Image Data Compression

Class Introduction
About the Instructor (me)

- **2001**: Novosibirsk State University (Russia), BSc in Physics/Informatics
  - Participated in writing Technical Design Report on TESLA Electron-Positron Linear Collider

- **2003**: Novosibirsk State University, MSc in Experimental Particle Physics
  - Collaboration member, data analysis in BELLE collider experiment in Tsukuba, Japan.

- **2008**: University of Alberta (Canada), PhD in Theoretical Particle Physics
  - Performed large-scale computer-aided analytic calculations in quantum field theory.

- **2008-2011**: KIT, Institute for Theoretical Particle Physics (TTP), Postdoctoral Researcher
  - Computed higher-order quantum corrections to Higgs boson production at LHC.

- **Since 2011**: Fraunhofer IOSB (Karlsruhe), Leader of 3DIM Research group
  - 2011-2017: KIT, Institute for Anthropomatics, IES Laboratory, Scientific Advisor
  - Feel free to explore new HiWi and research positions at [www.ies.anthropomatik.kit.edu](http://www.ies.anthropomatik.kit.edu)!

**Primary research areas**: Optical Metrology, 3D Reconstruction, Probabilistic Graphical Models, Inverse Problems, Sensor Data Fusion, Tracking.

- Contact: alexey.pak@iosb.fraunhofer.de, Fraunhoferstrasse 1, Tel. **0721-6091-667**
Books to read

  - (Theoretical and technical details of many formats, such as JPEG, MPEG, H.264)

  - (Foundations of optics, image acquisition and exploitation)

  - (Detailed overview of both subjects and relevant theoretical results)

  - (General information theory and communication theory)

  - (Digital image processing, including compression, enhancement, restoration etc.)
Why is data compression necessary?

We now have better CPUs and storage devices, but still need data compression:
• New sensors, faster digital transmission and processing allow better signal quality
• Data sizes grow on par (or faster) with storage or bandwidth capacity
• New applications: archiving, digital TV, video surveillance, telemedicine, search...
• Internally, modern CPU/GPU/supercomputer architectures are bandwidth limited; data compression is used even to send data to the next computing unit!
• In fact, efficient data compression becomes ever more crucial to modern applications!

Specific features of image and video data (as opposed to “common” data):
• 2D (or 3D) multi-channel signals (or higher-dimensional, e.g. with 3D-information)
• Pixel values are spatially and/or temporally correlated
• No simple and clear quantitative measure of “image / video quality”
• “Typical” natural images extremely non-random, very complex distribution

Example: an HD movie
(3 bytes/pixel) * (1920 x 1080 pixels/frame) * (24 frames / sec) * (1h = 3600 sec) = 5.37×10^{11} bytes. 

Compare to:
1-hr movie, MPEG compression: ~3 Gb = 3.2×10^{9} bytes!
Redundancy: how much information does an image contain?

A “natural” image: short and long-range correlations between pixels, un-even distribution of pixel values, some areas can be “predicted” from its neighborhood (e.g. sky color).

Example of an un-natural image: pixel values are random, no strong correlations at any scale.

When pixel values are not independent, each value contains redundancy, which can be removed in order to compress data without losses.

Above image: (2880 x 1800 pixels) * (3 bytes / pixel) = 15.5×10^6 bytes; however, can be stored without losses in a 9.9 Mb PNG file.
Relevance and irrelevance: **useful** information in an image

Given an image, we can exploit it differently depending on the given task:

1) Task (Computer Vision):
   - Identify people
   **Useful information:**
   Number of persons, names.

2) Task (Computer Vision):
   - Identify all cars
   **Useful information:**
   Car size, position, orientation

3) Task (Human Observer):
   - Enjoy aesthetical value of the image as an art piece
   **Useful information:**
   ?
   Perhaps everything perceivable by humans...

Useful information is task-dependent; the remaining information in the image is irrelevant. Lossy compression removes irrelevance from image preserving its usefulness for given task (i.e. task is tolerant to the introduced changes).
NTSC color TV standard (1953-2009):
- Artifacts “minimally noticeable” by humans

Discretization and sampling:
- 525 scan lines, 29.97 frames/sec (interlaced)

Transform:
- Luminance-chrominance color encoding: YCrCb (not RGB!)
- Luminance signal: ~6 MHz band
- Truncate chrominance signal: ~1.5 MHz, less resolution

- Lossy compression: irreversible removal of irrelevant information
- Compare to AI: an algorithm has to understand what a human brain can or cannot perceive.

[Diagram of compression process]

Source → Transform → Lossless coding → Channel coding
---

Remove irrelevance → Remove redundancy

Codec

Channel coding Modulation Channel Demodulation Error correction

Sink → Inverse transform → Decoding

Inject channel-adjusted redundant information (error protection)

Exploit, remove injected redundancy
What is this course about?

**Primary issues discussed in this class:**

- Origins and nature of information represented by images
- Statistical properties of “natural” images
- Human perception of visual information
- Different forms of image data representation
- Qualitative and quantitative metrics of information in images
- Changes in quality and quantity of information during common (digital) transformations and manipulations with images
- Non-conventional ways to exploit the information contained in images

Questions?
Image Data Compression

Data Reduction Techniques
Unavoidable data reduction: discretization/digitization

Continuous 2D signal (light intensity on sensor)

Fully digital signal

Discrete time signal (pixel voltage readings)

Position-dependent signal

Discrete value signal (# of electrons at CCD pixel)

Spatially discrete signal (pixel-averaged intensity)

Analog signal (light intensity at a pixel)

Time-dependent signal
Sampling a continuous-time signal

Continuous-time continuous-value signal (i.e. sampled at very high frequency)

Discrete-time continuous-value signal (re-sampled at given rate)
Simple data reduction: sub-sampling

$$x[n] \rightarrow y[m] = x[m \cdot M]$$

DFT: $$X[f] = \sum_{n=1}^{N} x[n] \cdot e^{-2\pi i (n-1)(f-1)/N}$$

- Discrete Fourier Transform

Original signal

Sub-sampled signal with $$M=3$$

Sampling rate: number of samples per unit of time

$$f_s(x)$$ - signal bandwidth

$$f_s(x)/2$$ - symmetry axis

$$f_g(x)$$ - sampling frequency (rate)

$$f_s(x) < \frac{f_s(x)}{M}$$

Bounds on sub-sampling:

|$$X[f]$$|

|$$Y[f]$$|

$$f_s(x) = \frac{f_s(x)}{M}$$

Anti-aliasing filtering + Sub-sampling = Decimation
Simple data reduction: up-sampling

\[ x'[n] = \begin{cases} 
  y[n/L], & n = m \cdot L \\
  0, & \text{otherwise} 
\end{cases} \]

The simplest way to find intermediate values:

- Add zeros between known values

Up-sampling + Anti-imaging filtering = Interpolation

Given signal sampled at low-frequency, recover values at higher frequency
Quantizing a continuous-value signal

Continuous-value signal (i.e. a large number of possible signal values)

$$g(t)$$

Discrete-value signal (limited number of chosen signal values)

$$g_q(t)$$
Simple data reduction: scalar quantization

Mapping from continuous 1D real-valued signal value to quantized (discrete) values:
- Quantization: \( x \rightarrow q \)
- Reconstruction: \( q \rightarrow y_q \)
- Quantization error: \( e_q = x - y_q \)

Uniform quantization with interval size \( \Delta \) – two important types:

- Midtread quantizer:
  \[
  q = \Delta \left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor \\
  y_q = q \cdot \Delta
  \]

- Midrise quantizer:
  \[
  q = \Delta \left( \frac{x}{\Delta} + 0.5 \right) \\
  y_q = (q - 0.5) \cdot \Delta
  \]

Assuming uniformly distributed signal, power of error in both cases is:

\[
p(e_q) = \begin{cases} \frac{1}{\Delta}, & |e_q| \leq \frac{\Delta}{2} \\ 0, & \text{otherwise} \end{cases}
\]

\[
\left\langle e_q \right\rangle = 0
\]

\[
\left\langle e_q^2 \right\rangle = \int e_q^2 \cdot p(e_q) \cdot de_q = \frac{\Delta^2}{12}
\]
Non-uniform quantization: simple cases

quantizer with a dead zone:

quantizer with limited amplitude:

quantization error:
Non-uniform quantization: PDF-optimized

Choice of reconstruction values:

- Consider interval: \( I = [x_q, x_{q+1}] \)
- Signal probability distribution function (PDF): \( p(x) \)
- To find: reconstruction value \( y_q \)
- Objective: minimize average error, \( \langle e_q \rangle_I \rightarrow 0 \)

\[
\begin{align*}
I &= x_q \cdot x_{q+1} \\
\langle x \rangle_I &= \frac{\int_{x_q}^{x_{q+1}} x \cdot p(x) \cdot dx}{\int_{x_q}^{x_{q+1}} p(x) \cdot dx}
\end{align*}
\]

Choice of interval boundaries:

Algorithm [Max (1960), Lloyd (1982)]

Given: \( p(x) \), initial boundaries \( \{x_q\} \)

repeat

\[
\begin{align*}
y_q &\leftarrow y_q^I(\{x_q\}) \\
x_q &\leftarrow \frac{y_q + y_{q+1}}{2}
\end{align*}
\]

endrepeat

Result: all intervals become equally likely
Other types of non-uniform quantization

Perception-optimized

- Variance of quantization error does not exactly correspond to how distortions are perceived; usually strong dependence on amplitude (cf. sound)
- Minimize power of average perceived error (computed with models, human tests, ...)
- Hard to optimize for all possible cases and applications!

SNR (Signal-to-noise ratio) – optimized quantization:
introduce some compander (Compressor+Expander) function \( c(x) \), pre-process signal

\[
x'' = c^{-1}([x']_Q)
\]

\[
x' = c(x)
\]

\[
[x']_Q = Q(x')
\]

\[
x'' = c^{-1}([x']_Q)
\]
Vector quantization: multi-dimensional signals

$g(x, y)$

$g(x_a, y_b)$

$\vec{v} = (R, G, B)$

$(R, G, B) \rightarrow v_q$
Vector quantization as classification problem

- **Quantization:** $\tilde{x} \rightarrow q$
- **Reconstruction:** $q \rightarrow \tilde{y}_q$
- **Codebook:**
  $$Y = \{\tilde{y}_i : i = 1, 2, \ldots, N\}$$
- **Distance metric:**
  $$d(\tilde{x}, \tilde{y}) \in R^+$$
- **Voronoi region:**
  $$V_i = \{\tilde{x} \in R^k : d(\tilde{x}, \tilde{y}_i) \leq d(\tilde{x}, \tilde{y}_j), \forall j \neq i\}$$
- **Non-overlapping space partitioning:**
  $$\bigcup_{i=1}^{N} V_i = R^k$$
  $$V_i \cap V_j = \emptyset \forall i \neq j$$
- **Example:** Euclidean metric
  $$d_e(\tilde{x}, \tilde{y}) = \sqrt{\sum (x_i - y_i)^2}$$
Vector quantization: codebook design

- Optimal design of a codebook for given input vectors is NP-hard (classification problem)

**LGB: simple (sub-optimal) algorithm** [Linde, Buzo, Gray ’80]:
1. Determine size N of the codebook
2. Select N random codewords as initial codebook Y
3. Classify set of M training vectors (i.e. find $V_i$ for each $x_k$)
4. Compute new codewords as means in each cluster:
   \[
   \vec{y}_i \leftarrow \frac{1}{N_i} \sum_{\vec{x}_k \in V_i} \vec{x}_k
   \]
5. Repeat steps 2 and 3 until convergence (i.e. changes in codewords become small)
6. Alternatively: threshold on relative quantization error change, $\Delta D / D < \varepsilon$:
   \[
   D = \frac{1}{M} \sum_i D_i, \quad D_i = \frac{1}{N_i} \sum_{\vec{x}_k \in V_i} d(\vec{x}_k, \vec{y}_i)
   \]
   (cf. k-means algorithm)

- There are other methods: Pairwise Nearest Neighbour (PNN), Simulated Annealing, Maximum Descent (MD), Frequency-Sensitive Competitive Learning (FSCL), etc.