Seedy Images: The SKIZ Operator and VOISE Algorithm for for Image Segmentation by Voronoi Diagrams

Jack Scantlebury

Department of Physics

University College London

2018

Contents

Abstract

WRITE ABSTRACT: BRIEF SUMMARY OF PROJECT AND RESULTS

1 Summary of Achievements

It has been impressed upon the students on this course that it should be clear what has been achieved over the course of each project. In the spirit of this, and in the interests of clarity, these are briefly summarised as follows:

1. A robust, efficient C++ implementation of the SKIZ operator algorithm as defined in [12], along with appropriate MEX Matlab bindings for extremely fast computation of discrete Voronoi diagrams

2. Extensive benchmarking of the SKIZ operator in its MEX binary form, compared with previous incremental and 'full' Voronoi diagram implementations. Improvements of between 18 and 105 times over previous incremental implementation and up to 10,000 times over 'full' native Matlab Qhull Voronoi diagram implementation

3. Incorporation of SKIZ MEX binaries into the existing VOISE algorithm Matlab code [6]

4. Profiling of existing VOISE code and conversion to C++ MEX of other bottleneck functions

5. Application of VOISE algorithm to images collected by Hubble space telescope of Jovian magnetosphere, including cluster analysis and other machine learning techniques.

2 Voronoi Diagrams: A Crash Course

While the concept of a Voronoi diagram (VD) was present as long ago as the 17th century, a mathematically rigorous definition and formalisation of the the general case is credited to Ukrainian mathematician Georgy Voronyi in 1908 14. In loose terms, a VD can be summarised as a space in n dimensions which has been split into different 'Voronoi regions' (VRs). Coordinates in the same VR are closer, according to some valid distance measure, to the same 'feature point' or seed s than any other seed in the space. The boundary of these VRs is the VD. An example of a VD in $\mathbb{R}^2$ which uses Euclidian distance can be seen in Fig. 1.

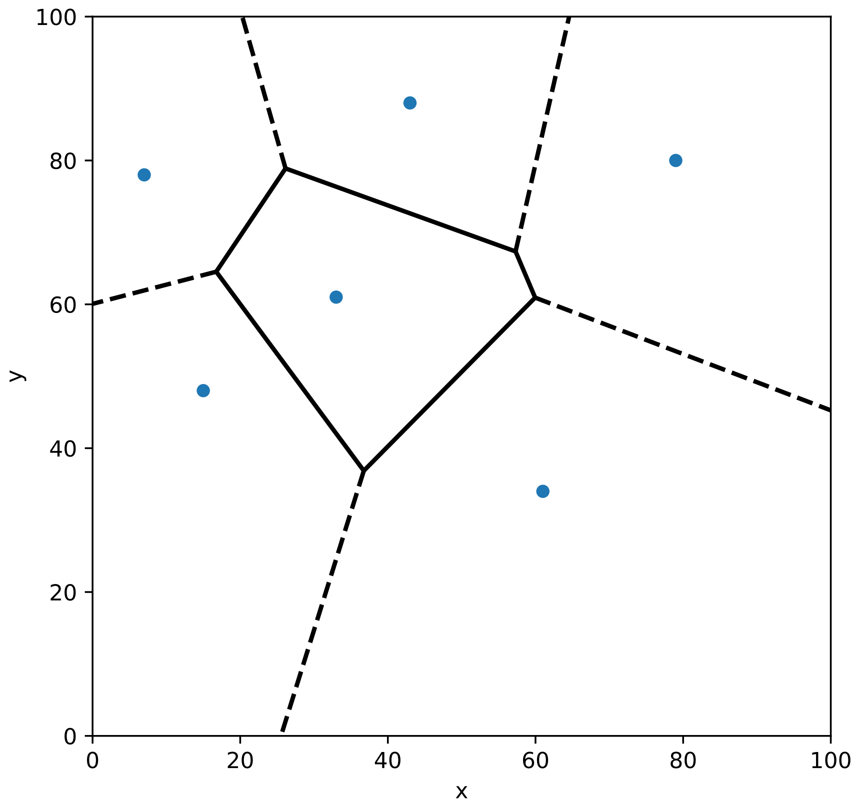

Figure 1: A simple Voronoi diagram in $\mathbb{R}^2$. Dots are seeds, dashed lines are half lines and solid lines are lines.

The seeds on the outside of the diagram are in unbounded, infinitely large VRs, and the lines that separate these are half-lines. The set of all seeds of this type form the convex hull P of the set of all seeds S (so $P \subseteq S$). The only fully enclosed VR in this case is in the middle.

More formally, the VR of a seed $\mathcal{R}(s)$ is defined in any metric space M with associated distance function d as:

$$\mathcal{R}(s) = \bigcap_{r \in S \setminus s} \{p \in M : d(p,s) < d(p,r)\} \tag{2.1}$$

The edge between two neighbouring VRs is then:

$$\partial\mathcal{R}(s) \cap \partial\mathcal{R}(r) = \bigcap_{t \in S \setminus \{r,s\}} \{p \in M : d(p,s) = d(p,r) \leq d(p,t)\} \tag{2.2}$$

Finally, the VD is composed of the edges:

$$\mathcal{V}(S) = \bigcup_{\substack{r,s \in S \\ s \neq r}} (\partial\mathcal{R}(s) \cap \partial\mathcal{R}(r)) \tag{2.3}$$

Equation 2.1 defines a half plane if Euclidian distance is used. $\mathcal{R}(s)$ is the intersection of half planes, which are themselves convex sets, so $\mathcal{R}(s)$ is a convex set. A proof for $\mathbb{R}^2$ can be found in [12], proposition 1.

2.1 Delaunay Triangulation

The dual $\mathcal{V}(S)$ is a Delaunay triangulation (DT or $\mathcal{D}(S)$). In $\mathbb{R}^2$, a DT is the triangulation of a set of points such that no point in the set lies in the circumcircle of any triangle in the triangulation. The points here are the seeds in the VD. A DT maximises the smallest angle, avoiding extremely thin triangles. This can be generalised to an arbitrarily high dimension; an example of a DT is shown in Fig. 2.

Connecting the circumcentres of neighbouring triangles in $\mathcal{D}(S)$ recovers $\mathcal{V}(S)$; in this way, knowledge of one representation can be used to find the other.

2.2 Degenerate Conditions

The above discussion about Delaunay triangulations leads naturally to the problem of degenerate seed configurations. It should be noted that when working in a restricted space, as we must when dealing with images made up of a finite number of pixels, the probability of observing these conditions grows from infinitessimal to improbable. The two classes that must be handled by any robust algorithm are cocirulcarity and colinearity, neither of which are trivial obstsacles and both of which are dealt with in later chapters. For now, definitions will suffice.

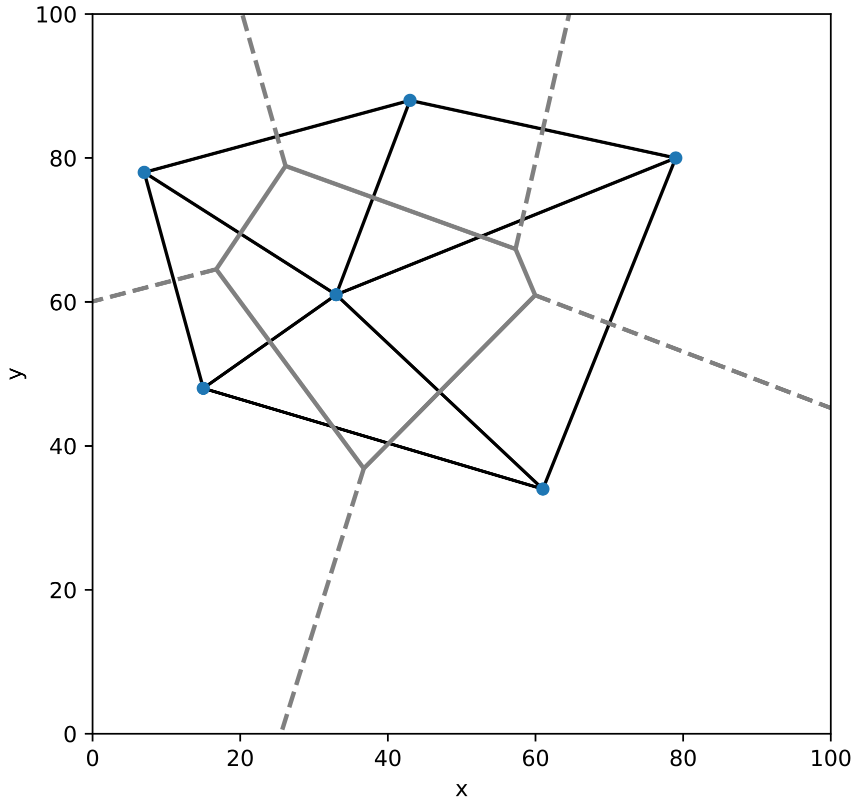

Figure 2: Voronoi diagram (gray) and Delaunay triangulation (black) in $\mathbb{R}^2$. Seeds are blue circles.

2.2.1 Cocircularity

$\mathcal{D}(S)$ is only a unique triangulation if there are precisely three seeds on the circumcircle of each triangle. If there are n cocircular points, there is no unique local (and therefore global) solution. In fact, there are $\binom{n}{3}$ triangles that could be formed.

2.2.2 Collinearity

If three or more neighbouring seeds form a straight line, there is no $\mathcal{D}(S)$ because it not possible to form a circumcircle. In this circumstance, $\mathcal{V}(S)$ still exists, although most algorithms (including Fortune's Algorithm) do not handle this eventuality.

3 Motivation

Voronoi diagrams are well-studied, and have found applications in almost every quantitative field. In computational chemistry, VDs in $\mathbb{R}^3$ - where the diagram is defined by the location of the nuclei - can be used to model charge distribution in molecules [3] and in crystallography, Weigner-Seitz cells and the space packing of metallic lattices can be modelled in a similar fashion. In biology, they can be used to model the growth of rival colonies of cells, the growth patterns of forests and the growth of bone tissue. They are used in engineering for finding triangulations that avoid small angles in finite element analysis [13], as well as in anthropology, cartography, robotics, geography, metallurgy and meteorology.

Of course, the particular interest here the use of VDs in astronomy. The 'cellular structure of the universe' is explored in [15], where VDs are used to expain the distribution of cosmological entities, and the size of the voids found between them. In [10], a VD is defined by the galaxies in images taken by telescopes, and the VD is used to build a picture of larger structures such as clusters. Other work in this area can be found in [8] and [11].

The work in this project builds on work in two papers: the VOISE algorithm [6] and the SKIZ Operator algorithm [12]. We shall outline the former, and explain the utility of the latter in its efficient implementation.

3.1 The VOISE Algorithm

The VOronoi Image SEgmentation algorithm (VOISE) is a self-regulating, iterative method for image segmentation - that is, the partitioning of images into regions $\mathcal{R}(s)$ where pixels in the same region are similar according to some measure (pixel intensity, for example). A segmented image has a number of advantages as a representation over its original source, some of which will become clear in the following discussion.

While VDs can be defined in arbitrary dimensions, for this discussion, we are working in what can be best described as 'pixel-space'. This is $\mathbb{R}^2$ restricted to integer values between zero and some maximum, defined by the resolution of the image. In [12], this this space is denoted by W, but here as in [6], we shall call it Ω. All seeds s and regions $\mathcal{R}(s)$ are only defined in Ω, such that $\mathcal{R}(s_i)$ is all of the pixels in the Voronoi region defined by seed s_i.

There are four stages to VOISE: initialise, divide, merge and regularise. Initialisation begins the process with a random distribution of seeds and an initial VD; in the dividing phase, Voronoi seeds are added to the image according to the measure homogeneity found in each Voronoi region. In the merging phase, neighbouring regions which are similar are merged, in order to remove excessive granularity of the resulting representation. In third and final stage, the seed positions which cause the often irregularly shaped tiles from the first two stages are changed, resulting in a more regular tesselation.

3.1.1 Initialisation

In order to be able to evaluate intra-cellular homogeneity, some initial VD must be generated which does not make any assumptions on the underlying structures to be extracted. Seed generation under a uniform distribution over a bounded region is a binomial point process. We then calculate the VD for those seeds, restricted to Ω, after which we can use information from the pixels in the raw image to better inform our placement of seeds in subsequent iterations.

3.1.2 Division

Most of the useful information to be extracted by VOISE is generated in this, the second and most important phase. After the initial distribution of seeds, each $\mathcal{R}(s_i), s_i \in S$ has associated with it a set of pixels in its region. The tile associated with each seed can then be filled according to some criterion, for example the median pixel value, to generate the representation at each iteration.

In order to continue creating cells in areas of heterogeneity, the homogeneity of a Voronoi region associated with seed s_i is defined:

$$\chi(\mathcal{I}, s_i) = \frac{\max_{p\in\mathcal{R}(s_i)}[\mathcal{I}(p)] - \min_{p\in\mathcal{R}(s_i)}[\mathcal{I}(p)]}{||\chi(\mathcal{I}, S)||_\infty} \tag{3.1}$$

where $\mathcal{I}$ is the image, $\chi(\mathcal{I}, s_i)$ is some merit function (such as pixel intensity) which we take as the measure of homonegeity of the pixels in $\mathcal{R}(s_i)$. The denominator is the largest value of the range of the numerator over all cells, so that $\chi(\mathcal{I}, s_i) \in [0, 1] \quad \forall s_i \in S$. This means that a low $\chi(\mathcal{I}, s_i)$ corresponds to a cell where pixels are similar and conversely a high $\chi(\mathcal{I}, s_i)$ means pixels are dissimilar. For each VR, we can then check if the homogeneity is below a threshold χ_m such that $\chi(\mathcal{I}, s_i) \geq \chi_m$, we split it by adding seeds and re-drawing the VD.

How many seeds to add, and where to place them, is explored in [6]. The number of seeds added is equal to the number of vertices of the polygon formed by the VR, which is the same as the number of neighbouring seeds $|\mathcal{N}(s_i)|$. Each new seed corresponds to a single vertex, and is placed somewhere on the line connecting that vertex to the seed of the inhomogeneous VR; one of the algorithm's parameters, $w_s \in (0, 1)$, is the proportion of the line length away from the original seed that the new seeds should be placed. Fig. 3 shows the effects of varying this parameter on the placement of seeds and the resulting VDs. A compromise of $w_s = 0.625$ was found empirically to be a good trade-off between convergence and relative size of the new regions.

The threshold χ_m is calculated using statistics on the set of homogeneity measures, $\{\chi(\mathcal{I}, s_i), s_i \in S\}$. The parameter p_D is the proportion (or empirical probability) of VRs that are designated homogeneous on any given iteration, regardless of absolute values:

$$\mathrm{P}(\chi(\mathcal{I}, s_i) < \chi_m) = \frac{p_D}{100} \tag{3.2}$$

Because regions with high $\chi(\mathcal{I}, s_i)$ are split into regions with lower $\chi(\mathcal{I}, s_i)$, the denominator in eq. (3.1) is monotonically decreacing as seeds are added. This has the effect of increacing $\chi(\mathcal{I}, s_i)$, increasing the strictness of the criteria. In this way, the features on the smaller length scales are picked up by VOISE.

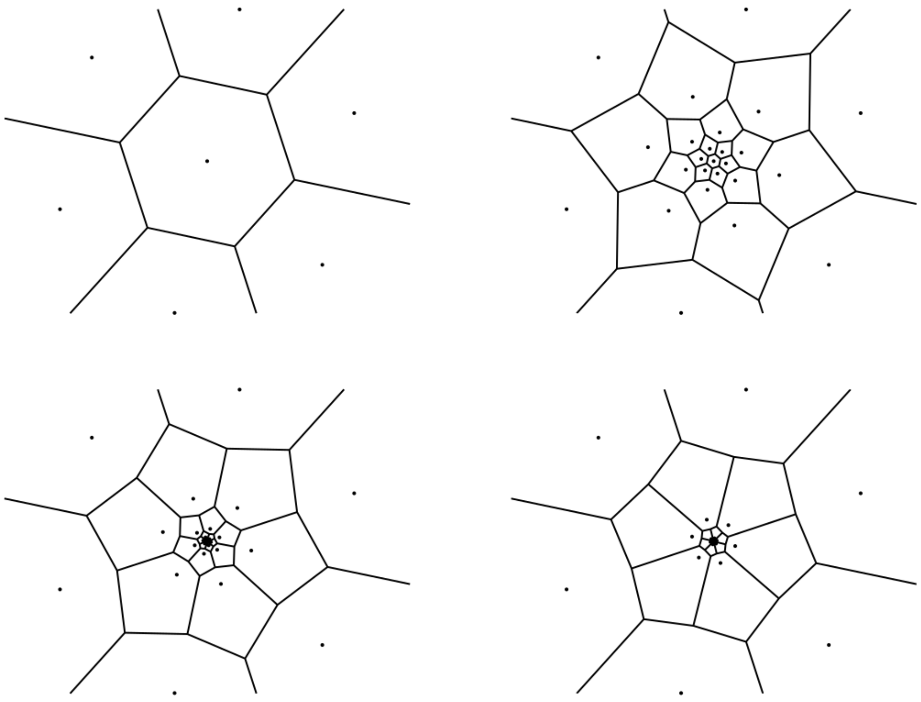
Figure 3: (From [6]) Effects of seed placement. Seeds are added to original VD (top right) recursively for three iterations, with $w_s = 0.25$ (top right), $w_s = 0.5$ (bottom left) and $w_s = 0.75$ (bottom right).

A p_D of 70-80%, with the median as merit function χ, was found in [6] to extract only the features of highest contrast, whereas $p_D = 60 - 70\%$ results in better detection of smaller or more noisey features but worse detection of high contrast features.

In order to force convergence under these changing conditions, a minimum inter-seed distance is enforced, so that the process stops either when all VRs meet the homogeneity criteria or no more seeds can be legally added. At this point, the number of seeds is at a maximum and the image is at its most fragmented.

3.1.3 Merging

Once a VR is deemed inhomogeneous in the division phase, seeds are added with no reference to the underlying image but rather according to a set of fixed geometric rules. This means that redundancy is introduced, where seeds are added which create adjacent VRs that are similar to one another, which has the undesirable effect of feature splitting. In order to reverse this, redundant seeds must be removed algorithmically, and as such, all VRs are checked for the three conditions. In the following, μ_i can be any metric over all pixel intensities in a VR, chosen in [6] to be the median.

(a) The VR must be homogeneous according to dynamic, statistically informed threshold analogous to that in the dividing phase:

$$\chi(\mathcal{I}, s_i) < \chi_m \tag{3.3}$$

where

$$\mathrm{P}(\chi(\mathcal{I}, s_i) < \chi_m) = \frac{p_M}{100} \tag{3.4}$$

p_M is another parameter to be specified in the configuration of VOISE.

(b) There exist one or more seeds in the neighbourhood of the seed, $\{s_j \in \mathcal{N}(s_i)\}$, which are themselves homogeneous according to (a), and who's μ_j is lower than a set relative threshold $\varDelta\mu$:

$$|\mu_i - \mu_j| < \varDelta\mu|\mu_i| \tag{3.5}$$

(c) The fraction of the total perimeter ($\mathcal{P}$) of the VR which is shared with neighbours which are not similar according to eq. (3.5) ($\mathcal{L}$) is not above the stipulated threshold $\varDelta\mathcal{H}$:

$$\frac{\mathcal{L}}{\mathcal{P}} < \varDelta\mathcal{H} \tag{3.6}$$

If all three conditions are met, the seed is removed and the VD is updated. The neighbouring cells then expand to include the pixels previously allocated to that seed. Because $\chi(\mathcal{I}, s_i)$ can only increase with the removal of seeds, condition (a) becomes more difficult to fulfill and so the merge phase concludes naturally when no more regions meet all three conditions. A more lax (larger) p_M leads to less merging and therefore a representation with more seeds. A smaller value of p_M gives a representation that more sparse, and results in a higher compression.

3.1.4 Regularlisation

Usually, the VRs obtained by the first three phases are irregular, with many acute angles and elongated polygons. In order to remedy this, an iterative process of seed movement can be employed, where each seed is moved to the centre of mass ξ_i of the VR:

$$\xi_i = \frac{\sum_{\mathbf{p} \in \mathcal{R}(s_i)} \mathbf{p}\rho(\mathbf{p})}{\sum_{\mathbf{p} \in \mathcal{R}(s_i)} \rho(\mathbf{p})} \tag{3.7}$$

Pixel intensity is used as the density function at pixel $\mathbf{p}$, $\rho(\mathbf{p})$, which is close to uniform within each VR after the previous three phases. Because the VD changes with the changing seed positions, this process must be iterative and stops either when the difference between seed position s_i and ξ_i is less than 1 for all $s_i \in S$ (because both seeds and the corresponding centres of mass are only defined in $\varOmega$), or when a predetermined iteration limit is reached. After regularlisation, the process is complete.

3.1.5 Applications of the VOISE algorithm

The tesselations arrived at by VOISE have potential for the removal of errors caused by dust particles in images taken both on earth and from space telescopes, as long as a granularity is used that is high so as not to remove features but lower than the level required to be significantly affected by blemishes. In [7], the authors of VOISE go on to use the seeds generated at the end of the division phase to estimate the disc parameters of planetary bodies, and in [9], the seeds are used in Partitioning Around Metroids (PAM) clustering [2] in the study of images of auroral emissions at the poles of Jupiter. **Another possible use for VOISE include using the resulting seed coordinates as input vectors into machine learning algorithms, especially if images are labelled, which could provide a new approach to image recognition.**

3.1.6 Previous Implementation of VOISE

While the VOISE algorithm has been implemented previously [6], the method of VD recalculation is slower than is optimum. In that paper, VD calculation does borrow some techniques from SKIZ and is dynamic, but full integration of the SKIZ operator remains to be seen. While running time is not a concern when the number of images to be processed is low, if the VOISE algorithm is to be scaled up and its outputs used as input vectors for machine learning algorithms then this is no longer the case.

3.2 The SKIZ Operator Algorithm

VOISE requires recalculation of the VD after each iteration. In [12], it is proven that adding and removing seeds only affects the VD in the neighbourhood of the added or removed seeds, a fact which is exploited to provide a dynamic algorithm which can be updated at a greatly reduced computational cost with respect to recalculating the entire VD. Using this, as well as a compiled language with a well optimised compiler (C++ with GCC), the VOISE algorithm can be made extremely fast.

3.2.1 Data structures

The SKIZ algorithm uses two matrix data structures, ν and λ, both of which have one element for each pixel. Their dimensions are the same as those of the image, but transposed; λ_{ij} contains the ID of the closest seed for the pixel in the i^{th} column of the j^{th} row of the image, with ν_{ij} being equal to zero if the closest seed is unique, or 1 if there exist two or more closest seeds:

$$\lambda_{ij} = s \quad \text{s.t.} \quad d(p,s) \leq d(p,r) \quad \forall r \in S \backslash \{s\} \tag{3.8}$$

$$\nu_{ij} = \begin{cases} 0 : d(p,\lambda_{ij}) < d(p,r) \quad \forall r \in S\backslash\{\lambda_{ij}\} \\ 1 : d(p,\lambda_{ij}) = d(p,r) \text{ for some } r \in S\backslash\{\lambda_{ij}\} \end{cases} \tag{3.9}$$

where seed p has coordinates $(p_1, p_2) = (j, -i)$, so that a cartesian (bottom left, column-first) pixel coordinates match up with (top-left, row-first) matrix coordinates. From hereoanfter, the distance operator d is assumed to be the squared Euclidian distance in $\mathbb{R}^2$ such that:

$$d(p, s) = (p_1 - s_1)^2 + (p_2 - s_2)^2 \tag{3.10}$$

Squared distance is used because for $p, s \in \Omega$, $d(p, s)$ is a positive integer and we avoid the problems of floating point arithmetic. In C++, ν and λ are stored as `Eigen::Array` objects [5], which are stored internally arrays, bringing the associated efficiencies, but can be accessed using the `()` operator by row and column index as well as a host of other useful features. Another extremely important structure is the list of neighbouring cells $\mathcal{N}(s_i)$ (`Nk` in code). In matlab, this is stored as a cell array, which in C++ is translated to as `std::map<double, std::vector<double>>`, such that the vector of neighbouring seed IDs can be accessed using `Nk.at(seed_id)`. At this point, it should be noted that for Matlab/C++ compatability, many variables (such as IDs) that would usually be stored as integers are stored as doubles. We now have the required data structures in place for efficient calculation of VDs.

3.2.2 Checking Voronoi Region Bounds (getRegion.cpp)

Imagine that a seed s^* is added to a VD, and that there is a pixel for which we would like to know the associated VR. One way of checking would be to compare $d(p, s_i) \quad \forall s_i \in S_k \cup \{s^*\}$ and choosing s_i with the lowest distance, however that operation would require the calculation of $(|S_k| + 1)$ distances and $(|S_k| + 1)$ comparisons.

Instead, it is proven in [12, sec. 2.2] that checking if a pixel is within a VR $\mathcal{R}(s)$ can be reduced to checking two inequalities. In sec. 2 of that paper, it is proven that the half-plane H which represents points that are closer to point

p than q can be represented by an inequality:

$$H(p,q) = \{x \in \mathbb{R}^2 : d(x,p) < d(x,q)\} = \{x \in \mathbb{R}^2 : ax_1 + bx_2 > c\} \quad (3.11)$$

$$\begin{aligned} \text{where} \quad & a = p_1 - q_1 \\ & b = p_2 - q_2 \\ & c = (p_1^2 + p_2^2 - q_1^2 - q_2^2)/2 \end{aligned}$$

It is also proven that $\mathcal{R}(s_i)$ is defined only by the intersection of halfplanes between s_i and its neighbours $\mathcal{N}(s_i)$; The number of inequalities of the form (3.11) is therefore limited by $(|\mathcal{N}(s_i)|$. If we are finding whether a point is in the halfplane $H(s_i, r), r \in \mathcal{N}(s_i)$, eq. (3.11) can be rearranged into one of the following inequalities, depending on the values of the coordinates s_1, s_2, r_1 and r_2:

$$s_1 - r_1 : \begin{cases} = 0 : \begin{cases} s_2 > r_2 : i < \frac{s_2+r_2}{2} \\ s_2 < r_2 : i > \frac{s_2+r_2}{2} \end{cases} \\ < 0 : \quad j < \frac{s_1^2+s_2^2-r_1^2-r_2^2}{2(p_1-q_1)} - (s_2 - r_2)i \\ > 0 : \quad j > \frac{s_1^2+s_2^2-r_1^2-r_2^2}{2(p_1-q_1)} - (s_2 - r_2)i \end{cases} \quad (3.12)$$

The most common of these are where $s_1 \neq r_1$, which gives a set of upper and lower bounds on j for each row i. If the value of j of a pixel on row i is above the maximum lower and below the minimum upper bounds, then it belongs to the VR of the seed s. Where $s_1 = r_1$, j cancels and the condition is soley based on the value of i. At this point, we have an $I \times 2$ matrix of lower and upper bounds for each row, which any pixel can be checked against for membership of the VR, where I is the total number of rows.

3.2.3 Adding a seed (addSeed.cpp)

The subscript k is used here to denote iteration number - it increments whenever a seed is added or removed. $\mathcal{N}_k$ is the map of neighbour relationships at iteration k, with S_k defined similarly. When a seed s^* is added, λ_{ij} and ν_{ij} change according to:

$$[\lambda_{ij,k+1}, \nu_{ij,k+1}] = \begin{cases} [\lambda_{ij,k}, \nu_{ij,k}] & \text{if } d(p, \lambda_{ij,k}) < d(p, s^*) \\ [\lambda_{ij,k}, 1] & \text{if } d(p, \lambda_{ij,k}) = d(p, s^*) \\ [s^*, 0] & \text{if } d(p, \lambda_{ij,k}) > d(p, s^*) \end{cases} \tag{3.13}$$

If we know $\mathcal{N}_{k+1}(s^*)$, we can use the inequalities found by eq. (3.12) to only check values of λ and ν that are in $\mathcal{R}(s^*)$, minimising the number of comparisons. We must therefore first find the new neighbour relationships of all seeds, $\mathcal{N}_{k+1}(s_i), s_i \in S_k \cup s^*$.

3.2.4 Finding $\mathcal{N}_{k+1}$

The approach to finding $\mathcal{N}_{k+1}$ described in [12] borrows from the *incremental method* [4], and involves building up $\mathcal{N}_{k+1}(s^*)$ by finding boundaries of $\mathcal{R}(s^*)$. Beginning with $\mathcal{N}_{k+1}(s^*) = \{\lambda\}$ where λ is the closest seed in S_k to the point now occupied by s^*.

References

[1]

[2] *Wiley Series in Probability and Statistics*, 1990.

[3] Célia Fonseca Guerra, Jan-Willem Handgraaf, Evert Jan Baerends, and F. Matthias Bickelhaupt. Voronoi deformation density (vdd) charges: Assessment of the mulliken, bader, hirshfeld, weinhold, and vdd methods for charge analysis. *Journal of Computational Chemistry*, 25(2):189–210, 2003.

[4] P. J. Green and R. Sibson. Computing dirichlet tessellations in the plane. *The Computer Journal*, 21(2):168–173, 1978.

[5] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[6] P. Guio and N. Achilleos. The voise algorithm: a versatile tool for automatic segmentation of astronomical images. *Monthly Notices of the Royal Astronomical Society*, 398(3):1254–1262, 2009.

[7] Patrick Guio and Nicholas Achilleos. A new method for determining geometry of planetary images. 10 2010.

[8] S. Ikeuchi and E. L. Turner. Quasi-periodic structures in the large-scale galaxy distribution and three-dimensional voronoi tessellation. *Monthly Notices of the Royal Astronomical Society*, 250(3):519–522, 1991.

[9] Patrick G. Achilleos N and R. Prange. A new method for determining geometry of planetary images. 10 2010.

[10] M. Ramella, W. Boschin, D. Fadda, and M. Nonino. Finding galaxy clusters using voronoi tessellations. *Astronomy and Astrophysics*, 368(3):776–786, 2001.

[11] Barbara S. Ryden. Measuring q 0 from the distortion of voids in redshift space. *The Astrophysical Journal*, 452:25, 1995.

[12] R. E. Sequeira and F. J. Preteux. Discrete voronoi diagrams and the skiz operator: a dynamic algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1165–1170, 1997.

[13] Suzanne Shontz. *Proceedings of the 19th International Meshing Roundtable.* Springer Berlin Heidelberg, 2010.

[14] Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

[15] L Zaninetti. On the large-scale structure of the universe as given by the voronoi diagrams. *Chinese Journal of Astronomy and Astrophysics*, 6(4):387–395, 2006.