Course: Vision as Bayesian Inference. Lecture 1

Alan Yuille
Department of Statistics, UCLA
Los Angeles, CA 90095
yuille@stat.ucla.edu

Abstract

Vision is Difficult. Probabilistic Approach. Edge Detection Example. Factorized Models.
NOTE: NOT FOR DISTRIBUTION!!

1 Introduction

What is vision? Image I is generated by the state of the world W. You need to estimate W from I.

Vision is very very hard. Humans are "vision machines". The large size of our cortex distinguishes us from other animals (except close relatives like monkeys). Roughly half the cortex is involved in vision (maybe sixty percent for monkeys). It is possible that the size of the cortex developed because of the evolutionary need to see and interpret the world. So intelligence might be a parasite of vision. Most animals have poor vision – e.g., cats and cobras can only see movement – and often rely on other senses (e.g., smell). Animals may also only use vision, and other senses, for basic tasks such as detecting predators/prey and for navigation. The ability to interpret the entire visual scene may be unique to humans and seem to develop comparatively late (claims that 18 year old adults are still learning scene perception).

Why is vision difficult? *The world W is complex and ambiguous*, see figure (1) – the world consists of many objects (20,000 based on estimates made by counting the names of objects in dictionaries) and plenty of "stuff" (texture, vegetation, etc.). *The mapping from the world W to the image I is very complex* – the image is generated by light rays bouncing off objects and reaching the eye/camera. Images are like an *encoding of the world*, but it is an encoding that is not designed for communication (unlike speech, or morse code, or telephones), see figure (2). Decoding an image I to determine the state of the world W that caused it is extremely difficult. The difficulty was first appreciated when AI workers started trying to design computer vision systems (originally thinking it would only take a summer).

Another way to understand the complexity of images is by looking at how many images there can be. A typical images is $1,024 \times 1,024$ pixels. Each pixel can take values $1 - 255$. This gives a number of images to be $(1,024 \times 1,024)^{256}$ which is enormous (much much bigger than the number of atoms in the Universe). If you only consider 10×10 images, you find there are more of them than can have been seen by all humans over evolutionary history (allowing 40 year for life, 30 images a second, and other plausible assumptions). So the space of images is really enormous.

If vision is so hard then how is it possible to see? Several people (Gibson, Marr) have proposed "ecological constraints" and "natural constraints" which mean that the nature of the world provides constraints which reduce the ambiguities of images (e.g., most surfaces are smooth, most objects move rigidly). More recently, due to the growing availability of datasets (some with groundtruth) it has become possible to determine statistical regularities and statistical constraints (as will be described in this course). In short, there must be a lot of structure and regularity in images I, the world W, and their relationship which can be exploited in order to make vision possible.

But how can we learn these structures/regularities? How can infants learn them and develop the sophisticated human visual system? How can researchers learn them and build working computer vision systems? It seems incredibly difficult, if not impossible, to learn a full vision system in one go. Instead it seems that the only hope is proceed

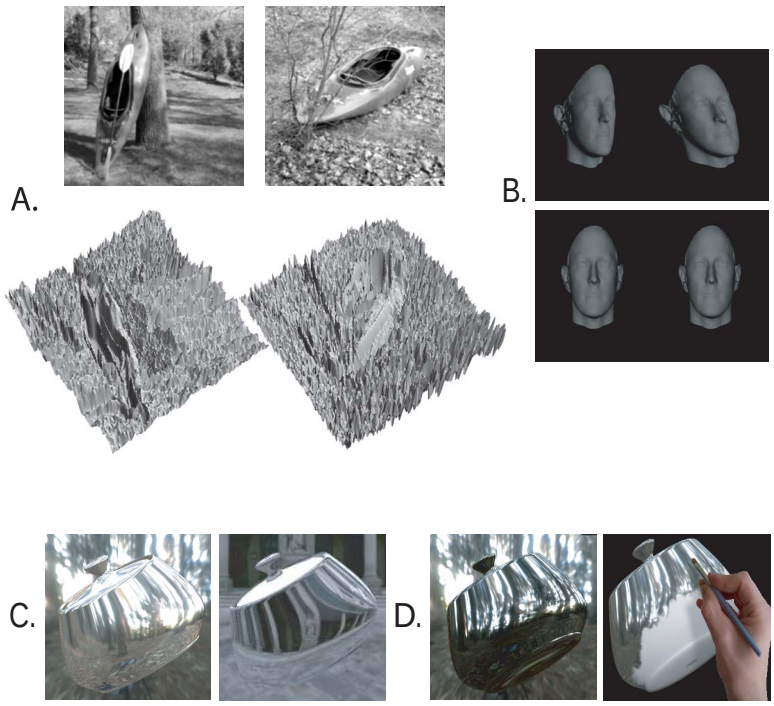

Figure 1: The Complexity and Ambiguity of Images. (A) The two images are of the same object (Dan Kersten's canoe) but the intensity profiles below (plots of the image intensity as a function of position) are very different. It would be very hard to look at these images (represented as plots) and determine that there is a canoe in each. (B) The face appears identical in the two bottom images, but the top two images show that one face is normal and the other is very distorted (more on this in the Lighting chapter). (C) Images of certain obkects (particularly of specular one – like polished metal) depend very strongly on the illumination conditions.

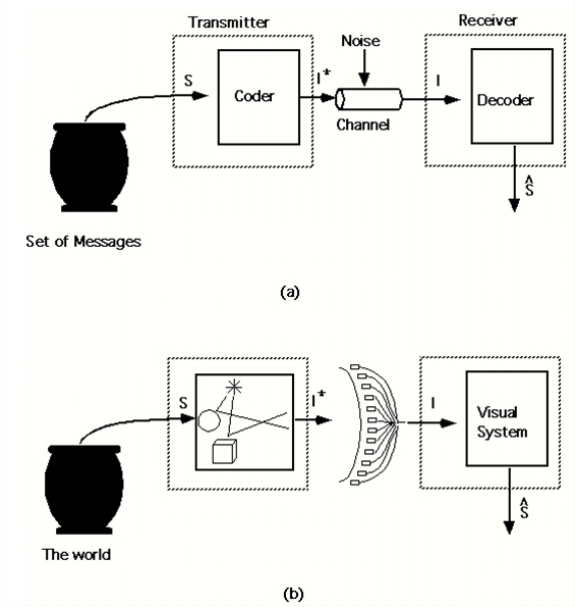

Figure 2: Information Decoding. In standard decoding (e.g., telephone, morse code) the information is encoded and decoded by a system that knows the encoder. The encoding is designed to make transmitting and decoding signals efficient. In vision, the encoding is performed by the physics of the world and hence the decoding is considerably harder.

incrementally learning the simpler parts of vision first and then proceeding to the more complex parts. The study of infants (e.g., Kellman) suggests that visual abilities develop in an orchestrated manner with certain abilities developing within particular time periods – and, in particular, infants are able to perform motion tasks before they can deal with static images. Hence vision may be developed as a series of modules where easily learnable modules may be used as prequisites to train more complex modules.

But why is this incremental/modular strategy possible? Why is it possible to vision incrementally? (You cannot learn to ride a bicycle or parachuting incrementally). Stuart Geman suggests this is because the structure of the world, and of images, is compositional (i.e., is build out of elementary parts). He provocatively suggests that if the world is not compositional then God exists! What he means that if the world/images are compositional (i.e. can be built up from elementary parts) then it is possible to imagine that an infant could learn these models. But if the world is non-compositional, then it seems impossible to understand how an infant could learnt at all (the idea of compositionality will be clarified as the course develops). Less religiously, Chomsky in the 1950's proposed that language grammars were innate (i.e., specified in the genes) because of the apparent difficulty of learning them (it is known that children develop grammars even if they are not formally taught – e.g., children of slaves). But this raises the question of how genes could "learn the grammars" in the first place. Others (e.g., Hinton) have questioned whether genes can convey enough information to specify grammars (at least for vision). Recent computational work, however, (Klein and Manning) has shown that grammars can, in fact, be learnt in an unsupervised manner (partially supervised – at least, the system must know something about the form of the grammar if not the parameter values). It seem more plausible that the brain has the ability to learn complex patterns (for vision, speech, language). But this only seems possible if the world is compositional. These ideas of compositionality and modularity will be developed during the course.

2 How to formally approach vision?

How to formulate vision taking into account the (unknown) structures/regularities and compositionality/modularity?

This course argues that vision should be formulated in terms of probability distributions formulated on structured representations (e.g., graphs and grammars). In the last few years, there has been a revolution in the scope of probabilistic modeling arising from combining ideas/concepts from Computer Science, Engineering, Mathematics, and Statistics. This revolution has been lead by the machine learning, neural networks, cognitive modeling, and artificial intelligence communities inspired by the need to design artificial systems capable of performing "intelligent tasks" (and also be researchers attempting to understand the complexities of human intelligence).

One main idea here is the formulation of vision as Bayesian inference. The idea is that we have probability distributions $P(I|W)$ for the image generation process and $P(W)$ for the state of the world. Then vision reduces to interpreting $W^* = \arg\max P(W|I)$, where $P(W|I) = P(I|W)P(W)/P(I)$. In vision, these ideas date back to Grenander's concept of Pattern Theory Grenander (1950's-1990's) was ahead of his time (slowness of computers, lack of effective inference algorithms, lack of knowledge about the complexities of vision).

The Bayesian approach can be illustrated by an example from Sinha, see figure (3). Suppose you have an image of a cube. The likelihood function $P(I|W)$ rules out all interpretations W which are not consistent with the image – so W could be a cube or another structure (abstract work of art) that can project to the image of a cube, but it cannot be a sphere, or a face, or a giraffe. But, after using the likelihood, there remain many possible interpretations. The prior is needed to narrow them down to a cube – because a cube is more likely (unless you are in an artist's studio).

Grenander's idea of inference was summarized as "analysis by synthesis". Taken literally, it corresponded to performing stochastic sampling from $P(I|W)P(W)$ until you generated an image that agreed with the input image. This is an example of top-down processing which is opposite to bottom-up processing (more standard in computer vision). Mumford speculated that pattern theory – analysis by synthesis – was able to explain the feedback connections observed in the brain (the feedforward connections correspond to bottom-up).

Specifying a problem by $P(I|W)P(W)$ is called *generative* since it enables us to obtain stochastic samples of images I corresponding to world state W. It can be contrasted to *discriminative* methods, used in machine learning, which (broadly speaking) attempt to model the posterior distribution $P(W|I)$ directly. Recent work in machine learning develops discriminative models $P(W|I)$ which are also defined on complex structures (graphs and grammars). Hence

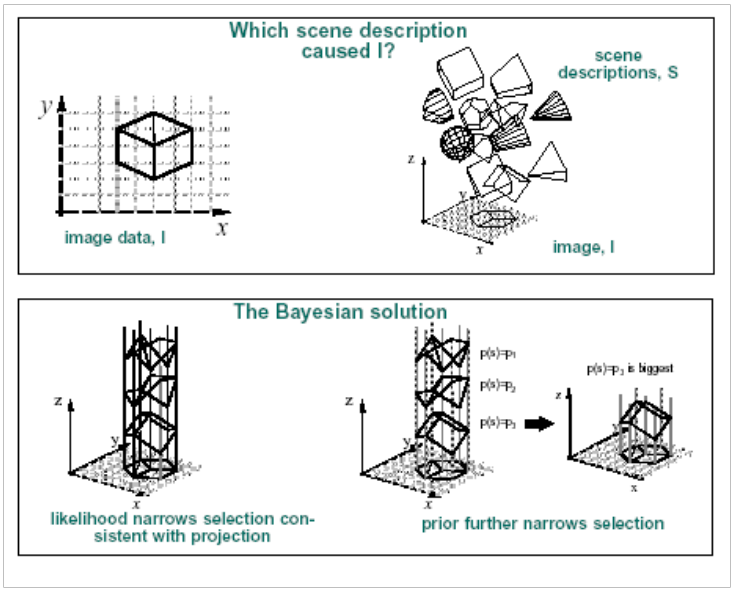

Figure 3: Sinha figure. The likelihood term $P(I|W)$ constrains the interpretation of I to scenes/objects which are consistent with it. But there remain many possibilities. The prior $P(W)$ is needed to give a unique interpretation by biasing towards those W which are most likely.

the boundary between generative and discriminative models is increasingly getting blurred (as we will see in the course).

This leads to a research program which is to develop generative and discriminative probability models which can capture the complexities and ambiguities of images. We will first start with simple probability distributions defined on simple structures (e.g., simple graphs) and then proceed to increasingly complex structures. In all cases, we will have to deal with three major questions: (I) Are the models rich enough to *represent* the complexities of the visual tasks. (II) Do we have inference algorithms to estimate $W^* = \arg\max_W P(I|W)P(W)$ efficiently? (III) Do we have learning algorithms so that we can learn these probability distributions with supervision, or partial supervision? We have to balance the computational aspects of the models (inference and learning) with their ability to represent the visual tasks.

The course describes how to develop this program. The ability to do so is a consequence of the compositionality/modularity of vision. We are able to solve vision tasks without having to solve the entire vision problem – i.e., we do not need to know the full distributions $P(I|W)P(W)$ in order to make progress.

3 Images and Edge Detection

An image is a set of intensity values: $I(\vec{x}) : \ x = 1, ..., 1024, \ y = 1, ..., 1024$ where $0 \leq I(\vec{x}) \leq 255$. Suppose you are given the intensity values – the numbers $I(\vec{x})$ not the intensity values – how would you start to interpret it? One of the first things to try is to detect places in the image where the intensity changes rapidly. These are *edges*. They typically occur at the boundaries of objects or at discontinuities in texture. For example, observe that there are big intensity discontinuities in the images in figure (1)(A) at the boundaries of the canoes.

The most straightforward way to design an edge detector is to calculate the intensity gradient $\vec{\nabla} I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$. Label a point $\vec{x}$ as an edge if $|\vec{\nabla} I(\vec{x})| > T$, where T is a threshold. (The derivatives are approximated by the differences on the image lattice). This will give a very crude edge detector. More generally, we can define the edge detector by a filter (linear or non-linear) $\vec{\phi} * I(\vec{x})$.

How to formulate this statistically in the spirit of this course? Get a dataset of images and label the edges, see figure (4). Then learn probability distributions $P(\phi * I(\vec{x})|\vec{x}\,\text{onedge})$ and $P(\phi * I(\vec{x})|\vec{x}\,\text{offedge})$, see figures (5). These

Figure 4: Upper Panels: the data images. Lower Panels: the groundtruth edge maps

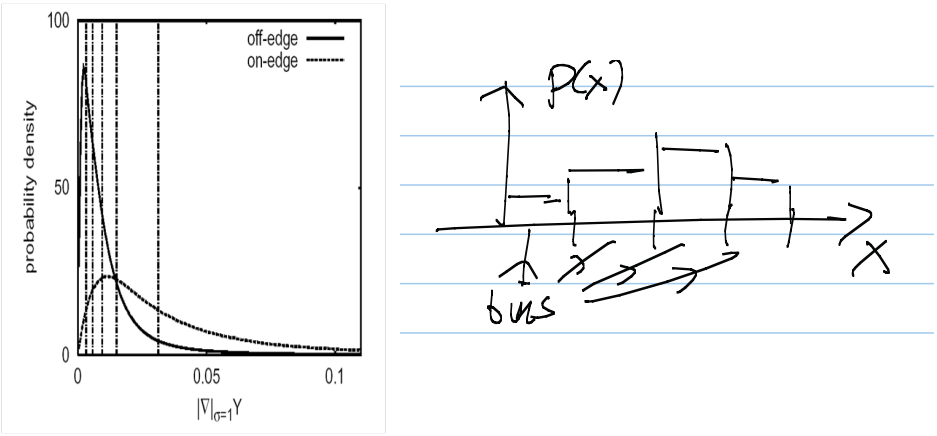

Figure 5: Left Panel: The distribution $P(.|on)$ and $P(.|off)$ represented as histograms. Observe that $logP(.|on)/P(.|off)$ is monotonically increasing as a function of the image gradient. Right Panel: a Histogram. NOTE: ADD $\log P(.|on)/P(.|off)$ FIGURE!!

distributions can be represented non-parametrically (e.g., histograms) or by parameterized models (e.g., Gaussians). (These distributions should be learnt on the training dataset and tested on the test dataset and perform cross-validation to ensure generalization).

Formulate edge detection as a log-likelihood ratio test. Label a point $\vec{x}$ as an edge if $\log \frac{P(\phi * I(\vec{x})|\vec{x} \text{ onedge})}{P(\phi * I(\vec{x})|\vec{x} \text{ offedge})} > T$, where ϕ is the edge detector filter and T is a threshold. You can plot the Receiver Operating Characteristic (ROC) curve of true positives as a function of false positives (each value of T determines a point on this curve), see figure (6).

Does this improve over simply thresholding the filter response – i.e. $\phi * I(\vec{x}) > T$, for some T. The answer is often no. The reason is that $\log \frac{P(\phi * I(\vec{x})|\vec{x} \text{ onedge})}{P(\phi * I(\vec{x})|\vec{x} \text{ offedge})}$ is typically a monotonic function of $\phi * I(\vec{x})$ because $P(\phi * I(\vec{x})|\vec{x} \text{ offedge})$ is usually peaked at 0 (the derivatives of an image are usually small at most places in the image) and then gradually decreases while, by contrast, $P(\phi * I(\vec{x})|\vec{x} \text{ onedge})$ is typically small at 0, then increases for larger $\phi * I(\vec{x})$ and then decreases again – see figure (5). This monotonic relationship means it does not matter if the threshold is placed on the filter response or on the log-likelihood. So there is no advantage in using the statistical approach.

But the situation changes if you combine two, or more, different edge detectors to obtain a vector-valued edge detector $\vec{\phi} * I(\vec{x})$. The statistical approach gives a natural way to combine these edge detectors. Label $\vec{x}$ as an edge if $\log \frac{P(\vec{\phi} * I(\vec{x})|\vec{x} \text{ onedge})}{P(\vec{\phi} * I(\vec{x})|\vec{x} \text{ offedge})} > T$. The results of this are superior to putting thresholds on the individual edge detectors, as can be seen from the ROC curves and other performance measures – see figure (8).

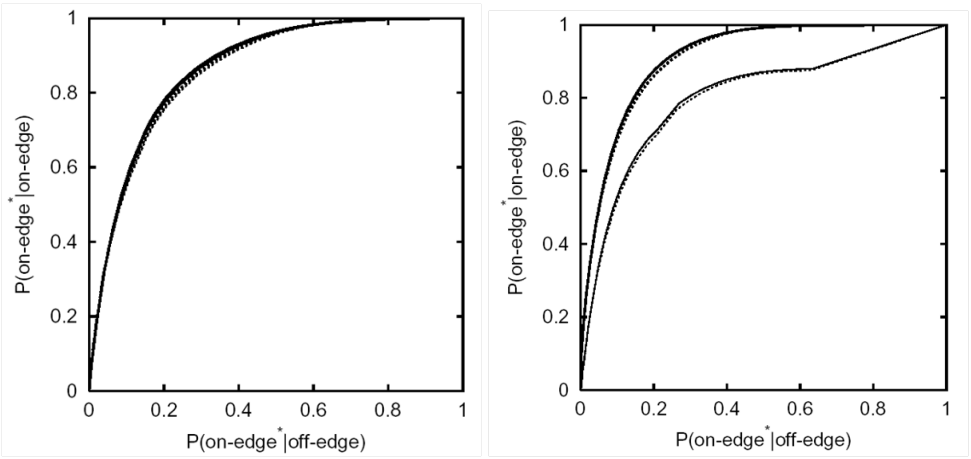

Figure 6: ROC curves. Left Panel: generalization – the ROC curves for the test and training datasets are very similar. Right Panel: failure to generalize – the ROC curves for the test and training datasets differ.

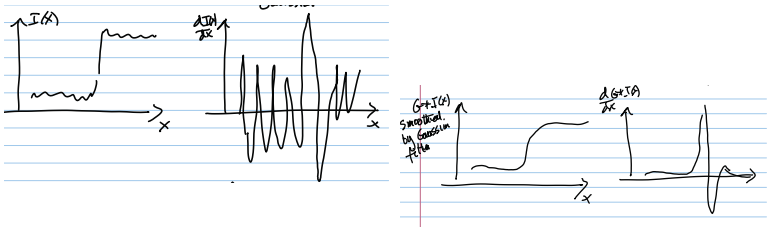

Figure 7: Derivatives of Images at different scales. Left Panel: the derivatives will be large at many places in a noisy image (even though there is only one large step edge). Right Panel: smoothing the image with a Gaussian and then differentiating will make the derivatives much smaller in most of the image but still keep a big response at the step edge (but weaker than before).

Where do these other edge detectors come from? There are a whole set of different edge detectors proposed in the computer vision literature. In particular, you can detect edges at different scales in the image. This is performed by smoothing the image by a linear filter (often a Gaussian) to obtain:

$$G_\sigma * I(\vec{x}) = \int \int dudvG(x-u, y-v:\sigma)I(u,v), \tag{1}$$

where σ is the standard deviation of the Gaussian (the larger σ the more the smoothing). This smooths the image by removing small image fluctuations, which might nevertheless have large intensity gradients, and hence respond strongly to edge detectors. Smoothing will also degrade large intensity gradients, which are more likely to be edges, but to a lesser extent – see figure (7). We can apply an edge detector $\phi(.)$ to the smoothed image to obtain a set of new edge detectors $\phi G_\sigma * I(\vec{x})$ by varying σ, see figure (7). We can combine these detectors to give a vector-valued detector – e.g. $\vec{\phi} = (\phi, \phi G_\sigma)$ – and learn the distributions $P(\vec{\phi} * I(\vec{x})|W(\vec{x}))$. *Important point* – there is a limit to how many filter you can combine without running out of training data. If you represent distributions by histograms, then the amount of data required scales like exponentially with the number of filters (quadratically if you use Gaussian distributions).

The performance of edge detectors can be evaluated by ROC curves or by measures of the difference between the distributions $P(.|on), P(.|off)$. The Chernoff information as a measure of the different between the distributions $P(.|on)$ and $P(.|off)$. The larger the Chernoff, the more the distributions differ. The Chernoff information $C(p,q) =$

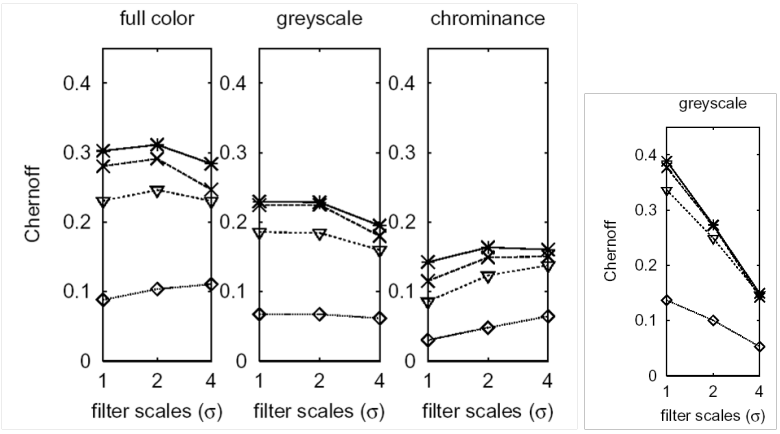

Figure 8: Evaluation of different filters and combinations of filters. Stars for the joint distribution of (N_1, N_2), crosses for N_1, triangle for $|\vec{\nabla}|$, diamonds for ∇^2. N_1, N_2 are the first and second eigenvalue of the matrix-valued operator $G * (\vec{G} * I)(\vec{G} * I)^T$, where G is a Gaussian and ^{T} denotes vector transpose.

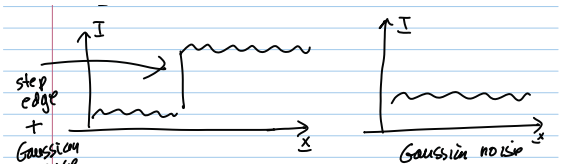

Figure 9: The assumptions made by the Canny edge detector. Left Panel: edges are step edges corrupted by additive Gaussian noise. Right Panel: non-edges are Gaussian noise.

$-\min_{0\leq\lambda\leq 1}\log\{\sum_y p^\lambda(y)q^{1-\lambda}(y)$. Chernoff is $(\mu_1-\mu_2)/(8\sigma^2)$ if $p(.), q(.)$ are Gaussian distributions with identical variance σ^2 and means μ_1, μ_2. The use of Chernoff is motivated by the fact that the error rate in labeling N samples $y_1, ..., y_N$ as either all from $p(.)$, or all from $q(.)$, behaves as $\exp\{-NC(p,q)\}$ for large N. The Chernoff information also can be used to bound the Bayes risk for classifying a single sample y as being from $p(.)$ or $q(.)$. Figure (8) shows how combining different filters can lead to better edge detectors.

These statistical edge-detectors are very successful when tested on large datasets. There are improvements to the methods described here – you can design special features (Berkeley – see later), you can use alternative methods for combining different edge detectors (e.g., AdaBoost). These will be described later in the course.

Now we compare statistical edge detectors to the default Canny edge detector (Canny 1983). This formulates edge detection as distinguishing between a step-edge with additive Gaussian noise and pure Gaussian noise, see figure (9) (this is an oversimplification). These can be thought of as generative models for edges and non-edges. How well do they correspond to real images? It turns out that step edges are a reasonable first order approximation to real edges (although there are a variety of other edge profiles). But non-edges are often not well described by pure Gaussian noise. It is a reasonable approximation for indoor images without texture, but fails in outdoor scenes because of vegetation (we say more about texture later in the course). Hence the performance of statistical edge detectors are significantly better than Canny on challenging outdoor images, see figure (**??**).

4 Edge Detection and the Bigger Picture

How does the edge detection example fit into the bigger picture of probability distributions defined on graphs, see figure (**??**)?

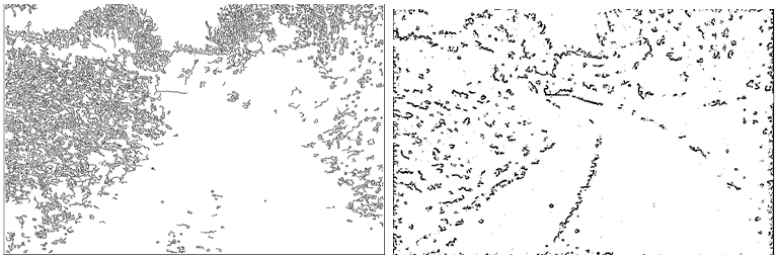

Figure 10: Comparison of Canny and Statistical Edge Detection. Left Panel: Canny edge. Right Panel: Statistical Edge Detector. Observe that Canny has bigger response on the texture in the background but also fails to detect the sides of the road.

It corresponds to a *discriminative* model $P(W|I)$ where $W = \{W(\vec{x}) : \vec{x} \in D\}$ is a set of binary-valued variables – $W(\vec{x}) = 1$ if $(\vec{x})$ is an edge, and $W(\vec{x}) = 0$ otherwise. The conditional distribution $P(W|I)$ is factorizable and can be *represented by the graphical structure* is illustrated in figure (**??**):

$$P(W|I) = \prod_{\vec{x}} P(W(\vec{x})|\vec{\phi} * I(\vec{x})), \tag{2}$$

where $P(W(\vec{x}) = 1|\vec{\phi} * I(\vec{x})) = \frac{P(\vec{\phi} * I(\vec{x})|W(\vec{x})=1)P(W(\vec{x})=1)}{P(\vec{\phi} * I(\vec{x}))}$, with $P(\vec{\phi} * I(\vec{x})|W(\vec{x}) = 1) = P(\vec{\phi} * I(\vec{x})|\vec{x}\,\text{onedge})$ and $P(\vec{\phi} * I(\vec{x})|W(\vec{x}) = 0) = P(\vec{\phi} * I(\vec{x})|\vec{x}\,\text{offedge})$. The prior $P(W(\vec{x}) = 1$ takes into account that far fewer image pixels are likely to be edges (roughly five to ten percent of image pixels are edges in typical images).

(NOTE: SHOULD HAVE A LOSS FUNCTION – E.G. SO THAT FALSE NEGATIVES COST MORE THAN FALSE POSITIVES) is the equivalent to the threshold T.

The factorizable form of $P(W|I)$ means that the probability of a pixel being an edge is independent of the probability of any other pixels, including its nearest neighbors, from being edges conditioned on the values of $\vec{\phi} * I(.,.)$. Factorization is a big approximation and, as we will show, non-factorized models perform better. But a big advantage of factorization is that the inference algorithm (and the learning) is very simple.

The *inference* algorithm to estimate $W^* = \arg\max_W P(W|I)$ is simple because of the factorizability of $P(W|I)$. At each image pixel $\vec{x}$ – label the pixel as edge if $P(W(\vec{x}) = 1|\vec{\phi} * I(\vec{x})) > P(W(\vec{x}) = 0|\vec{\phi} * I(\vec{x}))$ and as non-edge otherwise. This takes linear time in the size of the image.

The *learning* is also simple because of factorizability. The training set consists of two sets of image pixels $i \in X$ with edge labels $W(i) \in \{0, 1\}$ and with filter values $\vec{\phi} * I(i)$. We subdivide this into two sets $X_1 = \{i : \text{s.t.}W(i) = 1\}$ and $X_0 = \{i : \text{s.t.}W(i) = 0\}$. We learn $P(\vec{\phi} * I(i)|W(i) = 1)$ and $P(\vec{\phi} * I(i)|W(i) = 0)$ using X_1 and X_0 respectively. We also learn the prior $P(W(i))$ from the relative sizes of X_1 and X_0.

This is a discriminative model since it directly learns the posterior distribution $P(W|I)$ and has no explicit model $P(I|W)$ for generating the image. But it has a prior distribution $P(W) = \prod_{\vec{x}} P(W(\vec{x}))$. Observe that it does have a generative model for generating the features:

$$P(\vec{\phi} * I|W) = \prod_{\vec{x}} P(\vec{\phi} * I(\vec{x})|W(\vec{x})). \tag{3}$$

So we can use $P(W)$ and $P(\vec{\phi} * I|W)$ to generate an image of features $\vec{\phi} * I = \{\vec{\phi} * I(\vec{x})\}$. But there are two problems with this: (i) there is no guarantee that the generated feature image $\vec{\phi} * I$ are consistent with an underlying real image I, (ii) even if it is consistent, it will not correspond to the correct distribution on the image. We will return and clarify these problems later (Coughlan's examples).

5 Labelling Image Regions

We can apply the same approach to the task of labeling image pixels as a number of discrete regions – e.g., sky, road, vegetation, other. To do this we define $W = \{W(\vec{x})\}$ so that $W(\vec{x}) \in L$, where $L = \{sky, road, vegetation, other\}$ is a set of labels. We use similar filters $\vec{\phi}(.)$ as before to learn distributions $P(\vec{\phi} * I(\vec{x})|W(\vec{x}))$ for $W(\vec{x}) \in L$.

This results in a factorized model of similar form to equation (2) and with similar graph structure:

$$P(W|I) = \prod_{\vec{x}} P(W(\vec{x})|\vec{\phi} * I(\vec{x})), \tag{4}$$

The learning and inference are simple generalizations of those for the binary task of edge detection. The only difference is the large number of labels.

This simple model does surprisingly well. If the filters use color or texture or a combination – then they can achieve very high accuracy on labeling regions such as sky, road and vegetation. This success happens because these regions are comparatively homogeneous (i.e. all parts of the sky as similar to each other – FULL DEFINITION OF HOMOGENEITY SOMEWHERE??). But it is less successful for detecting non-homogeneous objects like houses. This motivates more advanced non-factorizable models which will be described later in the course.

PUT A FIGURE TO ILLUSTRATE KONISHI'S RESULTS ON REGIONS!!

6 Bayes Decision Theory and Loss Functions

So far, we have assumed that the goal of inference is to obtain the Maximum a Posteriori (MAP) estimate $W^* = \arg\max_W P(W|I)$. But why? Where does this come from? What are the alternatives?

The basis is Bayes Decision Theory which formulates problems as minimizing the expected loss. To make this precise, assume we have a joint distribution $P(W, I)$, a set Λ of decision rules $\alpha(.)$, and a loss function $L(\alpha(I), W)$ for making decision $\alpha(I)$ for input I when the true state is W.

Bayes Decision Theory states that you should pick the decision rule that minimize the risk which is defined to be the expected loss:

$$R(\alpha) = \sum_{I,W} L(\alpha(I), W)P(I, W), \tag{5}$$

where we replace the summations by integrals if I, W, or both are continuous valued.

$$\text{Bayes Rule } \alpha^* = \arg\min_{\alpha\in\Lambda} R(\alpha), \quad \text{Bayes Risk } R(\alpha^*) = \min_{\alpha\in\Lambda} R(\alpha). \tag{6}$$

NOTE: there are a few mathematical special cases where $R(\alpha^*) \neq \arg\min_{\alpha\in\Lambda} R(\alpha)$ but they very rarely occur outside mathematics books.

We can re-express the risk as:

$$R(\alpha) = \sum_I P(I)R(\alpha|I), \ \text{where} R(\alpha|I) = \sum_W L(\alpha(I), W)P(W|I). \tag{7}$$

Hence minimizing the Bayes risk is equivalent to minimizing the conditional risk $R(\alpha|I)$ for each I (i.e., the Bayes decision rule for input I is independent of its decision for other inputs – alternatively, the distribution $P(I)$ has no effect on determining the Bayes rule). This gives:

$$\alpha^*(I) = \arg\min_{\alpha(I)} \sum_W L(\alpha(I), W)P(W|I), \text{ discrete } W$$

$$\alpha^*(I) = \arg\min_{\alpha(I)} \int dW L(\alpha(I), W) P(W|I), \text{ continuous } W. \tag{8}$$

The MAP estimate arises as a special case for a particular choice of loss function.

In the discrete case, set $L(\alpha(I), W) = 1 - \delta(\alpha(I), W)$, where the delta function $\delta(\alpha(I), W)$ takes value 1 if $\alpha(I) = W$ and value 0 otherwise (i.e., correct responses pay no penalty but incorrect responses are weighted the same). The Bayes rule reduces to maximizing $\sum_W \delta(\alpha(I), W) P(W|I) = P(W = \alpha(I)|I)$, which is the MAP estimate.

For certain applications – e.g., edge detection, face detection – it may be better to use a different loss function which penalizes false negatives (failure to find edges/faces) more than false positive (finding edges/faces where they do not exist). The reason is that we can use later processing (i.e., other models) to eliminate the false positives. But it is harder to resurrect the false negatives.

In the continuous case, set $L(\alpha(I), W) = -\delta(\alpha(I) - W)$, where $\delta(.)$ is the Dirac delta function (i.e., $\delta(x) = 0,\ x \neq 0$ and $\int dx \delta(x) = 1$, provided the range of integration contains the point $x = 0$. The Bayes rule reduces to maximizing $\int dW \delta(\alpha(I), W) P(W|I) = P(W = \alpha(I)|I)$ which is the MAP estimator.

Observe that the Dirac delta function is a strange choice of loss function because it pays an (infinite) penalty unless the decision is perfectly correct. There is no partial credit for making a decision $\alpha(I)$ that differs from the correct decision W by an infinitesimal amount. This is highly unrealistic. More reasonable choices of loss function are to set $L(\alpha(I), W) = -G(\alpha(I) - W : \sigma)$, where G is a Gaussian whose variance/covariance determines how much partial credit to give. In practice, the simplicity of MAP estimation – and the difficulty of determining how to assign partial credit – means that MAP estimation is often used in practice. Alternatives will be described later in the course. They include the mean estimate $\int P(W|I) W dW$ which occurs for quadratic loss function $L(\alpha(I), W) = (\alpha(I) - W)^2$. In summary, MAP estimation for continuous variables should be used with caution.

Bayes decision theory and approximate models. There is a simple but important conceptual point to be made from the Bayes Risk (which several well-known vision researchers have got wrong!). The Bayes risk is obtained by minimizing with respect to all decision rules. If we reduce the set of allowable decision rules, for example if we do edge detection with a restricted class of filters, then we have to do worse.

Bayes decision theory has some implications which may be counter-intuitive. Suppose the likelihood term $P(I|W)$ is sufficient to determine the correct W. Then the prior $P(W)$ may bias the result. The reason is that Bayes decision theory assumes that you should make the best decision *on average*, which does not correspond to making the best decision on a specific example. Bayes decision theory can be a bad guide for how to make one-time only decisions – such as buying a house, or gambling on the stock market. In such case, it may be wiser to take into account the worst-case loss rather than the average case (i.e., how much money can you afford to loss in the stock market). But the mathematics for this is much more complicated and worrying about the worst case is often paranoid.

Appendices

A1: Filters: Discrete and Continuous

Although images are defined on a discrete image lattice it is often convenient to formulate theories on continuous image domain and then discretize them.

For continuous images $I(\vec{x}) : \vec{x} \in D$, we define (spatial invariant) linear filtering by a filter $\phi(u, v)$ to be:

$$\phi * I(\vec{x}) = \int \int du dv \phi(x - u, y - v) I(u, v). \tag{9}$$

This is convolution. We can apply Fourier transforms to obtain $\hat{\phi}(\omega_x, \omega_y) = \hat{\phi}(\omega_x, \omega_y) \hat{I}(\omega_x, \omega_y)$, where (ω_x, ω_y) is frequency. HOW MUCH TO SAY ABOUT FOURIER THEORY??

For discrete images $I(\vec{x}) : x = 1, ..., 1024\ y = 1, ..., 1024$ and a discrete filter $\phi(u, v)$, we define a linear filter to be:

$$\phi * I(\vec{x}) = \sum_u \sum_v \phi(x-u, y-v) I(u,v). \tag{10}$$

The simplest filters are derivatives –e.g. d/dx and d/dy. These can be converted into discrete filters such as $1, -1$ (in one dimension). Better derivative approximations can be performed (see standard textbooks).

Standard classes of filters. Derivatives of Gaussians. The idea is that the Gaussians smooth the image.

Other filters are Gabors: $G(\vec{x}) = \frac{1}{2\pi|\mathbf{\Sigma}}\exp\{-(1/2)\vec{x}^T\mathbf{\Sigma}^{-1}\vec{x}\}\exp i(\omega_x x + \omega_y y)$, where (ω_x, ω_y) is in the direction of the second eigenvector of $\mathbf{\Sigma}$. The Gabor filter is complex and can be decomposed into a cosine Gabor (the real part) and a sine Gabor (the imaginary part) ($\exp\{i\theta\} = \cos\theta + i\sin\theta$). *Energy filters* are defined by summing the cosine and sine parts.

Discuss wavelets.

A2: Image Formation

How are images formed?

Light rays hit objects and get reflected. This will be described in later chapters. There is Computer Graphics where the goal is to generate realistic images. This involves studies of models for how objects reflect light (radiosity models). Computer vision can be formulated as inverting this process. But unfortunately this is much harder. For certain types of illumination models this is relatively easy (e.g. Lambertian models) but for others it is very difficult. Learning may be more useful than Physics based models.

Cameras, focal points, blurring, and so on. We need to say something about this. People should be aware of the factors involved.

NEED SOME BACKGROUND MATERIAL HERE!! BUT NOT CRITICAL FOR THE COURSE!!

A3: Learning – Memorizing and Generalizing

The study of Machine Learning has taught us a lot about the differences of *generalizing* and *memorizing*.

All learning is done from a finite *training set* of data $\{(x_i, y_i) : i = 1, ..., N\}$ but we want decision rules to be valid for data that we have not seen yet. Most studies of this problem assume that the data samples are independent identically distributed (i.i.d.) from some unknown distribution $P(\vec{x})$. We design a decision rule $\alpha(.) \in \Lambda$ which maps x to y. We choose a loss function $L(\alpha(x), y)$ which is the penalty for making decision $\alpha(x)$ for data x when the true decision is y (i.e., usually $L(\alpha(x), y) = 0$ if $\alpha(x) = y$).

From the training dataset, we can learn a decision rule $\hat{\alpha}$ to make the *empirical risk* small, where the empirical risk (average loss over the training dataset) is defined by:

$$R_{emp}(\alpha) = \frac{1}{N}\sum_{i=1}^{M} L(\alpha(x_i), y_i). \tag{11}$$

But this is only *memorization* (i.e., applies only to the training data) unless $\hat{\alpha}$ also minimizes the *risk* (expected loss) of all the samples from the distribution:

$$R(\alpha) = \sum_x \sum_y P(\vec{x}) L(\alpha(x), y). \tag{12}$$

Memorization typically occurs when the amount N of training data is small and the set Λ of possible $\alpha(.)$ is large (technically the *capacity*). In this situation, it is too easy to find a decision rule that gives a good fit to the training data by chance (this is how conspiracy theories get started).

To obtain *generalization*, we require that $R(\hat{\alpha}) \approx R_{emp}(\hat{\alpha})$ (over-simplified?). How can we be sure that this is the case?

For certain classes of problem (making decisions) there is a beautiful theory due to Vapnik (see also Valiant) which shows that with probability greater than $1 - \delta$ that

$$R(\alpha) \leq R_{emp}(\alpha) + \phi(N, h, \delta). \tag{13}$$

where h is the VC dimension which is a measure of the capacity of the set of classifiers $\alpha \in \Lambda$. Provided N is much large than h and than $|\log \delta|$ then we can be sure that minimizing R_{emp} will make R small. But this theory is unfortunately mostly of conceptual use because the bounds are often not tight enough. The theory is slightly paranoid because it has to rule out the probability that a rule might classify the training data correctly because of a chance structure in the dataset (e.g. in high-dimension space d and only N datapoints it is always possible to get a rule to make any dichotomy).

Instead, in practice, we use the idea of cross-validation (of which there are many variants). This involves a training dataset and a test dataset. The decision rule is learnt on the training dataset and evaluated on the test dataset (this can be thought of as using the test dataset to estimate the Bayes risk). This is not rigorously guaranteeing that $R(\hat{\alpha}) \approx R_{emp}(\hat{\alpha})$ (because there is always the chance that both training and test datasets are atypical of the distribution $P(\vec{x})$). But how paranoid do you want to be?

A4: Adaption, Transfer Learning

Are all datasets the same? Suppose you learn an edge detector on one dataset – will it also work on another dataset? The answer is clearly not if the datasets come from different sources. E.g. suppose one dataset the images values have been thresholded to lie in the range $128 - 136$ (stranger things have happened in medical image datasets).

For example, the Soweby dataset and the South Florida dataset differ because Sowerby is taken outdoors in the English countryside while South Florida is taken mainly indoors and with textured regions suppressed. Looking at the images makes it easy to see that edges in the South Florida images are sharper.

Can you transfer learning from one domain to another? Or in biological terminology adapt (when one animal moves from the water to the land). For edges, this is not that hard by exploiting structure. I.e. you can estimates the filter statistics in the background by simply computing the statistics over the entire image (and the edges are few enough to only partially contaminate the data).

A5: Relations to Neuroscience

Computational neuroscientists model receptive field properties of neurons in the retina, LGN, and V1 (part of cortex) as linear and non-linear filters.

The receptive fields in the retina and LGN are often modeled as linear receptive fields. These models are believed to be fairly reliable – in the sense that neuroscientists can predict the response of these neurons to novel stimuli, or determine the image from the activity of the retina or LGN cells (Dan – check this!!). There has been some success (e.g, Atick) at deriving the linear receptive field properties of neurons in the retina and the LGN from information theoretic principles (i.e., the filters are designed to encode images so that they can be transmitted reliably and efficiently from the retina to the LGN and then from the LGN to the cortex (V1). The receptive field properties depend on the statistics on the input images and so will change (adaption) as the statistics vary.

Receptive field properties in the early visual cortex are more complicated. It is sometimes stated that the receptive fields can be modeled by Gabor functions (simple cells) and squares of Gabor functions (complex cells), but this seems only approximate (Ringach). In particular, these models do not successfully predict the responses of the neurons to real images (the experiments for measuring the receptive fields are performed on synthetic stimuli). There has been some success in predicting receptive field properties by sparse coding ideas and independent component analysis (these will be described later in the course).

A6: Relations to Machine Learning and Neural Networks

The ideas of memorization/generation were developed in the 1980's. Valiant' theory of the learning (two very short papers) inspired the development of PAC learning theory (Rivest). Vapnik's work was highly respected by a small group but only became seriously influential when Vapnik emigrated to the USA (from the Soviet Union) and his work lead to Support Vector Machines.

MOVE THIS TO SOMEWHERE WHERE WE DESCRIBE MORE SERIOUS LEARNING!!