A FAST METHOD FOR COMPUTING THE OUTPUT OF RANK ORDER FILTERS WITHIN ARBITRARILY SHAPED WINDOWS

Paul Murray and Stephen Marshall

Department of Electronic and Electrical Engineering, University of Strathclyde
Royal College Building, 204 George Street, G1 1XW, Glasgow, Scotland
phone: + 44 (0) 141 548 2205, fax: +44 (0) 141 552 2487, email: paul.murray@strath.ac.uk
web: www.strath.ac.uk/eee/research/cesip/

ABSTRACT

Rank order filters are used in a multitude of image processing tasks. Their application can range from simple pre-processing tasks which aim to reduce/remove noise, to more complex problems where such filters can be used to detect and segment image features. There is, therefore, a need to develop fast algorithms to compute the output of this class of filter. A number of methods for efficiently computing the output of specific rank order filters have been proposed [1]. *For example, numerous fast algorithms exist that can be used for calculating the output of the median filter. Fast algorithms for calculating morphological erosions and dilations - which are also a special case of the more general rank order filter - have also been proposed. In this paper we present an extension of a recently introduced method for computing fast morphological operators to the more general case of rank order filters. Using our method, we are able to efficiently compute any rank, using any arbitrarily shaped window, such that it is possible to quickly compute the output of any rank order filter. We demonstrate the usefulness and efficiency of our technique by implementing a fast method for computing a recent generalisation of the morphological Hit-or-Miss Transform which makes it more robust in the presence of noise. We also compare the speed and efficiency of this routine with similar techniques that have been proposed in the literature.*

1. INTRODUCTION

Rank order filters are a set of non-linear filters that are commonly used in image processing to solve a number of problems ranging from simple noise removal to more complex tasks such as object recognition. The output of a rank order filter, of rank *k*, at a point *p* in an image, may be computed in two steps. First, we must sort into ascending order, the image pixels that are coincident with a window, *B*, when it is centred on a point, *p*, as it scans the image. The value assigned to point *p* in the output image is then the value of the *k*th order statistic of the image pixels that are coincident with *B* when it is centred on *p*. For example, let $x_1, x_2 \ldots x_n$ represent a set of arbitrary pixel intensities that are coincident with some window, *B*, where $n = Card(B)$ i.e. the cardinality of the set *B*. If we sort these values in ascending order such that,

$$x_{(1)} \le x_{(2)} \le \ldots \le x_{(n)},$$

then $x_{(k)}$ represents the *k*th order statistic [2]. The output of, $\zeta_{B,k}$ a rank order filter, of rank *k* with window *B*, when centred at a point *p*, is the value $x_{(k)}$. For a point *p*, in an image *f*, the output of the rank order filter may be computed using,

$$\left[\zeta_{B,k}(f)\right](p) = k\text{th order} \underset{b \in B}{\text{statistic}} \left\{ f(p+b) \right\}.$$

Perhaps the best known and most commonly used rank order filter is the median filter which is often used as a pre-processing step in image analysis to remove/reduce noise while preserving edges. Over the years this operator has received a lot of attention and various techniques which exploit the redundancy that exists when computing the median of a set of pixels that coincide with a sliding window have been proposed for reducing the execution time of this filter, [2], [3] and [4]. In [3], Huang et al. present an efficient technique for computing the output of a median filter. The authors exploit the fact that when calculating the output of this filter using a sliding window, only a small number of the values that are considered in the calculation of the median actually change as the window moves from the current pixel to its neighbour. This means that instead of re-sorting every value in the window, we only need to consider the pixels that exit the window and the new pixels that enter it as it is translated from a pixel, *p*, to its neighbour before calculating the new median. This leads to an increase in speed which is further enhanced by a novel histogram technique that is used to sort the values that are coincident with the window and efficiently locate the median value. The authors describe and demonstrate their method using square and rectangular windows, and although it proves very effective and efficient, no technique is provided for the case that the window is of an arbitrary shape.

Standard morphological erosions and dilations are also a special case of rank order filters. This is discussed in [4] where it is shown that the minimum rank filter, where $k = 1$, using a window *B*, is equivalent to an erosion, ε_B, and the

maximum rank filter, where k = n, using B, is equivalent to a dilation, δ_B. That is,

$$\varepsilon_B = \zeta_{B,1}$$

and

$$\delta_B = \zeta_{B,n}.$$

In [1], Van Droogenbroeck and Talbot use a technique that is similar to the one used in [3] to compute fast erosions and dilations by searching the histogram for the minimum or maximum rank respectively instead of the median. The major contribution of this work however lies in the fact that the authors extend the techniques proposed in [3] such that it is possible to compute the output of the max/min filters using any arbitrarily shaped window. This is of particular importance for morphological operations where the shape of the structuring element (window) used to process an image is critical.

More recently, [5] introduced a technique which always outperforms the method proposed in [1] when erosions and dilations are performed. Although the fastest known method for calculating morphological applications is the one presented in [5], the authors state explicitly that their method cannot be extended to the implementation of general rank order filters.

In this paper, we extend the method described in [1] - which demonstrates fast computation of max (dilation) and min filters (erosion) - such that it is possible to compute the output of any rank order filter. In addition to this, we provide in Section 2, a mathematical formulation that can be used to calculate the so called "critical points" of a sliding window which is not given in [1]. We also demonstrate the usefulness of our extension by using these fast rank order filters to implement an extremely efficient Percentage Occupancy Hit-or-Miss Transform (POHMT) [6] as described in Section 3. The POHMT is a recently introduced extension the Hit-or-Miss Transform (HMT) that is robust to noise and other distortions that exist in digital images. Since this transform may be implemented using rank order filters as explained in Section 3, it is an ideal application to demonstrate the usefulness and efficiency of the method proposed here. Finally, we compare the efficiency of our routine with an optimised method for calculating rank order filters to show that our technique is faster.

2. A FAST METHOD FOR COMPUTING THE OUTPUT OF AN ARBITRALILY SHAPED RANK ORDER FILTER

In this section, we provide a concise description of the methods proposed in [1] and [3] for efficiently calculating the output of max, min and median filters before showing how we have extended these in order to quickly compute the output of any arbitrarily shaped rank order filter.

In [1], Van Droogenbroeck and Talbot use a sorting technique that is similar to the one used by Huang et al. in [3] to sort the pixels that coincide with the window, B. The method calculates a histogram of the image pixels that are coincident with the window as it scans the image. It is then possible to find the min/max

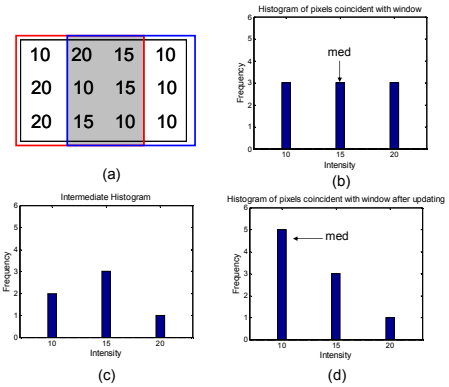

Figure 1 - Illustration of the histogram method used to calculate the median filter. (a) Arbitrary pixel values coincident with a 3x3 window before (red) and after translation (blue). Shaded area - pixels that do not change as window is translated. (b) Histogram of pixels in red window, median = 15. (c) Histogram of pixels that remain in the window after those leaving it have been subtracted. (d) Histogram with new values in blue window added, new median = 10.

from the sorted values in this histogram in order to calculate the output of an erosion/dilation respectively at each image pixel. However, instead of generating a new histogram for each translation of the window in the image, the histogram is simply updated by removing the values that exit the window and adding those that enter it as it is translated from pixel, p, to its neighbour. Updating the histogram is achieved by decrementing the count in the bins that correspond to the values exiting the window and incrementing the count in the bins corresponding to the values entering it. The min/max value is easily located using this technique since the pixels are already sorted in the histogram.

The authors point out in [1], that it is not always necessary to consult the histogram when searching for the maximum or minimum rank. In the case of erosion (similarly for dilation) it is only required to search for a new minimum if a value entering the window during translation is lower than the current minimum, or if the count in the histogram bin corresponding to the current minimum reaches zero. For this reason, the authors keep a record of the minimum value which is read for each translation of the window. A new minimum is only sought if one of the previous mentioned situations occurs.

Although the histogram must be searched and consulted much more frequently in the case of the median filter proposed by Huang et al. in [3], it is possible to save time by correctly directing the search. This is achieved by keeping a count of the number of pixels in the histogram that have intensity lower than the median. It is then possible to use this count to decide whether we must search up or down the histogram to find the new median. Let the count of pixels that have a grey level value less than the median be denoted c. If c is greater than the rank k corresponding to the median (window size / 2) we must look to the left (lower values within the histogram) of the current median to find the new one. If c is not greater than the rank k corresponding to the median then the new median may either: remain unchanged; or, if not, we must search to the right

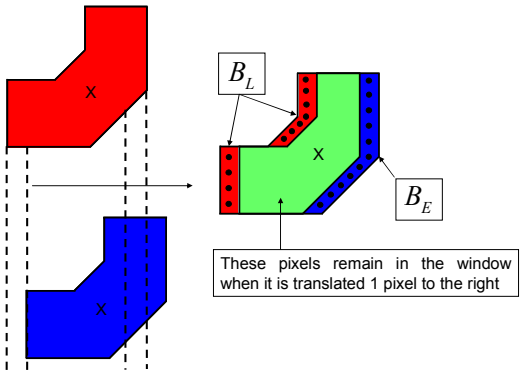

Figure 2 - Illustration of the critical points in the window. (Top) red window centred at (p). (Bottom) Blue window – red window translated 1 pixel right centred at point (p+1). (Right) Critical points of the window (red = B_L and blue = B_E) and the points that remain in the window (green) following a single pixel translation to the right.

(higher values within the histogram) of the current median to find the new one.

We demonstrate in Figure 1, the property of the overlapping window that is exploited to efficiently update the histogram and show by example how to calculate the value of the median using a 3x3 square window. It is more appropriate to demonstrate the calculation of the median rather than erosion or dilation since it facilitates the explanation of our extension of this operator such that we can calculate the outputs of any rank order filter.

Since the shape of a structuring element is extremely important for morphological operations, Van Droogenbroeck and Talbot [1], have extended the method proposed Huang et al. in [3] such that it is possible to compute fast erosions and dilations with any arbitrary window. Take for example the square window used in Figure 1. The only pixels which need to be updated in the histogram are those that are not included in the intersection (shown in grey region of window in Figure 1) of the window and its translation as it moves from one pixel to its neighbour. This concept is further illustrated in Figure 2 where the pixels leaving the window are denoted B_L and those entering it are denoted B_E. These points are called the "critical points" in [1] and may be computed for any window, initially centred at some point p, using,

$$B_L = B(p) \setminus \left(B(p) \cap B(p+1) \right) \quad (1)$$

and

$$B_E = B(p+1) \setminus \left(B(p) \cap B(p+1) \right) \quad (2)$$

Given (1) and (2) it is possible to compute the critical points for any arbitrarily shaped window and store these in memory. For each translation of the arbitrary window, we remove from the histogram the value of those pixels which coincide with the points B_L and add to the histogram the values in the image that coincide with the points of B_E.

By combining the techniques demonstrated in Figures 1 and 2, it is possible to extend the methods described previously such that we can compute the output of any rank order filter defined by any arbitrarily shaped window. By using the method of computing the critical points of the window and using these to update the histogram, we can then search for any rank, $1 \le k \le n$, using the method described for calculating the median filter as demonstrated in Figure 1. That is, by generating a histogram of the image pixels coincident with the window and updating this histogram with the pixels coincident with the critical points as the window moves, we can exploit the redundancy associated with re-sorting these values. Then, by keeping a count of the number of grey levels that are lower than the intensity of the pixel in any rank k, we can quickly compute the output of any rank order filter defined by any arbitrary window B.

This can be further generalised if the rank is specified as a percentage, where we would search for the P % rank, allowing the rank to be specified independently of the size and shape of the window. Regardless of the size or shape of the window it is possible to identify the kth rank using,

$$k = \text{floor}\left(\frac{P}{100} \times Card(B) \right). \quad (3)$$

Using (3) it is then possible to compute the output of any rank order filter where for example a dilation would be implemented when P=100%. Specifying the rank in this way is particularly useful when implementing the POHMT as described in Section 3.

3. A PERCENTAGE OCCUPANCY HIT-OR-MISS TRANSFORM IMPLEMENTED USING RANK ORDER FILTERS

We demonstrate here, the benefits of extending the method proposed in [1] to the more general case of computing rank order filters by using our extension described in Section 2 to implement a fast POHMT. The POHMT is a recent extension of the well known morphological HMT [4] which relaxes the fitting criteria of the structuring elements (SEs) such that it is robust when image data is noisy or otherwise distorted. We provide a brief description of the POHMT here, where the interested reader is referred to [6] for a fuller explanation.

The HMT of a binary image X is the intersection of an erosion of X and an erosion of the complement of X by a complementary pair of SEs B_{FG} and B_{BG} respectively where X, B_{FG} and B_{BG} are sets in 2D space, $E = \mathbb{Z}^2$. B_{FG} and B_{BG} are defined relative to a common origin in E where the composite SE $B = B_{FG} \cup B_{BG}$ and $B_{FG} \cap B_{BG} = \varnothing$. That is,

$$HMT_B(X) = \left\{ x \in E \,|\, (B_{FG})_x \subseteq X,\ (B_{BG})_x \subseteq X^c \right\},$$

where $(B)_x = \{ b + x \,|\, b \in B \}$. A feature is detected by the HMT if there is at least one point $x \in E$ such that the foreground SE $(B_{FG})_x$ is included in X whilst the background SE $(B_{BG})_x$ is simultaneously included in its complement, $X^c = E \setminus X$, see [4]. To remain consistent with the literature, we define the notation used here. Let E represent a two dimensional digital space ($E = \mathbb{Z}^2$) and T^E be the set of all grey level functions from a subspace of E to T where $T = \mathbb{R} \cup \{+\infty, -\infty\}$ or $T = \mathbb{Z} \cup \{+\infty, -\infty\}$ such that T is a complete lattice with

respect to the order "$\leq$". Let $I \in T^E$, denote a greyscale image, and $B \in E$ denote a flat SE.

The POHMT generalises the HMT by allowing partial fitting of a composite SE such that objects of interest can be detected in an image despite the presence of noise. Instead of using traditional erosions and dilations which require a perfect match between image features and the SEs, the POHMT considers the extent to which a composite SE, B, "fits" the image as it is raised through all grey levels, $t \in T$, when its origin is coincident with a point $x \in E$, $\forall x \in E$. A point $x \in E$ is marked in the output of the POHMT if - when the origin of the SE is coincident with $x \in E$ - P %, (where P is set prior to running the POHMT using a design tool that was introduced in [6]) of the points $B_{FG} \in B$ are beneath or at the same level as the signal, while simultaneously, P % of the points in $B_{BG} \in B$ are strictly above the signal. If, when the origin of B is coincident with a point $x \in E$, there is at least one level $t \in T$ for which this condition is satisfied, we deem that the composite SE is P % occupied in the image and this point is marked in the output of the transform. We may calculate the extent to which B_{FG} and B_{BG} are occupied $\forall x \in E$ and $\forall t \in T$, using,

$$PO_{FG_{x,t}} = \left[\frac{Card\left\{ b_{FG} \in B_{FG} \mid I\left(x + b_{FG}\right) \geq t \right\}}{Card\left(B_{FG}\right)} \right] \times 100 \qquad (4)$$

and

$$PO_{BG_{x,t}} = \left[\frac{Card\left\{ b_{BG} \in B_{BG} \mid I\left(x + b_{BG}\right) < t \right\}}{Card\left(B_{BG}\right)} \right] \times 100 \qquad (5)$$

The output of the POHMT may then be calculated by substituting (4) and (5) into,

$$POHMT_{[B \in B_{FG} \cup B_{BG}]}(x) = \begin{cases} 2^n\text{-}1 & \text{if } \max\limits_{t \in T}\left[\min\left\{ PO_{FG_{x,t}},\ PO_{BG_{x,t}} \right\} \right] \geq P \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

A direct implementation of the POHMT which processes the image in a single pass can be realised using (4), (5) and (6). Ultimately, the POHMT relaxes the strict conditions of the standard HMT by allowing the partial fitting of SEs where the extent to which the SEs fit a feature in the image may be measured using (4) and (5). A point $x \in E$ is then marked in the output of the transform if $\exists t \in T$ such that P % of B_{FG} fits the feature while P % of B_{BG} fits its background.

A common technique that can be used to relax the strictness of morphological operators is to implement more general rank order filters in place of traditional erosions and dilations [4], [6]. Indeed, an equivalent and more efficient implementation of the POHMT can be achieved using the fast rank order filters described in Section 2. Instead of calculating PO_{FG} and PO_{BG} using (4) and (5) and subsequently the output of the POHMT using (6) $\forall x \in E$ and $\forall t \in T$, it is equivalent to implement this transform using,

$$POHMT_{[B \in B_{FG} \cup B_{BG}]}(x) = \begin{cases} 2^n\text{-}1 & \text{if } \left[\zeta_{B_{FG},100-P}(I)\right](x) > \left[\zeta_{B_{BG},P}(I)\right](x) \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

It is clear from (7) that the POHMT may be calculated using rank order filters where the parameter P in (6) may be used to set the rank k for the filter. When implementing the POHMT in this way, we look for places in an image where the intensity in rank $(100 - P)$ is greater for the elements of B_{FG} than the intensity in rank P for the elements of B_{BG}. The output of the POHMT contains marker pixels for all $x \in E$ that are coincident with the origin of the SE when this condition is satisfied.

We give an example here of how this may be implemented. Say we determine for a particular image set, that P should equal 75% for successful detection of the features that we wish to locate in a noisy image. We can implement the POHMT using rank order filters such that the image pixel that is coincident with the origin of B is marked in the output if the rank corresponding to (100 - 75)% of B_{FG} is at an intensity level greater than the level contained in the rank corresponding to 75% in B_{BG}. This is consistent with the definition of the POHMT as given in (6).

4. EXPERIMENTAL RESULTS

We have implemented the technique described in Section 2 to measure the execution time of our generalised rank order filters. In this section, we compare the speed of our C routine with an optimised Matlab function which calculates the output of rank order filters using techniques described in [3] and [7]. Further, to demonstrate the usefulness of this implementation, we draw a comparison between the accuracy and efficiency of a fast POHMT (implemented using the rank order filters described here) with a similar technique that is proposed in [8], which makes the HMT more robust in the presence of noise. We show that our method achieves the same results when processing the image data presented in [8] but in a fraction of the time that is reported there.

We have measured the execution time of our routine when processing a noisy 512 x 512, 8 bit, greyscale image using an arbitrary window which increases in size. The number of elements in the window ranges from 289 points to 38809 points. We have computed the output of four rank order filters where P = 1, P = 30, P = 50 and P = 100 and have compared the execution time of our implementation and the optimised Matlab routine for each window of increasing size. The results of the comparison for each rank order filter are shown in Figure 3. By reference of Figure 3, it is clear that for all four rank order filters, and for all window sizes, our implementation always outperforms the competing technique by a factor of around 2.

The method presented in [8] uses an extension of the standard HMT to recognise so called "Low Surface Brightness Galaxies" (LSBs) in a set of very noisy astronomical

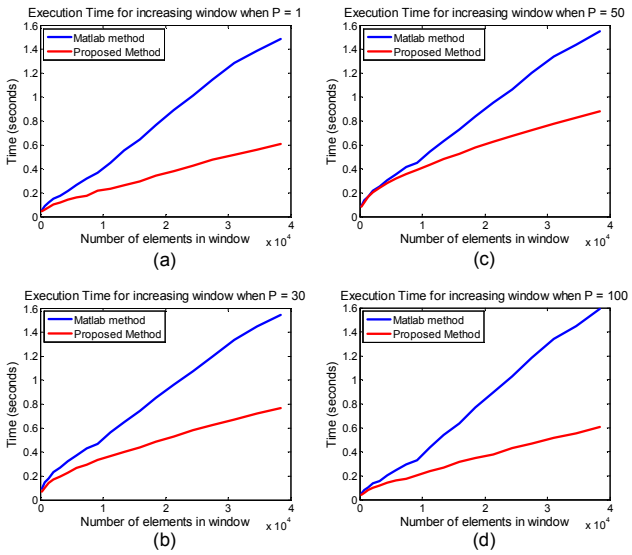

Figure 3 - Speed comparison of our fast rank order filters with optimised Matlab routine for increasing window size. (a) Minimum filter, P = 1 (b) 30% filter, P = 30 (c) median filter, P = 50 (d) Maximum filter, P = 100.

images. The authors state that an optimised implementation (using heuristic techniques) of their routine takes on average 2 minutes to process one image.

An example of two of these images is shown in Figure 4(a). Clearly, the data is extremely noisy and there are a number of features in the image, in addition the LSB, which we do not wish to detect. We show in Figure 4(b) the results that we achieve when processing this image using the fast POHMT. Note that in Figure 4(b) we have performed opening by reconstruction using the original image and the marker image produced by the POHMT purely for visualisation of the result. Clearly, the only feature that has been marked in the output of the transform is the LSB. Further, the average time taken to process one image using our method is 15 seconds which is a significant improvement on the 2 minutes reported in [8].

5. CONCLUSIONS

We have presented here, a fast algorithm for computing the output of any rank order filter within any arbitrary window. Our technique is a more general version of the method presented in [1] that can be used to compute the output of any rank order filter and is not limited to special cases such as erosion and dilation. In addition to this extension, we provide a mathematical formulation which is missing in [1] that can be used to calculate the so called critical points of the window that must be used when updating the histogram to avoid redundant computation.

We have compared the efficiency of our rank order filters with an optimised Matlab routine for calculating the same. In all cases our method is faster and executes in around half of the time taken by the competing technique. Further, we have shown that by using our fast rank order filters to implement a fast POHMT, we obtain similar results to those presented in [8], in a fraction of the time that is quoted there.

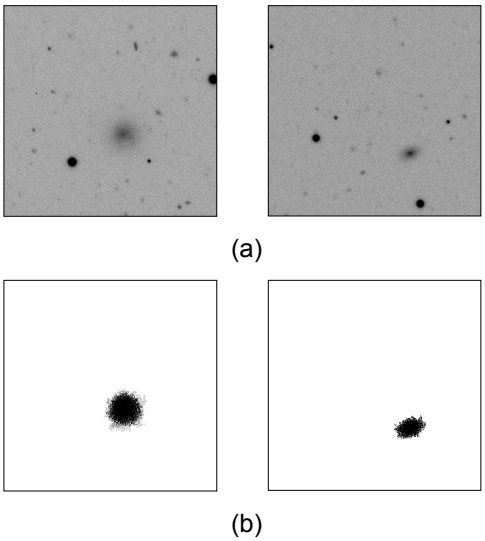

Figure 4 - Example of the fast POHMT detecting features of interest in very noisy data.. (a) Source images containing LSB and other features. (b) Output of the fast POHMT followed by an opening by reconstruction.

While the application of our more general algorithm for calculating the output of arbitrarily shaped rank order filters has been restricted to a fast implementation of the POHMT it is possible to use this technique in a number of applications. In fact, any image processing pipeline which uses median filters, morphological operations or more general rank filters to reduce noise or to achieve some other result could see a significant reduction in processing time by implementing these stages using the method proposed here.

REFERENCES

[1] M. Van Droogenbroeck, H. Talbot, "Fast computation of morphological operations with arbitrary structuring elements," *J. Pattern Recognition Lett.*, vol. 17, no. 14, pp. 1451-1460, 1996.

[2] I. Pitas and A. N. Venetsanopoulos, "Order statistics in digital image processing", *Proc. IEEE*, vol. 80, no. 12, pp. 1893 - 1921, 1992.

[3] T.S. Huang, G.J. Yang, and G.Y. Tang, "A fast two dimensional median filtering algorithm," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol 27, no. 1, pp. 13-18, Feb 1979.

[4] P. Soille, *Morphological Image Analysis: Principles and Applications*, New York: Springer, 2003.

[5] E. Urbach and M.H.F Wilkinson, "Efficient 2-D Grayscale Morphological Transforms with Arbitrary Flat Structuring Elements", *IEEE Trans. Image Processing*, Vol. 17, No.1, Jan. 2008

[6] P. Murray and S. Marshall, "A New Design Tool for Feature Extraction in Noisy Images based on grayscale Hit-or-Miss Transforms", *IEEE Trans. Image Processing*, doi: 10.1109/TIP.2010.2103952

[7] Haralick, Robert M., and Linda G. Shapiro, *Computer and Robot Vision*, Volume I, Addison-Wesley, 1992.

[8] B. Perret, S.Lefèvre, Ch.Collet, "A robust hit-or-miss transform for template matching applied to very noisy astronomical images," *J. Pattern Recognition*, Vol. 42, no. 11, pp. 2470-2480, Nov. 2009.