Dynamic Thresholding for Image Analysis

Statistical Consulting Report for Edward Chan

Clean Energy Research Center

University of British Columbia

by

Libo Lu

Department of Statistics
University of British Columbia

December 14, 2009
1 Background and Data Description

Edge detection, an important technique in image processing and computer vision, aims at identifying points in a digital image at which the image color (grey level) changes sharply. These points are called features, and the rest points are called textures. One major method for edge detection, the search-based method, uses the gradient magnitude as a measure of edge strength for each pixel and applies thresholds to decide whether edges are present or not at an image point.

In this project, a large number of gray-scale images were collected from experiments. To automate post-processing, an edge detection and thresholding algorithm was written that extracts features from each image. Two thresholds are used to classify the pixels with different gradient magnitudes. The texture threshold (usually a small value) is the upper bound of the gradient magnitudes of the pixels that would be ignored, and the feature threshold, which is greater than the texture threshold, is the lower bound of the gradient magnitudes of the pixels that should be extracted and recorded. The pixels whose gradient magnitudes fall in the interval between these two thresholds would be treated by a more sophisticated method.

Each image contains 256 × 256 pixels, whose gradient magnitude is a continuous variable ranging from 0 to 1. Two fixed threshold values were adopted for all images based on past experience and you would like to assign different threshold values for each distinct image dynamically. However, the criteria for optimal threshold values have not been decided. Therefore, at this stage, how to describe the distribution of the gradient magnitude is of interest as a basis for further study.

2 Statistical Questions

• Q1. How to estimate the gradient magnitude distribution of a digital image?

• Q2. How to define an appropriate linear combination of parametric distributions to represent the gradient magnitude distribution?

3 Suggestions

We first address the above two statistical questions, then briefly introduce some other options for edge detection.
3.1 Q1: Density Estimation

The gradient magnitude distribution could vary from one image to another. Therefore, we introduce two nonparametric (distribution-free) density estimation methods for statistical question Q1. Silverman [5] gives an overview of this topic. For each image, as you suggested, suppose you are willing to consider the gradient magnitudes of the pixels $X_1, X_2, \ldots, X_n$ as independent, identically distributed observations from a continuous random variable $X$ with probability density function $f$. The random variable $X$ ranges from 0 to 1, and $f$ is to be estimated.

Histogram

The oldest and most widely used density estimator is the histogram. As an example, let $m$ be an integer and define bins as:

$$B_1 = \left[ 0, \frac{1}{m} \right), \quad B_2 = \left[ \frac{1}{m}, \frac{2}{m} \right), \ldots, \quad B_m = \left[ \frac{m-1}{m}, 1 \right].$$

Define the binwidth $h = 1/m$ and let $Y_j$ be the number of observations in $B_j$. The histogram estimator is defined by

$$\hat{f}(x; h) = (nh)^{-1} \sum_{j=1}^{m} Y_j I(x \in B_j). \quad (1)$$

The underlying idea of the histogram estimator is that for $x \in B_j$ and small $h$,

$$E(\hat{f}(x; h)) = E(Y_j) = \frac{n \int_{B_j} f(u)du}{nh} = \frac{f(x)h}{h} = f(x).$$

Thus, $f(x)$ can be estimated by $\hat{f}(x; h)$ when $h$ is small.

The binwidth $h$ is a critical parameter for the histogram estimator. A smaller binwidth leads to a relatively jagged histogram, while a larger binwidth results in a smoother histogram. Therefore, the binwidth $h$ is usually called a smoothing parameter since it controls the degree of “smoothing” being applied to the data, which is also important for other types of nonparametric density estimators.

For exploratory data analysis, the histogram is a very useful tool for density estimation. However, the histogram estimator has three disadvantages:

- For the same data, histograms with different bin edges can be quite different, such as Figure 1 which contains five shifted histograms and the frequency polygon of their average for the Old Faithful geyser duration data.\(^2\)

\(^1\)The independence assumption may not be appropriate for your project, because the gradient magnitudes at the different pixels are likely to be spatially correlated.

\(^2\)Refer to Page 127 of [6].
• Histograms are not smooth.

• It is difficult to use the histogram to represent multivariate data.

So we need to consider other options.

Figure 1: Five shifted histogram with binwidth 0.5 and the frequency polygon.

Kernel Density Estimation

A function $K$ is called a kernel if it satisfies $\int K(x) = 1$. The formula for the kernel density estimator is:

$$\hat{f}(x; h) = (nh)^{-1} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

(2)

where the smoothing parameter or bandwidth $h$ is a positive real number. Usually $K$ is chosen to be a unimodal probability density function that is symmetric about zero, for example, the standard normal density. This ensures that $\hat{f}(x; h)$ is also a density function. Figure 2 is a sample plot comparing histogram and kernel density estimates of a sample of gradient magnitudes and details on an enlarged (vertical) scale.

To study the discrepancy of the kernel density estimator $\hat{f}$ from the true density $f$ at a single
point $x$, a natural measure is the *mean squared error* (MSE), defined as follows:

\[
\text{MSE}(\hat{f}(x; h)) = E\{\hat{f}(x; h) - f(x)\}^2
\]

\[
= \{E\hat{f}(x; h) - f(x)\}^2 + \text{var}\{\hat{f}(x; h)\},
\]

where $\{E\hat{f}(x; h) - f(x)\}^2$ is the squared bias and $\text{var}\{\hat{f}(x; h)\}$ is the variance of the estimated density at $x$.

To study the global discrepancy of the kernel density estimator $\hat{f}$ from the true density $f$, a widely used measure is the *mean integrated squared error* (MISE), defined as follows:

\[
\text{MISE}(\hat{f}(\cdot; h)) = E\int\{\hat{f}(x; h) - f(x)\}^2 dx
\]

\[
= \int\{E\hat{f}(x; h) - f(x)\}^2 dx + \int \text{var}\{\hat{f}(x; h)\} dx,
\]

where $\int\{E\hat{f}(x; h) - f(x)\}^2 dx$ is the integrated squared bias and $\int \text{var}\{\hat{f}(x; h)\} dx$ is the integrated variance.

Notice that, in both discrepancy measures, there is a trade-off between the bias and variance of the density estimator $\hat{f}(\cdot; h)$. You may control the trade-off by adjusting the bandwidth $h$. In general, smaller $h$ results in smaller bias and larger variance, while larger $h$ introduces larger bias but results in smaller variance. Therefore, choosing an appropriate bandwidth $h$ is a critical issue in density estimation.
There are several approaches to select the bandwidth $h$. Some simple approaches\textsuperscript{3} adopt computable formulae to find a bandwidth that is “reasonable” for the majority of situations. More delicate approaches\textsuperscript{4} search for the “optimal” bandwidth under certain criteria, for example, minimizing the MISE. However, the latter approaches usually require extensive computation, which is also a nonignorable problem. For your project, since most images exhibit similar patterns, you may try the former approaches for simpler computation.

In MATLAB, the \texttt{ksdensity} command implements kernel density estimation without dynamic bandwidth selection. A free MATLAB software package that incorporates a simple and fast bandwidth selection method is available from the MATLAB Central File Exchange\textsuperscript{5}.

\subsection*{3.2 Q2: Finite Mixture Modeling}

The use of finite mixtures of distributions is a powerful statistical modeling tool for density estimation. Many complicated density functions can be represented as linear combinations of diverse simple density functions, such as normal density\textsuperscript{6}. For example, the bimodal density in Figure \ref{fig:3} could be approximated by a linear combination of two normal densities as follows:

$$\frac{4}{5}N(0.01, \left(\frac{1}{200}\right)^2) + \frac{1}{5}N(0.07, \left(\frac{1}{30}\right)^2)$$

Figure 3: An example of a bimodal density.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{bimodal_density.png}
\caption{An example of a bimodal density.}
\end{figure}

\textsuperscript{3}Refer to section 3.4.1 and 3.4.2 of \cite{5}.
\textsuperscript{4}Refer to section 3.4.3 and 3.4.4 of \cite{5}.
\textsuperscript{5}\url{http://www.mathworks.com/matlabcentral/fileexchange/14034-kernel-density-estimator}
\textsuperscript{6}Refer to section 1.5.2 of \cite{3} for interesting examples.
Suppose \( f \) is the density to be estimated, and suppose the density of \( X \) can be written as:

\[
f(x) = \sum_{j=1}^{g} \pi_j f_j(x),
\]

(3)

where the \( f_j(\cdot) \)'s could be any density functions and the weights satisfy \( 0 \leq \pi_j \leq 1, j = 1, \cdots, g, \) and \( \sum_{j=1}^{g} \pi_j = 1 \). Based on the properties of \( f_j(\cdot) \) and \( \pi_j \), formula (3) defines a density, which is called a \( g \)-component finite mixture density. There are two major differences between formulae (2) and (3):

- Formula (2) is the summation of \( n \)-components, while formula (3) is the summation of \( g \)-components and \( g \) could be a small number, such as 2 or 3.
- Formula (2) is based on a single kernel \( K \), while the \( f_j(\cdot) \) of formula (3) could be diverse.

If \( f_j(\cdot) \) are specified parametric density functions \( f_j(\cdot; \theta_j) \), then the density \( f(\cdot) \) is represented as a linear combination of parametric distributions \( f(x) = \sum_{j=1}^{g} \pi_j f_j(x; \theta_j) \) and the parameters \( \theta_j \) and coefficients \( \pi_j \) need to be estimated. Therefore, we need to accomplish three tasks:

- Choose appropriate \( f_j(\cdot; \theta_j), j = 1, \cdots, g. \)
- Estimate \( \theta_j \) and \( \pi_j, j = 1, \cdots, g. \)
- Choose appropriate \( g \).

For the first task, the normal density is perhaps the most popular choice for finite mixture modeling\(^7\). However, strictly speaking, the normal density may not be suitable to your project because the gradient magnitude is always non-negative and is never greater than 1, while any normal density has non-zero probability of having outcomes falling outside of the interval \([0, 1]\).

Thus you might want to consider some continuous density functions that can only take values within \([0, 1]\), such as the uniform distribution and the Beta distribution. For example, Figure 4 are the histogram of a sample of gradient magnitudes and the distribution of the finite mixture

\[
f(x) = \pi_1 f_1(x; \theta_1, \theta_2) + \pi_2 f_2(x; \theta_3, \theta_4),
\]

where \( \pi_1 = 7/8, \pi_2 = 1/8, f_1 \) is a Beta distribution with parameters \( \theta_1 = 1, \theta_2 = 500 \), denoted as \( \text{Beta}(1, 500) \), and \( f_2 \) is a uniform distribution with parameters \( \theta_3 = 0.01, \theta_4 = 0.15 \), denoted as \( U(0.01, 0.15) \). Having chosen the \( f_j(\cdot; \theta_j) \)'s, the log likelihood function of the observations is:

\(^7\)Refer to Section 1.5, 1.6 of [3].
Figure 4: Finite mixture modeling of gradient magnitude.

The log likelihood function is given by:

$$
\log L(\Theta) = \sum_{i=1}^{n} \log f(x_i; \Theta)
= \sum_{i=1}^{n} \log \left\{ \sum_{j=1}^{g} \pi_j f_j(x_i; \theta_j) \right\}.
$$

Then for the second task, you could estimate $\theta_j$ and $\pi_j$ by maximizing likelihood\(^8\) that is, maximizing $\log L(\Theta)$ over the parameter space of $\theta_j$ and $\pi_j$. The solution for this maximization, $\theta^*_j$ and $\pi^*_j$ say, are the maximum likelihood estimates (MLE). Maximizing the likelihood function for a $g$-component finite mixture density is not a trivial task. We usually need to implement iterative algorithms, such as the expectation-maximization (EM) algorithm\(^9\) to find the optimal solution. You might also consider Markov Chain Monte Carlo (MCMC) methods\(^10\), a Bayesian approach, to estimate the parameters.

For the third task, Akaike’s information criterion\(^11\) (AIC) could be adopted for choosing $g$, the number of components. For any given $g$, the parameter estimates $\hat{\Theta}_g$ and the maximized log likelihood, $\log L(\hat{\Theta}_g)$, can be calculated. AIC then selects the value $g^*$ that maximizes

$$
-2 \log L(\hat{\Theta}_g) + 2d_g,
$$

where $d_g$ is the total number of parameters. Based on the above discussions, you may obtain the

\(^8\)Refer to Chapter 2 of \([3]\).
\(^9\)Refer to Section 2.8 of \([3]\).
\(^10\)Refer to Chapter 4 of \([3]\).
\(^11\)Refer to Chapter 6 of \([3]\).
following parametric estimation of $f$:

$$f(x; \Theta) = \sum_{j=1}^{g} \pi_j^* f_j(x; \theta_j^*). \quad (6)$$

Finite mixture modeling can be implemented in MATLAB using the package bayesf Version 2.0\textsuperscript{12}, which is based on the MCMC method. However, the Beta distribution is not available in that package yet. My suggestion is to try this approach with the distributions already available in the package. This should suffice to give you quite a good representation of the density $f$.

In practice, you might incorporate the approaches suggested for Q1 and Q2 together. For instance, the Q1 approach could be used to guide choices (of $f_j$ especially) in the approach for Q2.

### 3.3 Comments

The two methods introduced in the previous two subsections provide precise density estimation at the expense of considerable computational cost. The difficulty of kernel density estimation is the selection of the “optimal” bandwidth, while the difficulty of finite mixture modeling is the parameter estimation and the selection of the number of components. However, the more fundamental problem is that the density estimation may not necessarily help very much with the selection of texture and feature threshold values. At least this is not clear to me.

In fact, edge detection has long been a topic of research. Gonzalez and Woods \textsuperscript{1} give an overall introduction to digital image processing; their Chapter 10 focuses on edge detection. Several threshold value selection methods are discussed and compared, including a simple finite mixture modeling example.

From a statistical learning viewpoint, the goal of edge detection is to distinguish the feature pixels from the texture pixels. However, given any digital image, the pixels are unlabeled in general, and we need to assign all the pixels into feature and texture subsets so that the pixels in the same subset are similar in some sense. This procedure is called cluster analysis, the subsets being the clusters. Now we introduce two nonparametric cluster analysis methods which are based on the distances between the gradient magnitudes at different pixels. In these frameworks, you may take the spatial information of the pixel locations into consideration without assuming independence\textsuperscript{13}.

Suppose $X_i = (x_{i1}, x_{i2}, x_{i3})^T$ denotes the $i$th pixel, where $x_{i1}$ is the gradient magnitude and $(x_{i2}, x_{i3})$ is the pixel coordinate.

\textsuperscript{12}Refer to \textsuperscript{4} for details.
\textsuperscript{13}The gradient magnitude of the pixels are assumed to be independent in section \textsuperscript{3.1} and \textsuperscript{3.2}.
Agglomerative hierarchical Clustering

Agglomerative hierarchical clustering starts with \( n \) clusters, each containing one pixel \( X_i \). At each step, the method merges the two most “similar” (nearest) clusters, so that all pixels are combined into one single cluster \((k\) clusters\) after \( n - 1 \) \((n - k)\) steps. The distance between two clusters is based on the distance between two pixels. For example, one type of distance between pixels \( X_i \) and \( X_j \) is the Euclidean distance:

\[
d_{ij} = \left( \sum_{k=1}^{3} (x_{ik} - x_{jk})^2 \right)^{1/2}.
\]

If each of two clusters contains just one pixel, the distance between the two pixels \( d_{ij} \) is the distance between the two clusters. If either of the clusters contain multiple pixels, the distance between the two clusters also depends on the method of linkage of clusters. For example, single linkage defines the distance between clusters to be the minimum of the pairwise distances between pixels in the two clusters.

One point to notice for calculating the distance between pixels is the measurement scale of the coordinate. If the range of the coordinate is from 1 to 256, then the difference between two pixels’ positions, determined by their coordinates, may influence the “distance” much more than the difference between their gradient magnitudes, because the range of gradient magnitude is just from 0 to 1. To give more weight to gradient magnitude, the range of the coordinate should be small, for example, from 0 to 0.5.

In agglomerative hierarchical clustering, the pixels cannot be reallocated after having been grouped in an early stage, even though the clustering might not seem appropriate later. Consequently, the final result could be partially incorrect, and you may want to repeat the clustering procedure using different types of distances and linkages. If the clustering results obtained from different distances and linkages are roughly consistent, then you may distinguish the feature pixels from the texture pixels according to the agglomerative hierarchical clustering result. Otherwise, the agglomerative hierarchical clustering might not be suitable for this scenario.

In MATLAB, you can perform agglomerative clustering using the `linkage` command in the statistics toolbox. But the computation of agglomerative hierarchical clustering is not a trivial task, particularly when \( n \) is large, for example, \( n = 256 \times 256 \) for each of your images.

---

14 Refer to Chapter 12 of [2].
15 Refer to Section 12.2 of [2] for several definitions and comparison of distance between pixels.
16 Refer to Section 12.3 of [2] for several definitions and comparison of linkage of clusters.
**K-means Clustering**

K-means clustering\(^{17}\) is an iterative algorithm for finding clusters and their centers. Unlike agglomerative hierarchical clustering, the number of clusters \(K\) should be chosen before implementing the method. Given any \(K\) pixels as initial cluster centers, the algorithm iterates the following two steps until convergence.

- All pixels are allocated to the nearest cluster center, based on some distance measure, such as \((7)\):
- Calculate the means of the new clusters and regard them as the new centers.

There are two challenges in using K-means clustering:

- The algorithm is sensitive to initial values.
- The choice of \(K\) is critical to the success of K-means clustering.

For your project, there might exist several large texture areas and small feature areas. By observing the image, you may guess how many clusters there might be (the choice of \(K\)) and select the initial values for the algorithms. In MATLAB, you can perform K-means clustering using the \texttt{kmeans} command in the statistics toolbox.

There are many other clustering methods. In my opinion, however, the primary issues are to decide whether the definitions of feature and texture are to be based on gradient magnitude and spatial information or just on gradient magnitude, and to settle on appropriate definitions of the threshold values. Well-defined objectives are invaluable in selecting suitable statistical analysis tools.

### 4 Conclusions

In this report, we suggest two nonparametric methods for the first statistical question and one parametric method for the second statistical question. Then, regarding the scientific problem itself, we briefly discuss why it could be considered as a clustering problem and introduce two nonparametric clustering methods. Finally, we point out that having clear objectives maybe a key to this project.

\(^{17}\)Refer to Chapter 12 of \([2]\).
References


