Steganography & Steganalysis

INSTRUCTOR: JOHN ORTIZ
SENIOR COMPUTER ENGINEER
UTSA
STEGO@SATX.RR.COM

BIT-PLANE COMPLEXITY SEGMENTATION

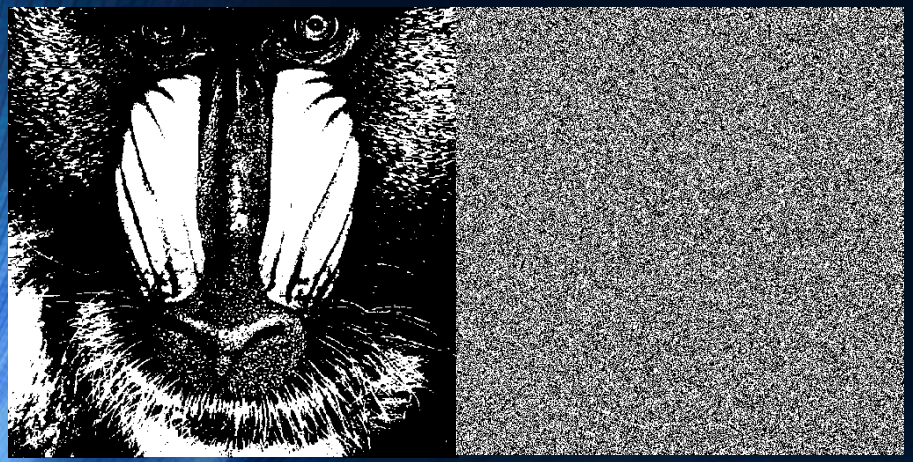
BPCS

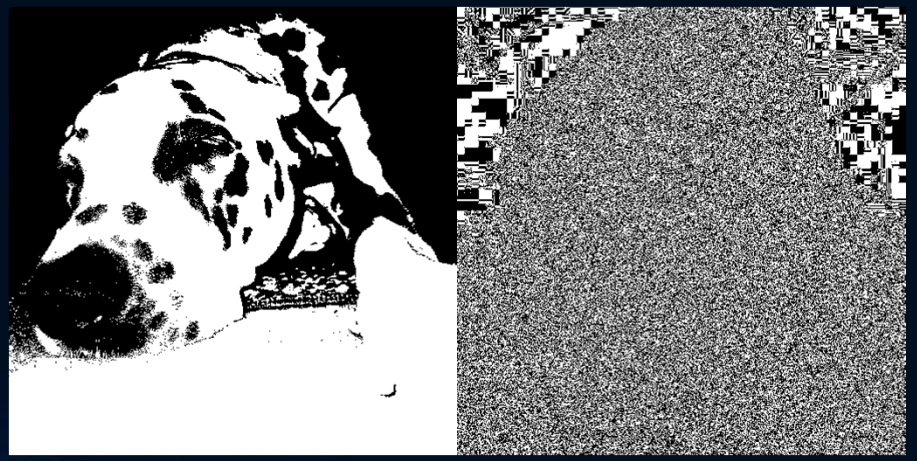
- BPCS embeds data into bit mapped images by slicing an image into individual bit planes
- It subdivides the bit-plane into an 8x8 block
- The complexity of the block is calculated
- If the complexity is greater than a threshold, data is embedded
- This technique takes advantage of the Human Visual System's inability to notice differences in complex images
  - Remember the mandrill's hairy face?

BPCS

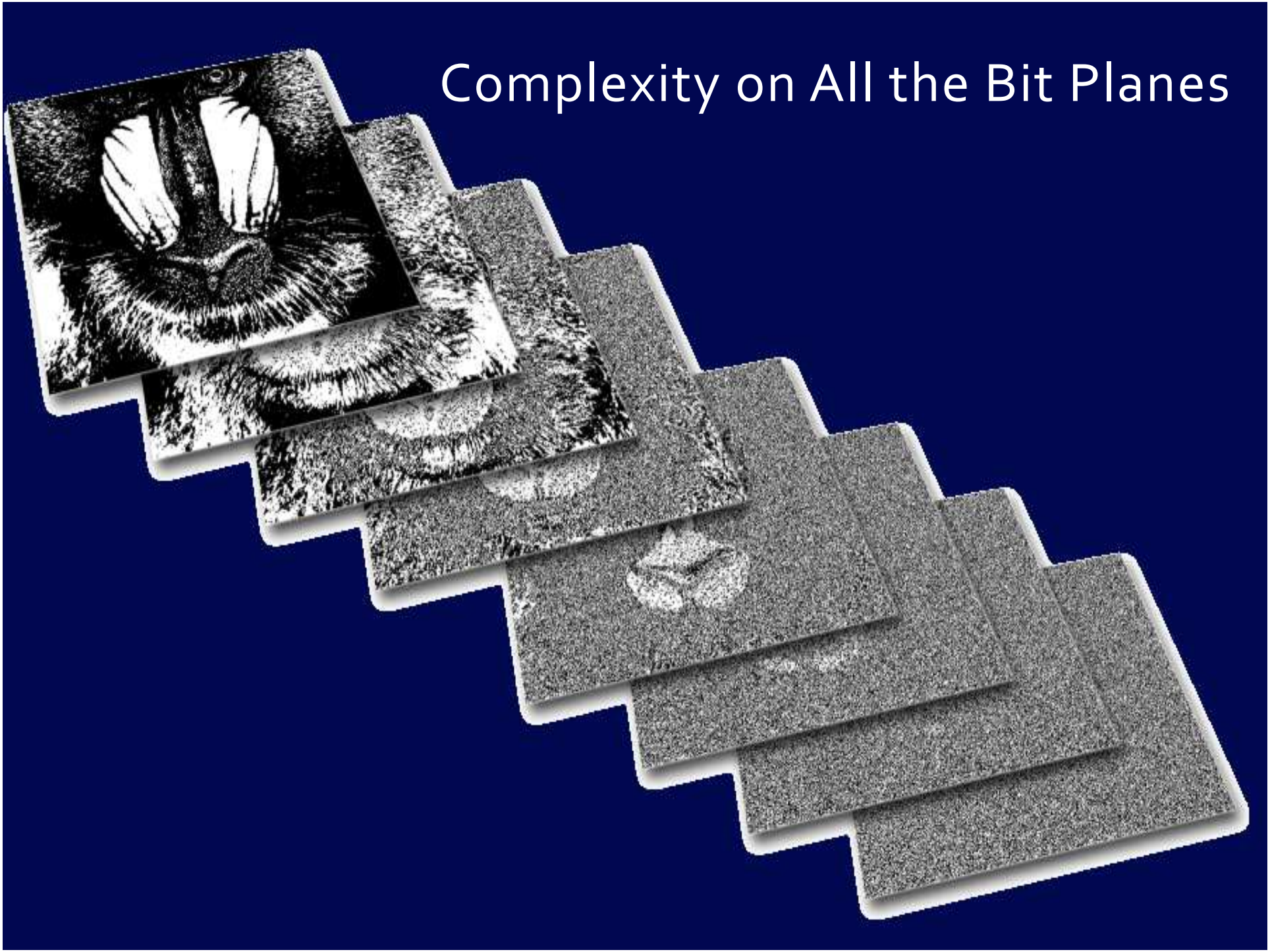
- BPCS is a more advanced substitution technique
  - Has a little less capacity than LSB
  - Harder to detect & extract
- The message is spread across several bit planes
  - Possibly even the MSB
- The concept is to hide in areas that are complex

MSB LSB MSB LSB





Complexity on All the Bit Planes



BPCS

- Pixel data is typically represented using "Pure Binary Coding" (PBC)
  - 0 = black, 255 = white, increase in value => increase in brightness
- Canonical Gray Coding (CGC) is used for BPCS because it provides higher capacity with less visual distortion

BPCS

- Canonical Gray Coding (CGC) means to use a gray code to represent the pixel values
- Recall that a gray code is one in which there is only a 1-bit difference between successive values
- 3-bit Gray Code:
  - 000, 001, 011, 010, 110, 111, 101, 100
- Use an 8-bit gray code, similarly constructed, to represent pixel values
- Easy to construct in software – mirror list in middle, add MSB to 2nd half

BPCS

- How does CGC help?
- The numbers 127 and 128 are just 1 value apart
- In binary they look like this:
  - 127: 0111 1111
  - 128: 1000 0000
- The left half is 127, the right half, 128

COMPLEX – Every bit changes

BPCS

- If a region were composed of these values in an alternating way, it would appear very flat, yet the complexity measure would be high
- With CGC, these two values only differ by one bit so both the mathematical difference and visual difference are small

BPCS Algorithm

- Convert pixel representation to CGC
- Start with LSB plane
  - Why do you think this would be a good implementation idea?

BPCS Algorithm

- Convert pixel representation to CGC
- Start with LSB plane
  - Why do you think this would be a good implementation idea?
  - If capacity is not maxed, start by modifying the least perceptible information
- Select an 8x8 block
  - This can be sequential or use some other selection technique such as a PRNG
- Calculate the complexity of the block

BPCS Complexity

- No standard complexity measure
- Several publications have proposed differing complexity measures
- The demo program uses a simple border length measurement
- The total length of the border is the sum of the number of one/zero changes along the rows and columns
  - Since we are using the measure on bit planes, every pixel is either a one or zero
  - Ex. A zero pixel, surrounded by all ones, has a border length of 4

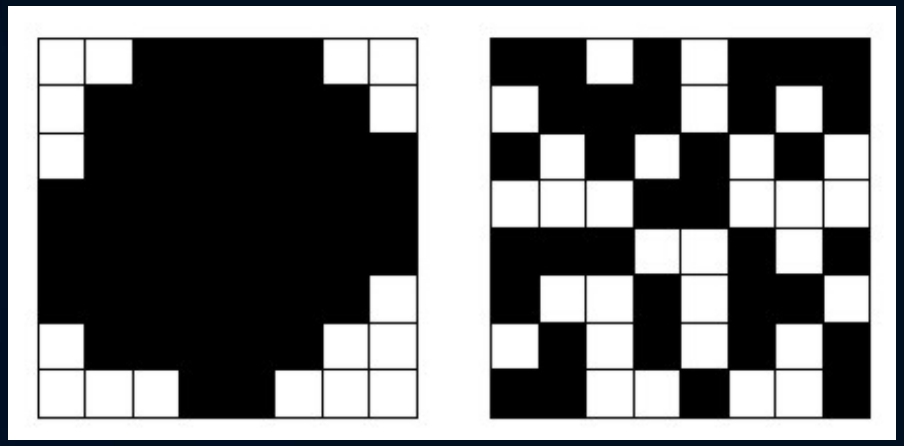
BPCS Complexity

- Complexity Formula:
- α = the complexity measure
- k = border length
- M = maximum border length
  - A checkerboard pattern is max
    - Irony: it should NOT be consider complex!
  - For a square region: M = 2 * (n^2 – n), n = matrix size
  - For an 8 x 8 square region, M = 112
- NOTE: 0 <= α <= 1

$$\alpha = \frac{k}{M}$$

BPCS Complexity

- Low complexity / High complexity



- α_{th} is the threshold
  - Experimentally determined to be good at 0.3
  - Must be less than 0.5 – Why? (hang on)

BPCS

- Once it is determined that a region is complex enough, the image data is directly replaced by the message data (substituted)
- An 8x8 matrix has 64 bits in each bit-plane
- The message data is transformed into an 8x8 bit array, then the original data is replaced by message data
- Does anyone see a problem during extraction?

BPCS

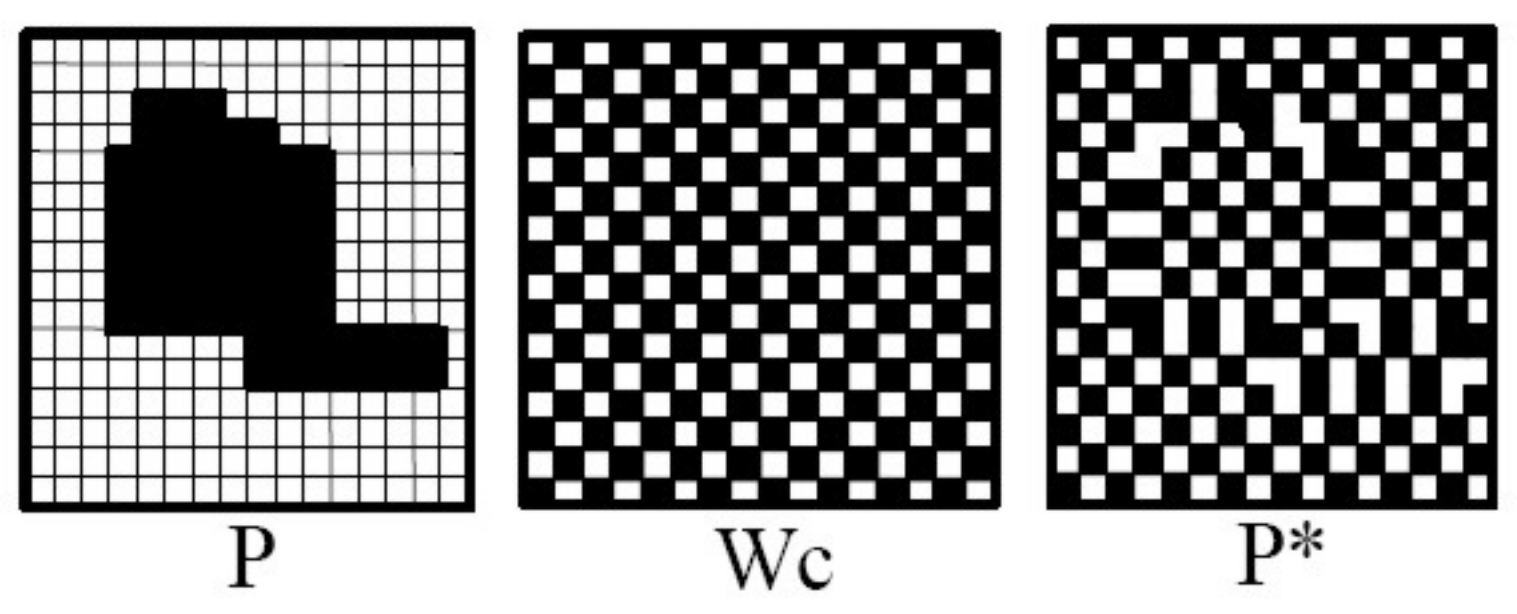
- Once it is determined that a region is complex enough, the image data is directly replaced by the message data (substituted)
- An 8x8 matrix has 64 bits in each bit-plane
- The message data is transformed into an 8x8 bit array, then the original data is replaced by message data
- Does anyone see a problem during extraction?
  - What if the message data is NOT complex?
  - Then, during extraction, it will not exceed the threshold and will be skipped!

BPCS

- The solution is to CONJUGATE the message data when it is not complex
  - This operation is NOT complex … it is simple
  - Exclusive-or the message data with a checkerboard pattern
  - Guaranteed complexity!
  - $\alpha_{next} = 1 - \alpha_{prev}$
  - If α_{prev} was less than 0.3, α_{next} is greater
  - This is why the threshold MUST be less than 0.5
  - Say $\alpha_{th} = 0.7$.
  - $\alpha_{prev} = 0.6$, after conjugation $\alpha_{next} = 0.4$

BPCS

- The Conjugate Operation Shown Graphically
  - P is the data
  - Wc is the white checkerboard
  - P* is the conjugate of P



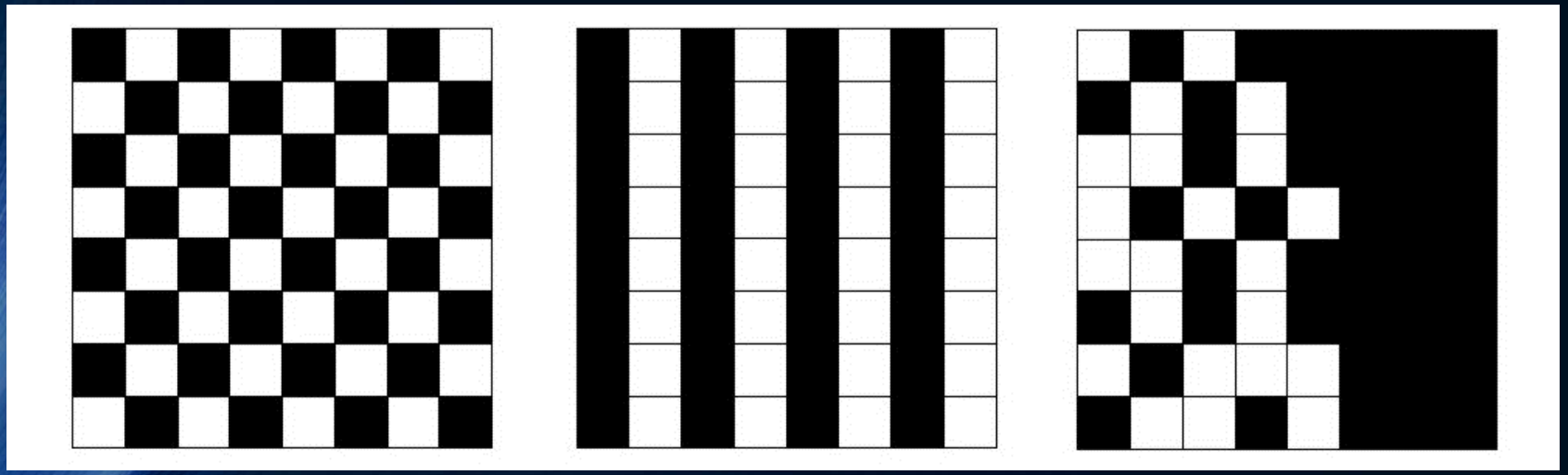
BPCS

- New problem: Which 8x8 blocks were conjugated?
- Reserve one bit of each region to indicate conjugation
  - Make the lower left bit of the 8x8 matrix a zero
  - If conjugation occurs, it will become a one
- This takes up 1/64 of the embedding capacity
- It complicates the implementation too
  - After getting 8 bytes of message, we have one extra bit

- MY Solution:
  - First byte indicates conjugation for this and the next 7 blocks

BPCS

- The original papers were published in 1998
- Later, Hioki Hirohisa determined that their complexity measure had a weakness
- The regions below would have high complexity, yet make a poor choice for data embedding
  - In the 3rd case, the noisy region would grow

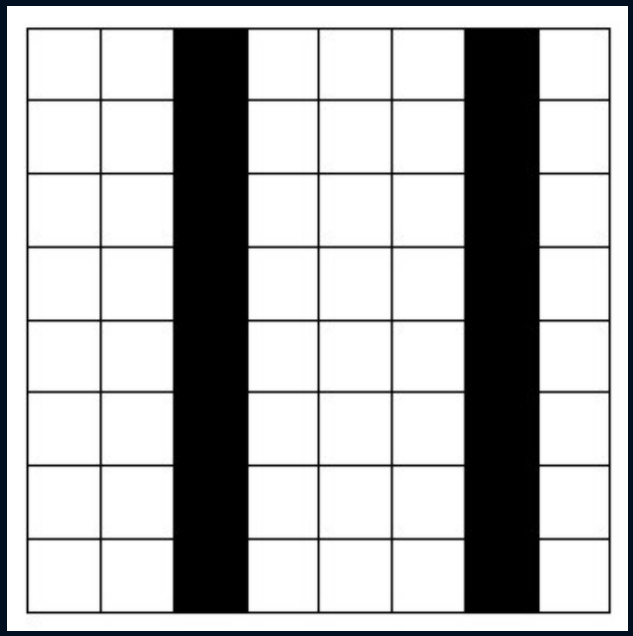


BPCS

- Hirohisa developed two new techniques for complexity measurement:
  - Run-Length Irregularity
  - Border noisiness
- Measures the non-uniformity of the region
- Uses an entropy-like calculation
- Based on the histogram of runs of zeros and ones in both rows and columns

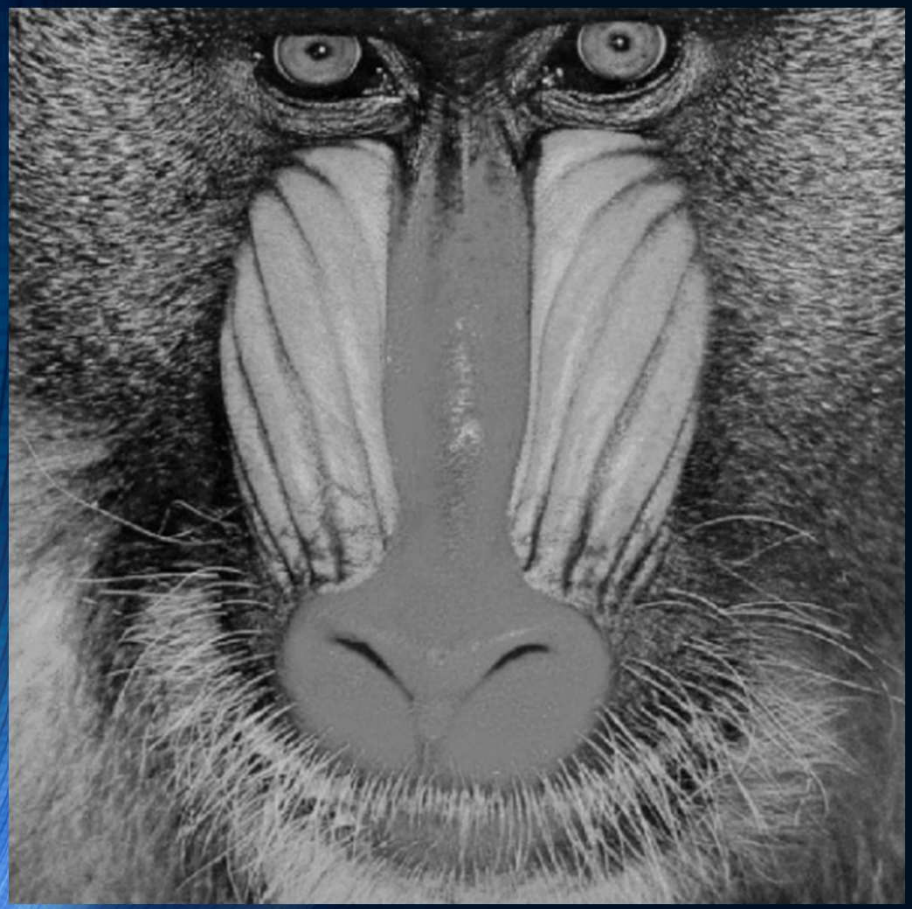
BPCS – Threshold = 0.3

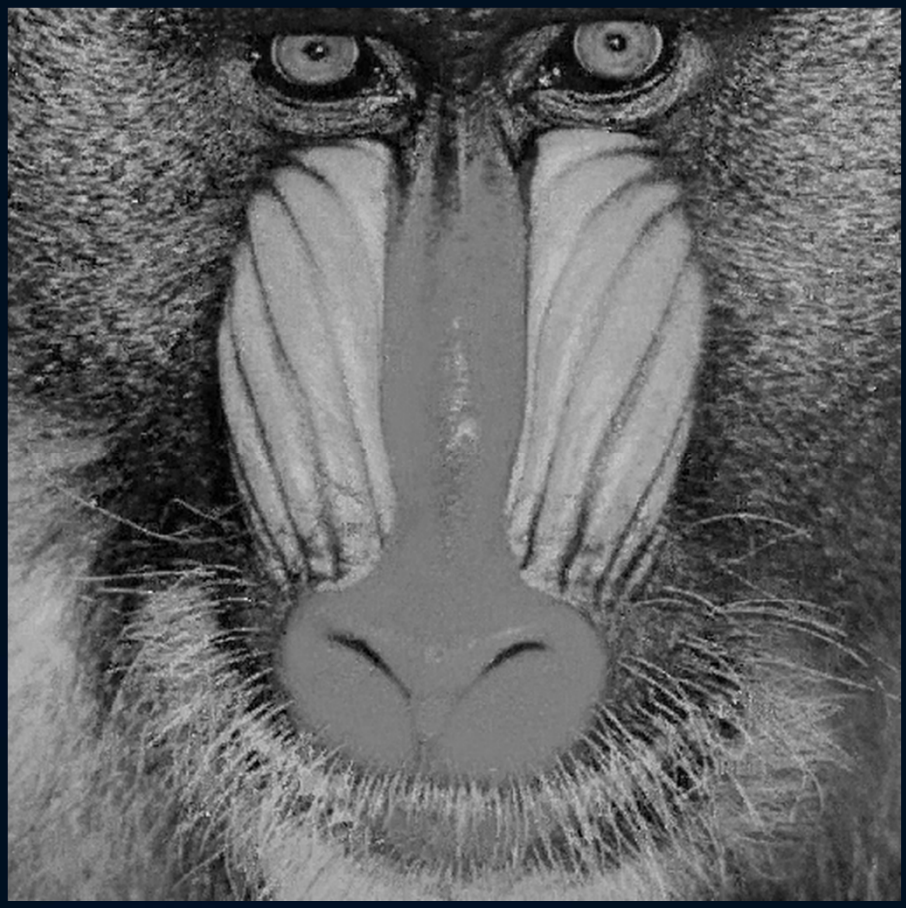
- As an example:



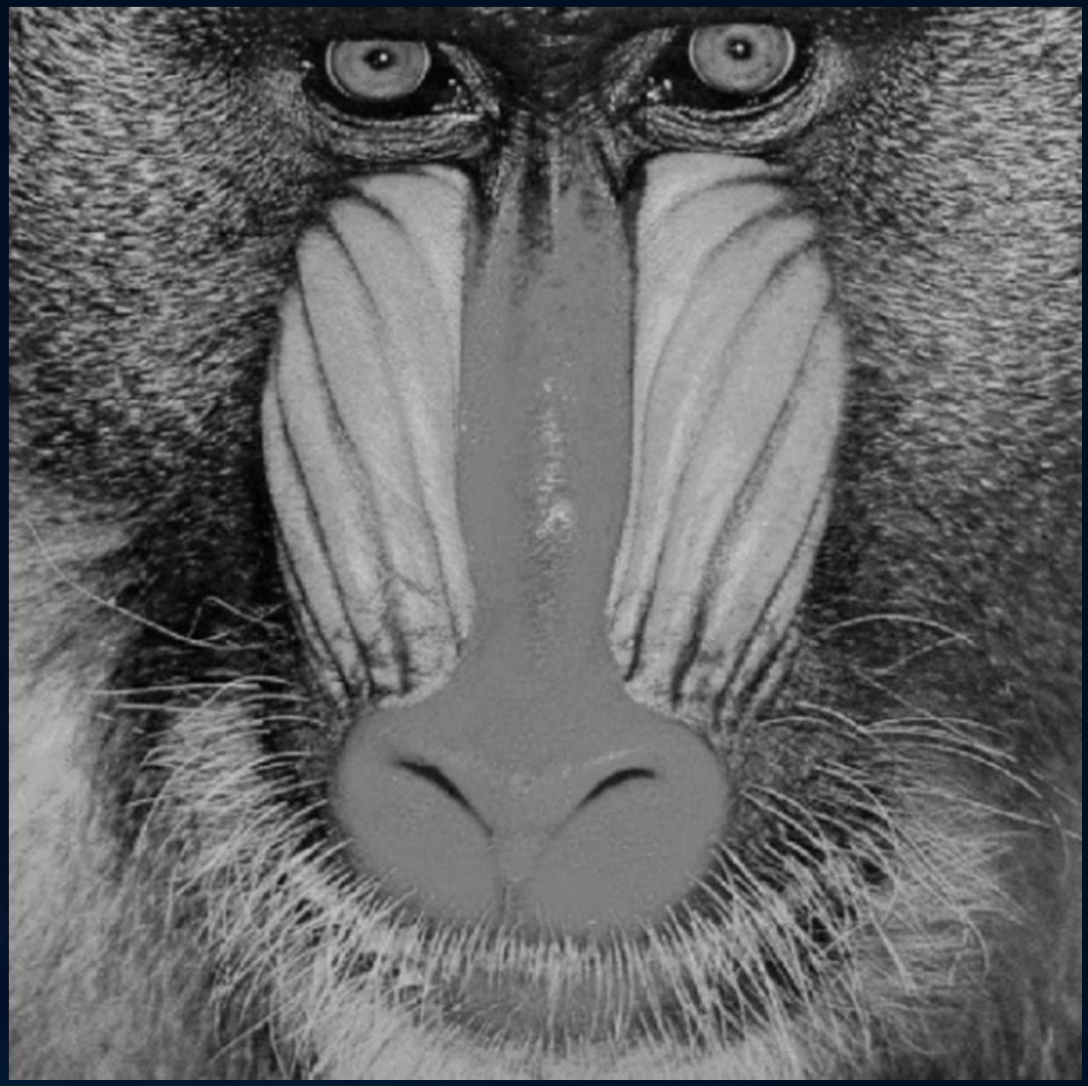
- Run-Length irregularity is directional →
- This region is complex in one direction, but not in the other
- This block is NOT suitable for embedding

BPCS

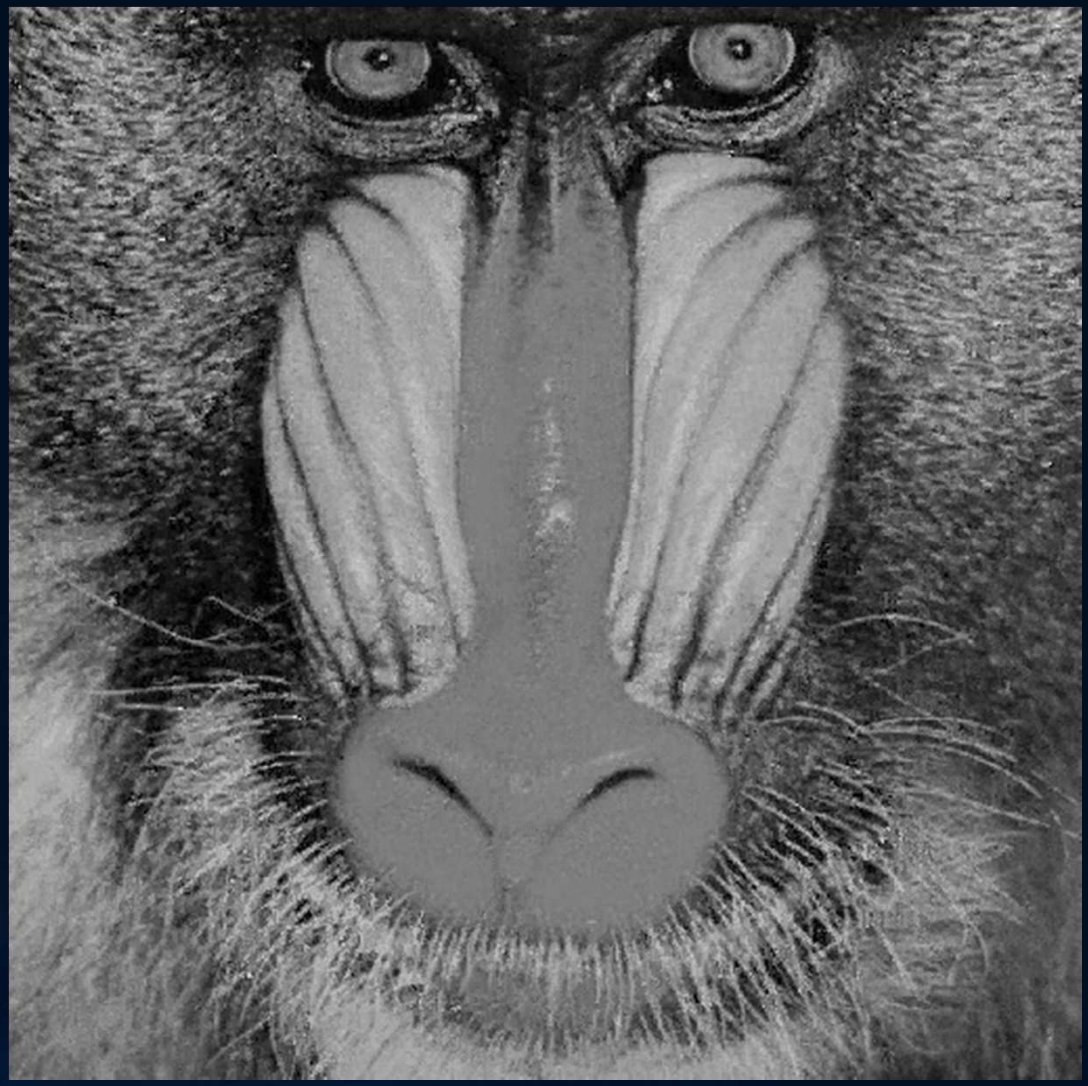




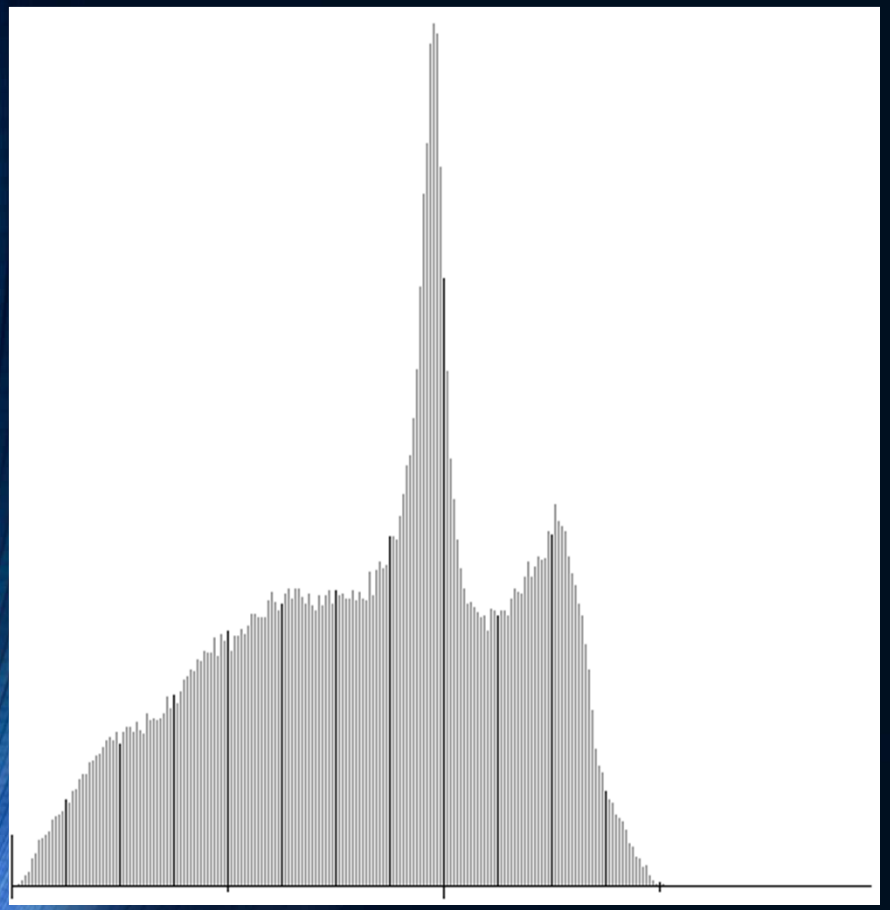
BPCS - Original



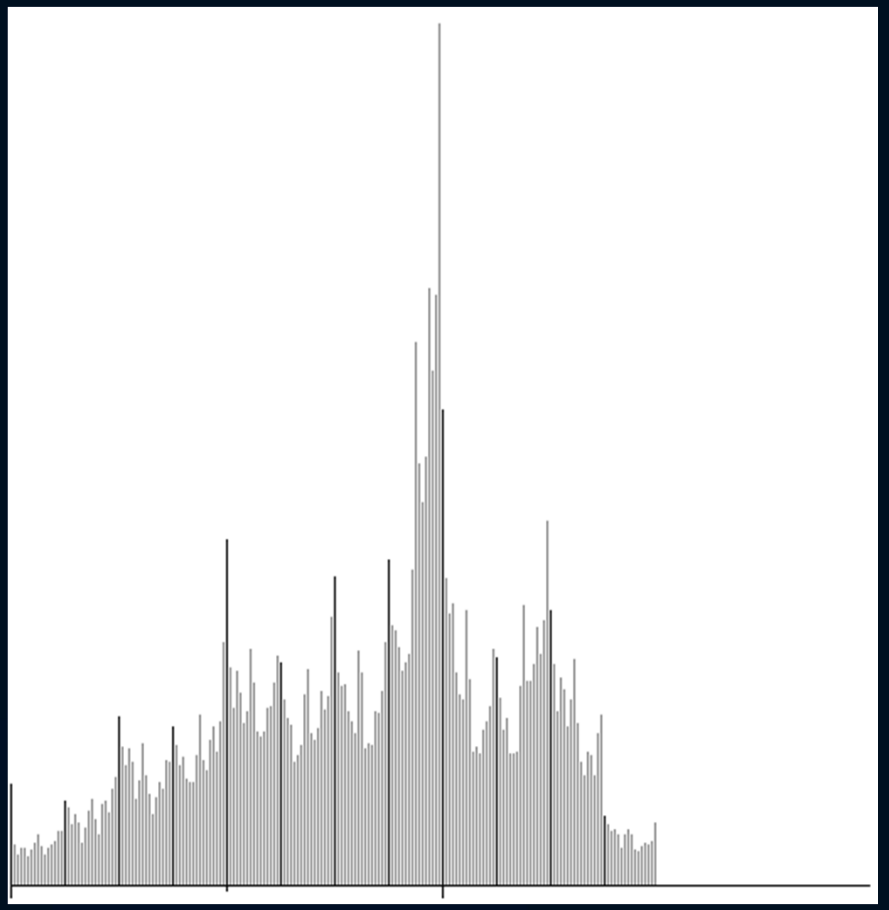
BPCS – Threshold = 0.3



BPCS – Histograms Still Effective

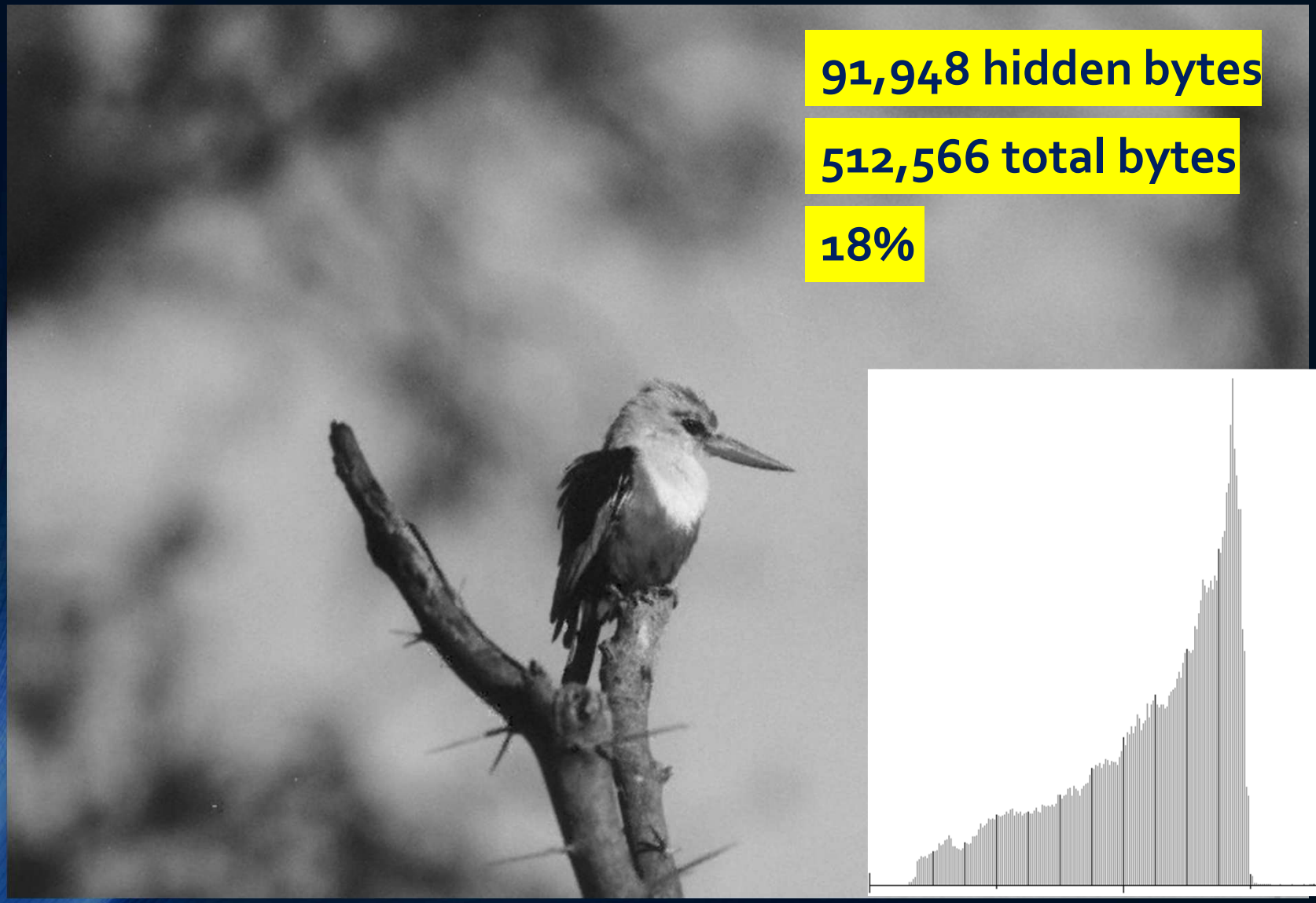


Original



Th=0.3, cap = 134KB/258KB

BPCS – Kingfisher: th= 0.4

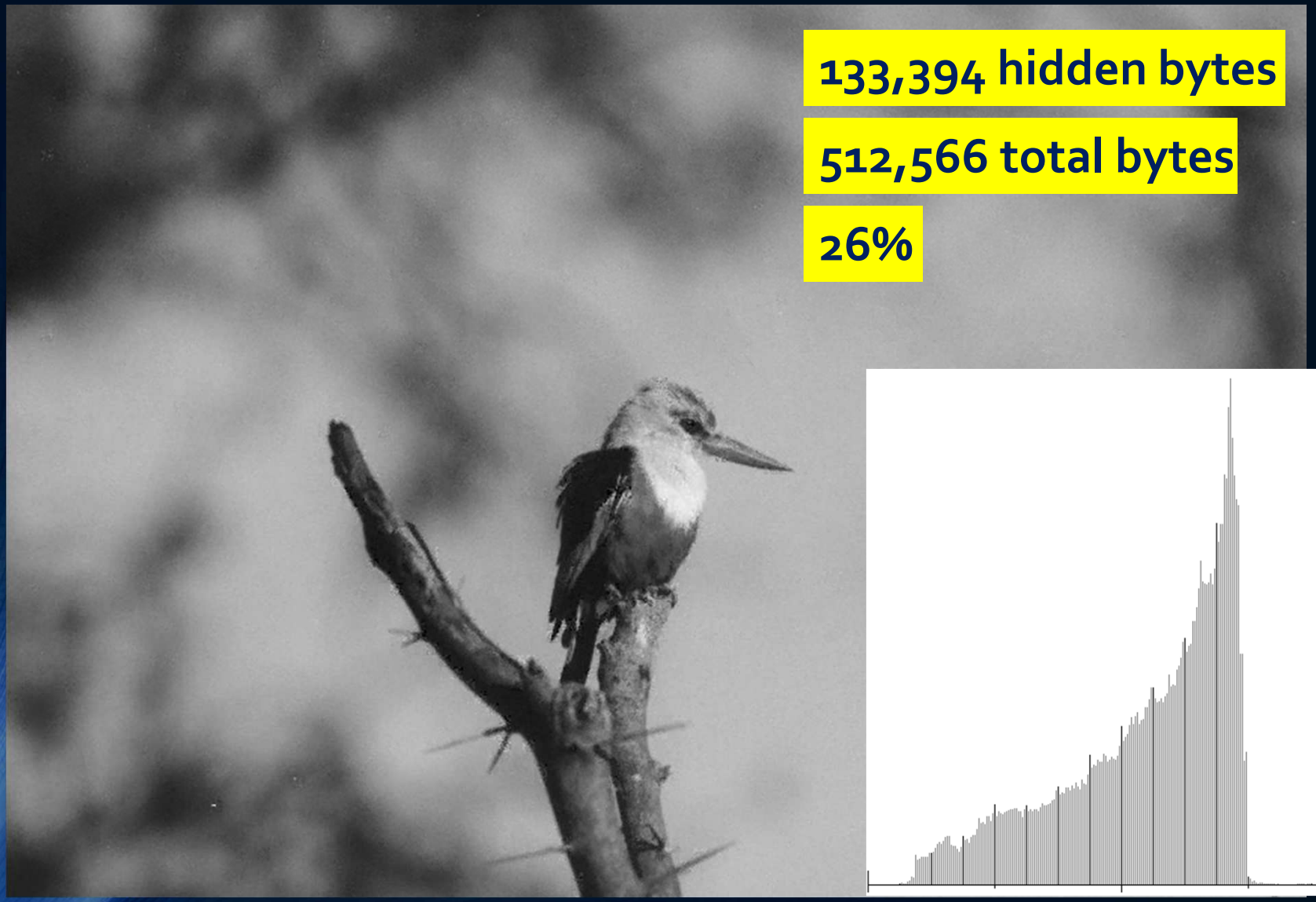


91,948 hidden bytes

512,566 total bytes

18%

BPCS – Kingfisher: th= 0.3

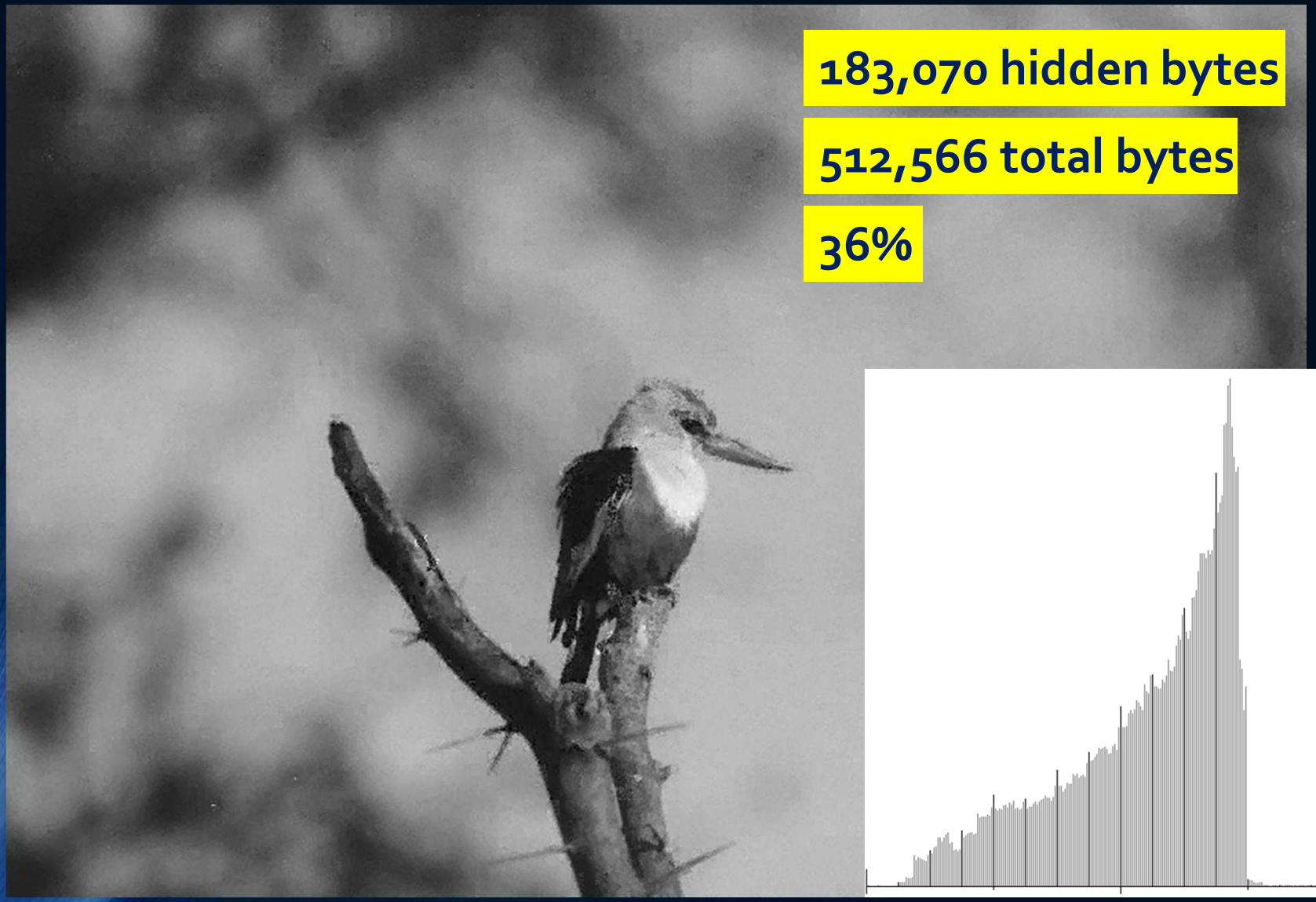


133,394 hidden bytes

512,566 total bytes

26%

BPCS – Kingfisher: th= 0.2

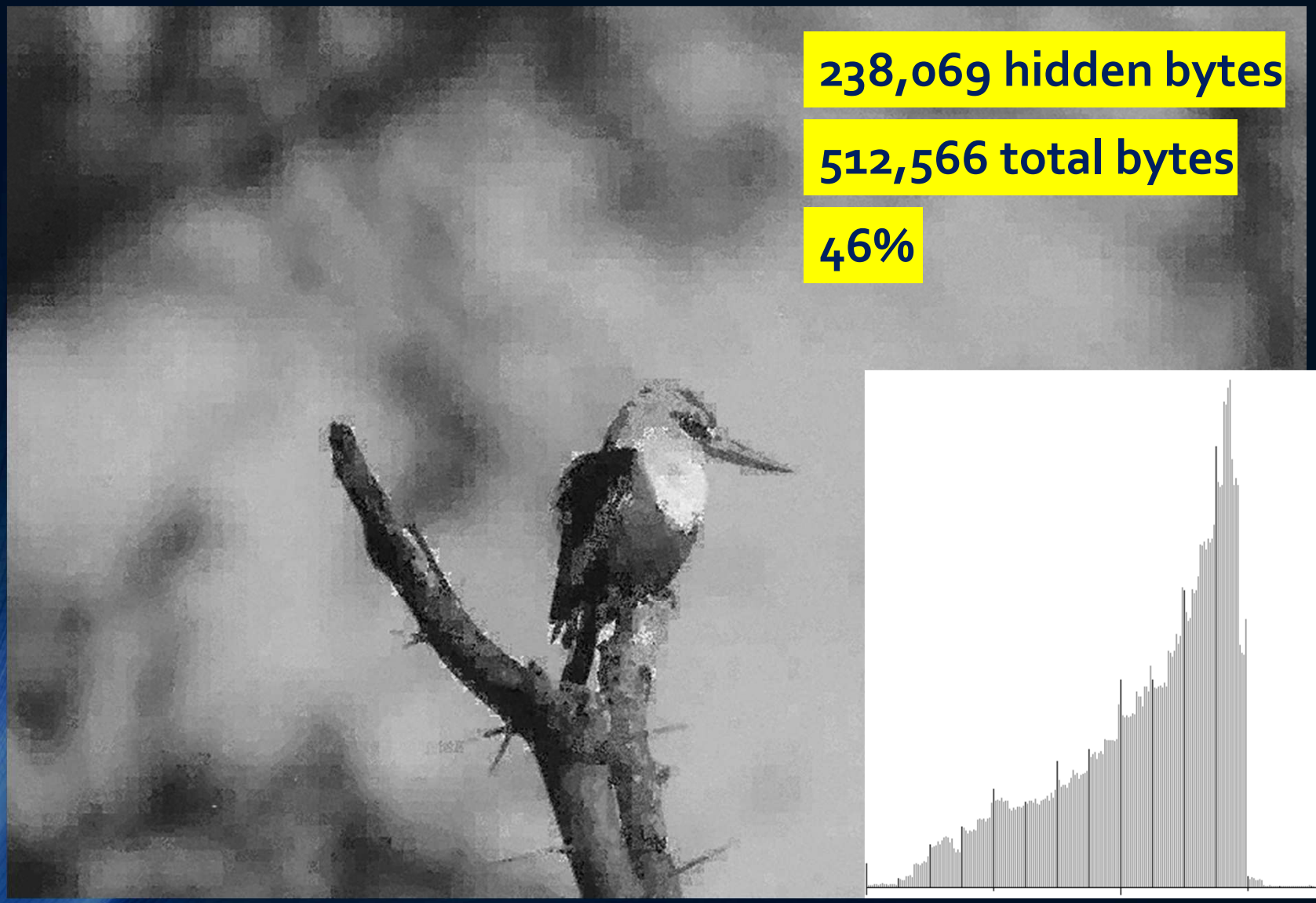


183,070 hidden bytes

512,566 total bytes

36%

BPCS – Kingfisher: th= 0.1



238,069 hidden bytes

512,566 total bytes

46%

BPCS – Lion in the Grass: th=0.4

181,440 hidden bytes

512,566 total bytes

35%

BPCS – Lion in the Grass: th=0.3

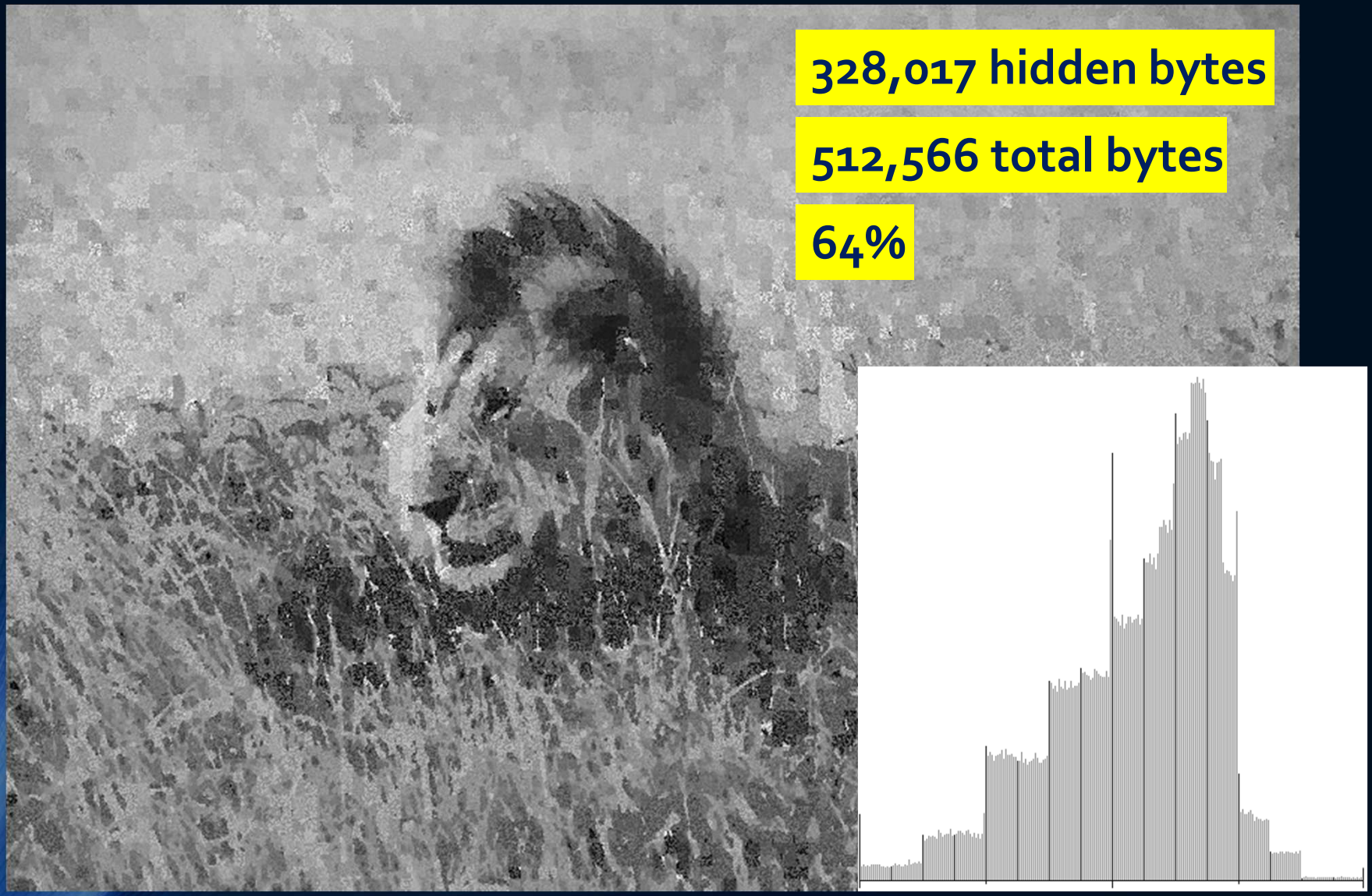
222,689 hidden bytes

512,566 total bytes

43%

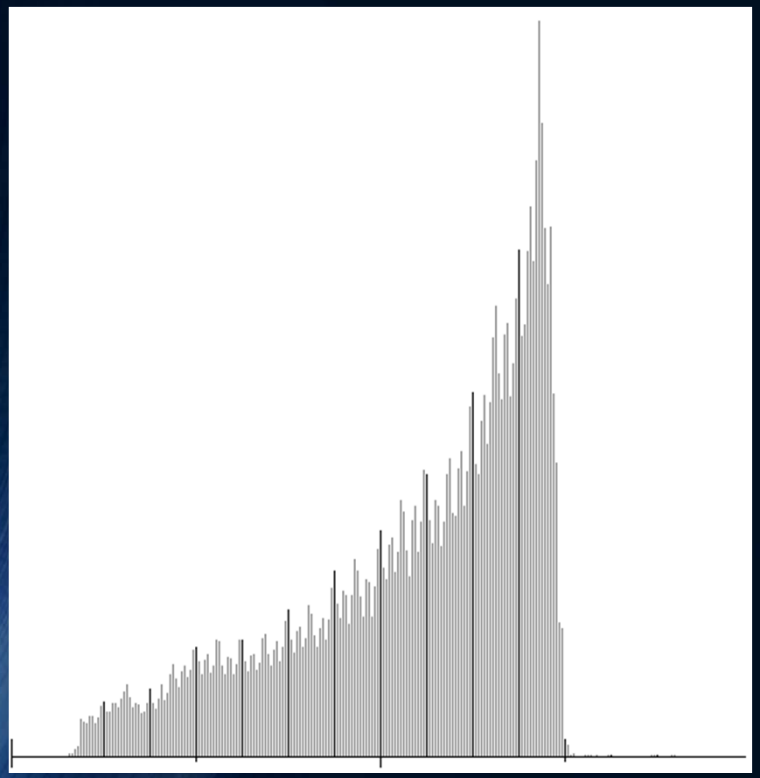
BPCS – Lion in the Grass: th=0.2

BPCS – Lion in the Grass: th=0.1

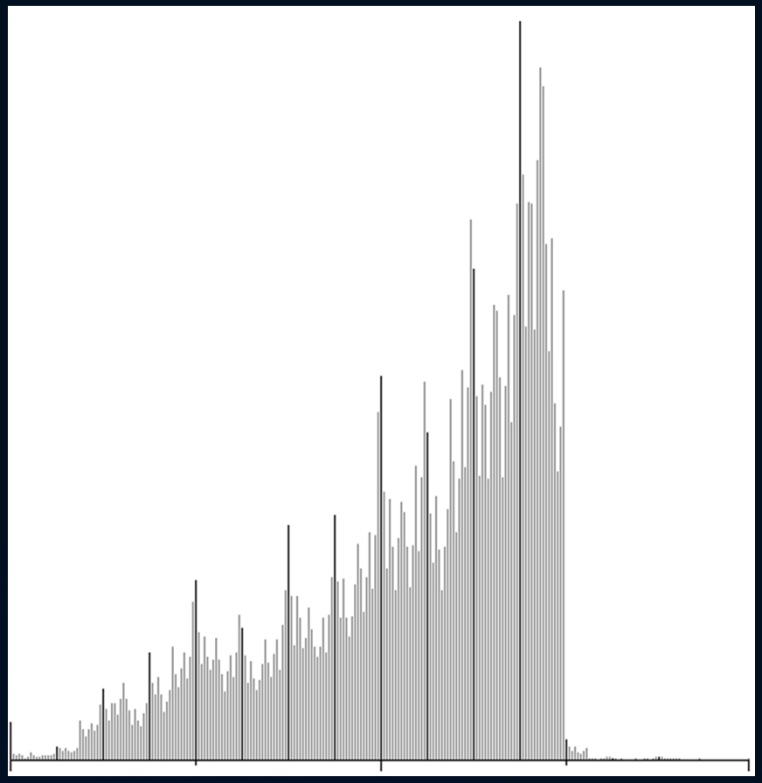


BPCS

- The prior examples had encrypted data hidden
- Check out the histograms when hiding text



Kingfisher: th=0.4



Kingfisher: th=0.1

Color BPCS

- BPCS on either paletted or 24-bit color images is not quite so successful

Lion: th=0.1

Lion: th=0.2

Color BPCS

- 8-bit Paletted

Lion: th=0.3

Lion: th=0.4

Color BPCS – 24-bit




Color BPCS – 8-bit

Color BPCS – 24-bit

References

- Based on
  - "A Research on Bit-Plane Complexity Segmentation Based Steganography" by Eiji Kawaguchi
  - "A Tutorial on BPCS Steganography and its Applications" by Richard Eason
- These two have collaborated on multiple other papers
- The Run-Length and Border Noisiness complexity measures came from
  - "A Data Embedding Method Using BPCS Principle with New Complexity Measures" by Hioki Hirohisa

Questions &Comments

- Complaints?