DYNAMIC THRESHOLDING METHODS

CIARÁN Ó CONAIRE

ABSTRACT. This report surveys a number of dynamic thresholding methods for object and event detection. The benefits and drawbacks of each method are discussed, along with some experiments on real and synthetic data that illustrate the performance of each method and highlight the relevant issues involved in choosing the appropriate implementation of a dynamic thresholding algorithm.

1. Introduction

We provide here a survey of algorithms for automatically choosing thresholds for event or object detection. *Dynamic* thresholding, also known as *adaptive* or automatic thresholding, is a widely researched area. The goal of thresholding is to select a value to distinguish objects (or events) from background noise. We will use the terms *object* and *event* interchangeably to mean the relevant signal that we wish to separate from background noise and clutter. Most of the methods examined are *non-parametric*.

A very thorough review of thresholding techniques can be found in [11]. In [1] also, details of many algorithm implementations are given. Here we review only a subset of the algorithms that are available.

We divide the algorithms into two groups: Histogram-based methods are covered in section 2 and other methods are discussed in 3. We show the results of our experiments in section 4 and summarise our conclusions in section 5.

2. Histogram-based methods

Many thresholding methods have been proposed based on selecting the threshold using properties of a histogram of the data. Histogram-based thresholding methods are easy to implement, have fast execution times and have low memory usage. In practice, a system designer would have to specify the number of bins in the histogram and the window-size (i.e. how many samples to use to obtain the histogram). Histograms have the advantage that they can be computed online, which is useful for applications that involve the continuous processing of temporal information (audio event detection, for example).

For the histogram-based methods we investigate, the following terminology will be used for consistency. The N-bin histogram of the data is $(h_1, h_2, \cdots, h_N)$. We also write A_j and B_j as:

$$\begin{aligned} A_j &= \sum_{i=1}^{j} h_i \\ B_j &= \sum_{i=1}^{j} i h_i \end{aligned}$$

A_j is a count of the total number of values that are in bins 1 through j. Both A_j and B_j will be used later to describe algorithm implementations. Two of the other relevant thresholding methods that we do not cover include the moment-preserving *Tsai* method [12] and the minimum error thresholding method of Kittler and Illingworth [4].

2.1. **Kapur thresholding.** The Kapur [3] method selects a threshold to divide the histogram into two probability distributions, one representing the object/event and one representing the background noise. The threshold, T, is chosen such that the sum of the entropies of these probability distributions is maximised. Entropy is a measure of information and the entropy of a distribution $p(x)$ is given by:

$$e = -\int p(x) \log p(x)\, dx \tag{2.1}$$

When a histogram is normalised so that $\sum_{i=1}^{N} h_i = 1$, it can be thought of as a probability distribution. If we write

$$E_j = -\sum_{i=1}^{j} h_i \log h_i \tag{2.2}$$

the Kapur method chooses T so as to maximise the sum of the two-class entropies, E. We set T to the value of bin j that maximises

$$E = \frac{E_j}{A_j} + \log A_j + \frac{E_N - E_j}{A_N - A_j} + \log(A_N - A_j) \tag{2.3}$$

2.2. **Otsu thresholding.** The Otsu thresholding method [6] chooses T to split the histogram into two classes, such that the between-class variance is maximised and the intra-class variance is minimised. The algorithm positions T midway between the means of the two classes. We express the class means as:

$$\begin{aligned} \mu_j &= \frac{B_j}{A_j} \\ \nu_j &= \frac{B_N - B_j}{A_N - A_j} \end{aligned}$$

and select T as the value of bin j that maximises:

$$A_j(A_N - A_j)(\mu_j - \nu_j)^2 \tag{2.4}$$

An iterative version of Otsu's method was developed in [7] and has the advantage of being generally faster than trying every possible threshold. An initial threshold

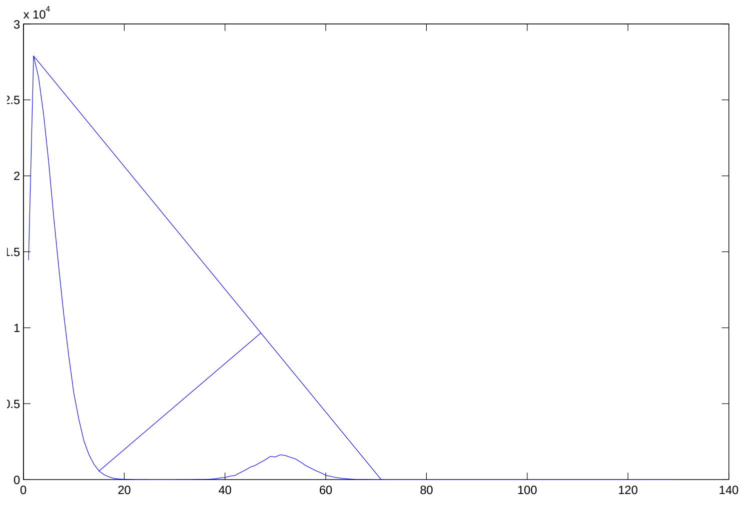

FIGURE 1. Geometric illustration of the Rosin thresholding method

is selected and the method uses it to compute a more accurate threshold, iterating until it converges. The result is, however, sensitive to the selection of the initial guess.

2.3. **Rosin unimodal thresholding.** Rosin's method [9] assumes that the noise has the dominant peak or that the histogram is *unimodal.* The method can easily be explained in geometric terms. First, a line is drawn connecting the histogram peak to the last non-empty bin. Next, for each bin between the peak and the last non-empty one, its perpendicular distance to the line is computed. The threshold is set at the bin whose distance is maximum. This is also known as the *histogram corner.* An extra precaution can be taken by requiring the bin height to be below the line and not above. The method will fail if the second peak (belonging to the object/event distribution) is larger than the background noise peak. In figure 1, an illustrative example of the Rosin method is shown. The plot shows a histogram with two distinct peaks, corresponding to the background noise (large peak) and the relevant signal (smaller peak). The data was synthetically generated to illustrate the method, but real data is far more noisy. Rosin thresholding can fail if the bin quantisation is too fine, as it will usually select an empty bin since it will be further from the line.

2.4. **Quick comparison.** Figure 2 shows the results of using the three methods from this section for thresholding audio loudness values. Kapur's method usually chooses larger threshold values than Rosin and therefore has a higher precision. Figure 3 shows a 256-bin histogram of the RMS data from figure 2 with the corresponding thresholds.

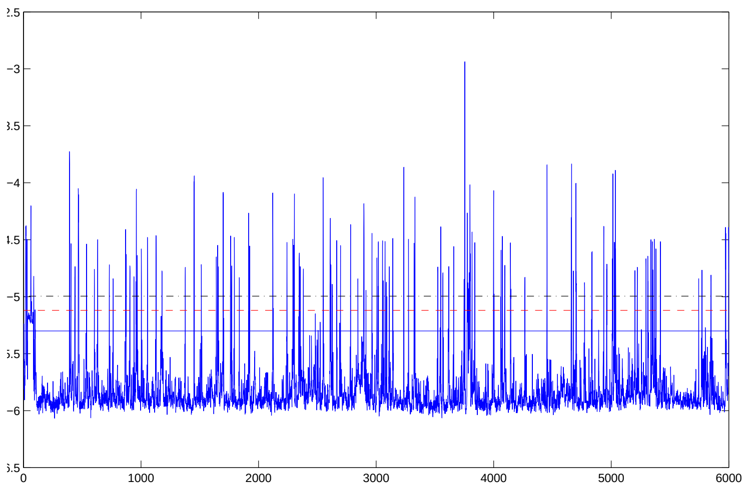

FIGURE 2. Audio Plot: log of the root-mean-squared (RMS) values. The computed thresholds for event detection are shown in red(Kapur), black(Otsu) and blue(Rosin).

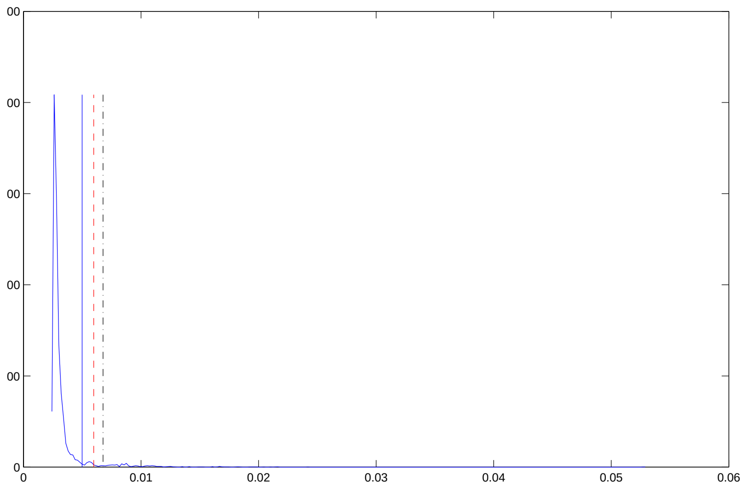

FIGURE 3. Histogram of audio RMS values. The computed thresholds are shown in red(Kapur), black(Otsu) and blue(Rosin).

3. OTHER THRESHOLDING METHODS

3.1. **Mean thresholding.** If the background noise is Gaussian (or can be well approximated as such) and is also the dominant mode in the histogram, it is possible to select a reasonable threshold by adding k standard deviations to the mean. This method is parametric and requires careful selection of the parameter k. Its value is usually between 1 and 4.

$$T_{mean} = \mu + k\sigma \tag{3.1}$$

If the number of background noise samples is much larger than the number of object samples, μ can be computed simply as the mean of the data. Otherwise, more robust methods should be used to cater for outliers. The standard deviation, σ, can be computed in a similar way. With $k = 1$, 68.26% of the noise is discarded. With $k = 2$, 95.45% of the noise is discarded. The larger the value of k, however, the greater the chance of accidentally discarding relevant samples. If the distributions are well separated, k can be set to larger values. Almost all thresholding methods benefit from a large separation between background and object distributions.

3.2. **Euler number thresholding.** In [8], a method for thresholding *difference images* is proposed. This method is specifically used for detecting any significant differences between two images. For example, the images could be of a mountain taken two weeks apart. The differences would show whether significant geological activity had taken place (such as a rock-slide). Another example is in a surveillance application, where one image is a background image and the second is the current image from the camera. Significant differences could correspond to objects moving through the scene, such as people or vehicles. This method is not suitable for one dimensional processing (such as audio thresholding), as it uses 2-D information in order to compute the threshold. The Euler-number of a binary image is the number of regions in the image, minus the number of holes in these regions. The principle behind this method is that, near the correct threshold, T can be varied and the number of regions will not change significantly. To select T, a graph relating Euler number to threshold is computed. The Rosin thresholding method is used on this graph to find the graph's *corner*, which is assumed to correspond to a stable region where the Euler number (and hopefully also the number of regions) does not vary much.

3.3. **Mutual information Thresholding.** When we have two separate, aligned, independent sources of data, *mutual information thresholding* [5] can be used. This method chooses two thresholds (one per source), such that the mutual information between the two resulting binary signals is maximised. The mutual information between sources A and B is a measure of the information contained in A about B, and vice versa (the measure is symmetric). In other words, given A, how much information does that give us about B. For two binary signals, A and B, the mutual information between them is given by:

$$I(A,B) = \sum_{x\in\{0,1\}} \sum_{y\in\{0,1\}} p(A=x, B=y) \log \frac{p(A=x, B=y)}{p(A=x)p(B=y)} \tag{3.2}$$

This method chooses thresholds T_1 and T_2 that are used to create the binary signals A and B from the original data signals. T_1 and T_2 are chosen to maximise $I(A,B)$.

4. Experiments

4.1. **Noise and scale performance.** In our first set of experiments, we created synthetic signals with different amounts of noise and with various proportions of

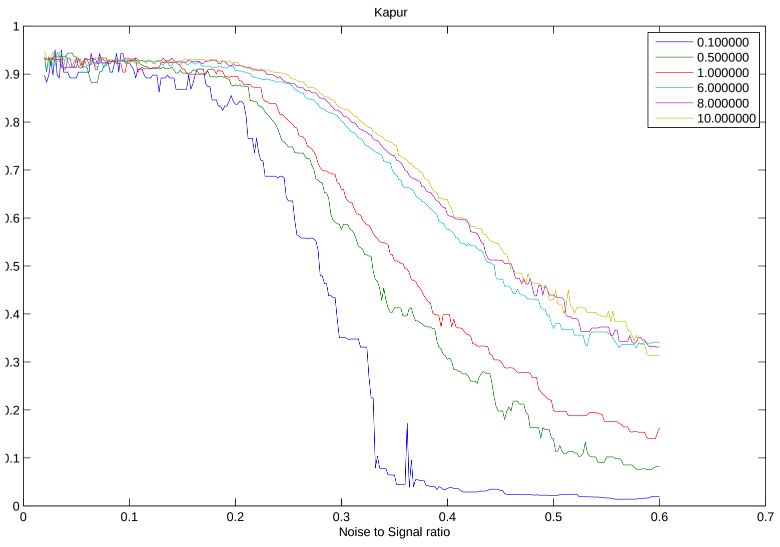

FIGURE 4. Kapur algorithm: Thresholding Performance

object and background samples. To gauge the threshold selection performance, we use the F-1 measure, or harmonic mean of precision(ρ) and recall(r) [13] [2], which is given by:

$$f = \frac{2}{\frac{1}{\rho} + \frac{1}{r}} = \frac{2\rho r}{\rho + r} \tag{4.1}$$

We evaluate 5 thresholding methods: Kapur, Otsu, Rosin and Mean thresholding with $k_1 = 1.96$ and $k_2 = 3.29$. The two values of k we used correspond to eliminating 95% and 99.9% of the noise respectively. In figures 4, 5, 6, 7 and 8, the results of our experiments are shown. The noise-to-signal ratio is shown on the x-axis. The six plots shown on each graph correspond to the change in percentage of object samples compared to the total samples. We used 0.1%, 0.5%, 1%, 6%, 8% and 10%.

4.2. **Histogram size effects.** The number of bins in the histogram can effect the resulting thresholds that are computed. In this experiment we vary the number of bins and plot the resulting thresholds computed by the Kapur, Otsu and Rosin methods. We use two real data signals and one synthetic signal. The real data signals are audio RMS values captured in a corridor and in a computer lab respectively. The first signal is shown in figure 2. The synthetic data is a constant event signal superimposed with Gaussian noise. It has a signal-to-noise-ratio (SNR) of 4 and 5% of the samples are *event* (95% are background noise). All signals contain 6,000 samples and are normalised between 0 and 100. The number of bins is varied from 8 to 1024. Figures 9, 10 and 11 show the results.

In choosing the number of bins for a histogram, we see that more bins means a more stable threshold. However, it also means more processing is needed to search

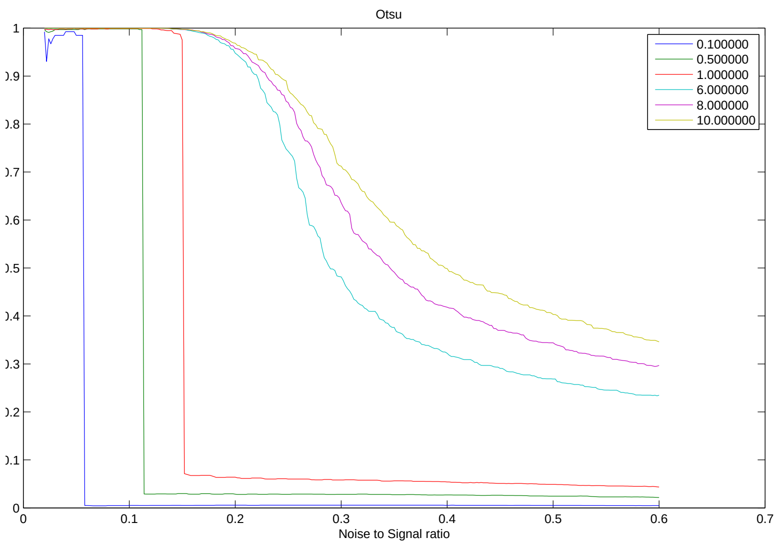

Figure 5. Otsu algorithm: Thresholding Performance

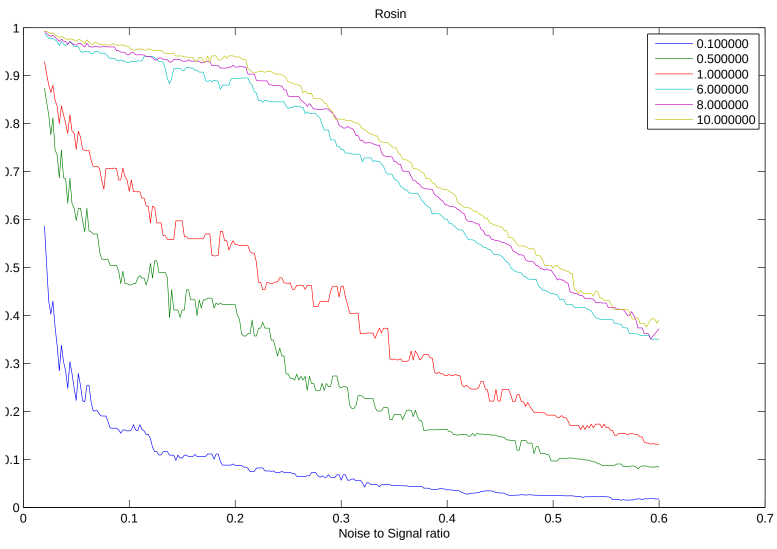

Figure 6. Rosin algorithm: Thresholding Performance

through the bins for the optimum threshold. At low numbers of bins, all methods are unstable. It seems that a minimum of about 64 bins is necessary for stability. It can be seen that the Kapur threshold is invariably greater than the Rosin threshold in all experiments. Kapur seems to have two stable ranges of correct threshold and between the two, there is an range of bins where the threshold value is unstable.

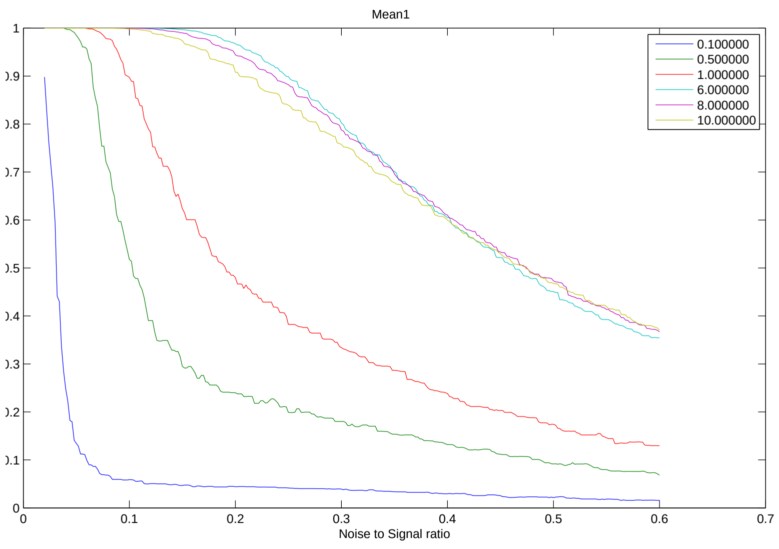

FIGURE 7. Mean-1 algorithm: Thresholding Performance

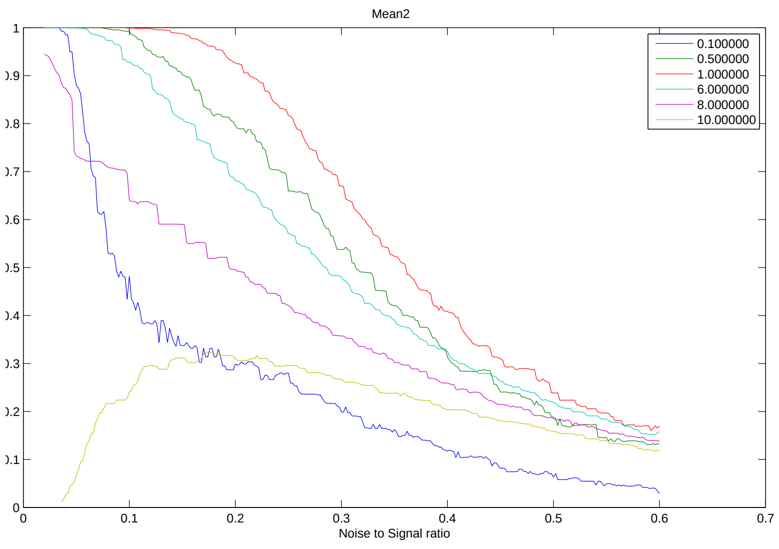

FIGURE 8. Mean-2 algorithm: Thresholding Performance

Otsu oscillates at low bin counts but stabilises quickly. Rosin is quite stable but in figure 11 it becomes unstable when large numbers of bins are used. This is due to the problem with empty bins, as mentioned in subsection 2.3.

It could be possible to select the number of bins dynamically, by examining a certain range of bins and ensuring that the threshold does not vary significantly.

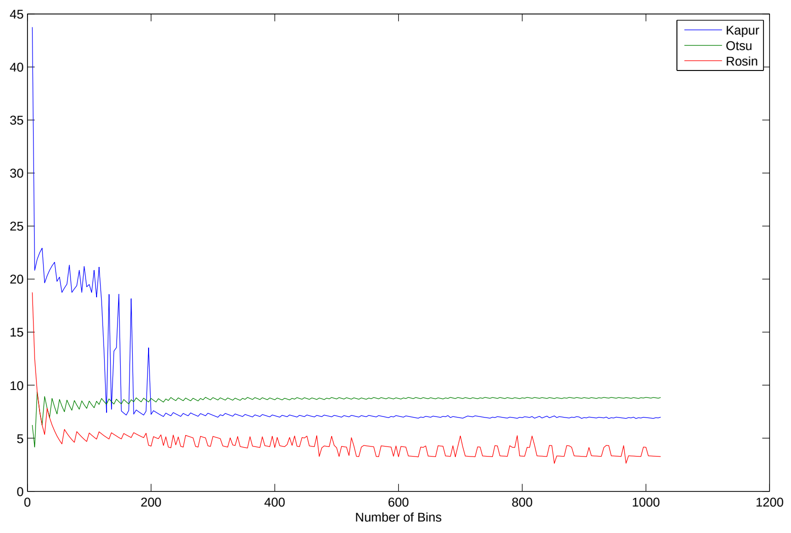

FIGURE 9. Audio data 1: The effect of varying the number of bins on the selected threshold.

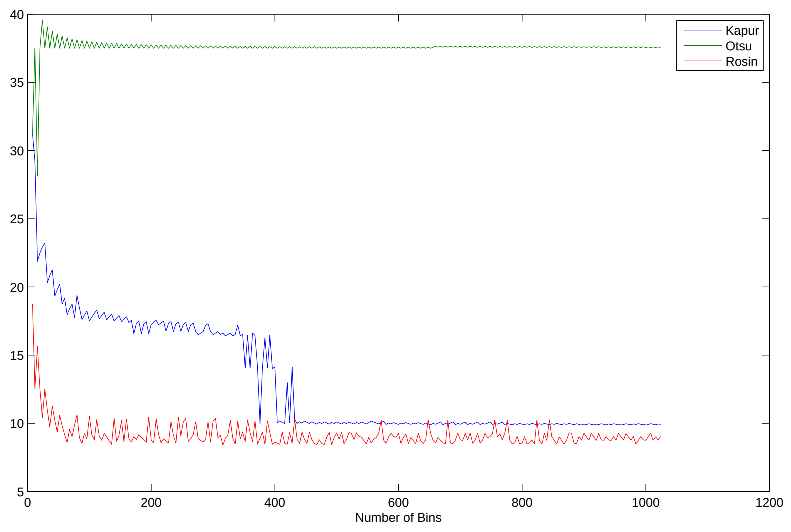

FIGURE 10. Audio data 2: The effect of varying the number of bins on the selected threshold.

Although these methods are non-parametric, the number of bins affects their performance, so might be considered a parameter.

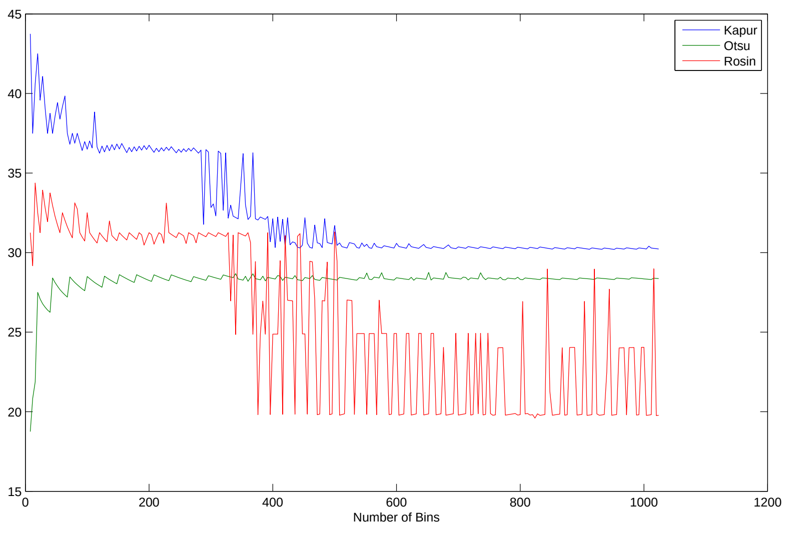

FIGURE 11. Synthetic data: The effect of varying the number of bins on the selected threshold.

5. Conclusion

This report described a number of algorithms to adaptively choose an appropriate threshold for object/event detection. In our experiments using synthetic data, Kapur thresholding was the most effective method overall, according to the f_1 measure of precision and recall. In an evaluation of thresholding methods using real data [10], Kapur also ranked as the top performing algorithm, closely followed by Rosin's algorithm. Both the simple mean threshold and Rosin threshold method do not make any attempt to model the relevant signal and instead only model the background noise, therefore they do not benefit from an increase in event frequency. We evaluated each method's sensitivity to the number of bins in the histogram and found that Kapur could be unstable in a certain range of thresholds, though it is always conservative in its selection of relevant samples, always choosing a threshold larger than the Rosin threshold. Otsu was found to be the most stable in relation to varying the number of bins.

References

1. F. Albregtsen, *Non-parametric histogram thresholding methodserror versus relative object area*, Proc. Eighth Scandinavian Conf. Image Analysis, 1993.
2. R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*, 1st edition ed., Addison Wesley, May 1999.
3. J. Kapur, P. Sahoo, and A. Wong, *A new method for graylevel picture thresholding using the entropy of the histogram*, Computer Graphics and Image Processing **29** (1985), no. 3, 273–285.
4. J. Kittler and J. Illingworth, *Minimum error thresholding*, Pattern Recognition **19** (1986), no. 1, 41–47.
5. C. Ó Conaire, N. O'Connor, E. Cooke, and A. Smeaton, *Detection thresholding using mutual information*, VISAPP: International Conference on Computer Vision Theory and Applications, Setúbal, Portugal (to be published), Feb 2006.

6. N. Otsu, *A threshold selection method from gray-level histogram*, IEEE Transactions on System Man Cybernetics **9** (1979), no. 1, 62–66.
7. T.W. Ridler and S. Calvard, *Picture thresholding using an iterative selection method*, IEEE Trans. Systems, Man, and Cybernetics **8** (1978), no. 8, 630–632.
8. P. Rosin, *Thresholding for change detection*, IEEE International Conference on Computer Vision, 1998, pp. 274–279.
9. P. L. Rosin, *Unimodal thresholding*, Pattern Recognition **34** (2001), no. 11, 2083–2096.
10. P.L. Rosin and E. Ioannidis, *Evaluation of global image thresholding for change detection*, Pattern Recognition Letters **24** (2003), no. 14, 2345–2356.
11. M. Sezgin and B. Sankur, *Survey over image thresholding techniques and quantitative performance evaluation*, Journal of Electronic Imaging **13** (2004), no. 1, 146–165.
12. W. Tsai, *Moment-preserving thresholding: A new approach*, Comput. Vision Graphics Image Process., **29** (1985), no. 3, 377–393.
13. C. J. Van Rijsbergen, *Information retireval*, 2nd edition ed., Dept. of Computer Science, University of Glasgow, Butterworths, London, 1979.

E-mail address: `oconaire@eeng.dcu.ie`