Filters and Statistical Edge Detection

- Formulating Edge Detection as Statistical Inference.
- Handout notes from Kokkinos to review linear filters.

Edge Detection

- *Edges* are places in the image where the intensity changes rapidly. They typically occur at the boundaries of objects or at discontinuities in texture.
- 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.
- It is often better to first filter the image by a smoothing kernel – e.g., a Gaussian – to smooth out small intensity fluctuations and then take the derivative. This is equivalent to convolving the image with the derivative of a Gaussian. Most classic edge detectors (e.g., Roberts, Sobel, Canny) are roughly of this form.
- Now we consider a statistical approach where we have an annotated dataset where pixels $\vec{x}$ are labelled as edge or non-edge. Then for any filter $\phi(.)$ we can learn probability distributions $P(\phi * I(\vec{x})|\vec{x} \mathrm{\ on} - \mathrm{edge})$ and $P(\phi * I(\vec{x})|\vec{x} \mathrm{\ off} - \mathrm{edge})$. These distributions could be represented non-parametrically (e.g., histograms) or by parameterized models (e.g., Gaussians).

Edge Detection as Statistical Inference (1)

- 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.
- If $\phi(.)$ is a simple derivative filter – e.g., $|\vec{\nabla} I(\vec{x})|$ – then this is equivalent to simply thresholding the gradient $|\vec{\nabla} I(\vec{x})|$.
- This is because the $\log \frac{P(\phi * I(\vec{x})|\vec{x} \text{ onedge})}{P(\phi * I(\vec{x})|\vec{x} \text{ offedge})}$ is typically a *monotonic function* of $\phi * I(\vec{x})$. $P(\phi * I(\vec{x})|\vec{x} \text{ offedge})$ 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.
- Monotonicity of the log-likelihood function (as a function of the filter response) means that putting a threshold on the log-likelihood is essentially the same as putting a threshold on the filter response (and vice versa). Hence learning the probability distributions serves no purpose.

Edge Detection as Statistical Inference (2)

- 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 $\log \frac{P(\vec{\phi}*I(\vec{x})|\vec{x} \text{ on}-\text{edge})}{P(\vec{\phi}*I(\vec{x})|\vec{x} \text{ off}-\text{edge})} > T$. The results of this are superior to putting thresholds on the individual edge detectors.
- The key idea is that the use of statistics (from the benchmarked dataset) gives us a principled way to combine different cues (i.e. filters) for edge detection.
- Smoothing will also degrade large intensity gradients, which are more likely to be edges, but to a lesser extent. 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. 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).

Edge Detection as Statistical Inference (3)

- The statistical edge detectors were very effective at detecting edges on the Sowerby dataset and performed much better than traditional edge detectors such as Canny (Canny knew that he should combine edge cues from different image scales but, lacking probabilities, he had no principled way to combine information from these cues). Statistical edge detectors performed similar to Canny on the South Florida edge dataset, but this dataset was not very challenging because it consisted of indoor images with texture regions removed.
- The Sowerby dataset was criticized because the ground truth edges were obtained by applying a set of edge detection algorithms and then used human annotators to prune out the false positive edges (the thresholds for the edge detectors were set to be small so that there were very few false negatives). UC Berkeley created the Berkeley Segmentation Dataset (BSD) where up to five student independently annotated edges in the images (these annotators were deliberatively not given detailed instructions for how to annotate, i.e. no definition of edge was given). The BDS dataset became the standard benchmark for evaluating edge detectors.
- Other statistical edge detectors were constructed. Their initial performance was similar to the log-likelihood methods described in the previous slides (hard to evaluate this precisely because BDS was not available when the log-likelihood detectors were implemented, but the follow-up work reimplemented a simplified variant of the log-likelihood method).

Edge Detection as Statistical Inference (4)

- The same approach could be used for semantic segmentation. But this only worked for homogeneous segmentation classes like sky and water, but not for inhomogeneous classes like cars and airplanes. This is because simple filters were able to capture the appearance of homogeneous textures, but many more filters would be needed to deal with inhomogenous texture. This is possible with methods like deep networks because they contain a very large number of different features. The log-likelihood method uses a small number of filters but combines them together using a complex decision rule.
- The log-likelihood method has conceptual advantages over recent methods like Deep Nets because they are generative. I.e., they learn $P(\phi * I(\vec{x})|\vec{x} \text{ on} - \text{edge})$ and $P(\phi * I(\vec{x})|\vec{x} \text{ off} - \text{edge})$. This enables domain transfer (between two different image domains). E.g., , Konishi et al. performed domain transfer between the Sowerby and South Florida datasets which had fairly similar statistics for $P(\phi * I(\vec{x})|\vec{x} \text{ on} - \text{edge})$, but very different statistics for $P(\phi * I(\vec{x})|\vec{x} \text{ off} - \text{edge})$ (due to very different textures). They observed that the off-edge statistics could be learnt (approximately) from either dataset without requiring any annotation because they were (approximately) the distributions over the entire images (most pixels in images are non-edges). The "background" distribution $P(\phi * I(\vec{x})|\vec{x} \text{ off} - \text{edge})$ could be learnt without annotations and we can borrow/adapt the distribution $P(\phi * I(\vec{x})|\vec{x} \text{ on} - \text{edge})$ from another (annotated) domain.