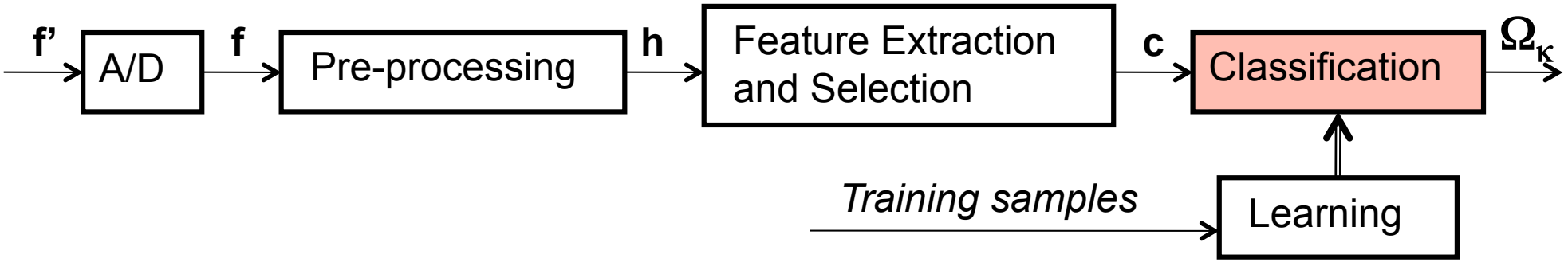
Self-Organizing Maps

Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Pattern Recognition Pipeline



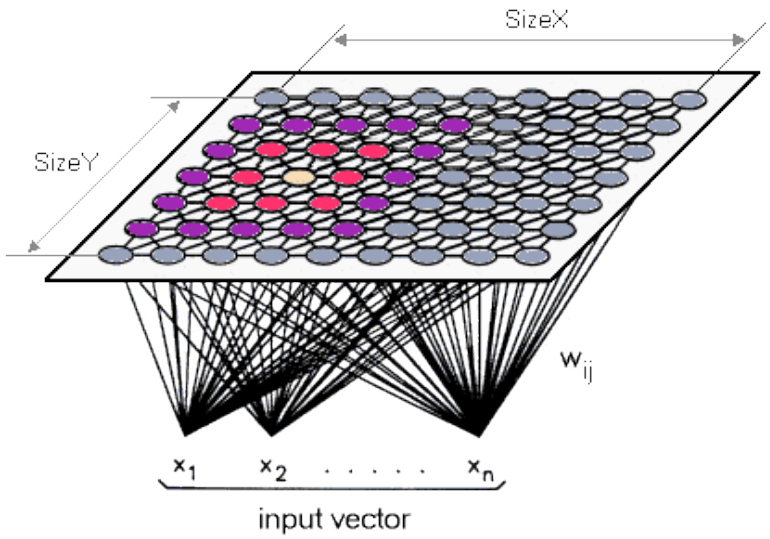
- Classification
  - Supervised Classification
    - Statistical classifiers
    - Polynomial classifiers
    - Non-Parametric classifiers
  - Unsupervised Classification
    - Self-Organizing Maps

Unsupervised Classification

- Based on unlabelled data.
- There is still a database of example input samples.
- There is **no label** associated with each sample. We do not know to which class each sample belongs.
- Unsupervised classification is a harder problem.
- Sometimes we do not have the expertise to label tha data.
- Classifier has to derive some form of similarity between the different samples presented to it.

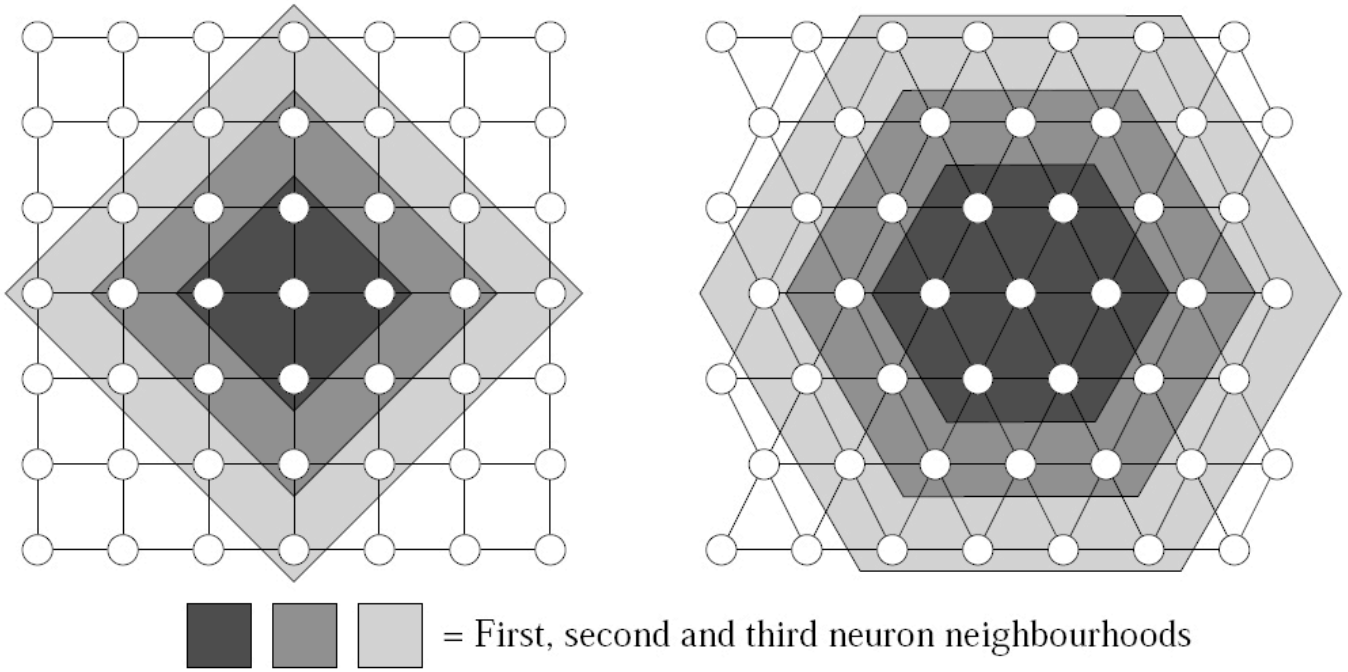
Basic Concept of SOMs

- A Self-Organizing Map, or Kohonen Map or Kohonen network is a type of Artificial Neural Network.
- It is composed of a lattice (grid) of neurons in the hidden layer.
- Each hidden neuron, n_{ij}, has:
  - a weight vector (sometimes referred to as codebook vector), $\vec{w} \in R^d$
    It has the same dimensionality as the input vector, $\vec{c} \in R^d$
  - a position (i, j) in the lattice (or map) space
  - an associated label, Ω_κ - optionally



Basic Concept - continued

- The nodes in the hidden layer can have different types of connectivity.
- Common layouts include the rectanular grid and a hexagonal grid.



Applications of SOMs

- Originally developed by Teuvo Kohonen as a means of representing high dimensional data in lower dimensions.
- SOMs are often used in dimensionality reduction.
- SOMs are also used in unsupervised classification, as they can cluster data in groups of similar value.
- SOMs are commonly used in vector quantization.

Operation of SOMs

- As in most Artificial Neural Networks, SOMS have two modes of operation:

1. Training

   During training a map is built using the training samples.

2. Mapping

   During mapping a new sample is automatically classified by using the map created during training.

SOM Training

1) Initialize each node's weights.

2) Choose a random vector from training data and present it to the SOM.

3) Every node is examined to find the Best Matching Unit (BMU).

4) The radius of the neighborhood around the BMU is calculated. The size of the neighborhood decreases with each iteration.

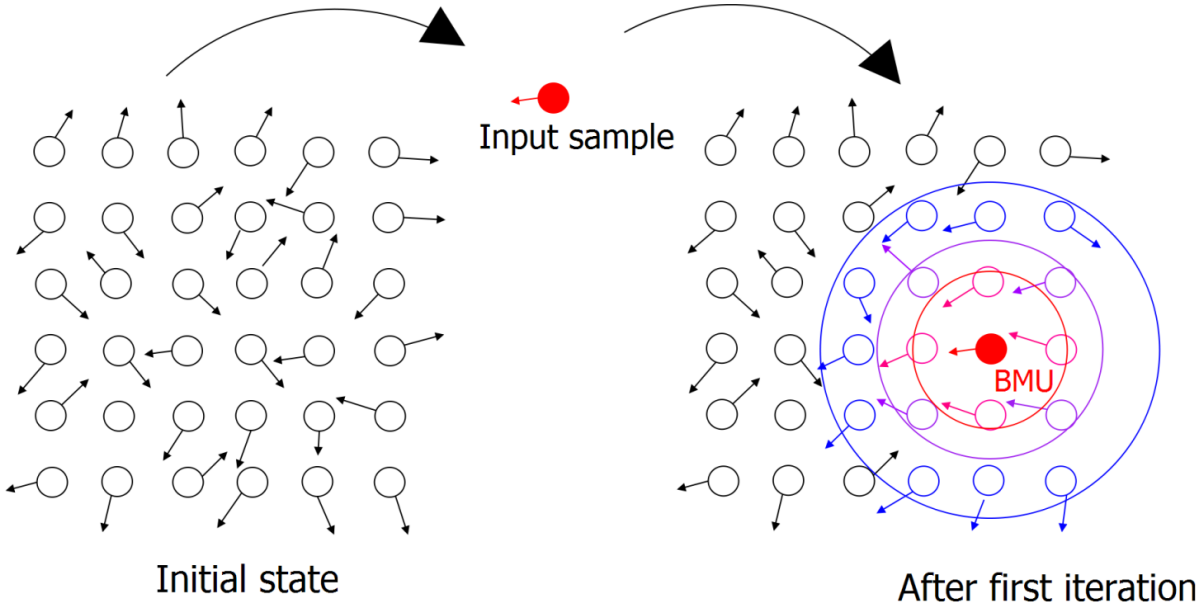
5) Each node in the BMU's neighborhood has its weights adjusted to become more like the BMU. Nodes closest to the BMU are altered more than the nodes furthest away in the neighborhood.

6) Repeat steps 2-6 for until convergence.

Calculating the Best Matching Unit

- Calculating the BMU is done according to the distance (often Euclidean distance) between the node's weights (w_1, w_2, … , w_d) and the input vector's values (c_1, c_2, … , c_d).
  - This gives a good measurement of how similar the two sets of data are to each other.

$$Dist = \sqrt{\sum_{i=0}^{d} (c_i - w_i)^2}$$



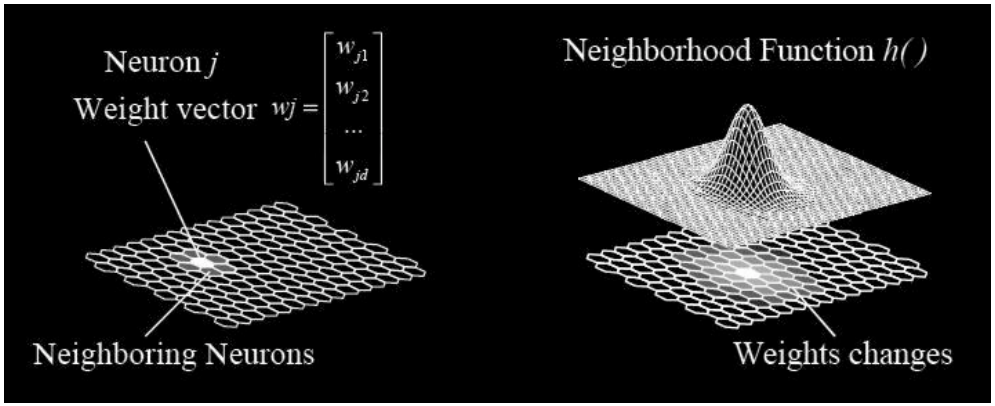
Determining the BMU neighborhood

- Size of the neighborhood: An *exponential decay* function is often used. It shrinks on each iteration until eventually the neighborhood is just the BMU itself.

$$\sigma(t) = \sigma_0 \; exp\left(-\frac{t}{\lambda}\right)$$

- Effect of location within the neighborhood: The neighborhood is defined by a gaussian curve so that nodes that are closer are influenced more than farther nodes.

$$\Theta(t) = exp\left(-\frac{dist^2}{2\sigma^2(t)}\right)$$



Modifying Nodes' Weights

- The new weight for a node is the old weight, plus a fraction (L) of the difference between the old weight and the input vector… adjusted (theta) based on distance from the BMU.

$$W(t+1) = W(t) + \Theta(t)L(t)\left(V(t) - W(t)\right)$$

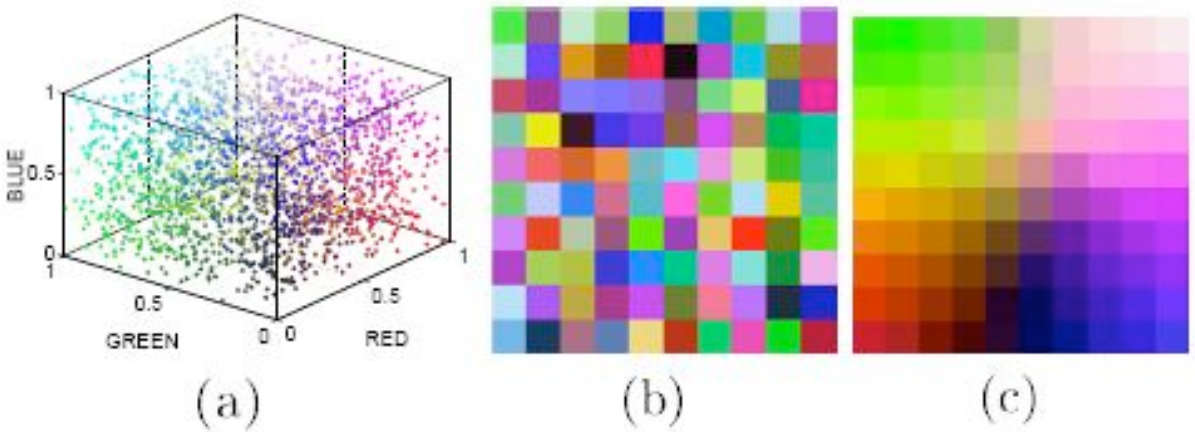
- The learning rate, L, is also an exponential *decay* function.
  - This ensures that the SOM will converge.

$$L(t) = L_0 \; exp\left(-\frac{t}{\lambda}\right)$$

- The lambda represents a time constant, and t is the time step

Simple Example

- 2-D square grid of nodes.
- Inputs are colors.
- SOM converges so that similar colors are grouped together.



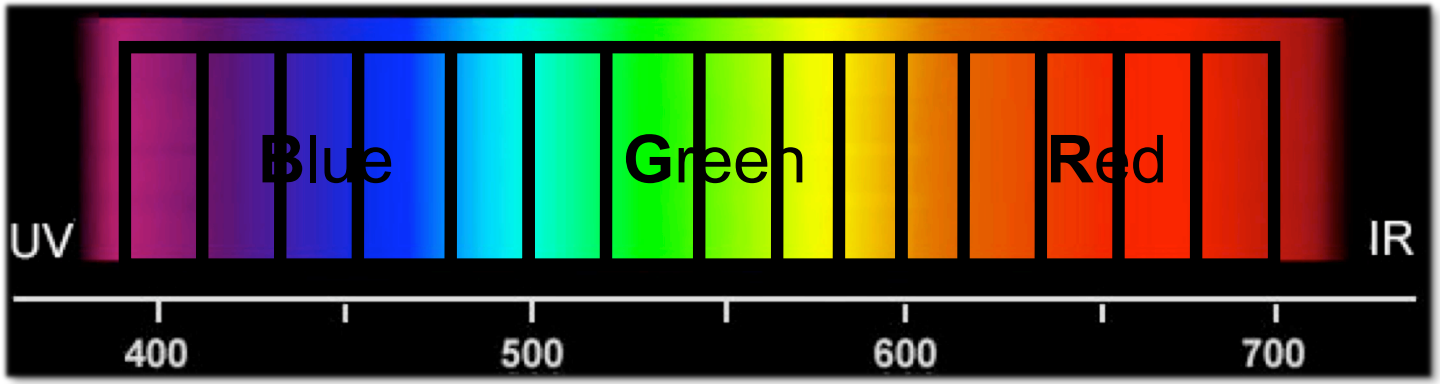
a) Input space
b) Initial weights
c) Final weights

A Very *Fresh* SOM Example

- We currently use SOMs at LME to classify multispectral data in classes of similar color.
- Why?
  - Dimensionality Reduction
  - Edge Detection
  - Segmentation
- Challenge: Which colors are considered similar in high dimensions?

Multispectral Imaging

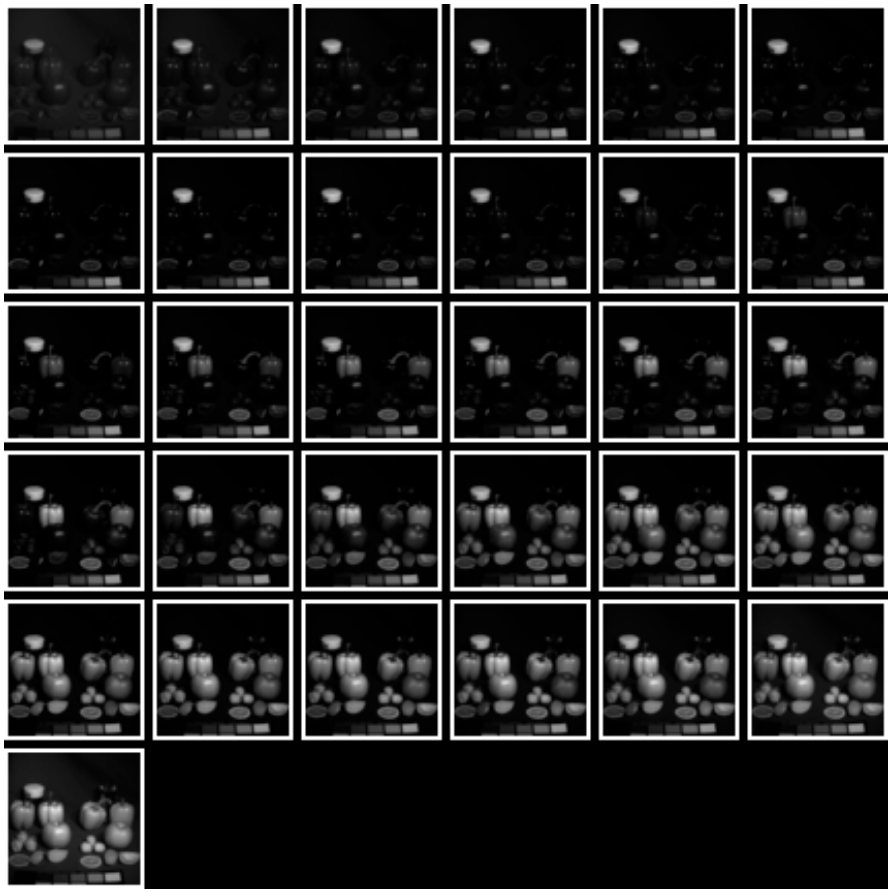
- RGB: composed of three intensity images



- Multispectral: composed of **N** intensity images
  - Bandfilter capture energy over smaller range of wavelengths
  - Many small bands (e.g. 30) → higher spectral resolution

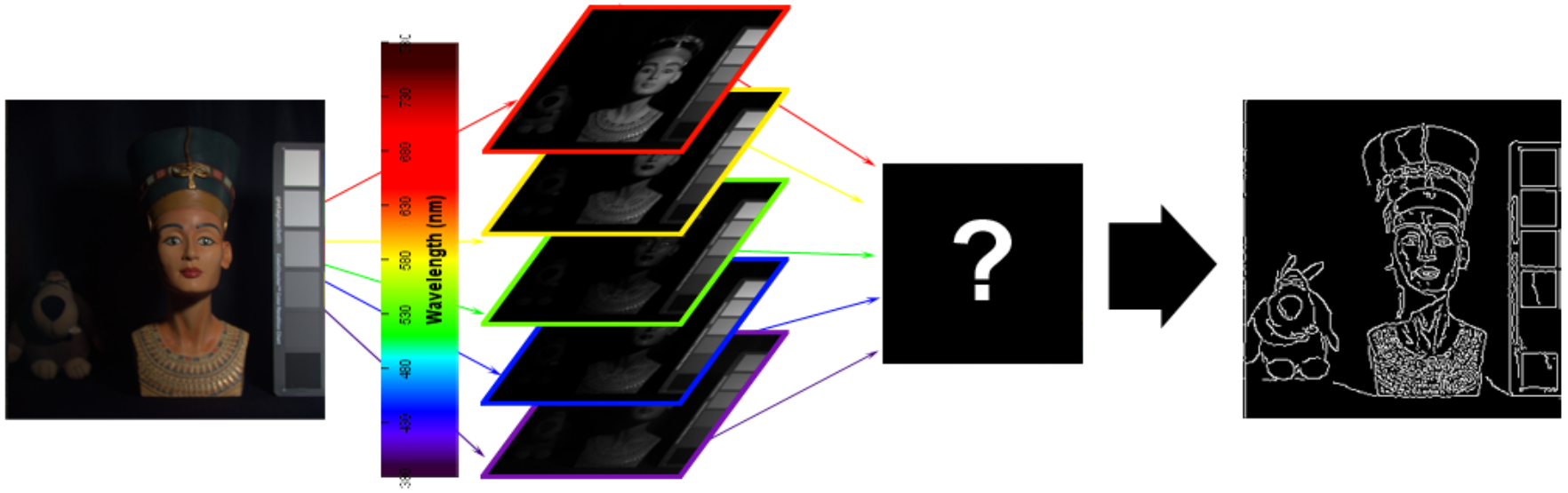
Multispectral Cube

Scene with fake and real fruits and vegetables captured with an RGB camera.



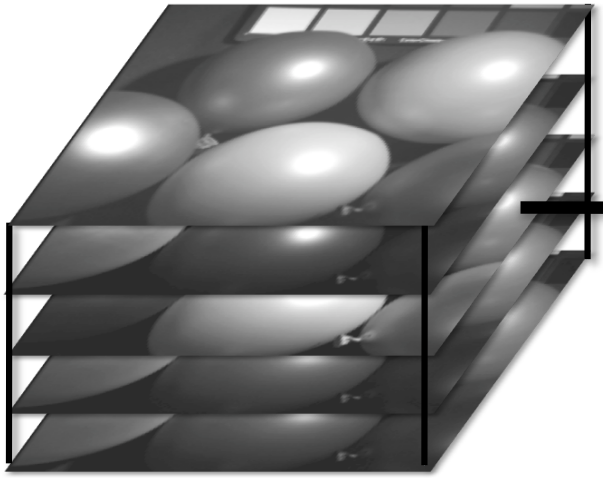
Same scene with captured with a multispectral camera with 31 color bands.

Edge detection on multi-spectral images (MSI)



- Aims:
  - Generate edge image out of multiple spectral bands
  - Reveal information that is hidden by normal RGB cameras
  - Keep artificial/false edges minimal

Edge Detection using SOM (Toivanen et. al, 2003)

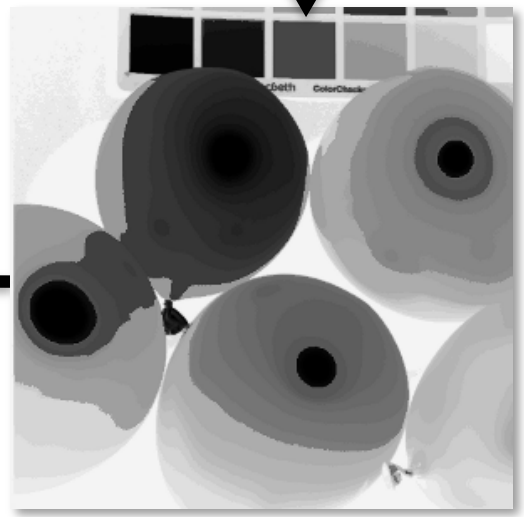


Multispectral image

SOM

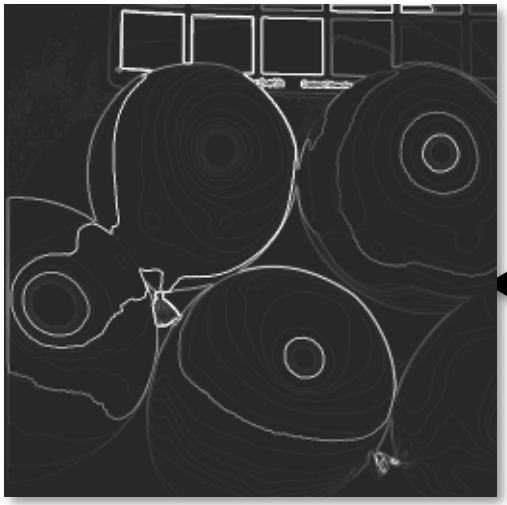
Linearization

Pixel ordering



Grayscale image

Grayscale edge detector



Edge image

SOM – Learning Process

- At iteration $t = 0$

  Initialize neurons randomly with a uniform distribution

- For $0 < t \leq t_{max}$

  Present representative input data to the SOM

  a) Randomly select a multispectral pixel vector $\boldsymbol{x^t}$

  b) Find the neuron $\boldsymbol{m_c^t}$, that is closest to this input vector

$$\left\|\boldsymbol{x^t} - \boldsymbol{m_c^t}\right\|_2 = \min_i \left\{ \left\|\boldsymbol{x^t} - \boldsymbol{m_i^t}\right\|_2 \right\}, \quad i = 1...M$$

  c) Shift this neuron and its neighborhood in direction of the input

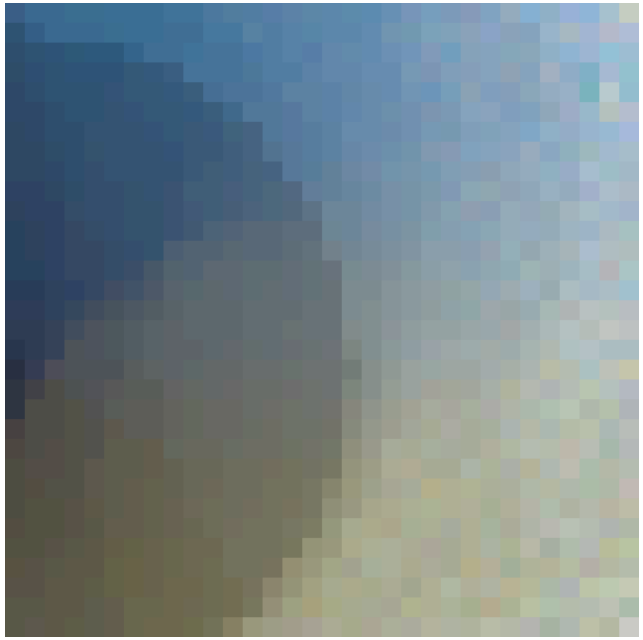
$$\boldsymbol{m_i^{t+1}} = \boldsymbol{m_i^t} + \epsilon^t \cdot h_c^t \cdot \left[\boldsymbol{x^t} - \boldsymbol{m_i^t}\right]$$

ϵ^t : time-dependent learning rate

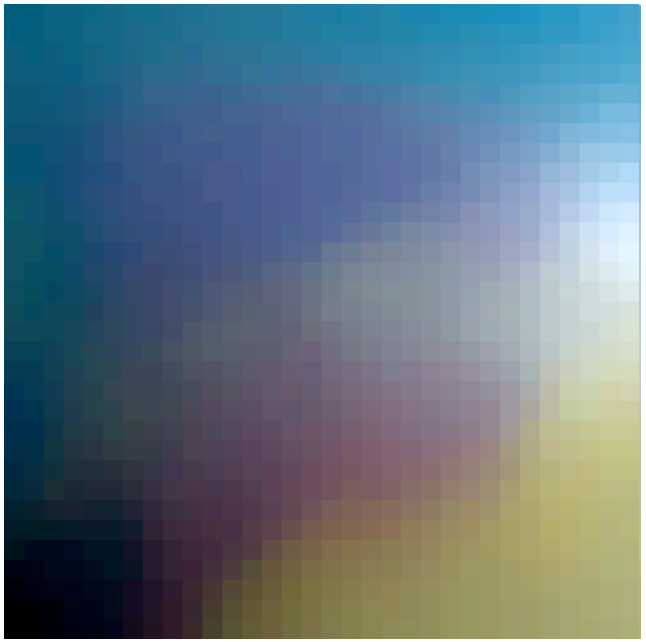
h_c^t : time and topolgy-dependent neighborhood function

SOM – Learning Process: Example

Input



100 iterations



5000 iterations

→ Clustering of the input space

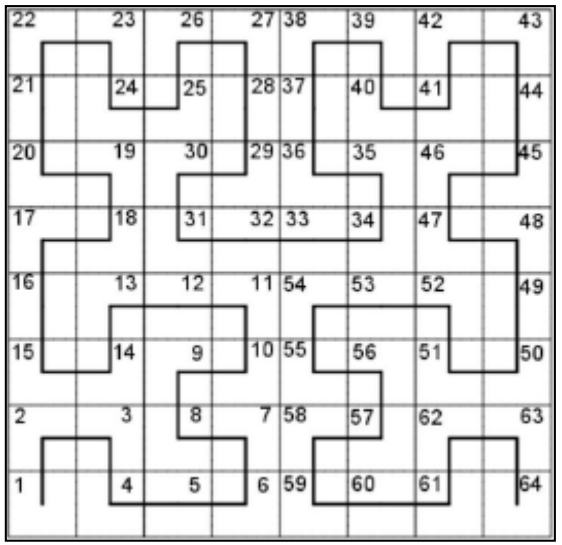
→ „Similiar colors“ move close to each other

Linearization of Space

- Aim: Mapping of SOM neurons to scalar values
  - Constraint: Neuron vectors that are spatially close to each other should also get similiar indices

- Solution: Traversing SOM with space-filling curve (e.g. Hilbert, Peano)

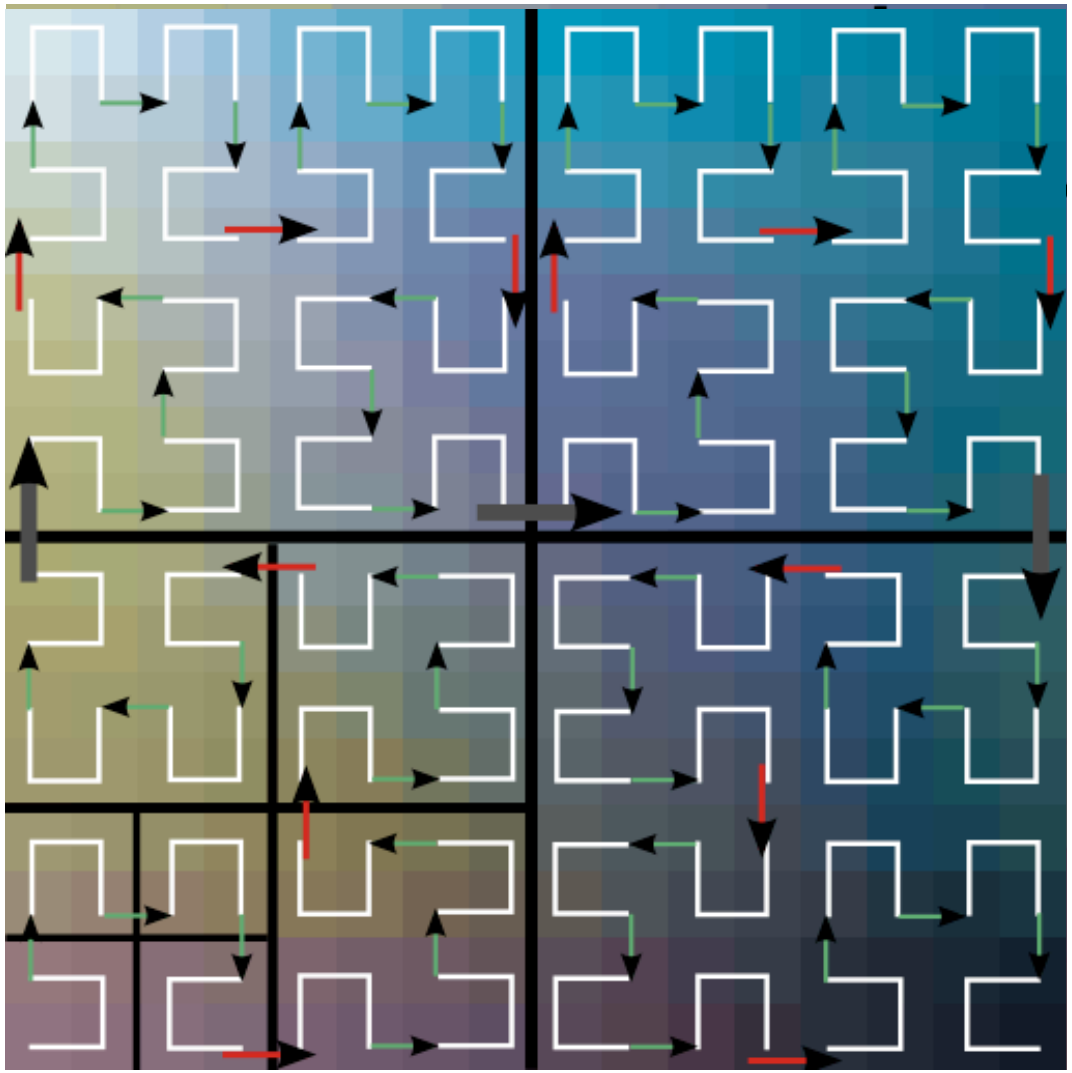
  → Ordering of neurons in SOM



Hilbert curve: 3rd order

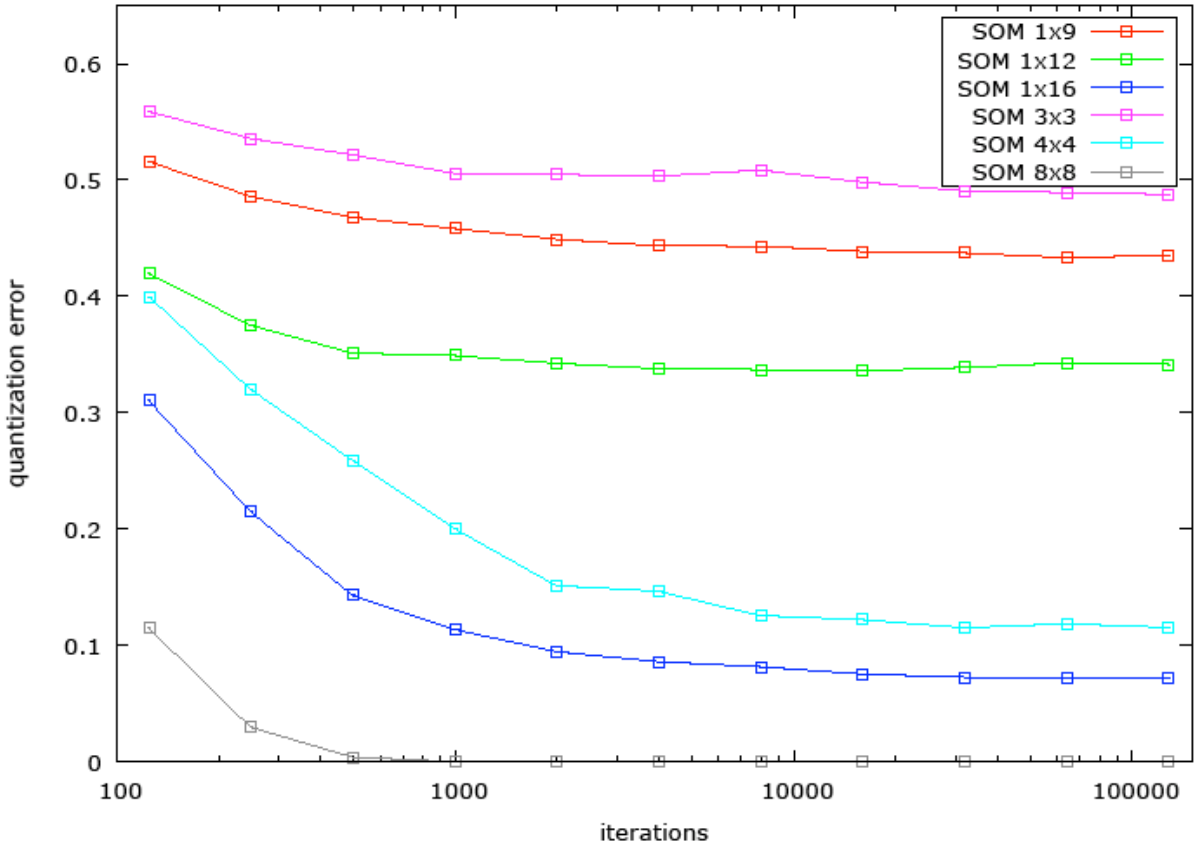
- Conversion to grayscale image
  - Determine best matching neuron for each multispectral pixel
  - Rank of this neuron becomes new pixel intensity

Hilbert-curve Construction



2nd order

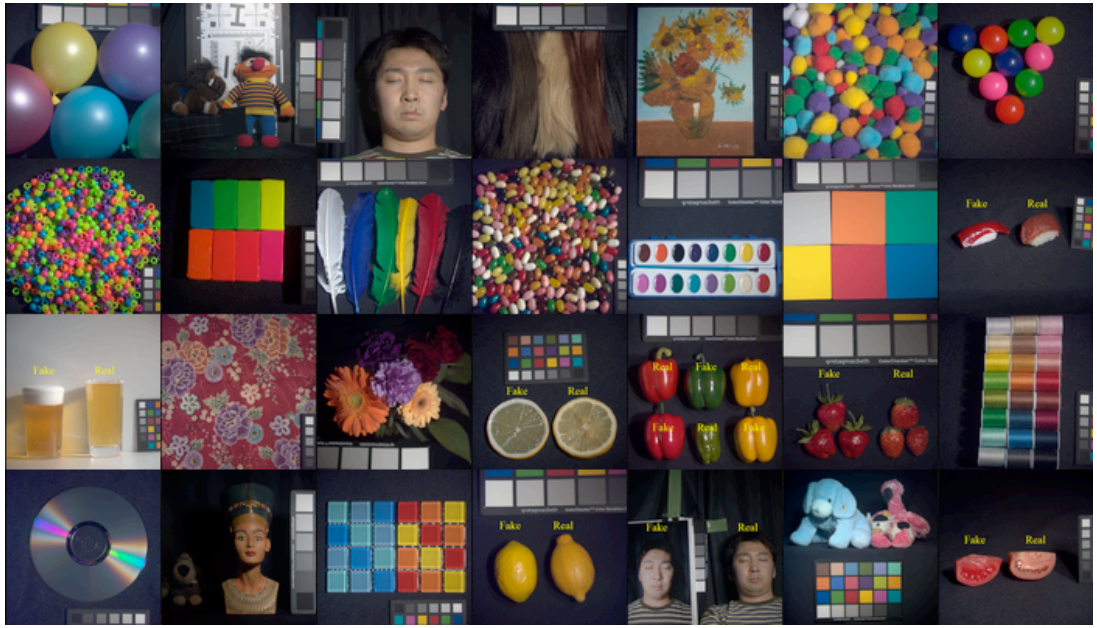
SOM – Learning Performance



- More neurons → lower quantization error
- 2D SOMs need some minimal size to evolve properly
- Remark: Linear correlation of running time and iterations on all SOMs

Real Data

- Cave - Multispectral image database
  - Wavelength range: 400nm – 700nm
  - 31 Bands, 10nm steps
  - Resolution: 512 x 512 pixel



www.cs.columbia.edu/CAVE/databases/multispectral/

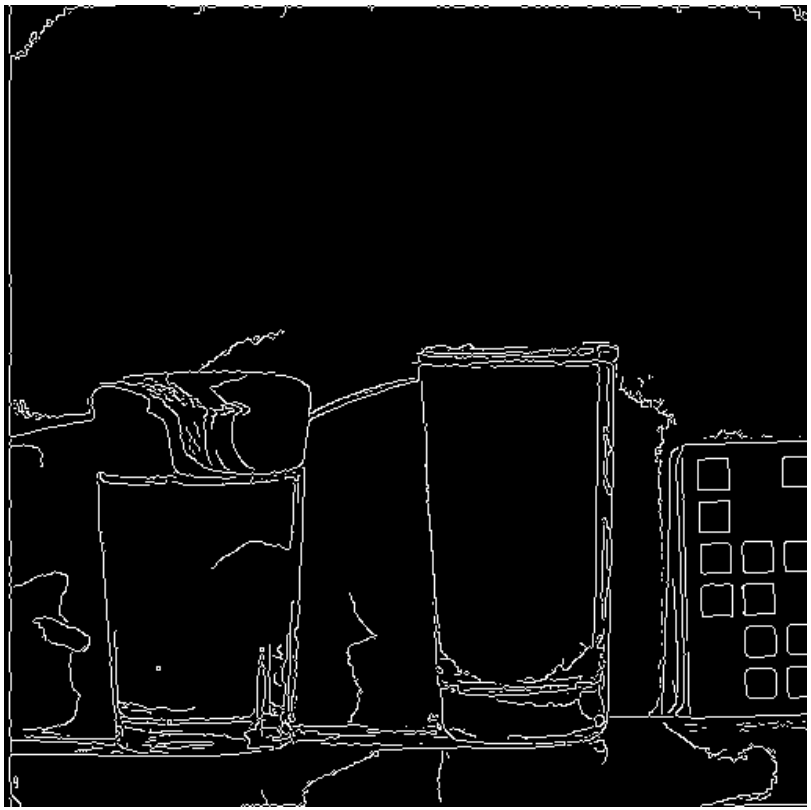
Multispectral Image: „fake & real beers“

Few colors, but specularities and translucent matarial

1D SOMs on „beers“

SOM 1x64

SOM 1x16



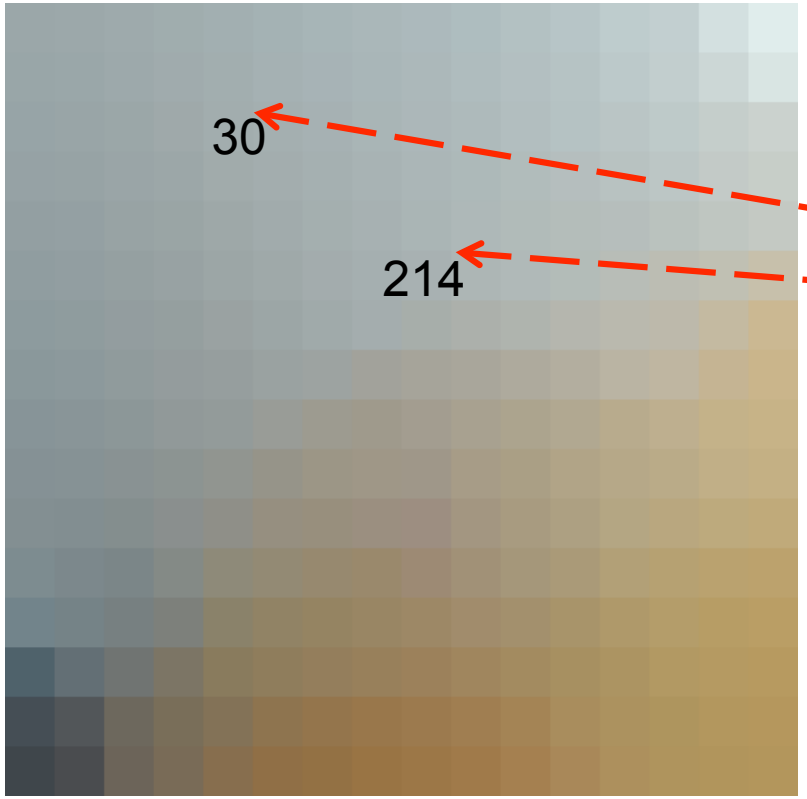
Canny thresholds: 15, 48

Canny thresholds: 35, 70

Rank images

2D SOM for Comparison

SOM 16x16

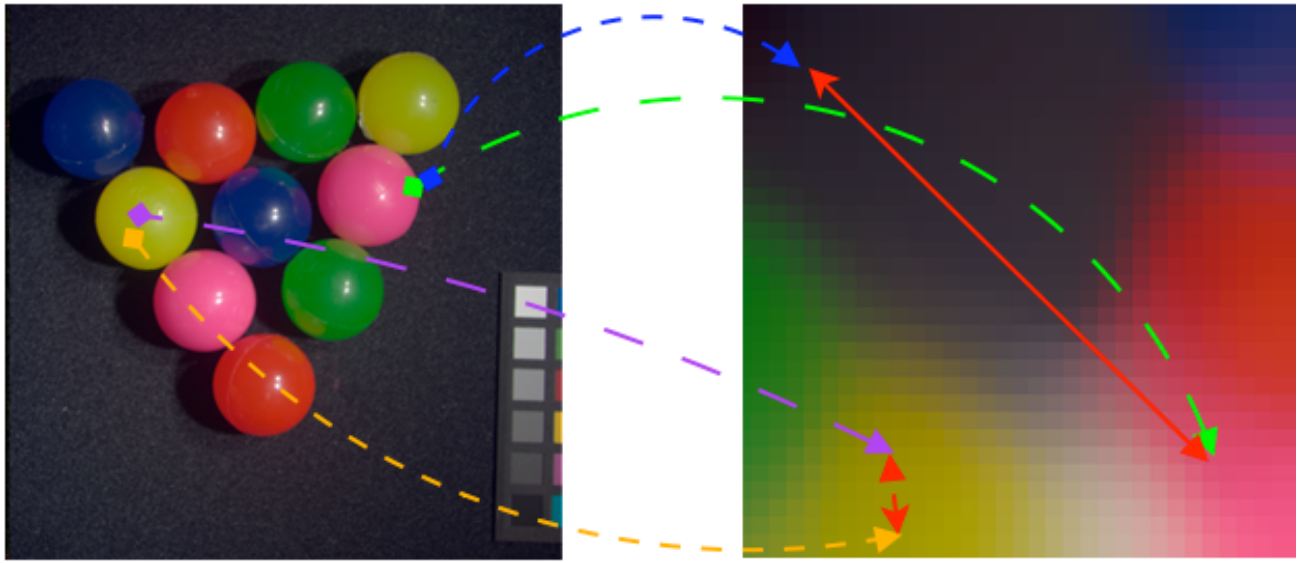


Edge image

Canny thresholds: 10,30 150

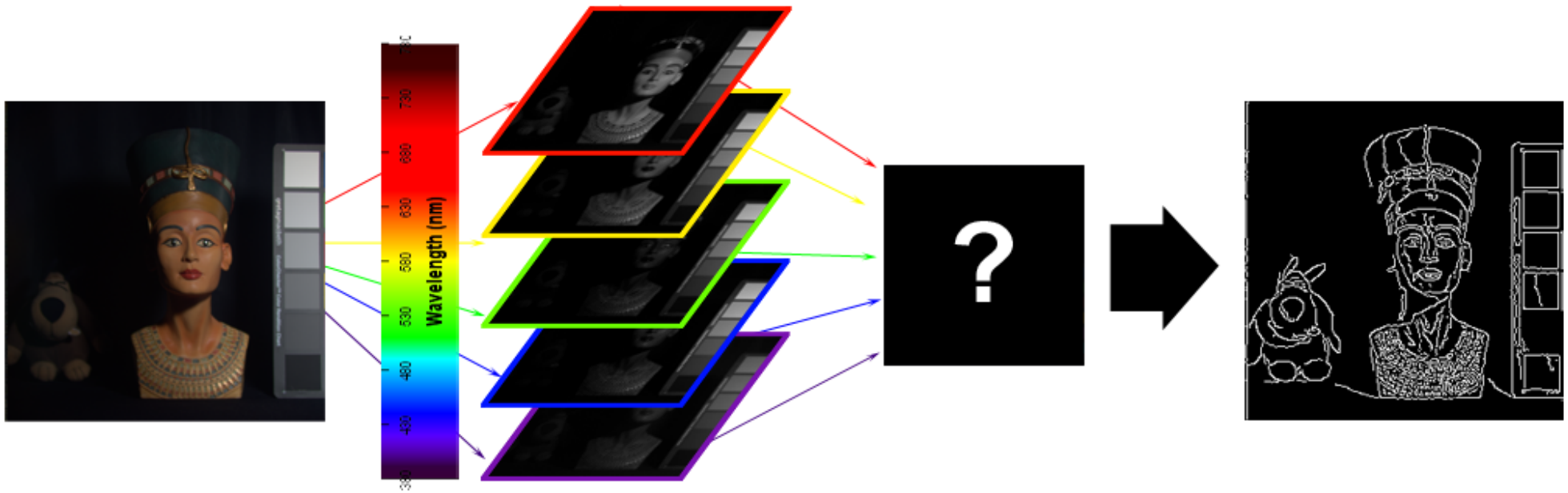
Linearization can cause wrong intensity differences!

Edge detection using SOM
Direct distance



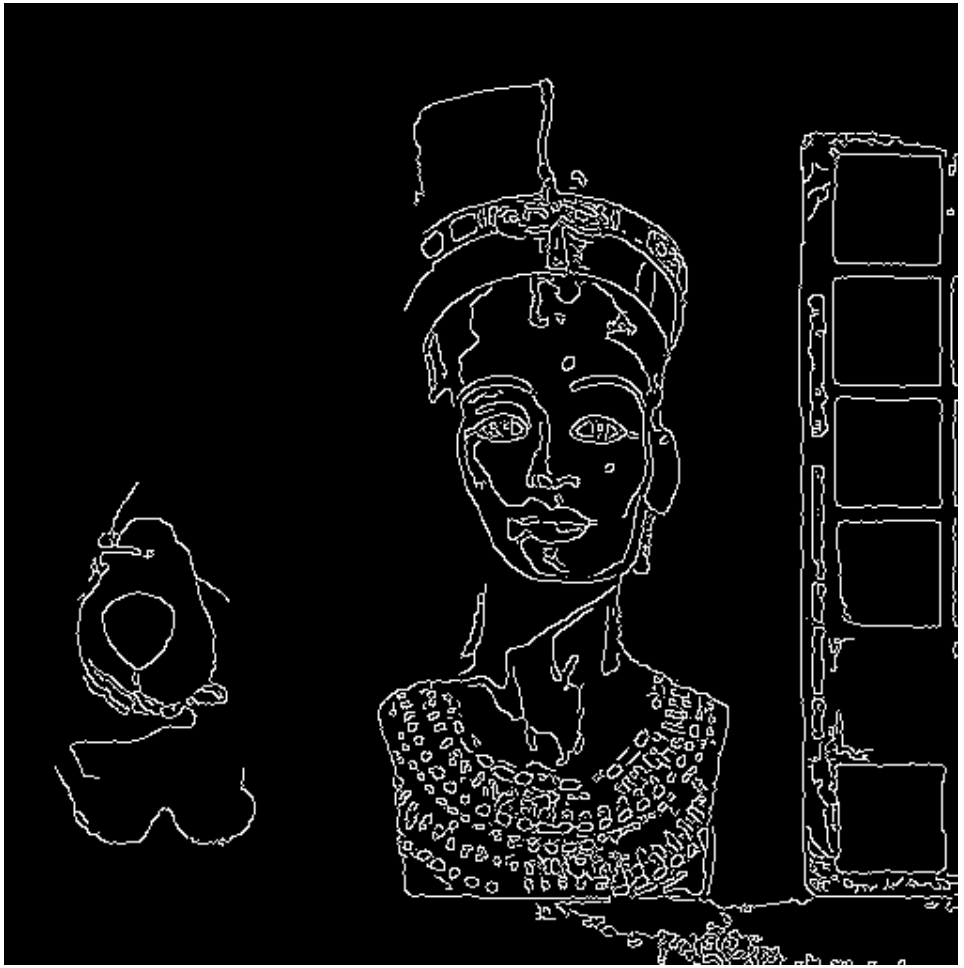
- Linearization introduces artificial artifacts => Skip linearization
- Calculate distances directly in SOM space to avoid linearization artifacts
- Map distances intensities. Run Canny on the “distance image”.

So How Did We Perform?



Original Idea: SOM Linearization

Input image



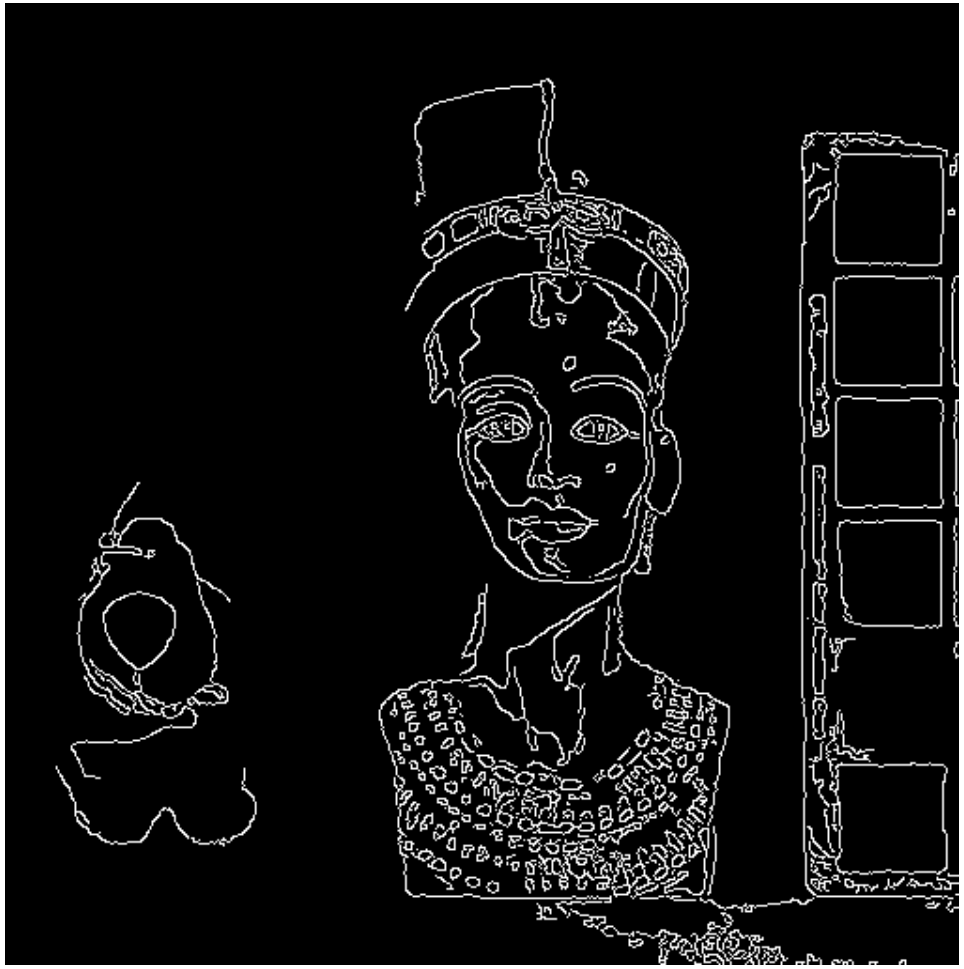
16x16 SOM, Canny on linearized SOM

New Idea: SOM Distances

Input image

16x16 SOM, Canny on SOM distances

Side by Side



16x16 SOM, Canny on linearized SOM

16x16 SOM, Canny on SOM distances

Side by Side, Higher Sensitivity Canny

16x16 SOM, Canny on linearized SOM

16x16 SOM, Canny on SOM distances

References

1. Some SOM slides are courtesy of Corbie Ziesman http://impact.asu.edu/cse591sp07/SOM.ppt
2. The multispectral imaging work presented here is work performed by Johannes Jordan, Felix Lugauer, Christoph Malskies and Ralph Muessig.