Noisy images and how to quiet them

Henry Dietz

Keeping Current, February 9, 2022

University of Kentucky
Electrical & Computer Engineering

University of Kentucky

Noisy images and how to quiet them

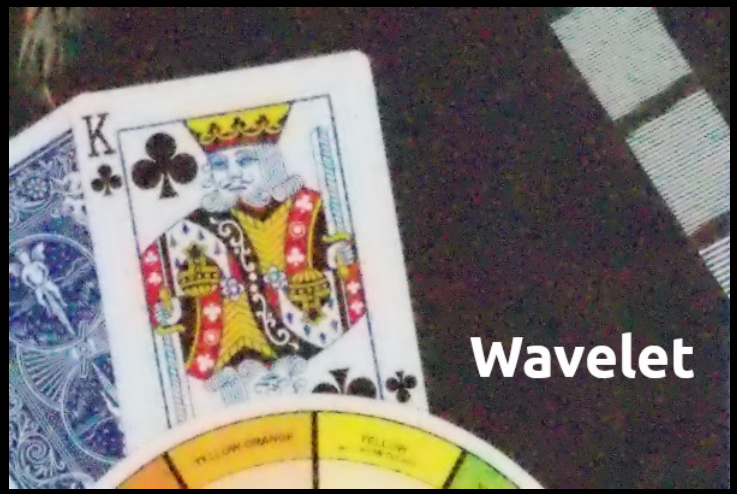
Camera sensors are improving, but a captured image always has noise. Noise reduction usually involves a combination of smoothing and sharpening or AI methods trained on a set of images. Instead, we start by empirically modeling the noise in an image as a set of probability density functions. We show a few ways that model can achieve state-of-the-art improvements in image quality, with only a fraction of the computation cost per image that alternative methods require.

University of Kentucky

What does a camera do?

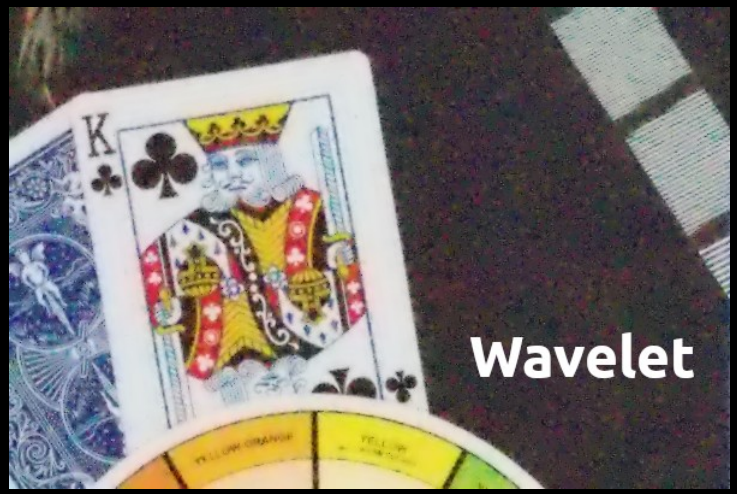
- Captures the light?
  - Time-of-flight distance measurements
  - Scientific imaging
- Makes pretty pictures?
  - Instagram filters
  - Lots of machine learning algorithms
- **Creates a model of scene appearance**

Not what the scene looked like



University of Kentucky

Fix it by modeling the noise...



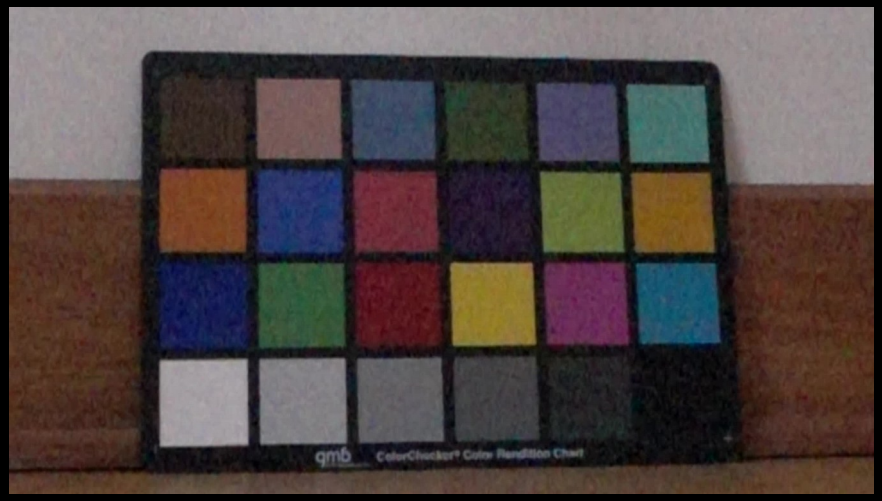
University of Kentucky

What is noise?

- Photon shot noise: statistical variations in the rate of photon emission by a light source
- Pixels Σ unit charge per photon
  - Static: QE, fixed pattern, ...
  - Dynamic: Dark current, read noise, ...
- Digital processing artifacts, encoding losses

Noise is the difference between the ***ideal pixel value*** and the ***value recorded***

Time heals *most* wounds?

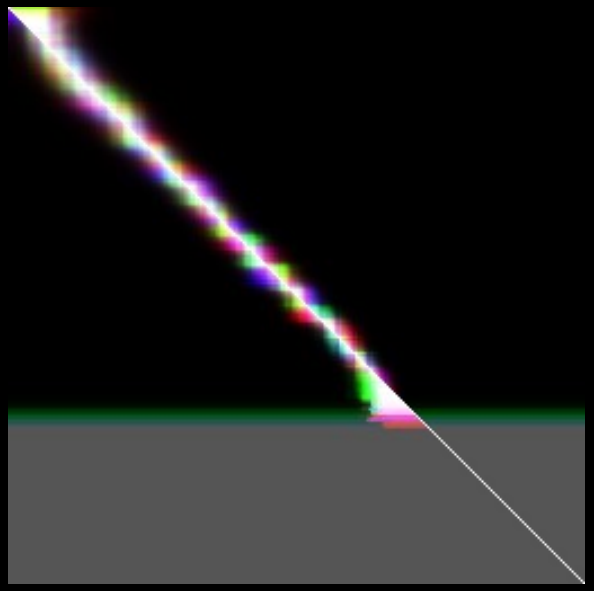


- With many samples over time, average pixel value ***tends to converge on ideal value***
- Variation in pixel value over time defines a **probability density function** (PDF) for an ideal value being sampled as any other value

University of Kentucky

What does a PDF look like for an full color image?

- A 2D color image
  - Y is **average** value
  - X is measured value
  - Normalized per-Y histogram is a PDF
  - Computed separately for Red, Green, & Blue
- Gray where no data
- Noiseless would be a thin line



University of Kentucky

Original 240FPS video

- Original 240FPS capture at 1/250s (~346°)

University of Kentucky

Same video averaging pixels while within noise bounds

- Virtual 240FPS video (original rate) with 360°
- 2015 TIK

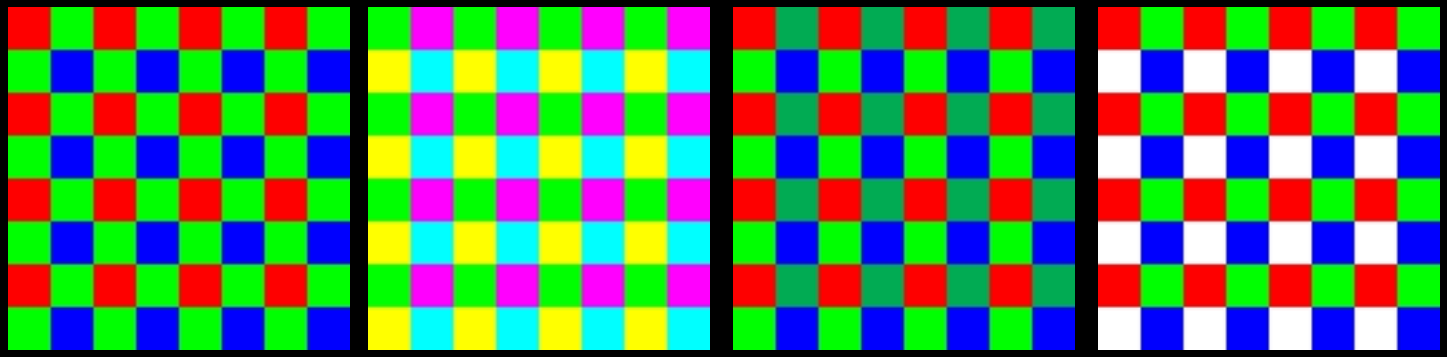
University of Kentucky

The new model using raw data

- Set of 2D PPM images
  - Y is **ideal** value
  - X is measured value
  - Scaled to actual range
  - Normalized per-Y histogram is PDF
- One image *per pattern of neighbors* in the *Color Filter Array*
- Approximated here as an RGB image

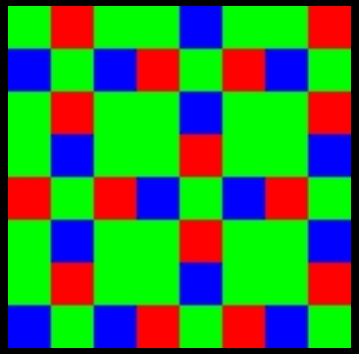
University of Kentucky

Raw image Color Filter Arrays



- Most **CFA**s use 2x2 repeat
  RG/GB, GM/YC, RE/GB, RG/WB

- Most isn't all...
  Fujifilm X-Trans uses 6x6 repeat:



Our code assumes a 2x2 repeat...

University of Kentucky

Computing the PDF model

- **TIK** computed PDF by measuring same pixel across many aligned exposures (not raw)
- Original **KREMY** PDF is interpolated from pixel variations within evenly shaded regions
- New **KREMY 2022** uses a type of histogram:
  - Most similar neighbor is same ideal value? No, but close enough...
  - If A and B might be the same ideal value, increment counts in square [A..B][A..B]

University of Kentucky

Sample new KREMY models

Rebel XT ISO 100

Iphone 7 ISO 20

A7C ISO 204800

University of Kentucky

Credible repair of raw data?

- **Texture synthesis** is normally used for **inpainting** credible values for missing pixels
- KARWY and original KREMY used model to ***adjust*** *pixel values* *within* ***bounds***
- Now, ***weight textures probabilistically***
  - **Every pixel value is replaced**
  - **Some pixel values can change a lot...**

Texture Synthesis (Simplified)

```
for (p is each pixel) {
 v = 0; w = 0;
 for (q is each same CFA context pixel) {
  wprob = patch similarity weighted
   probability p,q have same ideal value;
  v += (*q * wprob); w += wprob;
 }
 new *p = (v / w);
}
```

- Probabilities from $\texttt{model}_{\texttt{color(n)}}\texttt{[*q}_{\texttt{n}}\texttt{][*p}_{\texttt{n}}\texttt{]}$
- Weights reduce as distance increases

University of Kentucky

Implementation of new KREMY

- Use tool (e.g., **Adobe DNG Converter**) to convert raw to an uncompressed DNG
- Packages improved raw data as a DNG
  - Algorithm is only ~1200 lines C code
  - Uses **`raw2dng`** or **`dcraw`** to edit DNG data, but it could be built-into a raw processor
  - Sequential execution takes tens of seconds
- There is a WWW-interfaced version:

  `http://aggregate.org/DIT/KREMY`

University of Kentucky

The Aggregate: KREMY

localhost/cgi-bin/kremy20211031.cgi

Paused

Apps Projectors E-mount LargeFormat Everything

Current Image ID 1583550938

The above images are derived from the unprocessed raw file. First is a conventional thumbnail. Second is the pixel value error probability density function. The brightness of the pixel at X,Y in the square image maps the scaled probability that a value of X should ideally have been Y. This is empirically estimated from the image for each of four color channels in the CFA, but here is shown as a simple combined RGB image. A perfectly noiseless image would result in a probability density map with a thin white line from 0,0 to 255,255.

You can change to a different already uploaded image by entering the ID here: 1583550938

Change to image with specified ID

Delete ID 1583550938 from server

Enhancement Processing And Download

KREMY always operates on an uncompressed DNG, which is made from the submitted raw file using Adobe DNG Converter (ADC).

Click here to download the **unaltered** ADC-converted uncompressed DNG for ID 1583550938 (85574674 bytes). For some cameras, even the ADC-converted uncompressed DNG is not identical to the original raw image data, and the additional transformations performed by KREMY are not reversible, so you are advised to still keep your original raw file.

Click here to download the most recently **KREMY-enhanced** compressed DNG for ID 1583550938 (44664024 bytes).

The enhancement processing uses texture synthesis to suggest more appropriate pixel values, but those values are constrained by a pixel value error model constructed as a probability density function. The conditional probabilities are essentially raised to the 5th power because 5-pixel pattern matches are used; a strength parameter of 5 will compensate for this. However, you can set a higher or lower compensation strength, and higher numbers effectively boost the probability of accepting larger changes from the original pixel value.

5 strength

The second key parameter is the maximum distance to search for textural matches. Typically, limiting texture synthesis to consider only a relatively small area around each pixel works well; 8-16 pixels radius usually suffices. Processing time increases as the square of the texture radius.

12 texture radius

The third parameter is a selection of how many passes to make. Recursive application of this processing dramatically reduces noise, but can obliterate low-contrast gradients, resulting in somewhat cartoonish shading. Thus, 1 pass should be used unless noise survives other parameter changes. Processing time increases linearly with additional passes.

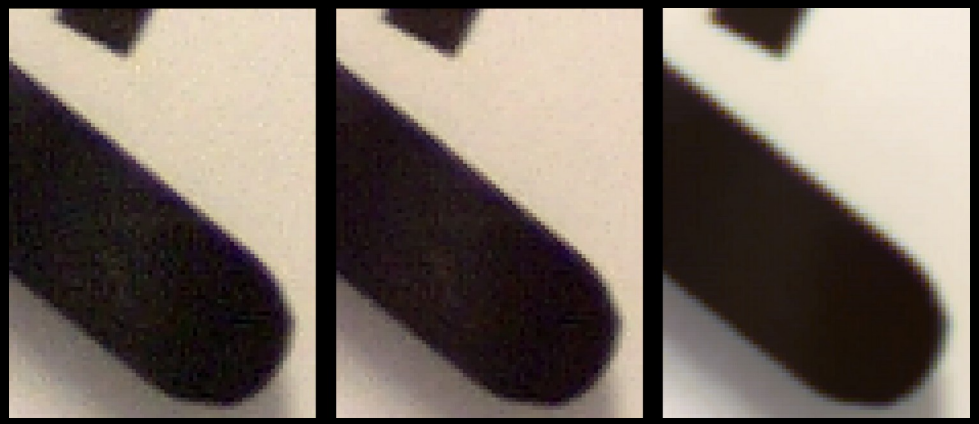
1 pass(es)

Enhance ID 1583550938 using above parameters (may take several minutes)

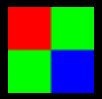
(enhanced DNG created; processing took 24 seconds)

University of Kentucky

raw | KREMY | new KREMY



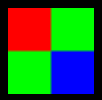
Canon Digital Rebel XT



ISO 100

new(3, 12)

Canon PowerShot S70



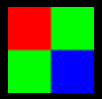
ISO 50

new(3, 12)

University of Kentucky

raw | KREMY | new KREMY

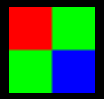
Olympus E-M1 Mark II



ISO 400

new(3, 12)

Apple iPhone 7



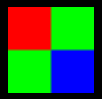
ISO 20

new(3, 12)

University of Kentucky

raw | KREMY | new KREMY

Nikon D810

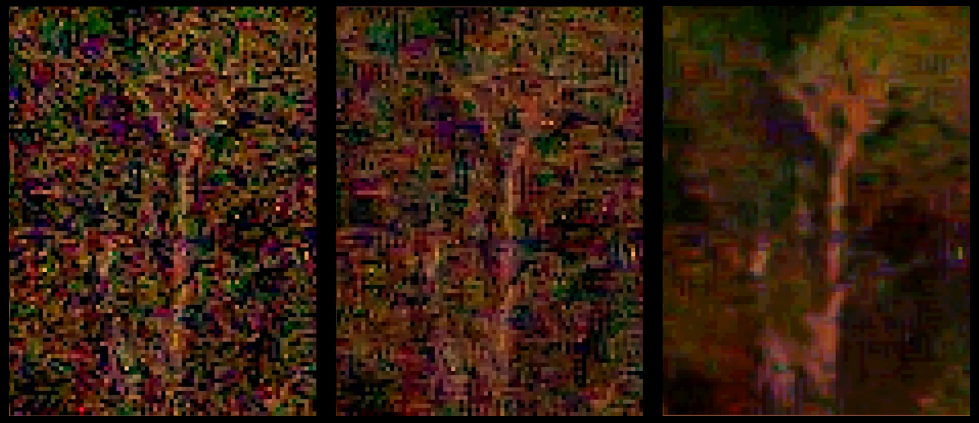


ISO 1600

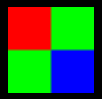
new(3, 12)

University of Kentucky

raw | KREMY | new KREMY



Sony NEX-7

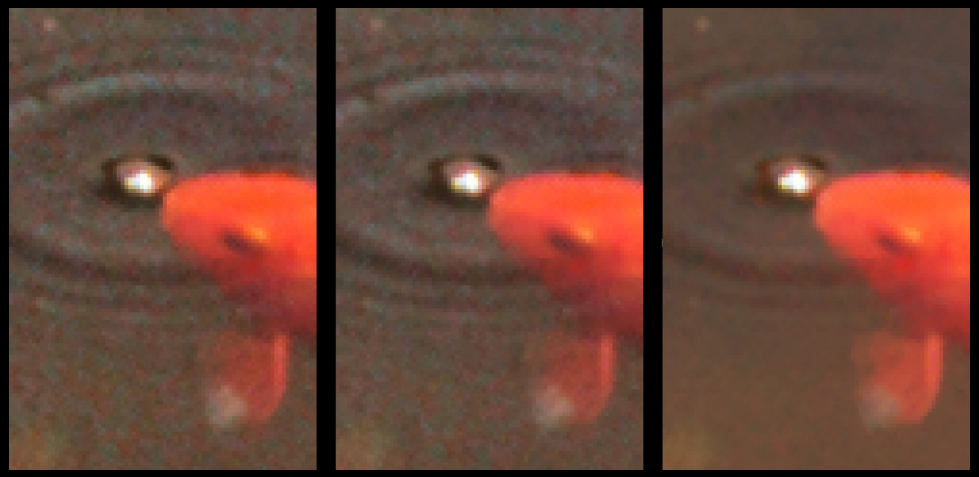


ISO 1600

new(5, 12)

University of Kentucky

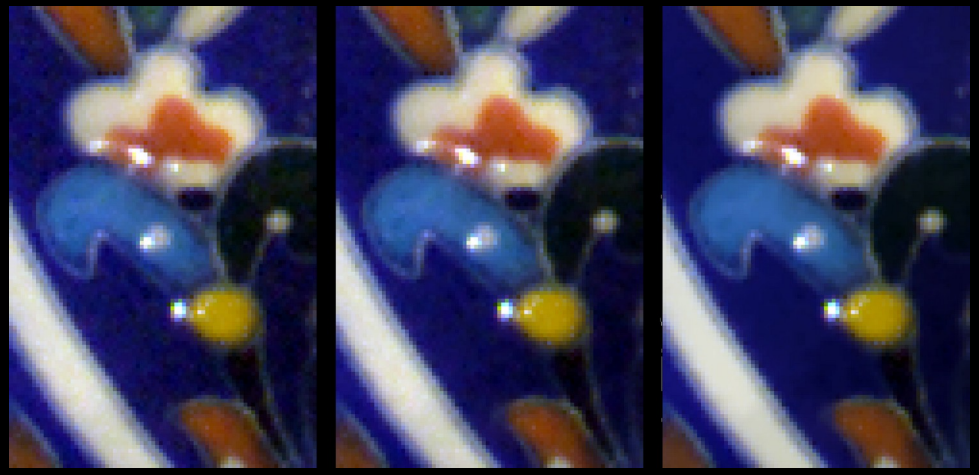
raw | KREMY | new KREMY



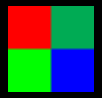
Canon PowerShot G1

ISO 100

new(3, 12)



Sony DSC F828

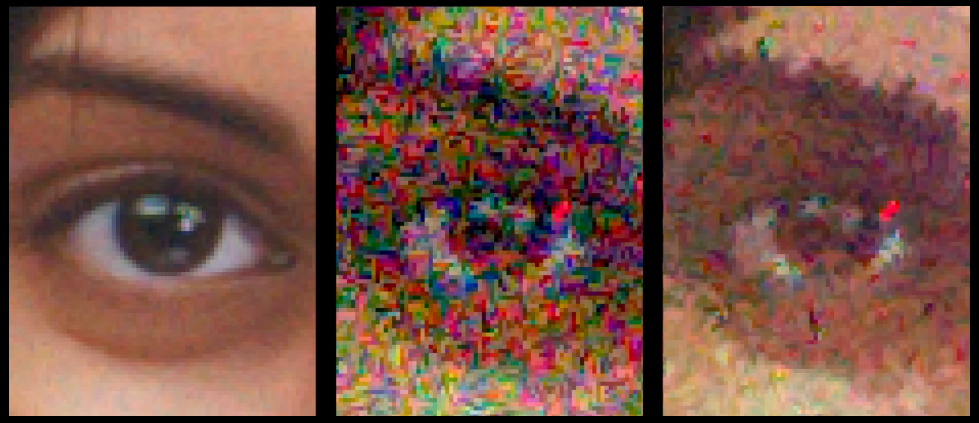


ISO 64

new(3, 12)

University of Kentucky

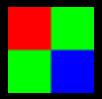
raw lo | raw hi | new KREMY



raw @ ISO 1600

raw @ ISO 204800

Sony A7C

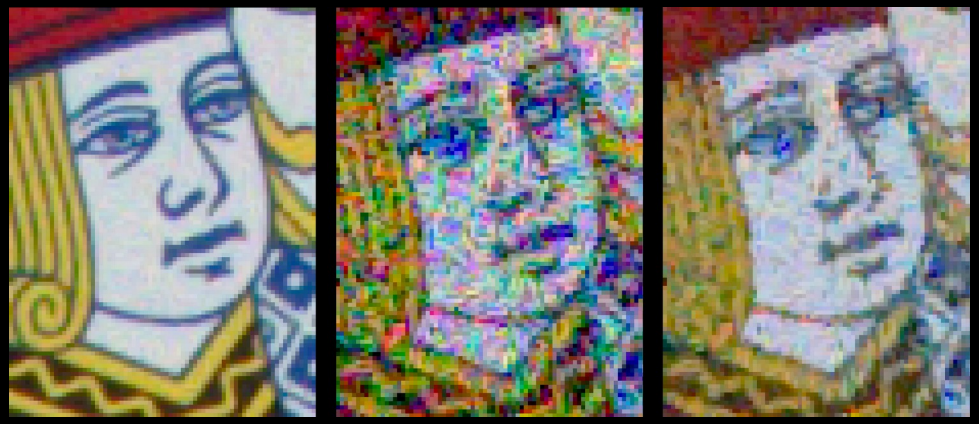


ISO 204800

new(5, 24)

University of Kentucky

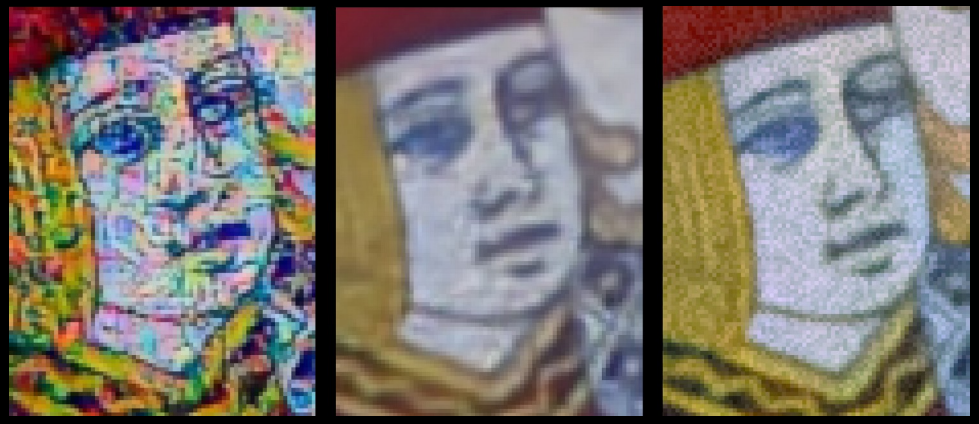
raw lo | raw hi | new KREMY



Canon R5

ISO 102400

new(5, 48)

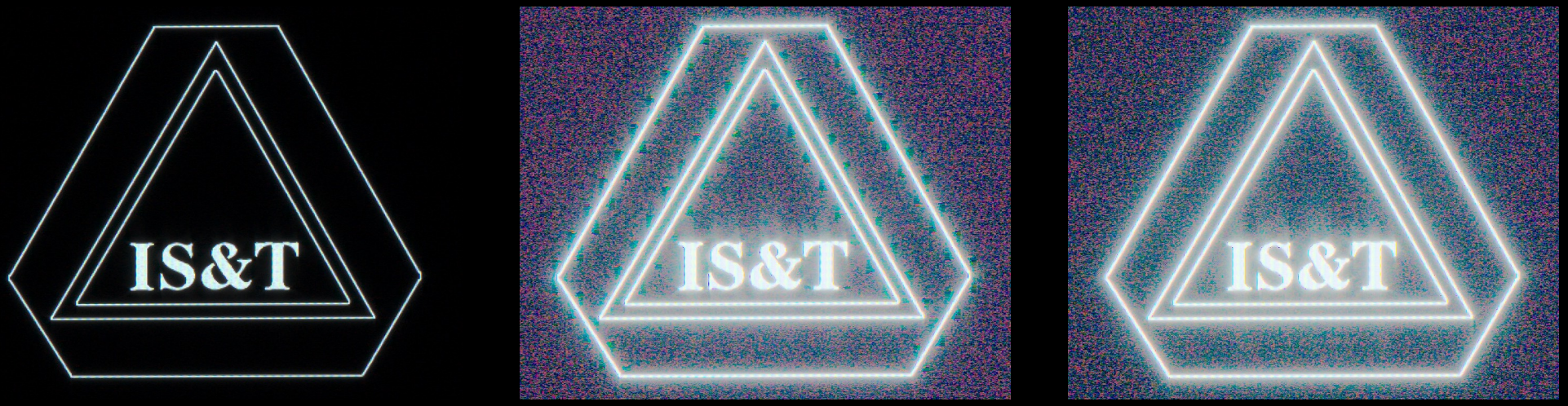


Best AI competitors:

Remini Professional
AI Image Denoiser
DxO PureRAW

University of Kentucky

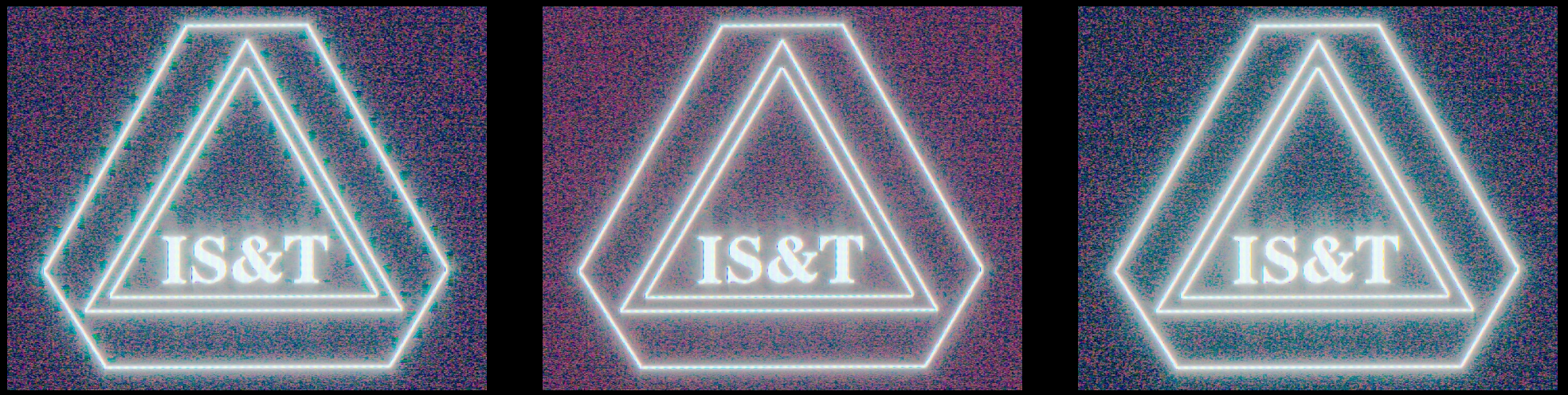
An example of encoding noise



- Sony ARW lossy-compressed raw format
  - Piecewise-linear tone mapping
  - Encoding of differences in 32-pixel blocks

University of Kentucky

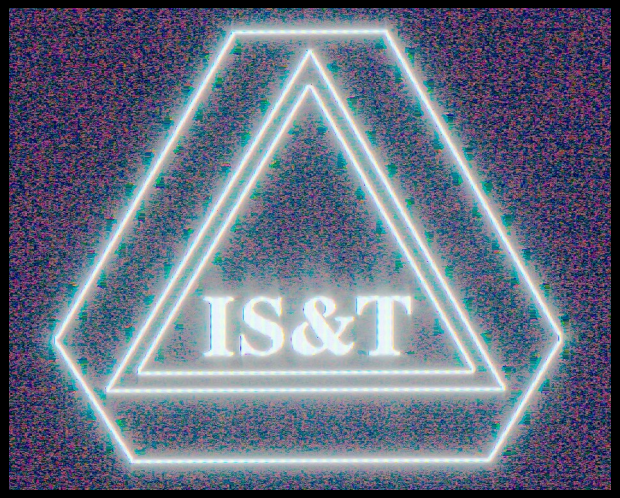
An example of encoding noise

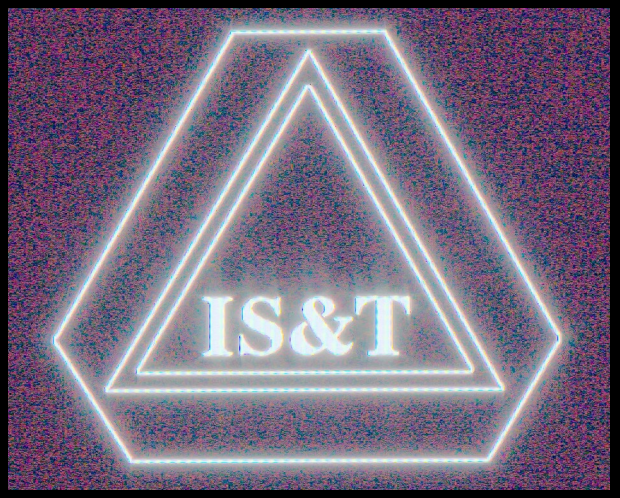


- 2015: **KARWY** computed ***per-pixel error bounds*** by reversing encoding algorithm
- Adjusted values within bounds, including adding noise matching surroundings

University of Kentucky

Does KARWY 2022 handle this?





- New KREMY doesn't model encoding error
- Or does it? Using the PDF-driven texture synthesis reduces noise enough to make encoding artifacts imperceptible

University of Kentucky

Conclusions

- Probability-based pixel value model works
  - Can compute from a single capture
  - Models combined effect of all noise
  - **Probability density image** is efficient to use, editable, and visually meaningful
  - Probabilities model more extreme noise
  - Can be fast enough to execute in camera
- Texture synthesis in new KREMY
  - Simple, untrained, deterministic, algorithm
  - Very effective ***credibly improving raws***