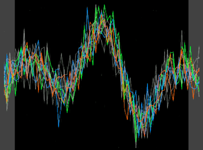
# Edge Detection in Multispectral Images Using the N-Dimensional Self-Organizing Map

September 14, 2011

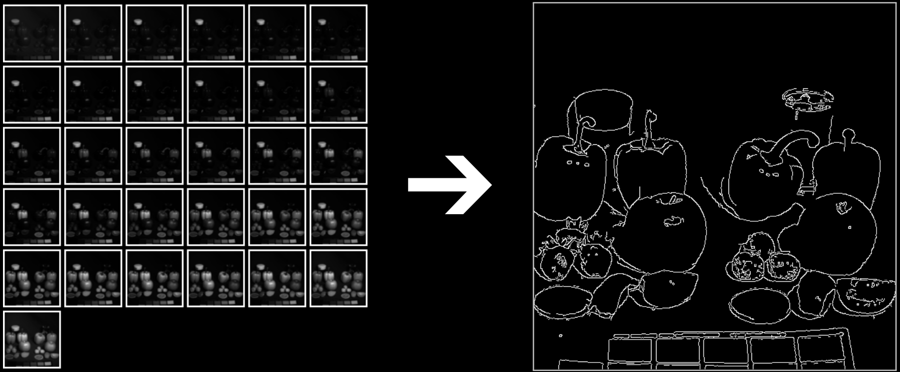


**Johannes Jordan**
Elli Angelopoulou

Pattern Recognition Lab (CS 5)
Friedrich-Alexander-University Erlangen-Nuremberg

# Motivation

# Motivation



# Motivation

**Problem:** Traditional edge detection works on monochromatic images.

- Multispectral images: vectors of 7 to 200 dimensions per pixel
- Reducing dimensionality not trivial
- No extensive research on edge-detection in this domain

# Motivation

**Problem:** Traditional edge detection works on monochromatic images.

- Multispectral images: vectors of 7 to 200 dimensions per pixel
- Reducing dimensionality not trivial
- No extensive research on edge-detection in this domain

**State of the art:** Toivanen et al., *Edge detection in multispectral images using the self-organizing map*, 2003

- Underlying idea: global pixel ordering

# Motivation

**Problem:** Traditional edge detection works on monochromatic images.

- Multispectral images: vectors of 7 to 200 dimensions per pixel
- Reducing dimensionality not trivial
- No extensive research on edge-detection in this domain

**State of the art:** Toivanen et al., *Edge detection in multispectral images using the self-organizing map*, 2003

- Underlying idea: global pixel ordering

**Our contribution:**

- Examine Toivanen's method on recent data
- A new method with improved performance

# Self Organizing Map

The *Self Organizing Map (SOM)* is an artificial neural network that reduces dimensionality.

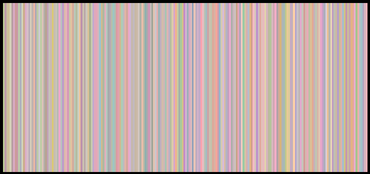
- 1D array or 2D grid of *model vectors (neurons)*
- Mapping: spectral vector $\rightarrow$ model vector ($L_2$ distance)
- Ordering provided by model vector coordinates

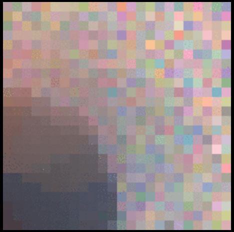
# Self Organizing Map

The *Self Organizing Map (SOM)* is an artificial neural network that reduces dimensionality.

- 1D array or 2D grid of *model vectors (neurons)*
- Mapping: spectral vector $\rightarrow$ model vector ($L_2$ distance)
- Ordering provided by model vector coordinates

**Training:**





# SOM for Edge Detection 1D

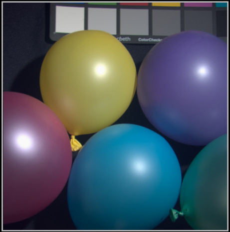
*Input image in sRGB*

- A 1D SOM is trained on the image
- Then, each image pixel is mapped to a neuron
- Neuron indices are interpreted as intensities
- Canny is applied on intensity map

*Trained SOM with 32 neurons in sRGB*

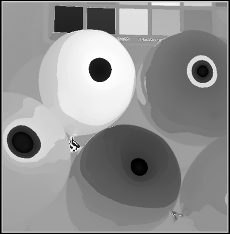
*Trained SOM with 256 neurons in sRGB*

# SOM for Edge Detection 1D



*Input image in sRGB*

- A 1D SOM is trained on the image
- Then, each image pixel is mapped to a neuron
- Neuron indices are interpreted as intensities
- Canny is applied on intensity map



*Intensity map*

*Canny output*

# Problems with this approach

*Input image in sRGB*

*Intensity map (64 neurons)*

# Problems with this approach

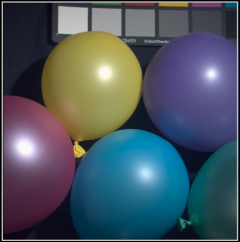
*Input image in sRGB*

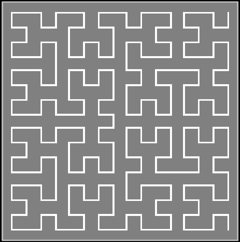
*Intensity map (64 neurons)*

*Trained SOM with 256 neurons in sRGB*

# SOM for Edge Detection 2D

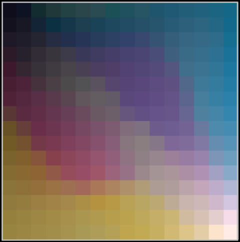
*Input image in sRGB*

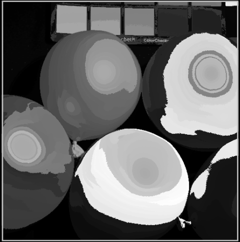




*Space filling curve (16x16)*

*Trained SOM in sRGB*

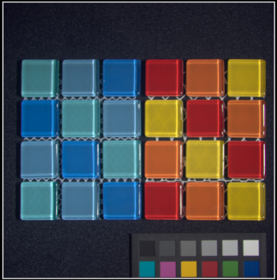




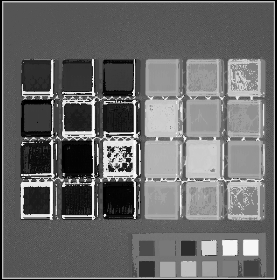
*Intensity map*

A 2D SOM is trained, then a 1D indexing is generated.

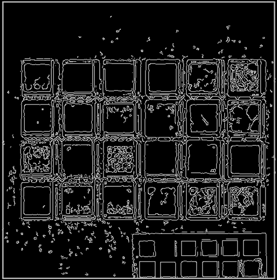
# Problems with this approach



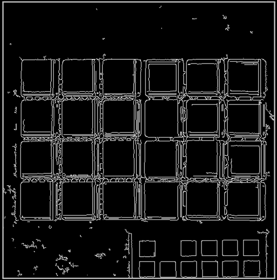
*Input image in sRGB*



*Intensity map (16x16 neurons)*



*Canny output, 2D SOM*

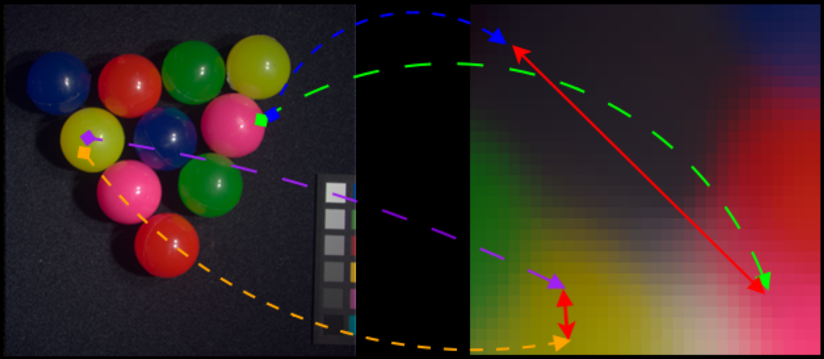


*Canny output, new method*

# New Method

**Observation:** For Canny to work, we only need differential data!

- We omit the creation of an intensity map, linearization
- Difference between pixels is defined by neuron distance ($L_2$)



# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

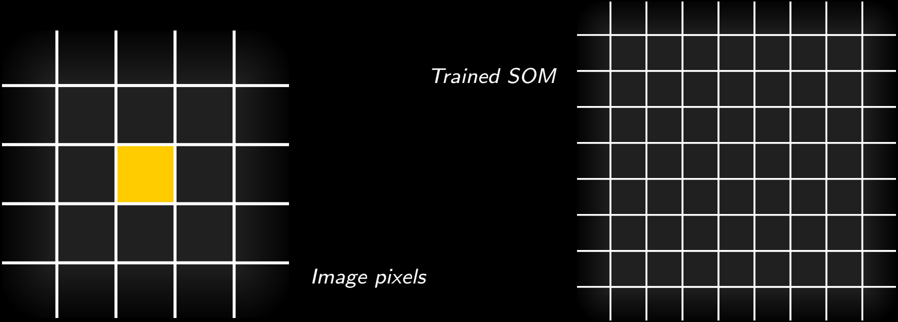
**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

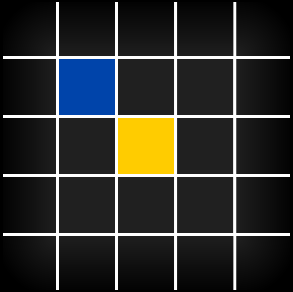


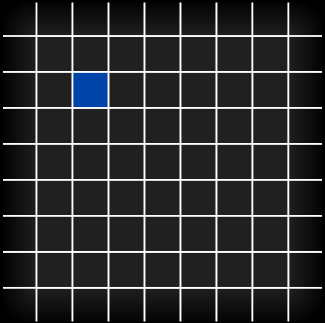
# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.



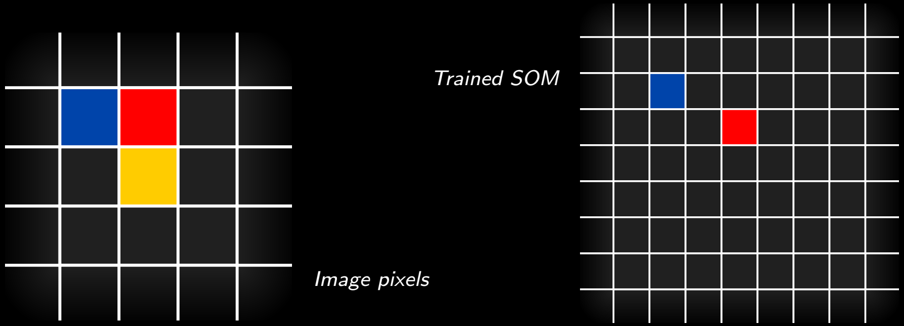



# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.



# Method details

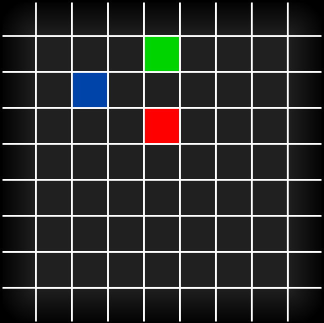
$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Problem:**
Canny expects input from Sobel operator

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

*Image pixels*

*Trained SOM*

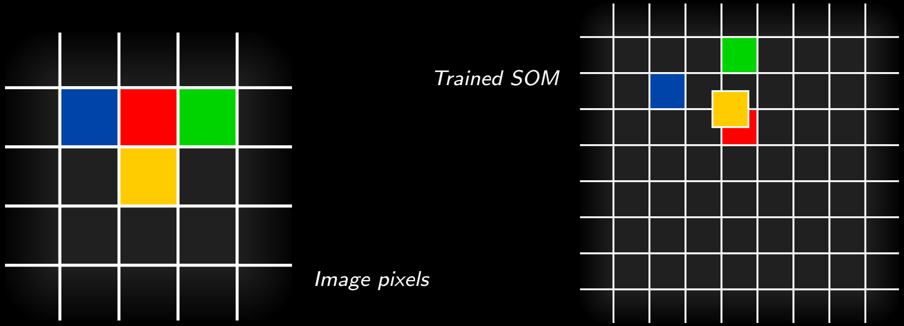


# Method details

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Problem:**
Canny expects input from Sobel operator

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

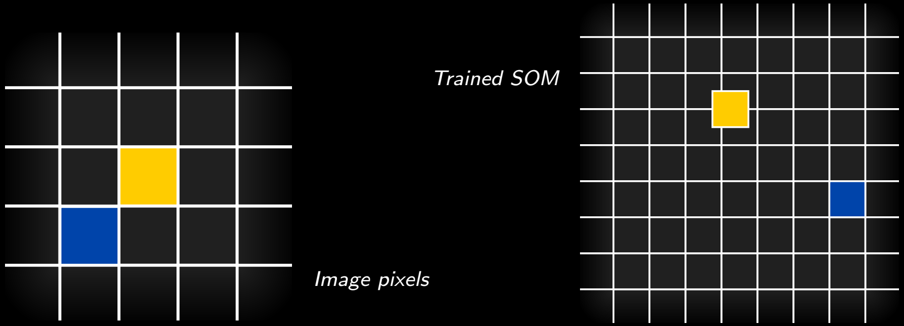


# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.



# Method details

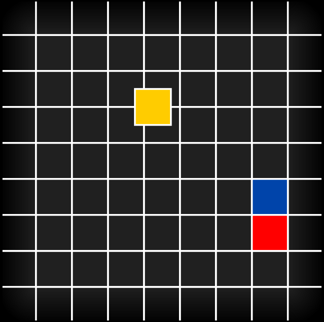
**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

*Image pixels*

*Trained SOM*

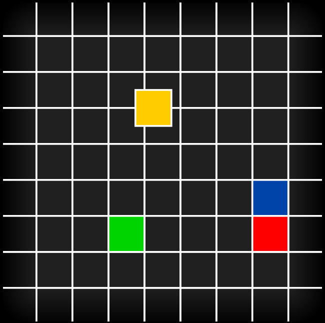


# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.

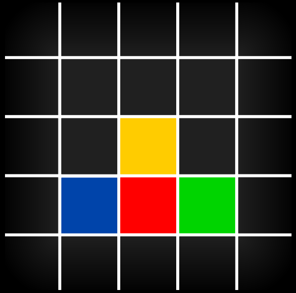



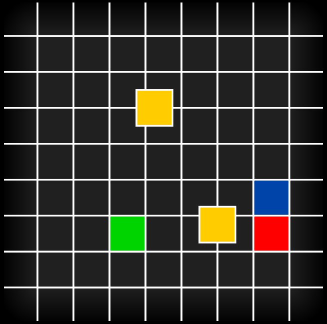
# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.



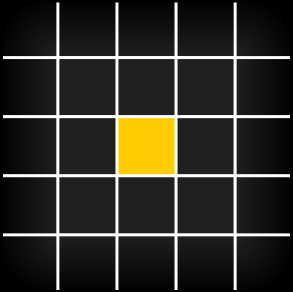



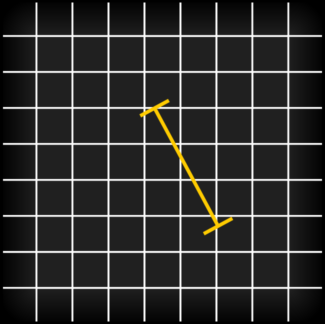
# Method details

**Problem:**
Canny expects input from Sobel operator

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

**Solution:** We compute weighted means from the neuron positions of adjoining pixels. From these we take the distance.



# Evaluation

Testing data:

- Images from the *CAVE Multispectral Image Database*
- Objects of different materials in a laboratory setting
- High quality, $512 \times 512$ pixels, $400\,$nm - $700\,$nm in $31$ bands

# Evaluation

Testing data:

- Images from the *CAVE Multispectral Image Database*
- Objects of different materials in a laboratory setting
- High quality, $512 \times 512$ pixels, $400\,\mathrm{nm}$ - $700\,\mathrm{nm}$ in $31$ bands

Canny parametrization:

- Thresholds selected per image and method
- Two goals: **Maximum edge preservation** vs. **Minimum noise**

# Results *Fake and Real Food*

*Input image in sRGB*

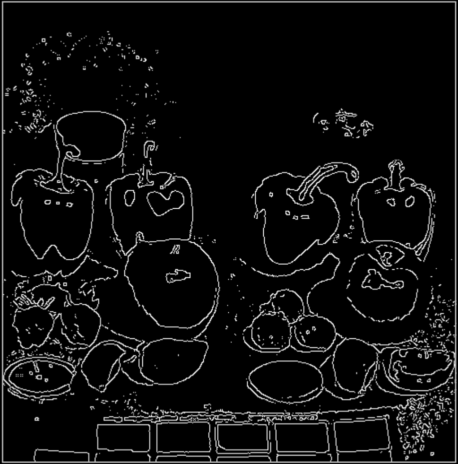
*Canny output, 1D-SOM (64 neurons)*

# Results *Fake and Real Food* (2)

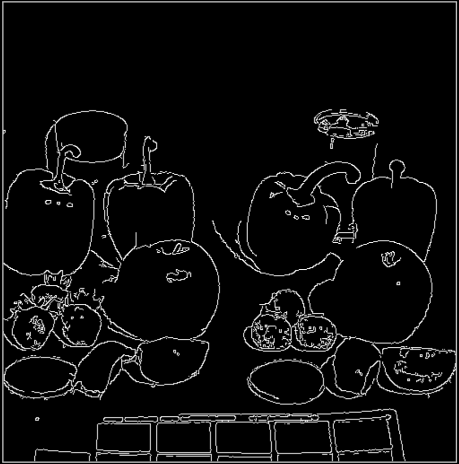
*Canny output, 2D-SOM (16x16 neurons)*

*Canny output, new method (16x16 neurons)*

# Results *Fake and Real Food* (3)



*Canny output, 2D-SOM (16x16 neurons)*



*Canny output, new method (16x16 neurons)*

# Results *Egyptian Statue*


*Input image in sRGB*

*Canny output, 1D-SOM (64 neurons)*

# Results *Egyptian Statue* (2)

*Canny output, 2D-SOM (16x16 neurons)*

*Canny output, new method (16x16 neurons)*

# Results *Egyptian Statue* (3)

*Canny output, 2D-SOM (16x16 neurons)*

*Canny output, new method (16x16 neurons)*

# Conclusions

We examined edge detection on multispectral images.

- Self Organizing Maps do provide usable results
- Previous methods suffer from pixel ordering artifacts

# Conclusions

We examined edge detection on multispectral images.

- Self Organizing Maps do provide usable results
- Previous methods suffer from pixel ordering artifacts

**Our method provides:**

- Better edge detection quality
- More reliable results
- Higher flexibility

# Thank you for your attention!

Source code will be released as free software as part of the *Gerbil* framework: http://gerbil.sf.net/