UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Distance metrics and clustering algorithms for detection and classification of process sensitive patterns

Permalink

https://escholarship.org/uc/item/885689vs

Author

Ghan, Justin

Publication Date

2010

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Distance metrics and clustering algorithms for detection and classification of process sensitive patterns

by

Justin Ghan

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kameshwar Poolla, Chair
Professor Costas Spanos
Professor Andrew Packard

Fall 2010

Distance metrics and clustering algorithms
for detection and classification of process sensitive patterns

Copyright © 2010

by

Justin Ghan

Abstract

Distance metrics and clustering algorithms
for detection and classification of process sensitive patterns

by

Justin Ghan

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor Kameshwar Poolla, Chair

Detection of process sensitive patterns known as hotspots is critical to maximising yield in integrated circuit layouts. Design rule checking is a common tool in the industry for fast hotspot detection, but it is becoming impractical as we approach smaller technology nodes, due to an exponentially growing rule book which must be manually maintained.

A pattern matching based system for hotspot detection is explored in this dissertation which has the advantage of being automatically generated while still performing detection with a runtime comparable to design rule checking. Instead of scanning a design layout for certain combinations of distances found in a library of rules, we use a pattern matching tool to find occurrences in the layout of patterns which match or closely resemble known bad configurations.

In order for such a system to detect hotspots effectively, we require a comprehensive catalogue of hotspot patterns for the pattern matcher to use. Of course, it is not feasible to simply store all known hotspots patterns in the catalogue, as the memory requirements and pattern matching runtimes would be enormous. Instead, the catalogue must contain a smaller number of hotspot classes, where each class represents a set of geometrically similar patterns which embody the same type of hotspot. The pattern matching tool is able to find matches to a hotspot class by matching to a particular representative pattern of the class, and allowing for small geometric deviations from this representative.

To build such a catalogue, a large diverse set of hotspot clips must be analysed and reduced to a small number of hotspot classes. To perform this reduction meaningfully, a deep understanding of what it means for two hotspots to be the same "type" of hotspot is required. We perform an in-depth analysis of the process sensitivity of patterns. From this we design a distance metric which quantifies the extent to which the process sensitivities of two hotspots are caused by similar geometry of surrounding features.

Equipped with this distance metric, clustering algorithms are used to group a dataset of clips into a small number of clusters containing similar hotspots. Because the number of clips to analyse must be very large in order to build a comprehensive catalogue, it is important to make this process as computationally efficient as possible. To this end, the distance metric and the clustering algorithm are designed and tailored for maximum efficiency.

To test our hotspot detection system, we build a catalogue of hotspot classes based on a dataset of clips extracted from a device layout. A pattern matching tool then uses our catalogue to detect hotspots, and the results are presented.

Contents

List of Figures

Chapter 1

Introduction

Design for manufacturability is playing an ever-increasing role in the production of integrated circuits. As we push towards smaller and smaller devices, the computational complexity of hotspot detection, optical proximity correction and double patterning colouring grows. The need for faster and more accurate solutions to these tasks is becoming more urgent as they threaten to become an insurmountable obstacle in the design flow.

The elimination of hotspots from a layout is among the primary concerns in device design. Small variations in the lithography process cause the printed contours to deviate from the nominal condition contours. Patterns which have a particularly high sensitivity to process variation may cause a reduction in yield if these contour deviations are significant enough to affect the electrical performance of the device. Such patterns are called hotspots, and it is critical to detect and fix these to ensure high yield.

Model-based approaches to hotspot detection are the most precise and robust, and these methods will always have their place in the design flow. However, in the early stages of the flow, where the ability to perform many iterations of design and verification is desirable, simulation is too computationally expensive to be relied upon as the only tool. Faster solutions are required, even if at the slight expense of accuracy.

To play this role, various rule-based approaches such as design rule checking have been introduced. These methods are typically very fast; however, they are becoming impractical as the radius of interaction between features grows. An exploding number of increasingly complex rules are required, and each of these rules must be manually designed and coded. Decisions must be made as to how restrictive to make the rules, and ultimately

not all possible geometries can be taken into account.

In this dissertation, we present an alternate hotspot detection system, which has the capability and speed of traditional design rule checking, but has the important advantage that criteria for detection can be automatically generated without human analysis or labour. Instead of detecting hotspots by checking combinations of distances in various configurations, our hotspot detection system utilises pattern matching. Pattern matching technology is very fast and powerful, but its use has not yet become widespread in the industry. We believe that it can potentially be used to solve many major problems in the field of design for manufacturability, and our hotspot detection system is just a first example.

1.1 Overview of hotspot detection

Design rule checking has been the industry standard for fast layout verification for many years. In earlier generations, a small number of simple design rules were sufficient to assure printability of a layout. An example of a simple early design rule is the tip-to-line rule. This rule states the minimum allowable space width between the tip of a terminating line and a second perpendicular line, as shown in Fig. 1.1. The mask polygons are shown in black, and the resulting printed image contours are shown in red. Due to diffractive effects in the lithography system, the image contours do not exactly match the mask polygons. The design rule states that, so long as as the space width, a, is greater than the threshold, a_{th}, as shown on the left, then the two lines will resolve separately, and will function electrically as intended. However, if the minimum space width is violated, as shown on the right, then the two lines may be joined in the resulting image. This is called bridging, and can cause circuit failure.

Simple rules such as these were reliable in the past. However, as we print smaller and smaller lines with each subsequent technology node, the number of surrounding features which affect a single edge placement is increasing. Because of this, simple rules are no longer able to adequately guarantee printability.

As an example, let us examine the tip-to-line behaviour at a current technology node. Suppose we attempted to use the same simple tip-to-line rule, determining the space width threshold, a_{th}, by examining isolated pairs of lines as depicted in the previous examples (Fig. 1.1). Now consider a more complex geometry, such as line terminating inside a U shape, as shown in Fig. 1.2. Here, the space width, a, is greater than the

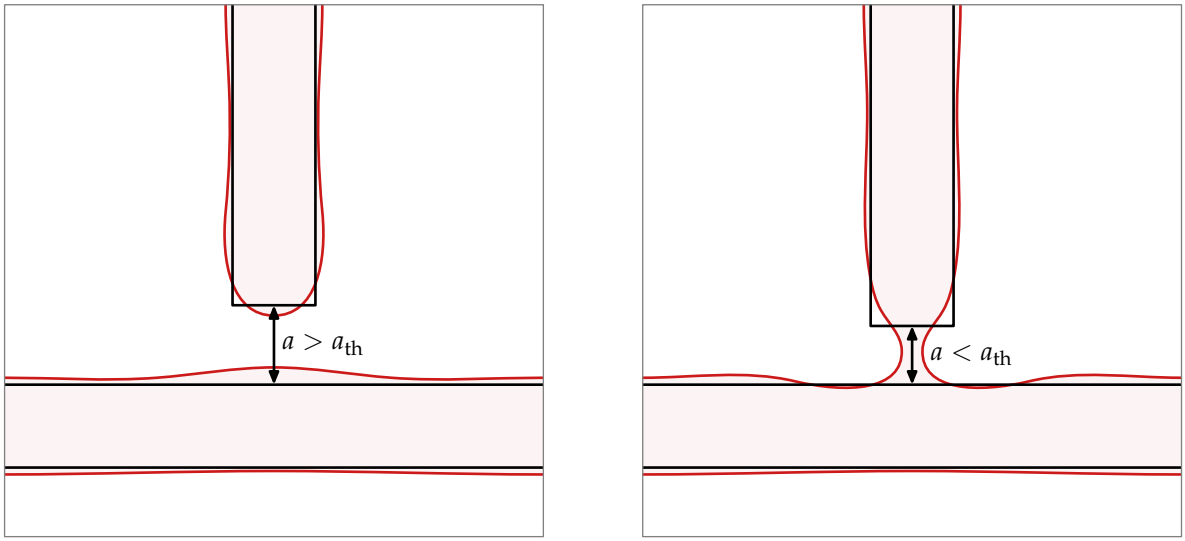

Figure 1.1: A simple design rule.

threshold, so that in the absence of other features the space would print correctly. At an older technology node, the surrounding features would have negligible effect, so that the space would still print correctly; but at newer technology nodes, the additional lines may cause bridging to occur, despite satisfying the tip-to-line design rule.

This could be fixed simply by increasing the space width threshold for the rule. However, for many tip-to-line cases such as the first isolated configuration, this would mean forcing a larger space than is really necessary for printability. Instead of sacrificing pattern density, it is preferable to define a separate rule which defines the minimum space width for this particular tip-to-line configuration. In this way, when a design rule is translated for the next technology node, it may often branch into multiple rules which are more complex.

Another consequence of printing smaller lines is that the perturbations in the printed image caused by small process variations are now more significant and potentially harmful. Figure 1.3 shows another example of a line terminating inside a U shape, but here the space width has been made sufficiently large so that, at nominal process conditions, the resulting printed image contours, shown in red, do not exhibit any bridging. However, a small change in the process conditions may cause the image contours shown in blue to be printed instead.

Thus, we can see that, for each technology node, the design rules must become more complex and specific to account for a greater number of surrounding features, and must

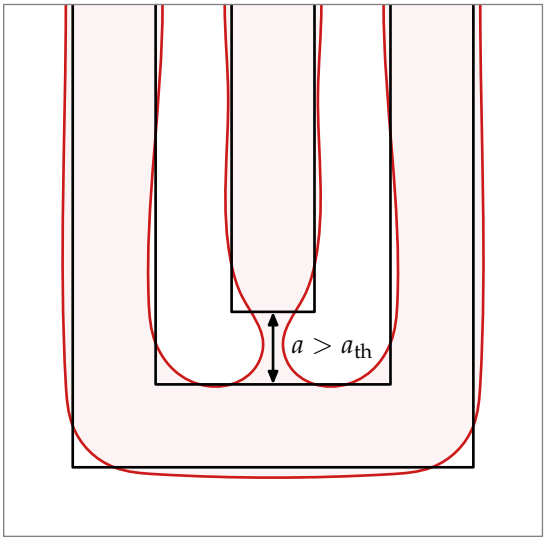

Figure 1.2: A bridging defect caused by surrounding features.

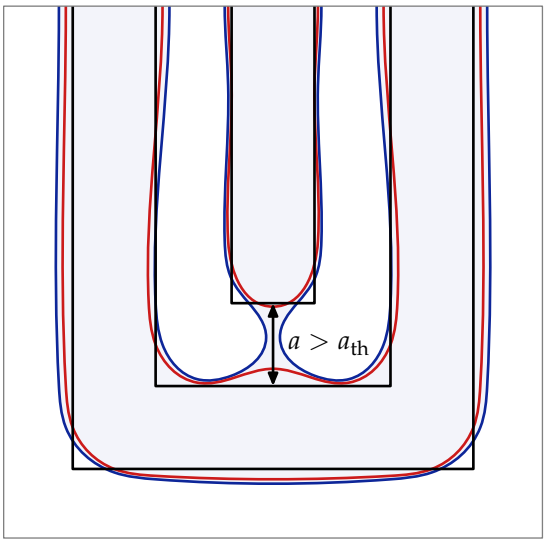

Figure 1.3: A bridging defect caused by process variation.

also take into account process variation. For these reasons, the number of required rules has become enormous, and the average number of mathematical operations per rule also increases with each node [16]. Additionally, there is a growing number of lithography technologies available, each with its own specific design rules [11]. Thus, the amount of work involved in designing and coding these rules is exploding.

One challenge in defining design rules is determining the various thresholds involved in each rule. Typically an ad hoc procedure is utilised, where many variations of the same geometry with slightly different dimensions are simulated across the process window [9]. The results are analysed in order to determine a set of thresholds. Setting the thresholds too conservatively will ensure higher yield at the cost of feature density. Balancing yield against density is difficult to achieve consistently across all design rules, and there will inevitably be some rules which are too lenient, thus allowing yield detracting patterns to slip through.

Torres [15] showed that we can speed up model-based hotspot detection by choosing to simulate only those process conditions at which critical dimensions will have their worst deviations. However, even with these runtime reductions, full-chip process window simulation is still orders of magnitude slower than design rule checking.

Hung [10] proposed a hybrid system for hotspot detection which combines design rule checking with simulation. Traditional design rule checking is first used to search the layout, but the rule thresholds are relaxed in order to achieve a higher detection rate. Since this also results in a large number of false positives being tagged, a process window simulation step follows in order to separate these from the true hotspots. Because the simulation only needs to be carried out on small regions surrounding the locations tagged by the design rule checking, this step does not take as long as a full-chip simulation. This approach has improved reliability over traditional design rule checking, but at the cost of speed, and the issue of maintaining a set of design rules is not circumvented.

Yao [18] and Xu [17] generalised the concept of design rules by defining range patterns which describe hotspots, and demonstrated an efficient algorithm for finding matches to range patterns in a layout. A range pattern is a specific two-dimensional geometric configuration of shapes whose dimensions are permitted to vary within specified constraints. While range patterns may be more flexible than traditional design rules, they still require a similar amount of effort to design and code.

Dai [5] generalised design rules further with DRC Plus, in which a specific layout clip takes the role of a design rule, and pattern matching is used the detection mechanism. The DRC Plus rules are much simpler than traditional design rules, consisting of known

bad configurations. However, it is not clear that unweighted pattern matching with some mismatch tolerance will provide accurate hotspot detection.

A method of automatically creating DRC Plus rules was proposed [4]. Process window simulations are carried out on training layouts in order to build a dataset of hotspots. Hotspots in the same geometric configuration are grouped together, and printability statistics of each group are used to determine whether the geometric configuration will be turned into a DRC Plus rule.

Our work shares some ideas with DRC Plus, in that we automatically build a catalogue of representative hotspot configurations by analysing a training dataset of hotspots, and feed this catalogue to a pattern matching tool to perform hotspot detection in new layouts. We introduce a sophisticated model-based distance metric which provides a physically meaningful comparison between hotspots. Together with an efficient clustering algorithm, this metric allows us to reduce a very large dataset to a compact catalogue of representatives. Our metric also provides weighted templates for the pattern matching tool to ensure our hotspot detection is more accurate and robust.

1.2 Outline of work

The pattern matching tool at the core of our system takes a template pattern, then searches a layout for instances of patterns which are geometrically similar to the template within some specified tolerance. In order to perform hotspot detection using this tool, we must first build a catalogue of templates and corresponding tolerances. The catalogue is constructed by analysing a large dataset of example hotspot patterns, taken from existing device layouts. The dataset is sorted into groups in such a way that patterns in the same group are considered to be the same "type" of hotspot. A representative template and corresponding tolerance can then be extracted from each group in such a way that the pattern matching tool can use these to find this type of hotspot in new layouts.

In order to determine what is meant by two hotspots being the same "type", we carry out a detailed analysis in Chapter 2 of what causes a pattern to be process sensitive. Starting from first principles, we determine the sensitivities of the position of an edge to variations in exposure and focus. Then, we quantitatively determine how these sensitivities are influenced by features in the neighbourhood of the edge. In other words, we are able to determine which features are causing a hotspot to be hot.

In Chapter 3, we use our analysis to compare the sensitivities of two hotspots. From this, we are able to define a distance metric which describes the similarity between two hotspots. This similarity is based both on the similarity of their process sensitivities, and on the geometric similarity between the features causing their process sensitivities.

Armed with this distance metric, in Chapter 4 we are able to take a dataset of hotspot clips and organise them into groups of similar hotspots using a clustering algorithm. In order to process large datasets, our clustering algorithm needs to be very efficient. We describe an incremental clustering algorithm called BUBBLE which is suitable for our needs, and streamline the algorithm further to attain a substantial improvement in runtime.

Finally, in Chapter 5, we take the clusters created by our algorithm and form our catalogue of hotspot classes. The pattern matching tool is then used to find patterns in design layouts belonging to our hotspot classes. We test our system on an actual design and are successful in achieving a high rate of hotspot detection.

Chapter 2

Hotspot sensitivity analysis

While the lithography process is tightly controlled, there is inevitably some amount of process variation present. This variation manifests itself as deviations in the printed patterns on the wafer from the nominal contours. Certain geometric configurations are more sensitive to process variation, and will demonstrate larger deviations in response; these configurations are called hotspots. In locations critical to the electrical performance of the layout, hotspots can significantly reduce the production yield.

There are various different categories of hotspots, corresponding to different modes of failure which may occur – for example, bridging, pinching, or line end shortening. However, in all cases, we can say that hotspots are caused by edge placement sensitivity to process variation. In some cases, the variation of a single edge placement may lead to a failure, and in other cases it may be the combination of two or more edge placements.

The two main sources of process variation are defocus and exposure dose error, and these are the two which we will analyse in this chapter. There are other factors, such as lens aberrations, pitch variations, or resist thickness variations, which may also be significant sources of variation, and the techniques used here can be generalised to handle these as well.

The overall aim of our work is to be able to compare and classify hotspots based upon the geometry of the features in the local neighbourhood. In order to achieve this, we must have an understanding of how the surrounding features cause a hotspot to be "hot". In this chapter, we start with a model of the lithography system and its process variations. We then study the movement of an edge placement in response to the process variations, and analyse the effect of neighbouring regions on the edge placement sensitivity. This

analysis yields a quantitative map of the contribution of features towards causing a hotspot, which will play an important role in defining similarity metrics to compare hotspots.

2.1 Lithography optics review

First, we review the optical model which is used for the lithography system. The wavelength of the exposure light is denoted by λ, and the numerical aperture of the projection system by NA. Throughout this work, a normalised scheme is employed where quantities in the spatial domain, such as coordinates (x, y) and focal error z, are normalised by $\frac{\lambda}{NA}$. (Mask plane coordinates are defined in units of their corresponding points in the image plane and normalised in the same way.) Quantities in the frequency domain such as coordinates (f, g) are normalised by $\frac{NA}{\lambda}$.

2.1.1 Hopkin's equation

Suppose we have a mask with transmission function $O(x, y)$. In our work we consider only binary masks, so that $O(x, y) = 1$ for points where light is transmitted, and $O(x, y) = 0$ otherwise. Then a corresponding aerial image intensity $I(x, y)$ is formed on the wafer plane. Hopkin's equation [1] gives the Fourier transform of the aerial image intensity as

$$\mathcal{I}(f, g) = \iint \mathcal{T}(f' + f, g' + g; f', g')\mathcal{O}(f' + f, g' + g)\mathcal{O}^*(f', g')\,\mathrm{d}f'\,\mathrm{d}g', \tag{2.1}$$

where $\mathcal{O}(f, g)$ is the Fourier transform of the mask transmission function $O(x, y)$, the asterisk $*$ denotes complex conjugation, and $\mathcal{T}(f', g'; f'', g'')$ is the transmission cross-coefficient,

$$\mathcal{T}(f', g'; f'', g'') = \iint \mathcal{J}(f, g)\mathcal{K}(f + f', g + g')\mathcal{K}^*(f + f'', g + g'')\,\mathrm{d}f\,\mathrm{d}g. \tag{2.2}$$

Here, $\mathcal{J}(f, g)$ is the source function. For conventional circular illumination, this is given by

$$\mathcal{J}(f, g) = \begin{cases} \frac{1}{\pi s^2} & \text{if } f^2 + g^2 < s^2, \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

where s is the partial coherence factor. Finally, $\mathcal{K}(f,g)$ is the pupil function. If we have zero focal error, this is equal to 1 for those frequencies which are transmitted in the pupil plane, and 0 elsewhere. So, for a circular pupil, we have

$$\mathcal{K}_0(f,g) = \begin{cases} 1 & \text{if } f^2+g^2 < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

If we have nonzero focal error, z, then the pupil function becomes

$$\mathcal{K}(f,g) = \mathcal{K}_0(f,g)e^{\mathrm{i}\pi z(f^2+g^2)}. \tag{2.5}$$

The aerial image intensity $I(x,y)$ is then obtained by taking the inverse Fourier transform of $\mathcal{I}(f,g)$. Note that this image intensity is also normalised so that a clear field mask transmission, $O(x,y) \equiv 1$, results in unity image intensity, $I(x,y) \equiv 1$, regardless of the exposure dose. The effects of exposure dose are discussed below.

2.1.2 Variational defocus expansion

In order to express the image intensity explicitly in terms of the focus error, z, Eq. (2.1) may be reformulated [19] as

$$I(x,y) = \sum_{n=0}^{\infty} z^{2n} I_{2n}(x,y). \tag{2.6}$$

The first term, $I_0(x,y)$, gives the in-focus aerial image intensity, where $z = 0$. The higher order terms, $I_{2n}(x,y)$, are called the variational aerial image intensities, and they are given in the frequency domain by

$$\mathcal{I}_n(f,g) = \iint \mathcal{T}_n(f'+f, g'+g; f', g')\mathcal{O}(f'+f, g'+g)\mathcal{O}^*(f', g')\,\mathrm{d}f'\,\mathrm{d}g', \tag{2.7}$$

where these variational transmission cross-coefficients are

$$\mathcal{T}_n(f',g';f'',g'') = \frac{(-\mathrm{i}\pi)^n}{n!}\sum_{k=0}^{n}\binom{n}{k}(-1)^k\mathcal{T}_{n,k}(f',g';f'',g''), \tag{2.8}$$

$$\mathcal{T}_{n,k}(f',g';f'',g'') = \iint \Big((f+f')^2+(g+g')^2\Big)^k\Big((f+f'')^2+(g+g'')^2\Big)^{n-k} \times \mathcal{J}(f,g)\mathcal{K}_0(f+f',g+g')\mathcal{K}_0^*(f+f'',g+g'')\,\mathrm{d}f\,\mathrm{d}g. \tag{2.9}$$

If the focus error is small, then we can neglect the z^4 and higher order terms of the infinite series (2.6) to yield the approximation

$$I(x,y) \approx I_0(x,y) + z^2 I_2(x,y). \tag{2.10}$$

2.1.3 Resist exposure threshold

We use a constant threshold model for the resist, so that the resulting contours are given by $I(x,y) = I_{\text{th}}$, where I_{th} is the resist exposure threshold, normalised to the same units as the image intensity. The accuracy of this model may be significantly improved by applying a constant bias to the image intensity contours. However, since this bias does not affect our investigation, we assume a zero bias.

Due to the normalisation, the resist exposure threshold I_{th} is actually a function of the exposure dose d, since increasing the exposure dose will also increase the normalisation factor. The relationship between the two is given by

$$I_{\text{th}} = \frac{d_0}{d}, \tag{2.11}$$

where d_0 is the dose-to-clear [2]. At a particular dose called the isofocal dose, d_{iso}, the image is least sensitive to defocus; this is the nominal exposure dose, as it yields the maximum expected process window.

2.2 Sum of coherent systems

The model presented for the lithography system is not well-suited to fast computation of the image intensity. The sum of coherent systems decomposition [3, 14] provides a much faster route to computing a high accuracy approximation of the image intensity, and it is particularly suitable for our hotspot sensitivity analysis. A singular value decomposition is used to express the transmission cross-coefficient, $\mathcal{T}(f',g';f'',g'')$, in the following form:

$$\mathcal{T}(f',g';f'',g'') = \sum_{k=1}^{\infty} \mathcal{H}_k(f',g')\mathcal{H}_k^*(f'',g''). \tag{2.12}$$

Then the in-focus aerial image intensity is given by

$$I_0(x,y) = \sum_{k=1}^{\infty} |E_k(x,y)|^2, \tag{2.13}$$

where each electrical field component $E_k(x,y)$ is given by

$$E_k(x,y) = (H_k \star O)(x,y), \tag{2.14}$$

where the star $\star$ denotes convolution. The functions $H_k(x,y)$ are called kernels. Assuming the kernels are sorted in order of decreasing L^2-norm, then we can truncate the infinite sum, Eq. (2.13), to a finite number of terms by discarding those terms such that $\|H_k(x,y)\|_2$ is smaller than some threshold. Then, we are left with the approximation

$$I_0(x,y) \approx \sum_{k=1}^{N} |E_k(x,y)|^2. \tag{2.15}$$

Assuming a sufficient number of terms are taken, this yields a very good approximation of the true image intensity.

We can decompose the variational transmission cross coefficients in an identical manner:

$$\mathcal{T}_n(f',g';f'',g'') = \sum_{k=1}^{\infty} \mathcal{H}_{n,k}(f',g')\mathcal{H}^*_{n,k}(f'',g''). \tag{2.16}$$

Then the variational aerial image intensities are given by

$$I_n(x,y) = \sum_{k=1}^{\infty} |E_{n,k}(x,y)|^2, \tag{2.17}$$

where each electrical field component $E_{n,k}(x,y)$ is given by

$$E_{n,k}(x,y) = (H_{n,k} \star O)(x,y). \tag{2.18}$$

Again, if the kernels are sorted in order of decreasing L^2-norm, then we can truncate this to a finite number of terms,

$$I_n(x,y) \approx \sum_{k=1}^{N_n} |E_{n,k}(x,y)|^2. \tag{2.19}$$

2.3 Sensitivity analysis

Now, we have a model which describes the changes in the image intensity when the focus and exposure dose stray from the nominal. We would like to investigate how this affects individual edge placement errors. To do this, we choose one point on one edge, and examine the movement of this point as the focus and exposure dose are varied.

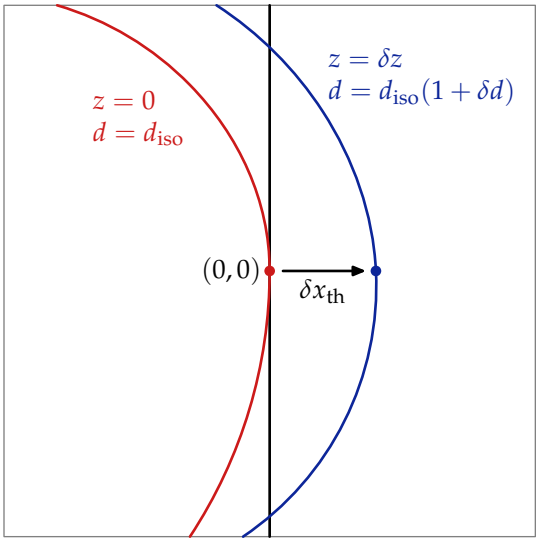

Figure 2.1: Nominal and variational intensity contours for an edge.

We consider a binary mask $O(x, y)$ which prints a vertical edge passing through the point $(0, 0)$ at the nominal focus ($z = 0$) and exposure ($d = d_{\text{iso}}$). This is shown by the red contour in Fig. 2.1. From Eq. (2.11), we have

$$I_0(0, 0) = \frac{d_0}{d_{\text{iso}}}. \tag{2.20}$$

Now we want to characterise the movement of this edge when we have some small nonzero focus error and dose error. Suppose the focus is now $z = \delta z$, and the dose is now $d = d_{\text{iso}}(1 + \delta d)$. This will cause some small perturbation of the edge contour, shown by the blue contour in Fig. 2.1. We are interested in the horizontal displacement of the contour at the point $(0, 0)$. Let δx_{th} be this horizontal displacement, so that the new contour passes through the point $(\delta x_{\text{th}}, 0)$, so we have

$$I(\delta x_{\text{th}}, 0) = \frac{d_0}{d_{\text{iso}}(1 + \delta d)}. \tag{2.21}$$

We can use the approximation for the defocus image, Eq. (2.10), to expand the left hand side. Then, using Eq. (2.20) and approximating $1/(1 + \delta d) \approx 1 - \delta d$, we get

$$I_0(\delta x_{\text{th}}, 0) + I_2(\delta x_{\text{th}}, 0)\, \delta z^2 \approx I_0(0, 0)(1 - \delta d). \tag{2.22}$$

Since δx_{th} is small, we can make the following approximations:

$$I_0(\delta x_{\text{th}}, 0) \approx I_0(0,0) + \frac{\partial I_0}{\partial x}(0,0)\,\delta x_{\text{th}}, \tag{2.23}$$

$$I_2(\delta x_{\text{th}}, 0) \approx I_2(0,0) + \frac{\partial I_2}{\partial x}(0,0)\,\delta x_{\text{th}}. \tag{2.24}$$

Therefore,

$$I_0(0,0) + \frac{\partial I_0}{\partial x}(0,0)\,\delta x_{\text{th}} + I_2(0,0)\,\delta z^2 + \frac{\partial I_2}{\partial x}(0,0)\,\delta x_{\text{th}}\,\delta z^2 \approx I_0(0,0)(1-\delta d). \tag{2.25}$$

We can neglect the highest order term, $\delta x_{\text{th}}\,\delta z^2$. Rearranging, we obtain

$$\frac{\partial I_0}{\partial x}(0,0)\,\delta x_{\text{th}} \approx -I_0(0,0)\,\delta d - I_2(0,0)\,\delta z^2. \tag{2.26}$$

We can see from this equation that we may approximate the edge placement error as the sum of two terms; one deriving from the exposure dose error, and the other from the defocus error.

Now, we define the exposure sensitivity, S, as

$$S = \frac{\partial x_{\text{th}}}{\partial d}. \tag{2.27}$$

This tells us the relationship between the exposure dose error and the edge placement error. A larger magnitude of S means that the placement of this particular edge is more sensitive to exposure dose error. A positive value means that increasing the exposure dose will move the edge in the positive x direction. From Eq. (2.26), we can see that S is given by

$$S = -\frac{I_0(0,0)}{\frac{\partial I_0}{\partial x}(0,0)}. \tag{2.28}$$

Next, since the first partial derivative $\frac{\partial x_{\text{th}}}{\partial z}$ is zero, we define the focus sensitivity, T, as

$$T = \frac{1}{2}\frac{\partial^2 x_{\text{th}}}{\partial z^2}. \tag{2.29}$$

This tells us the relationship between the defocus error and the edge placement error. A larger magnitude of T means that the placement of this particular edge is more sensitive

to defocus error. A positive value means that defocus error will move the edge in the positive x direction. From Eq. (2.26), we can see that T is given by

$$T = -\frac{I_2(0,0)}{\frac{\partial I_0}{\partial x}(0,0)}. \tag{2.30}$$

These quantities, S and T, can be each be viewed as a partial measures of the "hotness", or process sensitivity, of a hotspot.

Equation (2.26) can now be written as

$$\delta x_{\text{th}} \approx S\,\delta d + T\delta z^2. \tag{2.31}$$

2.3.1 Exposure sensitivity

The exposure sensitivity S depends upon the mask features in the context surrounding the point $(0,0)$. We would like to know in which regions the features have a greater impact on S. That is, for each point (X,Y) of the mask, we want to determine the impact of $O(X,Y)$ on this sensitivity, S. This can be quantified by the functional derivative,

$$\frac{\partial S}{\partial O(X,Y)}.$$

We call this the exposure sensitivity contribution map. Actually $O(X,Y)$ takes only binary values, but for the purpose of the analysis, we may assume it can take continuous values.

Equations (2.15) and (2.14) yield

$$I_0(0,0) = \sum_k |E_k(0,0)|^2, \tag{2.32}$$

$$E_k(0,0) = \iint O(x,y) H_k^-(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{2.33}$$

where $H_k^-(x,y)$ is a notational shorthand for $H_k(-x,-y)$. From these, we calculate

various functional derivatives,

$$\frac{\partial E_k(0,0)}{\partial O(X,Y)} = H_k^-(X,Y), \tag{2.34}$$

$$\frac{\partial I_0(0,0)}{\partial O(X,Y)} = \sum_k 2\,\mathrm{Re}[E_k(0,0)^* H_k^-(X,Y)], \tag{2.35}$$

$$\frac{\partial}{\partial O(X,Y)}\left(\frac{\partial E_k}{\partial x}(0,0)\right) = \frac{\partial H_k^-}{\partial x}(X,Y), \tag{2.36}$$

$$\frac{\partial}{\partial O(X,Y)}\left(\frac{\partial I_0}{\partial x}(0,0)\right) = \sum_k 2\,\mathrm{Re}\left[\frac{\partial E_k}{\partial x}(0,0)^* H_k^-(X,Y) + E_k(0,0)^* \frac{\partial H_k^-}{\partial x}(X,Y)\right]. \tag{2.37}$$

Finally, we can use these to calculate the exposure sensitivity contribution map by taking the functional derivative of Eq. (2.28):

$$\frac{\partial S}{\partial O(X,Y)} = \frac{I_0(0,0) \cdot \frac{\partial}{\partial O(X,Y)}\left(\frac{\partial I_0}{\partial x}(0,0)\right) - \frac{\partial I_0(0,0)}{\partial O(X,Y)} \cdot \frac{\partial I_0}{\partial x}(0,0)}{\left[\frac{\partial I_0}{\partial x}(0,0)\right]^2}. \tag{2.38}$$

Note that calculating this contribution map over a neighbourhood of $(0,0)$ is fast. It can be seen that it is the weighted sum of the kernels and their spatial derivatives, which can all be precalculated. Thus, for a given mask, we need only compute the scalar weighting coefficients of these functions. These coefficients are functions of the electrical field components $E_k(0,0)$ and their spatial derivatives $\frac{\partial E_k}{\partial x}(0,0)$; the cost of calculating these is equivalent to the cost of simulating the nominal aerial image and one spatial derivative at a point.

The exposure sensitivity contribution map tells us which features of the mask are contributing to the exposure sensitivity, which are detracting, and which have little effect. For example, suppose that, within some region of the mask, $O(X,Y) = 1$ and $\frac{\partial S}{\partial O(X,Y)}$ is large and positive. This means that this feature is causing S to be more positive. In the case that $S > 0$, this feature is contributing to the exposure sensitivity at $(0,0)$. If $S < 0$, then this feature is reducing the exposure sensitivity.

Figure 2.2 shows an example of an exposure sensitivity contribution map for a lithography system with coherent illumination. The red regions show a positive contribution, the blue regions show negative contribution, and the yellow regions show zero contribution. It can be seen that the regions of the mask most heavily affecting the exposure sensitivity of the edge placement are those closest to the edge. The contribution map tends to zero as we get further away from the origin, and outside a certain radius we can say that the effects of features on the exposure sensitivity are negligible.

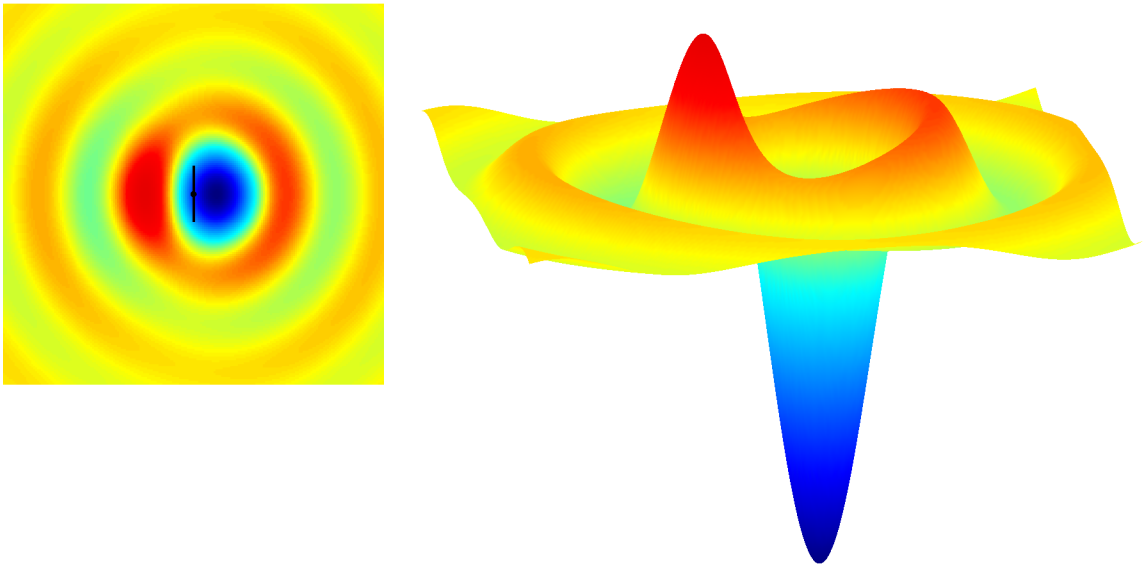

Figure 2.2: An example of an exposure sensitivity contribution map.

2.3.2 Focus sensitivity

Next, for each point (X, Y) of the mask, we want to determine the impact of $O(X, Y)$ on the focus sensitivity, T. This can be quantified by the functional derivative,

$$\frac{\partial T}{\partial O(X, Y)}.$$

We call this the focus sensitivity contribution map.

Equations (2.19) and (2.18) yield

$$I_2(0,0) = \sum_k |E_{k,2}(0,0)|^2, \tag{2.39}$$

$$E_{2,k}(0,0) = \iint O(x,y) H^-_{2,k}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{2.40}$$

From these, we calculate various functional derivatives,

$$\frac{\partial E_{2,k}(0,0)}{\partial O(X,Y)} = H^{-}_{2,k}(x,y), \tag{2.41}$$

$$\frac{\partial I_2(0,0)}{\partial O(X,Y)} = \sum_k 2\,\mathrm{Re}[E_{2,k}(0,0)^* H^{-}_{2,k}(X,Y)]. \tag{2.42}$$

Finally, we can use these to calculate the focus sensitivity contribution map by taking the functional derivative of Eq. (2.30):

$$\frac{\partial T}{\partial O(X,Y)} = \frac{2\,\mathrm{Re}\left[I_2(0,0)\cdot \frac{\partial}{\partial O(X,Y)}\left(\frac{\partial I_0}{\partial x}(0,0)\right) - \frac{\partial I_2(0,0)}{\partial O(X,Y)}\cdot \frac{\partial I_0}{\partial x}(0,0)\right]}{\left[\frac{\partial I_0}{\partial x}(0,0)\right]^2}. \tag{2.43}$$

Again, note that calculating this contribution map over a neighbourhood of $(0,0)$ is fast. As for the exposure sensitivity contribution map, it is the weighted sum of the kernels and their spatial derivatives. The scalar weighting coefficients are again functions of $E_k(0,0)$ and $\frac{\partial E_k}{\partial x}(0,0)$, and for the focus sensitivity contribution map we also require the variational electrical field components $E_{2,k}(0,0)$; calculating these is equivalent to the cost of simulating the variational aerial image at a point.

The focus sensitivity contribution map tells us which features of the mask are contributing to the focus sensitivity. Figure 2.3 shows an example of a focus sensitivity contribution map for a lithography system with coherent illumination. As for the exposure sensitivity contribution map, the focus sensitivity contribution map tends to zero as we get further away from the origin, and outside a certain radius we can say that the effects of features on the focus sensitivity are negligible.

2.4 Overall process sensitivity

Thus far we have considered the exposure and focus sensitivities separately. However, in practice, we are interested in the overall process sensitivity of an edge placement. Therefore, we will now combine these two sensitivities into a single overall measure of an edge placement process sensitivity, and calculate the corresponding contribution map. In order to determine how to weight the exposure sensitivity and the focus sensitivity, we need to examine the process model and utilise the uncertainties in the exposure dose and the focus.

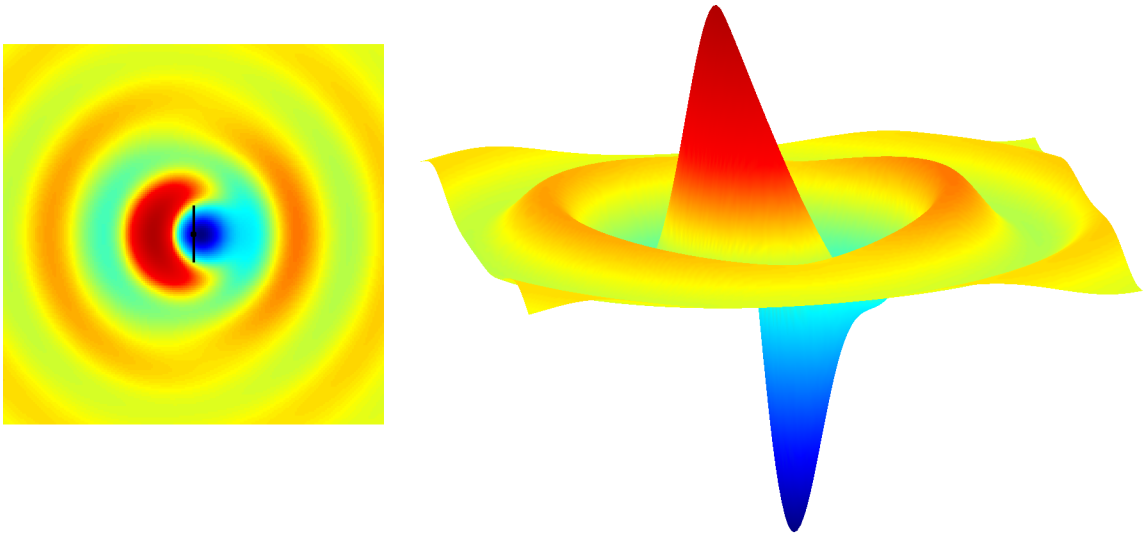

Figure 2.3: An example of a focus sensitivity contribution map.

Suppose d and z are independently normally distributed, and their variances σ_d^2 and σ_z^2 are known, $\delta d \sim \mathcal{N}(0, \sigma_d^2)$ and $\delta z \sim \mathcal{N}(0, \sigma_z^2)$. Now we want to investigate the statistical properties of x_{th}. Starting from Eq. (2.31), the mean of x_{th} is

$$\begin{aligned}
\mu_{x_{\text{th}}} &= E[\delta x_{\text{th}}] \\
&\approx E\left[S\,\delta d + T\,\delta z^2\right] \\
&= S\,E[\delta d] + T\,E\left[\delta z^2\right] \\
&= T\sigma_z^2.
\end{aligned} \tag{2.44}$$

Next, the variance of x_{th} is

$$\begin{aligned}
\sigma_{x_{\text{th}}}^2 &= E\left[\delta x_{\text{th}}^2\right] - E[\delta x_{\text{th}}]^2 \\
&\approx E\left[\left(S\,\delta d + T\,\delta z^2\right)^2\right] - \left(T\sigma_z^2\right)^2 \\
&= \left(S^2 E\left[\delta d^2\right] + T^2 E\left[\delta z^4\right] + 2ST\,E[\delta d]\,E\left[\delta z^2\right]\right) - T^2\sigma_z^4 \\
&= \left(S^2\sigma_d^2 + 3T^2\sigma_z^4 + 0\right) - T^2\sigma_z^4 \\
&= S^2\sigma_d^2 + 2T^2\sigma_z^4.
\end{aligned} \tag{2.45}$$

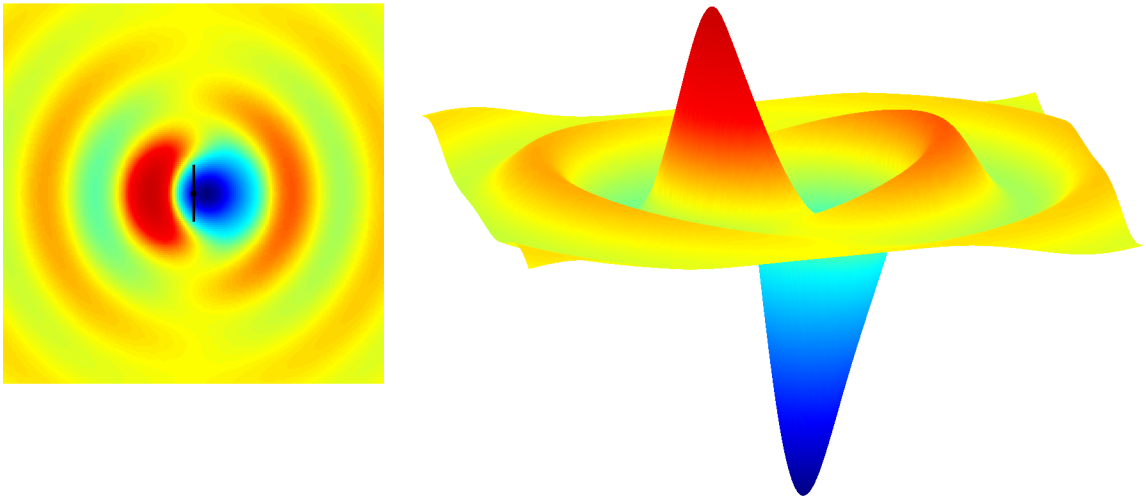

Figure 2.4: An example of an edge placement variance contribution map.

This variance can be used as a measure of the overall edge placement process sensitivity.

Now we would like to determine an overall contribution map which will quantify the effect of each region of the mask on the edge placement variance, $\sigma^2_{x_{\text{th}}}$. This can be found by taking the functional derivative of Eq. (2.45),

$$\frac{\partial \sigma^2_{x_{\text{th}}}}{\partial O(x,y)} \approx 2\sigma_d^2 S \frac{\partial S}{\partial O(x,y)} + 4\sigma_z^4 T \frac{\partial T}{\partial O(x,y)}. \tag{2.46}$$

Figure 2.4 shows an example of an edge placement variance contribution map for a lithography system with coherent illumination.

2.5 Generalised sensitivities

Investigating the variation in a single edge placement is useful when we are looking at line end shortening. If we are instead interested in the process sensitivity of a line width or space width, then we can generalise the above analysis to consider multiple edge placements simultaneously.

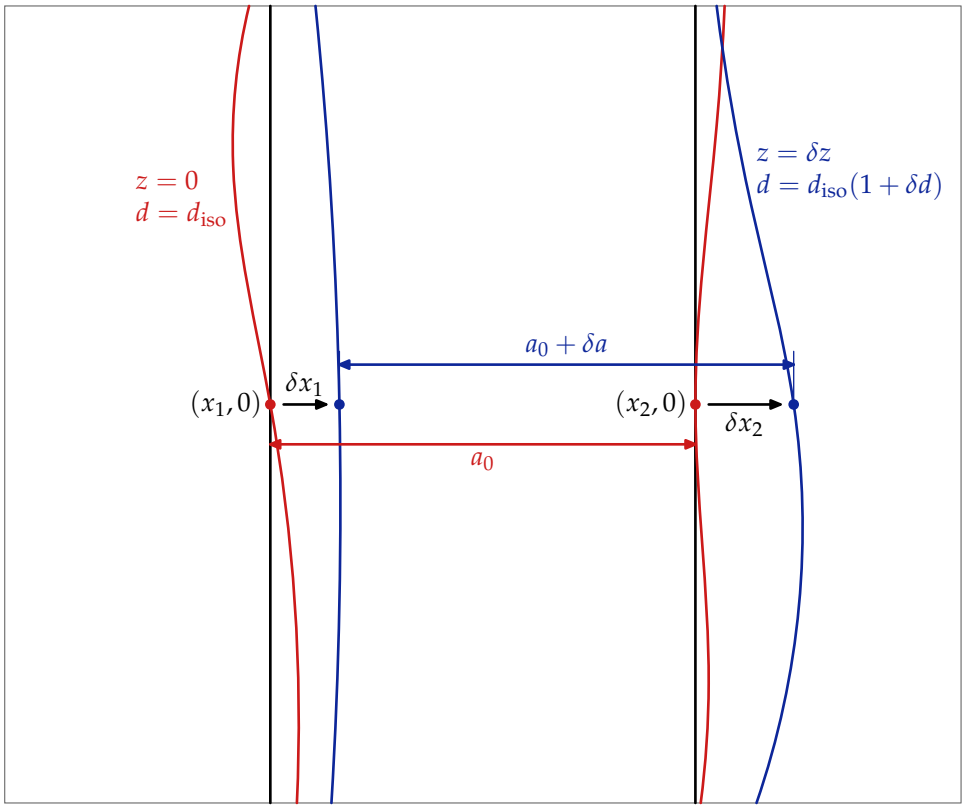

Figure 2.5: Nominal and variational intensity contours for the two edges of a line.

First, let us investigate the case for pinching hotspots. Suppose we have a mask $O(x, y)$ which prints a vertical line whose left and right edges pass through the points $(x_1, 0)$ and $(x_2, 0)$ respectively at nominal focus and exposure. Therefore, the nominal line width is $a_0 = x_2 - x_1$. This is shown by the red contours in Fig. 2.5.

Next, suppose the focus is now $z = \delta z$, and the dose is now $d = d_{\text{iso}}(1 + \delta d)$. This will cause some small perturbations of the two edge contours, shown by the blue contours in Fig. 2.5. We are interested in the horizontal displacements of the contours at $(x_1, 0)$ and $(x_2, 0)$. Let δx_1 be the horizontal displacement of the first contour at $(x_1, 0)$, and δx_2 be the horizontal displacement of the second contour at $(x_2, 0)$. Then the line width is now $a_0 + \delta a$, with $\delta a = \delta x_2 - \delta x_1$. We let S_1 and S_2 be the respective exposure sensitivities of the two edge placements, and T_1 and T_2 be the respective focus sensitivities.

Now we can investigate the statistical properties of a, as we did in the previous section

with x_{th}. The mean of a is

$$\begin{aligned}\mu_a &= E[a] \\ &= a_0 + E[\delta a] \\ &= a_0 + E[\delta x_2] - E[\delta x_1] \\ &\approx a_0 + (T_2 - T_1)\sigma_z^2. \end{aligned} \tag{2.47}$$

Next, the variance of a is

$$\begin{aligned}\sigma_a^2 &= E\left[\delta a^2\right] - E[\delta a]^2 \\ &= E\left[(\delta x_2 - \delta x_1)^2\right] - (E[\delta x_2] - E[\delta x_1])^2 \\ &\approx E\left[\left((S_2 - S_1)\,\delta d + (T_2 - T_1)\,\delta z^2\right)^2\right] - \left((T_2 - T_1)\sigma_z^2\right)^2 \\ &= (S_2 - S_1)^2\sigma_d^2 + 2(T_2 - T_1)^2\sigma_z^4. \end{aligned} \tag{2.48}$$

This shows that, if the two edge placements have identical exposure and focus sensitivities, then the line width a will be exactly constant. This is intuitive, since this means that the two edges will have identical displacements in response to exposure and focus errors.

Now we would like to determine an overall contribution map which will quantify the effect of each region of the mask on the line width variance, σ_a^2. This can be found by taking the functional derivative of Eq. (2.48),

$$\begin{aligned}\frac{\partial \sigma_a^2}{\partial O(x,y)} \approx\ & 2\sigma_d^2(S_2 - S_1)\left(\frac{\partial S_2}{\partial O(x,y)} - \frac{\partial S_1}{\partial O(x,y)}\right) \\ & + 4\sigma_z^4(T_2 - T_1)\left(\frac{\partial T_2}{\partial O(x,y)} - \frac{\partial T_1}{\partial O(x,y)}\right). \end{aligned} \tag{2.49}$$

The procedure for examining the variation of the width of a space is exactly the same, so we can apply the same formulae to bridging hotspots.

We can generalise this even further to consider more edge placements. For example, we may be interested in the variation in the area of a contact, in which case we would consider four edge placements.

Chapter 3

Distance metric

In this chapter, we design a distance metric which may be used to compare two mask patterns. The ultimate goal of this distance metric is to let us decide which patterns are the same "type" of hotspot. To this end, we consider two factors in our concept of "similarity". First of all, similar patterns should have a similar process sensitivity. Secondly, their process sensitivity should be caused by geometrically similar features in the mask patterns.

In order to compare the process sensitivity of two patterns, we could perform lithography simulations of the two patterns to yield an exact measure of the process sensitivities of each, using the formulae derived in the previous chapter. However, this is extremely computationally expensive. Instead, we perform variational analysis of the expressions for process sensitivity, and obtain an efficient formula for estimating the difference between the process sensitivities of the two patterns. Next, we adapt this formula to incorporate a measure of geometric similarity between the features in the two patterns, and this becomes our distance metric.

In order to make our computation of this distance metric as efficient as possible, we consider only small clips from masks surrounding the point of interest, and show that this reduction has negligible impact on our distance metric. Other methods of improving the computational speed of the distance metric are also discussed.

3.1 Variance prediction

Suppose we have a mask, $O_1(x, y)$, containing a particular dimension of interest, a_1, such as a line end position, or a space width. The variance of this dimension represents the "hotness" of this feature. Suppose this variance, $\sigma_{a_1}^2$, is known. Then the variance contribution map may be used to extrapolate variances of similar dimensions in other masks.

Consider a second mask, $O_2(x, y)$ with an identical dimension of interest, a_2, located at the same coordinates, but with a different surrounding context. That is, if a_1 represents the horizontal position of a line end which nominally passes through $(0, 0)$ in the first mask, then a_2 represents the horizontal position of a line end which nominally passes through $(0, 0)$ in the second mask. We can consider all masks containing the same feature at the same coordinates to belong to the same category.

Note that, when we refer to a "second" mask, this does not imply that the mask is necessarily distinct from the initial mask. Typically, the second mask is simply a recoordinatisation $O_2(x, y) = O_1(x + \Delta x, y + \Delta y)$ so that we can compare two features in different areas of the same mask.

Regarding the second mask as a perturbation of the first mask, a first order approximation yields:

$$\sigma_{a_2}^2 \approx \sigma_{a_1}^2 + \iint \frac{\partial \sigma_{a_1}^2}{\partial O_1(x, y)} \left[O_2(x, y) - O_1(x, y) \right] \mathrm{d}x \, \mathrm{d}y. \tag{3.1}$$

Thus the difference between the variances of the dimensions between the two masks is given by the integral over the plane of the mask "perturbation", $[O_2(x, y) - O_1(x, y)]$, weighted by the variance contribution map. The accuracy of these predictions depends upon the similarity between the two masks. If the two masks differ in only a few small regions so that the perturbation is zero almost everywhere, then the predictions can be very accurate.

Figure 3.1 shows the result of using the variance of a reference mask dimension to predict variances of the corresponding dimension in 200 other masks. In this case, the variance of a line end position in a reference mask was determined, and the variance contribution map was calculated using Eq. (2.46). Then, Eq. (3.1) was used to predict the variances of other line end positions based on the reference variance. These predicted variances were compared to the variances calculated individually for each of the line ends using Eq. (2.46).

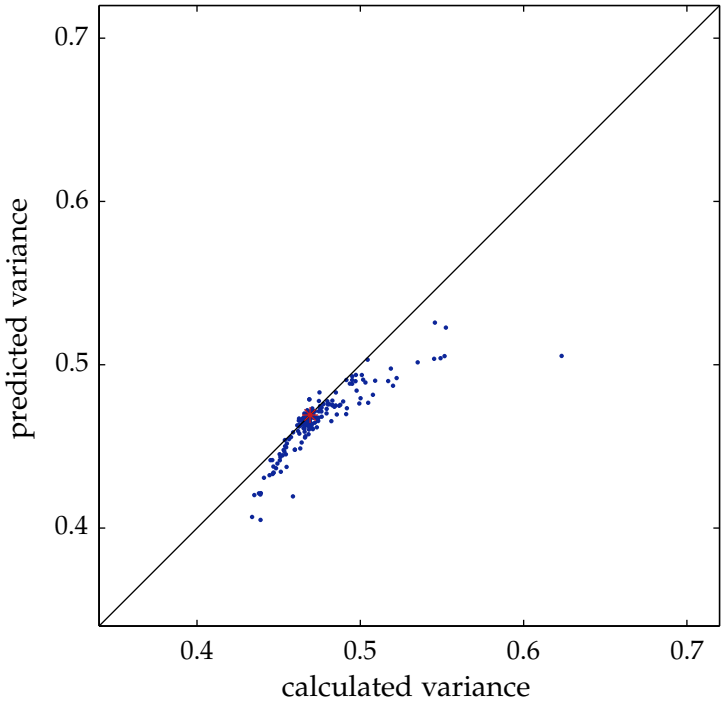

Figure 3.1: Accuracy of the variance prediction formula, Eq (3.1).

In the figure, the red asterisk shows the variance of the line end position in the reference mask. The blue dots show the predicted and calculated variances in 200 other masks. It can be seen that, for masks with very similar variances, Eq. (3.1) provides an accurate prediction of the actual variance. For masks which are less similar, the prediction is less accurate. Similar results are obtained for other choices of reference mask.

3.2 Comparing hotspots

The preceding section has given us a method of estimating the difference between the variances of two features of the same category in different masks:

$$\left|\sigma_{a_2}^2 - \sigma_{a_1}^2\right| \approx \left| \iint \frac{\partial \sigma_{a_1}^2}{\partial O_1(x,y)} \left[O_2(x,y) - O_1(x,y)\right] \mathrm{d}x \, \mathrm{d}y \right|. \tag{3.2}$$

However, although we may use this formula to estimate whether two patterns have similar "hotness", it does not tell us much about whether the two patterns themselves are geometrically similar. Two patterns which have completely different features surrounding their hotspot may nevertheless produce exactly the same line end variance. We would like to design a function which has a small value only if two patterns have a similar dimension variance *and* they have geometrically similar features.

We take the approximation, Eq. (3.2), and bound it above:

$$\begin{aligned} \left|\sigma_{a_2}^2 - \sigma_{a_1}^2\right| &\approx \left| \iint \frac{\partial \sigma_{a_1}^2}{\partial O_1(x,y)} \left[O_2(x,y) - O_1(x,y)\right] \mathrm{d}x \, \mathrm{d}y \right| \\ &\leq \iint \left| \frac{\partial \sigma_{a_1}^2}{\partial O_1(x,y)} \right| \left| O_2(x,y) - O_1(x,y) \right| \mathrm{d}x \, \mathrm{d}y \\ &= \iint\limits_{O_1 \neq O_2} \left| \frac{\partial \sigma_{a_1}^2}{\partial O_1(x,y)} \right| \mathrm{d}A, \end{aligned} \tag{3.3}$$

where the last integral is taken over the differing area between the two masks – that is, the set of points (x, y) such that $O_1(x, y) \neq O_2(x, y)$. These last two lines are equal since we only consider binary masks, so that $|O_2(x, y) - O_1(x, y)| = 1$ iff $O_1(x, y) \neq O_2(x, y)$. Figure 3.2 shows an example of the differing area between two masks.

The last line of Eq. (3.3) can be interpreted as a weighted distance between the two masks (since the unweighted distance between the two masks is simply the standard area of the

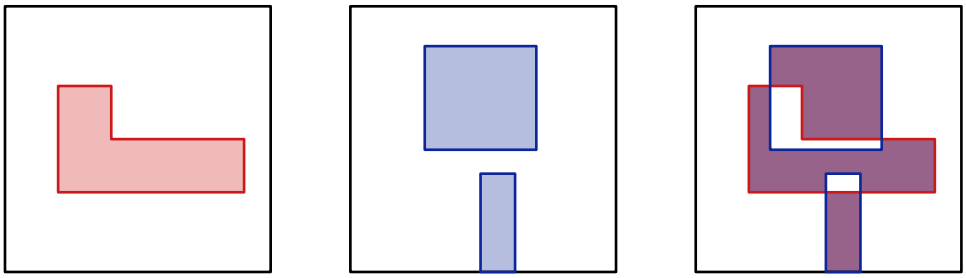

Figure 3.2: Two mask clips and the area where the two masks differ.

difference, $\int\int_{O_1 \neq O_2} \mathrm{d}A$). Since the weighting is the magnitude of the variance contribution map, it emphasises those regions which have a greater effect on the variance of interest. Therefore, regions far away from the pertinent feature are given negligible weighting, so only the immediate surrounding context of the feature is considered.

Note that this inequality in the second line of Eq. (3.3) relies upon the first order approximation in the first line being accurate. The accuracy depends upon the neglected second and higher order terms being small. We expect this to be true when the differing area is small. In fact, we expect the approximation to be accurate when the last line of Eq. (3.3), the weighted distance, is small.

This is demonstrated in Fig. 3.3. As in Fig. 3.1, the variance of a line end position in a reference mask was used to predict the corresponding variances in 200 other masks using Eq. (3.1). Then the variance prediction error was plotted against the weighted distance between the masks.

It can be seen that, when the weighted distance between the masks is small, the variance prediction is very accurate. As the weighted distance increases, the variance prediction error may increase. Thus, we see that the variance prediction will only be inaccurate in cases where there is a significant difference between two masks. Similar results are obtained for other choices of reference mask.

In summary, when the weighted distance is small, the approximation in the first line of Eq. (3.3) will be very accurate, so that the weighted distance is a true upper bound of the difference between the variances of the two masks. Therefore, the two masks must have very similar variances. Additionally, the weighted distance itself is a description of the geometric similarity between the two patterns. So this weighted distance serves our two criteria for comparing patterns perfectly.

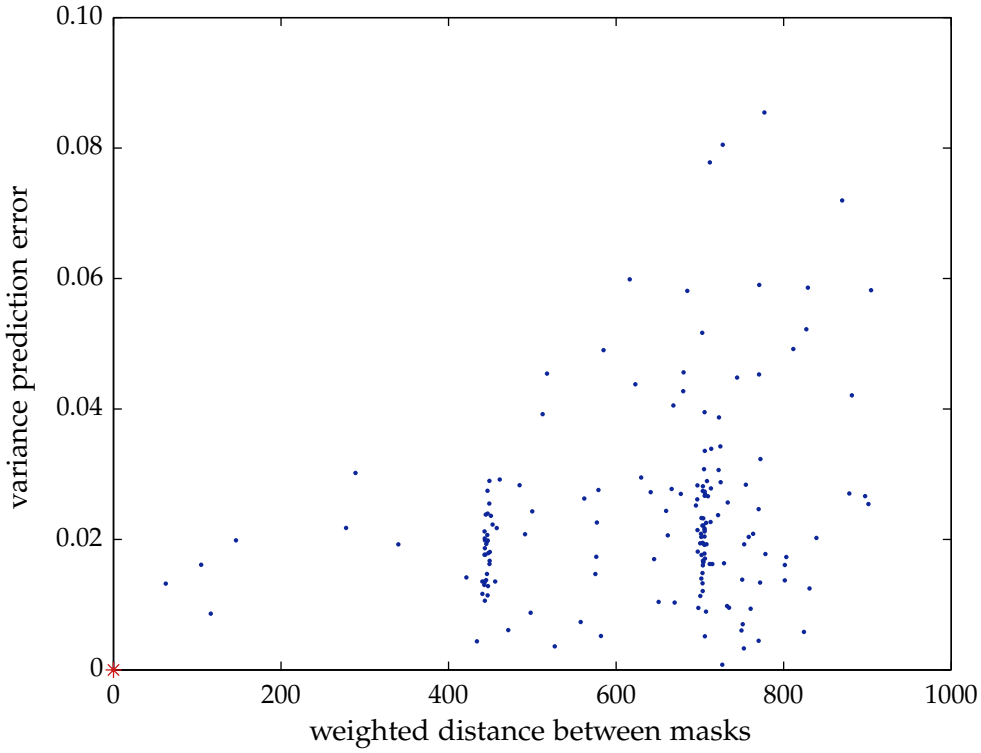

Figure 3.3: Variance prediction error against weighted distance.

3.3 General weighting functions

From our analysis in the preceding section, we now define the function

$$f(O_1, O_2) = \left[\iint\limits_{O_1 \neq O_2} w_{O_1}(x, y) \, \mathrm{d}A \right]^{\frac{1}{2}}, \tag{3.4}$$

where $w_{O_1}(x, y)$ is a weighting function given by

$$w_{O_1}(x, y) = \left| \frac{\partial \sigma_{a_1}^2}{\partial O_1(x, y)} \right|. \tag{3.5}$$

The function f is specific to the category and coordinates of the feature whose dimension, a, is of interest, and both arguments of the function, O_1 and O_2, must contain this feature at the correct location, or else the function is not meaningful.

We have shown that this function is useful for comparing hotspot patterns. However, there are two reasons that this function is not suitable as a metric. First, the function is not symmetric, since we arbitrarily choose to use one of the masks as our reference, and the weighting function is based upon this mask. Second, it will be extremely slow to calculate, since we have to compute a different weighting function for every mask. In order to combat both of these problems, we would like to have a single weighting function which we can use for all hotspots patterns of the same category. To investigate whether this is feasible, we examine the shape of our mask-dependent weighting functions.

Figure 3.4 shows the variance contribution maps for a line end position in two different masks. We can see that the contribution maps are almost identical. Figure 3.5 shows the $y = 0$ cross-section of the variance contribution maps for a line end position in 50 different masks. They have almost identical forms, with the chief difference between them being only a small variation in magnitude. Thus it can be seen that, for mask variances of this category, the contribution map is almost constant. Similar investigations for other hotspot categories yield the same conclusion.

Therefore we can use a single weighting function for all masks in the same category without significant loss of accuracy. We define a new function,

$$\rho(O_1, O_2) = \left[\iint\limits_{O_1 \neq O_2} w(x, y) \, \mathrm{d}A \right]^{\frac{1}{2}}, \tag{3.6}$$

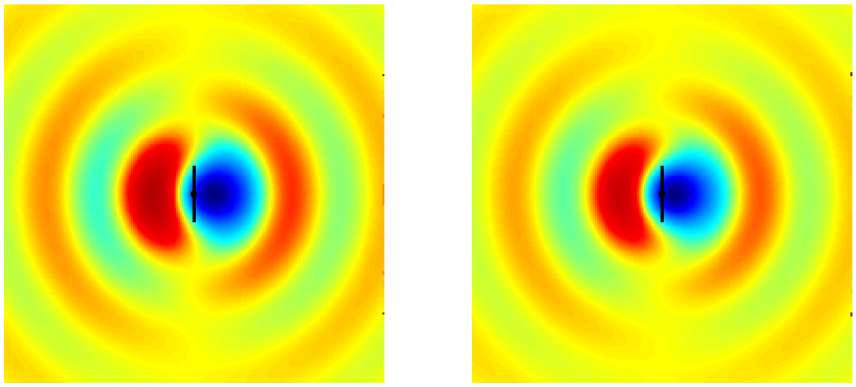

Figure 3.4: Edge placement variance contribution maps for two different clips.

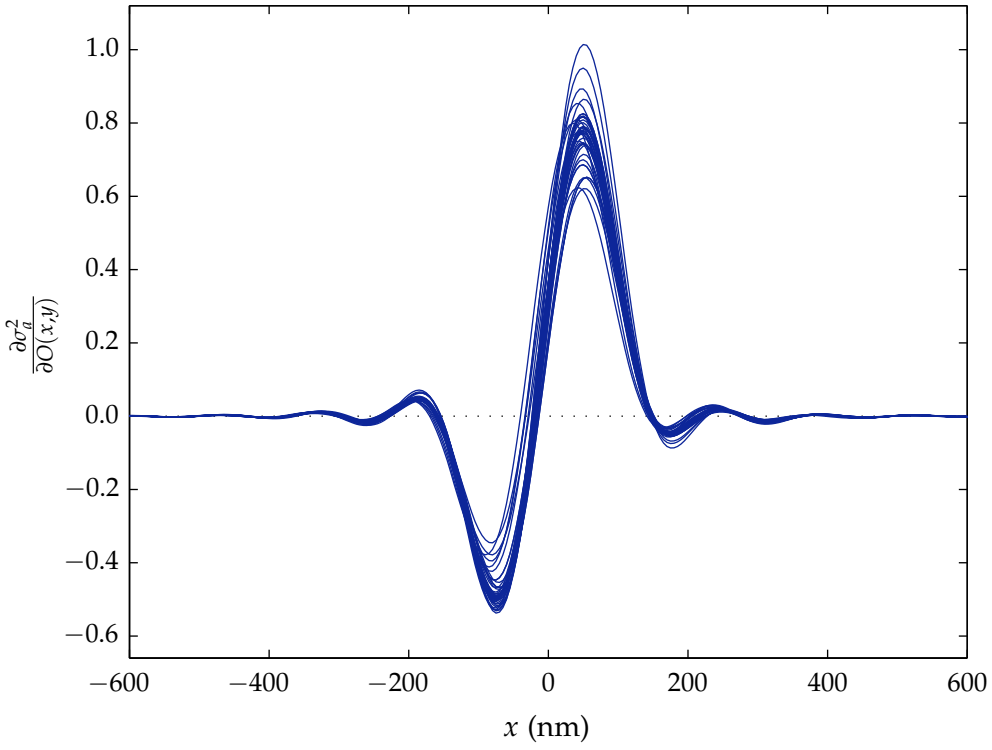

Figure 3.5: Cross sections of variance contribution maps for 50 different clips.

where $w(x,y)$ is a pattern-independent weighting function. We determine this new weighting function by taking a large number, N, of masks, $O_i(x,y)$, in the same category and taking the root-mean-square value of their pattern-dependent weighting functions:

$$w(x,y) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial\sigma_{a_i}^2}{\partial O_i(x,y)}\right)^2\right]^{\frac{1}{2}}. \tag{3.7}$$

Since the weighting function $w(x,y)$ is positive almost everywhere, it is easy to show that the function $\rho(O_1,O_2)$ is a metric.

Using a single weighting function for all masks in the same category vastly reduces the runtime of our distance metric computations for clustering, as will be shown in Section 3.6. However, once we have completed our clustering and reduced our dataset to a smaller set of representative patterns, we can then use the pattern-specific weighting functions for pattern matching to provide increased detection accuracy without a significant runtime penalty. This will be discussed further in Chapter 5.

3.4 Rotations and reflections

A limitation of the metric defined above is that two masks may be considered far apart with respect to the metric, even if they have very similar patterns except that one is a rotation or a reflection of the other. Assuming that the lithography system treats the horizontal and vertical directions equally (as is the case for many illumination conditions), this is undesirable. Since the two patterns will print similarly and have a similar sensitivity to process variation, the distance between them should be small.

Therefore, the following pseudometric is defined:

$$\hat{\rho}(O_1,O_2) = \min_{\tau\in D_4}\rho\big(O_1,\tau(O_2)\big), \tag{3.8}$$

where D_4 is the dihedral symmetry group of eight transformations (corresponding to the eight symmetries of a square): the four rotations of the pattern, and the four rotations of the reflection of the pattern (see Fig. 3.6). Thus, if O_1 is a reflection or rotation of O_2, then $\hat{\rho}(O_1,O_2)$ will be zero. (This is a pseudometric since $\hat{\rho}(O_1,O_2)=0$ only guarantees O_1 and O_2 are equivalent up to transformations, but not necessarily identical.)

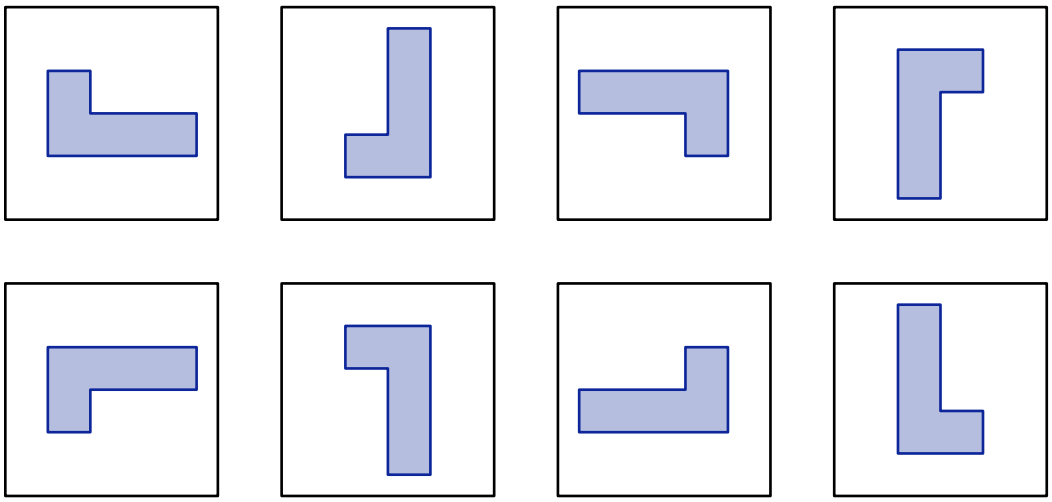

Figure 3.6: The eight transformations of a clip.

If we have an illumination source where the horizontal and vertical directions are unequal, such as a dipole source, then we can replace the symmetry group D_4 by D_2 in the definition of the function, $\hat{\rho}$.

The pseudometric, $\hat{\rho}$, serves as the distance metric which we will use in the remainder of this work to compare patterns.

3.5 Clips

The integral in the definition of our distance metric is taken over the entire plane. In practice, however, it is unnecessary to integrate over the entire plane, since the magnitude of the weighting function is negligible outside a neighbourhood of the feature of interest. Thus, we can achieve sufficient accuracy by considering only a smaller region surrounding the feature.

For the purpose of easy computation, we reduce each mask to a square clip centred about the feature. Define S_R as the square region defined by $x, y \in [-R, R]$. We can choose the radius R by examining the weighting function, $w(x, y)$, as follows. Once we have limited

our masks to the region S_R, the maximum value of our distance metric is given by

$$\rho_{\max} = \left[\iint_{S_R} w(x,y)\,\mathrm{d}A \right]^{\frac{1}{2}}. \tag{3.9}$$

This maximum is achieved when $O_1 \neq O_2$ throughout the region S_R. The distance metric will also have some error caused by ignoring the masks outside S_R. The maximum value of this error is given by

$$\tilde{\rho}_{\max} = \left[\iint_{S_R'} w(x,y)\,\mathrm{d}A \right]^{\frac{1}{2}}, \tag{3.10}$$

where S_R' is the complement of S_R. This maximum error is achieved when $O_1 = O_2$ throughout S_R, but $O_1 \neq O_2$ elsewhere. We would like the ratio $\tilde{\rho}_{\max}/\rho_{\max}$ to be small enough that we may consider the error negligible. This ratio is monotonically decreasing with R, so we can choose the radius R according to an error ratio threshold that we choose.

For each mask, $O_i(x,y)$, we define a corresponding clip, $\Theta_i(x,y)$, which is equal to the mask within the restricted domain S_R:

$$\Theta_i(x,y) = \begin{cases} O_i(x,y) & \text{if } (x,y) \in S_R, \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Our distance metric for clips is now

$$\hat{\rho}(\Theta_1,\Theta_2) = \min_{\tau \in D_4} \rho\big(\Theta_1, \tau(\Theta_2)\big), \tag{3.12}$$

where

$$\rho(\Theta_1,\Theta_2) = \left[\iint_{\Theta_1 \neq \Theta_2} w(x,y)\,\mathrm{d}A \right]^{\frac{1}{2}}, \tag{3.13}$$

and the region of integration is guaranteed to be a subset of S_R.

Figure 3.7 shows two examples of hotspot clips. The red line shows the location of the hotspot. The first clip is centred around a bridging hotspot. The second clip is centred around a pinching hotspot, and the location of the hotspot is taken as the position where the line width error is greatest.

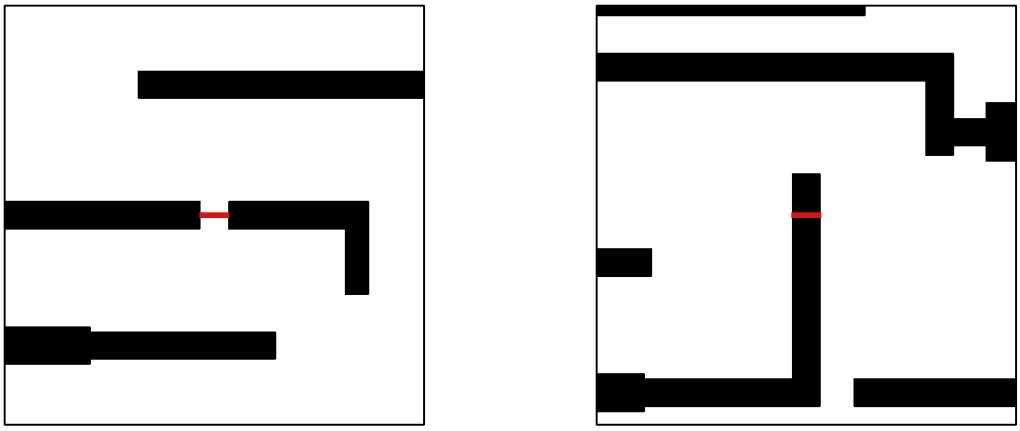

Figure 3.7: Two hotspot clips with dimensions 1130 nm × 1130 nm.

3.6 Implementation

In order to perform clustering, the distance metric needs to be calculated for many pairs of clips. Therefore it is necessary to make this calculation as efficient as possible.

To calculate the distance metric, Eq. (3.12), definite integrals of the weighting function must be calculated over several polygons. However, even rough approximations of the definite integrals are computationally expensive, and calculating them several times for each pair of clips becomes prohibitively slow.

This problem can be resolved by pre-integrating the weighting function. An efficient algorithm for computing the distance metric utilising pre-integration is presented here:

1. The weighting function is pre-integrated over the domain of the clips, S_R.
2. Each clip pattern is decomposed into a set of disjoint rectangles.
3. The (weighted) area of each clip pattern is computed.
4. For each pair of clips, the (weighted) area of the overlap is computed for each rotation/reflection.
5. The maximum overlap is chosen and used to obtain the value of the distance metric between them.

Each step is elaborated upon below.

A grid of points is created over the domain S_R. For each gridpoint (x', y'), the definite integral

$$W(x', y') = \int_0^{x'} \int_0^{y'} w(x, y) \, \mathrm{d}y \, \mathrm{d}x \tag{3.14}$$

is calculated and stored in a look-up table. (Note that the number of definite integrals calculated here is far less than the number which would be calculated if the distance for each pair of clips was calculated separately. Symmetry may also be used to further decrease the number of integrals calculated.) Then, using this table, definite integrals of the weighting function can be easily calculated over arbitrary rectangles within the clip, since they can be expressed as a sum/difference of values of W in the look-up table:

$$\begin{aligned} &\int_{x_1}^{x_2} \int_{y_1}^{y_2} w(x, y) \, \mathrm{d}y \, \mathrm{d}x \\ &= W(x_1, y_1) - W(x_1, y_2) + W(x_2, y_2) - W(x_2, y_1). \end{aligned} \tag{3.15}$$

If the vertices of the rectangle do not coincide with the grid points at which the weighting function was pre-integrated, the values of W can be approximated using linear interpolation between the values at the closest grid points.

Next, we need to be able to determine the regions where two clips differ. Each clip is stored in memory as a set of polygons, each polygon being stored as a list of vertex coordinates. Each polygon represents the outside boundary of the dark field portions of the pattern within the clip, and the remaining regions are light field. In fact, we can equivalently represent each clip as a set of disjoint rectangles, by decomposing each polygon into rectangles. (This is possible assuming the polygons in the layout are all right-angled. There are many algorithms to perform such decompositions, such as Gourley and Green's [8].) An example is shown in Fig. 3.8.

Suppose clip Θ_1 is represented by rectangles $\{A_1, A_2, \ldots, A_m\}$. The weighted area of each rectangle, $|A_i|$, can be found by Eq. (3.15), so we can find the total weighted area of the clip as $|\Theta_1| = \sum_{i=1}^{m} |A_i|$ (where $|\cdot|$ represents weighted area). Next suppose clip Θ_2 is represented by rectangles $\{B_1, B_2, \ldots, B_n\}$, and we want to find the total weighted area where the two clips differ. First we find the weighted area of the overlap between the rectangles of clip Θ_1 and the rectangles of clip Θ_2,

$$|\Theta_1 \cap \Theta_2| = \sum_{i=1}^{m} \sum_{j=1}^{n} \left| A_i \cap B_j \right|. \tag{3.16}$$

This is demonstrated diagrammatically in Fig. 3.9.

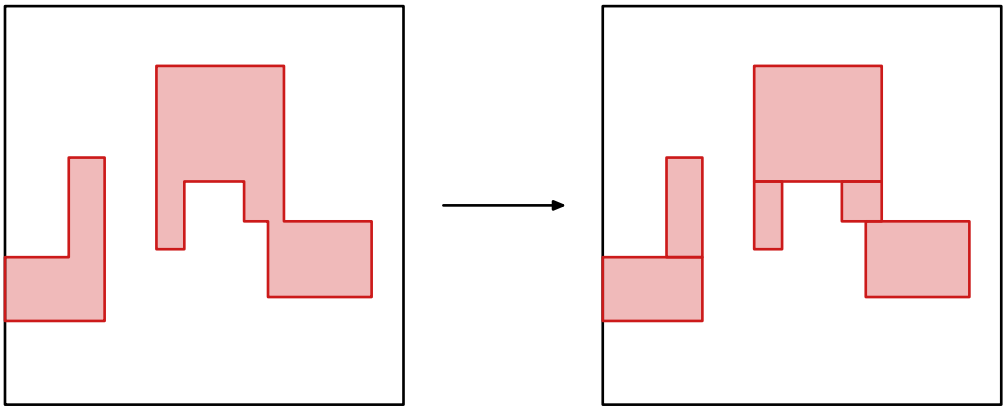

Figure 3.8: Decomposition of a mask clip into rectangles.

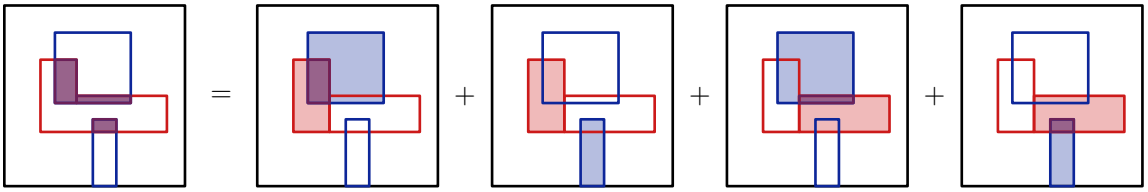

Figure 3.9: Visual explanation of Eq. (3.16).

For each pair (i, j), it can be easily decided from the coordinates of the rectangles whether A_i and B_j overlap. If they do, the coordinates of the rectangle $A_i \cap B_j$ can be found and the weighted area can be found using Eq. (3.15). If they do not, $|A_i \cap B_j| = 0$. Finally, the total weighted area where the two clips differ is given by the XOR of the two sets of rectangles:

$$|\Theta_1 \oplus \Theta_2| = |\Theta_1| + |\Theta_2| - 2\,|\Theta_1 \cap \Theta_2|\,. \tag{3.17}$$

This is demonstrated diagrammatically in Fig. 3.10.

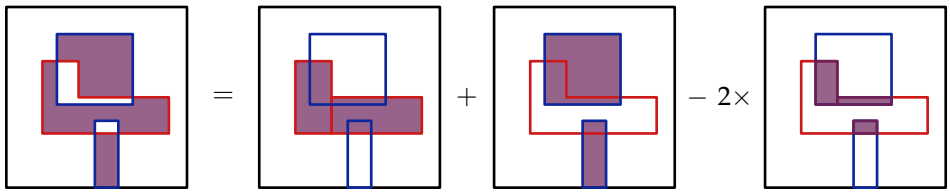

Figure 3.10: Visual explanation of Eq. (3.17).

Then the distance metric between Θ_1 and Θ_2 is

$$\rho_w(\Theta_1, \Theta_2) = \Big[\, |\Theta_1 \oplus \Theta_2| \, \Big]^{\frac{1}{2}}. \tag{3.18}$$

If we are considering rotations and reflections, then

$$\begin{aligned} \hat{\rho}_w(\Theta_1, \Theta_2) &= \min_{\tau \in D_4} \rho_w\big(\Theta_1, \tau(\Theta_2)\big) \\ &= \min_{\tau \in D_4} \Big[|\Theta_1| + |\tau(\Theta_2)| - 2\,|\Theta_1 \cap \tau(\Theta_2)| \, \Big]^{\frac{1}{2}} \\ &= \Big[|\Theta_1| + |\Theta_2| - 2 \max_{\tau \in D_4} |\Theta_1 \cap \tau(\Theta_2)| \, \Big]^{\frac{1}{2}}, \end{aligned} \tag{3.19}$$

so the weighted area of the overlap needs to be calculated for all eight transformations of clip Θ_2, and the maximum is used in the final distance computation.

Chapter 4

Clustering

We would like to take a large dataset of example hotspot clips, and build from this a catalogue of hotspot classes. Each class represents a particular type of hotspot, and each of the clips from our original dataset is represented by one of the classes. This catalogue can then be used to detect hotspots in new design layouts.

To build the catalogue, we require a clustering algorithm which can process the dataset and group the clips into a much smaller number of clusters in such a way that all of the clips in each cluster are the same type of hotspot, in terms of process sensitivity and geometric similarity. Mathematically, each cluster should have the property that the distance between any two clips belonging to it should be small, according to the distance metric designed in Chapter 3. Additionally, in general, clips should be closer to other clips within the same cluster than to clips in other clusters.

The clustering algorithm used for our work is based on the BUBBLE algorithm, which is particularly suited to large datasets in arbitrary metric spaces. After analysis of the clustering results, we modify the algorithm to tune its performance to the particular characteristics of hotspot clip datasets.

4.1 Hotspot clip datasets

The hotspot class catalogue created from the dataset will be used for hotspot detection in new design layouts. Any hotspot types which are not included in the catalogue will be

missed in the detection process. Therefore, it is important that our original dataset is as large and diverse as can be practically obtained.

The datasets are extracted from device layers containing hotspots. Multiple device layers may be used to form a single dataset, as long as they share an identical lithography process. It may be suitable to use early revisions of layouts instead of final manufacture layouts, since these are likely to contain a greater number of hotspots.

A full process window simulation is carried out on the layers, from which we get the set of image contours across the process window. Then geometric checks are used to compare these contours to the desired target patterns. These checks detect hotspots of various categories, such as bridging, pinching, and line end shortening hotspots. Each detected hotspot can be extracted by cutting out a square clip centred around the defect, with the size of the clip determined by the analysis in Section 3.5. These hotspot clips form our datasets. Each dataset corresponds to a specific category of hotspots (such as bridging hotspots), and is analysed separately from the other datasets.

For the results shown in this work, the datasets were extracted from a metal layer of a 755 µm × 622 µm layout at the 45 nm node. Our datasets were built from pinching hotspots, where at least one process condition contour violated a minimum width check. The check results were partitioned into a set of high-severity hotspots which failed to meet specification, and a set of low-severity hotspots which were marginally within specification. The size of the hotspot clips extracted from the layouts are 1.13 µm × 1.13 µm. Dataset I comprises 782 high-severity hotspots, and Dataset II comprises 13 200 low-severity hotspots.

4.2 Incremental clustering

Traditional clustering algorithms are global in nature. The basic approach of a global clustering algorithm is to calculate the distance between all pairs of objects in the dataset. The resulting distance matrix is then analysed using various techniques depending on the clustering method.

Various global clustering algorithms were tested on our hotspot clip datasets [13]. However, as the datasets grow in size, global algorithms become impractical. As the number of objects to be clustered becomes very large, it is computationally impractical to calculate the entire distance matrix. For a dataset of n objects, it is necessary to carry out $\binom{n}{2}$ distance metric computations. The runtime of the actual clustering algorithm quickly

becomes insignificant compared to the runtime of the distance matrix computation. This is especially true in our case where the computation of the distance metric is relatively complex (compared to, for example, a Euclidean distance between points). In addition to the problem of excessive runtime, it is also necessary to store the entire distance matrix in memory. For a dataset of 10 000 objects, the $10\,000 \times 10\,000$ distance matrix, stored as 64-bit numbers, requires around 750 MB. For 100 000 objects, the distance matrix requires around 75 GB.

For these reasons, global algorithms are infeasible for our purpose. Therefore, incremental clustering algorithms were investigated. Instead of looking at the entire dataset simultaneously, incremental algorithms start by analysing a small subset of the data, then incrementally accept further small subsets until the entire dataset has been processed. Distance metrics are only calculated as needed, instead of between all possible pairs of objects in the dataset. For this reason, smart incremental algorithms can have a drastically reduced runtime compared to global algorithms. Additionally, incremental algorithms only require a small amount of data in memory at any given point, so the memory requirements are also much smaller.

The BIRCH algorithm [20] is a very efficient incremental clustering algorithm for very large datasets. The algorithm makes use of a clustering tree which grows incrementally as objects from the dataset are fed to it. Only a single scan of the dataset is required, and the memory requirements of the algorithm are very small. However, the BIRCH algorithm assumes that the objects are represented by points in Euclidean space, which is not the case for our datasets.

The BUBBLE algorithm [6] is a generalisation of the BIRCH algorithm to arbitrary metric spaces. This algorithm is specifically targeted for metric spaces in which computation of the distance metric may be very expensive. Therefore, the BUBBLE algorithm was chosen as the most suitable algorithm to cluster our hotspot clip datasets.

4.3 BUBBLE algorithm

4.3.1 Description

The centrepiece of the BUBBLE incremental clustering algorithm is a clustering tree which evolves as objects from the dataset are sequentially fed into it. There are two parameters controlling the algorithm: the cluster radius threshold, t, which determines

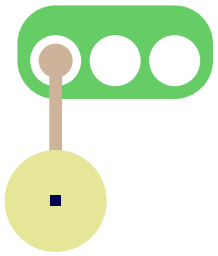

Figure 4.1: The initial configuration of the clustering tree.

the maximum radius of clusters produced by the algorithm, and the branching factor, B, which determines the maximum amount of children which may branch out from any node of the tree.

The tree is initialised as a single node, whose only child is a cluster containing the first object in the dataset, as shown in Fig. 4.1. In the figure, nodes are represented by green shapes, and each node is linked to its children via brown lines. Clusters are represented by yellow circles, and the hotspot clips are the small blue squares contained in the clusters.

As the clustering algorithm progresses, the tree grows to contain many nodes and clusters, shown in Fig. 4.2. At any point during the algorithm, each node of the tree has at least 1 and at most B children. (For the examples in the figures, the branching factor is $B = 3$.) The children of leaf nodes of the tree are clusters, while the children of nonleaf nodes are other nodes of the tree.

In order to reduce memory requirements, we do not keep all objects in memory as they are added to the tree. Instead, we store the structure of the clustering tree, and for each node and cluster in the tree we store a small summarised representation called the generalised cluster feature (CF*). The tree structure and the CF*s are updated as objects are added.

Each cluster has a special member which is the centre of the cluster. The centre object is the object which has the smallest mean square distance to the other objects in the cluster. In the figures, the centre object is represented by a darker colour compared to the other objects in the cluster. The CF* for a cluster contains the centre of the cluster and various other statistics of the cluster which are required by the algorithm.

While the CF* for a cluster has the centre object as its single representative object, the CF* for a node of the tree contains several representative objects which belong to the subtree formed by the node's descendants. These objects are selected and updated to summarise

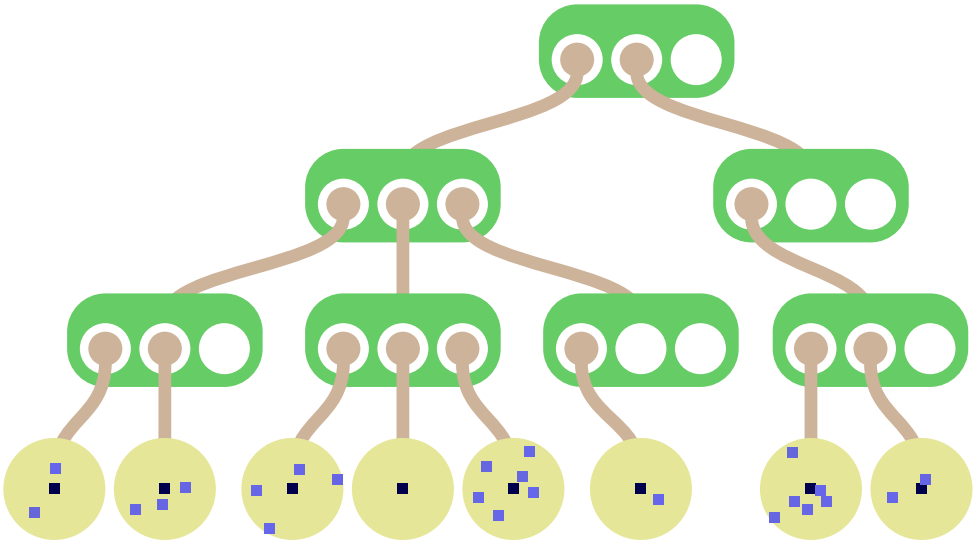

Figure 4.2: A configuration of the clustering tree after more objects are added.

the current contents of the node's subtree. The CF* for a node also contains various other statistics about the subtree which the algorithm utilises.

When a new object from the dataset is inserted into the clustering tree, it begins at the root node. It must first be decided which of the child nodes the object will descend to. This is achieved by taking the representative objects of each of the child nodes and calculating the distance metrics between the new object and each of these. The object then descends to the child node such that the mean square distance to the child node's representative objects is the smallest.

An example of a branching decision at a nonleaf node is shown in Fig. 4.3. The new object is shown in red. The upper diagram shows the three child nodes which the object considers. The lower diagram shows the object descending to one of the three child nodes.

This branching decision is repeated until the new object reaches a leaf node, at which point the distance metric is calculated between the object and the centre object of each of the child clusters. If the smallest distance is within the cluster radius threshold, t, then the object is added into the corresponding cluster. However, if all of the distances are greater than t, then a new cluster is formed as a child of the leaf node, with the new object as its only member.

An example of a branching decision at a leaf node is shown in Fig. 4.4. The upper diagram

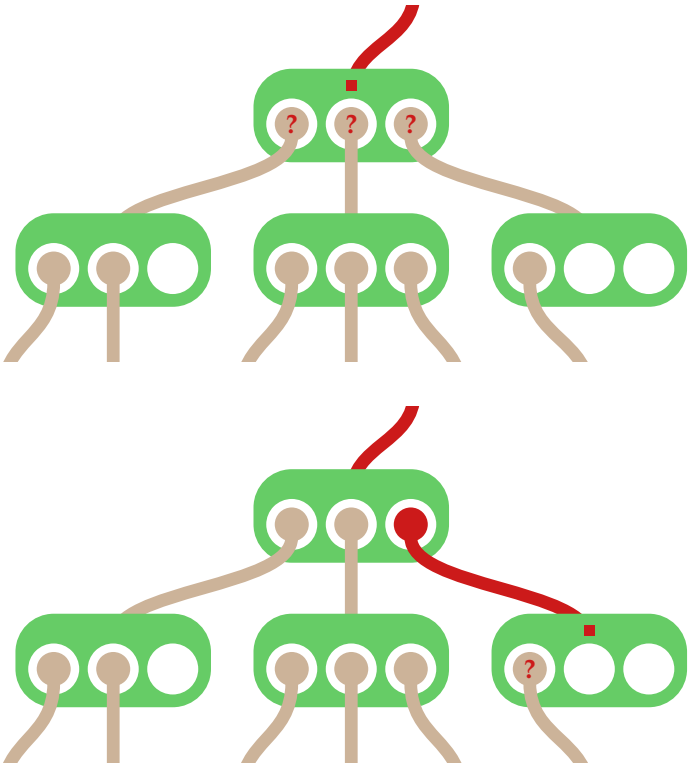

Figure 4.3: Branching decision at a nonleaf node of the clustering tree.

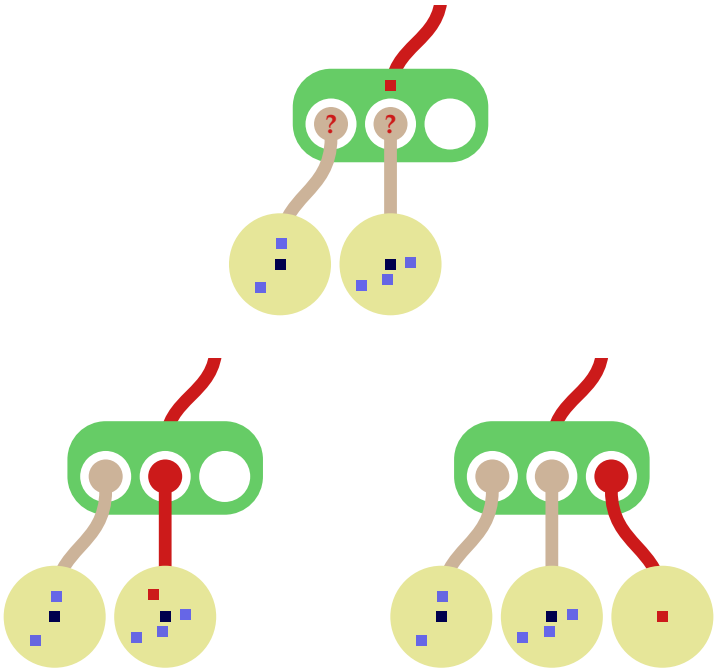

Figure 4.4: Branching decision at a leaf node of the clustering tree.

shows the two child clusters which the object considers. The lower left diagram shows the object descending to one of the two child clusters. The lower right diagram shows the object alternatively forming its own child cluster, because it is not close enough to the centres of the other two child clusters.

Occasionally, when a new object is added to a cluster, the current centre object may no longer have the smallest mean square distance to the other objects in the cluster. In this case, the centre object will be updated, as shown in Fig. 4.5.

Sometimes the formation of a new child cluster causes the leaf node to have more than B children. In order to rectify this, the node is split into two new nodes (with the same parent). The children of the original node are distributed between the two new nodes in such a way as to maximise the distance between the two new groups. Note that the splitting of the node may in turn cause its parent node to have more than B children, in which case this parent node must too be split. This may continue until in some cases the root node of the tree itself may be split, and a new root node is formed as the parent of the two new nodes, and thus the height of the clustering tree is increased. An example of the clustering tree height increasing is shown in Fig. 4.6.

This algorithm does not provide any guarantees on the separateness of the resulting

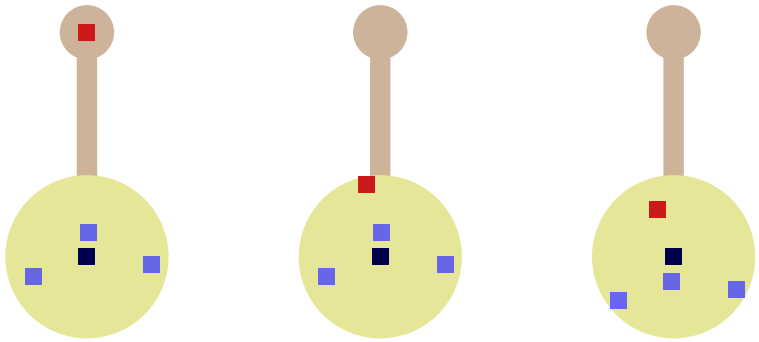

Figure 4.5: Updating the centre object of a cluster.

clusters. It can happen that two clusters in different parts of the tree are in fact very close and are better represented as a single cluster. To merge close clusters of this kind, a final global clustering algorithm may be applied to the representative objects of each cluster.

Further details of the implementation of the BUBBLE algorithm to cluster hotspot clip datasets are given in previous works [7, 12].

4.3.2 Results

The BUBBLE algorithm was used to cluster the 782 hotspot clips from Dataset I. The clips were grouped into 58 clusters. An example of a cluster produced by the algorithm is shown in Fig. 4.7. The centre clip of the cluster is shown in the top left. The sizes of the clusters produced are shown in Fig. 4.8. It can be seen that there is great variation in the sizes of the clusters, with the largest cluster containing almost one-fifth of the clips, and many clusters containing only a single clip.

Figure 4.9 shows the time elapsed against the number of clips clustered, during a single run of the BUBBLE algorithm for the 13 200 hotspot clips from Dataset II. Note that we can infer from this graph the total time required to cluster any number of hotspot clips up to 13 200 using the BUBBLE algorithm, since the algorithm performs a complete clustering of the current set of objects before taking the next object. Thus the plot shows us the total runtime of the BUBBLE algorithm against the number of hotspot clips being clustered.

Of course, the exact runtime depends upon the properties of the dataset being clustered, as well as the values chosen for the cluster radius threshold, t, and the branching factor, B. For example, a dataset with objects which fit into just two clusters would be clustered

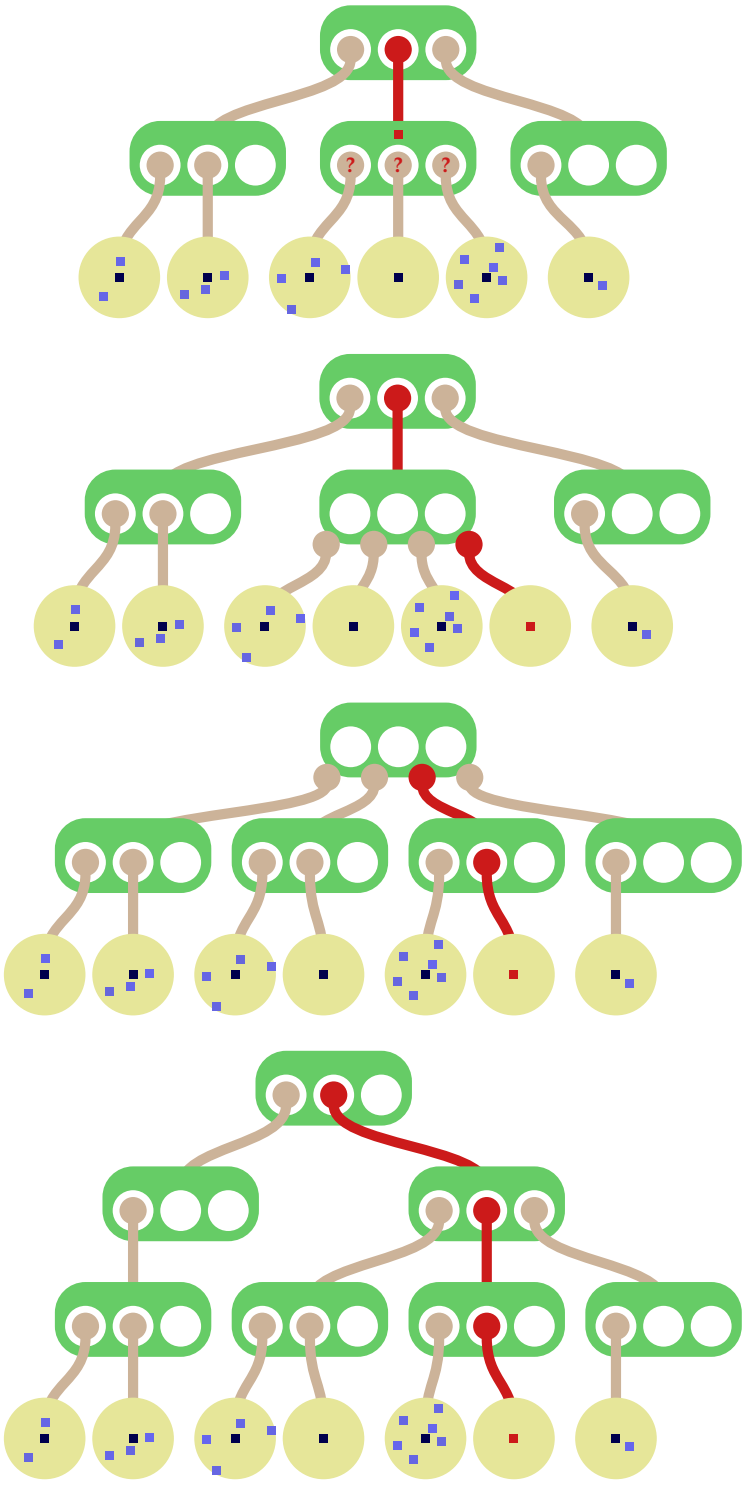

Figure 4.6: Increasing the height of the clustering tree, due to node splitting.

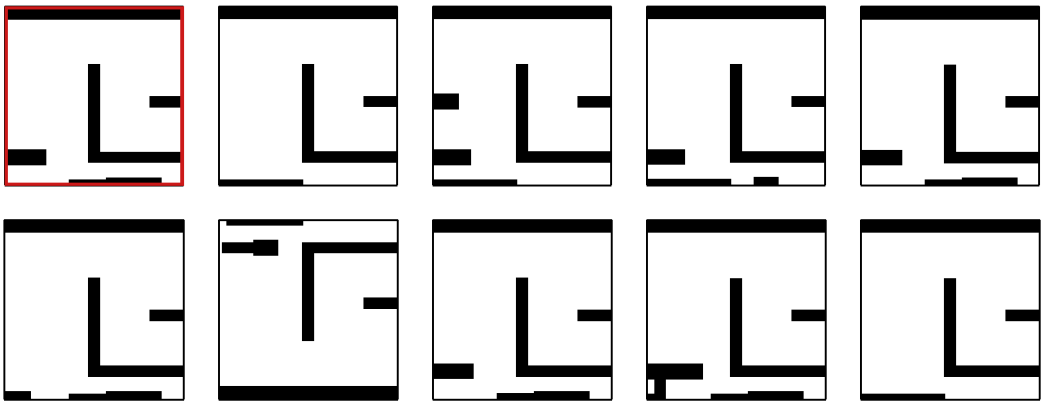

Figure 4.7: An example of a cluster produced by the BUBBLE algorithm.

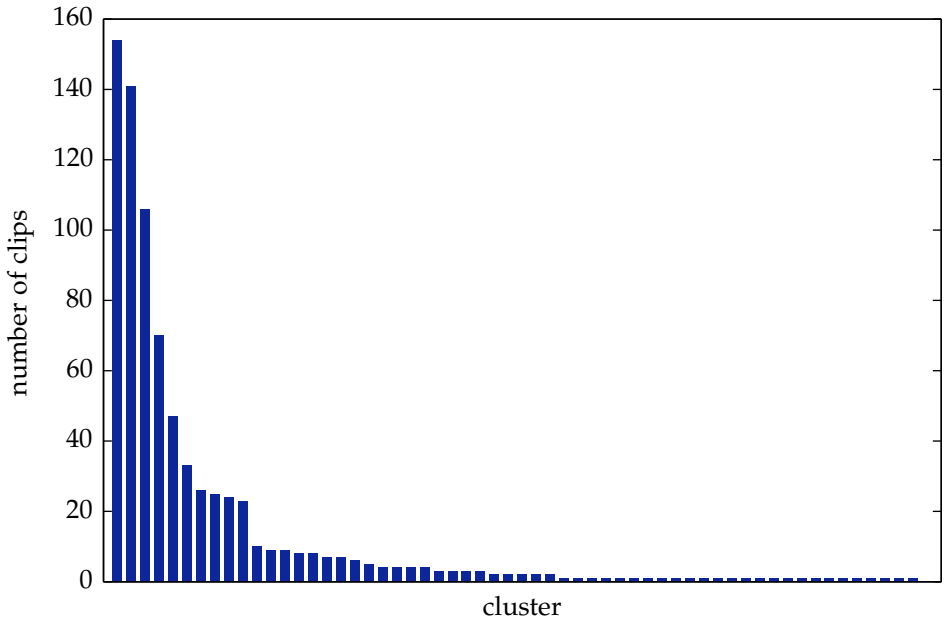

Figure 4.8: Sizes of the clusters produced by the BUBBLE algorithm.

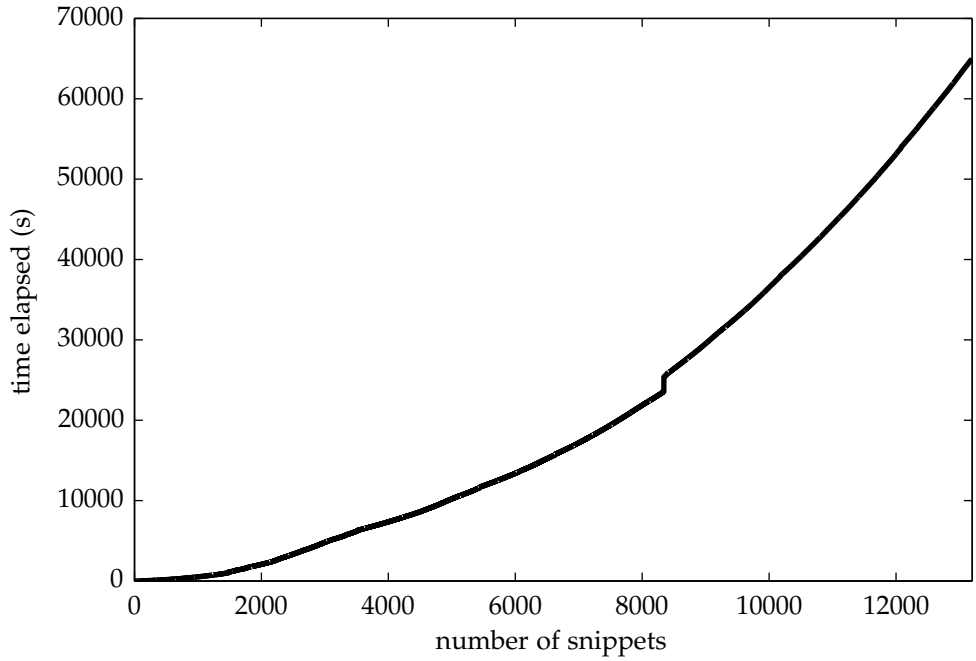

Figure 4.9: Total runtime of the BUBBLE algorithm.

much faster with this algorithm, since at most two distance metric calculations would be required to place each object in the tree.

4.4 Plucking modification

4.4.1 Description

Upon examining the clustering produced by the BUBBLE algorithm for various sets of layout clips, it was found in all cases that there were a small number of clusters containing a large proportion of the clips. For example, Fig. 4.8 shows that 649 of the 782 clips are contained in just 10 out of the 58 clusters.

This is due to the amount of repetition present in the source layouts from which the datasets were extracted. The presence of these large clusters motivates a modification of the clustering algorithm which optimises for such datasets.

If, at some point during the clustering algorithm, a particular cluster is growing much

faster than the average cluster, then it is reasonable to expect that this cluster will continue to collect a significant proportion of the incoming objects, since the datasets are not ordered in any special way. Therefore, the clustering algorithm is modified as follows.

A new parameter, L, is chosen as a threshold size of a cluster. If an object is added to a cluster so that it now contains more than L objects, then the BUBBLE algorithm is interrupted. The remaining objects in the dataset yet to be inserted into the clustering tree are taken one at a time, and checked to see whether they belong to this large cluster by calculating the distance metric from each to the centre object. Those objects which are within distance t of the centre object are added to the cluster, and this cluster is then removed from the clustering tree, since it has found all its members and will not grow further. If there are any nodes left childless, then these nodes are also removed from the clustering tree. Then the BUBBLE algorithm is resumed, and the remaining objects are added sequentially as before.

We call this the plucking modification, since the cluster is plucked from the tree once it is full. Figure 4.10 shows an example of plucking. In the upper diagram, the highlighted cluster has reached the threshold size. The remaining objects are scanned and some of these are added to the cluster. The cluster is then removed from the clustering tree, as shown in the middle diagram. Since its parent is now childless, this node is also removed from the clustering tree, as shown in the lower diagram.

The benefit of the plucking modification can be seen in the following ad hoc analysis. At the time of the interruption, assume that there are M unclustered objects remaining in the dataset, m of which belong to the large cluster. Additionally, assume that the remaining objects require an average of k distance metric computations to traverse from the root of the tree into a cluster. (Here, k is on the order of the branching factor, B, multiplied by the height of the tree.)

Then, in the original BUBBLE algorithm, an average of k distance metric computations are required to insert each of the M remaining objects. With the modification, the large cluster is filled in by performing one distance computation for each of the M objects, then the remaining $M - m$ objects are inserted into the tree as usual with an average of k distance metric computations. Therefore, our modification saves us $Mk - (M + (M - m)k) = mk - M$ distance computations. Thus, we achieve a reduction in runtime if $mk - M > 0$. Reformulating, we can say that we achieve a reduction in runtime if the proportion of remaining objects which are expected to belong to the the large cluster, m/M, is greater than $1/k$.

The precise overall reduction in runtime achieved by the plucking modification to the

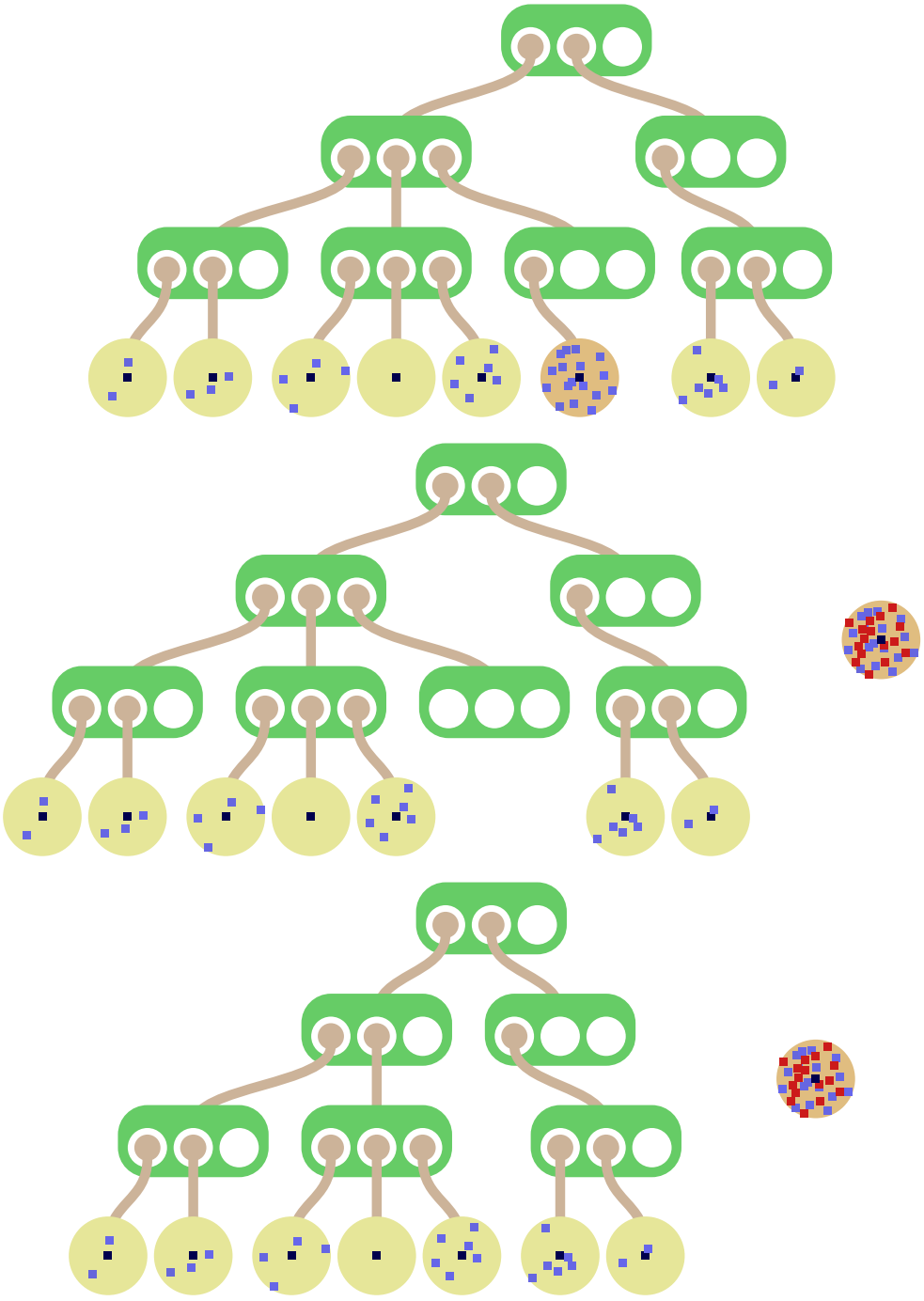

Figure 4.10: Plucking a large cluster from the clustering tree.

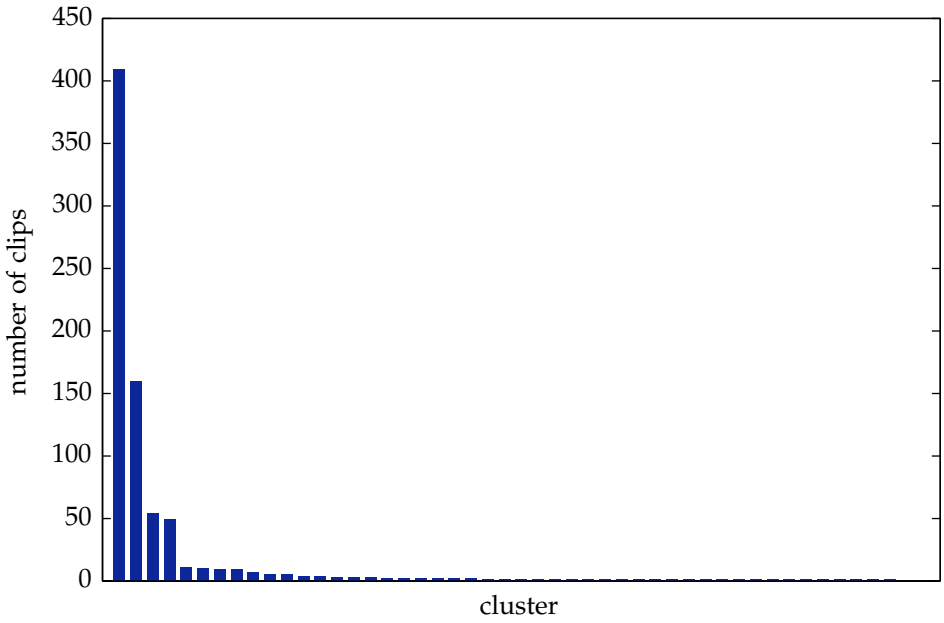

Figure 4.11: Sizes of the clusters produced by the BUBBLE algorithm using plucking.

BUBBLE algorithm is highly dependent on the dataset being clustered. For example, if we have more than k clusters of roughly equal size, then our modification would not improve the runtime at all. (It would not increase the runtime either; no plucking would occur due to the lack of large clusters.) Therefore, we demonstrate the benefit of the plucking modification via experimental results in the following section.

4.4.2 Results

The BUBBLE algorithm with plucking was used to cluster the 782 hotspot clips from Dataset I. This time, the clips were grouped into 47 clusters. Figure 4.11 shows the sizes of the clusters produced. We can see that there are now even larger clusters present. Comparing the clustering to that produced without plucking, we find that some of the larger clusters have merged.

Figure 4.12 shows the time elapsed against the number of clips clustered, during a single run of the BUBBLE algorithm using plucking for 13 200 hotspot clips. We can see that the algorithm clusters large numbers of clips very quickly at the beginning due to plucking,

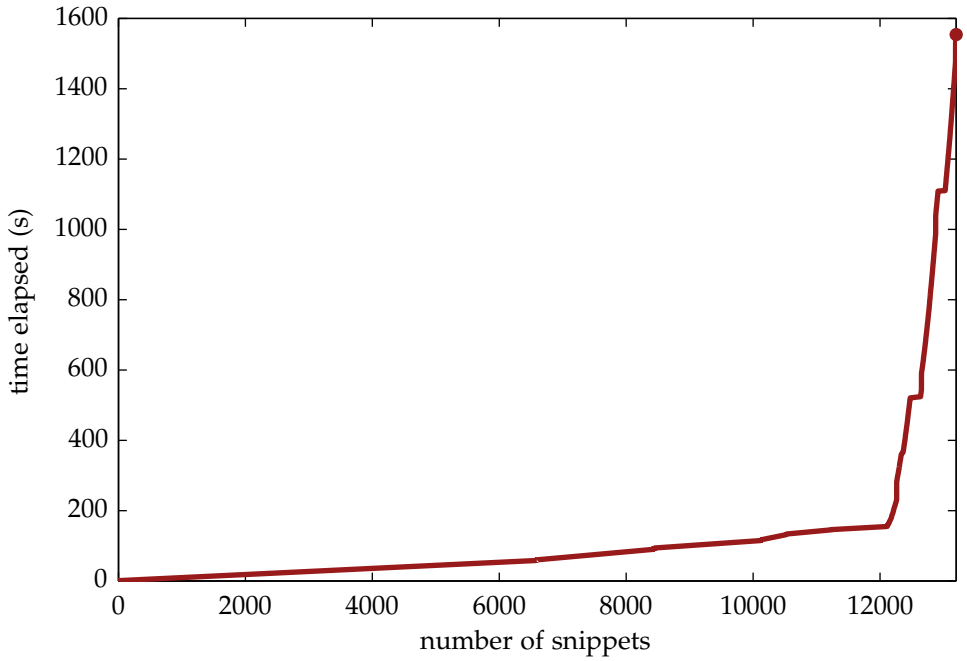

Figure 4.12: Time elapsed during a run of the BUBBLE algorithm using plucking.

and slows down significantly towards the end as it clusters all those clips which were not added to one of the plucked clusters.

In contrast to the BUBBLE algorithm without plucking, we cannot infer from this graph the total time required to cluster a smaller number of clips, because in this case the algorithm will cluster the objects in a different order due to the plucking breaks, thus affecting the runtime. Figure 4.13 shows the total time taken for the BUBBLE algorithm using plucking for various numbers of hotspot clips. The runtime is roughly linear with the number of clips. (The lighter lines show the cumulative time elapsed against the number of clips clustered for each individual run of the BUBBLE algorithm using plucking.)

It can be seen that the plucking modification makes the BUBBLE algorithm significantly faster at performing clustering. Figure 4.14 shows a comparison of the runtimes for the BUBBLE algorithm with plucking (shown by the coloured line) and without plucking (shown by the black line). The reduction in runtime is greater for larger datasets. For the largest dataset of 13 200 clips, the modified algorithm is over forty times faster.

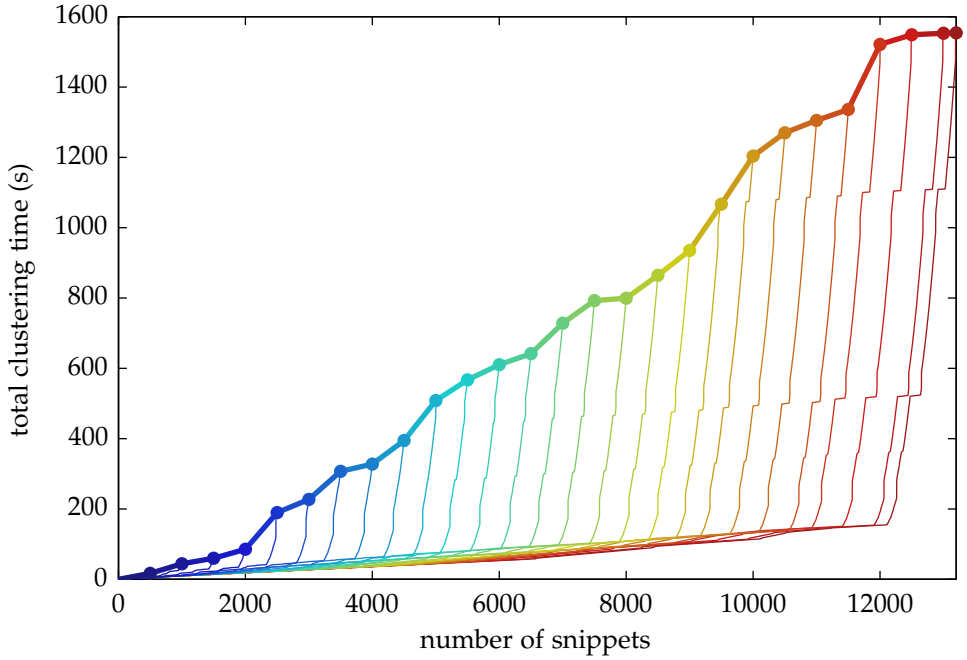

Figure 4.13: Total runtime of the BUBBLE algorithm using plucking.

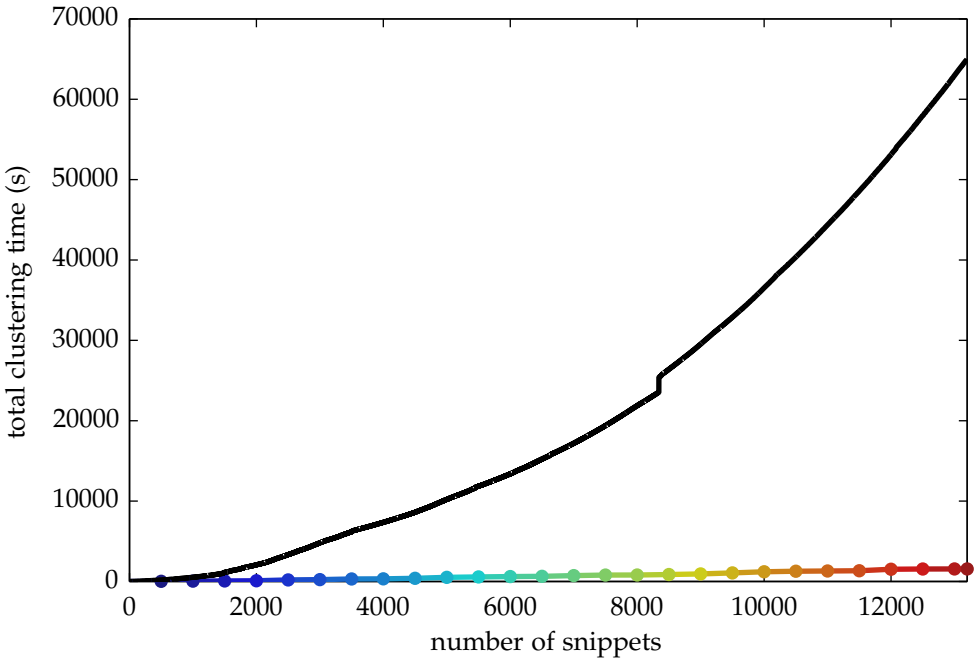

Figure 4.14: Runtime comparison of the BUBBLE algorithm with and without plucking.

Chapter 5

Hotspot detection

Now we are ready to build our catalogue of hotspot classes which will allow us to detect hotspots in new layouts. We will be using pattern matching tools to scan layouts in order to detect patterns which match the hotspot classes in our catalogue. In order to achieve this, we must analyse our clusters of hotspots to produce a template and matching threshold which the pattern matching tool can use.

5.1 Hotspot classes

Each of the clusters produced by our clustering algorithm is a subset of our original dataset of hotspot clips. The clips belonging to a particular cluster have been grouped together due to similar process sensitivity and pattern geometry. However, clearly there are other possible hotspot patterns which were not among the clips in the original dataset but which share a similar process sensitivity and pattern geometry to those in the cluster. Thus, instead of considering the set of concrete clips in the cluster, we define an abstract set of possible clips which are all of the same hotspot "type" as those in the cluster. We call this a hotspot *class*.

In contrast to the hotspot cluster, which consists of a discrete set of objects, a hotspot class represents continuous set of clips. Corresponding to each of the hotspot clusters, we define a hotspot class, $\mathcal{H}_i$, in terms of our distance metric, $\hat{\rho}$, by specifying by a centre clip, Θ_i, and radius, r_i,

$$\mathcal{H}_i = \{\Theta : \hat{\rho}(\Theta_i, \Theta) < r_i\}\,. \tag{5.1}$$

The centre clip is simply determined by the centre of the corresponding cluster. The radius is determined by the maximum distance from the centre clip to the remaining objects in the cluster. If the cluster consists of just a single object, the radius is set to zero. Then each of the radii can be increased by some small nominal amount, because in most cases the members of the clusters should not define the exact boundaries of the hotspot class but rather lie inside them.

Of course, it will often be the case that not every possible clip belonging to the class $\mathcal{H}_i$ will be a hotspot. It is preferable to choose the radius conservatively, because false positive results are easily checked by performing a simulation on a small clip, while false negatives can cause expensive delays further on in the design flow.

An optional method of increasing the detection rate of the pattern matching is to define the hotspot classes in terms of a pattern-specific weighting function, instead of the pattern-independent root-mean-square weighting function as defined in Eq. (3.7). In hotspot class $\mathcal{H}_i$, we are specifically looking for patterns which have a similar process sensitivity to Θ_i, so it makes sense to use the pattern-specific weighting function, $w_{\Theta_1}(x, y)$, as defined in Eq. (3.5). Since there are only a small number of hotspot classes, the computation required to calculate these weighting functions is not very significant. The distances to the other objects in the cluster should be recalculated in order to determine the radius, r_i, in terms of the new pattern-specific distance metric.

5.2 Pattern matching

Conveniently, current pattern matching tools are able to search layouts precisely for patterns belonging to sets of the form given by Eq. (5.1). The pattern matching tool used in our work is Cadence's ÉclairPM software. Given a template pattern, the tool seeks to find the same or similar patterns in a layer. It achieves the comparison between the template pattern to the layer at each test position by checking a value called the match factor.

The template pattern, $\bar{T}(x, y)$, is a real- or complex-valued function defined within some domain $\mathcal{D}$, so $\bar{T}(x, y) = 0$ for $(x, y) \notin \mathcal{D}$. The domain $\mathcal{D}$ is commonly a rectangle, but it can be any set we like. The origin $(0, 0)$ is called the anchor point of the template, and each position (X, Y) in the layer is checked by overlaying the template on the layer with the anchor point above (X, Y). The match factor at this point, $MF(X, Y)$, is then defined as the inner product between the template and the layer. Importantly, the pattern

matching tool considers the layer numerically equal to 1 where light is transmitted and -1 otherwise, in contrast to the usual convention in our work which uses values of 1 and 0. Thus the matching area of the layer is given by

$$\bar{O}_{(X,Y)}(x,y) = \begin{cases} 1 & \text{if } (x,y) \in \mathcal{D} \text{ and } O(X+x, Y+y) = 1, \\ -1 & \text{if } (x,y) \in \mathcal{D} \text{ and } O(X+x, Y+y) = 0, \\ 0 & \text{if } (x,y) \notin \mathcal{D}. \end{cases} \tag{5.2}$$

Then, the match factor is given by

$$MF(X,Y) = \iint \bar{T}(x,y)\bar{O}_{(X,Y)}(x,y)\,\mathrm{d}A \tag{5.3}$$

Given a template pattern and a match factor threshold MF_{th}, the pattern match tool will return all test positions (X,Y) in the layer for which $MF(X,Y) > MF_{\text{th}}$.

We would like to use the match factor threshold as a means to enforce the distance metric threshold describing our hotspot class, given by Eq. (5.1). This can be achieved by defining our template as

$$\bar{T}_i(x,y) = w(x,y)\bar{\Theta}_i(x,y), \tag{5.4}$$

where $\bar{\Theta}_i(x,y)$ is the same as the centre clip, $\Theta_i(x,y)$, adjusted to follow the convention of the pattern matching tool,

$$\bar{\Theta}_i(x,y) = \begin{cases} 1 & \text{if } (x,y) \in \mathcal{D} \text{ and } \Theta(x,y) = 1, \\ -1 & \text{if } (x,y) \in \mathcal{D} \text{ and } \Theta(x,y) = 0, \\ 0 & \text{if } (x,y) \notin \mathcal{D}, \end{cases} \tag{5.5}$$

where the domain is chosen to match the square domain of our clips, $\mathcal{D} = S_R$.

Using this template, the match factor is now

$$\begin{aligned} MF(X,Y) &= \iint \bar{T}_i(x,y)\bar{O}_{(X,Y)}(x,y)\,\mathrm{d}A \\ &= \iint w(x,y)\bar{\Theta}_i(x,y)\bar{O}_{(X,Y)}(x,y)\,\mathrm{d}A \end{aligned} \tag{5.6}$$

Now, note that since $\bar{\Theta}_i(x,y) = \pm 1$ and $\bar{O}_{(X,Y)}(x,y) = \pm 1$, we have

$$\bar{\Theta}_i(x,y)\bar{O}_{(X,Y)}(x,y) = \begin{cases} 1 & \text{if } \bar{\Theta}_i(x,y) = \bar{O}_{(X,Y)}(x,y), \\ -1 & \text{if } \bar{\Theta}_i(x,y) \neq \bar{O}_{(X,Y)}(x,y). \end{cases} \tag{5.7}$$

Then,

$$\begin{aligned} MF(X,Y) &= \iint\limits_{\bar{\Theta}_i=\bar{O}_{(X,Y)}} w(x,y)\,\mathrm{d}A - \iint\limits_{\bar{\Theta}_i\neq\bar{O}_{(X,Y)}} w(x,y)\,\mathrm{d}A \\ &= \iint\limits_{\mathcal{D}} w(x,y)\,\mathrm{d}A - 2\iint\limits_{\bar{\Theta}_i\neq\bar{O}_{(X,Y)}} w(x,y)\,\mathrm{d}A \\ &= \iint\limits_{\mathcal{D}} w(x,y)\,\mathrm{d}A - 2\rho(\bar{\Theta}_i,\bar{O}_{(X,Y)}). \end{aligned} \tag{5.8}$$

Therefore, we can set

$$MF_{\text{th}} = \iint\limits_{\mathcal{D}} w(x,y)\,\mathrm{d}A - 2r_i, \tag{5.9}$$

so that

$$\begin{aligned} \{(X,Y) : MF(X,Y) > MF_{\text{th}}\} &= \left\{(X,Y) : \hat{\rho}(\Theta_i,\bar{O}_{(X,Y)}) < r_i\right\} \\ &= \left\{(X,Y) : \bar{O}_{(X,Y)} \in \mathcal{H}_i\right\}. \end{aligned} \tag{5.10}$$

Thus, with this particular choice of template and match factor, the pattern matching tool will return exactly those positions in the layer where the pattern belongs to the hotspot class, $\mathcal{H}_i$.

An example of a template pattern, $\bar{\Theta}_i(x,y)$, created by multiplying a centre clip and the weighting function is shown in Fig. 5.1. In the figure, the yellow regions correspond to near-zero values, red regions correspond to positive values and blue regions correspond to negative values.

5.3 Results

In order to test the capability of our system in detecting hotspots in new layouts, 500 of the 782 high-severity hotspot clips from Dataset I were chosen randomly as a training set. Clustering was performed on these 500 clips, and a catalogue of hotspot classes was formed. A pattern matching tool then used this catalogue to scan the layout from which the Dataset I hotspots were taken. The aim of the experiment was to determine how many of the remaining 282 high-severity hotspots would be found by the pattern matching tool.

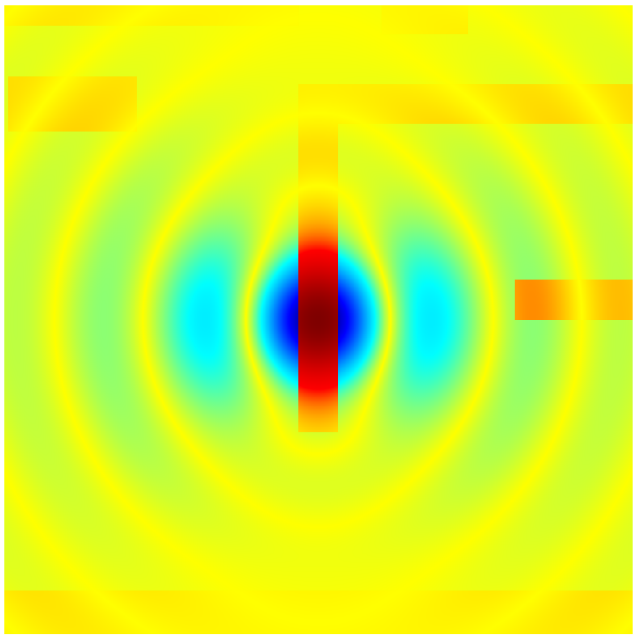

Figure 5.1: An example of a weighted clip used by the pattern matching tool.

The incremental clustering algorithm (with the plucking modification) grouped the training set into 41 clusters. A global clustering algorithm (specifically the single-linkage hierarchical algorithm) was then used to merge close clusters, resulting in 30 clusters. These clusters were then analysed to obtain a catalogue of 30 hotspot classes.

The pattern matching tool was run 30 times. Each run yielded a list of results consisting of coordinates in the layout for a match location, and a pattern match factor. The coordinates of each result in the lists were analysed to determine whether the pattern at the location was a high-severity hotspot, a low-severity hotspot, or a cold spot. The pattern match factors were converted to distance metric values, using Eq. (5.8).

A summary of the results for a single run of the pattern matcher is shown in Fig. 5.2. Here the template used was the centre of a medium-sized cluster containing 34 high-severity hotspots. The graph is a cumulative breakdown of the results into high-severity hotspots, low-severity hotspots and cold spots, as the distance metric threshold is increased. The left end of the graph shows the pattern match results with the smallest distances; that is, those patterns which were most similar to the template. It can be seen that all of these early results are high-severity hotspots. It is only when we increase the distance metric threshold that some low-severity hotspots and cold spots appear in the results.

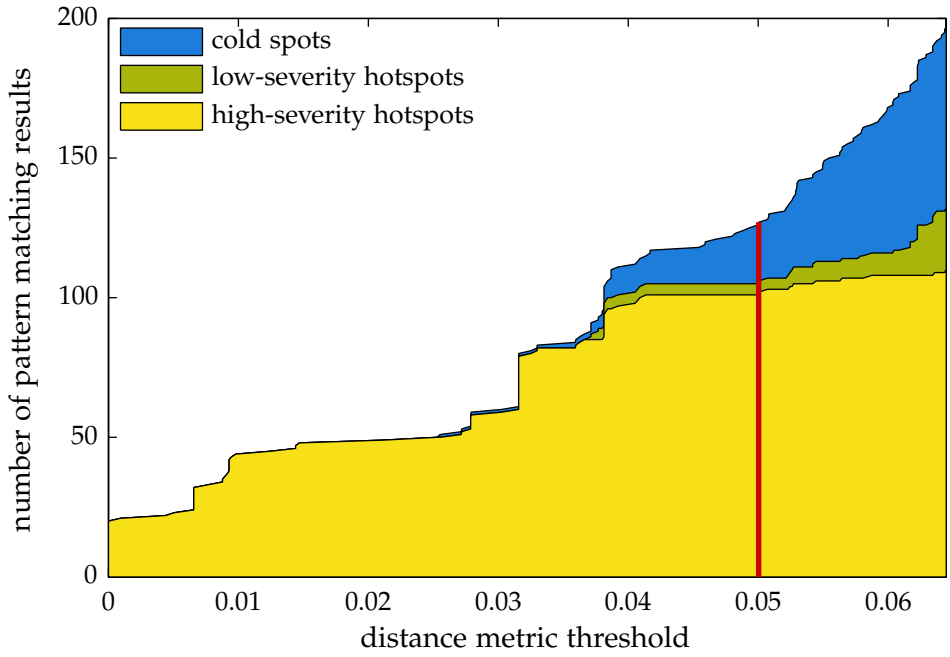

Figure 5.2: Pattern matching results for a medium-sized cluster.

The red vertical line represents the distance metric value corresponding to the radius of the class, so this was the cut-off point used for the results. At this point, there were 127 pattern matching results. 102 of these were high-severity hotspots, and 4 of these were low-severity hotspots, leaving only 21 cold spots in the results. All of the 34 high-severity hotspots from the original cluster are included in the results. Thus it can be seen that the pattern matching in this case is very successful in detecting hotspots, and there is a low false positive rate.

Similar favourable results were found for most of the 30 pattern matching runs. However, some of larger clusters yielded a higher false positive rate. The results from the centre of one of the largest clusters containing 99 high-severity hotspots is shown in Fig. 5.3. It can be seen that a large proportion of the early results with the smallest distance metric values are high-severity hotspots. However, as the distance metric threshold decreases, this proportion drops. At the cut-off point corresponding to the cluster radius, there were 2411 pattern matching results. 333 of these were high-severity hotspots, and 220 of these were low-severity hotspots, leaving 1858 cold spots. Of course, the cost of testing these 2411 locations for process sensitivity is much less than the cost of performing a full process sensitivity simulation across the entire layout. It may be possible to reduce

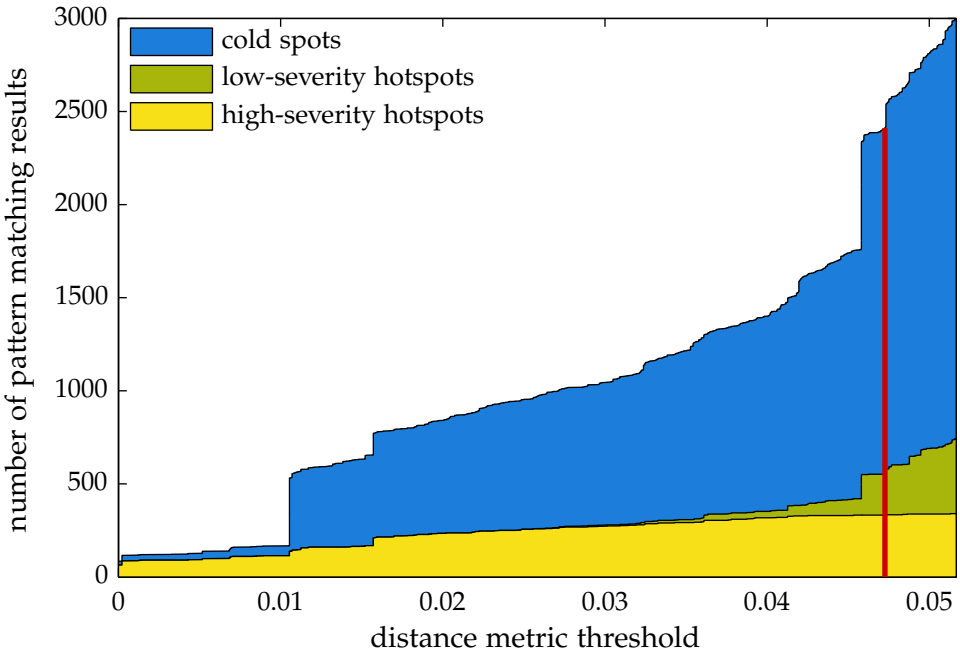

Figure 5.3: Pattern matching results for a large cluster.

the overall false positive rate by modifying the global clustering algorithm to produce smaller or rounder clusters.

Finally, the 30 sets of results were amalgamated to produce a list of potential hotspots identified by the system. Of the 282 high-severity hotspots in the layout which were not included in the training set, 269 of these were found within the total pattern matching results, leaving only 13 undetected.

Those high-severity hotspots not found within the pattern matching results were unable to be detected because their class of hotspots was not represented in the training set. For example, hotspot clips #29, #37 and #39 were not found in the pattern matching results. From the clustering result of the full Dataset I, it can be seen that these three clips comprise a cluster. However, none of these hotspots were included in the training set of 500 clips. Since this class of hotspots is not represented in the training set, they are not included in the hotspot classification catalogue, and therefore hotspots of this type are unable to be identified by the resulting system.

Note that the failure to find a few of the hotspots was an expected outcome of this experiment, due to the limited nature of the training set. In a true implementation, the

catalogue would not be built just from hotspots taken from a single device layer, as was the case here. In order to increase the detection rate, the training set should be drawn from a number of layouts, including a combination of test layouts and early designs.

This is a fundamental limitation of a system which does not have a simulation-based approach to hotspot detection. Just as traditional design rule checks are unable to detect a hotspot unless a previously found similar hotspot has prompted the definition of a rule to prevent it, the accuracy of our system in detecting all of the hotspots in a new layout is dependent upon the size and diversity of the training dataset used to build the catalogue. As stated from the outset, the goal of our hotspot detection system is not to replace a full process window simulation for catching hotspots in tape out, but to act as a fast detection tool in early design stages, without the burden of needing to manually devise and implement design rules.

Chapter 6

Conclusions

In this dissertation, we described the generation and usage of a hotspot detection and classification system. In contrast to traditional design rule checking systems, where design rules must be manually created, our system can automatically build a catalogue of hotspot classes by analysing input device layers containing hotspots. This catalogue can then be used together with a pattern matching tool to detect hotspots in any new design layouts sharing the same lithography process.

In order to build the catalogue, hotspot clips are first extracted from the provided device layers. Then these clips are grouped into clusters containing similar hotspots. These clusters can then be analysed to form the catalogue.

Two original inventions were developed in order to form meaningful clusters from our dataset of hotspot clips. The first is a distance metric which is designed to describe the similarity between two hotspot patterns. The key component of this distance metric is the variance contribution map, which quantifies the effect of features across the mask on the process variance of a particular dimension.

The second original invention is a clustering algorithm used to efficiently process a very large dataset. This clustering algorithm is based on the BUBBLE algorithm, and analysis of clustering results leads to our plucking modification which results in enormous runtime improvements for our datasets.

Finally, we demonstrate the application of our catalogue to hotspot detection. The form taken by our distance metric conveniently allows us to use existing pattern matching tools to find instances of our hotspot classes in new layouts. Note that our system not only detects the hotspots in a layout, but also determines the hotspot class to which each

found hotspot belongs. This feature is useful to designers because it enables them to group similar results and work on multiple instances of the same type of hotspot together.

The system was tested by using a subset of hotspots in a device layer as a training set, and using the resulting hotspot catalogue to detect the remaining hotspots. A high detection rate was achieved, although with a significant number of false positive results. A hotspot catalogue built from a more extensive dataset, collected from several layouts, would be expected to produce better results.

Our hotspot detection system demonstrates the power of pattern matching to provide efficient design for manufacturability solutions. We believe that hotspot detection is just one example of its use, and envision similar systems being created in the future to accelerate various other design steps, such as optical proximity correction or double patterning colouring.

Bibliography

[1] Max Born and Emil Wolf. *Principles of optics: Electromagnetic theory of propagation, interference and diffraction of light*. Cambridge University Press, 7th edition, 1999.

[2] Timothy Brunner and Richard Ferguson. Approximate models for resist processing effects. In *Optical Microlithography IX*, volume 2726 of *Proceedings of SPIE*, 1996.

[3] Nicolas B. Cobb and Avideh Zakhor. Fast, low-complexity mask design. In *Optical/Laser Microlithography VIII*, volume 2440 of *Proceedings of SPIE*, 1995.

[4] Vito Dai, Luigi Capodieci, Jie Yang, and Norma Rodriguez. Developing DRC plus rules through 2D pattern extraction and clustering techniques. In *Design for Manufacturability through Design-Process Integration III*, volume 7275 of *Proceedings of SPIE*, 2009.

[5] Vito Dai, Jie Yang, Norma Rodriguez, and Luigi Capodieci. DRC Plus: augmenting standard DRC with pattern matching on 2D geometries. In *Design for Manufacturability through Design-Process Integration*, volume 6521 of *Proceedings of SPIE*, 2007.

[6] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke, Allison Powell, and James French. Clustering large datasets in arbitrary metric spaces. *Proceedings of the 15th International Conference on Data Engineering*, pages 502–511, 1999.

[7] Justin Ghan, Ning Ma, Sandipan Mishra, Costas Spanos, Kameshwar Poolla, Norma Rodriguez, and Luigi Capodieci. Clustering and pattern matching for an automatic hotspot classification and detection system. In *Design for Manufacturability through Design-Process Integration III*, volume 7275 of *Proceedings of SPIE*, 2009.

[8] Kevin Gourley and Douglas Green. A polygon-to-rectangle conversion algorithm. *IEEE Computer Graphics and Applications*, 3(1):31–36, January 1983.

[9] Jonathan Ho, Yan Wang, Joanne Wu, Ya-Ching Hou, and Kechih Wu. Lithography-simulation-based design for manufacturability rule development: an integrated circuit design house's approach. *Journal of Micro/Nanolithography, MEMS and MOEMS*, 6(3):031008, 2007.

[10] Chi-Yuan Hung, Andrew M. Jost, and Qingwei Liu. Model-based DRC for design and process integration. In *Design for Manufacturability through Design-Process Integration II*, volume 5992 of *25th Annual BACUS Symposium on Photomask Technology*, 2005.

[11] Kevin Lucas, Stanislas Baron, Jerome Belledent, Robert Boone, Amandine Borjon, Christophe Couderc, Kyle Patterson, Lionel Riviere-Cazaux, Yves Rody, Frank Sundermann, Olivier Toublan, Yorick Trouiller, Jean-Christophe Urbani, and Karl Wimmer. Investigation of model-based physical design restrictions. In *Design and Process Integration for Microelectronic Manufacturing III*, volume 5756 of *Proceedings of SPIE*, 2005.

[12] Ning Ma. *Automatic IC Hotspot Classification and Detection using Pattern-Based Clustering*. PhD thesis, The University of California, Berkeley, 2009.

[13] Ning Ma, Justin Ghan, Sandipan Mishra, Costas Spanos, Kameshwar Poolla, Norma Rodriguez, and Luigi Capodieci. Automatic hotspot classification using pattern-based clustering. In *Design for Manufacturability through Design-Process Integration II*, volume 6925 of *Proceedings of SPIE*, 2008.

[14] Y. C. Pati and T. Kailath. Phase-shifting masks for microlithography: automated design and mask requirements. *Journal of the Optical Society of America A*, 11(9):2438–2452, 1994.

[15] J. Andres Torres and Nick Cobb. Study towards model-based DRC verification. In *25th Annual BACUS Symposium on Photomask Technology*, volume 5992 of *Proceedings of SPIE*, 2005.

[16] Clair Webb. Layout rule trends and effect upon CPU design. In *Design and Process Integration for Microelectronic Manufacturing IV*, volume 6156 of *Proceedings of SPIE*, 2006.

[17] Jingyu Xu, Subarna Sinha, and Charles C. Chiang. Accurate detection for process-hotspots with vias and incomplete specification. In *The International Conference on Computer-Aided Design*, pages 839–846, 2007.

[18] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai. Efficient process-hotspot detection using range pattern matching. In *The International Conference on Computer-Aided Design*, pages 625–632, 2006.

[19] Peng Yu, David Z. Pan, and Chris A. Mack. Fast lithography simulation under focus variations for OPC and layout optimizations. In *Design and Process Integration for Microelectronic Manufacturing IV*, volume 6156 of *Proceedings of SPIE*, 2006.

[20] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.