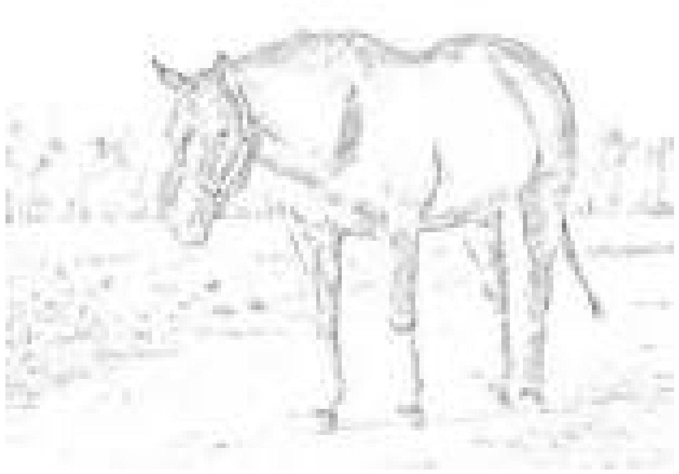
# *Edge Detection: Linear Filtering and Bayesian Inference*

A.L. Yuille (JHU)

Edge Detection and (simple) Semantic Segmentation
as examples of low-level vision.

## *Why do we care about edges?*

- A Line Drawing  is simple representation that contains a lot of information about images (far fewer bits than a normal image).
- We can make a Line Drawing of an image automatically by estimating edge map.  Sometimes this  sufficient to interpret an entire image.

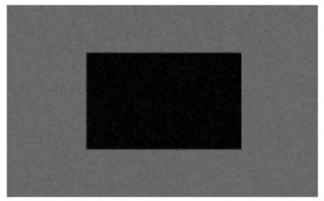


## *What are edges and how to find them?*

- Classic Edge detection: It is a local operation which filters an image and detects places where there is a local discontinuity. (Canny edge detector).

$$\begin{bmatrix} 120 & 118 & 110 & 115 & 116 & 120 \\ 115 & 21 & 20 & 16 & 19 & 121 \\ 112 & 19 & 17 & 18 & 20 & 117 \\ 119 & 118 & 121 & 117 & 116 & 112 \end{bmatrix}$$

(a) Intensity matrix of an image



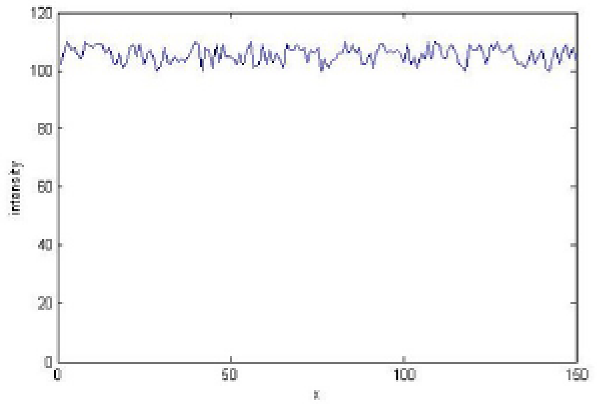
(b) Image of the matrix (150×100)

Images (right) correspond to numbers (left).

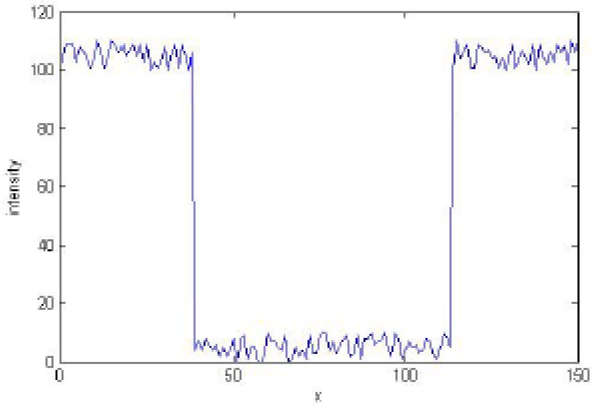
- But the purpose of edge detection is to detect the boundaries or objects or significant texture changes (like color changes on a shirt).
- Boundaries and Texture changes often occur at places where there are discontinuities, but this is not sufficient (see later).

## *Edges often occur at places where the intensity gradient changes.*

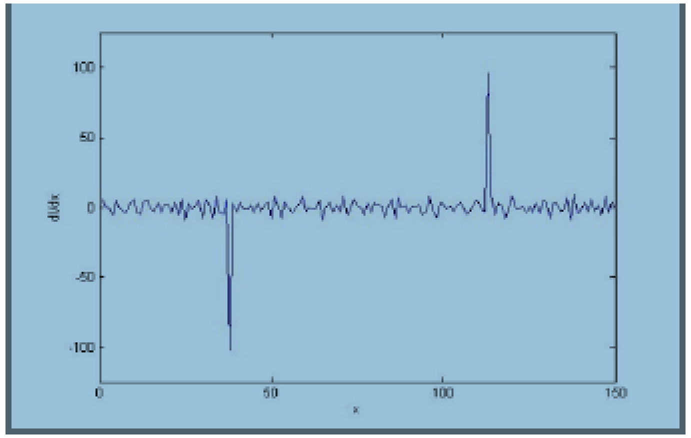
- To find discontinuities take the derivative of Image dI(.x)/dx in one dimension or the image gradient in two dimensions.  The detect edges by edges by thresholding.
- This reduces edge detection to thresholding derivatives.



(a) Intensity values (y=5)



(b) Intensity values (y=50)



(c) Derivative Filter response (y=50)

# *Images: continuous functions or discrete values.*

- Images can be thought of a depth maps – the intensity is the depth.
- Image intensities are *continuous* valued, but in practice we quantize them to obtain *discrete* values in the range [0,255] defined over a discrete image lattice.
- Derivatives of continuous functions are replaced by differences of discrete functions.

**Images as functions**

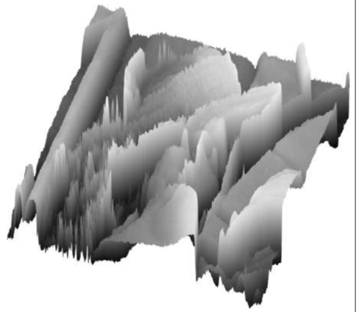
Continuous

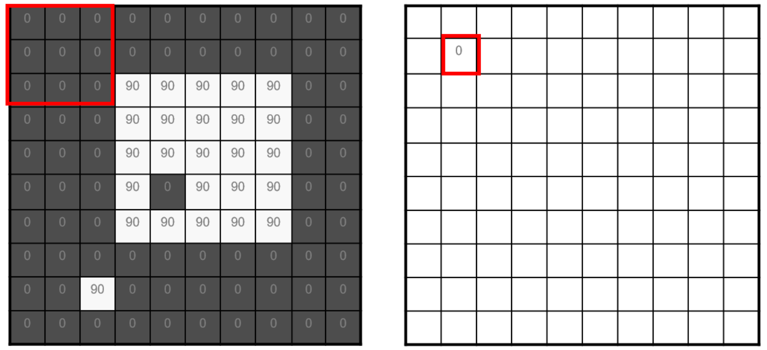
$$f : R^2 \to R^d$$
$$x = (h, v)$$

Discrete

$$f : Z^2 \to R^d$$
$$n = (n_1, n_2)$$

d=1: Gray
d=3: Color



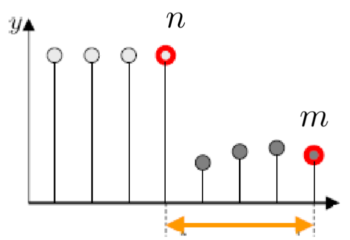


|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

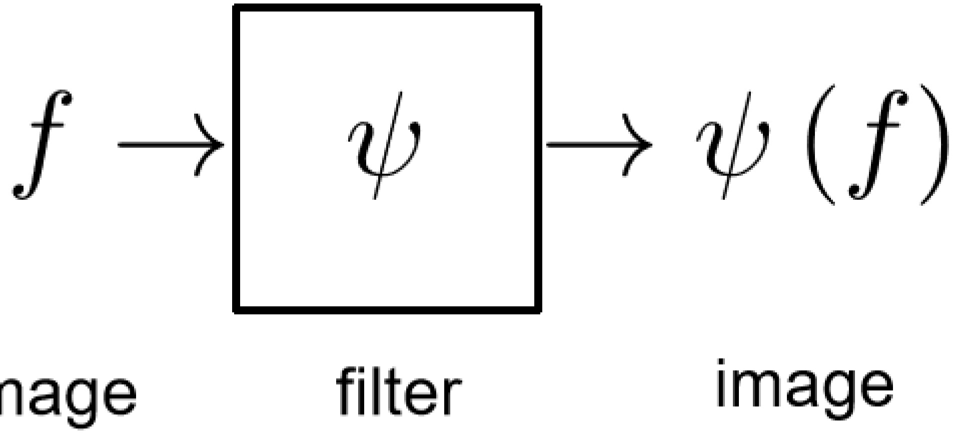
# *Edges and Linear Filters*

- We detect edges by filtering with a derivative filter.
- Convolve the image by the filter (moving average).
- A derivative filter will have a big response at the edges.



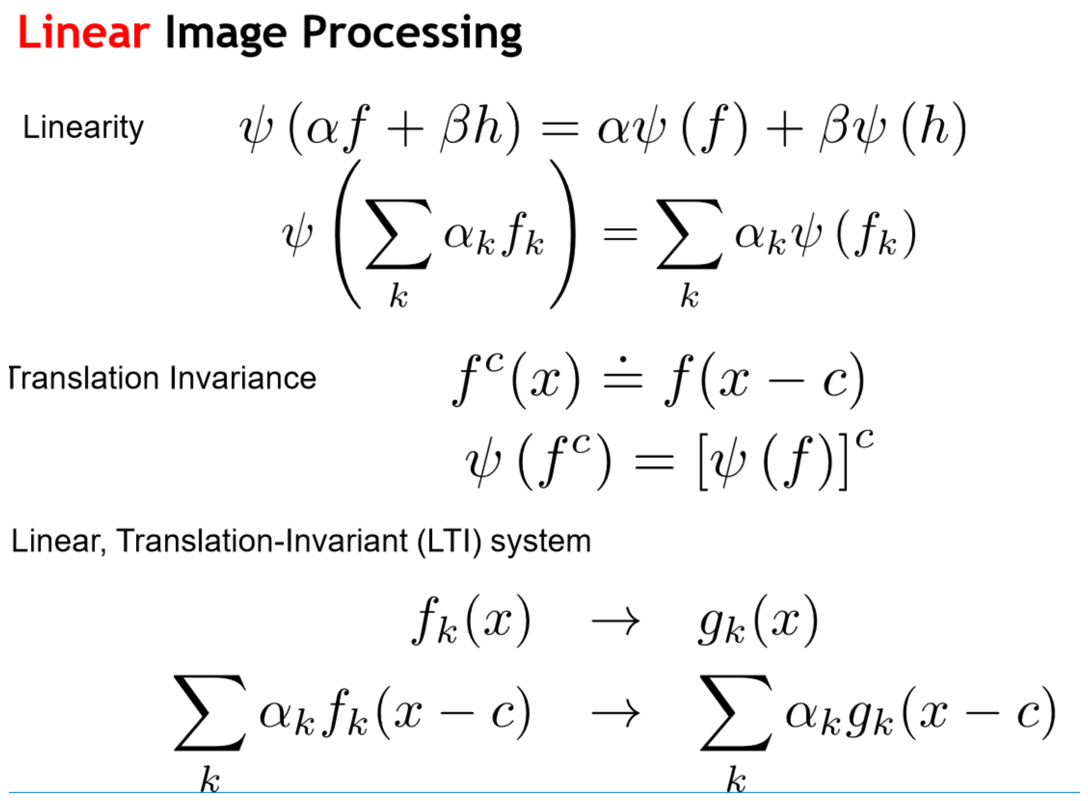
$$g[n] = \sum_m w(|n-m|) f(m)$$

$$w[k] = \begin{cases} \frac{1}{2l+1}, & k \le l \\ 0, & \text{otherwise} \end{cases}$$



# *Properties of Linear Filters (vast literature)*

- Superposition. Translation (Convolution).



**Linear Image Processing**

Linearity

$$\psi(\alpha f + \beta h) = \alpha\psi(f) + \beta\psi(h)$$

$$\psi\left(\sum_k \alpha_k f_k\right) = \sum_k \alpha_k \psi(f_k)$$

ranslation Invariance

$$f^c(x) \doteq f(x-c)$$

$$\psi(f^c) = [\psi(f)]^c$$

Linear, Translation-Invariant (LTI) system

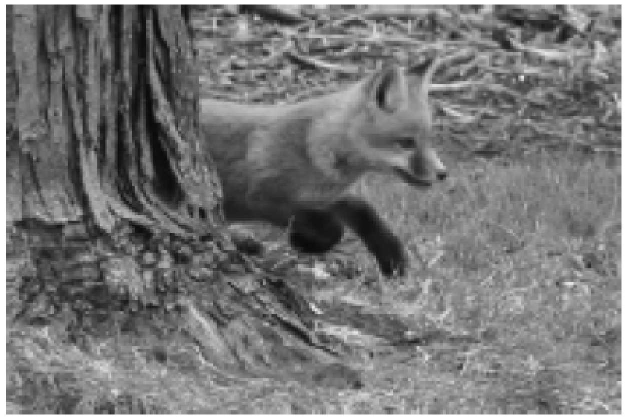
$$\begin{array}{ccc} f_k(x) & \rightarrow & g_k(x) \\ \sum_k \alpha_k f_k(x-c) & \rightarrow & \sum_k \alpha_k g_k(x-c) \end{array}$$

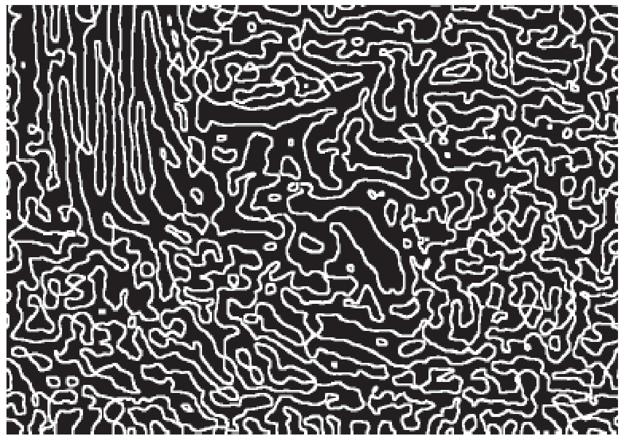
## *Filterbanks*

- To detect edges in images (which are two-dimensional) there are several strategies.
- 1. Make a filterbank of filters which take the derivatives of images in different dimensions and specify a way to combine their responses. This is the most common strategy.
- 2. Detect the image gradient and take the image derivative in this direction.
- 3. Compute the magnitude of the image gradient and threshold it.

## *Detecting edges by differentiation has problems.*

- 1. The image gradient can be very small at the boundaries of an object.
- 2. There are places in the image where the gradient is high but do not correspond to object boundaries or texture changes.
- Fox Image: Thresholding the gradient gives poor results. There are edge in the images at different scales –objects, texture (fur, bark, grass).(Caveat).





# *Images have ``structure” at different scales.*

- How to deal with scale?
- The easiest method is to blur the image by linear filter with Gaussians. This removes the high frequencies,

- Nonlinear filtering is better,
- Images have structures at different scales.
- Illusions of Lincoln, Marilyn, and Shakespeare.

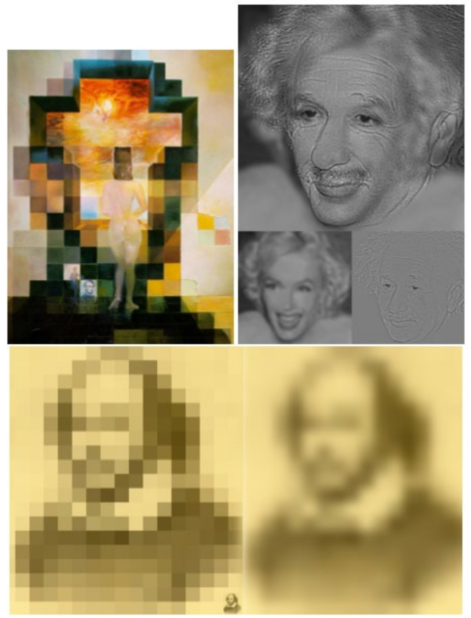
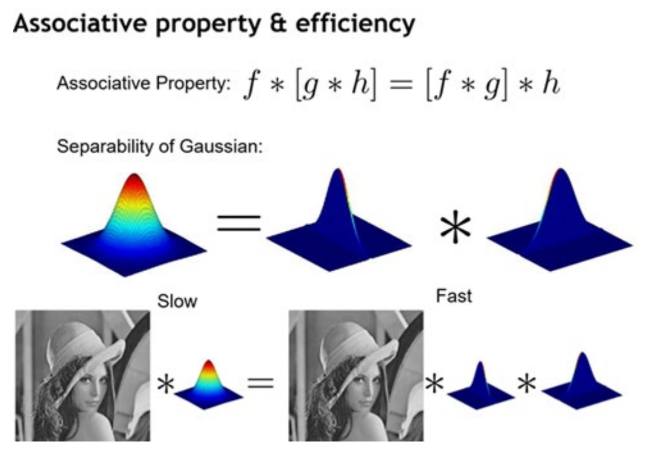
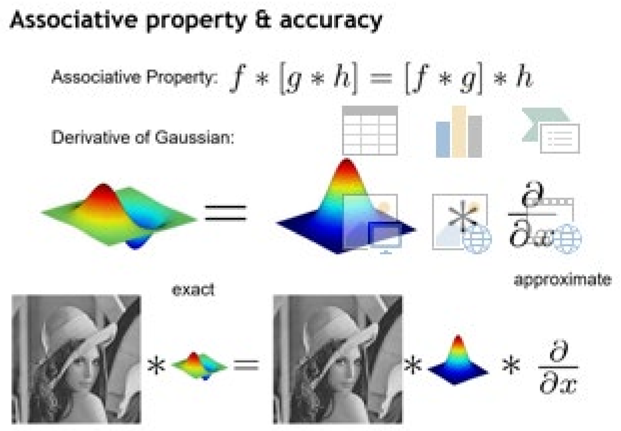


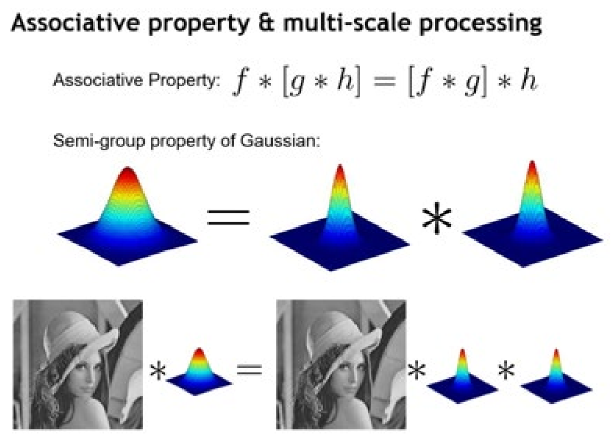
Figure 15: Left Panel: If you blur the image (by squinting) you see Lincoln. But without blurring you see a picture by Dali. illusion. Center Panel: top is a superposition of Monroe (low frequencies, bottom left) and Einstein (high frequencies, bottom right). Shakespeare block filtered (i.e. and image restored by blurring.

## *Smooth and Differentiate*

- We can smooth the image by convolving it with Gaussian filters. This diffuses the image (hence Diffusion Models) by increasing the variance of the Gaussian filter.
- We can smooth and differentiate images by a single linear filter.


Associative property & efficiency
Associative Property: $f * [g * h] = [f * g] * h$
Separability of Gaussian:
Slow
Fast


Associative property & accuracy
Associative Property: $f * [g * h] = [f * g] * h$
Derivative of Gaussian:
$\frac{\partial}{\partial x}$
exact
approximate

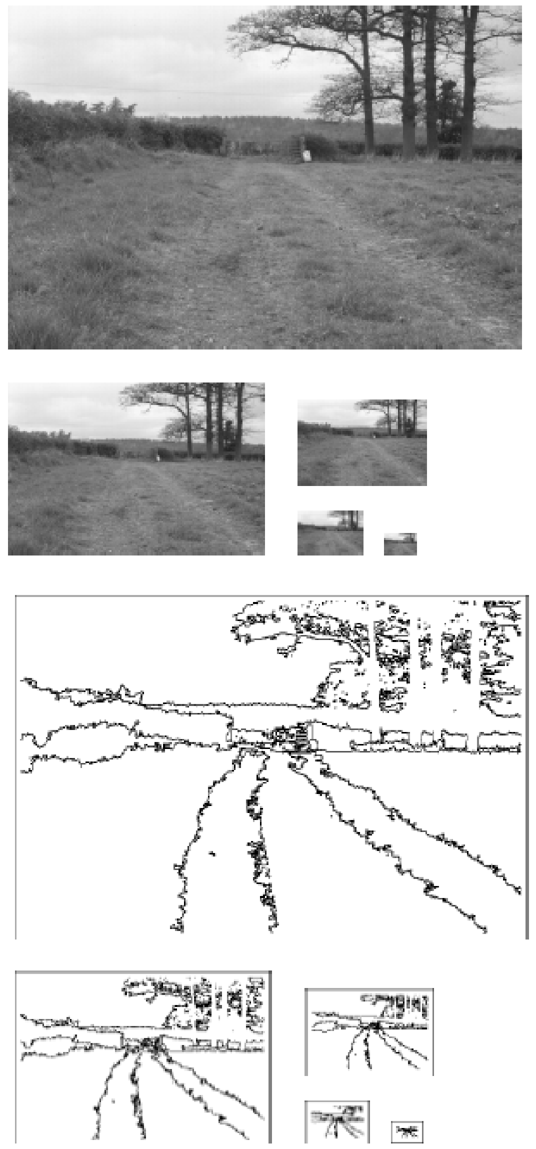

Associative property & multi-scale processing
Associative Property: $f * [g * h] = [f * g] * h$
Semi-group property of Gaussian:

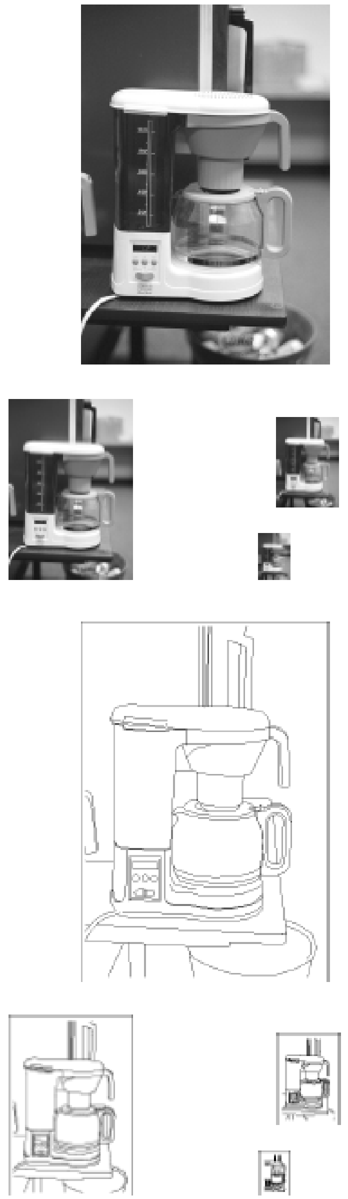
## *Edge Detection and Learning*

- Classic approach – define an ideal model of an edge and obtain an optimal edge detector (Canny 1986).
- This reduces to filtering and thresholding images.
- In the late 1990's researchers created datasets of images with the positions of the edges annotated. The first two datasets were called Sowerby and South Florida.
- The classic approach worked well on clean images with simple background (South Florida) but failed very badly on image with complex background (Sowerby).

# *Two Datasets.*

- The datasets have image (top panels) annotated by edges (lower panels). Images consist of edges and background (non-edges).
- Sowerby (right) is of outdoor images with much texture and vegetation in the background.
- South Florida (far right) is  indoor images with very simple backgrounds.
- Edge detection requires detecting the targets (annotated edges) and distinguishing them from background.





# *Sowerby Dataset: Groundtruth.*

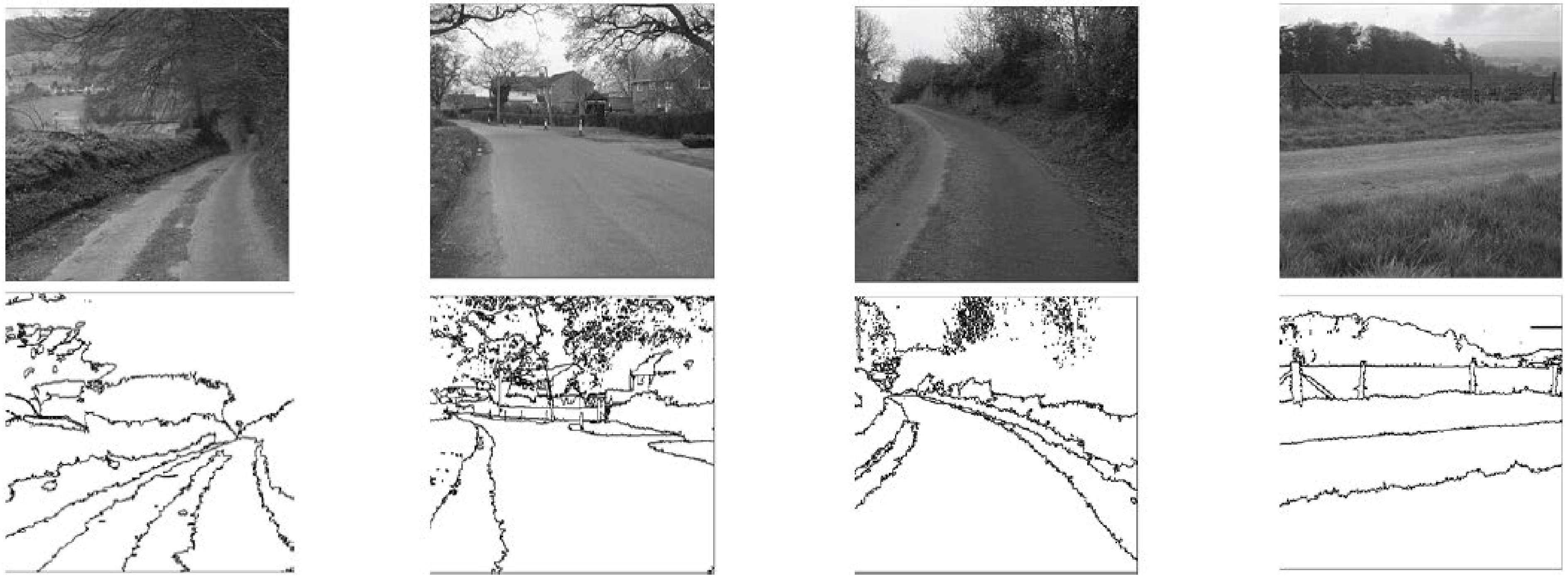


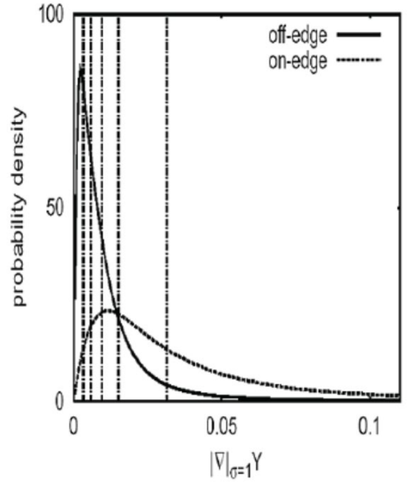
Figure 5. Upper Panels: the data images. Lower Panels: the groundtruth edge maps

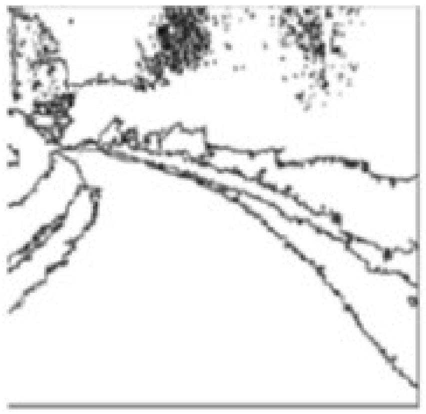
## *The Value of Benchmarked Datasets & Learning*

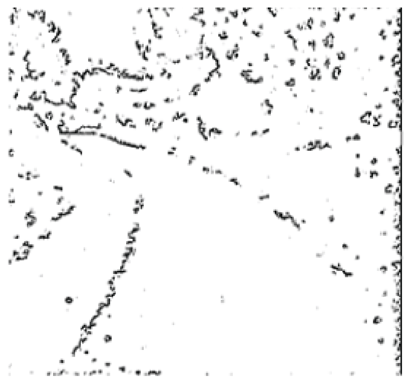
- Benchmarks give a way to evaluate edge detectors.
- They enables us to learn how to combine different cues for edges. Canny knew that he should combine edge cues from different scales but it was difficult to do this without annotated data for guidance.
- They enable us to detect edges where there may not be local discontinuities.
- Caveat: there are issues with annotated datasets. It is not easy to decide what edges to annotate in an image. Arguably this leads to competitions which rely on how well algorithms can fit the noise of the annotators.
- See X. Hou et al. 2014.

# *Edge Detection by Bayesian Inferences*

- Learn conditional probability distributions of image features conditioned on whether there is an edge or not (on-edge, off-edge).
- 1. Probability distributions. 2. Image. 3. Ground Truth. 4. Output.
- This was the first work to perform edge detection statistically (S. Konishi PhD thesis. Konishi et al. 1999).







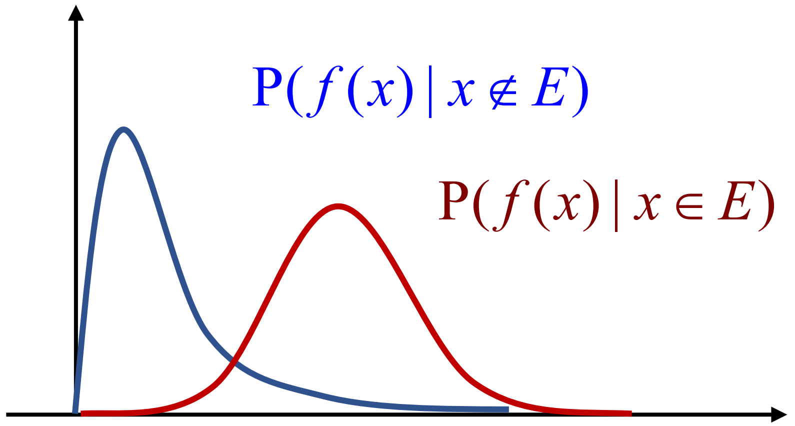
## A Bayesian strategy to learn edge detection.

(Konishi et al., 1999, 2003)

**Learn distributions**

$$\begin{cases} \mathrm{P}(f(x) \mid x \in E) \\ \mathrm{P}(f(x) \mid x \notin E) \end{cases}$$

$E = \{x : e(x) = 1\}$ : set of edges



Example (1-dimension): $f(x) = \left|\frac{dI}{dx}\right|$

$\left|\frac{dI}{dx}\right|$ is typically larger on edges than on non-edges

But, sometimes $\left|\frac{dI}{dx}\right|$ is small on edges and big on non-edges

**How to represent $\mathrm{P}$?** Parametric Distribution? (e.g. Gaussian – requires assumptions)
or non-parametric? (e.g. histogram – requires more data)

**Learn** $P\big(f(x)\,|\,x \in E\big), P\big(f(x)\,|\,x \notin E\big)$

Note: Memorization and Generalization

- Learn these distributions using only part of the dataset – training set.
- Evaluate/test on other parts of the dataset – testing set.

To ensure generalization and prevent over-fitting the training data.

Decision Rule (classification function):

$$\alpha(x) = 1\,(\text{edge}) \qquad \text{if } \log \frac{P\big(f(x)|x\in E\big)}{P\big(f(x)|x\notin E\big)} \geq T$$

$T$: threshold

$$0\,(\text{not edge}) \text{ if } \log \frac{P\big(f(x)|x\in E\big)}{P\big(f(x)|x\notin E\big)} < T$$

## Changing the threshold affects:

i. The false positives – the number of image pixels wrongly classified to be edges

ii. The false negatives – number of image pixels which really are edges, but are classified as non-edges

## What threshold to use? ➔ Depends on task

- Impossible to find a threshold which gives perfect results (i.e., has no false positive and no false negatives). Recall that the purpose of edge detection is to make hypotheses for the boundaries of object.
- Best not to make hard decisions too early. False negatives worse than false positives.
  - Use context and higher level information to decide later

## Comparisons Canny Edge detector vrs: $\log \frac{P(f(x) \mid x \in E)}{P(f(x) \mid x \notin E)}$

Performance depends on filters used

If $f(x) = |\nabla I(x)|$ , then performance is similar to Canny (see later slides).

But for more sophisticated filters, then $\log \frac{P(f(x) \mid x \in E)}{P(f(x) \mid x \notin E)}$ is much better

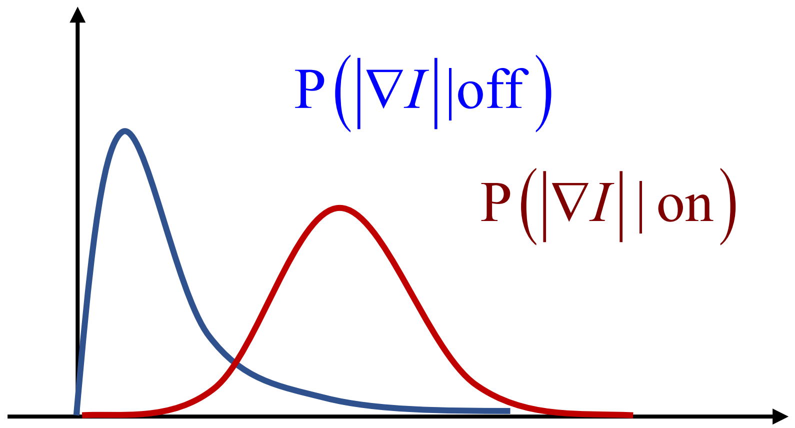
E.g., $f(x) = (|\nabla I(x)|, |\nabla G * I(x)|)$, where $G * I$ is a smoothed image.

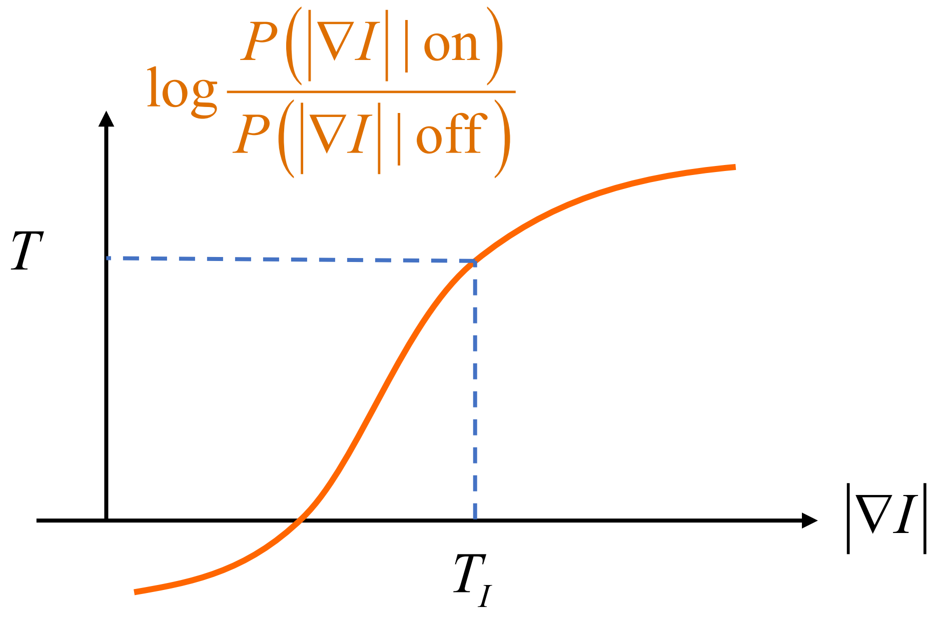
➔i.e., combine information at different scales of the image.

We need data to learn, but this gives us a principled way to combine information from multiple-scales or other sources.

**Note:** If you only use $|\nabla I|$ filter then results are disappointing

$\log \frac{P(|\nabla I| \mid \text{on})}{P(|\nabla I| \mid \text{off})}$ is monotonic in $|\nabla I|$





➔ Thresholding $\log \frac{P(|\nabla I| \mid \text{on})}{P(|\nabla I| \mid \text{off})}$ reduces to thresholding $|\nabla I|$

(so we rediscover the Sobel edge detector)

## *Results on Benchmarks*

- This Bayesian approach gives results which are slightly better than conventional methods (Canny and others) when evaluated on the South Florida dataset.
- But the results are orders of magnitude better than Canny when evaluated on the Sowerby datasets. This dataset requires exploiting edge cues at different scales. Small texture edges will be detected by derivative filters which have little smoothing, but not by derivative filters with large smoothing. Hence the Bayesian approach can learn to reject then.

## Datasets: Problems and Perils

Dataset Bias: Is the data representative of the set of natural images?

Experimental Design

EG South Florida dataset –contains images with little texture

→ edge detection is easy (Canny performs similar to log-likelihood)

Sowerby dataset – contains images with a lot of texture (e.g. vegetation)

→ edge detection is hard

An edge detector trained on South Florida will perform badly on Sowerby

An edge detector trained on Sowerby will perform fairly well on South Florida

➔Domain transfer – see notes.

## Labeling bias → where does ground-truth come from?

In Berkeley BDS datasets, 5~6 students label each image independently

But, not all labelers agree

- "Strong Edges" are labeled by all 5 students
- "Weak Edges" are labeled by 1 student only

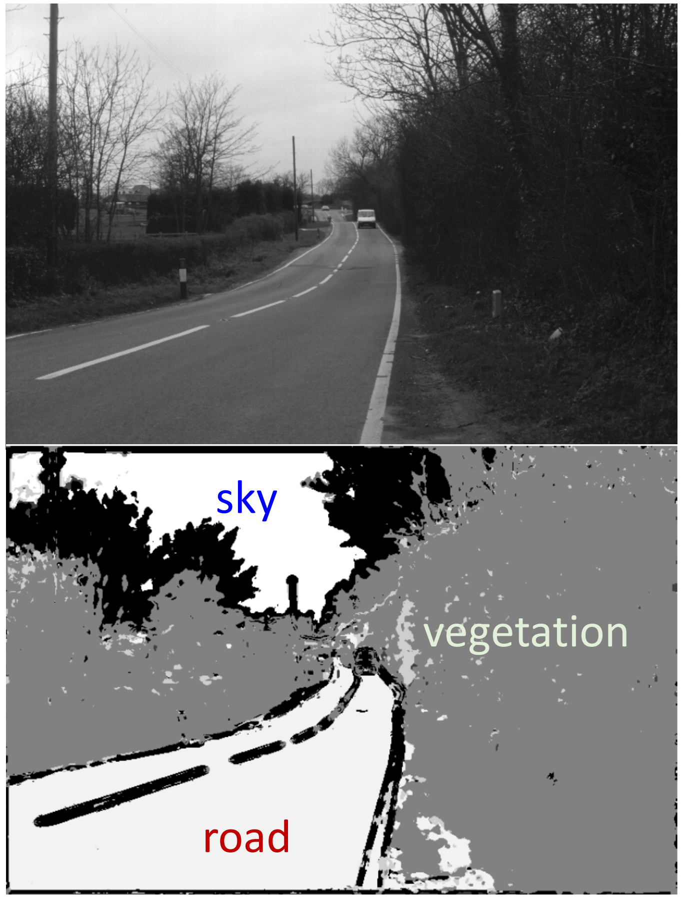
Psychophysics experiments (X. Hou et al., CVPR 2014) show that weak edges are poorly defined

→ Problematic to use weak edges for training and testing

## Summary

- Current edge detectors are based on deep networks trained on large datasets (examples later in the course).
- Bayesian approach still has value because it helps address the difficulty of domain transfer.

## Extend to other vision tasks / What else can we do locally?



(Konishi & Yuille, 1999)

EG Classify an image pixel as {sky, vegetation, road, other}

Dataset: Sowerby

Strategy: same as before

Learn $\mathrm{P}\left(f(x) \mid s\right)$

$s$ is a label → e.g., sky, vegetation, etc.

$f(x)$: filters

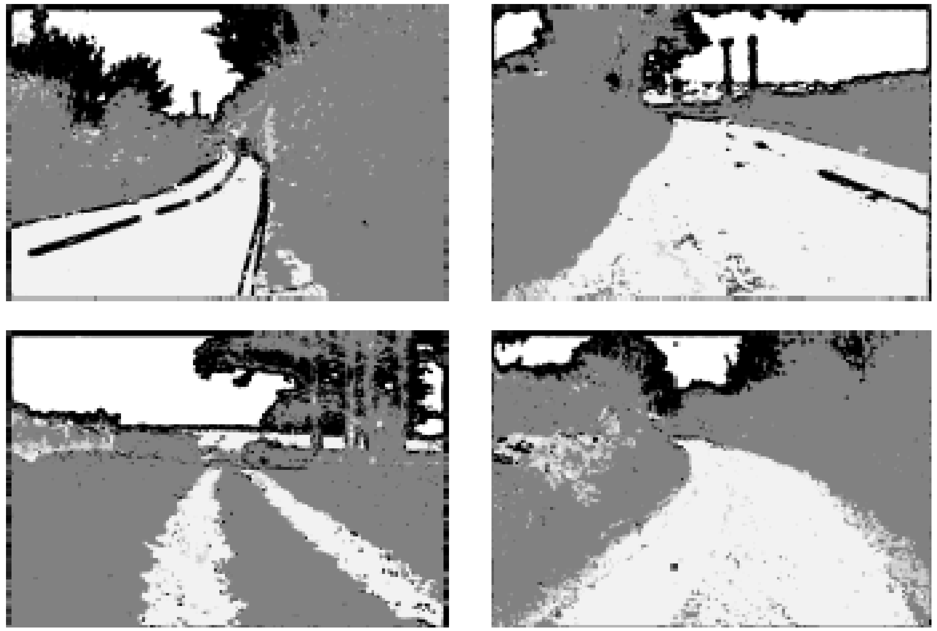
Classify $x$ by $\hat{s} = \arg\max P\left(f(x) \mid s\right)$

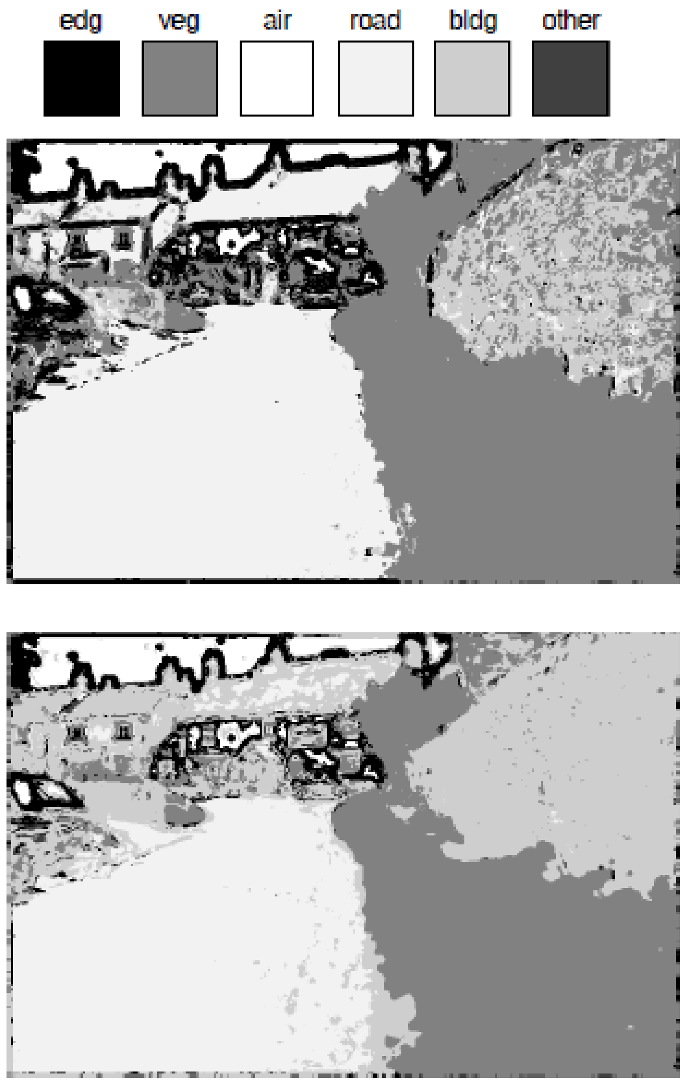
## *Semantic Segmentation:*

- This was one of the first papers on semantic segmentation. The term "semantic segmentation" did not exist at that time so we gave the task a more complicated title (which didn't catch on).
- Our approach was only able to classify regions which were roughly homogeneous like "sky", "vegetation", "water".
- The features exploited simple cues like texture and color.
- At that time it was amazing how well the algorithms worked. Alternative strategies for segmenting image into regions, without semantic knowledge and training data, were much less successful.

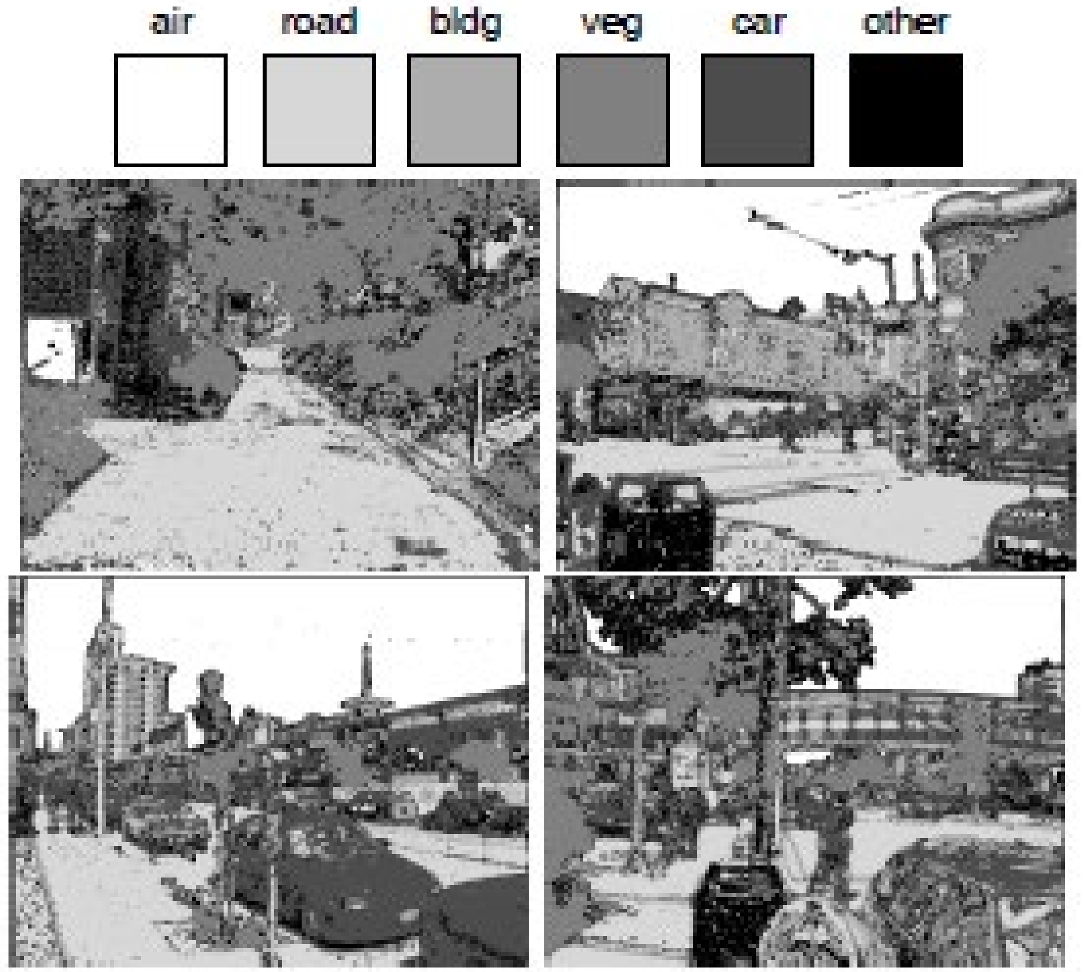
# *Semantic Segmentation on Sowerby*

- Konishi and Yuille. CVPR. 2018.
- Output examples.





# *Semantic Segmentation on San Francisco*


air
road
bldg
veg
car
other