Computers & Graphics (2021)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

An integer representation for periodic tilings of the plane by regular polygons

José Ezequiel Soto Sánchez[a], Tim Weyrich[b], Asla Medeiros e Sá[c], Luiz Henrique de Figueiredo[a,*]

[a] IMPA, Rio de Janeiro, Brazil
[b] University College London, United Kingdom
[c] FGV EMAp, Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received October 23, 2020
Revised December 7, 2020
Revised January 18, 2021

Keywords: tessellations, symmetry, representation schemes, geometric modeling, procedural modeling

ABSTRACT

We describe a representation for periodic tilings of the plane by regular polygons. Our approach is to represent explicitly a small subset of seed vertices from which we systematically generate all elements of the tiling by translations. We represent a tiling concretely by a $(2+n)\times 4$ integer matrix containing lattice coordinates for two translation vectors and n seed vertices. We discuss several properties of this representation and describe how to exploit the representation elegantly and efficiently for reconstruction, rendering, and automatic crystallographic classification by symmetry detection.

1. Introduction

A *tiling* of the plane by polygons is a subdivision of the plane into bounded closed polygonal faces. We consider only *edge-to-edge tilings*, whose faces either are disjoint, share a vertex, or share an edge. A tiling is *periodic* when it is invariant under two independent translations of the plane. Restricting tilings to be periodic and their faces to be regular polygons imposes much rigidity while still allowing much interesting variety (Fig. 1). Therein lies the lure of this subject. Indeed, understanding how to tile the plane periodically with regular polygons is an absorbing subject [1, 2, 3, 4] whose history [5] goes back 400 years to Kepler's book "Harmonices Mundi" of 1619. Yet, a complete classification of these tilings remains elusive.

Besides their intrinsic mathematical interest, aesthetic appeal, and artistic potential, tilings of the plane are useful in graphics and geometric modeling [6, 3, 7, 8]. Crucial to any computational study or application of tilings is a representation that makes it convenient to synthesize, compare, explore, and analyze tilings. Using standard representations of subdivisions of the plane to represent tilings introduces unneeded complexity and, worse, potential numerical problems due to inevitable irrational vertex coordinates. Thus, specialized representations are needed.

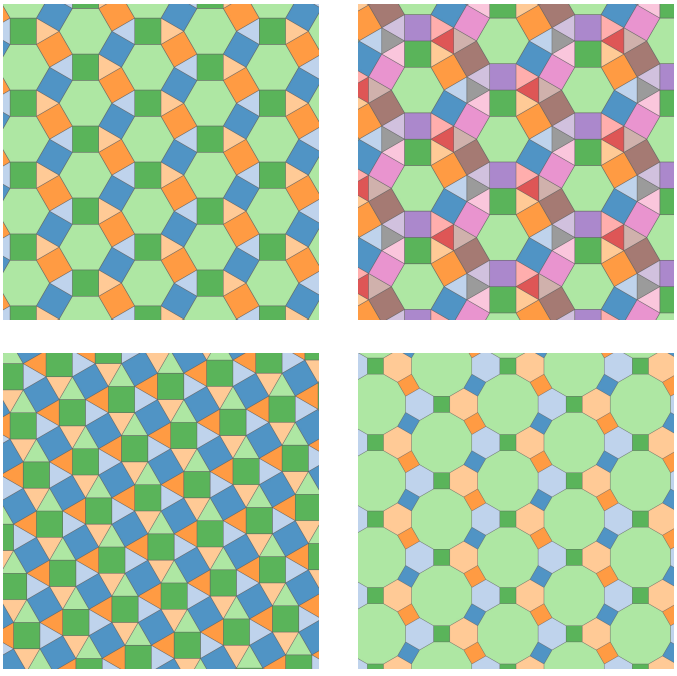


Fig. 1. Variety of periodic tilings of the plane by regular polygons.

*Corresponding author: Luiz Henrique de Figueiredo (lhf@impa.br)

In this paper, we describe in detail a compact, low-complexity representation for periodic tilings of the plane by regular polygons that is built entirely on integer arithmetic and thus supports highly efficient and robust algorithms. We describe our representation in §3: it is a $(2+n)\times 4$ integer matrix containing lattice coordinates for two translation vectors and n seed vertices, from which we generate the whole tiling by translations. These coordinates come from representing the vertices of the tiling as points in $\mathbf{Z}[\omega]$, where ω is the principal 12th root of unity. We describe how to reconstruct tilings from the representation in §4 and §5. We describe how to find and classify the symmetries of a tiling in §6. We discuss the properties of our representation at length in §7, related work in §8, and directions for future work in §9. Large collections of tilings in our representation and reference implementations are freely available at our project web page [9].

Contributions. This paper describes a general, uniform, integer-based representation for periodic tilings of the plane by regular polygons, and discusses its properties in full detail. It significantly expands our previous work on the subject [10, 11]. New material includes: full details on the representation and its motivation, an in-depth discussion of its properties, and new algorithms for tiling reconstruction and symmetry detection that are simple and robust, since they do not rely on nearest-neighbors searches and geometric computations with numerical tolerances.

2. Background, motivation, and overview

From now on, *tiling* means a periodic edge-to-edge tiling of the plane by regular polygons. How does one design a computational representation for such tilings? We start with the natural requirements. A good representation scheme should be:

- *Simple* and be easy to understand, create, and use.
- *Concise* and contain just enough data to reconstruct a tiling.
- *Comprehensive* and represent all possible tilings.
- *Unambiguous* so that each valid representation represents exactly one tiling.
- *Verifiable* so that it is possible to decide whether a given representation corresponds to a valid tiling.
- *Comparable* so that it is possible to decide whether two given representations represent the same tiling.

Since a tiling is an infinite object, it is not possible to represent explicitly all its topological elements. However, since the tiling is periodic and repeats itself, it suffices to represent explicitly a finite subset of its topological elements and explain how to replicate it to cover the whole plane. We call such a subset a *patch* for the tiling.

Standard topological data structures are too general for representing tilings: the rigid geometry and topology of tilings should allow more concise representations than explicitly listing all vertices, edges, and faces of a patch (see §4). A common way to represent a tiling concisely is to exploit its symmetries to reduce the amount of data needed (see §8). A *symmetry* of the tiling is a rigid transformation (translation, rotation, reflection) of the plane that leaves the tiling invariant. To use the minimal amount of data to represent a tiling, we need to understand its symmetries completely. The mathematics for this are quite well known: there are exactly 17 symmetry groups, and a simple procedure finds the symmetry group of a tiling from its symmetries. Each tiling has a *minimal crystallographic fundamental region* that reconstructs the tiling when replicated by repeated application of its symmetries. This region is the smallest possible patch for the tiling and is well understood (see §6).

In this symmetry approach to representing tilings, the key to satisfying the requirements above is to balance the amount of geometric and topological data with the complexity of the replication process, to ensure that a given region of the plane is covered without repetitions. Moreover, finding the symmetries of a tiling algorithmically requires a computational representation of the tiling.

Our approach avoids this dilemma and strikes a balance by using the simplest topological elements, the vertices of the tiling, and the simplest symmetries, the translations of the tiling. Then the only geometric data required are coordinates for vertices and translation vectors. As we explain in detail in §3, these coordinates come from representing the vertices as complex numbers, more precisely as points in $\mathbf{Z}[\omega]$. Our representation satisfies all the requirements above (see §7). Moreover, our representation allows finding all symmetries of a tiling efficiently (see §6) using a simple data structure: a hash table that we call a *cloud* of vertices (see §4).

Having proven its worth within our previous works [10, 11] on rendering and acquisition of tilings, our representation is both convenient and general enough to show great potential for many more operations and analyses of tilings. Accordingly, the remainder of this paper offers for the first time a comprehensive account of the representation, presenting its properties, common topological and geometric operations, and sample applications.

3. A representation for periodic tilings by regular polygons

We shall now describe in detail our integer representation for periodic tilings of the plane by regular polygons, which we shall call simply *tilings* from now on. We start with a few preliminary notions about the discrete nature of the problem.

Rigidity. Only very few types of regular polygons can tile the plane. A local constraint is how the angles of the faces fit around a vertex of the tiling. We get $\sum_{k\geq 3} n_k \alpha_k = 360^\circ$, when there are n_k faces of k sides around the vertex, each contributing an internal angle $\alpha_k = 180^\circ - \frac{360^\circ}{k}$. Therefore, $\sum_{k\geq 3} n_k \left(1-\frac{2}{k}\right) = 2$. The 17 integer solutions of this equation give 21 possible face configurations around a vertex, up to cyclic permutations and reflections [5]. Due to global constraints, only the 15 configurations shown in Fig. 2 can actually fit together without gaps or overlaps to tile the plane in very specific, non-trivial ways. Thus, tiling plane by regular polygons is a discrete problem with, in a sense, rigid solutions. In particular, all tilings are made of triangles, squares, hexagons, octagons, and dodecagons. There is exactly one tiling that contains octagons: the truncated square tiling composed solely of vertices of type J. We shall disregard this singular tiling. This allows us to align the edges of all tilings with the 12th roots of unity and thus represent vertices in $\mathbf{Z}[\omega]$.

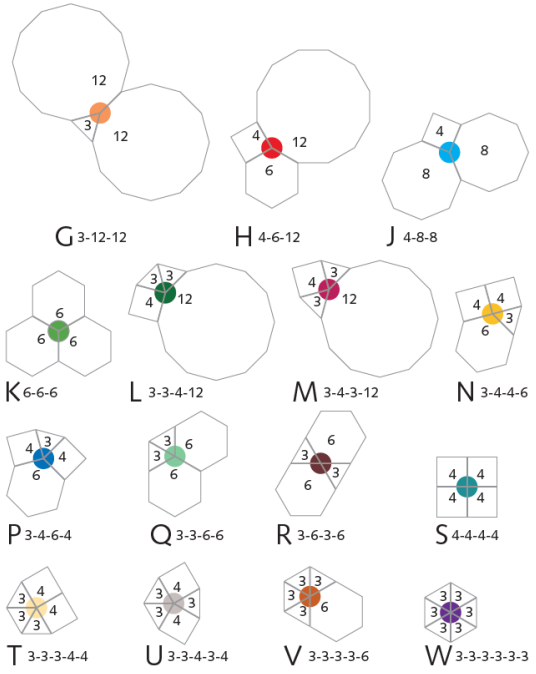


Fig. 2. The 15 vertex types that appear in tilings by regular polygons [12].

Philosophy. We consider the vertices as the central elements of the tiling. Our approach is to represent explicitly a small finite subset of *seed vertices* from which we systematically generate all vertices of the tiling by translations only, thus avoiding the complexity of crystallographic fundamental regions. Our approach is reminiscent of the interlocking regular systems of points discussed in the classic book by Hilbert and Cohn-Vossen [13, chapter 2]. The edges and the faces of the tiling are easily reconstructed from the vertices because the tiling is composed of regular polygons (Fig. 3). We shall describe algorithms for reconstructing the whole tiling from its vertices in §4.

Translation grid. A periodic tiling is invariant under two independent translations. The corresponding *translation vectors* t_1 and t_2 induce a subdivision of the plane into an infinite *grid*

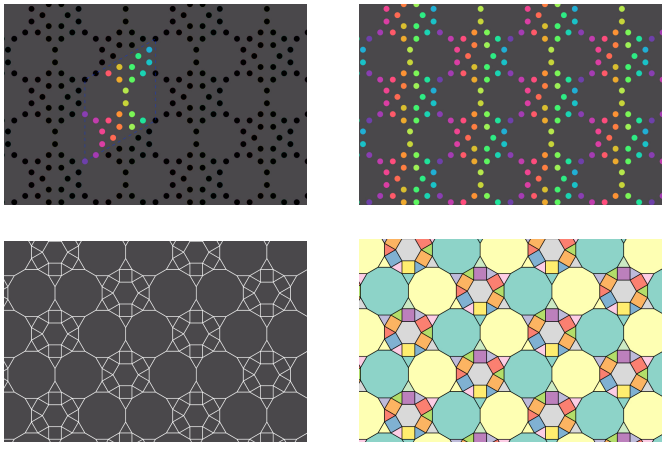


Fig. 3. From left to right, top to bottom: Vertices and seeds, interlocking systems of points, edges, and complete tiling.

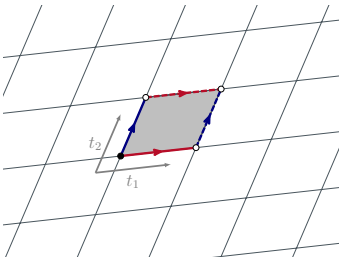


Fig. 4. Grid of translation cells, translation vectors at the origin (in black), and basic translation cell (in gray).

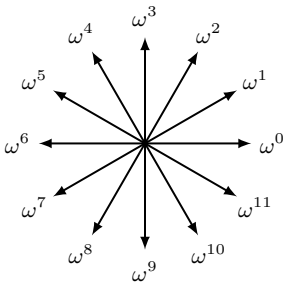


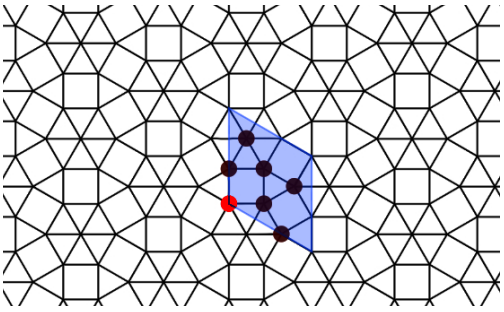
Fig. 5. The twelve basic directions in the complex plane.

of parallelograms called *translation cells* (Fig. 4). Invariance means that everything in one cell of the grid is repeated exactly in all cells. The *basic translation cell* T_0 is the one containing the origin: $T_0 = \{\lambda_1 t_1 + \lambda_2 t_2 : \lambda_1, \lambda_2 \in [0,1)\}$. Each translation cell is a translation of the basic cell by *integer multiples* of t_1 and t_2 and so can be written as $T_0 + n_1 t_1 + n_2 t_2$ with $n_1, n_2 \in \mathbf{Z}$. The pair (n_1, n_2) is the *id* of the cell. By definition, and as illustrated in Fig. 4, the basic translation cell is *half-open*, that is, open at the sides not containing the origin. Different translation cells are thus disjoint. Therefore, the translation cells partition the plane: each point in the plane, and so each vertex of the tiling, belongs to exactly one translation cell. However, the edges and the faces of the tiling are not always completely inside a cell and often straddle cell boundaries.

Translational equivalence. Two points p and q in the plane are *equivalent* under the translations of the tiling if they coincide when their translation cells are moved one on top of the other by a translation. This happens exactly when $p = q + n_1 t_1 + n_2 t_2$ for some $n_1, n_2 \in \mathbf{Z}$. In algebraic terms, p and q are equivalent under translations exactly when they are in the same coset of the *translation lattice* $\mathbf{Z}t_1 + \mathbf{Z}t_2$, a discrete additive subgroup of $\mathbf{R}^2$ [14].

Seeds. Each translation cell contains exactly one representative of each equivalence class of points. In particular, each translation cell contains exactly one representative of each equivalence class of vertices, and so contains the same number of vertices. The vertices in the basic cell are called *seeds*. Thus, every vertex of the tiling is equivalent to exactly one seed. The seeds and their translations decompose the set of vertices into interlocking regular systems of points sharing the same translation group in the sense of Hilbert and Cohn-Vossen [13] (Fig. 3). The seeds and the translation grid are the basis of our representation: the seeds describe *what* needs to be replicated and the translation vectors describe *how* they are replicated.

Flat torus. Modulo translational equivalence, the plane becomes a *flat torus*: opposite sides of the basic cell are identified and distances are measured on the plane. In other words, a flat torus has the topology of a torus but the metric of the plane. The difficulty in tiling the plane periodically with regular polygons is how to place seeds in a parallelogram and join them in the toroidal sense such that all polygon edges have the same length in



a_0	a_1	a_2	a_3
−1	0	2	1
2	1	−1	−1
0	0	0	0
0	0	0	1
0	0	1	1
1	0	0	0
1	0	0	1
1	1	0	0
2	0	−1	0

Fig. 6. A tiling and its basic elements. The translation vectors define the basic cell (in blue), which contains the seeds (in black). The external representation data is a $(2+7)\times 4$ integer matrix. The first two rows represent the translation vectors. The remaining rows represent the seven seeds, one of which is the origin (in red).

the plane. As we shall see in §5, the only possible parallelograms for this task are those whose area is an integer linear combination of the areas of an equilateral triangle and a square. This is another manifestation of rigidity.

Abstract representation. Abstractly, our representation of a tiling has two components: a pair of translation vectors that define a translation grid for the tiling and a set of seeds in the basic translation cell that represent every vertex of the tiling by translations (Fig. 6); the polygons remain implicit. To make the representation concrete we need to give coordinates to these elements. In principle, we could simply use Cartesian coordinates. However, the Cartesian coordinates of the vertices in a tiling are rarely exact rational numbers. (In fact, the only exception is the square tiling composed solely of vertices of type S.) We seek a numerically exact representation. We shall see presently how to give *integer* coordinates to all elements. This leads us to discuss the edges of the tiling.

Normalization. As in most of Euclidean geometry, we consider tilings up to uniform scalings and rigid transformations (translations, rotations, reflections). To choose coordinates for the translation vectors and the seeds, we must fix a scale, an origin, and reference directions for orientation. We reduce the number of choices by partially normalizing the tiling, as follows.

We normalize the scale of the tiling by taking edges of unit length, since all edges have the same length, being sides of adjacent regular polygons. We partially normalize the position of the tiling by choosing one of the vertices as the origin. (We shall see that this is a key choice.) We partially normalize the orientation of the tiling by choosing a canonical set of directions for the edges. Since we explicitly disregard the one tiling that contains octagons, all tilings are made of triangles, squares, hexagons, and dodecagons, and so the angles between edges are all multiples of 30°. Thus, we partially normalize the orientation of the tiling by aligning the edges with the 12th roots of unity in the complex plane. (This is another key choice, which is natural in this context. Others independently reached the same conclusion [15, 8].) These twelve *basic directions* are the powers of $\omega = \exp(\frac{2\pi i}{12}) = \frac{\sqrt{3}+i}{2}$, the principal 12th root of unity (Fig. 5).

This normalization of the geometry of the tiling is partial because there is room for choosing the vertex to be the origin (not all vertices are equivalent under translations) and for choosing an edge and fixing a basic direction for it (not all edges are equivalent under rotations). Once we fix a direction for a single edge, the orientation of the whole tiling is rigidly fixed.

Lattice coordinates. We shall now make our representation concrete by giving integer coordinates to all vertices of the tiling. This automatically gives integer coordinates to the translation vectors since they take the origin to another vertex of the tiling.

The key idea is that every vertex can be reached by following a path from the origin along the edges of the tiling. (Hence the importance of placing a vertex at the origin.) Since edges have unit length and are aligned with the twelve basic directions, which are powers of ω, these paths are given by integer polynomial expressions in ω of degree less than 12 (Fig. 7). Thus, the vertices form a subset of the *cyclotomic integers* $\mathbf{Z}[\omega]$ (the set of integer polynomial expressions in ω) and can be given 12-dimensional integer coordinates. However, these coordinates are *not* unique. For instance, ω^6 and -1 are two different expressions for the same point in the complex plane.

The matter is cleared by considering the *minimal* polynomial equation that ω satisfies: $\omega^4 - \omega^2 + 1 = 0$ (it corresponds to the 12th cyclotomic polynomial). Every polynomial expression in ω is therefore equal to a *unique* cubic expression: its remainder modulo the minimal polynomial $\omega^4 - \omega^2 + 1$. This is the key observation here: $\mathbf{Z}[\omega]$ is actually the set of all integer polynomial expressions in ω of degree at most 3, and different expressions represent different points. In other words, $\{1, \omega, \omega^2, \omega^3\}$ is an additive basis for $\mathbf{Z}[\omega]$ over $\mathbf{Z}$, in the sense that $\mathbf{Z}[\omega] = \mathbf{Z}1 + \mathbf{Z}\omega + \mathbf{Z}\omega^2 + \mathbf{Z}\omega^3$. In particular, each vertex of the tiling is represented uniquely by four integers $[a_0, a_1, a_2, a_3]$, corresponding to the point $a_0 1 + a_1\omega + a_2\omega^2 + a_3\omega^3 \in \mathbf{Z}[\omega]$. We call these integers the *lattice coordinates* of the vertex. (As an additive group, $\mathbf{Z}[\omega] \cong \mathbf{Z}^4$, a *lattice* in $\mathbf{R}^4$.)

The lattice coordinates of a vertex are found by following any path connecting the origin to the vertex, adding each edge in the path to obtain a polynomial expression in ω (Fig. 7). This expression is then simplified to a cubic expression either by taking its remainder modulo the minimal polynomial or equivalently

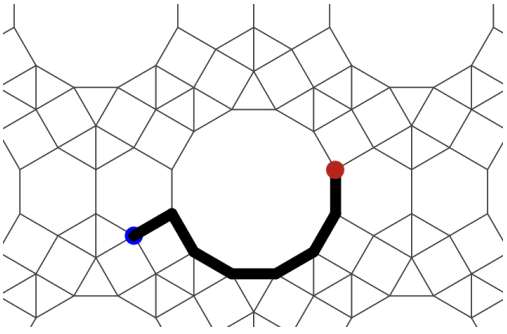


Fig. 7. Paths joining vertices are given by integer polynomial expressions in ω. This path from the blue origin to the red vertex is given by $\omega^1 + \omega^{10} + \omega^{11} + \omega^0 + \omega^1 + \omega^2 + \omega^3$, which reduces to $2 + 3\omega$, since $\omega^4 - \omega^2 + 1 = 0$. Thus, the red vertex has lattice coordinates $[2,3,0,0]$ in $\mathbf{Z}[\omega]$.

by using the identities below, since only the basic directions $1, \omega, \omega^2, \ldots, \omega^{11}$ appear in paths:

$$\begin{aligned}
\omega^4 &= -1+\omega^2 &&= [-1,0,1,0] & \omega^8 &= -\omega^2 = [0,0,-1,0]\\
\omega^5 &= -\omega+\omega^3 &&= [0,-1,0,1] & \omega^9 &= -\omega^3 = [0,0,0,-1]\\
\omega^6 &= -1 &&= [-1,0,0,0] & \omega^{10} &= -\omega^4 = [1,0,-1,0]\\
\omega^7 &= -\omega &&= [0,-1,0,0] & \omega^{11} &= -\omega^5 = [0,1,0,-1]
\end{aligned}$$

The conversion from a polynomial expression $b_0+b_1\omega+b_2\omega^2+\cdots+b_{11}\omega^{11}$ to lattice coordinates $[a_0,a_1,a_2,a_3]$ is thus a simple matrix multiplication. Although many different paths join the origin to the vertex, the corresponding expressions simplify to the same cubic expression, which depends only on the vertex.

Concrete representation. In our concrete representation, each data item in the abstract representation (translation vector or seed) is given by its lattice coordinates as a point in $\mathbf{Z}[\omega]$, and so is represented uniquely by four integers, as above. Thus, each tiling is represented by a $(2+n)\times 4$ integer matrix, where n is the number of seeds in the tiling. The first two rows of the matrix represent the translation vectors and the remaining n rows represent the seeds (Fig. 6). Exactly one of these rows contains only zeros, because the origin is a seed. This integer matrix is our concrete representation for the tiling. We also call it the *external representation*, because it can be saved to files, published, and shared. We shall see presently an internal representation that complements the external representation.

In the example in Fig. 6, the translation vectors are

$$\begin{aligned}
t_1 &= \omega^3+\omega^2+\omega^4 = [-1,0,2,1]\\
t_2 &= 1+\omega^{10}+\omega^{11} = [2,1,-1,-1]
\end{aligned}$$

and the seeds are

$$\begin{aligned}
s_1 &= 0 &&= [0,0,0,0] & s_5 &= 1+\omega^3 &&= [1,0,0,1]\\
s_2 &= \omega^3 &&= [0,0,0,1] & s_6 &= 1+\omega &&= [1,1,0,0]\\
s_3 &= \omega^3+\omega^2 &&= [0,0,1,1] & s_7 &= 1+\omega^{10} &&= [2,0,-1,0]\\
s_4 &= 1 &&= [1,0,0,0]
\end{aligned}$$

As we have explained above, these lattice coordinates are found by adding the lattice coordinates of the basic directions followed in a path from the origin to each vertex.

This completes the description of our representation of tilings. We have acquired [11] and published [9] a large collection of tilings in this representation. We shall now explain how to reconstruct the tiling from a representation.

4. Topological primitives

The standard tool to represent a general subdivision of the plane is a topological data structure [16, chapter 2] that explicitly represents all topological elements (vertices, edges, faces) of the subdivision and a subset of their adjacency relationships. A topological data structure complements the geometric information given by the position of the vertices and the shape of the edges. Most importantly, it provides convenient and efficient traversal of the elements of the subdivision, for primitive tasks such as finding the edges and the faces around a vertex, the vertices of a face, and the faces adjacent to a face. These tasks are vital for rendering, processing, and analyzing the subdivision.

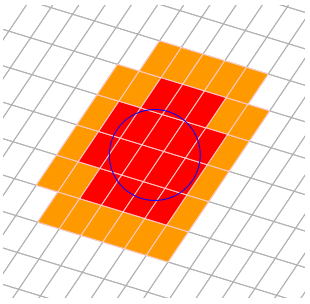


Fig. 8. A circular window (in blue). The inner cells (in red) intersect the window. The outer cells (in orange) form a ring around the inner cells.

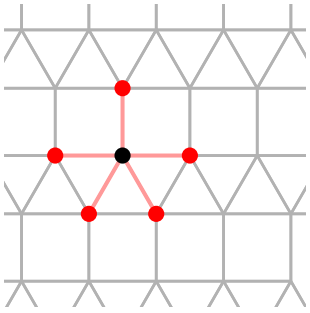


Fig. 9. The red vertices form the star of the black vertex. The edges are $\omega^0, \omega^3, \omega^6, \omega^8, \omega^{10}$ and so the star is represented by the list $\langle 0,3,6,8,10\rangle$.

Although tilings are subdivisions of the plane, topological data structures are too general for representing periodic tilings of the plane by regular polygons, whose geometry and topology are quite rigid. In sharp contrast, our representation does not represent all the elements of a tiling, only a key finite subset of its vertices: the seeds. Our representation does not represent any adjacency relationships at all, but, as we shall see below, it allows them to be easily and efficiently computed on demand.

The window. We shall describe how to use our representation to reconstruct the tiling inside a given *window*, a bounded region of interest in the plane (typically, but not necessarily, an axis-aligned rectangle). Reconstructing the tiling means finding all its elements (vertices, edges, faces) that can be seen in the window.

We start by finding all translation cells that intersect the window, which we call the *inner cells* (Fig. 8). This is equivalent to "rasterizing" the window on the translation grid. The solution depends on the geometry of the window and is outside the scope of this paper. (It is relatively easy for a rectangular window.) We assume that the problem has been solved and the set of inner cells with their ids is available. Since the edges and faces of the tiling often straddle cell boundaries, to reconstruct the edges and faces in the window we also need to find all translation cells adjacent to the inner cells. These *outer cells* form a ring around the inner cells and provide complete toroidal neighborhoods for the inner cells (Fig. 8). We assume that the set of outer cells with their ids is available when needed.

Reconstructing vertices. To reconstruct the vertices of the tiling in the window, we simply translate the seeds along the translation vectors from the basic cell to each inner cell. This gives lattice coordinates to all *inner vertices*. More precisely, the vertices in an inner cell $T = T_0+n_1t_1+n_2t_2$ are $s+n_1t_1+n_2t_2$, where s runs through all seeds. Thus, the lattice coordinates of the vertices in a cell are obtained from the lattice coordinates of the seeds and the lattice coordinates of the translation vectors using the id of the cell. The same procedure finds the *outer vertices*, that is, those in the outer cells.

The vertex cloud. We have found it convenient to store inner and outer vertices in a hash table indexed by their lattice coordinates, which we call the *cloud*. The cloud maps lattice coordinates of vertices to task-specific information about the vertices. For topology reconstruction and symmetry detection, just the presence of the vertices in the cloud suffices. The keys for hashing are typically a string like `"1,2,3,4"` or a 64-bit integer like `0x0001000200030004`. The details of hashing are unimportant; the important feature of the cloud is that querying a vertex based on its lattice coordinates takes amortized constant time. This is central for the simplicity and the efficiency of our algorithms, because we can avoid nearest-neighbors searches and geometric computations with numerical tolerances. The cloud is an explicit *internal representation* of the tiling that complements the compact implicit external representation and is consistent with our focus on the vertices as the central elements of the tiling.

Edges around a vertex. The *star* of a vertex v is the list of vertices w such that vw is an edge of the tiling, ordered circularly around v (Fig. 9). The stars provide convenient systematic traversal of the edges around a vertex, a basic topological primitive.

Let v be an inner vertex. Since each edge vw in the star of v is aligned with a basic direction ω^k, in the sense that $w = v + \omega^k$, we represent the star of v by the ordered list of the exponents that appear in these edges (Fig. 9). We find this ordered list in constant time by testing whether $v + \omega^k$ is in the cloud for $k = 0, 1, \ldots, 11$, as in Algorithm 1. The lattice coordinates of $v + \omega^k$, needed for querying the cloud, are easily computed from the lattice coordinates of v and the lattice coordinates of ω^k given in §3.

Algorithm 1

```
procedure star(v)
  s ← []
  n ← 0
  for k = 0 to 11 do
    if v + ω^k in cloud then
      n ← n + 1
      s[n] ← k
    end
  end
  return s, n
end
```

Reconstructing edges. We reconstruct the edges of the tiling in the window by finding the stars of the inner vertices. The stars actually give oriented edges. Therefore, reconstructing edges with stars finds the edge between two inner vertices v and w twice: once as vw and once as wv. Tasks that need to process edges exactly once, such as rendering, should process an edge vw only when w is an outer vertex or $v < w$ in the lexicographical order of their lattice coordinates, as in Algorithm 2. Running Algorithm 2 for all inner vertices processes exactly once all edges of the tiling that are visible in the window.

Faces around a vertex. A *corner* of a face is a vertex v and a pair of circularly adjacent entries k and k' in the star of v. If

Algorithm 2

```
procedure edges(v)
  for k = 0 to 11 do
    w ← v + ω^k
    if w in cloud and (w is outer or v < w) then
      process(v, w)
    end
  end
end
```

$k' < k$, take $k' + 12$ instead. (For the vertex in Fig. 9, two corners are $v, 0, 3$ and $v, 10, 12$.) The internal angle of the face at v is $(k' - k)30°$ and so determines the number m of sides of the face at that corner:

$$(k' - k)30° = \left(1 - \frac{2}{m}\right) 180° \implies m = \frac{12}{6 - (k' - k)}$$

The table below summarizes the results of this computation:

internal angle	60°	90°	120°	150°
$k' - k$	2	3	4	5
m	3	4	6	12

The list of m for all pairs k, k' gives the vertex type as in Fig. 2. For the vertex in Fig. 9, the list of $k' - k$ is $\langle 3, 3, 2, 2, 2 \rangle$ and so the list of m is $\langle 4, 4, 3, 3, 3 \rangle$, which gives vertex type T:3-3-3-4-4 after a cyclic reordering.

When traversing the edges of a face with m sides in counterclockwise order, the directions of the edges change by the external angle $\frac{360°}{m}$. Therefore, the vertices of the face having a corner at v, k, k' are obtained by starting at v, moving in direction k, adding $\frac{12}{m}$ to the direction mod 12, and repeating, as in Algorithm 3. Running Algorithm 3 for all pairs of circularly adjacent directions k, k' in the star of v finds all faces around v.

Algorithm 3

```
procedure face(v, k, k')
  m ← 12/(6 − (k' − k))
  f ← []
  for j = 1 to m do
    f[j] ← v
    v ← v + ω^k
    k ← (k + 12/m) mod 12
  end
  return f
end
```

Reconstructing faces. To avoid reconstructing faces multiple times (once per adjacent vertex), we do it only when the vertex is the lowest leftmost vertex of a face. In this case we say that the vertex is the *anchor* of that face (Fig. 10). Anchors correspond to corners having directions $k, k' \in \{10, 11, 0, 1, 2, 3\}$. There are at most two such corners at a vertex because the internal angles of the faces are at least 60°. Algorithm 4 finds the faces anchored at a given vertex, if any. (Here *star* is an extension of Algorithm 1 that accepts a range of directions to test.) Running Algorithm 4 for all inner vertices processes exactly once all faces of the tiling that are visible in the window.

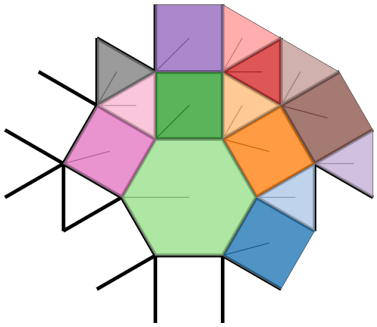


Fig. 10. Faces and their anchors. The anchor of a face is the lowest leftmost vertex of a face. A thin line joins the center of a face to its anchor.

Algorithm 4

procedure *faces*(v)
  $s,n \leftarrow star(v,10,15)$
  for $j = 1$ **to** $n-1$ **do**
    $k \leftarrow s[j]$
    $k' \leftarrow s[j+1]$
    $process(face(v,k,k'))$
  end
end

Faces adjacent to an edge. The faces adjacent to an edge correspond to a pair of adjacent corners at one of its vertices. More precisely, if the edge is vw and v,k,k' and v,k',k'' are adjacent corners with $w = v+\omega^{k'}$, then the faces adjacent to the edge vw where are given by $face(v,k,k')$ and $face(v,k',k'')$.

Finding duals. The *dual* of a tiling is the "tiling" whose vertices are the centers of the faces of the original *primal* tiling and whose edges join two centers when the corresponding primal faces share an edge. Although duals rarely have regular faces (hence the quotes above), they too are very interesting periodic subdivisions of the plane with the same strong aesthetic appeal and artistic potential (Fig. 11).

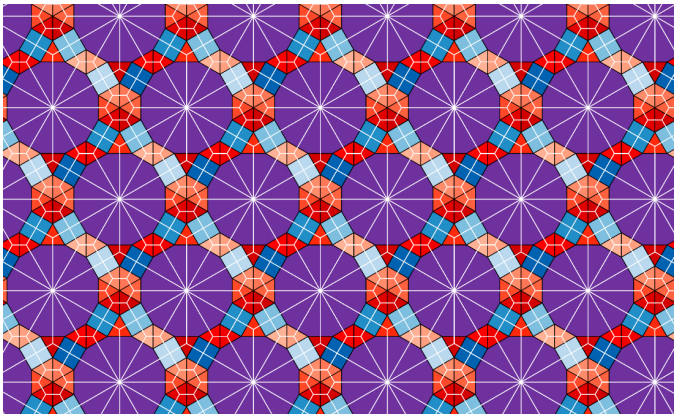


Fig. 11. A tiling (in color) and its dual (in white).

A dual edge connects the centers of the faces of two adjacent corners at the same primal vertex. A dual face has as vertices the centers of the faces around a primal vertex. Thus, finding the dual tiling reduces to topological primitives over the primal tiling, combined with the geometry of face centers. These centers are the dual vertices and the central elements of the dual tiling.

The center of a face can be computed as the barycenter of its vertices or as the midpoint of a pair of opposite vertices in the face, when the face is not a triangle. When the vertices of the face are not readily available, the center c can also be computed directly from a face corner v,k,k' by solving

$$(v-c)\omega^{12/m} = (v+\omega^k) - c$$

since $\omega^{12/m}$ gives the central angle of the face and $v+\omega^k$ is the next vertex after v; m is computed from $k'-k$ as before. In the simplest case, $v=0$ and $k=0$ (the vertex is at the origin and the first edge is horizontal) and we get

$$c_m = \frac{1}{1-\omega^{12/m}}$$

This point can be written explicitly in (fractional) lattice coordinates as follows:

m	3	4	6	12
c_m	$\frac{1}{3}[1,0,1,0]$	$\frac{1}{2}[1,0,0,1]$	$[0,0,1,0]$	$[0,0,1,1]$

In the general case, the center of a face having a corner at v,k,k' is $v+c_m\omega^k$, corresponding to a rotation in the direction k followed by a translation to v. The multiplication $c_m\omega^k$ is easily done using the lattice coordinates of the powers of ω given in §3. It is convenient to write the fractional lattice coordinates of face centers uniformly with denominator 6, that is, in $\frac{1}{6}\mathbf{Z}[\omega]$.

Local reconstruction. Everything that happens in the basic cell is repeated exactly in all cells. Therefore, it suffices to reconstruct the tiling around the seeds, that is, by taking the basic cell as the window. The set of all edges emanating from the seeds is the *skeleton* of the tiling. The set of all faces anchored at the seeds is the *patch* of the tiling (Fig. 12). These fundamental pieces are *translational units*: they can be repeated by translations to reconstruct the whole tiling in large windows.

This alternative, local reconstruction does not require storing all vertices in the cloud, just the ones in the basic cell and in its outer cells, nine cells in all. On the other hand, it contains

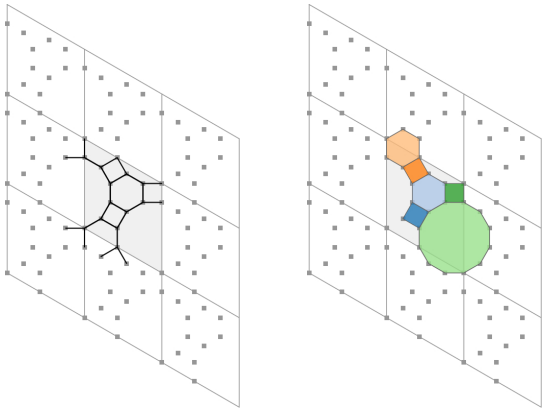


Fig. 12. Skeleton (left) and patch (right) of a tiling. The basic cell is the central parallelogram in gray (left). It is surrounded by eight outer cells.

duplicate vertices and edges. This duplication may be relevant for rendering, modeling, and further processing. The tradeoffs between local and global reconstruction are analogous to those between retained and immediate mode in graphics. We find global reconstruction easier to understand and use.

5. Geometric operations

The topological primitives described in §4 are exact, integer algorithms: they use lattice coordinates directly and never need Cartesian coordinates. Rendering a tiling requires the Cartesian coordinates of its vertices, but only at the last moment, before issuing graphics primitives in floating-point coordinates. The traversal of the topological elements to assemble graphics primitives is entirely based on integer coordinates, which attests to the appeal of our numerically robust representation.

Cartesian coordinates. The Cartesian coordinates of the additive basis $\{1, \omega, \omega^2, \omega^3\}$ of $\mathbf{Z}[\omega]$ are given by

$$\begin{pmatrix} 1 \\ \omega \\ \omega^2 \\ \omega^3 \end{pmatrix} = \begin{pmatrix} \cos(0°) & \sin(0°) \\ \cos(30°) & \sin(30°) \\ \cos(60°) & \sin(60°) \\ \cos(90°) & \sin(90°) \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

or, more concisely, $W = BE$. Here, e_1, e_2 is the canonical basis of $\mathbf{R}^2 \cong \mathbf{C}$. Therefore, the Cartesian coordinates (x, y) of a vertex with lattice coordinates $[a_0, a_1, a_2, a_3]$ are

$$\begin{pmatrix} x & y \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \end{pmatrix} B = \frac{1}{2} \begin{pmatrix} 2a_0 + a_2 + a_1\sqrt{3} \\ a_1 + 2a_3 + a_2\sqrt{3} \end{pmatrix}^{\mathsf{T}}$$

Thus, the Cartesian coordinates of the vertices can be determined from their lattice coordinates with as much precision as needed, since the fractional part of the Cartesian coordinates comes from a single multiplication by $\sqrt{3}$, eliminating the scope for numerical cancellation.

Grid coordinates. An important geometric task when processing periodic tilings is locating a point p in the plane with respect to the translation grid, that is, finding the translation cell that contains p. This problem is easily solved by expressing p with respect to the basis formed by the translation vectors t_1, t_2. Indeed, if $p = \lambda_1 t_1 + \lambda_2 t_2$, then the translation cell that contains p is $T_0 + n_1 t_1 + n_2 t_2$, where T_0 is the basic cell and $n_i = \lfloor \lambda_i \rfloor$. We call λ_1, λ_2 the *grid coordinates* of p.

The natural way to find the grid coordinates of p from its Cartesian coordinates is to use the Cartesian coordinates of t_1, t_2, which are found from their lattice coordinates as above:

$$\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix} W = AW = ABE$$

The Cartesian coordinates of t_1, t_2 are the rows of the 2×2 matrix AB. This matrix is invertible because t_1, t_2 are linearly independent. Thus, the grid coordinates of $p = (x, y)$ are

$$\begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} (AB)^{-1}$$

When p has lattice coordinates $[a_0, a_1, a_2, a_3]$ and grid coordinates λ_1, λ_2, then

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 \end{pmatrix} W = p = \begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} AW$$

and so

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 \end{pmatrix} = \begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} A$$

Therefore,

$$\begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \end{pmatrix} A^+$$

where $A^+ = A^{\mathsf{T}}(AA^{\mathsf{T}})^{-1}$ is the pseudoinverse of A, which satisfies $AA^+ = I_2$. Note that AA^{T} is invertible because A has row rank 2 since the translation vectors are linearly independent. It is easy to compute $(AA^{\mathsf{T}})^{-1}$ explicitly because AA^{T} is a 2×2 matrix with integer entries. In particular, $(AA^{\mathsf{T}})^{-1}$ has rational entries with $\det(AA^{\mathsf{T}})$ in the denominator. Thus, every point in $\mathbf{Z}[\omega]$ has *rational* grid coordinates.

Lattice reduction. Grid coordinates are useful for *lattice reduction*: given a vertex v of the tiling, find the seed s that is equivalent to v under translations. By definiton, $s = v \bmod L$, where $L = \mathbf{Z}t_1 + \mathbf{Z}t_2$ is the translation lattice. Given the grid coordinates λ_1, λ_2 of v computed as above, we have that $s = v - n_1 t_1 - n_2 t_2$ is a vertex in the basic cell and so is a seed. (Here, as before, $n_i = \lfloor \lambda_i \rfloor$.) Therefore, $s = v \bmod L$. As we shall see in §7, lattice reduction is instrumental in testing the validity of a representation and in deciding the equivalence of two representations.

Area. Recall that the Cartesian coordinates of the translation vectors are the rows of the 2×2 matrix AB. The absolute value of the determinant of this matrix gives the area of the basic cell, which we call the *area* of the tiling. The area is given explicitly in terms of the lattice coordinates of t_1, t_2 by $\frac{1}{2}\left|a + b\sqrt{3}\right|$, where a and b are integers:

$$a = 2a_{11}a_{24} + a_{11}a_{22} + a_{12}a_{23} + a_{13}a_{24} - 2a_{14}a_{21} - a_{12}a_{21} - a_{13}a_{22} - a_{14}a_{23}$$
$$b = a_{11}a_{23} + a_{12}a_{24} - a_{13}a_{21} - a_{14}a_{22}$$

These two integers a and b are unique up to sign because $\sqrt{3}$ is irrational. In particular, the area is zero iff $a = b = 0$.

The area does not depend on the choice of translation vectors. Indeed, two pairs of translation vectors T and T' determine the same translation lattice iff there is a 2×2 unimodular matrix U such that $T' = UT$. (A *unimodular* matrix is an integer matrix with determinant ± 1, or equivalently, an integer matrix with integer inverse.) Write $T = AW$ and $T' = A'W$ as above. Then, $T' = UT$ iff $A' = UA$, because the basic directions in W are linearly independent over $\mathbf{Z}$. Therefore, $\det(A'B) = \det(UAB) = \det(U)\det(AB) = \pm\det(AB)$, and so the area is the same.

The area of a minimal translation cell is a simple measure of the complexity of a tiling. Indeed, the area of the basic cell constrains the number of seeds since they must be one unit apart in the flat torus. For tilings consisting of just triangles and squares, which have the highest density of vertices, the formula for Euler's characteristic of the torus implies area $\leq n \leq 2$ area, where n is the number of seeds.

6. Symmetry

We shall now explain how to find the symmetry group of a tiling by looking at its vertices, the central elements of the tiling.

Symmetries. A *symmetry* of a pattern in the plane is a rigid transformation of the plane (translation, rotation, reflection) that leaves the pattern invariant. The symmetries of a pattern form a group, its *symmetry group*. The symmetries of periodic patterns in the plane are well understood and have been completely classified: there are exactly 17 symmetry groups, known as the *wallpaper groups* [2]. Once one knows the symmetries of a pattern, its wallpaper group is found by following the flowchart in Fig. 13, which was proposed by Washburn and Crowe [17] and is widely used in the literature [18].

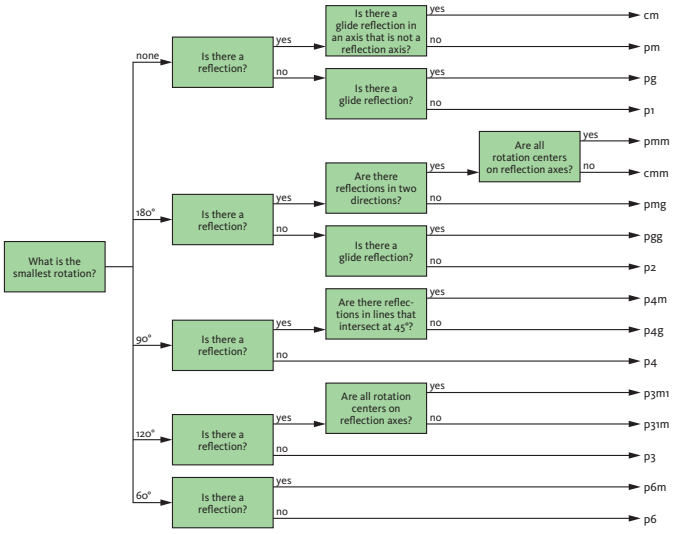


Fig. 13. Washburn–Crowe [17] flowchart for classifying a symmetry group into a wallpaper group using the crystallographic notation (after [18]).

Methodology. To find the symmetry group of a tiling, we need to find the rigid transformations of the plane that leave it invariant. A translation must send every vertex to another vertex. A rotation must be centered at a vertex, at the centroid of a face, or at the midpoint of an edge. A reflection must be across an edge, across the bisector of an edge, or across the bisector of an internal angle of a face. Since the tiling is invariant under translations, it suffices to consider the *local* symmetries of the tiling, that is, those originating in the patch. Thus, there are only a few candidate symmetry transformations for a given tiling.

Let σ be a candidate symmetry for the tiling. Since the tiling can be reconstructed from its vertices, σ leaves the tiling invariant iff σ maps vertices to vertices. In fact, because σ is *rigid*, it leaves the tiling invariant iff it maps the *key vertices* (the seeds and t_1, t_2) to vertices of the tiling. We test this using the cloud; we just need to ensure that the cloud contains the images of the key vertices under all local symmetries of the tiling. It turns out that it suffices to add to the cloud all vertices inside the disk centered at the centroid of the basic cell with radius $4D$, where D is the diameter of the basic cell.

Translations. The subgroup generated by the translation vectors of a tiling may not be the smallest translation subgroup because the representation of the tiling may not be a minimal one (see §7). To find the smallest translation subgroup, we test for each seed s_0 (except the origin) whether the translation by s_0 leaves the tiling invariant, as follows: for each seed s we check whether $s + s_0$ is in the cloud. If they all are, then s_0 defines a translation for the tiling. Thus, since there are n seeds, in time $O(n^2)$ we find all possible candidates for minimal translation vectors for the tiling. Among those, we take as t_1 the smallest translation vector and as t_2 the smallest translation vector that is linearly independent with t_1. However, by construction, these vectors are constrained to lie inside the original basic cell and so may not be the smallest possible ones, even though they define the smallest translation subgroup. The best translation vectors are found using Algorithm 5, an algorithm by Lagrange and Gauss that generalizes the Euclidean algorithm for finding the greatest common divisor of two integers [14]. (Here $\lceil \mu \rfloor = \lceil \mu - 0.5 \rceil$ is the integer nearest to μ.) Although it is best understood in the plane, this algorithm works for every pair of linearly independent vectors in any dimension and so can be applied in $\mathbf{R}^4$ directly to the lattice coordinates of t_1 and t_2.

Algorithm 5

procedure *minbasis*(v_1, v_2)
  repeat
    if $\|v_1\| > \|v_2\|$ **then**
      swap v_1 and v_2
    end
    $\mu \leftarrow (v_1 \cdot v_2)/(v_1 \cdot v_1)$
    $v_2 \leftarrow v_2 - \lceil \mu \rfloor v_1$
  until $\|v_1\| \leq \|v_2\|$
  return v_1, v_2
end

Rotations. We test all rotations centered at the seeds, at the centroids of the faces of the patch, and the midpoints of the edges of the patch. The possible rotation angles are 60°, 90°, 120°, 180°. Thus the rotations correspond to multiplication by ω^2, ω^3, ω^4, ω^6. It turns out that all these rotations can be expressed as integer matrices acting on lattice coordinates, even though centroids and midpoints may have fractional lattice coordinates (see §4).

A rotation around a seed that leaves the tiling invariant must leave the star of the seed invariant. While this severely restricts the possible rotations around a seed, it is simplest to test all four possible rotations around a seed. A rotation around the centroid of a face that leaves the tiling invariant must be of an angle that is a multiple of the central angle of the face. A rotation around the midpoint of an edge must be of 180°. In all cases, we test the rotations in increasing order of angle and stop as soon as we find a rotation that works, if any.

Reflections. A reflection that leaves the tiling invariant must be across an edge, across the bisector of an edge, or across the bisector of an internal angle of a face. All these reflections can be expressed as integer matrices acting on lattice coordinates.

Glide reflections. A glide reflection is a reflection across a line followed by a translation along that line. They are more complex than the other symmetries and harder to detect visually.

Let $\gamma = \tau\rho$ be a minimal glide reflection that leaves the tiling invariant. Here τ is a translation and ρ is a reflection across the axis of τ. Then, $\gamma^2 = \tau^2$ since $\rho\tau = \tau\rho$ and $\rho^2 = id$. Therefore, τ^2 must be a minimal translation that leaves the tiling invariant and so τ must be of the form $\tau = \frac{1}{2}(n_1 t_1 + n_2 t_2)$, where $n_1, n_2 \in \{-2,-1,0,1,2\}$; this is very useful, since the mirror is the line containing τ. Since the transformed points must be in $\mathbf{Z}[\omega]$, glide reflection axes must pass through seeds or midpoints of edges of the patch. This gives the possible candidates for glide reflections that leave the tiling invariant.

Transitive equivalence. Two points in the plane are *transitively equivalent* with respect to a given symmetry group when there is a symmetry in that group that takes one point to the other. Transitively equivalent vertices in a tiling must have the same type according to Fig. 2. A tiling is *k-uniform* when it has k equivalent classes of vertices. A tiling is *m-Archimedean* when it has m types of vertices. Using our symmetry detection algorithm, we have classified all tilings in the collections acquired [11] and have confirmed the classification given by Galebach [19]. The supplementary material for this paper contains examples of tilings having each of the 17 wallpaper groups.

Crystallographic fundamental region. Each tiling has a *minimal crystallographic fundamental region* that contains exactly one representative of each transitive equivalence class of its symmetry group. Naturally, the crystallographic fundamental region is smaller than the basic translation cell. In fact, the basic translation cell can be decomposed into copies of the crystallographic fundamental region under symmetries. The rigid geometric structure of this decomposition was identified by Schattschneider [20], who thus explained how to reconstruct the crystallographic fundamental region of the tiling from its wallpaper group. The tiling can be reconstructed from its crystallographic fundamental region by repeated application of its symmetries. While mathematically elegant, this reconstruction is computationally much more complex than the reconstruction described in §4, because close attention is required to handle duplicate vertices and to reconstruct faces from its pieces.

7. Properties of the representation

We shall now discuss some properties of our representation. We follow closely the framework proposed by Requicha [21] for discussing representation schemes and their properties. A *representation scheme* is a relation from a modeling space M of mathematical objects to a representation space R of syntactically correct representations. In other words, a representation scheme is a set of pairs $(m,r) \in M \times R$. In our case, the modeling space M is the space of all periodic tilings of the plane by regular polygons, up to Euclidean similarity. The representation space R is the set of all $(2+n)\times 4$ integer matrices, where $n \geq 1$. These matrices are interpreted according to the concrete representation defined in §3: each row contains lattice coordinates of a point in $\mathbf{Z}[\omega]$; the first two rows represent the translation vectors; the remaining n rows represent the seeds. In the remainder, we discuss properties of our representation organized by Requicha's criteria for a representation scheme.

Domain. The domain of a representation scheme is the set of all objects in M that can be represented in R. The domain characterizes the descriptive power of the scheme. As we have argued in §3, all tilings can be represented in our scheme after applying a Euclidean similarity for normalization. Therefore, the domain of our representation is the full modeling space of all periodic tilings of the plane by regular polygons, up to Euclidean similarity. In this sense, our representation is *comprehensive*.

Validity. The set of valid representations is the range of the representation scheme, that is, the set of all representations in R that represent an object in M. Each representation is a $(2+n)\times 4$ integer matrix. Under the concrete interpretation, this matrix corresponds to a finite non-empty set of cosets of an additive subgroup of $\mathbf{Z}[\omega]$ of rank at most 2: the first two rows generate a subgroup of $\mathbf{Z}[\omega]$ and the remaining n rows are coset representatives for this subgroup. A valid representation is a matrix that represents a tiling. Not every representation is valid: not every matrix corresponds to a tiling because the matrix neither contains nor implies any geometric constraints, such as edges having unit length. At the very least, the matrix must satisfy three basic requirements:

- The two translation vectors must be linearly independent in $\mathbf{R}^2$: equivalently, the area of their parallelogram cannot be zero. This can be checked exactly using the lattice coordinates of the translation vectors, as explained in §5. If the check fails, the representation is not valid.
- The seeds must be inside the basic cell: a point in $\mathbf{Z}[\omega]$ is a seed iff it is equal to its reduction modulo the translation lattice. This can be checked by using grid coordinates, as discussed in §5. If the check fails, the representation is not valid; but it can be repaired by using the reduced points instead of the original points.
- One of the seeds must be the origin: this is trivial to check. If the check fails, the representation is not valid; but it can be repaired by adding the origin as a seed.

The important requirement is that a matrix does represent a tiling. We can check this by trying to reconstruct the patch of the tiling, that is, the faces anchored at the seeds (Fig. 12). If the reconstruction succeeds, then the representation is valid, because we can reconstruct the whole the tiling by translating the patch. We do not need to fully reconstruct the faces of the patch, just check that it can be done. Start by using the basic cell as a window, adding the seeds as inner vertices and the corresponding outer vertices to the cloud, as in the local reconstruction described in §4. Then compute the star of the seeds with Algorithm 1 and run Algorithm 6 (a simplified version of Algorithm 3) for the corners at each seed. The representation is valid iff Algorithm 6 succeeds for all seeds. No gaps exist between faces because vertex stars must be consistent [22]. In summary, not every $(2+n)\times 4$ integer matrix represents a tiling, but we can test whether it does in time $O(n)$.

Algorithm 6

procedure *facetest*(v,k,k')
  for $j = 1$ **to** m **do**
    $m \leftarrow \frac{12}{6-(k'-k)}$
    $v \leftarrow v + \omega^k$
    $k \leftarrow (k + 12/m) \bmod 12$
    if v not in cloud **then**
      return false
    end
  end
  return true
end

Unambiguous. A representation r in R is *unambiguous* when there is exactly one object m in M that is represented by r. A representation scheme is *unambiguous* when all valid representations are unambiguous. Our representation for tilings is unambiguous because a tiling can be unambiguously reconstructed from a matrix that represents it, if the matrix does represent a tiling.

Uniqueness. A representation scheme is *unique* when all valid representations are unique: every object in M has exactly one representation in R. As remarked by Requicha [21], most representation schemes for geometric objects are not unique for conceptually trivial reasons: duplications and permutations of elements listed in the representation (when lists are used as proxies for sets) and position of geometric elements. Our representation for tilings is subject to this trivial non-uniqueness: the order of the translation vectors and of the seeds in the matrix is conceptually immaterial but gives different matrices. (Our abstract representation does not suffer from this non-uniqueness because it uses sets, but it cannot be used as a representation space.)

The choices made during normalization are also sources of non-uniqueness: the vertex chosen to be the origin and the orientation of the tiling with respect to the basic directions directly affect its matrix representation. This is an inevitable consequence of defining M as the space of tilings up to Euclidean similarity.

Our representation suffers from non-uniqueness also for non-trivial algebraic reasons: Different pairs of translation vectors can define the same translation lattice. More precisely, given a pair of translation vectors $T = AW$, the pair of translation vectors given by $T' = UAW$ define the same translation lattice when U is 2×2 unimodular matrix. There are infinitely many unimodular matrices. If the seeds are given by SW, then $S' = USW$ are the corresponding seeds. In other words, the matrices R and $R' = UR$ define the same tiling. Thus, every tiling has infinitely many equivalent matrix representations.

Finally, there is the related issue of *minimality*, which we have avoided until now. Redundant information is a source of non-uniqueness. A representation that includes duplicate seeds is trivially redundant. Indeed, the matrix means to provide a *set* of seeds, but it can only provide an *ordered list* of seeds. Duplicate seeds are inconvenient but harmless: the same set of vertices, and hence the same tiling, will be reconstructed. Duplicate seeds can be removed by reduction modulo the translation lattice. Nontrivial redundancy is easy to create: Take as new translation vectors $t'_1 = n_1 t_1$ and $t'_2 = n_2 t_2$, with $n_1, n_2 \in \mathbf{Z}^*$, and take as new seeds the vertices inside the parallelogram defined by t'_1, t'_2. Then the tiling is also represented by the larger matrix implied by these new elements. This raises the question: how to remove all redundancy from a given representation and convert it into a minimal one? This cannot be answered merely by looking at the translation vectors: they may not be integer multiples of vectors in another pair; even when they are, the seeds may not be redundant. We handle minimality ab initio, by finding a pair of minimal translations vectors for the tiling directly from its vertex cloud, as discussed in §6.

Equivalence. Closely related to uniqueness is the issue of deciding whether two representations define the same tiling. This is a crucial task for identifying and removing duplicate tilings when acquiring [11] or generating large collections of tilings by systematic enumeration. The simplest approach to deciding equivalence of two representations is to generate two large vertex clouds, one for each representation, and test whether they match after applying a translation or a rotation. More precisely, for each seed in one representation, translate the translation grid so that the seed becomes the origin. Then, test the vertex cloud containing the vertices in the cells adjacent to the basic cell as follows. For each basic direction ω^k, rotate the vertex cloud about the origin by multiplying by ω^k, and check whether the rotated vertex cloud is contained in the vertex cloud of the other representation (see §6). Finally, repeat the tests after inverting the roles of the two representations.

Conciseness. A representation should contain just enough information to represent an object. Some redundancy is allowed, if it simplifies some operations. Our representation for tilings is very concise, even when the representation is not minimal. By definition, minimal representations contain no redundancy. The entries in the matrices of minimal representations are typically small integers.

Ease of creation. Our representation for tilings is suitable for both manual creation and automatic extraction from images. We described in a previous paper [11] how to extract a minimal representation for a tiling automatically from an image of it. To extract a representation for a tiling manually from an image or even a sketch of it, follow the recipe in §3. First fix a vertex to be the origin and rotate the drawing around the origin so that some edge becomes horizontal. Then locate two vertices nearest to the origin that are equivalent to the origin by two linearly independent translations.[1] The seeds are the vertices inside the parallelogram defined by the translation vectors. The lattice coordinates for all elements are found by following paths along the edges. This process automatically gives a minimal representation because the vertices defining the translation vectors are the feasible ones closest to the origin.

[1] This can be done by printing two semi-transparent drawings of the tiling and sliding one over the other until they match. This is the hard step for a human, especially for large repetitive tilings.

Efficacy in the context of applications. Our representation is simple to understand, as explained in §3: it focuses on the vertices, the simplest elements of the tiling, and gives them integer coordinates that are easy to extract from paths along the edges. It is simple to reconstruct a tiling from a representation, as explained in §4. Complementing the compact external representation (an integer matrix) with an explicit internal representation of its vertices (the cloud) simplifies many algorithms. No complex data structures and no nearest-neighbors searches are needed. Moreover, most tasks have robust solutions based on error-free integer arithmetic. They need no geometric tests and no arbitrary tolerances that plague geometric algorithms.

8. Related work

Representation. Kaplan [3] gave a highly detailed symbolic description of isohedral tilings, in which all tiles are transitively equivalent. He developed edge shape parameterization and tiling vertex parameterization for each type of isohedral tiling. He also considered coloring rules over his representation. However, that approach does not generalize in an obvious way for periodic tilings having more than one type of tile.

Dress and Huson [23] and Delgado-Friedrichs [24] described a general data structure for representing periodic tilings using graph symbols and adjacency functions between "chambers", a triangulation of the original tiles. Huson [25] used these symbols to enumerate tile-k-transitive tilings of the Euclidean plane, the sphere, and the hyperbolic plane. Recently, Zeller et al. [26] published Tegula, a program for displaying and exploring two-dimensional periodic tilings. Their graph symbol and chamber labeling give all the symmetries of the tiling. However, rendering such symbols in large scale is costly, since the classification of each point relies only on the neighboring chambers. Our translation-only approach gives a much simpler and faster method for drawing tilings with regular polygons inside arbitrarily large regions of the plane.

Our approach is closer in spirit to that of Ostromoukhov [27], who explained how to represent a plane ornamental pattern, such as an Islamic pattern, by manually analyzing its structure. He also showed how to render the pattern from its representation. A key point in his approach is strand analysis, which describes how the symmetries of the pattern behave in a fundamental region. Our approach with seeds is similar but, as we argued in §6, is much simpler, since it only requires translational symmetry.

Complex numbers have been used to represent aperiodic tilings. Shutov and Maleev [28] refined the Baake construction of Penrose tilings using 5th roots of the unity. Pautze [29] used 2nth roots of the unity to represent substitution aperiodic tilings with dihedral symmetry.

Preliminary forms of our representation were described briefly in the context of rendering and acquisition [10, 11]. Our previous reconstruction algorithm [10] relied on nearest-neighbors searches and geometric computations with numerical tolerances, which worked well for that task but may not work reliably for more general applications. Our previous method for deciding the equivalence of two representations [11] relied on the Hermite normal form of integers matrices. In both cases, the algorithms described here are much simpler and robust because they rely solely on the cloud.

Catalogs. Efforts in producing catalogs of tilings by regular polygons are recurrent in the literature. Our work was motivated by the catalog by Sá and Sá [12]. In 2009, Lenngren [4] reviewed the efforts in the enumeration of k-uniform tilings: $k = 1$ by Kepler in 1619 and by Sommerville in 1905, $k = 2$ by Krötenheerdt in 1969, $k = 3$ in by Chavey in 1984, and $k = 4, 5, 6$ by Galebach in 2002. Recent work on tilings by regular polygons includes Chavey on dodecagon dense tilings [30] and Connelly–Dickinson on circle packings [15].

Galebach's collection of tilings [19] remains the state of the art in the classification of k-uniform tilings [31]. Unfortunately, only low-resolution images of line drawings are available, but no vertex coordinates or code. Wikipedia includes a smaller catalog [32] of tilings represented as SVG, citing Chavey [33] and Galebach [19] as sources. We have acquired [11] and published [9] representations for all tilings in Galebach's collection and have confirmed their stated uniformity and Archimedean character using the methodology described in §6.

Symmetry. Liu et al. [34] carefully reviewed the field of computational symmetry, including the long history of symmetry detection algorithms, dating back to 1932. Our algorithms for detecting symmetries in tilings work on exact representations instead of images and look for a small set of symmetries that is known a priori. Thus, we are able to find exact symmetries efficiently.

9. Conclusion

No complete classification exists for periodic tilings of the plane by regular polygons. We described here a representation for such tilings that is simple and supports robust algorithms. Applications that deal with tilings by regular polygons can use our representation and the data we have published. We hope that our representation will encourage and enable the comparison and classification of existing tilings and a systematic search for new tilings. These are clear directions for long-term future work. As a shorter-term goal, we propose the investigation of normal forms for our representations of tilings, which would turn deciding equivalence of two representations into a simple comparison of their normal forms. Catalogs of tilings in our representation would then be able to use standard entries for each tiling.

Acknowledgements. J. E. Soto Sánchez was partially supported by a CNPq doctoral scholarship [22]. L. H. de Figueiredo is partially supported by a CNPq research grant. This research was done in the Visgraf Computer Graphics laboratory at IMPA, at FGV EMAp, and at the Department of Computer Science, University College London. Visgraf is supported by the funding agencies FINEP, CNPq, and FAPERJ, and also by gifts from IBM Brasil, Microsoft, NVIDIA, and other companies.

References

[1] Grünbaum, B, Shephard, GC. Tilings and patterns. W. H. Freeman; 1989.
[2] Conway, JH, Burgiel, H, Goodman-Strauss, C. The symmetries of things. AK Peters; 2008.
[3] Kaplan, CS. Introductory tiling theory for computer graphics. Synthesis Lectures on Computer Graphics and Animation 2009;4(1):1–113.
[4] Lenngren, N. *k*-uniform tilings by regular polygons. Tech. Rep. U.U.D.M. project report 2009:23; Uppsala University; 2009.
[5] Grünbaum, B, Shephard, GC. Tilings by regular polygons. Mathematics Magazine 1977;50(5):227–247.
[6] Kaplan, CS. Islamic star patterns from polygons in contact. In: Proceedings of the Graphics Interface 2005. 2005, p. 177–185.
[7] Nasri, A, Benslimane, R. Parametric shape grammar formalism for Moorish geometric design analysis and generation. Journal on Computing and Cultural Heritage 2017;10(4):1–20.
[8] Peng, CH, Pottmann, H, Wonka, P. Designing patterns using triangle-quad hybrid meshes. ACM Transactions on Graphics 2018;37(4):107.
[9] Soto Sánchez, JE, Medeiros e Sá, A, de Figueiredo, LH. Periodic tilings of regular polygons. 2020. URL: chequesoto.info/tilings.html.
[10] Medeiros e Sá, A, de Figueiredo, LH, Soto Sánchez, JE. Synthesizing periodic tilings of regular polygons. In: Proceedings of SIBGRAPI 2018. IEEE Computer Press; 2018, p. 17–24.
[11] Soto Sánchez, JE, Medeiros e Sá, A, de Figueiredo, LH. Acquiring periodic tilings of regular polygons from images. The Visual Computer 2019;35(6):899–907.
[12] Sá, R, Medeiros e Sá, A. Sobre malhas arquimedianas. Olhares; 2017.
[13] Hilbert, D, Cohn-Vossen, S. Geometry and the imagination. Chelsea; 1952.
[14] Bremner, MR. Lattice basis reduction. CRC Press; 2012.
[15] Connelly, R, Dickinson, W. Periodic planar disc packings. Philosophical Transactions of the Royal Society A 2014;372(2008):20120039.
[16] Botsch, M, Kobbelt, L, Pauly, M, Alliez, P, Levy, B. Polygon Mesh Processing. AK Peters; 2010.
[17] Washburn, DK, Crowe, DW. Symmetries of culture: Theory and practice of plane pattern analysis. University of Washington Press; 1988.
[18] Bonner, J. Islamic geometric patterns. Springer; 2017.
[19] Galebach, B. *n*-uniform tilings. 2002. URL: probabilitysports.com/tilings.html.
[20] Schattschneider, D. The plane symmetry groups: their recognition and notation. American Mathematical Monthly 1978;85(6):439–450.
[21] Requicha, AG. Representations for rigid solids: Theory, methods, and systems. ACM Computing Surveys 1980;12(4):437–464.
[22] Soto Sánchez, JE. On periodic tilings with regular polygons. Ph.D. thesis; IMPA; 2020. URL: chequesoto.info/thesis.html.
[23] Dress, AWM, Huson, D. On tilings of the plane. Geometriae Dedicata 1987;24(3):295–310.
[24] Delgado-Friedrichs, O. Data structures and algorithms for tilings I. Theoretical Computer Science 2003;303(2):431–445.
[25] Huson, DH. The generation and classification of tile-k-transitive tilings of the euclidean plane, the sphere and the hyperbolic plane. Geometriae Dedicata 1993;47(3):269–296.
[26] Zeller, R, Friedrichs, OD, Huson, DH. Tegula – exploring a galaxy of two-dimensional periodic tilings. 2020. arXiv:2007.10625.
[27] Ostromoukhov, V. Mathematical tools for computer-generated ornamental patterns. In: Electronic Publishing, Artistic Imaging, and Digital Typography; vol. 1375 of *Lecture Notes in Computer Science*. Springer; 1998, p. 193–223.
[28] Shutov, AV, Maleev, AV. Penrose tilings as model sets. Crystallography Reports 2015;60(6):797–804.
[29] Pautze, S. Cyclotomic aperiodic substitution tilings. Symmetry 2017;9(2):19.
[30] Chavey, D. Tilings by regular polygons iii: dodecagon-dense tilings. Symmetry: Culture and Science 2014;25(3):193–210.
[31] The On-Line Encyclopedia of Integer Sequences, . A299780. 2018. URL: oeis.org/A299780.
[32] Wikipedia, . Euclidean tilings by convex regular polygons. 2020. URL: en.wikipedia.org/wiki/Euclidean_tilings_by_convex_regular_polygons.
[33] Chavey, D. Tilings by regular polygons. II. A catalog of tilings. Computers & Mathematics with Applications 1989;17:147–165.
[34] Liu, Y, Hel-Or, H, Kaplan, CS, Gool, LJV. Computational symmetry in computer vision and computer graphics. Foundations and Trends in Computer Graphics and Vision 2010;5(1-2):1–195.