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 $P(\phi * I(\vec{x})|\vec{x}\ \mathrm{on-edge})$ and $P(\phi * I(\vec{x})|\vec{x}\ \mathrm{off-edge})$. 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 $\log \frac{P(\vec{\phi} * I(\vec{x}) | \vec{x} \text{ onedge})}{P(\vec{\phi} * I(\vec{x}) | \vec{x} \text{ offedge})} > T$. The results of this are superior to putting thresholds on the individual edge detectors.
- 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).