A Generalization of Otsu's Method and Minimum Error Thresholding

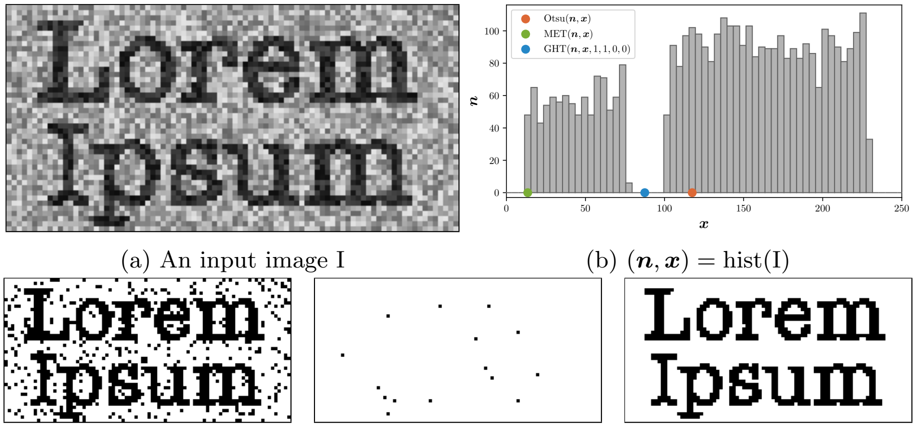
Jonathan T. Barron

Google Research
barron@google.com

Abstract. We present Generalized Histogram Thresholding (GHT), a simple, fast, and effective technique for histogram-based image thresholding. GHT works by performing approximate maximum a posteriori estimation of a mixture of Gaussians with appropriate priors. We demonstrate that GHT subsumes three classic thresholding techniques as special cases: Otsu's method, Minimum Error Thresholding (MET), and weighted percentile thresholding. GHT thereby enables the continuous interpolation between those three algorithms, which allows thresholding accuracy to be improved significantly. GHT also provides a clarifying interpretation of the common practice of coarsening a histogram's bin width during thresholding. We show that GHT outperforms or matches the performance of all algorithms on a recent challenge for handwritten document image binarization (including deep neural networks trained to produce per-pixel binarizations), and can be implemented in a dozen lines of code or as a trivial modification to Otsu's method or MET.

1 Introduction

Histogram-based thresholding is a ubiquitous tool in image processing, medical imaging, and document analysis: The grayscale intensities of an input image are used to compute a histogram, and some algorithm is then applied to that histogram to identify an optimal threshold (corresponding to a bin location along the histogram's x-axis) with which the histogram is to be "split" into two parts. That threshold is then used to binarize the input image, under the assumption that this binarized image will then be used for some downstream task such as classification or segmentation. Thresholding has been the focus of a half-century of research, and is well documented in several survey papers [20, 27]. One might assume that a global thresholding approach has little value in a modern machine learning context: preprocessing an image via binarization discards potentially-valuable information in the input image, and training a neural network to perform binarization is a straightforward alternative. However, training requires significant amounts of training data, and in many contexts (particularly medical imaging) such data may be prohibitively expensive or difficult to obtain. As such, there continues to be value in "automatic" thresholding algorithms that do not require training data, as is evidenced by the active use of these algorithms in the medical imaging literature.



(a) An input image I

(b) $(\boldsymbol{n}, \boldsymbol{x}) = \mathrm{hist}(\mathrm{I})$

(c) $\mathrm{I} > \mathrm{Otsu}(\boldsymbol{n}, \boldsymbol{x})$

(d) $\mathrm{I} > \mathrm{MET}(\boldsymbol{n}, \boldsymbol{x})$

(e) $\mathrm{I} > \mathrm{GHT}(\boldsymbol{n}, \boldsymbol{x}; 1, 1, 0, \cdot)$

Fig. 1: Despite the visually apparent difference between foreground and background pixels in the image (a) and its histogram (b), both Otsu's method (c) and MET (d) fail to correctly binarize this image. GHT (e), which includes both Otsu's method and MET as special cases, produces the expected binarization.

2 Preliminaries

Given an input image I (Figure 1a) that is usually assumed to contain 8-bit or 16-bit quantized grayscale intensity values, a histogram-based automatic thresholding technique first constructs a histogram of that image consisting of a vector of histogram bin locations $\boldsymbol{x}$ and a corresponding vector of histogram counts or weights $\boldsymbol{n}$ (Figure 1b). With this histogram, the algorithm attempts to find some threshold t that separates the two halves of the histogram according to some criteria (*e.g.* maximum likelihood, minimal distortion, *etc.*), and this threshold is used to produce a binarized image $\mathrm{I} > t$ (Figures 1c-1e). Many prior works assume that $x_i = i$, which addresses the common case in which histogram bins exactly correspond to quantized pixel intensities, but we will make no such assumption. This allows our algorithm (and our descriptions and implementations of baseline algorithms) to be applied to arbitrary sets of sorted points, and to histograms whose bins have arbitrary locations. For example, it is equivalent to binarize some quantized image I using a histogram:

$$(\boldsymbol{n}, \boldsymbol{x}) = \mathrm{hist}(\mathrm{I}) \,, \tag{1}$$

or using a vector of sorted values each with a "count" of 1:

$$\boldsymbol{n} = \vec{1} \,, \qquad \boldsymbol{x} = \mathrm{sort}(\mathrm{I}) \,. \tag{2}$$

This equivalence is not used in any result presented here, but is useful when binarizing a set of continuous values.

Most histogram-based thresholding techniques work by considering each possible "split" of the histogram: each value in $\boldsymbol{x}$ is considered as a candidate for t, and two quantities are produced that reflect the surface statistics of $\boldsymbol{n}_{\boldsymbol{x} \leq t}$ and $\boldsymbol{n}_{\boldsymbol{x} > t}$. A critical insight of many classic histogram-based thresholding techniques is that some of these quantities can be computed recursively. For example, the sum of all histogram counts up through some index i ($w_i^{(0)} = \sum_{j=0}^{i} n_j$) need not be recomputed from scratch if that sum has already been previously computed for all the previous histogram element, and can instead just be updated with a single addition ($w_i^{(0)} = w_{i-1}^{(0)} + n_i$). Here we construct pairs of vectors of intermediate values that measure surface statistics of the histogram above and below each "split", to be used by GHT and our baselines:

$$\begin{aligned}
\boldsymbol{w}^{(0)} &= \mathrm{csum}(\boldsymbol{n}) & \boldsymbol{w}^{(1)} &= \mathrm{dsum}(\boldsymbol{n}) \\
\boldsymbol{\pi}^{(0)} &= \boldsymbol{w}^{(0)} / \left\|\boldsymbol{n}\right\|_1 & \boldsymbol{\pi}^{(1)} &= \boldsymbol{w}^{(1)} / \left\|\boldsymbol{n}\right\|_1 = 1 - \boldsymbol{\pi}^{(0)} \\
\boldsymbol{\mu}^{(0)} &= \mathrm{csum}(\boldsymbol{n}\boldsymbol{x}) / \boldsymbol{w}^{(0)} & \boldsymbol{\mu}^{(1)} &= \mathrm{dsum}(\boldsymbol{n}\boldsymbol{x}) / \boldsymbol{w}^{(1)} \\
\boldsymbol{d}^{(0)} &= \mathrm{csum}\big(\boldsymbol{n}\boldsymbol{x}^2\big) - \boldsymbol{w}^{(0)} \left(\boldsymbol{\mu}^{(0)}\right)^2 & \boldsymbol{d}^{(1)} &= \mathrm{dsum}\big(\boldsymbol{n}\boldsymbol{x}^2\big) - \boldsymbol{w}^{(1)} \left(\boldsymbol{\mu}^{(1)}\right)^2
\end{aligned} \tag{3}$$

All multiplications, divisions, and exponentiations are element-wise, and csum() and dsum() are cumulative and "reverse cumulative" sums respectively. $\left\|\boldsymbol{n}\right\|_1$ is the sum of all histogram counts in $\boldsymbol{n}$, while $\boldsymbol{w}^{(0)}$ and $\boldsymbol{w}^{(1)}$ are the sums of histogram counts in $\boldsymbol{n}$ below and above each split of the histogram, respectively. Similarly, each $\boldsymbol{\pi}^{(k)}$ respectively represent normalized histogram counts (or mixture weights) above and below each split. Each $\boldsymbol{\mu}^{(k)}$ and $\boldsymbol{d}^{(k)}$ represent the mean and distortion of all elements of $\boldsymbol{x}$ below and above each split, weighted by their histogram counts $\boldsymbol{n}$. Here "distortion" is used in the context of k-means, where it refers to the sum of squared distances of a set of points to their mean. The computation of $\boldsymbol{d}^{(k)}$ follows from three observations: First, $\mathrm{csum}(\cdot)$ or $\mathrm{dsum}(\cdot)$ can be used (by weighting by $\boldsymbol{n}$ and normalizing by $\boldsymbol{w}$) to compute a vector of expected values. Second, the sample variance of a quantity is a function of the expectation of its values and its values squared ($\mathrm{Var}(X) = \mathrm{E}[X^2] - \mathrm{E}[X]^2$). Third, the sample variance of a set of points is proportional to the total distortion of those points.

Formally, $\mathrm{csum}(\cdot)_i$ is the inclusive sum of every vector element at or before index i, and $\mathrm{dsum}(\cdot)_i$ is the exclusive sum of everything after index i:

$$\begin{aligned}
\mathrm{csum}(\boldsymbol{n})_i &= \sum_{j=0}^{i} n_j = \mathrm{csum}(\boldsymbol{n})_{i-1} + n_i \,, \\
\mathrm{dsum}(\boldsymbol{n})_i &= \sum_{j=i+1}^{\mathrm{len}(\boldsymbol{n})\text{-}1} n_j = \mathrm{dsum}(\boldsymbol{n})_{i+1} + n_{i+1} \,.
\end{aligned} \tag{4}$$

Note that the outputs of these cumulative and decremental sums have a length that is one less than that of $\boldsymbol{x}$ and $\boldsymbol{n}$, as these values measure "splits" of the data, and we only consider splits that contain at least one histogram element.

The subscripts in our vectors of statistics correspond to a threshold *after* each index, which means that binarization should be performed by a greater-than comparison with the returned threshold ($\boldsymbol{x} > t$). In practice, $\operatorname{dsum}(\boldsymbol{n})$ can be computed using just the cumulative and total sums of $\boldsymbol{n}$:

$$\operatorname{dsum}(\boldsymbol{n}) = \|\boldsymbol{n}\|_1 - \operatorname{csum}(\boldsymbol{n}) \ . \tag{5}$$

As such, these quantities (and all other vector quantities that will be described later) can be computed efficiently with just one or two passes over the histogram.

3 Algorithm

Our Generalized Histogram Thresholding (GHT) algorithm is motivated by a straightforward Bayesian treatment of histogram thresholding. We assume that each pixel in the input image is generated from a mixture of two probability distributions corresponding to intensities below and above some threshold, and we maximize the joint probability of all pixels (which are assumed to be IID):

$$\prod_{x,y} \left(p^{(0)}(\mathrm{I}_{x,y}) + p^{(1)}(\mathrm{I}_{x,y}) \right) \ . \tag{6}$$

Our probability distributions are parameterized as:

$$p^{(k)}(x) = \pi^{(k)} \mathcal{N}\left(\mathrm{I}_{x,y} \,|\, \mu^{(k)}, \sigma^{(k)} \right) \chi^2_{\mathrm{SI}}\left(\sigma^{(k)} \,|\, \pi^{(k)}\nu, \tau^2 \right) \mathrm{Beta}\left(\pi^{(k)} \,|\, \kappa, \omega \right) \ . \tag{7}$$

This is similar to a straightforward mixture of Gaussians: we have a mixture probability ($\pi^{(0)}$, $\pi^{(1)} = 1 - \pi^{(0)}$), two means ($\mu^{(0)}$, $\mu^{(1)}$), and two standard deviations ($\sigma^{(0)}$, $\sigma^{(1)}$). But unlike a standard mixture of Gaussians, we place conjugate priors on the model parameters. We assume each standard deviation is the mode (*i.e.*, the "updated" τ value) of a scaled inverse chi-squared distribution, which is the conjugate prior of a Gaussian with a known mean.

$$\chi^2_{\mathrm{SI}}(x \,|\, \nu, \tau^2) = \frac{(\tau^2\nu/2)^{\nu/2}}{\Gamma(\nu/2)} \, \frac{\exp\left(\frac{-\nu\tau^2}{2x} \right)}{x^{1+\nu/2}} \ . \tag{8}$$

And we assume the mixture probability is drawn from a beta distribution (the conjugate prior of a Bernoulli distribution).

$$\mathrm{Beta}(x \,|\, \kappa, \omega) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha, \beta)} \, , \quad \alpha = \kappa\omega + 1 \, , \quad \beta = \kappa(1-\omega) + 1 \, . \tag{9}$$

Where B is the beta function. Note that our beta distribution uses a slightly unconventional parameterization in terms of its concentration κ and mode ω, which will be explained later. Also note that in Equation 7 the degrees-of-freedom parameter ν of the scaled inverse chi-squared distribution is rescaled according to the mixture probability, as will also be discussed later. This Bayesian mixture of Gaussians has four hyperparameters that will determine the behavior of GHT: $\nu \geq 0, \tau \geq 0, \kappa \geq 0, \omega \in [0, 1]$.

From this probabilistic framework we can derive our histogram thresholding algorithm. Let us rewrite Equation 6 to be in terms of bin locations and counts:

$$\prod_i \left(p^{(0)}(x_i) + p^{(1)}(x_i) \right)^{n_i} . \tag{10}$$

Taking its logarithm gives us this log-likelihood:

$$\sum_i n_i \log\left(p^{(0)}(x_i) + p^{(1)}(x_i) \right) . \tag{11}$$

Now we make a simplifying model assumption that enables a single-pass histogram thresholding algorithm. We will assume that, at each potential "split" of the data, each of the two splits of the histogram is generated entirely by one of the two Gaussians in our mixture. More formally, we consider all possible sorted assignments of each histogram bin to either of the two Gaussians and maximize the expected complete log-likelihood (ECLL) of that assignment. Jensen's inequality ensures that this ECLL is a lower bound on the marginal likelihood of the data that we would actually like to maximize. This is the same basic approach (though not the same justification) as in "Minimum Error Thresholding" (MET) [10] and other similar approaches, but where our technique differs is in how the likelihood and the parameters of these Gaussians are estimated. For each split of the data, we assume the posterior distribution of the latent variables determining the ownership of each histogram bin by each Gaussian (the missing "z" values) are wholly assigned to one of the two Gaussians according to the split. This gives us the following ECLL as a function of the assumed split location's array index i:

$$\mathcal{L}_i = \sum_{j=0}^{i} n_j \log\left(p^{(0)}(x_j) \right) + \sum_{j=i+1}^{\mathrm{len}(\boldsymbol{n})\text{-}1} n_j \log\left(p^{(1)}(x_j) \right) . \tag{12}$$

Our proposed algorithm is to simply iterate over all possible values of i (of which there are 255 in our experiments) and return the value that maximizes $\mathcal{L}_i$. As such, our algorithm can be viewed as a kind of inverted expectation-maximization in which the sweep over threshold resembles an M-step and the assignment of latent variables according to that threshold resembles an E-step [21]. Our GHT algorithm will be defined as:

$$\mathrm{GHT}(\boldsymbol{x}, \boldsymbol{n}; \nu, \tau, \kappa, \omega) = x_{\arg\max_i (\mathcal{L}_i)} . \tag{13}$$

This definition of GHT is a bit unwieldy, but using the preliminary math of Equation 3 and ignoring global shifts and scales of $\mathcal{L}_i$ that do not affect the argmax it can be simplified substantially:

$$
\begin{aligned}
\boldsymbol{v}^{(0)} &= \frac{\boldsymbol{\pi}^{(0)}\nu\tau^2 + \boldsymbol{d}^{(0)}}{\boldsymbol{\pi}^{(0)}\nu + \boldsymbol{w}^{(0)}} \qquad\qquad \boldsymbol{v}^{(1)} = \frac{\boldsymbol{\pi}^{(1)}\nu\tau^2 + \boldsymbol{d}^{(1)}}{\boldsymbol{\pi}^{(1)}\nu + \boldsymbol{w}^{(1)}} \\
\boldsymbol{f}^{(0)} &= -\frac{\boldsymbol{d}^{(0)}}{\boldsymbol{v}^{(0)}} - \boldsymbol{w}^{(0)}\log\left(\boldsymbol{v}^{(0)}\right) + 2\left(\boldsymbol{w}^{(0)} + \kappa\omega\right)\log\left(\boldsymbol{w}^{(0)}\right) \\
\boldsymbol{f}^{(1)} &= -\frac{\boldsymbol{d}^{(1)}}{\boldsymbol{v}^{(1)}} - \boldsymbol{w}^{(1)}\log\left(\boldsymbol{v}^{(1)}\right) + 2\left(\boldsymbol{w}^{(1)} + \kappa(1-\omega)\right)\log\left(\boldsymbol{w}^{(1)}\right) \\
\mathrm{GHT}(\boldsymbol{x}, \boldsymbol{n}; \nu, \tau, \kappa, \omega) &= x_{\arg\max_i\left(\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}\right)}
\end{aligned}
\tag{14}
$$

Each $\boldsymbol{f}^{(k)}$ can be thought of as a log-likelihood of the data for each Gaussian at each split, and each $\boldsymbol{v}^{(k)}$ can be thought of as an estimate of the variance of each Gaussian at each split. GHT simply scores each split of the histogram and returns the value of $\boldsymbol{x}$ that maximizes the $\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}$ score (in the case of ties we return the mean of all $\boldsymbol{x}$ values that maximize that score). Because this only requires element-wise computation using the previously-described quantities in Equation 3, GHT requires just a single linear scan over the histogram.

The definition of each $\boldsymbol{v}^{(k)}$ follows from our choice to model each Gaussian's variance with a scaled inverse chi-squared distribution: the ν and τ^2 hyperparameters of the scaled inverse chi-squared distribution are updated according to the scaled sample variance (shown here as distortion $\boldsymbol{d}^{(k)}$) to produce an updated posterior hyperparameter that we use as the Gaussian's estimated variance $\boldsymbol{v}^{(k)}$. Using a conjugate prior update instead of the sample variance has little effect when the subset of $\boldsymbol{n}$ being processed has significant mass (*i.e.*, when $\boldsymbol{w}^{(k)}$ is large) but has a notable effect when a Gaussian is being fit to a sparsely populated subset of the histogram. Note that $\boldsymbol{v}^{(k)}$ slightly deviates from the traditional conjugate prior update, which would omit the $\boldsymbol{\pi}^{(k)}$ scaling on ν in the numerator and denominator (according to our decision to rescale each degrees-of-freedom parameter by each mixture probability). This additional scaling counteracts the fact that, at different splits of our histogram, the total histogram count on either side of the split will vary. A "correct" update would assume a constant number of degrees-of-freedom regardless of where the split is being performed, which would result in the conjugate prior here having a substantially different effect at each split, thereby making the resulting $\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}$ scores not comparable to each other and making the argmax over these scores not meaningful.

The beta distribution used in our probabilistic formulation (parameterized by a concentration κ and a mode ω) has a similar effect as the "anisotropy coefficients" used by other models [8, 19]: setting ω near 0 biases the algorithm towards thresholding near the start of the histogram, and setting it near 1 biases the algorithm towards thresholding near the end of the histogram. The concentration κ parameter determines the strength of this effect, and setting $\kappa = 0$ disables this regularizer entirely. Note that our parameterization of concentration κ differs slightly from the traditional formulation, where setting $\kappa = 1$ has no effect and setting $\kappa < 1$ moves the predicted threshold *away* from the mode. We instead assume the practitioner will only want to consider biasing the algorithm's threshold *towards* the mode, and parameterize the distribution accordingly. These hyperparameters allow for GHT to be biased towards outputs where a particular fraction of the image lies above or below the output threshold. For example, in an OCR / digit recognition context, it may be useful to inform the algorithm that the majority of the image is expected to not be ink, so as to prevent output thresholds that erroneously separate the non-ink background of the page into two halves.

Note that, unlike many histogram based thresholding algorithms, we do not require or assume that our histogram counts $\boldsymbol{n}$ are normalized. In all experiments, we use raw counts as our histogram values. This is important to our approach, as it means that the behavior of the magnitude of $\boldsymbol{n}$ and the behavior induced by varying our ν and κ hyperparameters is consistent:

$$\forall a \in \mathbb{R}_{>0} \;\; \mathrm{GHT}(\boldsymbol{x}, a\boldsymbol{n}; a\nu, \tau, a\kappa, \omega) = \mathrm{GHT}(\boldsymbol{n}, \boldsymbol{x}; \nu, \tau, \kappa, \omega)\,. \tag{15}$$

This means that, for example, doubling the number of pixels in an image is equivalent to halving the values of the two hyperparameters that control the "strength" of our two Bayesian model components.

Additionally, GHT's output and hyperparameters (excluding τ, which exists solely to encode absolute scale) are invariant to positive affine transformations of the histogram bin centers:

$$\forall a \in \mathbb{R}_{>0} \; \forall b \in \mathbb{R} \;\; \mathrm{GHT}(a\boldsymbol{x} + b, \boldsymbol{n}; \nu, a\tau, \kappa, \omega) = \mathrm{GHT}(\boldsymbol{n}, \boldsymbol{x}; \nu, \tau, \kappa, \omega)\,. \tag{16}$$

One could extend GHT to be sensitive to absolute intensity values by including conjugate priors over the means of the Gaussians, but we found little experimental value in that level of control. This is likely because the dataset we evaluate on does not have a standardized notion of brightness or exposure (as is often the case for binarization tasks).

In the following subsections we demonstrate how GHT generalizes three standard approaches to histogram thresholding by including them as special cases of our hyperparameter settings: Minimum Error Thresholding [10], Otsu's method [17], and weighted percentile thresholding.

3.1 Special Case: Minimum Error Thresholding

Because Minimum Error Thresholding (MET) [10], like our approach, works by maximizing the expected complete log-likelihood of the input histogram under a mixture of two Gaussians, it is straightforward to express it using the already-defined quantities in Equation 3:

$$\begin{gathered}\boldsymbol{\ell} = 1 + \boldsymbol{w}^{(0)} \log\left(\frac{\boldsymbol{d}^{(0)}}{\boldsymbol{w}^{(0)}}\right) + \boldsymbol{w}^{(1)} \log\left(\frac{\boldsymbol{d}^{(1)}}{\boldsymbol{w}^{(1)}}\right) - 2\left(\boldsymbol{w}^{(0)} \log\left(\boldsymbol{w}^{(0)}\right) + \boldsymbol{w}^{(1)} \log\left(\boldsymbol{w}^{(1)}\right)\right) \\ \mathrm{MET}(\boldsymbol{x}, \boldsymbol{n}) = x_{\arg\min_i(\boldsymbol{\ell})}\end{gathered} \tag{17}$$

Because GHT is simply a Bayesian treatment of this same process, it includes MET as a special case. If we set $\nu = 0$ and $\kappa = 0$ (under these conditions the values of τ and ω are irrelevant) in Equation 14, we see that $\boldsymbol{v}^{(k)}$ reduces to $\boldsymbol{d}^{(k)}/\boldsymbol{w}^{(k)}$, which causes each score to simplify dramatically:

$$\nu = \kappa = 0 \implies \boldsymbol{f}^{(k)} = -\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(k)} \log\left(\frac{\boldsymbol{d}^{(k)}}{\boldsymbol{w}^{(k)}}\right) + 2\boldsymbol{w}^{(k)} \log\left(\boldsymbol{w}^{(k)}\right) . \tag{18}$$

Because $\boldsymbol{w}^{(0)} + \boldsymbol{w} = \|\boldsymbol{n}\|_1$, the score maximized by GHT can be simplified:

$$\nu = \kappa = 0 \implies \boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)} = 1 - \|\boldsymbol{n}\|_1 - \boldsymbol{\ell} . \tag{19}$$

We see that, for these particular hyperparameter settings, the total score $\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}$ that GHT maximizes is simply an affine transformation (with a negative scale) of the score $\boldsymbol{\ell}$ that MET maximizes. This means that the index that optimizes either quantity is identical, and so the algorithms (under these hyperparameter settings) are equivalent:

$$\mathrm{MET}(\boldsymbol{n}, \boldsymbol{x}) = \mathrm{GHT}(\boldsymbol{n}, \boldsymbol{x}; 0, \cdot, 0, \cdot) . \tag{20}$$

3.2 Special Case: Otsu's Method

Otsu's method for histogram thresholding [17] works by directly maximizing inter-class variance for the two sides of the split histogram, which is equivalent to indirectly minimizing the total intra-class variance of those two sides [17]. Otsu's method can also be expressed using the quantities in Equation 3:

$$\boldsymbol{o} = \boldsymbol{w}^{(0)}\boldsymbol{w}^{(1)} \left(\boldsymbol{\mu}^{(0)} - \boldsymbol{\mu}^{(1)}\right)^2 , \quad \mathrm{Otsu}(\boldsymbol{n}, \boldsymbol{x}) = x_{\arg\max_i(\boldsymbol{o})} . \tag{21}$$

To clarify the connection between Otsu's method and GHT, we rewrite the score $\boldsymbol{o}$ of Otsu's method as an explicit sum of intra-class variances:

$$\boldsymbol{o} = \|\boldsymbol{n}\|_1 \left\langle \boldsymbol{n}, \boldsymbol{x}^2 \right\rangle - \left\langle \boldsymbol{n}, \boldsymbol{x} \right\rangle^2 - \|\boldsymbol{n}\|_1 \left(\boldsymbol{d}^{(0)} + \boldsymbol{d}^{(1)}\right) . \tag{22}$$

Now let us take the limit of the $\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}$ score that is maximized by GHT as ν approaches infinity:

$$\lim_{\nu \to \infty} \boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)} = -\frac{\boldsymbol{d}^{(0)} + \boldsymbol{d}^{(1)}}{\tau^2} + 2\boldsymbol{w}^{(0)} \log\left(\frac{\boldsymbol{w}^{(0)}}{\tau}\right) + 2\boldsymbol{w}^{(1)} \log\left(\frac{\boldsymbol{w}^{(1)}}{\tau}\right) . \tag{23}$$

With this we can set the τ hyperparameter to be infinitesimally close to zero, in which case the score in Equation 23 becomes dominated by its first term. Therefore, as ν approaches infinity and τ approaches zero, the score maximized by GHT is proportional to the (negative) score that is indirectly minimized by Otsu's method:

$$\nu \gg 0, \, \tau = \epsilon \implies \boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)} \approx -\frac{\boldsymbol{d}^{(0)} + \boldsymbol{d}^{(1)}}{\tau^2} , \tag{24}$$

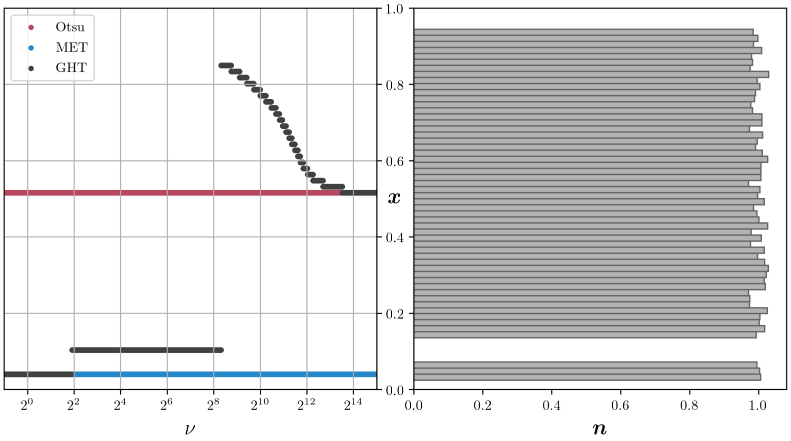


Fig. 2: On the right we have a toy input histogram (shown rotated), and on the left we have a plot showing the predicted threshold (y-axis) of GHT ($\tau = 0.01$, $\kappa = 0$, shown in black) relative to the hyperparameter value ν (x-axis). Note that the y-axis is shared across the two plots. Both MET and Otsu's method predict incorrect thresholds (shown in red and blue, respectively) despite the evident gap in the input histogram. GHT, which includes MET and Otsu's method as special cases ($\nu = 0$ and $\nu = \infty$, respectively) allows us to interpolate between these two extremes by varying ν, and produces the correct threshold for a wide range of values $\nu \in [\sim 2^2, \sim 2^8]$.

where ϵ is a small positive number. This observation fits naturally with the well-understood equivalence of k-means (which works by minimizing distortion) and the asymptotic behavior of maximum likelihood under a mixture of Gaussians model as the variance of each Gaussian approaches zero [12]. Given this equivalence between the scores that are optimized by MET and GHT (subject to these specific hyperparameters) we can conclude that Otsu's method is a special case of GHT:

$$\mathrm{Otsu}(\boldsymbol{n}, \boldsymbol{x}) = \lim_{(\nu,\tau)\to(\infty,0)} \mathrm{GHT}(\boldsymbol{n}, \boldsymbol{x}; \nu, \tau, 0, \cdot)\,. \tag{25}$$

We have demonstrated that GHT is a generalization of Otsu's method (when $\nu = \infty$) and MET (when $\nu = 0$), but for this observation to be of any practical importance, there must be value in GHT when ν is set to some value in between those two extremes. To demonstrate this, in Figure 2 we visualize the effect of ν on GHT's behavior. Similar to the example shown in Figure 1, we see that both Otsu's method and MET both perform poorly when faced with a simple histogram that contains two separate and uneven uniform distributions: MET prefers to express all histogram elements using a single Gaussian by selecting a threshold at one end of the histogram, and Otsu's method selects a threshold that splits the larger of the two modes in half instead of splitting the two modes apart. But by varying ν from 0 to ∞ (while also setting τ to a small value) we see that a wide range of values of ν results in GHT correctly separating the two modes of this histogram. Additionally, we see that the range of hyperparameters that reproduces this behavior is wide, demonstrating the robustness of performance with respect to the specific value of this parameter.

3.3 Relationship to Histogram Bin Width

Most histogram thresholding techniques (including GHT as it is used in this paper) construct a histogram with as many bins as there are pixel intensities — an image of 8-bit integers results in a histogram with 256 bins, each with a bin width of 1. However, the performance of histogram thresholding techniques often depends on the selection of the input histogram's bin width, with a coarse binning resulting in more "stable" performance and a fine binning resulting in a more precisely localized threshold. Many histogram thresholding techniques therefore vary the bin width of their histograms, either by treating it as a user-defined hyperparameter [2] or by determining it automatically [7, 16] using classical statistical techniques [3, 24]. Similarly, one could instead construct a fine histogram that is then convolved with a Gaussian before thresholding [13], which is equivalent to constructing the histogram using a Parzen window. Blurring or coarsening histograms both serve the same purpose of filtering out high frequency variation in the histogram, as coarsening a histogram is equivalent blurring (with a box filter) and then decimating a fine histogram.

This practice of varying histogram bins or blurring a histogram can be viewed through the lens of GHT. Consider a histogram $(\boldsymbol{n}, \boldsymbol{x})$, and let us assume (contrary to the equivalence described in Equation 2) that the spacing between the bins centers in $\boldsymbol{x}$ is constant. Consider a discrete Gaussian blur filter $\boldsymbol{f}(\sigma)$ with a standard deviation of σ, where that filter is normalized ($\|\boldsymbol{f}(\sigma)\|_1 = 1$). Let us consider $\boldsymbol{n} * \boldsymbol{f}(\sigma)$, which is the convolution of histogram counts with our Gaussian blur (assuming a "full" convolution of $\boldsymbol{n}$ and $\boldsymbol{x}$ with zero boundary conditions). This significantly affects the sample variance of the histogram v, which (because convolution is linear) is:

$$v = \frac{\sum_i (\boldsymbol{n} * \boldsymbol{f}(\sigma))_i (x_i - \mu)^2}{\|\boldsymbol{n}\|_1} = \frac{\|\boldsymbol{n}\|_1 \sigma^2 + d}{\|\boldsymbol{n}\|_1}, \qquad d = \sum_i n_i (x_i - \mu)^2 . \tag{26}$$

We use v and d to describe sample variance and distortion as before, though here these values are scalars as we compute a single estimate of both for the entire histogram. This sample variance v resembles the definition of $\boldsymbol{v}^{(k)}$ in Equation 14, and we can make the two identical in the limit by setting GHT's hyperparameters to $\nu = \epsilon \|\boldsymbol{n}\|_1$ and $\tau = \sigma/\sqrt{\epsilon}$, where ϵ is a small positive number. With this equivalence we see that GHT's τ hyperparameter serves a similar purpose as coarsening/blurring the input histogram — blurring the input histogram with a Gaussian filter with standard deviation of σ is roughly equivalent to setting GHT's τ parameter proportionally to σ. Or more formally:

$$\mathrm{MET}(\boldsymbol{n}, \boldsymbol{x} * \boldsymbol{f}(\sigma)) \approx \mathrm{GHT}\big(\boldsymbol{n}, \boldsymbol{x}; \|\boldsymbol{n}\|_1 \epsilon, \cdot, \sigma/\sqrt{\epsilon}, \cdot\big) . \tag{27}$$

Unlike the other algorithmic equivalences stated in this work, this equivalence is approximate: changing σ is not exactly equivalent to blurring the input histogram. Instead, it is equivalent to, at each split of $\boldsymbol{n}$, blurring the left and right halves of $\boldsymbol{n}$ independently — the histogram cannot be blurred "across" the split. Still, we observed that these two approaches produce identical threshold results in the vast majority of instances.

This relationship between GHT's τ hyperparameter and the common practice of varying a histogram's bin width provides a theoretical grounding of both GHT and other work: GHT may perform well because varying τ implicitly blurs the input histogram, or prior techniques based on variable bin widths may work well because they are implicitly imposing a conjugate prior on sample variance. This relationship may also shed some light on other prior work exploring the limitations of MET thresholding due to sample variance estimates not being properly "blurred" across the splitting threshold [1].

3.4 Special Case: Weighted Percentile

A simple approach for binarizing some continuous signal is to use the nth percentile of the input data as the threshold, such as its median. This simple baseline is also expressible as a special case of GHT. If we set κ to a large value we see that the score $\boldsymbol{f}^{(0)} + \boldsymbol{f}^{(1)}$ being maximized by GHT becomes dominated by the terms of the score related to $\log(\boldsymbol{\pi}^{(k)})$:

$$\kappa \gg 0 \implies \tfrac{\boldsymbol{f}^{(0)}+\boldsymbol{f}^{(1)}}{2} \approx \left(\|\boldsymbol{n}\|_1 + \kappa\right) \log\left(\|\boldsymbol{n}\|_1\right) + \kappa \left(\omega \log\!\left(\boldsymbol{\pi}^{(0)}\right) + (1-\omega) \log\!\left(1 - \boldsymbol{\pi}^{(0)}\right)\right) . \tag{28}$$

By setting its derivative to zero we see that this score is maximized at the split location i where $\pi_i^{(0)}$ is as close as possible to ω. This condition is also satisfied by the (100ω)th percentile of the histogrammed data in I, or equivalently, by taking the (100ω)th weighted percentile of $\boldsymbol{x}$ (where each bin is weighted by $\boldsymbol{n}$). From this we can conclude that the weighted percentile of the histogram (or equivalently, the percentile of the image) is a special case of GHT:

$$\operatorname{wprctile}(\boldsymbol{x}, \boldsymbol{n}, 100\omega) = \lim_{\kappa \to \infty} \operatorname{GHT}(\boldsymbol{n}, \boldsymbol{x}, 0, \cdot, \kappa, \omega) . \tag{29}$$

This also follows straightforwardly from how GHT uses a beta distribution: if this beta distribution's concentration κ is set to a large value, the mode of the posterior distribution will approach the mode of the prior ω.

In Figure 3 we demonstrate the value of this beta distribution regularization. We see that Otsu's method and MET behave unpredictably when given multimodal inputs, as the metrics they optimize have no reason to prefer a split that groups more of the histogram modes above the threshold versus one that groups more of the modes below the threshold. This can be circumvented by using the weighted percentile of the histogram as a threshold, but this requires that the target percentile ω be precisely specified beforehand: For any particular input histogram, it is possible to identify a value for ω that produces the desired threshold, but this percentile target will not generalize to another histogram

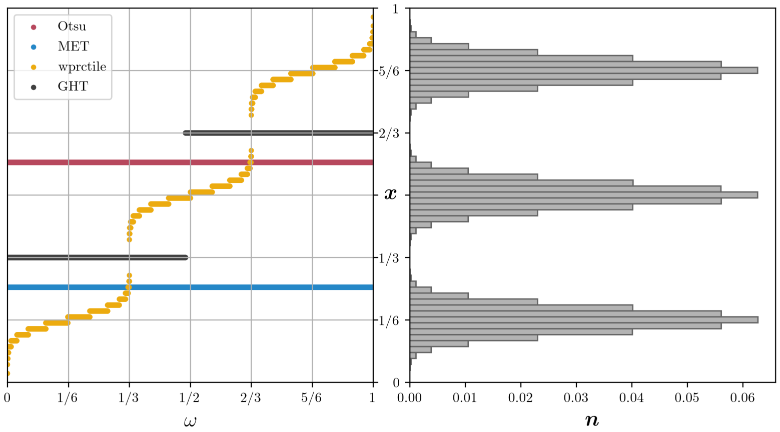


Fig. 3: On the right we have a toy input histogram (shown rotated), and on the left we have a plot showing the predicted threshold (y-axis) of GHT ($\nu = 200$, $\tau = 0.01$, $\kappa = 0.1$, shown in black) relative to the hyperparameter value ω (x-axis). Note that the y-axis is shared across the two plots. Both MET and Otsu's method (shown in red and blue, respectively) predict thresholds that separate two of the three modes from each other arbitrarily. Computing the weighted percentile of the histogram (shown in yellow) can produce *any* threshold, but reproducing the desired split requires exactly specifying the correct value of ω, which likely differs across inputs. GHT (which includes these three other algorithms as special cases) produces thresholds that accurately separate the three modes, and the location of that threshold is robust to the precise value of ω and depends only on it being below or above $^1/_2$.

with the same overall appearance but with a slightly different distribution of mass across that desired threshold level. GHT, in contrast, is able to produce sensible thresholds for all valid values of ω — when the value is less than $^1/_2$ the recovered threshold precisely separates the first mode from the latter two, and when the value is greater than $^1/_2$ it precisely separates the first two modes from the latter one. As such, we see that this model component provides an effective means to bias GHT towards specific kinds of thresholds, while still causing it to respect the relative spread of histogram bin counts.

4 Experiments

To evaluate GHT, we use the 2016 Handwritten Document Image Binarization Contest (H-DIBCO) challenge [18]. This challenge consists of images of handwritten documents alongside ground-truth segmentations of those documents, and algorithms are evaluated by how well their binarizations match the ground truth. We use the 2016 challenge because this is the most recent instantiation of this challenge where a global thresholding algorithm (one that selects a single

Algorithm	ν	τ	κ	ω	$F_1 \times 100 \uparrow$	PSNR ↑	DRD [15] ↓
GHT (MET Case)	-	-	-	-	60.40 ± 20.65	11.21 ± 3.50	45.32 ± 41.35
Kefali *et al.* [18, 22]					76.10 ± 13.81	15.35 ± 3.19	9.16 ± 4.87
Raza [18]					76.28 ± 9.71	14.21 ± 2.21	15.14 ± 9.42
GHT (wprctile Case)	-	-	10^{60}	$2^{-3.75}$	76.77 ± 14.50	15.44 ± 3.40	12.91 ± 17.19
Sauvola [18, 23]					82.52 ± 9.65	16.42 ± 2.87	7.49 ± 3.97
Khan & Mollah [18]					84.32 ± 6.81	16.59 ± 2.99	6.94 ± 3.33
Tensmeyer & Martinez [14, 25, 26]					85.57 ± 6.75	17.50 ± 3.43	5.00 ± 2.60
de Almeida & de Mello [18]					86.24 ± 5.79	17.52 ± 3.42	5.25 ± 2.88
Otsu's Method [17, 18]					86.61 ± 7.26	17.80 ± 4.51	5.56 ± 4.44
GHT (No wprctile)	$2^{50.5}$	$2^{0.125}$	-	-	87.16 ± 6.32	17.97 ± 4.00	5.04 ± 3.17
GHT (Otsu Case)	10^{60}	10^{-15}	-	-	87.19 ± 6.28	17.97 ± 4.01	5.04 ± 3.16
Otsu's Method (Our Impl.) [17]					87.19 ± 6.28	17.97 ± 4.01	5.04 ± 3.16
Nafchi *et al.* - 1 [18, 28]					87.60 ± 4.85	17.86 ± 3.51	4.51 ± 1.62
Kligler [6, 9, 11]					87.61 ± 6.99	18.11 ± 4.27	5.21 ± 5.28
Roe & de Mello [18]					87.97 ± 5.17	18.00 ± 3.68	4.49 ± 2.65
Nafchi *et al.* - 2 [18, 28]					88.11 ± 4.63	18.00 ± 3.41	4.38 ± 1.65
Hassaïne *et al.* - 1 [5, 18]					88.22 ± 4.80	18.22 ± 3.41	4.01 ± 1.49
Hassaïne *et al.* - 2 [4, 18]					88.47 ± 4.45	18.29 ± 3.35	3.93 ± 1.37
Hassaïne *et al.* - 3 [4, 5, 18]					88.72 ± 4.68	18.45 ± 3.41	3.86 ± 1.57
GHT	$2^{29.5}$	$2^{3.125}$	$2^{22.25}$	$2^{-3.25}$	88.77 ± 4.99	18.55 ± 3.46	3.99 ± 1.77
Oracle Global Threshold					90.69 ± 3.92	19.17 ± 3.29	3.57 ± 1.84

Table 1: Results on the 2016 Handwritten Document Image Binarization Contest (H-DIBCO) challenge [18], in terms of the arithmetic mean and standard deviation of F1 score (multiplied by 100 to match the conventions used by [18]), PSNR, and Distance-Reciprocal Distortion (DRD) [15]. The scores for all baseline algorithms are taken from [18]. Our GHT algorithm and it's ablations and special cases (some of which correspond to other algorithms) are indicated in bold. The settings of the four hyperparameters governing GHT's behavior are also provided.

threshold for use on all pixels) can be competitive: Later versions of this challenge use input images that contain content outside of the page being binarized which, due to the background of the images often being black, renders any algorithm that produces a single global threshold for binarization ineffective. GHT's four hyperparameters were tuned using coordinate descent to maximize the arithmetic mean of F_1 scores (a metric used by the challenge) over a small training dataset (or perhaps more accurately, a "tuning" dataset). For training data we use the 8 hand-written images from the 2013 version of the same challenge, which is consistent with the tuning procedure advocated by the challenge (any training data from past challenges may be used, though we found it sufficient to use only relevant data from the most recent challenge). Input images are processed by taking the per-pixel max() across color channels to produce a grayscale image, computing a single 256-bin histogram of the resulting values, applying GHT to that histogram to produce a single global threshold, and binarizing the grayscale input image according to that threshold.

In Table 1 we report GHT's performance against all algorithms evaluated in the challenge [18], using the F_1, PSNR, and Distance-Reciprocal Distortion [15] metrics used by the challenge [18]. We see that GHT produces the lowest-error of all entrants to the H-DIBCO 2016 challenge for two of the three metrics used. GHT outperforms Otsu's method and MET by a significant margin, and also outperforms or matches the performance of significantly more complicated techniques that rely on large neural networks with thousands or millions of learned parameters, and that produce an arbitrary per-pixel binary mask instead of GHT's single global threshold. This is despite GHT's simplicity: it requires roughly a dozen lines of code to implement (far fewer if an implementation of MET or Otsu's method is available), requires no training, and has only four parameters that were tuned on a small dataset of only 8 training images.

We augment Table 1 with some additional results not present in [18]. We present an "Oracle Global Threshold" algorithm, which shows the performance of an oracle that selects the best-performing (according to F_1) global threshold individually for each test image. We present the special cases of GHT that correspond to MET and Otsu's method, to verify that the special case corresponding to Otsu's method performs identically to our own implementation of Otsu's method and nearly identically to the implementation presented in [18]. We present additional ablations of GHT to demonstrate the contribution of each algorithm component: The "wprctile Case" model sets κ to an extremely large value and tunes ω on our training set, and performs poorly. The "No wprctile" model sets $\kappa = 0$ and exhibits worse performance than complete GHT.

See the supplement for reference implementations of GHT, as well as reference implementations of the other algorithms that were demonstrated to be special cases of GHT in Sections 3.1-3.4.

5 Conclusion

We have presented Generalized Histogram Thresholding, a simple, fast, and effective technique for histogram-based image thresholding. GHT includes several classic techniques as special cases (Otsu's method, Minimum Error Thresholding, and weighted percentile thresholding) and thereby serves as a unifying framework for those discrete algorithms, in addition to providing a theoretical grounding for the common practice of varying a histogram's bin width when thresholding. GHT is exceedingly simple: it can be implemented in just a dozen lines of python (far fewer if an implementation of MET or Otsu's method is available) and has just four tunable hyperparameters. Because it requires just a single sweep over a histogram of image intensities, GHT is fast to evaluate: its computational complexity is comparable to Otsu's method. Despite its simplicity and speed (and its inherent limitations as a *global* thresholding algorithm) GHT outperforms or matches the performance of all submitted techniques on the 2016 H-DIBCO image binarization challenge — including deep neural networks that have been trained to produce arbitrary per-pixel binarizations.

References

1. Cho, S., Haralick, R., Yi, S.: Improvement of Kittler and Illingworth's minimum error thresholding. Pattern Recognition (1989)
2. Coudray, N., Buessler, J.L., Urban, J.P.: A robust thresholding algorithm for unimodal image histograms (2010)
3. Doane, D.P.: Aesthetic frequency classifications. The American Statistician (1976)
4. Hassaïne, A., Al-Maadeed, S., Bouridane, A.: A set of geometrical features for writer identification. ICNIP (2012)
5. Hassaïne, A., Decencière, E., Besserer, B.: Efficient restoration of variable area soundtracks. Image Analysis and Stereology (2009)
6. Howe, N.R.: Document binarization with automatic parameter tuning. International Journal on Document Analysis and Recognition, (2013)
7. Kadhim, N., Mourshed, M.: A shadow-overlapping algorithm for estimating building heights from VHR satellite images. IEEE Geoscience and Remote Sensing Letters (2018)
8. Kapur, J.N., Sahoo, P.K., Wong, A.K.: A new method for gray-level picture thresholding using the entropy of the histogram. Computer vision, graphics, and image processing (1985)
9. Katz, S., Tal, A., Basri, R.: Direct visibility of point sets. SIGGRAPH (2007)
10. Kittler, J., Illingworth, J.: Minimum error thresholding. Pattern Recognition (1986)
11. Kligler, N., Tal, A.: On Visibility and Image Processing. Ph.D. thesis, Computer Science Department, Technion (2017)
12. Kulis, B., Jordan, M.I.: Revisiting K-means: New Algorithms via Bayesian nonparametrics. ICML (2012)
13. Lindblad, J.: Histogram thresholding using kernel density estimates. In: SSAB Symposium on Image Analysis (2000)
14. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CVPR (2015)
15. Lu, H., Kot, A.C., Shi, Y.Q.: Distance-reciprocal distortion measure for binary document images. IEEE Signal Processing Letters (2004)
16. Onumanyi, A., Onwuka, E., Aibinu, A., Ugweje, O., Salami, M.J.E.: A modified Otsu's algorithm for improving the performance of the energy detector in cognitive radio. AEU-International Journal of Electronics and Communications (2017)
17. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics (1979)
18. Pratikakis, I., Zagoris, K., Barlas, G., Gatos, B.: ICFHR 2016 handwritten document image binarization contest (H-DIBCO 2016). ICFHR (2016)
19. Pun, T.: Entropic thresholding, a new approach. Computer Graphics and Image Processing (1981)
20. Sahoo, P.K., Soltani, S., Wong, A.K.: A survey of thresholding techniques. Computer vision, graphics, and image processing (1988)
21. Salakhutdinov, R., Roweis, S.T., Ghahramani, Z.: Optimization with EM and Expectation-Conjugate-Gradient. ICML (2003)
22. Sari, T., Kefali, A., Bahi, H.: Text extraction from historical document images by the combination of several thresholding techniques. Advances in Multimedia (2014)
23. Sauvola, J.J., Pietikäinen, M.: Adaptive document image binarization. Pattern Recognition (2000)

24. Sturges, H.A.: The choice of a class interval. Journal of the American Statistical Association (1926)
25. Tensmeyer, C., Martinez, T.: Document image binarization with fully convolutional neural networks. ICDAR (2017)
26. Wolf, C., Jolion, J.M., Chassaing, F.: Text localization, enhancement and binarization in multimedia documents. Object Recognition Supported by User Interaction for Service Robots (2002)
27. Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: A survey of unsupervised methods. Computer Vision and Image Understanding (2008)
28. Ziaei Nafchi, H., Farrahi Moghaddam, R., Cheriet, M.: Historical document binarization based on phase information of images. ACCV Workshops (2013)