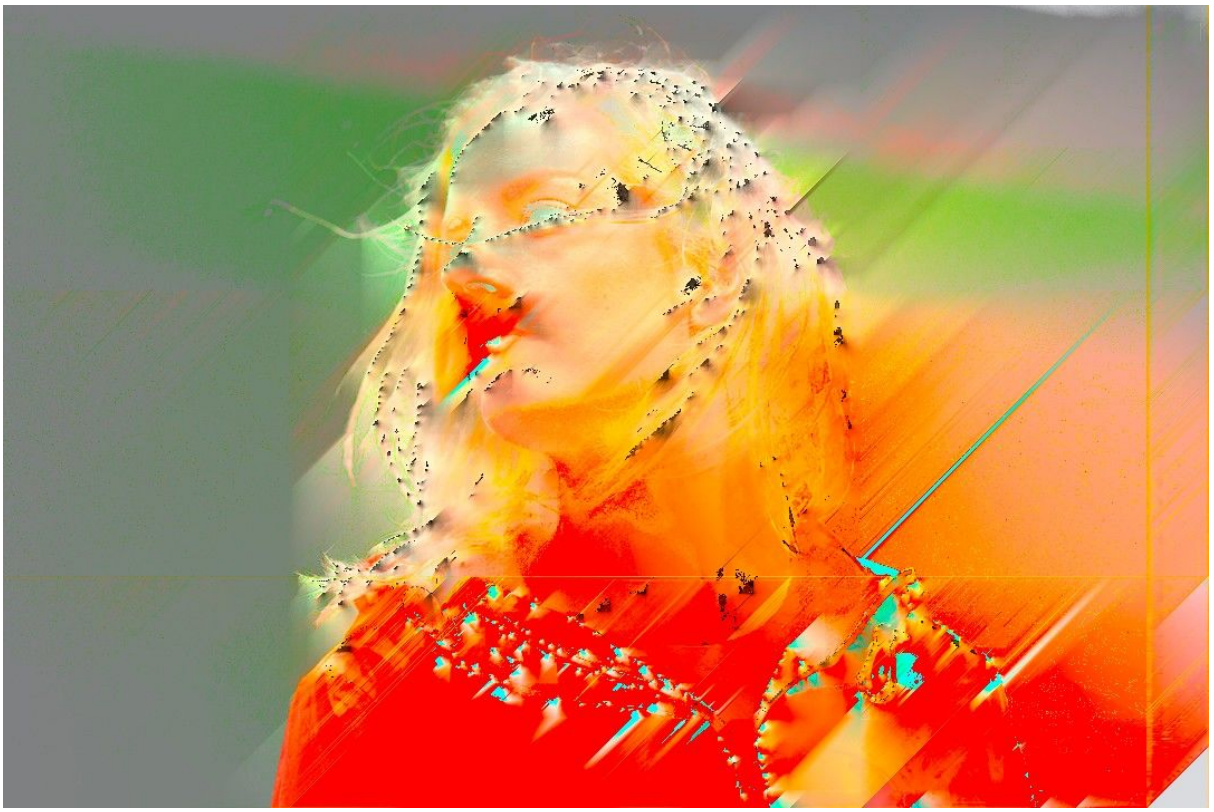
Second header

PRED_H/PRED_PAETH/PRED_ANGLE

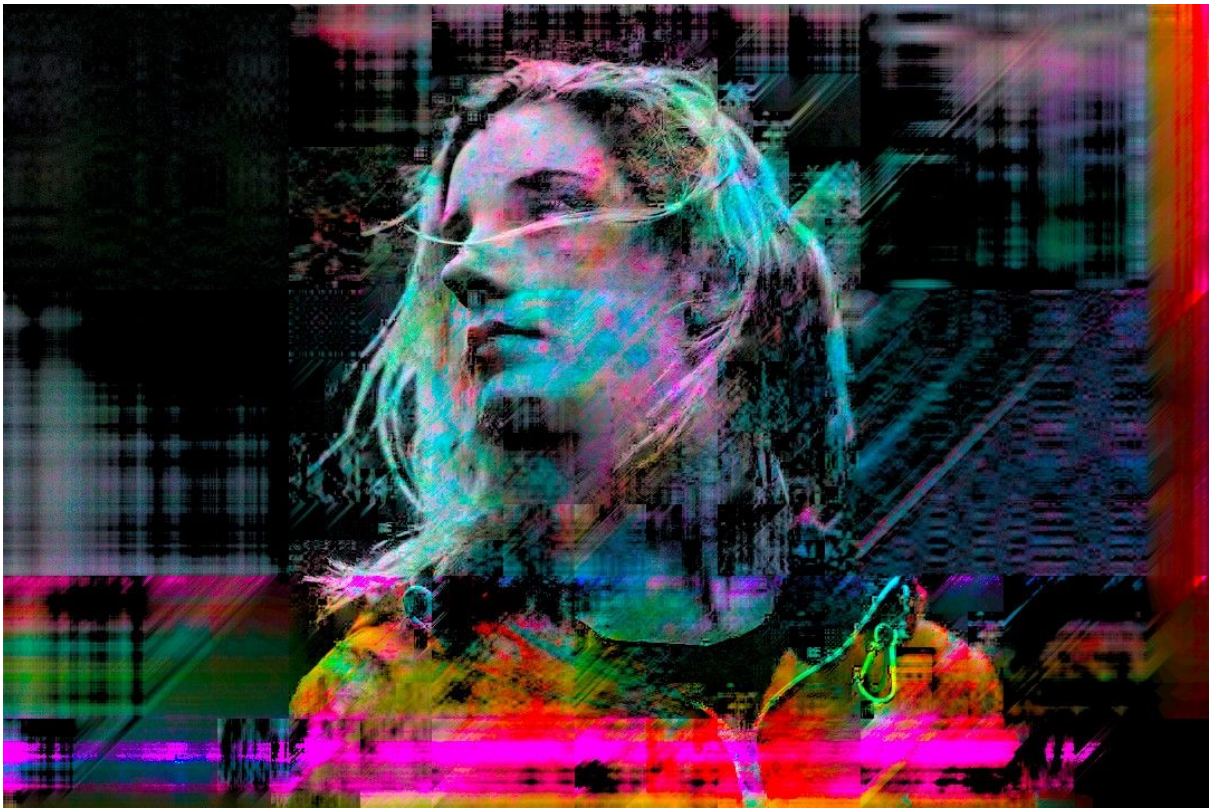
PRED_CORNER/PRED_V/PRED_DCMEDIAN, quantization=0x10, CLAMP_MOD256



only CLAMP_MOD256

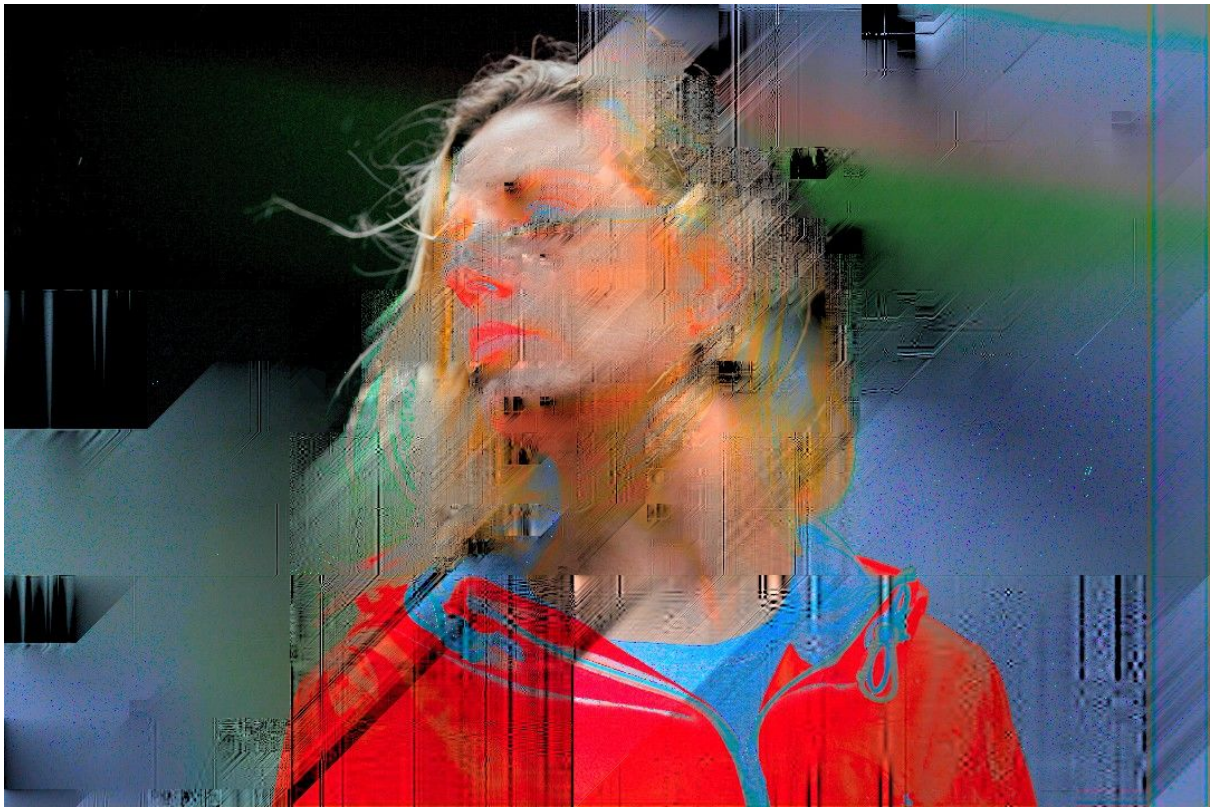


TRANSTYPE_WPT



TRANSTYPE_WPT, BIORTHOGONAL11/DAUBECHIES9/LEGENDRE3

Segmentation data databend



Prediction data databend

copy/paste

Image data databend