Approximating Imprecise Planar Tesselations with Voronoi Diagrams

Narciso Aguilera
PhD Program
University of Valladolid, Spain
and Univ. Nacional de Ingeniería,
Managua, Nicaragua
Email: njaguilera@uni.edu.ni

Belén Palop
Didáctica de la Matemática
Departmento de Didáctica
University of Valladolid
Segovia, Spain
Email: bpalop@infor.uva.es

Hebert Pérez-Rosés
Department of Mathematics
University of Lleida, Spain
and conjoint fellow
Univ. of Newcastle, Australia
Email: hebert.perez@gmail.com

***Abstract*—Natural growth processes tend to generate shapes in the form of imprecise planar tesselations, where the tiles do not match exactlyand leave some space among them. In this paper we model these tesselations by means of Voronoi Diagrams. We look for the set of 2D-sites whose Voronoi Diagram better approximates the given imprecise tessellation. Since we conjecture the Inverse Voronoi Problem (also known as Voronoi-fitting Problem) to be NP-hard, we describe in this paper a heuristic algorithm that looks for the optimal set of sites.. We study the algorithm's performance and validity on a set of tesselations extracted from real-life images. With the aid of these experiments, we find optimal values for a tunable parameter of the algorithm. In the long run, our main goal is to develop a tool that can automatically analize an image of a tessellation that is expected to be modelled with a Voronoi Diagram (e.g., pictures from chrystals, trees in a forest, etc), and decide whether the growth process was or was not affected by some external force. This inverse computation should be able tell how far is the image from its theoretical model.**

Keywords–Planar tesselations; Voronoi Diagram; local search; Inverse Voronoi Diagram.

I. Introduction

Voronoi Diagrams (or Dirichlet tesselations) are a fundamental geometric structure with many applications in Graphics and Image Processing, among other uses. Informally, the Voronoi Diagram generated by a set S of points in the Euclidean plane is a subdivision of the plane into (convex) cells, where each cell is associated with one point $p \in S$, and consists of the points on the plane that are closer to p than to any other $q \in S$. In other words, the cell associated with some point p can be thought of as a sort of *sphere of influence* of p. For the precise definition, as well as the most important properties of Voronoi Diagrams, we refer the reader to any one of the many standard texts on the subject, such as [1].

The above definition lends itself very naturally to many generalizations, such as using sites other than points, considering metrics other than the Euclidean metric, dimensions higher than two, or surfaces other than the plane. Since many growth processes in nature follow this very simple model, in this paper, we will restrict ourselves to the Euclidean plane, the Euclidean metric and point sites. In fact, due to the applied focus of this paper, we will restrict the area of study to a bounded rectangle in the 2D-plane where the tessellation is given, and perform the algorithm's adjustments using real-life image pictures.

Planar Euclidean Voronoi Diagrams have been applied to describe images and other visual or spatial patterns since the late 1970s [2], [3] and [4]. In particular, [2] attempts to approximate a cellular pattern by a Voronoi Diagram and this is, to the best of our knowledge the first reference to what we now call *the Inverse Voronoi Problem*. In this paper, we present an algorithm that focuses on finding how a picture, that looks like a Voronoi Diagram to the naked eye, can (or cannot) be approximated by that model. We conjecture that finding the model that better approximates the final picture of a growth process can give useful information on how the process itself happened and if unforseen external circumstances may have altered it. The Inverse Voronoi Problem (IVP) can be stated as follows:

Problem 1 (Inverse Voronoi Problem): Given a Voronoi Tesselation T in the Euclidean Plane, find the set of points P that generate T.

After [2], this problem was addressed again in [5] and [6]. The approach followed by Suzuki and Iri in [6] is similar to that of [2]: their aim is to find a Voronoi Diagram that best approximates the tesselation T, according to some measure of approximation. On the other hand, in [5] Ash and Bolker try to find the Voronoi Diagram that fits T *exactly*. Unfortunately, this is not always possible, since not every tesselation T corresponds exactly to a Voronoi Diagram, even if all the cells of T are convex.

In this paper, we continue along the lines of [2] and [6]: We consider a relaxation of the concept of planar tesselation, and we devise an algorithm that tries to find the Voronoi Diagram that best approximates this relaxed tesselation.

In a standard tesselation of the plane, the tiles cover the entire plane, and their intersection is a set of *edges*, with measure zero. However, this ideal situation does not correspond to reality in many cases. In many actual spatial patterns, including images, the cell boundaries may have some thickness, or to put it in another way, the tiles do not match exactly, leaving some space between them. This situation was recently considered in [7], where the Voronoi edges have some constant width $w > 0$. In this paper, we consider a more general situation, which we may call *imprecise planar tesselation*.

Definition 1 (Imprecise planar tesselations): An imprecise (or incomplete) tesselation of a bounded region $R \subseteq \mathbb{R}^2$ is a set $\mathcal{P}$ of polygons, or tiles (not necessarily convex), such that each $P \in \mathcal{P}$ is strictly contained in R, has a non-empty interior; and $\bigcap_{P \in \mathcal{P}} P$ has zero measure.

Note that we preserve the condition that the intersection between cells has zero measure, but we do not require that $\bigcup_{P \in \mathcal{P}} P$ covers the entire Euclidean plane $\mathbb{R}^2$, hence $\mathcal{P}$ is not a tesselation in the usual sense. The set $R - \bigcup_{P \in \mathcal{P}} P$ will be called *the grey area*. When we try to approximate $\mathcal{P}$ by a Voronoi Diagram, we care about where each $P \in \mathcal{P}$ lies with respect to the Voronoi Diagram, but we are indifferent as to where some part of the grey area goes.

Section II describes the heuristic algorithm to approximate planar imprecise tesselations. In Section III, we give examples of imprecise tesselations of the plane taken from real-life applications, and we run the algorithm with those tesselations, in order to analyze its performance. From these experiments, we draw conclusions regarding a certain tunable parameter of the aforementioned algorithm. Please note that, since this is a Work In Progress, the obtained values and the decisions made in order to tune our algorithm are still subject to further research.

II. Approximating imprecise tesselations

In the following, we will assume that $\mathcal{P}$ consists of n polygons $\{P_1, \dots P_n\}$, and our aim is to approximate it by a Voronoi Diagram $\mathcal{V}$ with exactly n cells $\{V_1, \dots V_n\}$, i.e. one per polygon. For each $1 \leq i \leq n$, the Voronoi cell V_i should cover the tile P_i as much as possible, and perhaps some of the gray area, but should refrain from invading any other tile P_j, with $j \neq i$.

Using the area of the missclasified region as a measure of the fitness of a candidate solution, we can define our goal as follows: For each Voronoi cell V_i, we compute the area of the portions of polygons in $\mathcal{P}$ other than P_i that lie inside V_i. The sum of the areas of all regions for each $1 \leq i \leq n$ is the area of the missclasified region. (Note that the portions of P_i that do not belong to V_i will be counted when the containing Voronoi cell is processed.)

Since we will, in practice, never work with the entire plane but with a bounded region, typically a rectangular section R, if $a(X)$ represents the area of a polygon X, we may define the *normalized unwanted area* of $\mathcal{V}$, relative to $\mathcal{P}$, as follows:

$$\begin{aligned} \Upsilon_{\mathcal{P}}(\mathcal{V}) &= \frac{1}{a(R)} \sum_{i=1}^{n} (a(P_i) - a(P_i \cap V_i)) \\ &= \frac{1}{a(R)} \sum_{i=1}^{n} a(P_i) - \frac{1}{a(R)} \sum_{i=1}^{n} a(P_i \cap V_i) \end{aligned} \tag{1}$$

Hence, we treat the problem of approximating $\mathcal{P}$ as an optimization problem, where we have to minimize the objective function Υ, which in general, is a non-linear and non-differentiable function. Note that the first term of (1), i.e. $\frac{1}{a(R)} \sum_{i=1}^{n} a(P_i)$, is constant, therefore, minimizing Υ amounts to maximizing the term $\sum_{i=1}^{n} a(P_i \cap V_i)$.

In order to minimize Υ, we may use any of the well-known heuristics developed for non-linear optimization, such as any variant of local search, simulated annealing, genetic algorithms, etc. (see [8], for instance). In this paper, we settle for a simple variant of local search, which is roughly described in the algorithm in Fig. 1 as a first attempt to explore the solution space. We will also see that looking for this minimum is not an easy task and that, being the optimal location for each point dependant on every point other than itself, careful tuning needs to be performed in order not to *fall* in a local minimum. Starting with the centroids of the tiles, the algorithm is going to try to move one site at a time in any of the 8 main directions.

Input: An imprecise tesselation $\mathcal{P} = \{P_1, \dots P_n\}$, an approximation ϵ.
Output: A set of sites $S = \{s_1, \dots s_n\}$ whose Voronoi Diagram approximates $\mathcal{P}$.

```
                                    ▷ Compute First Candidate Solution
1:  for each polygon $P_i \in \mathcal{P}$ do
2:      $s_i \leftarrow$ centroid of $P_i$
3:  end for
4:  $\mathcal{V} \leftarrow$ Voronoi Diagram generated by $S$
5:  $A \leftarrow \Upsilon_{\mathcal{P}}(\mathcal{V})$
                                    ▷ Improve the solution
6:  while solution still improving do
7:      for each site $s_i \in S$ do
8:          for each direction {N,NE,E,SE,S,SW,W,NS} do
9:              $s'_i \leftarrow$ translation($s_i$,direction,distance)
10:             candidate $\leftarrow S$ with $s'_i$ instead of $s_i$
11:             $\mathcal{V}' \leftarrow$ Voronoi Diagram(candidate)
12:             $A' \leftarrow \Upsilon_{\mathcal{P}}(\mathcal{V}')$
13:             $(s^*_i, A^*) \leftarrow$ best among all directions
14:         end for
15:         $s_i \leftarrow s^*_i$; $A = A^*$
16:     end for
17:     if solution not improving then
18:         distance $\leftarrow$ some factor of distance
19:     end if
20: end while
```

Figure 1. Voronoi Approximation of Imprecise Tesselation

The best among these positions for that particular site, will be chosen and the site will be reassigned to that location. We iterate this process until no site in the current solution can be moved. When this happens, we decrease the distance that we are going to test, looking for locations that are closer to the site. Since we don't expect real-world Voronoi Diagrams to be centroidal, at the beginning, farther positions from the centroid are explored. As the algorithm proceeds, the sites tend to get closer to the optimal location and smaller steps are given. We will see in Section III that how we reduce the distance in those tests, drastically affects the final solution and the running times.

Algorithm in Fig. 1 shows the main idea that we have followed in order to implement the descent method for our problem. Even though many different approaches can be tested and small modifications can slightly improve the final result or the computational time, we describe here the main decisions that have been taken in order to optimize a given imprecise tesellation.

- A candidate solution is found when translating one site in the previous candidate solution. Note that this movement has a direct effect on all Voronoi neighbouring cells and, in the long run, might force the move of any other site in S.
- Translation vectors have been carefully studied showing that 8 directions give good enough approximations in practical cases.
- Translation distance happens to be one of the most important parameters, showing our results that both optimization and performance benefit from a distance reduction by some factor each time the algorithm gets

stuck in a local minimum for the present distance.

- The algorithm terminates when the improvements in the normalized unwanted area is smaller than ϵ after several attempts.

We have implemented and tested several variations on this algorithm in order to find a good balance between achieving a good approximation and the computational time needed to obtain it. Note that, whenever a new candidate solution is tested, the whole Voronoi Diagram might have to be recomputed. We briefly discuss in the following the choices that we have made in our implementation and justify them according to our experimental results:

- If site s_i moves in along a certain direction in one iteration, we have seen that it is very probable that the same direction will be chosen in the next move. Therefore, we have also implemented versions for the algorithm where not all 8 directions are tested, but we move a site if the same direction as the previous iteration already gives an improvement without needing to test the other 7 directions. This choice does reach similar results in the approximation bounds while being more efficient.
- Once all sites are tested and no movement is found to improve the present solution, we decrease the distance for the translation (which will be proportional to its distance to its nearest neighbour). Intuitively, this process allows the points to approach fast their best location and improve within that location afterwards.
- Even though when we approach the best location improvements tend to be small, this condition might be also met during the whole process. We establish the number of rounds in which we will already consider that no further improvement is expected.

III. Computational experiments

In order to test the performance of Algorithm 1 we have chosen some imprecise tesselations, obtained by segmenting real-life images coming from different domains.

The example, Fig. 2 [9] shows a honeycomb, which is an almost perfect hexagonal tiling of the plane. With this image, since the Voronoi Diagram originating perfect hexagonal tilings is centroidal (i.e. generators lie on the centroids of the cells), the objective function is essentially minimized at the initialization step. On the other hand, we have several non-centroidal Voronoi tesselation of the plane. Fig. 3 [10] is a microscopic image of common waterweed, Canadian pondweed (*Elodea canadensis*), an aquatic plant from North America. Fig. 4 [11] is another microscopic image, from a metal's crystalline structure. Finally, Fig. 5 [12] is a satellite image of a farming area in Oxfordshire, UK. Despite the limits between farms not appearing through a natural growth process, their resemblance to a Voronoi Diagram is remarkable (In the best run, $\Upsilon = 4.6\%$, even smaller than our best run for the crystal structure).

All the aforementioned images show a clear pattern of tiles that tesselate the plane in an imprecise manner. We have segmented the images in order to obtain imprecise planar tesselations and, for each one of these tesselations, we have run our algorithm with different step reduction factors. For each experiment, we have measured the number of iterations, the running times and the initial and the final values of the

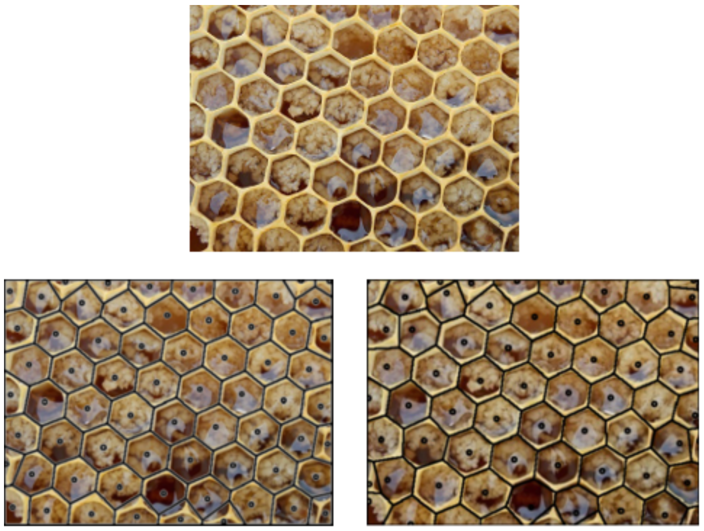
Figure 2. Original Honeycomb picture (top), VD for centroids (left) and optimal VD after running the algorithm (right).

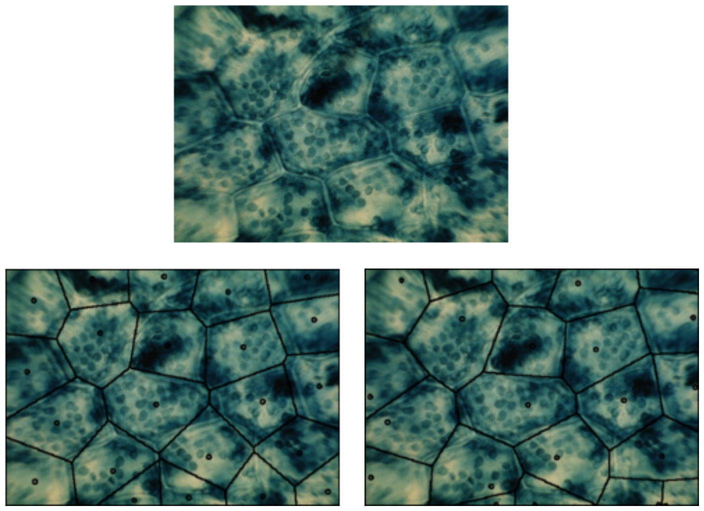
Figure 3. Original Cells of Elodea canadensis picture (top), VD for centroids (left) and optimal VD after running the algorithm (right).

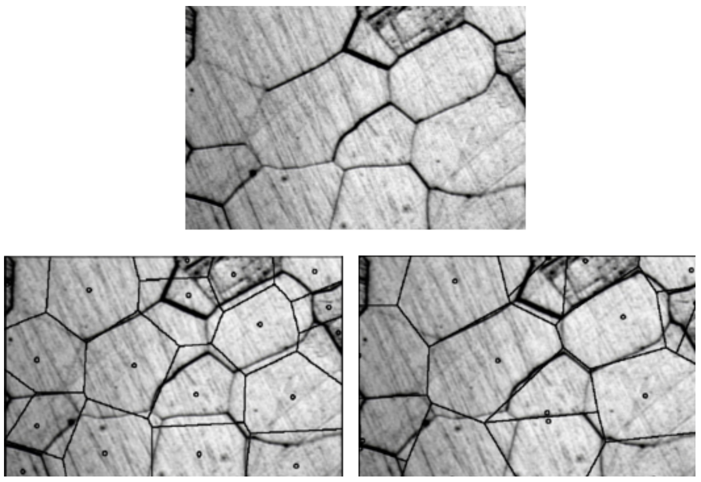
Figure 4. Original Crystal grain picture (top), VD for centroids (left) and optimal VD after running the algorithm (right).

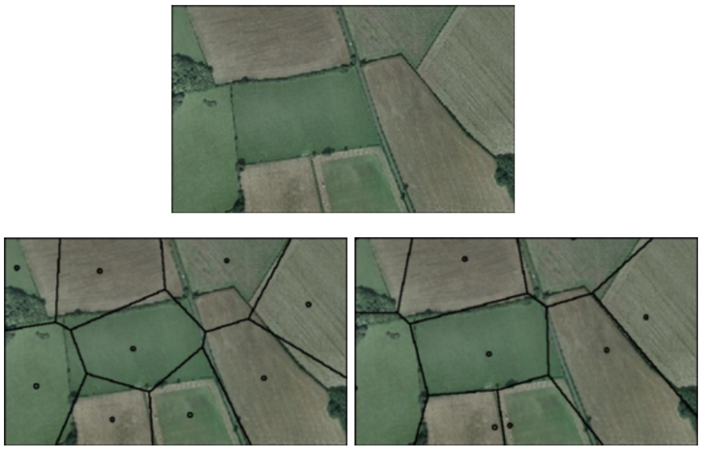

Figure 5. Original Farmland plateau in Oxfordshire (UK) picture (top), VD for centroids (left) and optimal VD after running the algorithm (right).

objective function. We have randomized the order in which the sites are processed in each loop. Even though neither the running times nor the optimal solution is dramatically affected by this randomness, we have run the algorithm ten times for each distance factor, and have taken the average of these ten values for each parameter. In each case we have also computed the relative cost, which is the number of iterations that are necessary to gain one percentage point of accuracy in the approximation. Tables I, II and III contain the results of those measurements. The experiments have been carried out on a personal computer equipped with an Intel Core i5 processor, with 8 GB RAM, running under Windows 10 and the algorithm has been implemented using the R language.

TABLE I. COMPUTATIONAL RESULTS FOR FIG. 3

Initial Υ	10.6%						
Step reduction	0.05	0.15	0.30	0.50	0.70	0.85	0.95
# iterations	228	139	122	115	108	74	3
Time (secs.)	210	122	132	163	238	299	104
Final Υ	7.4%	6.5%	5.5%	5.1%	4.7%	5.8%	9.8%
Relative cost	71.2	33.9	23.9	20.9	18.3	15.4	3.7

TABLE II. COMPUTATIONAL RESULTS FOR FIG. 4

Initial Υ	20.4%						
Step reduction	0.05	0.15	0.30	0.50	0.70	0.85	0.95
# iterations	266	117	119	98	102	59	1
Time (s)	204	91	107	130	216	247	49
Final Υ	16.4%	16.2%	13.3%	12%	10.6%	13%	20%
Relative cost	66.5	28	16.7	11.6	10.6	7.8	2.5

TABLE III. COMPUTATIONAL RESULTS FOR FIG. 5

Initial Υ	11.1%						
Step reduction	0.05	0.15	0.30	0.50	0.70	0.85	0.95
# iterations	238	59	59	59	46	14	0
Time (secs.)	91	23	27	34	46	33	10
Final Υ	6.6%	7.5%	6.3%	5%	4.6%	8.1%	11.1%
Relative cost	53	16.4	12.2	9.6	7	4.6	−−

Fig. 6 shows the degree of approximation achieved by Algorithm 1 for the images above. Although the level of approximation varies substantially from one image to another, note that the maximum approximation is attained in all cases with a step reduction factor approximately equal to 0.7.

Besides the degree of approximation, we are also interested in the amount of work done to achieve the desired approximation. Fig. 7 makes a graphical comparison of the relative cost computed in Tables I, II, and III. Note that the minimum cost is attained in all cases when the step reduction factor approaches 1.

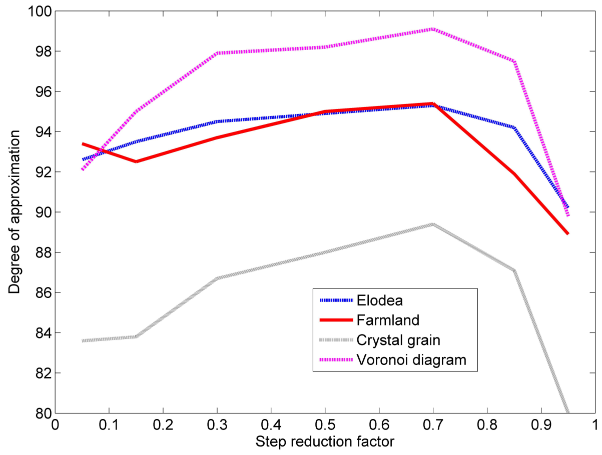

Figure 6. Degree of approximation as a function of the step reduction factor

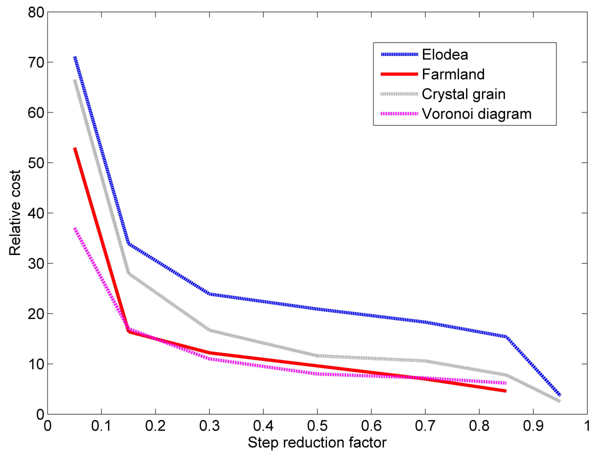

Figure 7. Relative cost as a function of the step reduction factor

IV. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced the concept of imprecise planar tesselation, in order to model images arising in many practical situations. Our experiments show that imprecise planar tesselations can be approximated quite accurately with the aid of Voronoi Diagrams, which provides a nice geometric model for several types of images. Moreover, we have detected a coincidence in relation with the step reduction factor: in all four images, the best approximation is achieved when the step reduction factor is approximately equal to 0.7, and the algorithm performs more efficiently with a larger step reduction factor, although the approximation decreases sharply. This gives us some rough guidelines for tuning the step reduction factor, depending on our priorities at each moment. As a rule of thumb, a step reduction factor between 0.7 and 0.75 seems to be a good choice, leaning towards 0.7 if we want more accuracy, or towards 0.75 if we need more efficiency. However, these guidelines have to be confirmed by more extensive experiments.

There are many ways to improve our method for approximating imprecise planar tesselations. We can replace our algorithm by a more sophisticated metaheuristic, such as simulated annealing, or an evolutionary algorithm. Another improvement may be derived by considering a more general variant of the Inverse Voronoi Problem, where we would allow more than one Voronoi cell per tile. This *Generalized Inverse Voronoi Problem* has already been considered before in [13] and [14], for instance, but these two papers have focused on the *exact* version of the problem only. The approximated version might yield better results, in the sense that a reasonably good aproximation might be found with far fewer Voronoi generators than those used in [13] and [14].

ACKNOWLEDGMENTS

Narciso Aguilera is supported by an EURICA PhD scholarship for Latin American students. The EURICA program is coordinated by the University of Groningen, the Netherlands, together with several associated universities in Europe and Latin America. Belen Palop has been partially supported by project MTM2015-63791-R. Hebert Pérez-Rosés acknowledges partial support from project MTM2013-46949-P, funded by the Spanish Ministry for Economy and Competitiveness (MINECO). The authors would like to thank Manuel Abellanas (Universidad Politécnica de Madrid) for the fruitful discussions and ongoing research on this problem.

REFERENCES

[1] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, Spatial Tesselations: Concepts and Applications of Voronoi Diagrams. John Wiley and Sons, 2000.

[2] H. Honda, “Description of Cellular Patterns by Dirichlet Domains: The Two-Dimensional Case,” Journal of Theoretical Biology, vol. 72, pp. 523–543, 1978.

[3] N. Ahuja, B. An, and B. Schachter, “Image Representation using Voronoi Tesselation,” Computer Vision, Graphics and Image Processing, vol. 29, pp. 286–295, 1985.

[4] M. Tanemura, “Statistical Distributions of Poisson Voronoi Cells in Two and Three Dimensions,” Forma, vol. 18, pp. 221–247, 2003.

[5] M. M. Ash and E. D. Bolker, “Recognizing Dirichlet Tesselations,” Geometriae Dedicata, vol. 19, pp. 175–206, 1985.

[6] A. Suzuki and M. Iri, “Approximation of a tessellation of the plane by a Voronoi diagram,” Journal of the Operations Research Society of Japan, vol. 29, pp. 69–96, 1986.

[7] M. Ferraro and L. Zaninetti, “On the statistics of area size in two-dimensional thick Voronoi diagrams,” Physica A, vol. 391, pp. 4575–4582, 2012.

[8] T. Gonzalez, Ed., Handbook of Approximation Algorithms and Metaheuristics. Chapman and Hall - CRC, 2007.

[9] ”Pixabay,” 2016, URL: https://pixabay.com/es/panal-las-abejas-hex%C3%A1gonos-peine-330755 [accessed: 2016-01-02]

[10] Howard Sidney Thomas — Cell senescence, 2016, URL: http://www.sidthomas.net/SenEssence/Evolution/cellsen.htm [accessed: 2016-01-02].

[11] Wikimedia commons — File: CrystalGrain.jpg, 2016, URL: https://commons.wikimedia.org/wiki/File:CrystalGrain.jpg [accessed: 2016-01-02]

[12] Oxforshire county council — Landscape Types: Farmland Plateau Aerial View, 2016, URL: http://owls.oxfordshire.gov.uk/wps/wcm/connect/occ/OWLS/Home/Oxfordshire+Landscape+Types/Farmland+Plateau/Farmland+Plateau+Aerial+View [accessed: 2016-01-02].

[13] S. Banerjee et al., “On the Construction of a Generalized Voronoi Inverse of a Rectangular Tesselation,” in Proceedings of the 9th Int. IEEE Symp. on Voronoi Diagrams in Science and Engineering, June 27–29, 2012, New Brunswick, NJ, USA. IEEE, pp. 132–137, 2012.

[14] G. Aloupis, H. Pérez-Rosés, G. Pineda-Villavicencio, P. Taslakian, and D. Trinchet, “Fitting Voronoi Diagrams to Planar Tesselations,” Lecture Notes in Computer Science, vol. 8288, pp. 349–361, 2013.