Méthodes basées sur les pavages pour informatique graphique

Victor Ostromoukhov Université de Lyon

Avec participation de

Roger Hersch EPFL

David Salesin University of Washington

Branko Grunbaum University of Washington

Craig Kaplan University of Waterloo

Charles Donohue University of Montreal

Pierre-Marc Jodoin University of Montreal

David Coeurjolly Université de Lyon

Florent Wachtel Université de Lyon

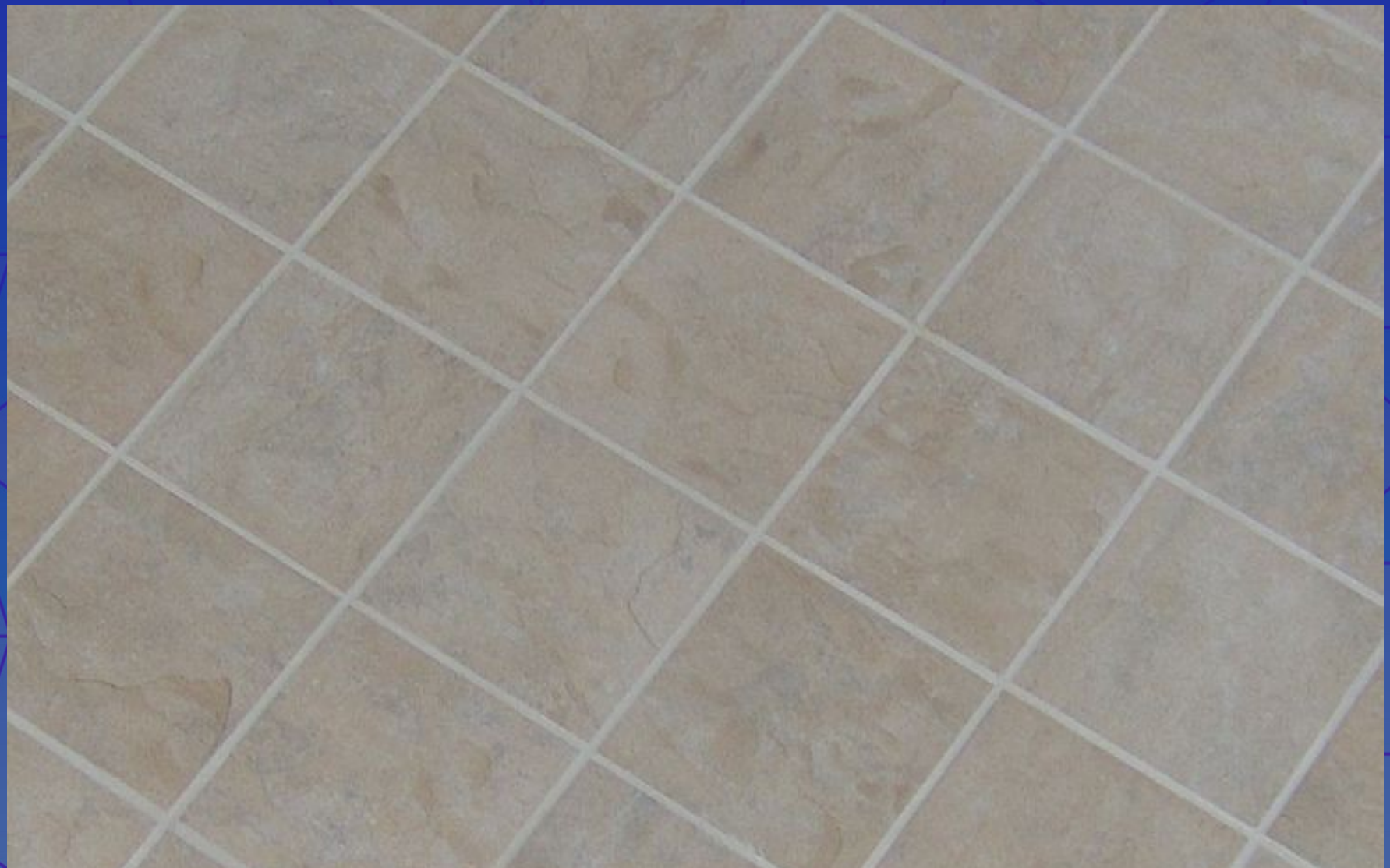
Adrien Pilleboue Université de Lyon

Mathieu Desbrun Caltech

Fernando de Goes Caltech

Catherine Breeden Stanford

What is Tiling?



Victor Ostromoukhov

Professor (PR1)

Presently: Université de Lyon 1, CNRS-LIRIS
Formerly: Université de Montréal, DIRO
Contact information

Bio:
Victor Ostromoukhov is a professor at Université de Lyon 1/CNRS. He graduated from Phys-Tech (MIPT, Moscow, 1980). He then spent several years with prominent European and American industrial companies (SG2, Paris; Olivetti, Paris and Milan; Canon Information Systems, Cupertino, CA) as a research scientist and/or computer engineer. He received his PhD in Computer Science from Swiss Federal Institute of Technology (EPFL, Lausanne, 1995), where he then continued to work as a lecturer and senior researcher. He was an invited professor at University of Washington (Seattle, 1997) and at iMAGIS/INRIA (Grenoble, 2002). He spent a year as research scientist at Massachusetts Institute of Technology, Cambridge, MA, in 1999-2000. He has been an Associate Professor at Université de Montréal, from 2000 to 2009. He has been appointed as Full Professor at Université de Lyon 1, since September 2009, In 2010, he received a Distinguished Chair from French National Research Agency (Chaire d'Excellence ANR).
Curriculum Vitae: short or detailed

Research:

- Research interests: Digital Halftoning, Photorealistic and Nonphotorealistic Rendering, Importance Sampling, Color Science, Tilings and Patterns, Low-Discrepancy Sequences, Mote Carlo and Quasi-Monte Carlo Methods
- Recent publications: Abstracts & PDFs, list

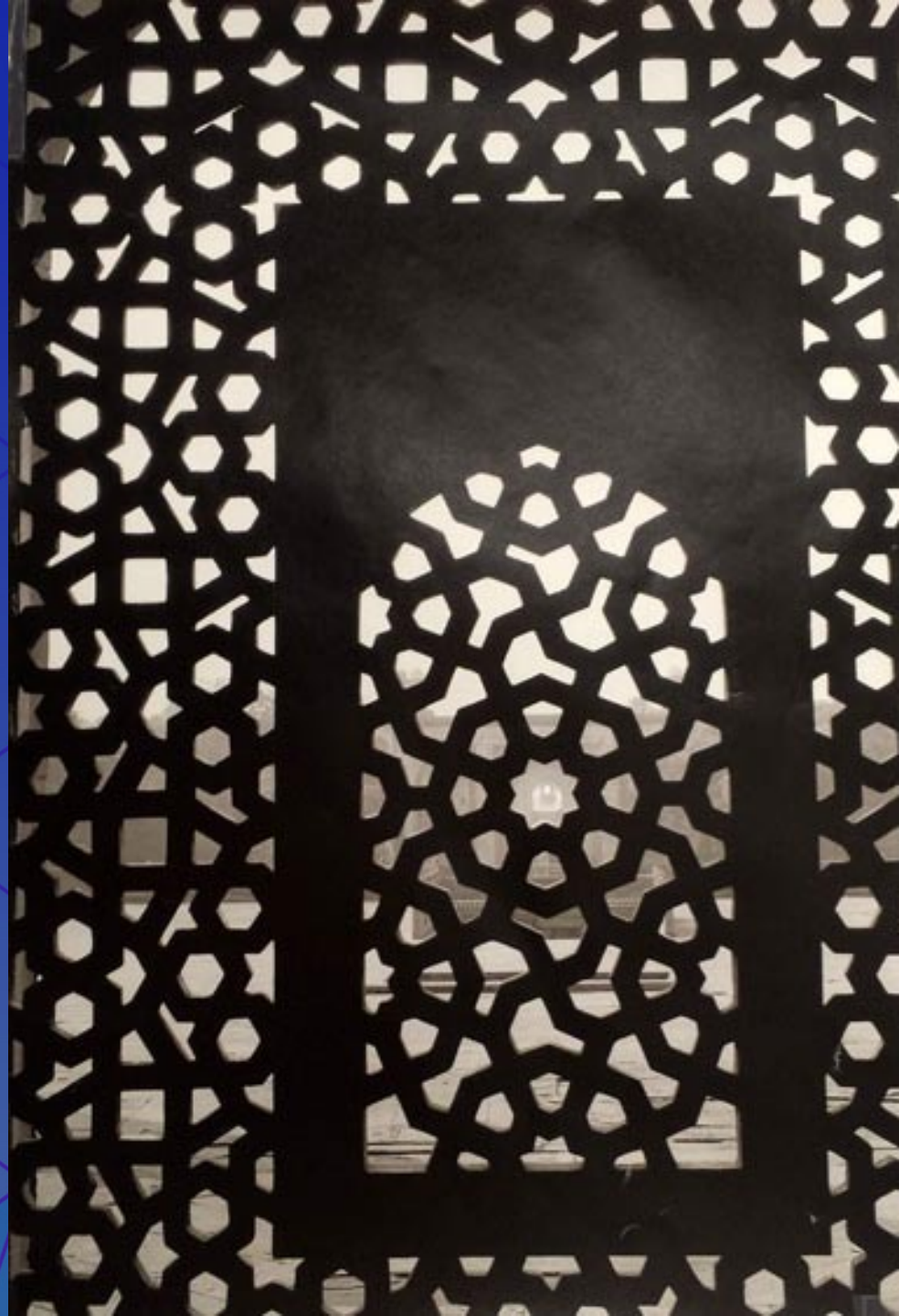
Teaching:

What is Tiling?

Definition **Tiling**. A *tiling* is a countable collection $\mathcal{T}$ of *tiles* $\{T_1, T_2, \ldots\}$, such that:

1. Every tile is a closed topological disk.
2. Every point in the plane is contained in at least one tile;
3. The interiors of the tiles are pairwise disjoint; and
4. The tiles are uniformly bounded.

Tiling in Architecture

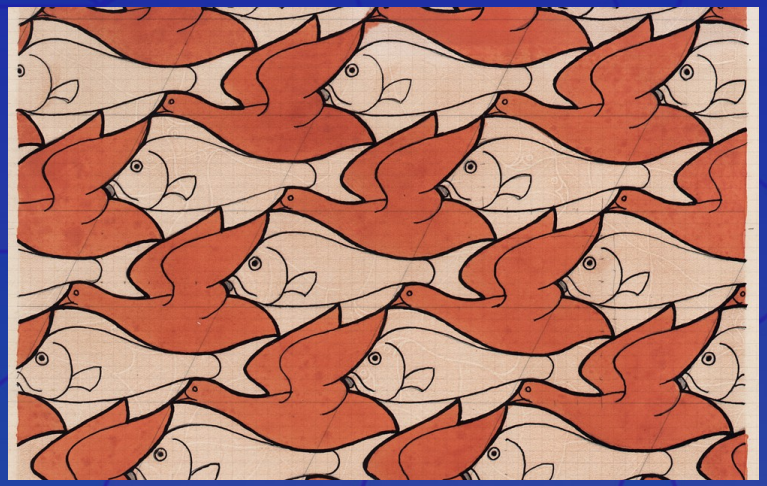


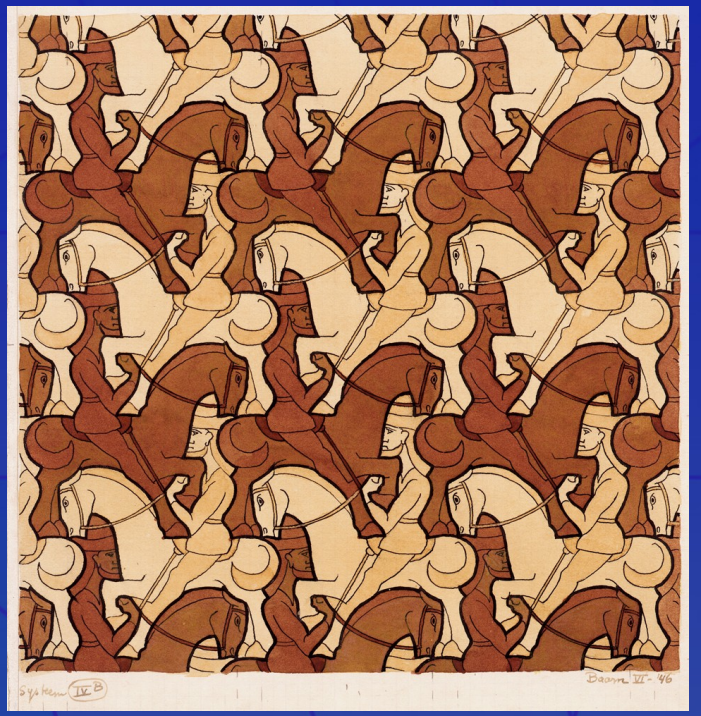
Tiling in Architecture

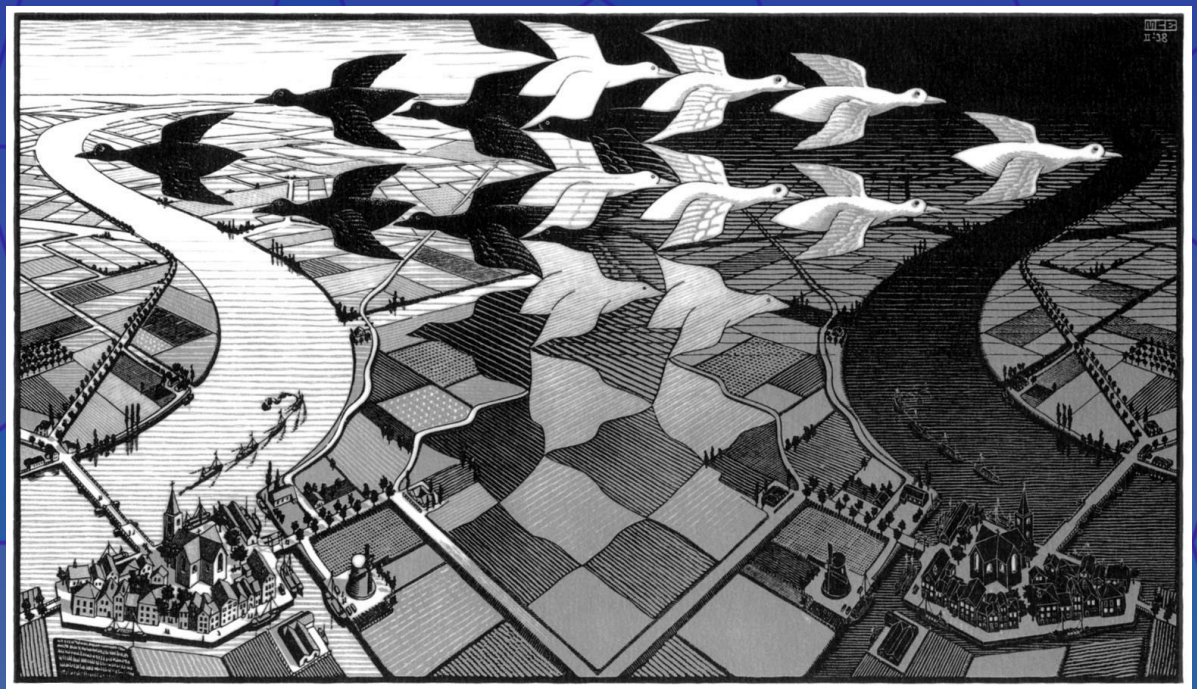
Tiling in Architecture

Tiling in Architecture

Escher

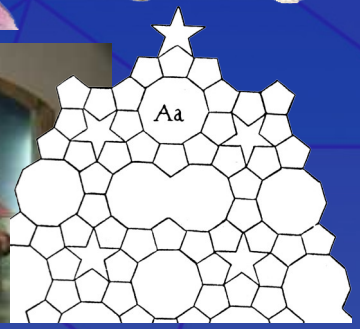




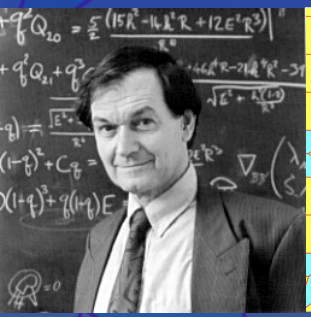


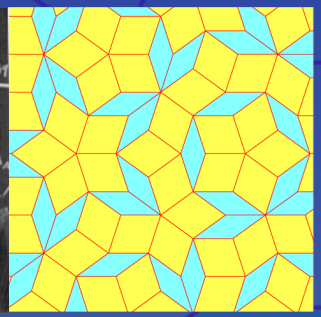
Penrose Tiling: Milestones

- Circa 1200 AD, Fibonacci (Leonardo of Pisa)
  Rabbit Sequence, Fibonacci Numbers



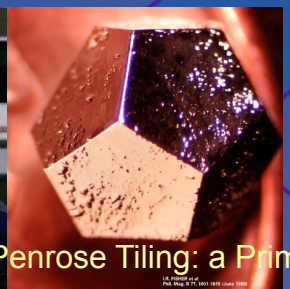
- 1619, Johannes Kepler
  Harmonice Mundi, 5-fold Tiling Problem





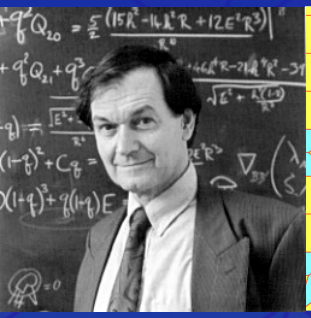
- 1974, Sir Roger Penrose
  Pentaplexity, Penrose Tiling

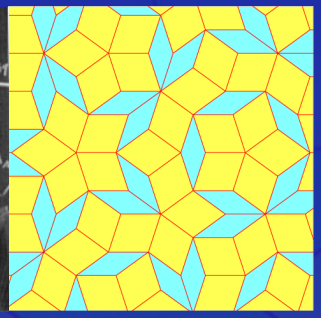
Diffraction Pattern

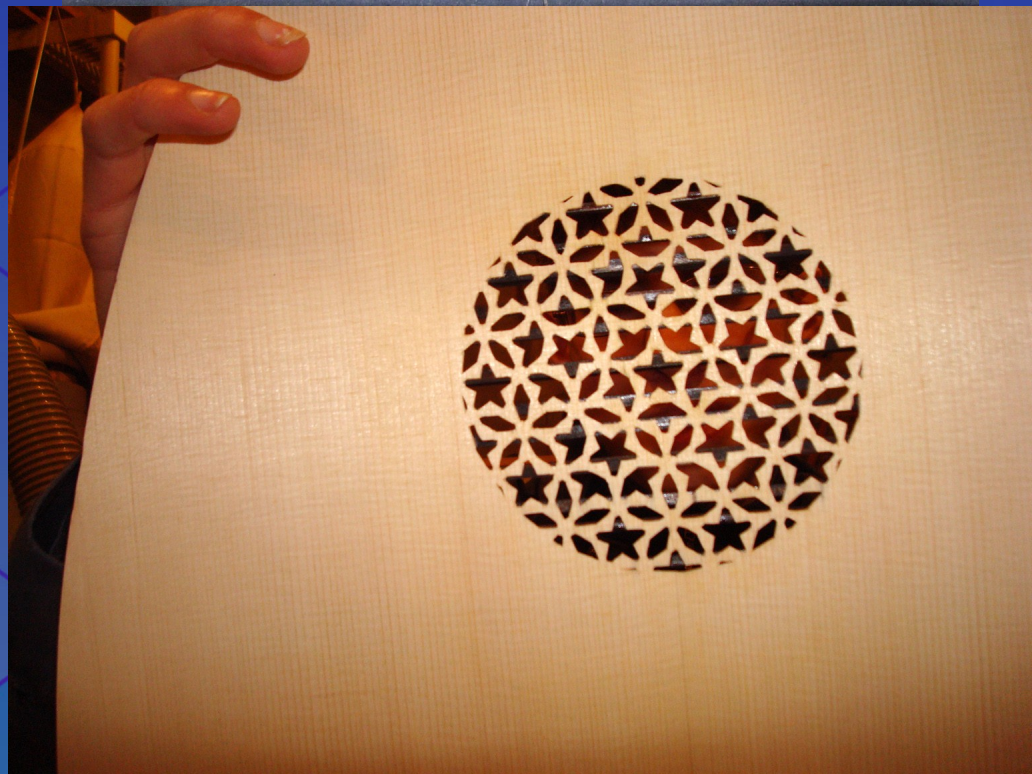


- 1984, Dan Shechtman *et al.*
  Discovery of Quasi-Crystals

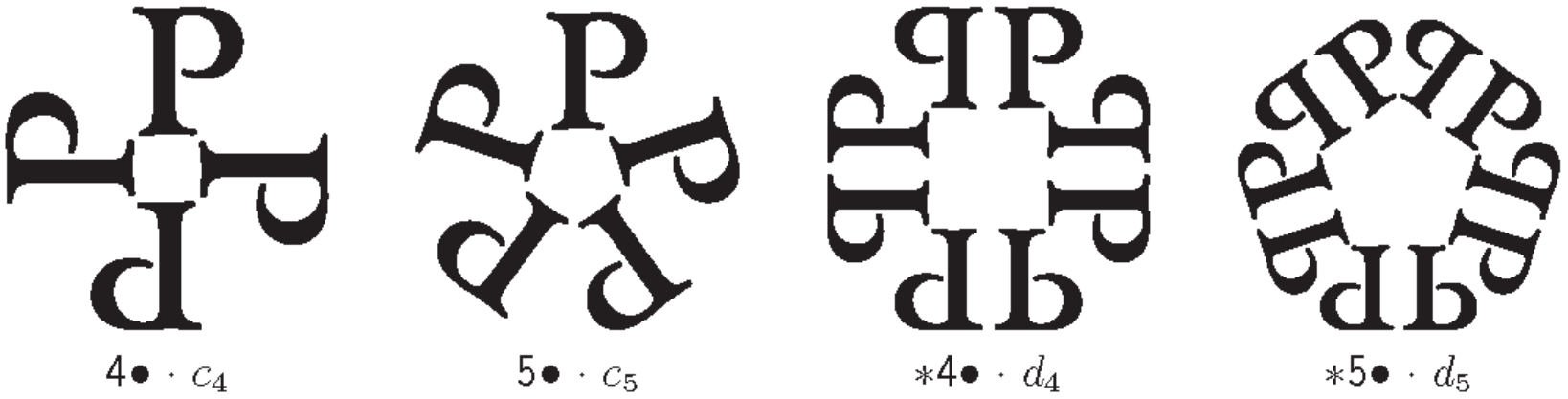
Penrose Tiling





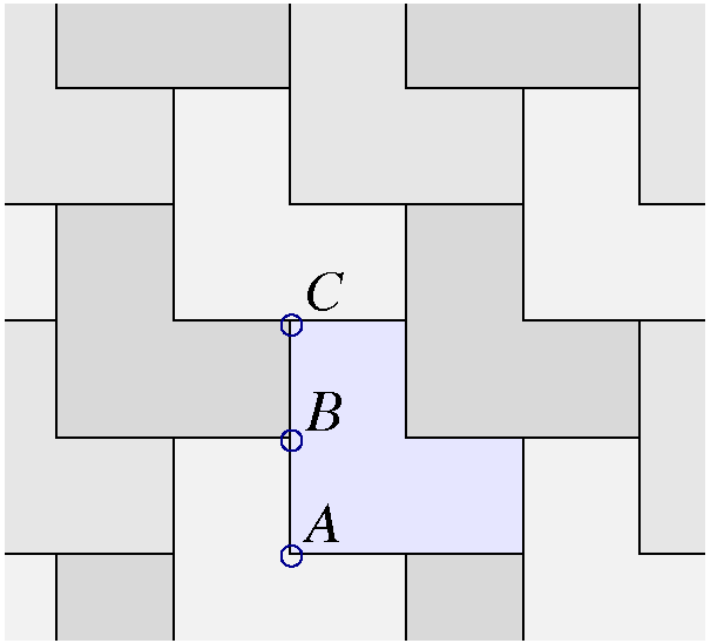


Symmetry



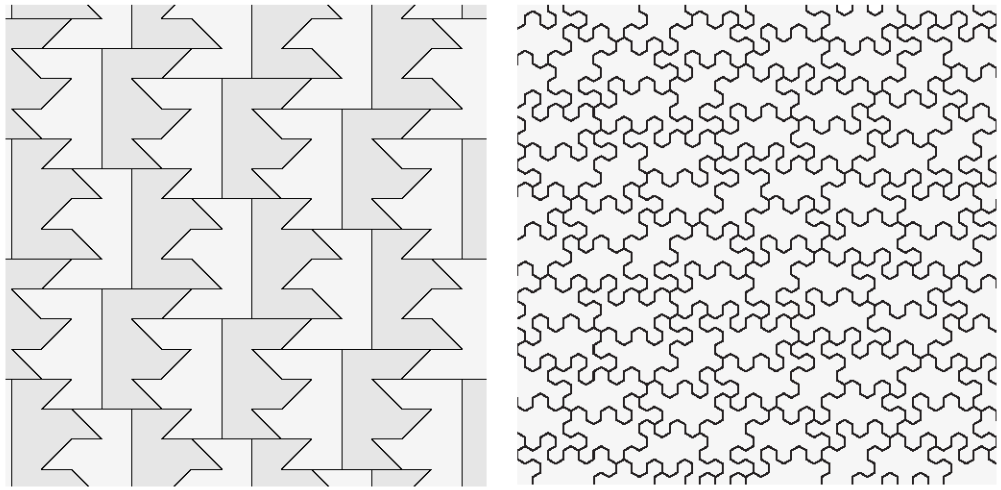
Examples of figures in the Euclidean plane with cyclic or dihedral symmetry. Each figure is labeled with both its orbifold signature and the name of the abstract group given by its symmetries.

Monohedral Tiling with Translational Symmetry



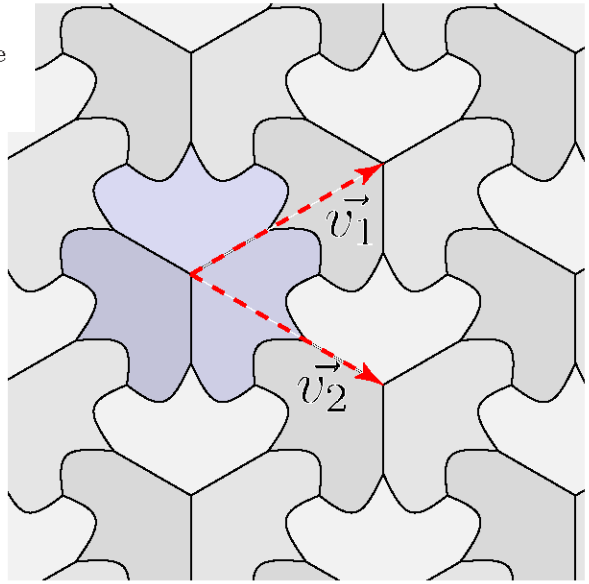
The features of a polygonal tiling. For the tile highlighted in blue, A is a shape vertex but not a tiling vertex, B is a tiling vertex but not a shape vertex, and C is both a tiling vertex and a shape vertex. Because the tiling vertices and shape vertices do not coincide, the tiling is not edge-to-edge.

Monohedral/Polyhedral, Isohedral Tilings



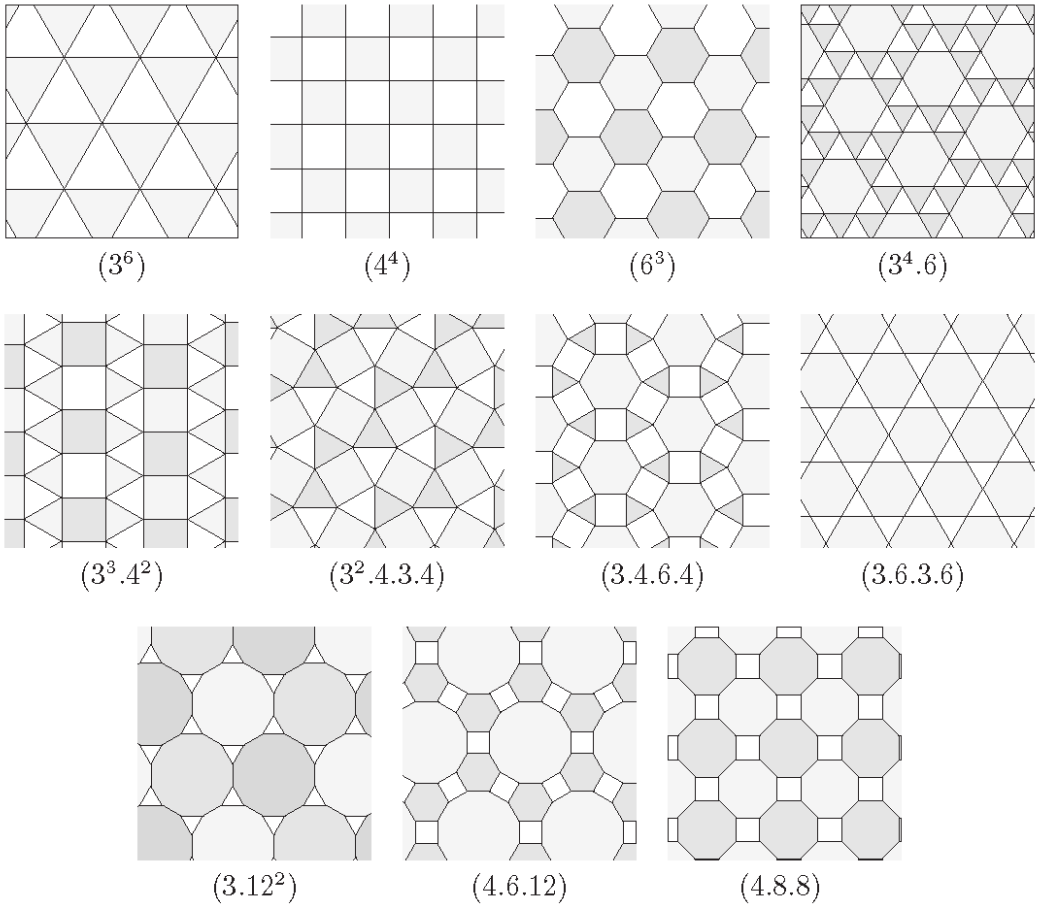
(a) (b)

Anisohedral prototiles. Heesch's 1935 2-anisohedral prototile is shown in (a). The current record holder, a 10-anisohedral 16-hex discovered by Joseph Myers, is shown in (b).



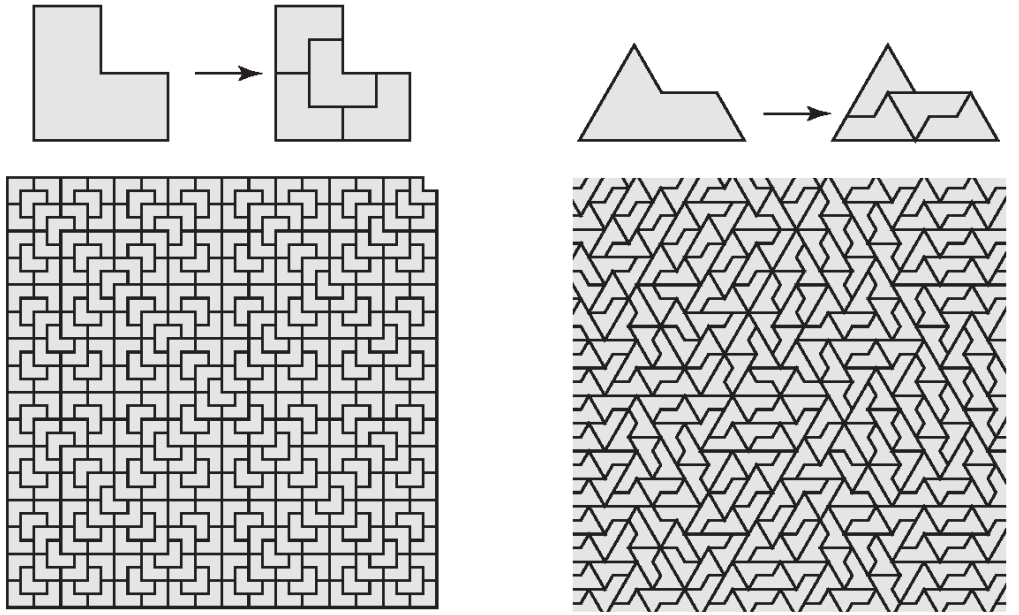
An example of an isohedral tiling of type IH16. A single translational unit of t tiling is shown through the two translation vectors $\vec{v_1}$ and $\vec{v_2}$ and the three coloured aspects

LAVES TILINGS



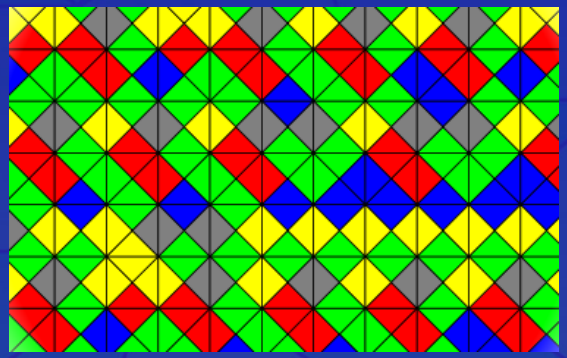
The eleven uniform Euclidean tilings, also known as Archimedean tilings. The tiling $(3^4.6)$ occurs in left-handed and right-handed forms.

Non-Periodic vs. Aperidic Tiling



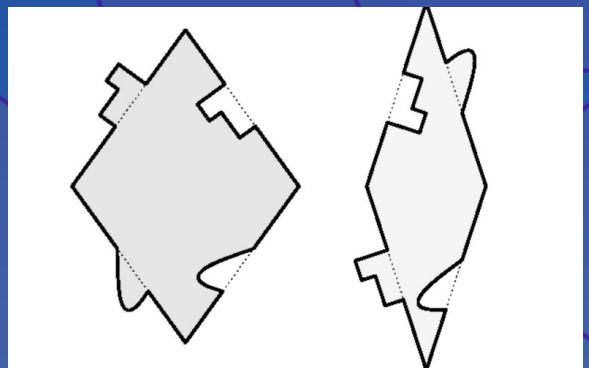
Two well-known examples of rep-tiles: the Chair tiling on the left and the Sphinx tiling on the right.

APERIODIC TILINGS



WangTiling

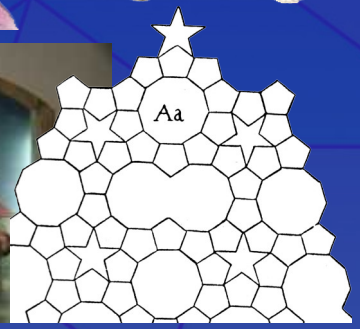
RobinsonTiling



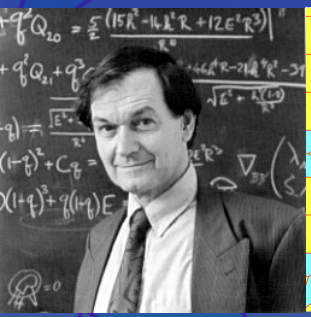
Penrose Tiling

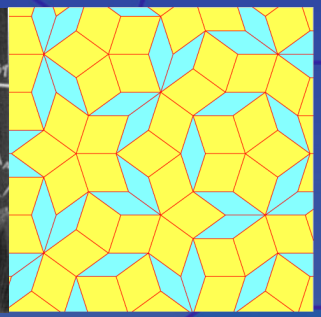
Penrose Tiling: Milestones

- Circa 1200 AD, Fibonacci (Leonardo of Pisa)
  Rabbit Sequence, Fibonacci Numbers



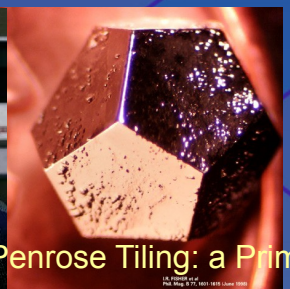
- 1619, Johannes Kepler
  Harmonice Mundi, 5-fold Tiling Problem





- 1974, Sir Roger Penrose
  Pentaplexity, Penrose Tiling

Diffraction Pattern



- 1984, Dan Shechtman *et al.*
  Discovery of Quasi-Crystals

Penrose-based Sampling System [SIGGRAPH'2004]

Fast Hierarchical Importance Sampling with Blue Noise Properties

Victor Ostromoukhov*
University of Montreal

Charles Donohue †
University of Montreal

Pierre-Marc Jodoin ‡
University of Montreal

Problem Statement

Given:

- Importance Density
  I(x,y)

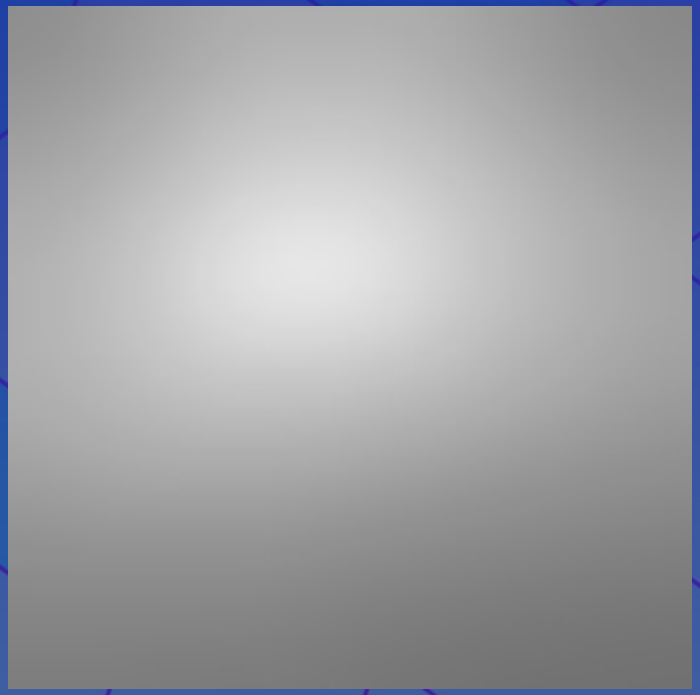
Problem Statement

Given:

- Importance Density *I(x,y)*

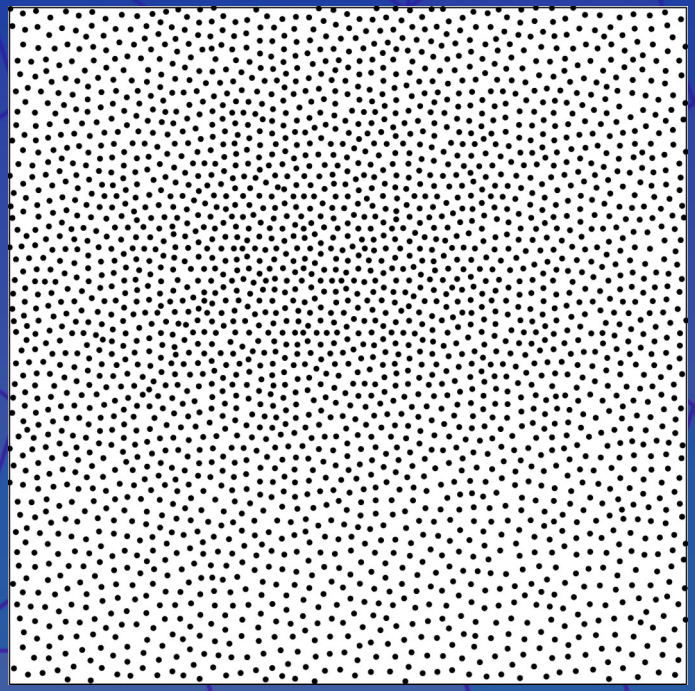
high

low



Find:

- Discrete Sample Distribution Locally Proportional to *I(x,y)*



Problem Statement

Given:

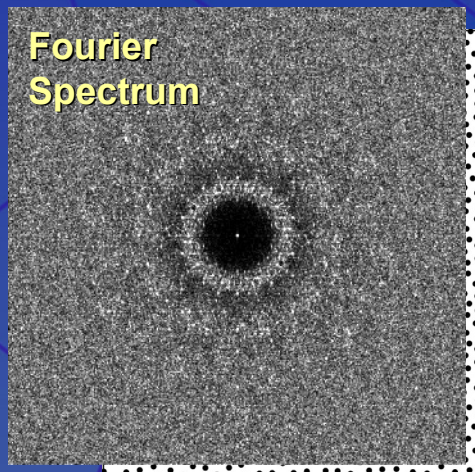
- Importance Density *I(x,y)*

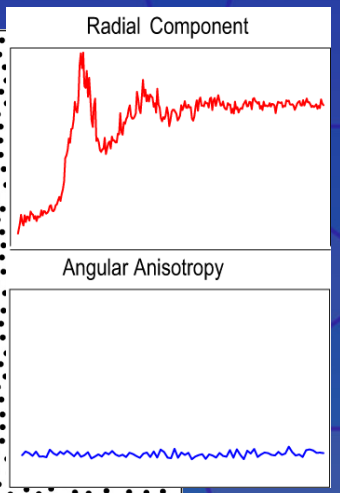
high

low

Find:

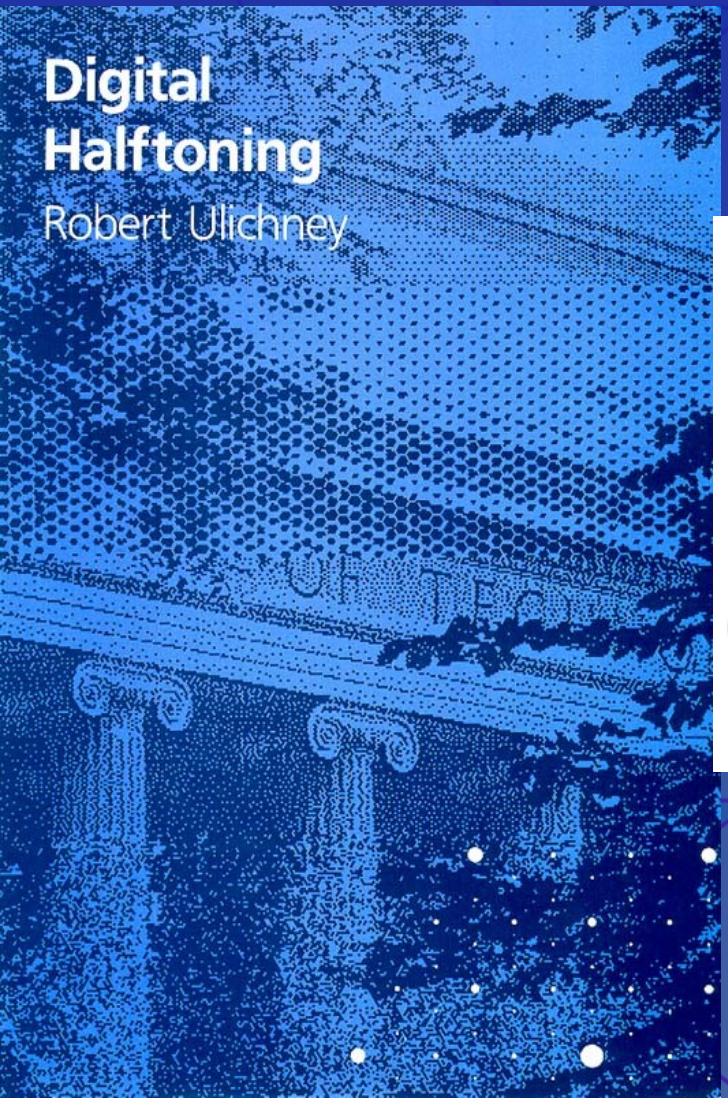
- Discrete Sample Distribution Locally Proportional to *I(x,y)*

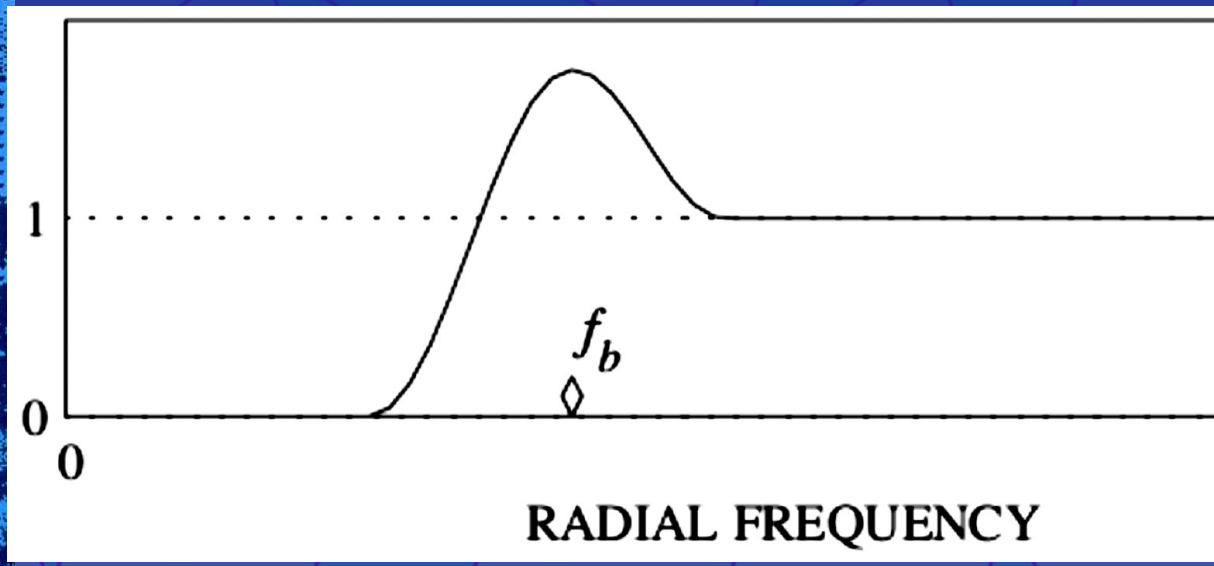




- With Blue Noise Fourier Spectrum

Variance estimation in Monte Carlo Sampling and Blue-Noise Spectrum





[Ulichney 1987]

Variance estimation in Monte Carlo Sampling and Blue-Noise Spectrum

$$\mathrm{V}(\Delta) = \int \mathrm{V}\left(\hat{\mathbf{S}}(\omega)\right) \left(\hat{f}_{\Pi}(-\omega)\right)^2 \mathrm{d}\omega$$

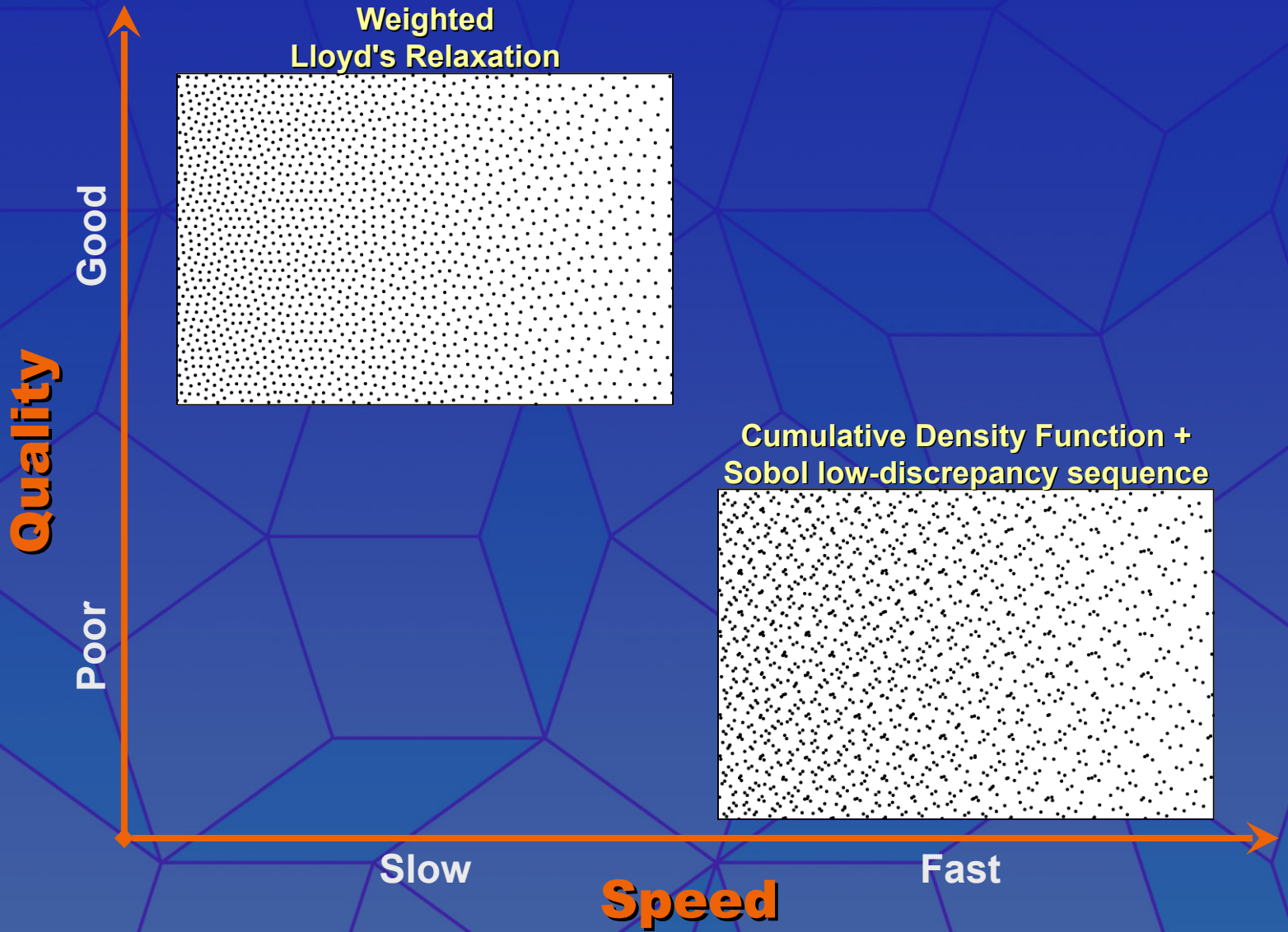
To appear in ACM TOG 32(4).

Fourier Analysis of Stochastic Sampling Strategies for Assessing Bias and Variance in Integration

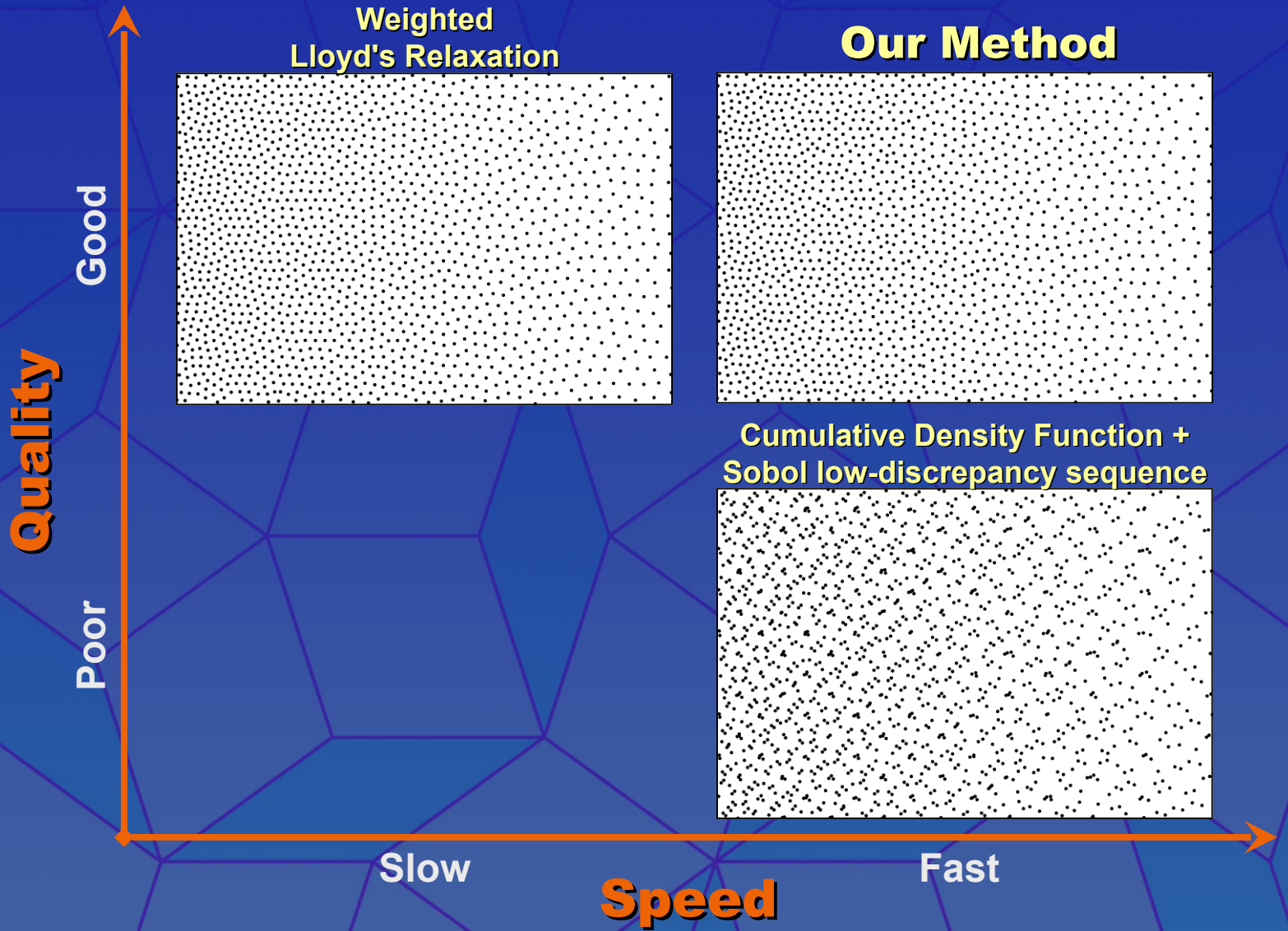
Kartic Subr*
University College London

Jan Kautz†
University College London

Quality vs. Speed

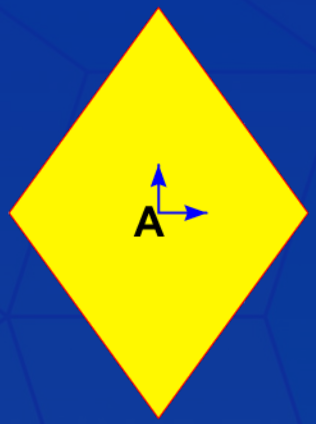


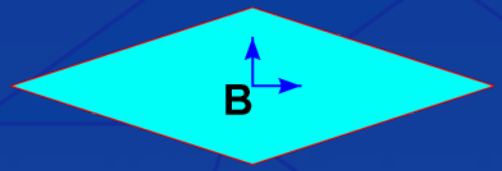
Quality vs. Speed



Penrose Tiling: A Primer

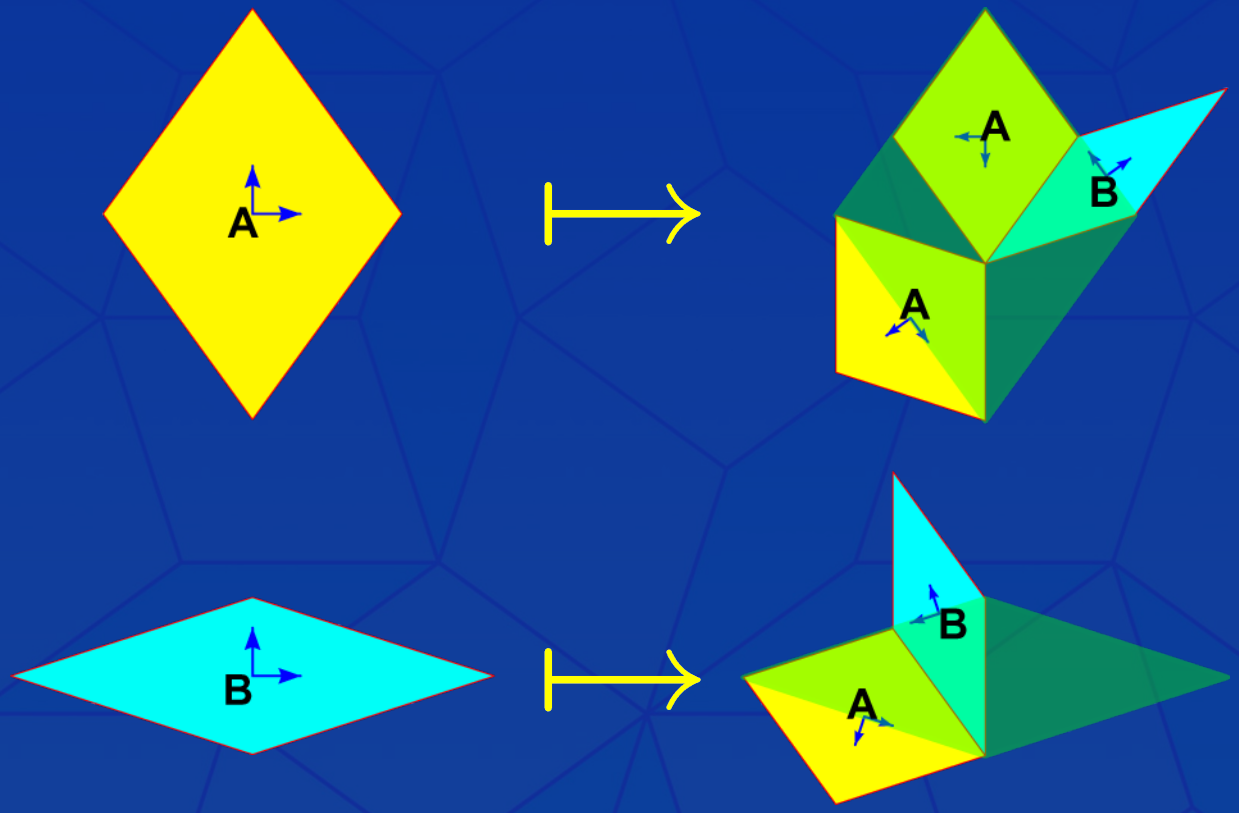
Penrose Tiling: Definition by Production Rules



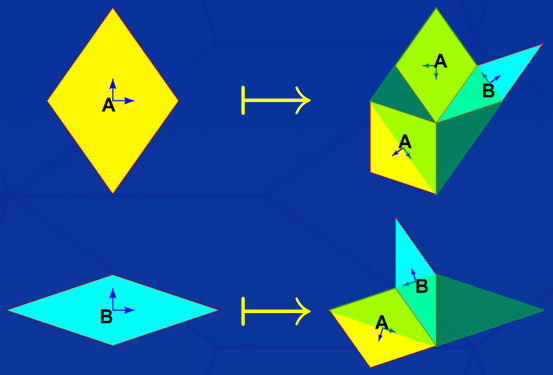


Penrose Tiling: Definition by Production Rules

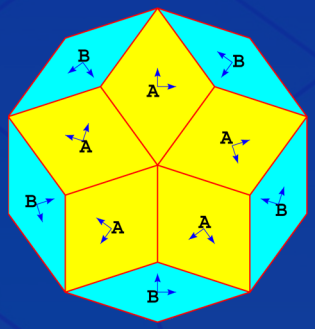
$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



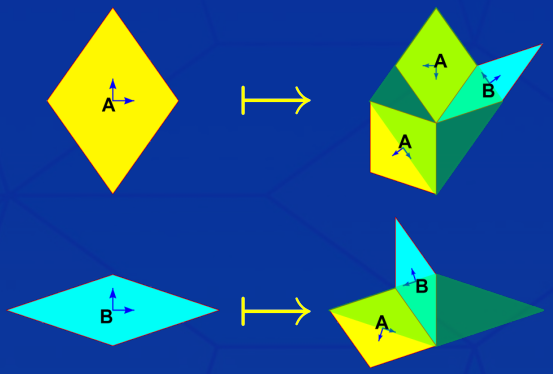
Penrose Tiling: Definition by Production Rules



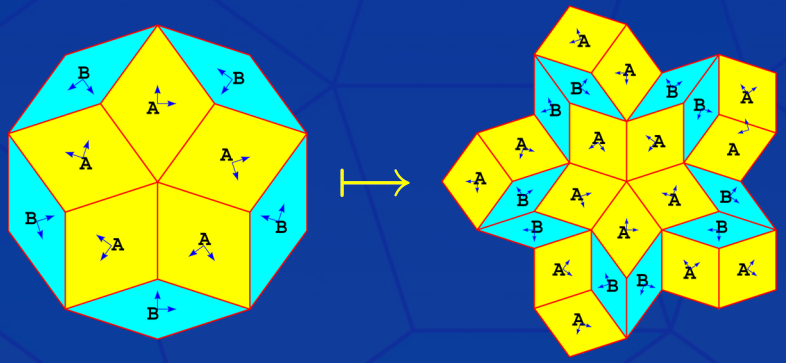
$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



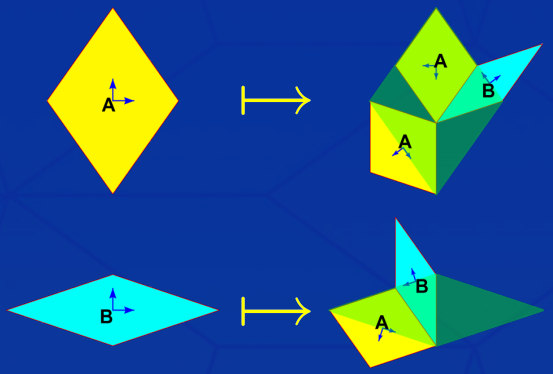
Penrose Tiling: Definition by Production Rules



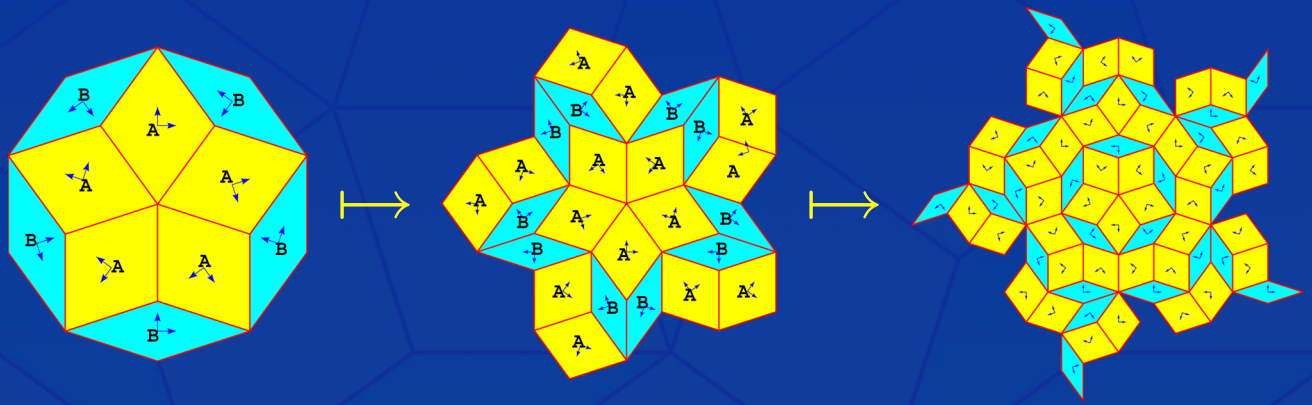
$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



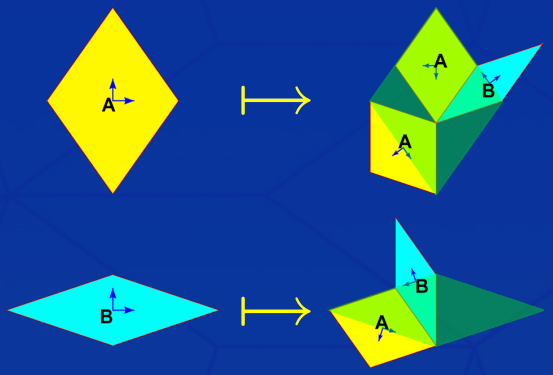
Penrose Tiling: Definition by Production Rules



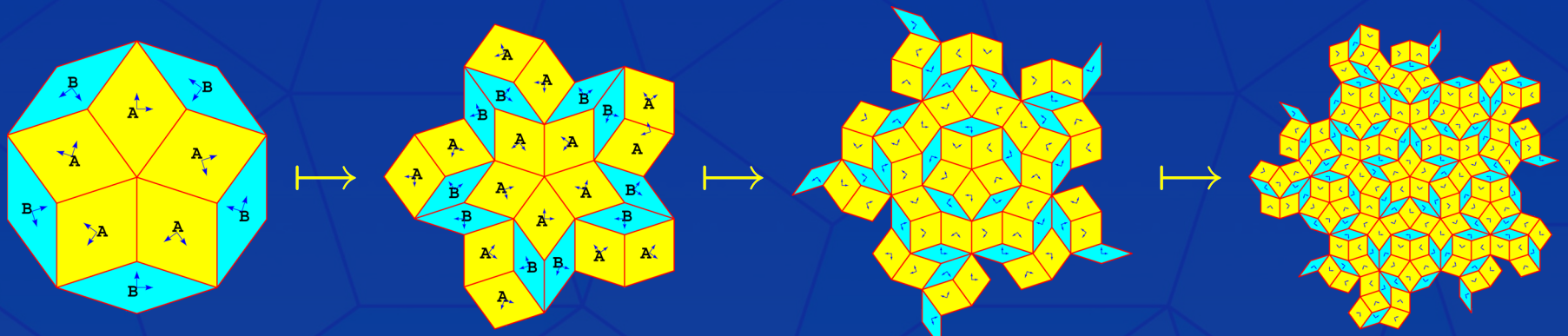
$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



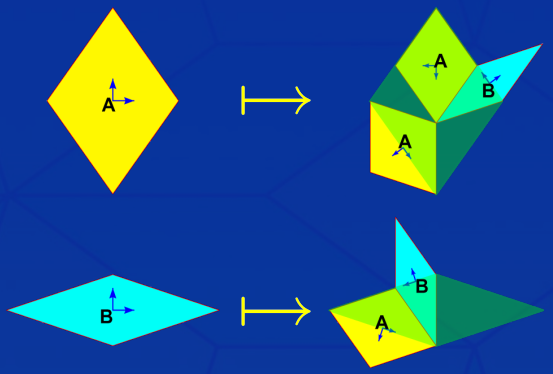
Penrose Tiling: Definition by Production Rules



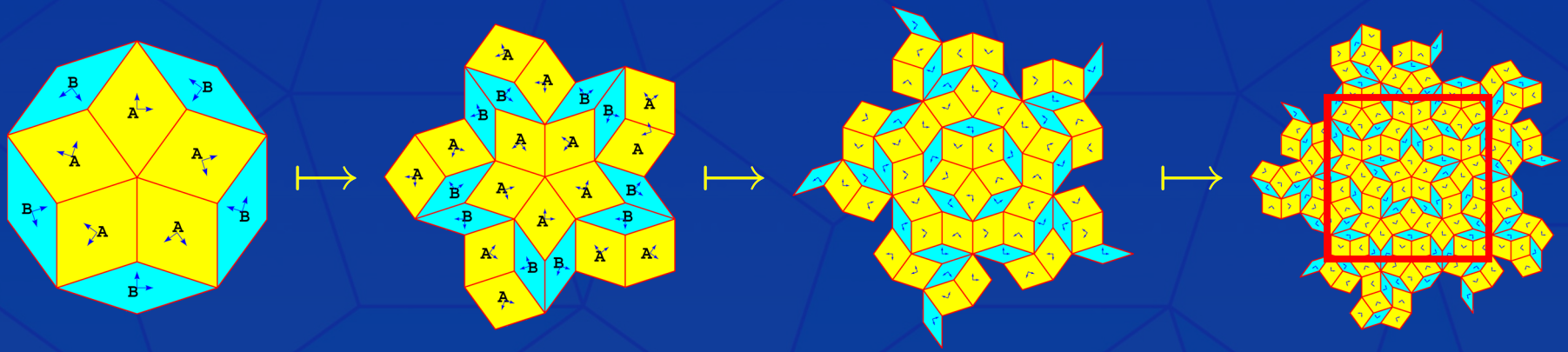
$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



Penrose Tiling: Definition by Production Rules

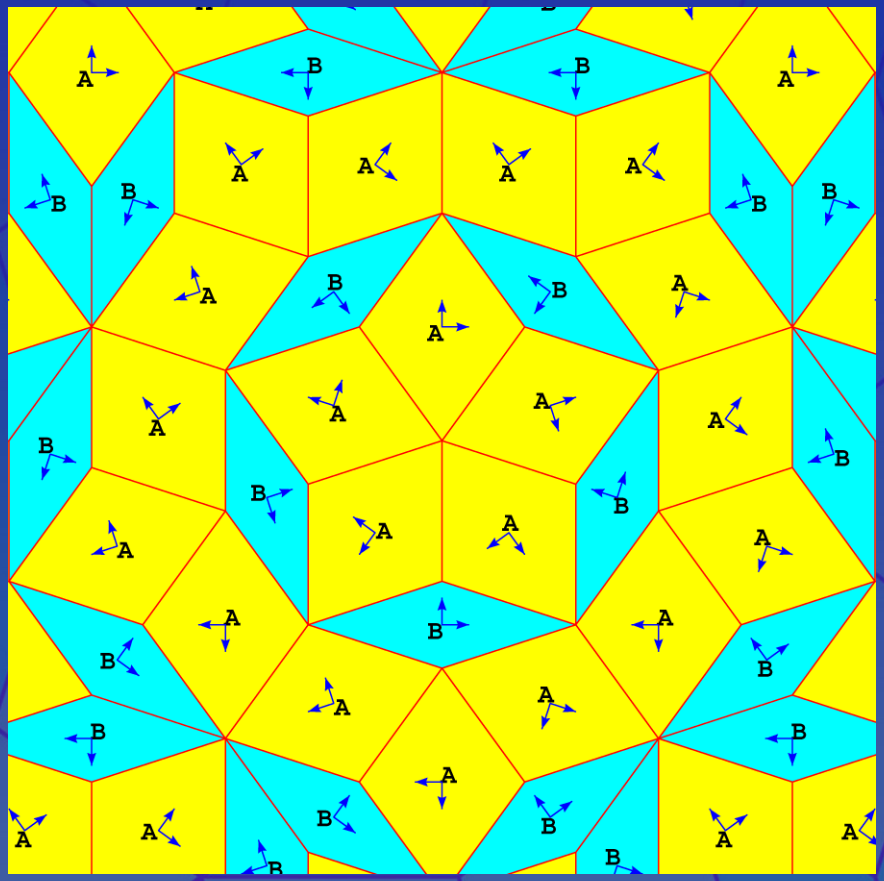


$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



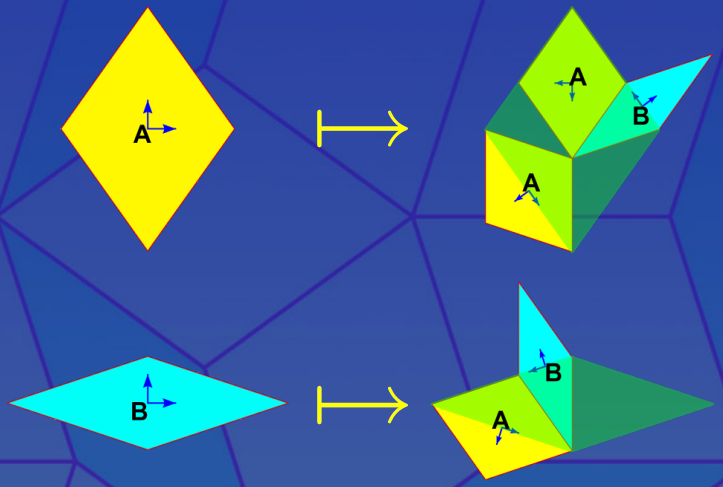
Penrose Tiling: Definition by Production Rules

Iteration *N*

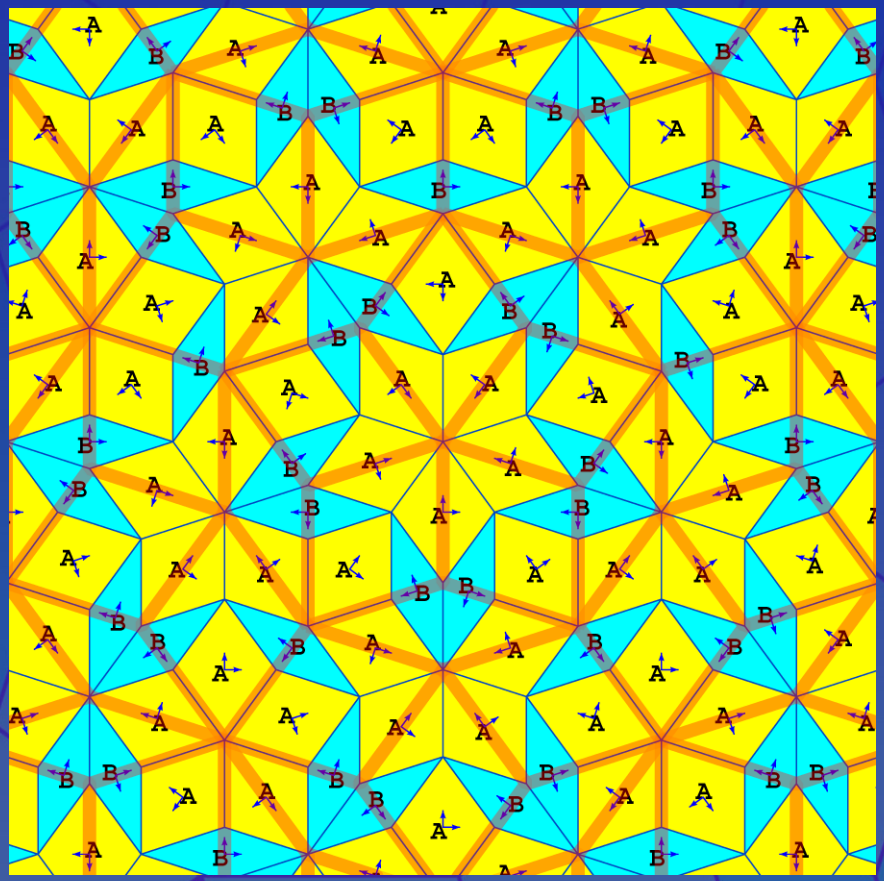


Penrose Tiling: Definition by Production Rules

$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



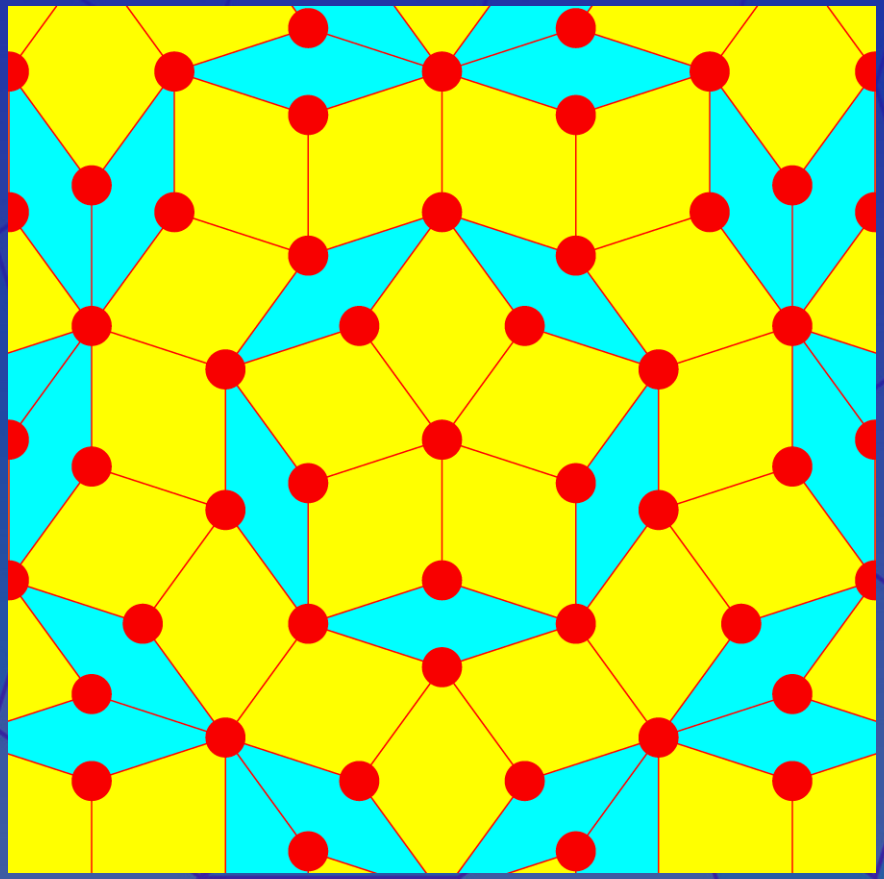
Iteration *N+1*



Ref: *"Tilings and Patterns"* by B. Grunbaum and G.C. Shephard

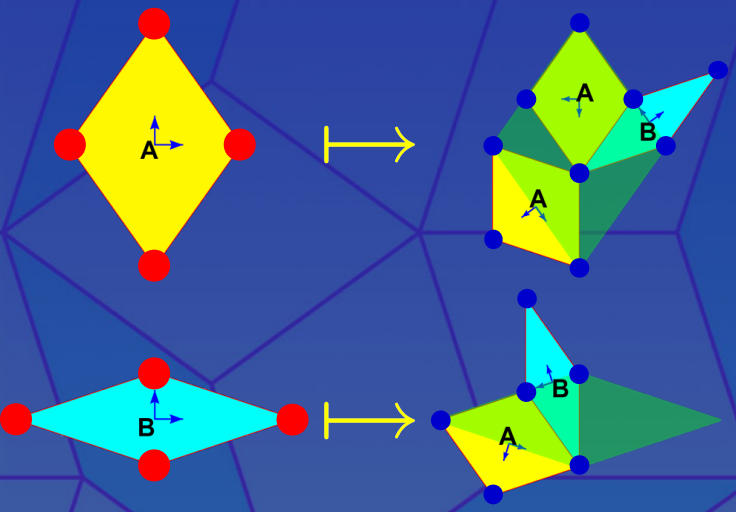
Penrose Tiling: Vertices

Iteration *N*

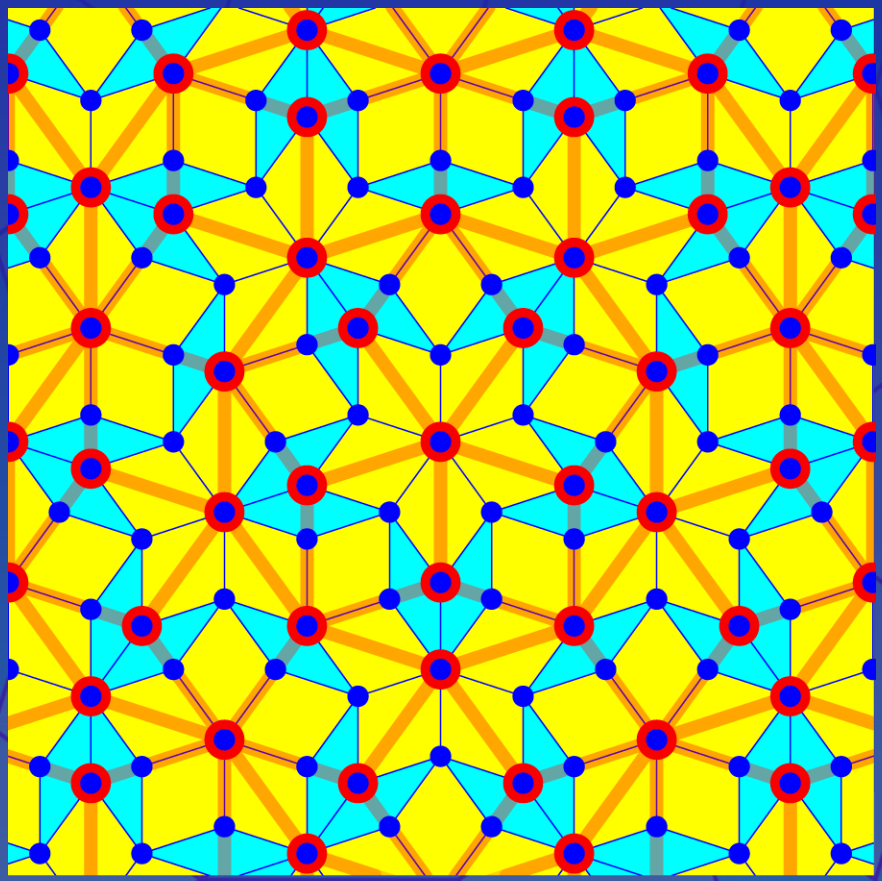


Penrose Tiling: Vertices

$$\mathcal{P}_{Penrose} := \begin{cases} A \mapsto \{A, B, A\} \\ B \mapsto \{A, B\} \end{cases}$$



Iteration *N+1*

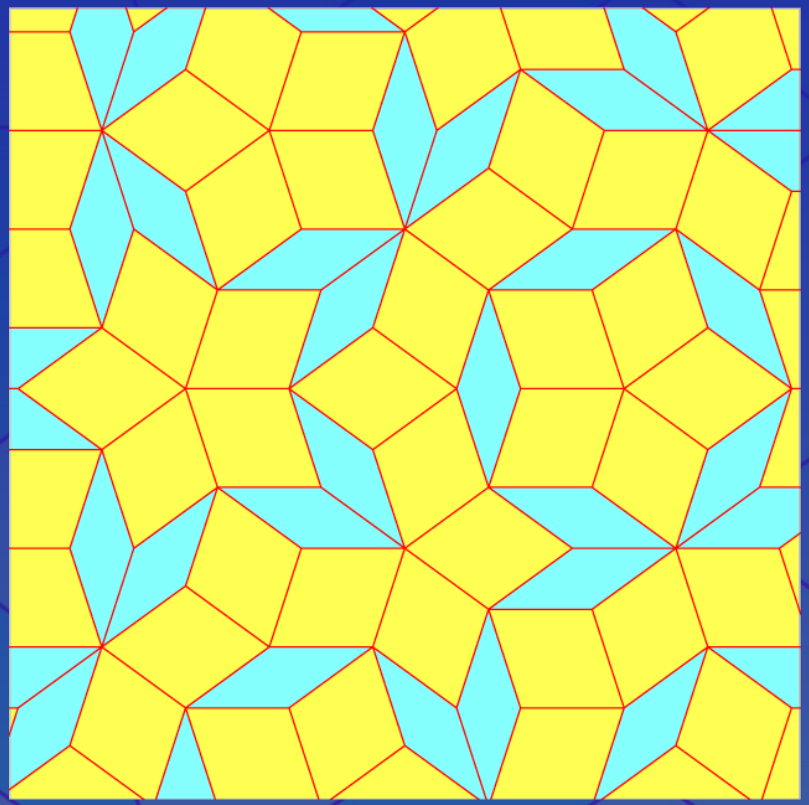


Sampling System Fundamentals

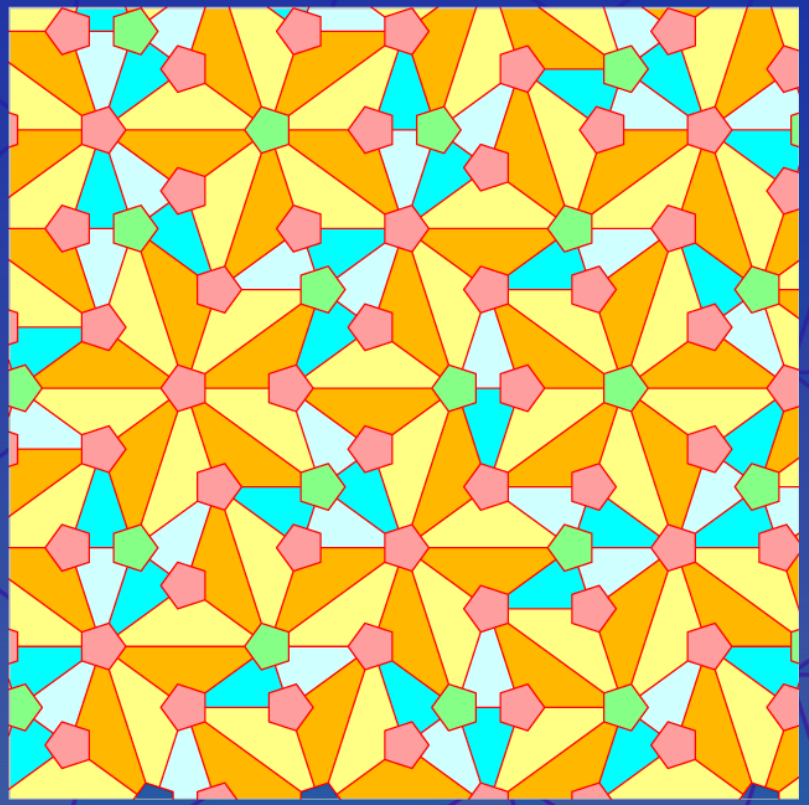
Sampling System Fundamentals

- Extension to Penrose Tiling
- Fibonacci Number System
- Adaptive Subdivision
- Corrective Vectors Lookup Table

Extension of Penrose Tiling

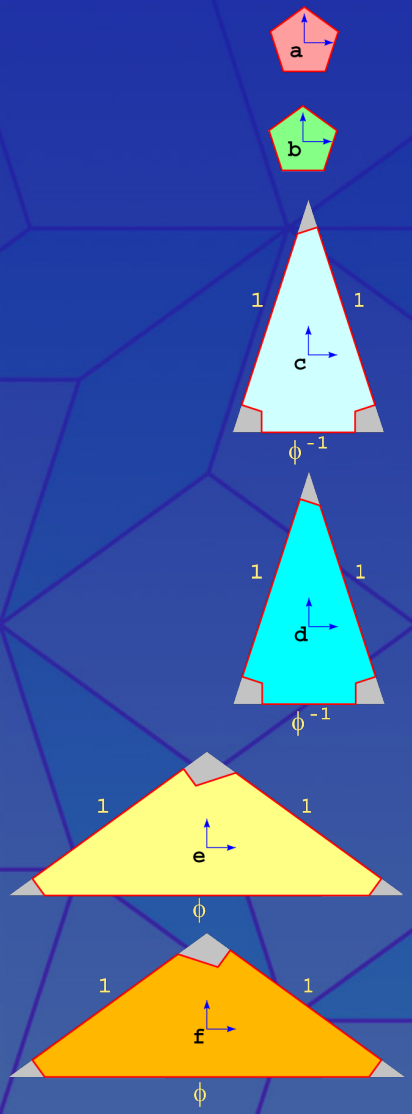


Original



Our Extension

Extension of Penrose Tiling



a

b

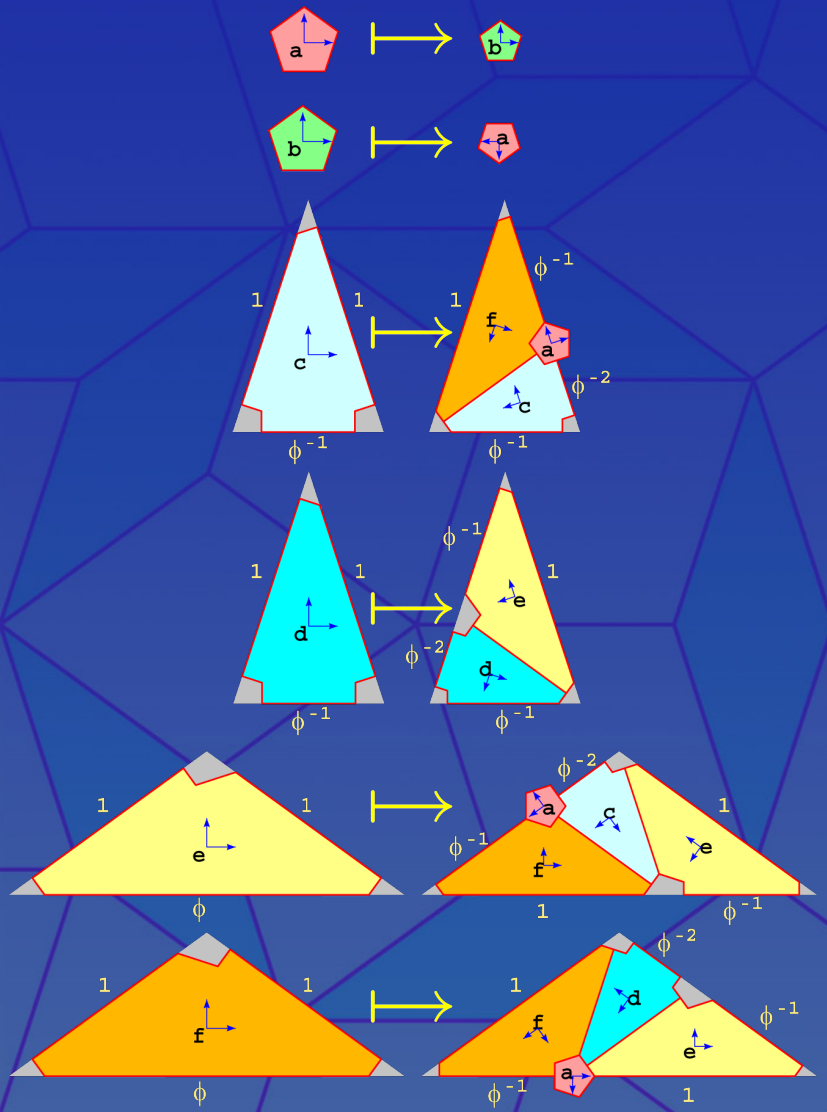
c

d

e

f

Extension of Penrose Tiling

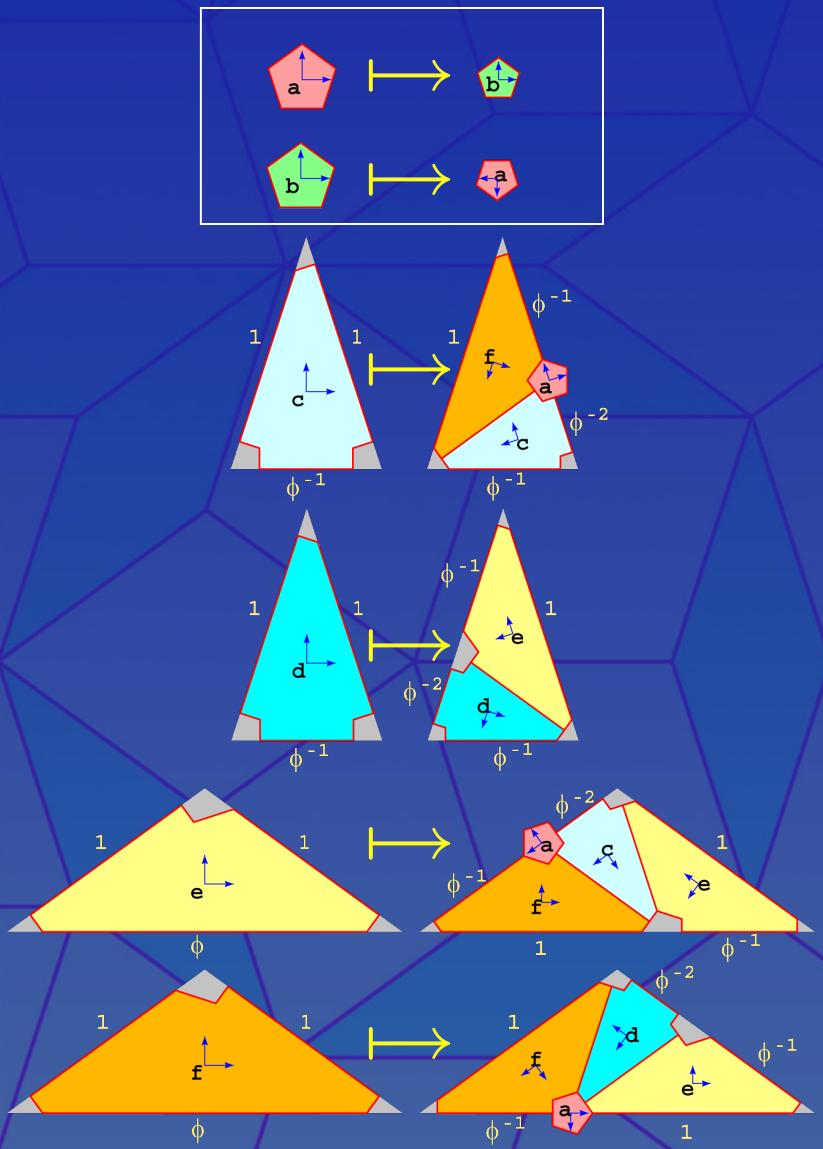


$$\mathcal{P} := \begin{cases} a \mapsto \{b\} \\ b \mapsto \{a\} \\ c \mapsto \{f, c, a\} \\ d \mapsto \{e, d\} \\ e \mapsto \{f, c, e, a\} \\ f \mapsto \{e, d, f, a\} \end{cases}$$

$$\frac{A_{new}}{A_{old}} = \phi^2$$

$$\phi = \frac{1+\sqrt{5}}{2} \text{ (Golden Ratio)}$$

Extension of Penrose Tiling

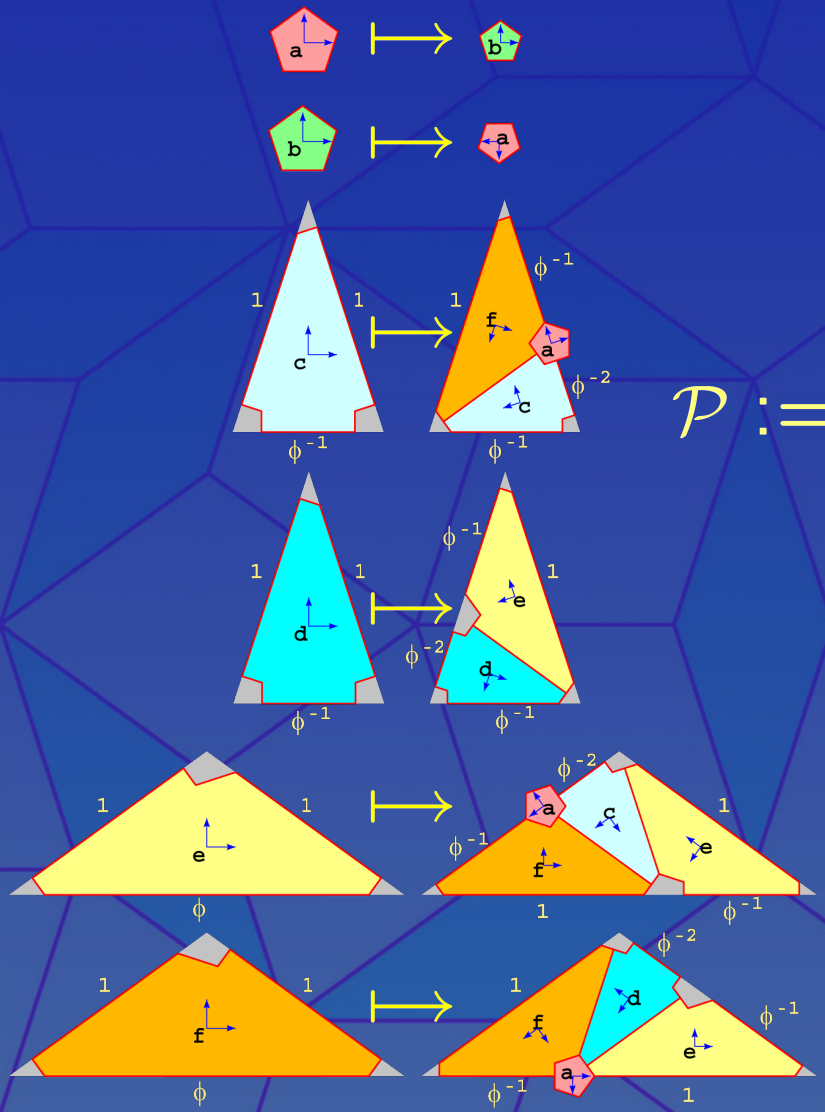


$$\mathcal{P} := \begin{cases} a \mapsto \{b\} \\ b \mapsto \{a\} \\ c \mapsto \{f, c, a\} \\ d \mapsto \{e, d\} \\ e \mapsto \{f, c, e, a\} \\ f \mapsto \{e, d, f, a\} \end{cases}$$

$$\frac{A_{new}}{A_{old}} = \phi^2$$

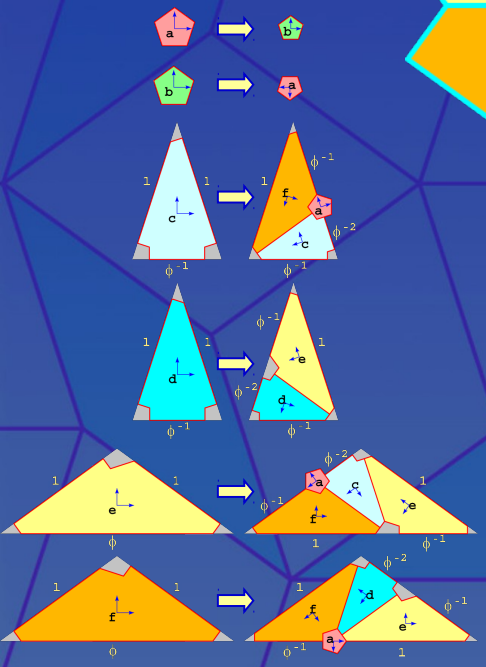
$$\phi = \frac{1+\sqrt{5}}{2} \text{ (Golden Ratio)}$$

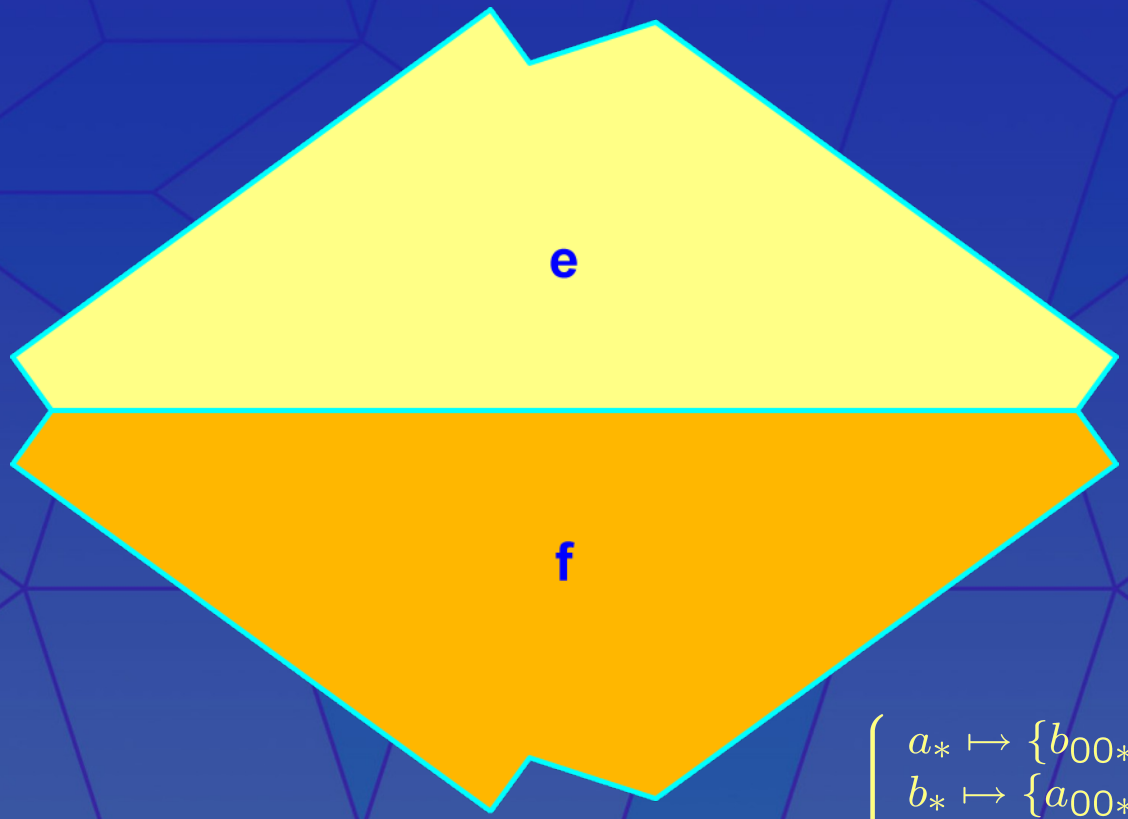
Fibonacci Number System



$$\mathcal{P} := \begin{cases} a_* \mapsto \{b_{00*}\} \\ b_* \mapsto \{a_{00*}\} \\ c_* \mapsto \{f_{00*}, c_{10*}, a_{10*}\} \\ d_* \mapsto \{e_{00*}, d_{10*}\} \\ e_* \mapsto \{f_{00*}, c_{10*}, e_{01*}, a_{10*}\} \\ f_* \mapsto \{e_{00*}, d_{10*}, f_{01*}, a_{01*}\} \end{cases}$$

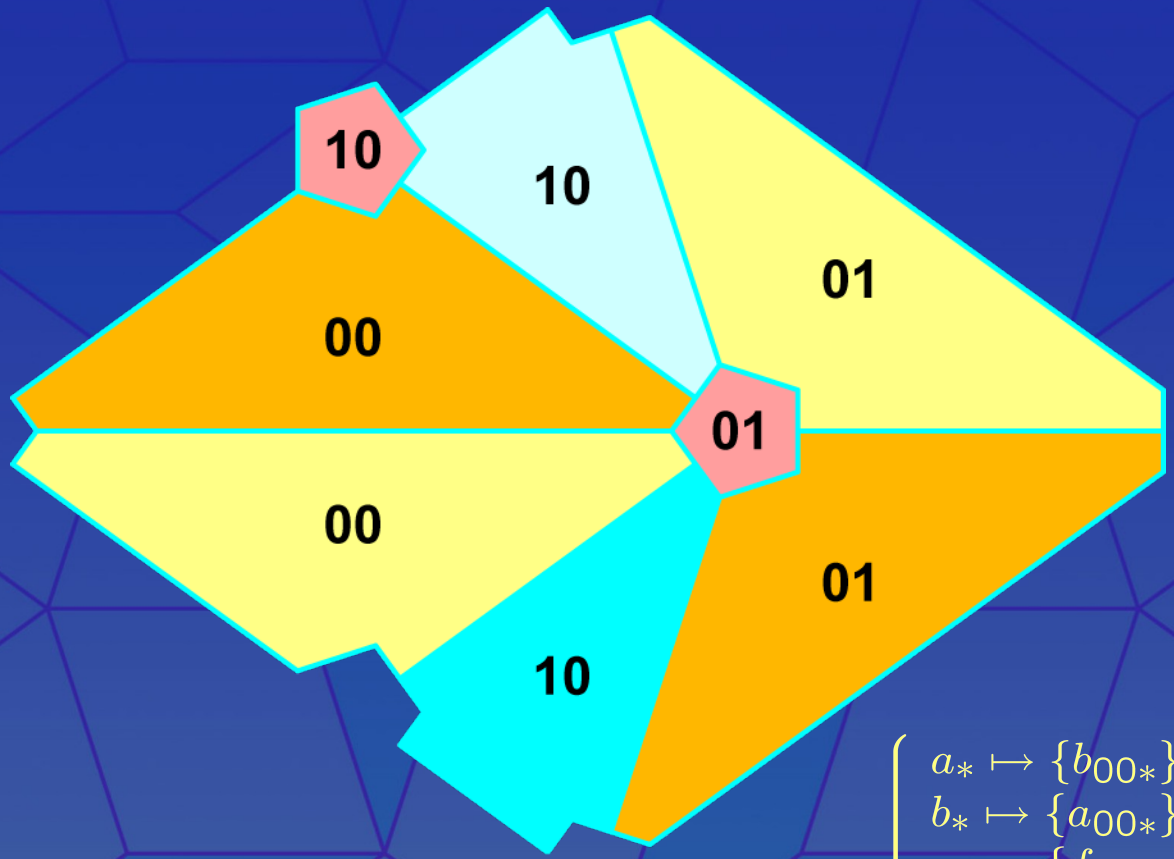
Fibonacci Number System

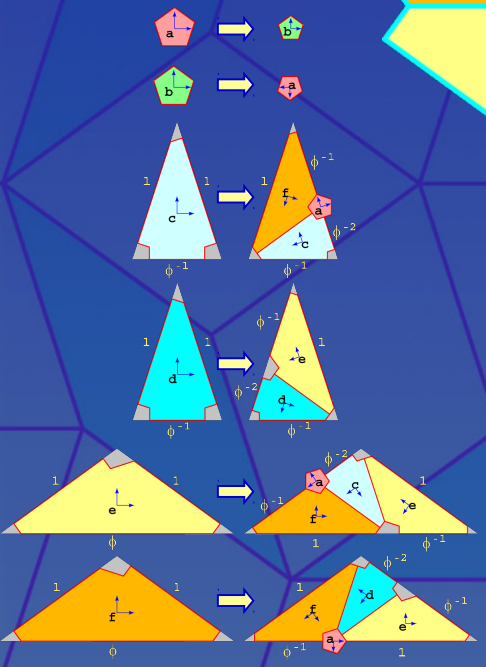




$$\mathcal{P} := \begin{cases} a_* \mapsto \{b_{00*}\} \\ b_* \mapsto \{a_{00*}\} \\ c_* \mapsto \{f_{00*}, c_{10*}, a_{10*}\} \\ d_* \mapsto \{e_{00*}, d_{10*}\} \\ e_* \mapsto \{f_{00*}, c_{10*}, e_{01*}, a_{10*}\} \\ f_* \mapsto \{e_{00*}, d_{10*}, f_{01*}, a_{01*}\} \end{cases}$$

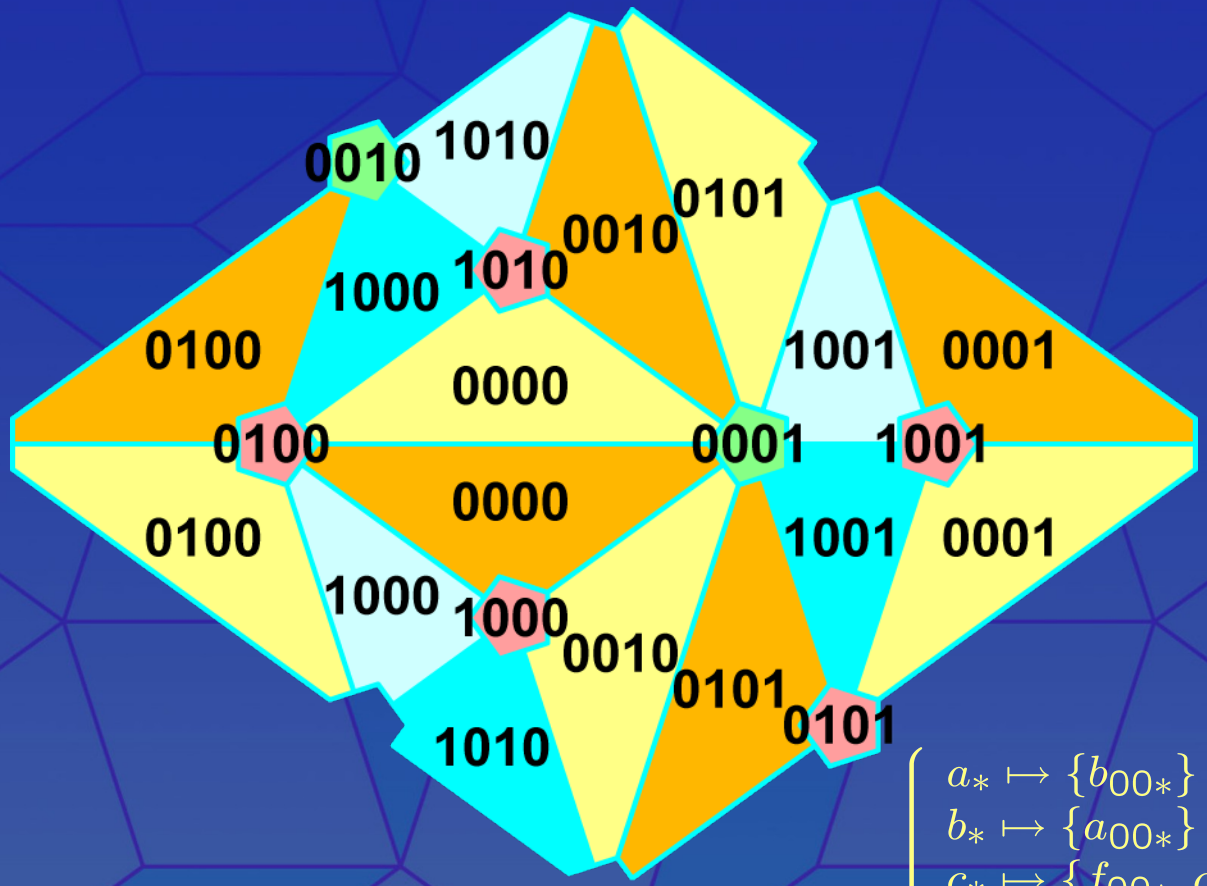
Fibonacci Number System

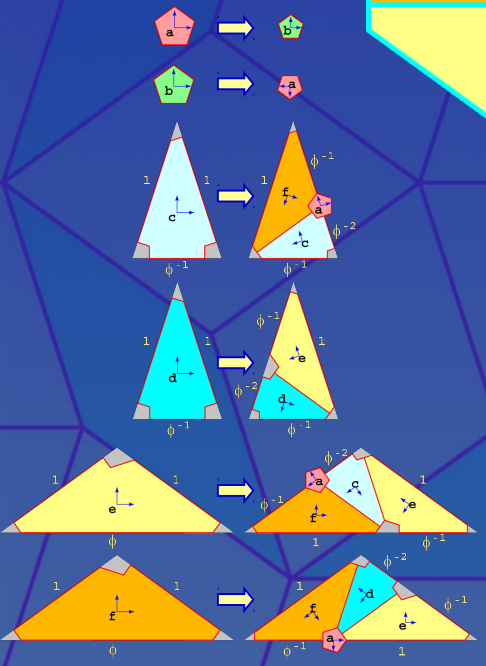




$$\mathcal{P} := \begin{cases} a_* \mapsto \{b_{00*}\} \\ b_* \mapsto \{a_{00*}\} \\ c_* \mapsto \{f_{00*}, c_{10*}, a_{10*}\} \\ d_* \mapsto \{e_{00*}, d_{10*}\} \\ e_* \mapsto \{f_{00*}, c_{10*}, e_{01*}, a_{10*}\} \\ f_* \mapsto \{e_{00*}, d_{10*}, f_{01*}, a_{01*}\} \end{cases}$$

Fibonacci Number System





$$\mathcal{P} := \begin{cases} a_* \mapsto \{b_{00*}\} \\ b_* \mapsto \{a_{00*}\} \\ c_* \mapsto \{f_{00*}, c_{10*}, a_{10*}\} \\ d_* \mapsto \{e_{00*}, d_{10*}\} \\ e_* \mapsto \{f_{00*}, c_{10*}, e_{01*}, a_{10*}\} \\ f_* \mapsto \{e_{00*}, d_{10*}, f_{01*}, a_{01*}\} \end{cases}$$

Fibonacci Number System

Ref: *"The Art of Computer Programming, Vol. 1,"* by D.E. Knuth

- Binary Number System:

$$n = (a_m a_{m-1} \ldots a_1 a_0)_2 \iff n = \sum_{j=0}^{m} a_j 2^j.$$

- Fibonacci Number System:

$$n = (b_m b_{m-1} \ldots b_3 b_2)_F \iff n = \sum_{j=2}^{m} b_j F_j.$$

- Fibonacci Numbers:

$$F_i = F_{i-1} + F_{i-2} \quad \{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots\}$$

Fibonacci Number System

Binary Number System

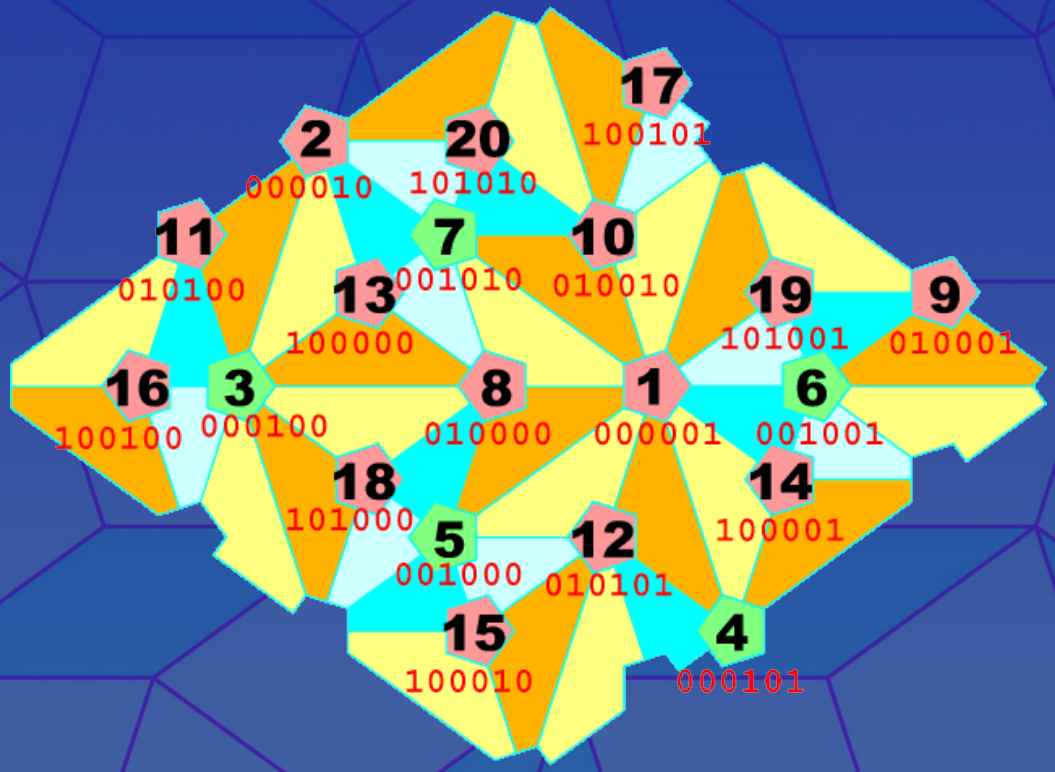
N_{10}	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

Fibonacci Number System

N_{10}	F_6 (8)	F_5 (5)	F_4 (3)	F_3 (2)	F_2 (1)
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	1	0	1
5	0	1	0	0	0
6	0	1	0	0	1
7	0	1	0	1	0
8	1	0	0	0	0
9	1	0	0	0	1
10	1	0	0	1	0
11	1	0	1	0	0
12	1	0	1	0	1

Fibonacci Number System

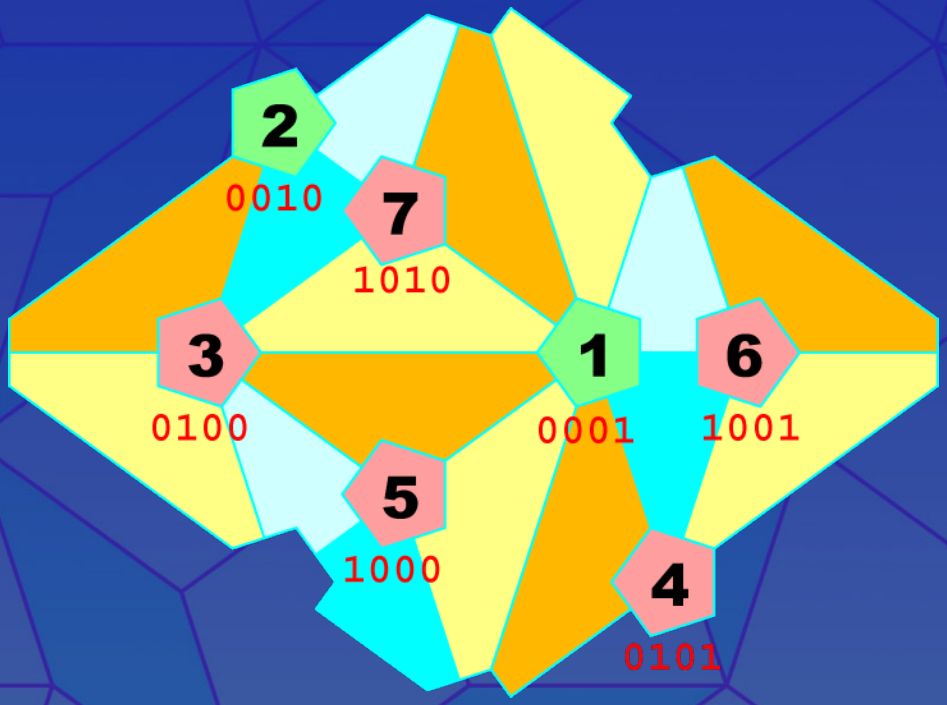
$$(00001)_F = 1, \quad (00010)_F = 2, \quad (00100)_F = 3,$$
$$(00101)_F = 4, \quad (01000)_F = 5, \quad (01001)_F = 6,$$
$$(01010)_F = 7, \quad (10000)_F = 8, \quad (10001)_F = 9,$$
$$(10010)_F = 10, \quad (10100)_F = 11, \quad (10101)_F = 12.$$



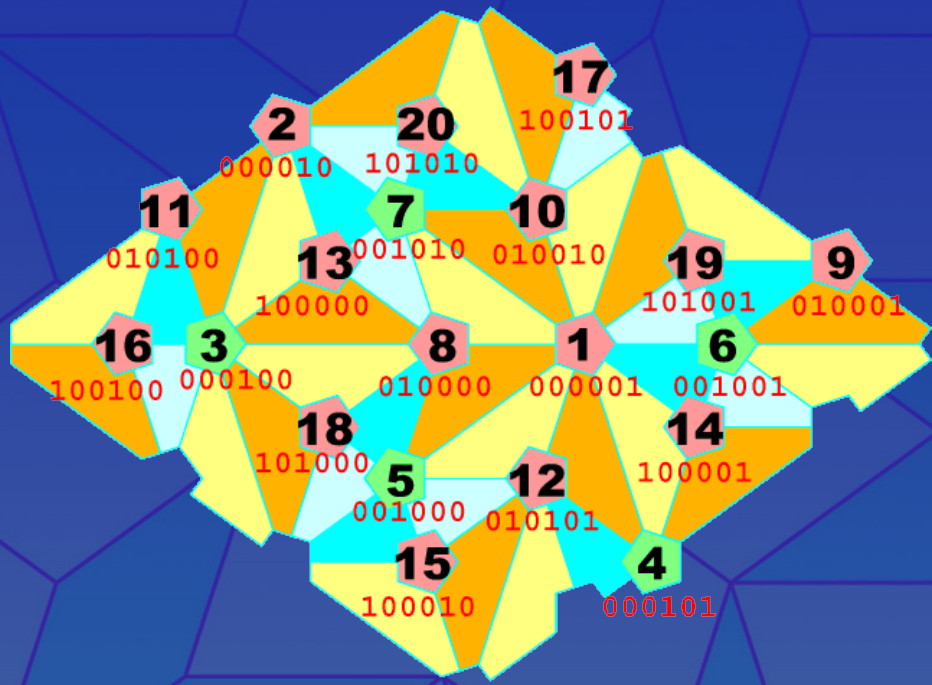
Pentagonal Tiles Only

Fibonacci Number System

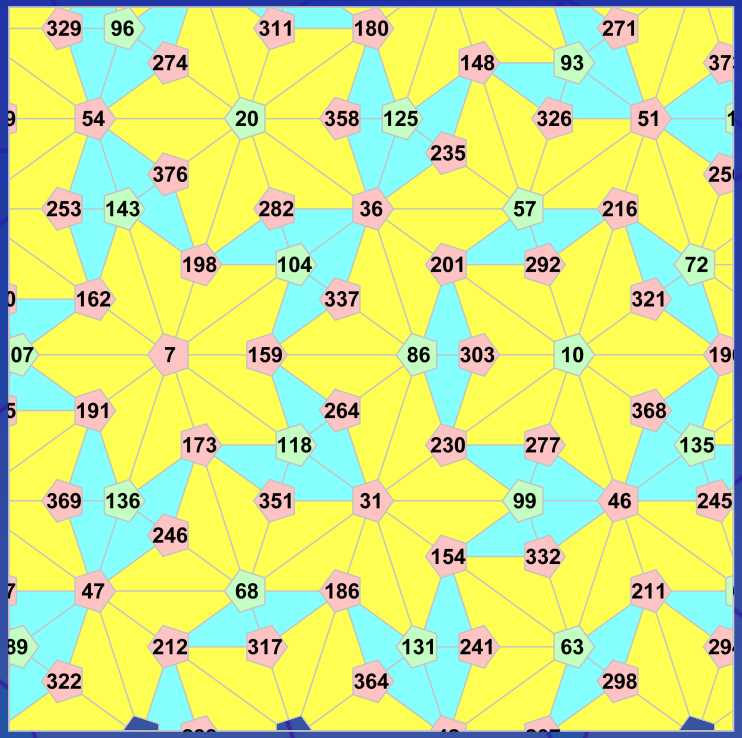
Iteration *4*



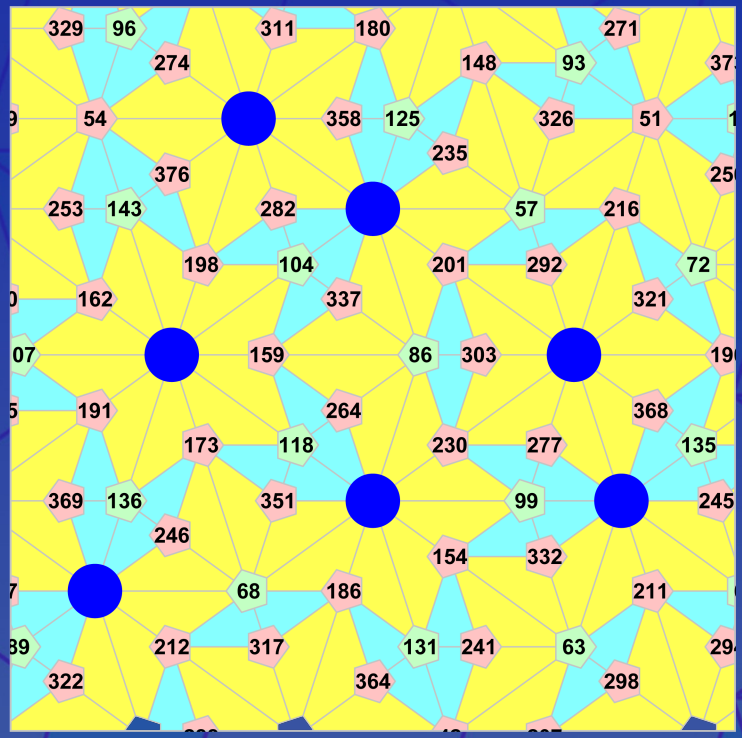
Iteration *5*



Fibonacci Number System Thresholding

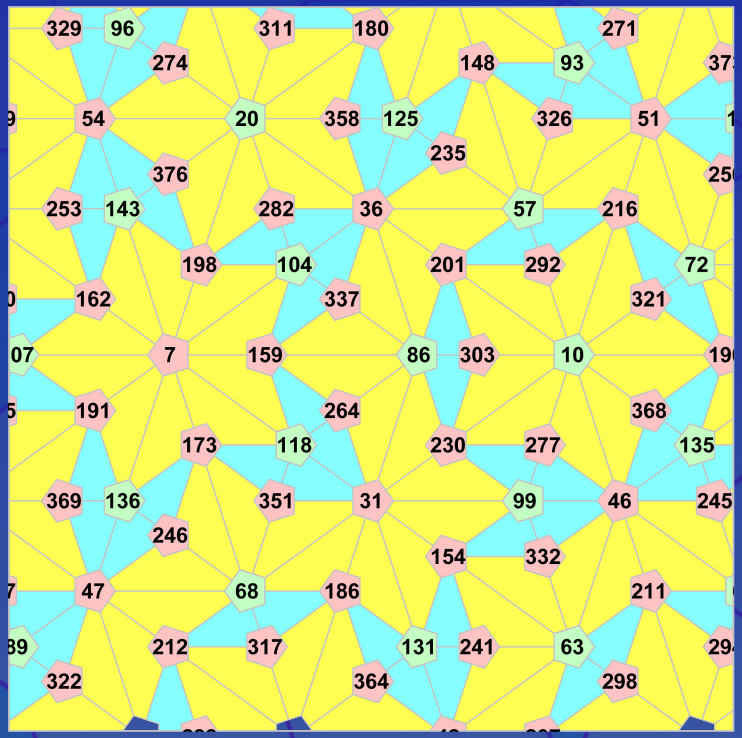


F-codes, in range [1..376]

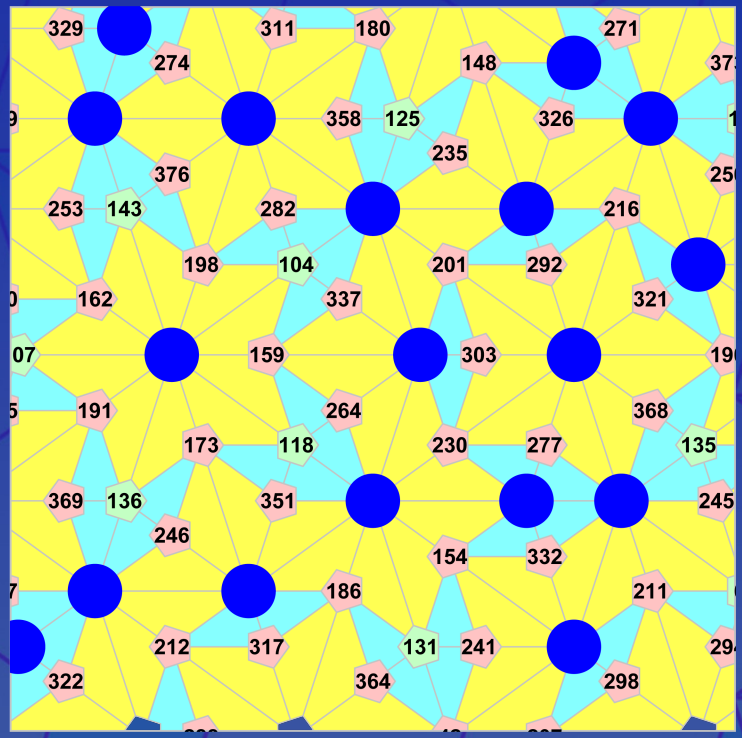


Thresholding at value = 50

Fibonacci Number System Thresholding

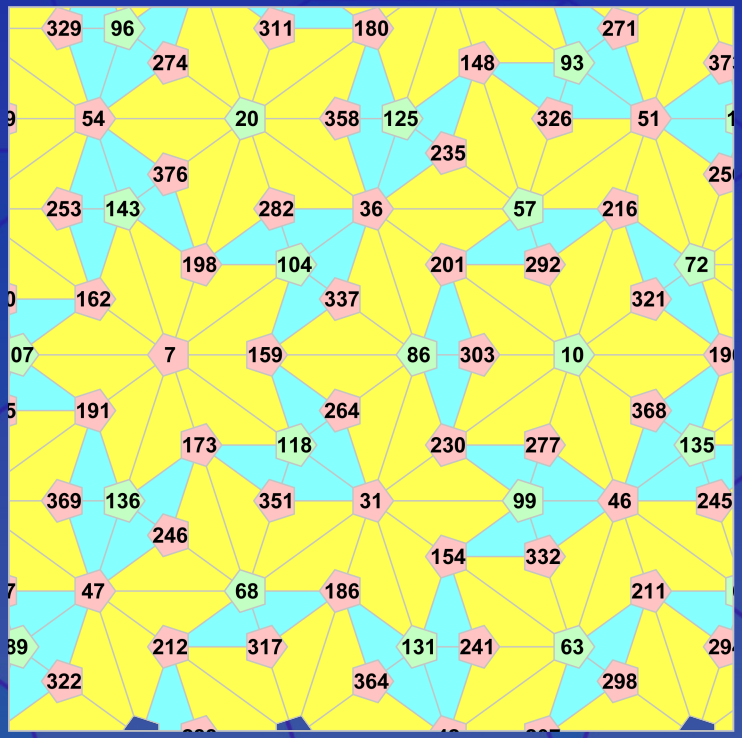


F-codes, in range [1..376]

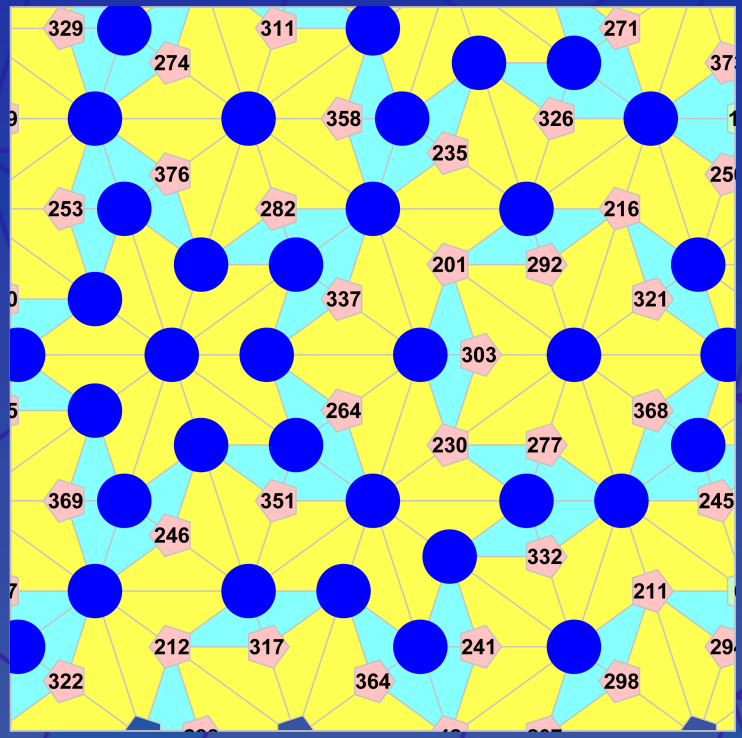


Thresholding at value = 100

Fibonacci Number System Thresholding



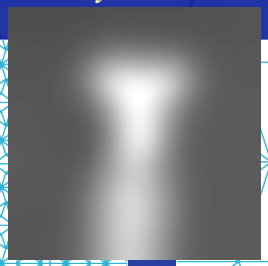
F-codes, in range [1..376]

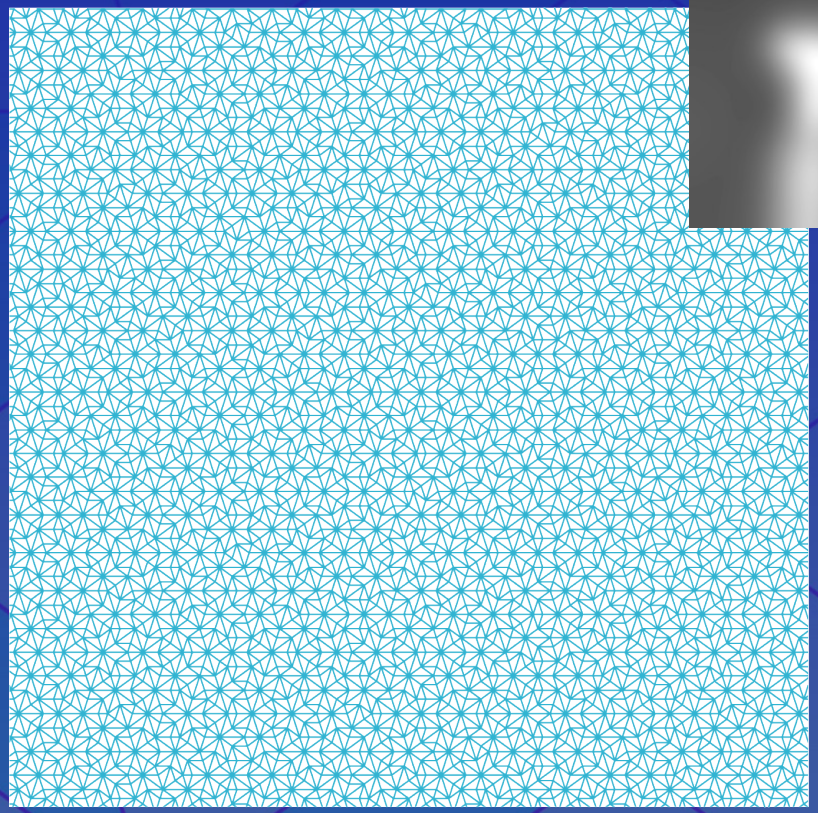


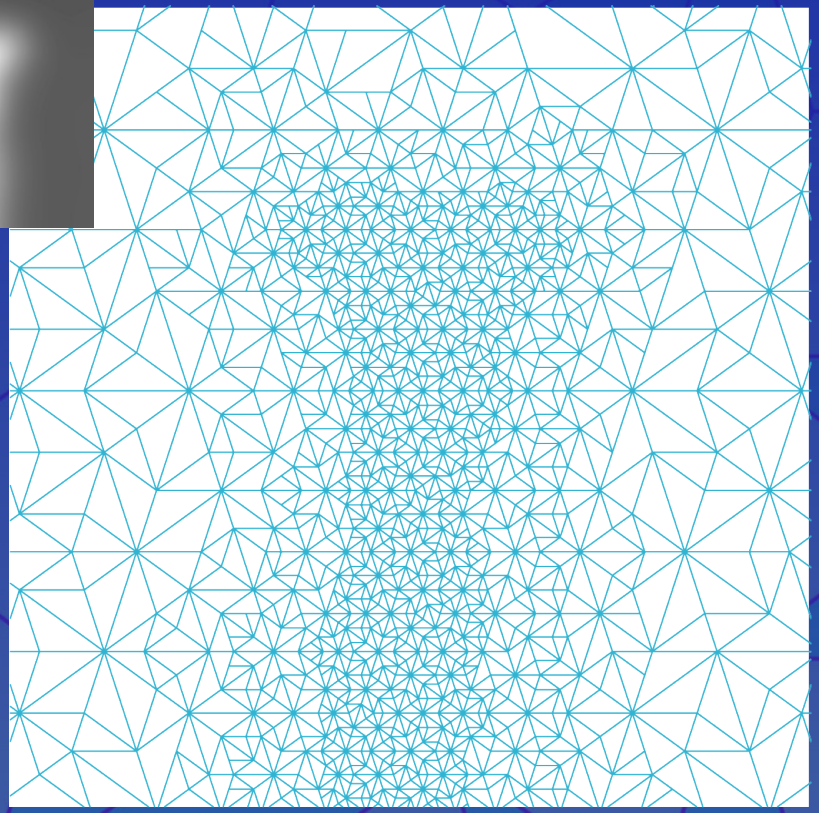
Thresholding at value = 200

Adaptive Subdivision

Importance Density Function







Non-adaptive

Adaptive

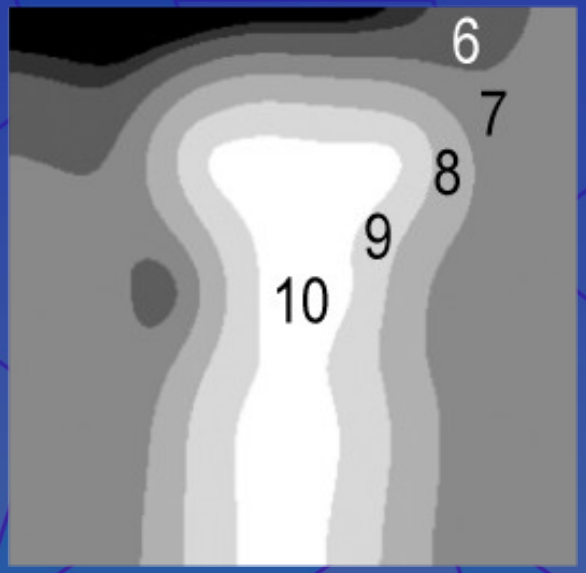
Adaptive Subdivision

$$\lambda(x) = \log_{\phi^2}(x) + C$$

$$\kappa = \lceil \max_{tile} \lambda(mag \cdot I(x,y)) \rceil$$

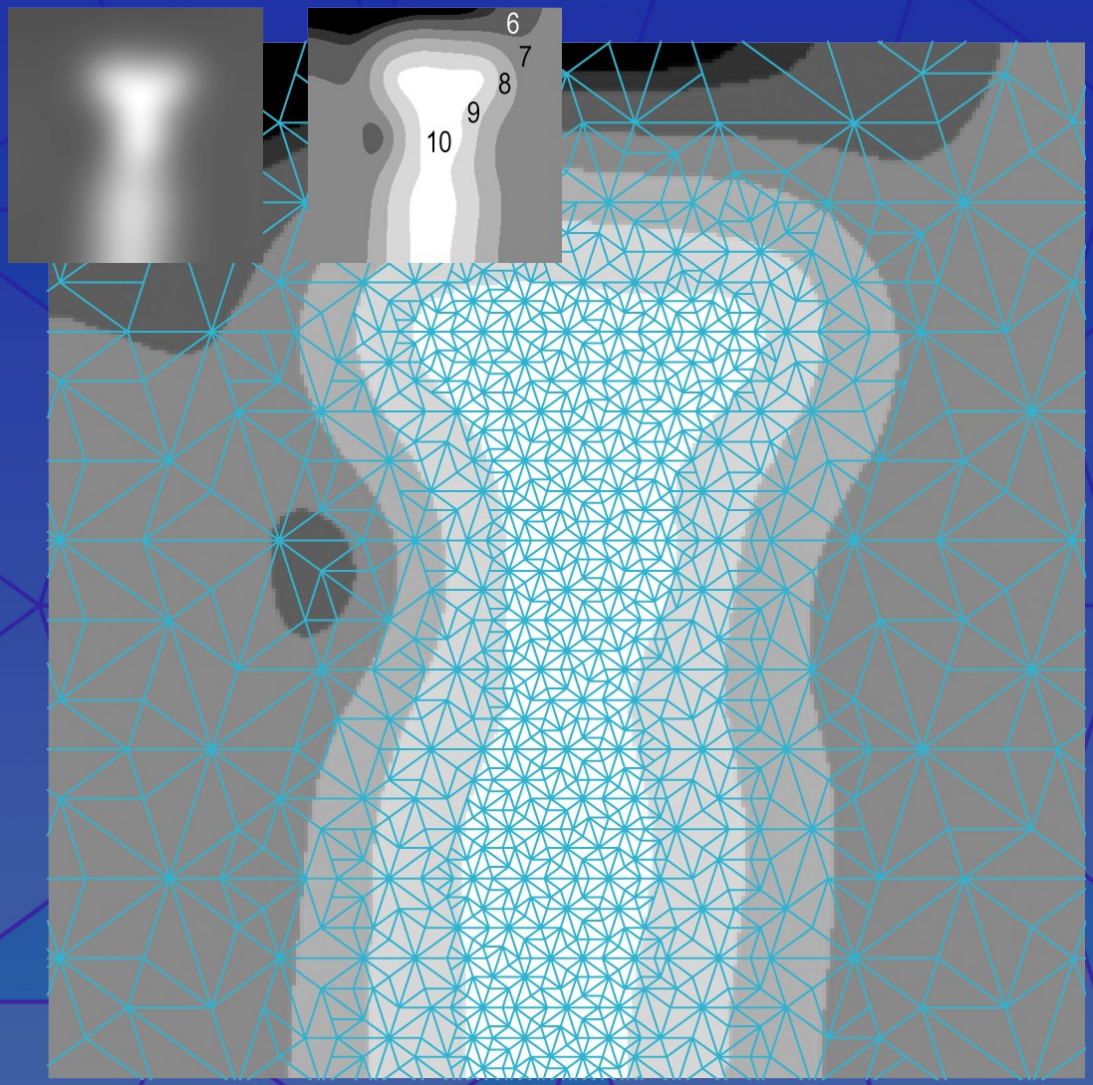
Importance Density Function

$I(x,y)$

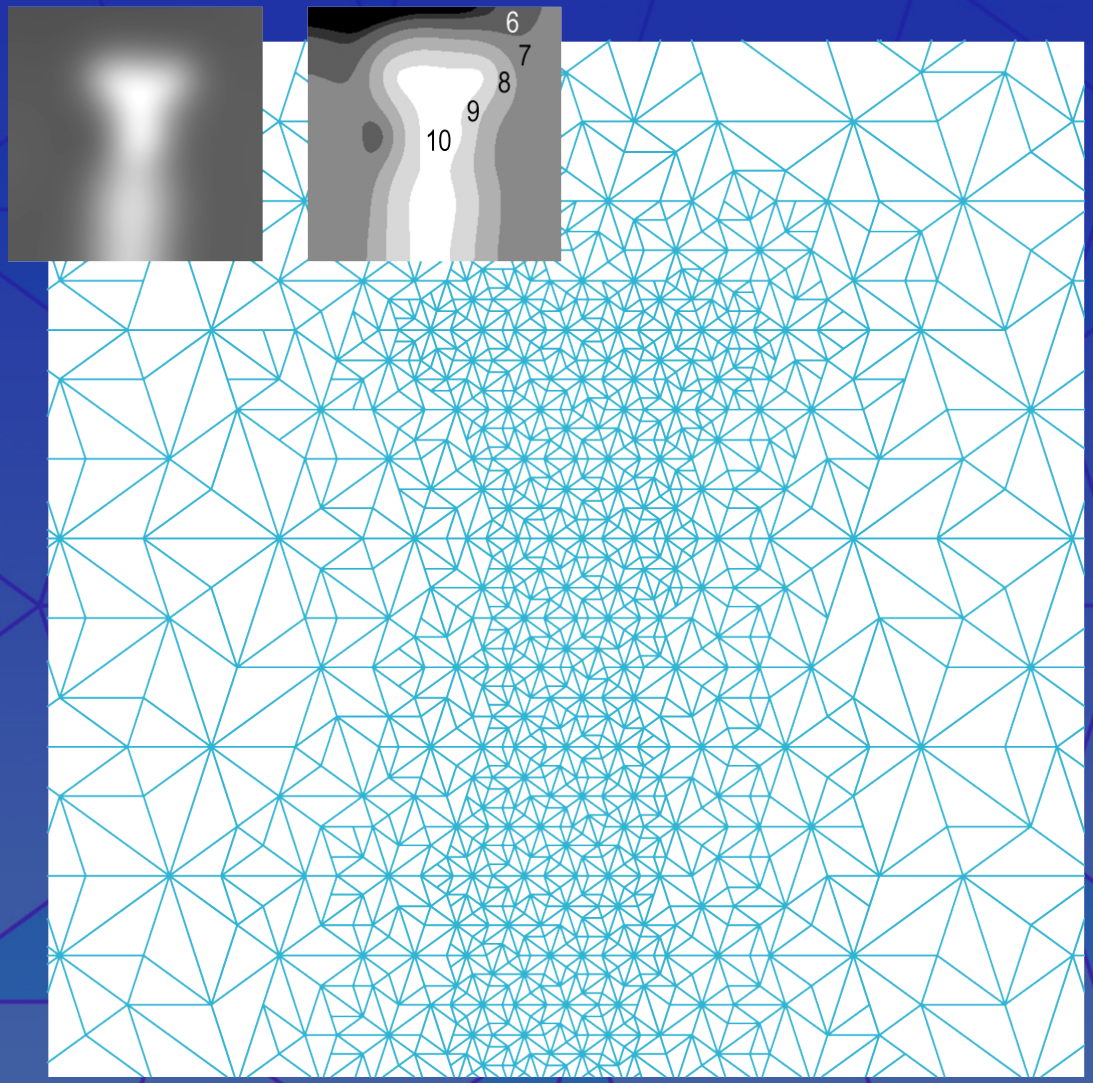


$\lceil \lambda(I(x,y)) \rceil$

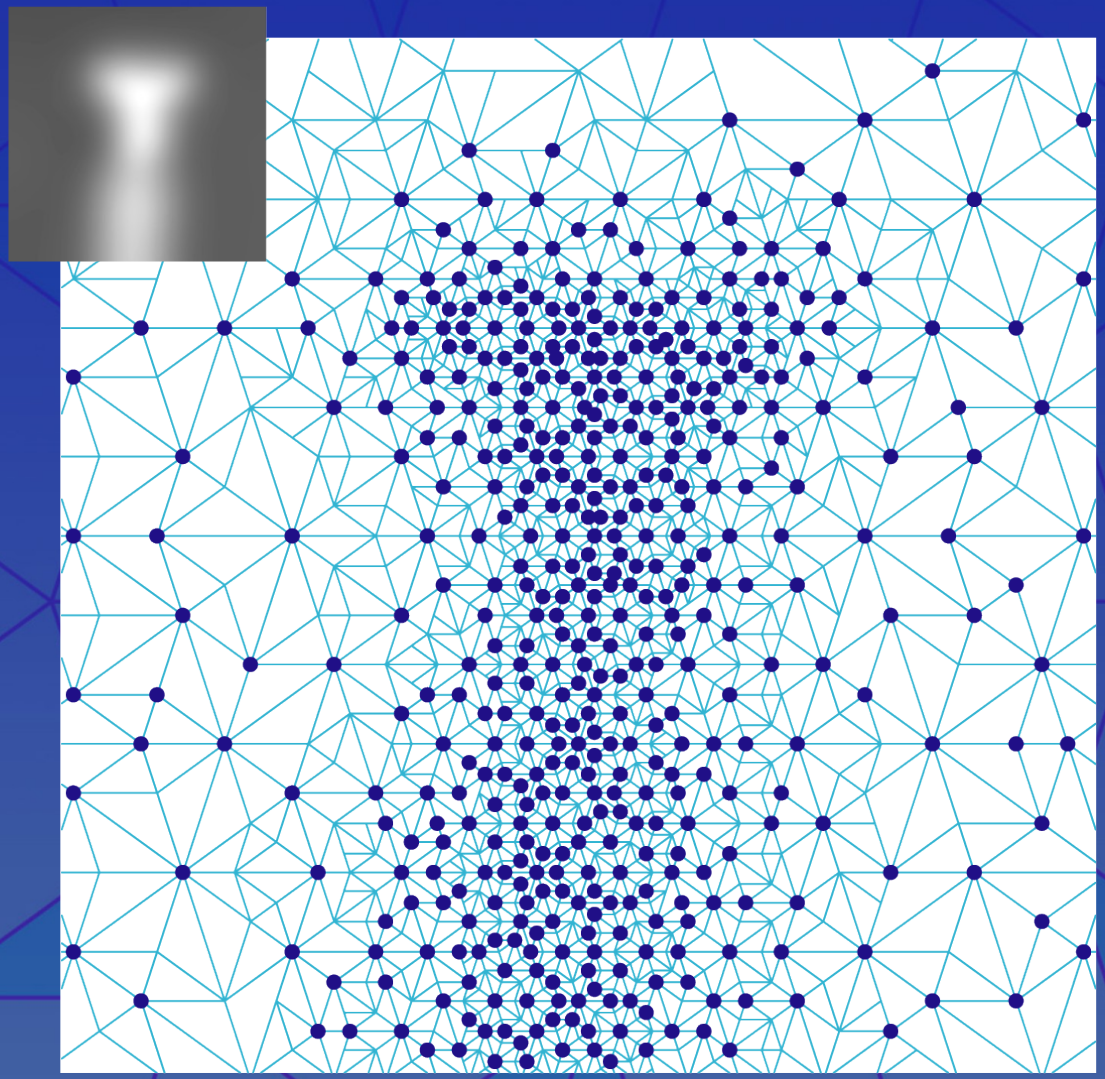
Adaptive Subdivision



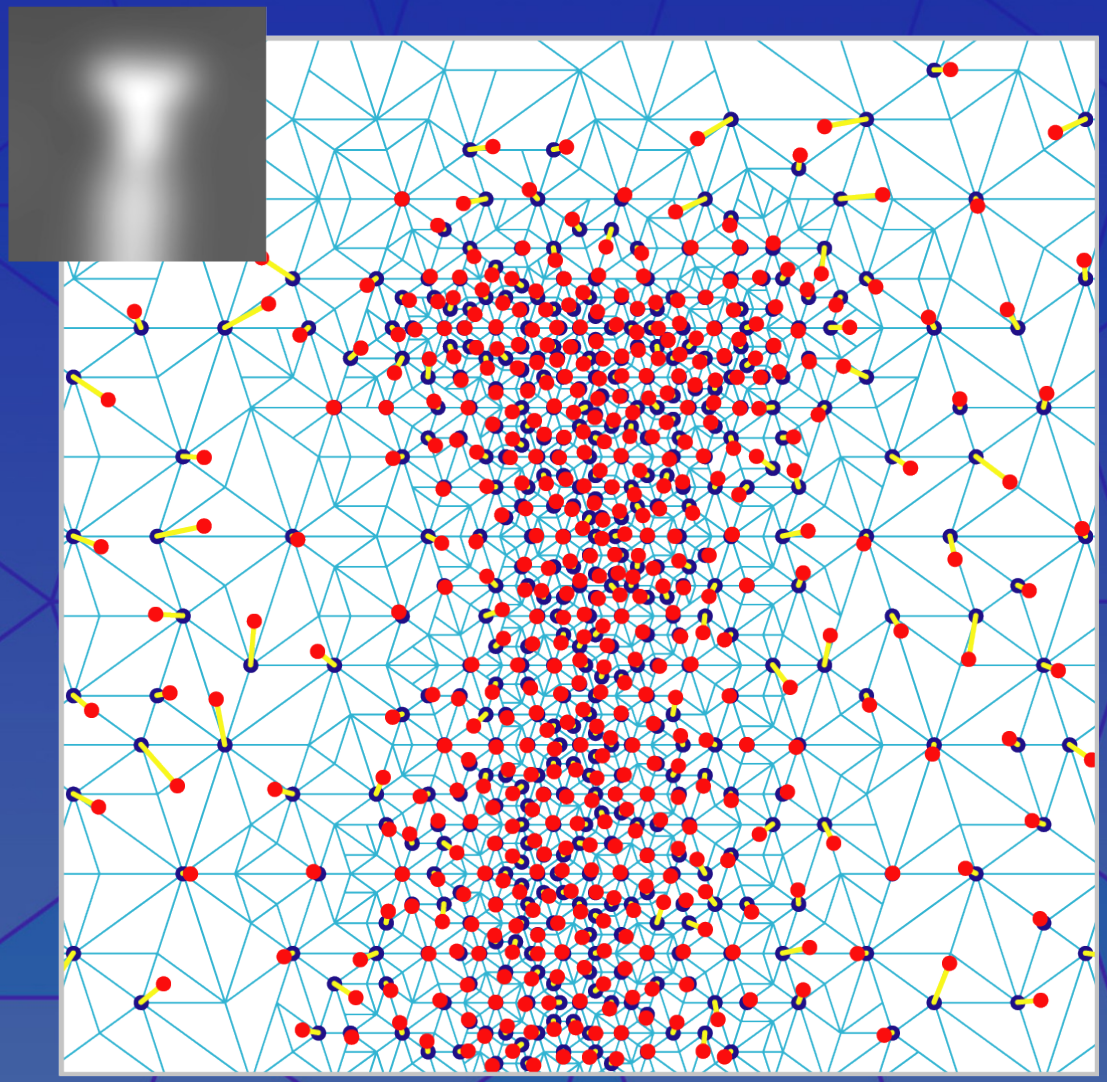
Adaptive Subdivision



Corrective Vectors Lookup table

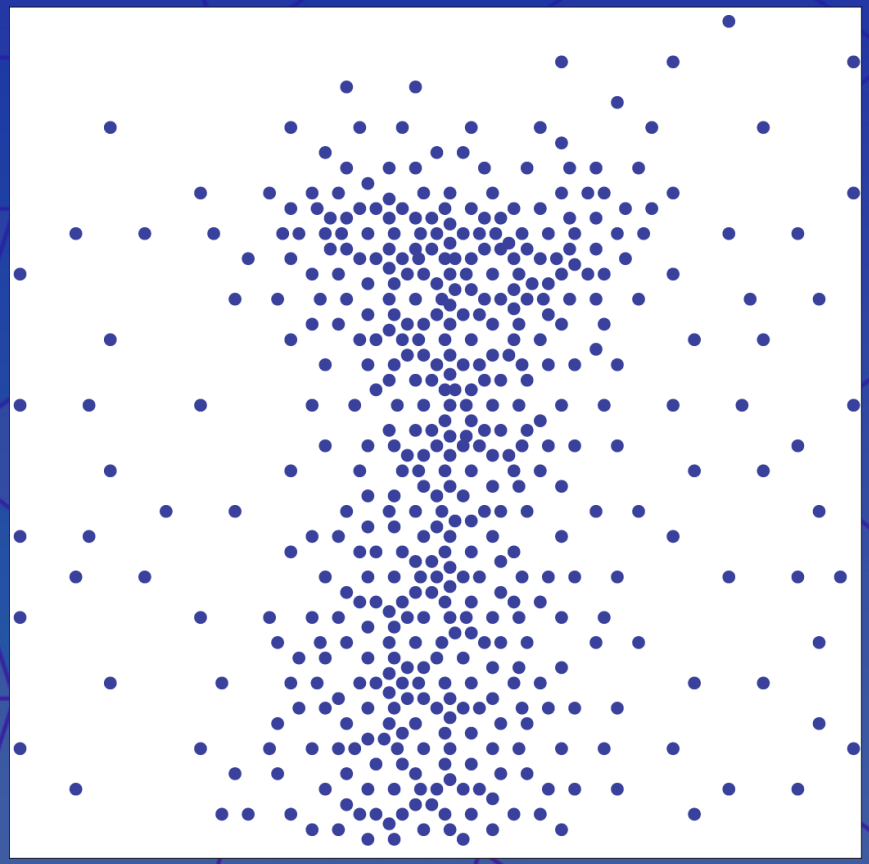


Corrective Vectors Lookup table

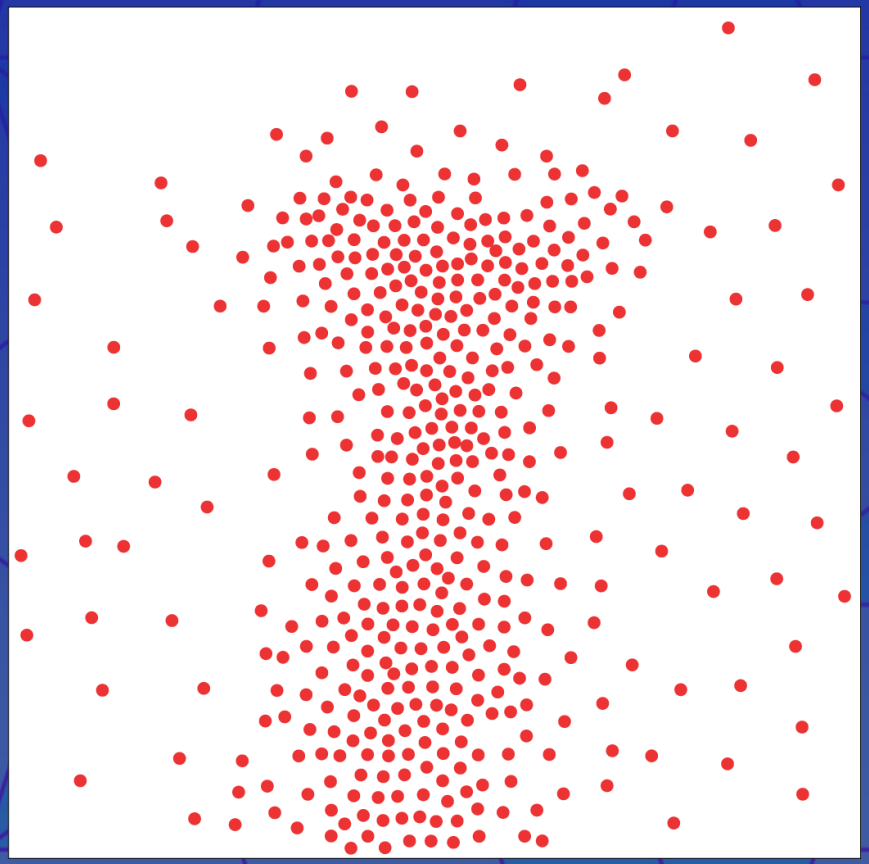


Corrective Vectors Lookup table

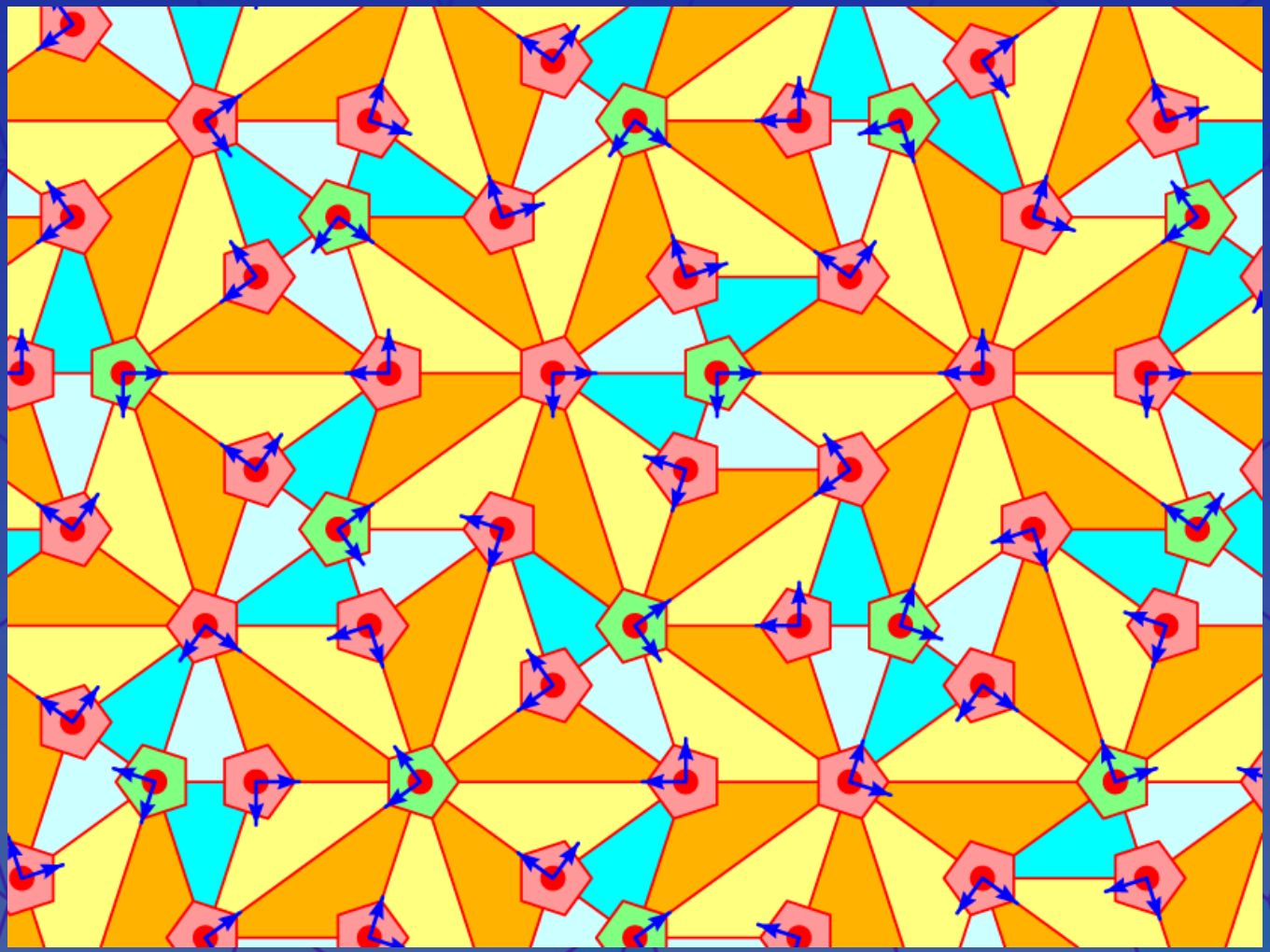
Before Correction



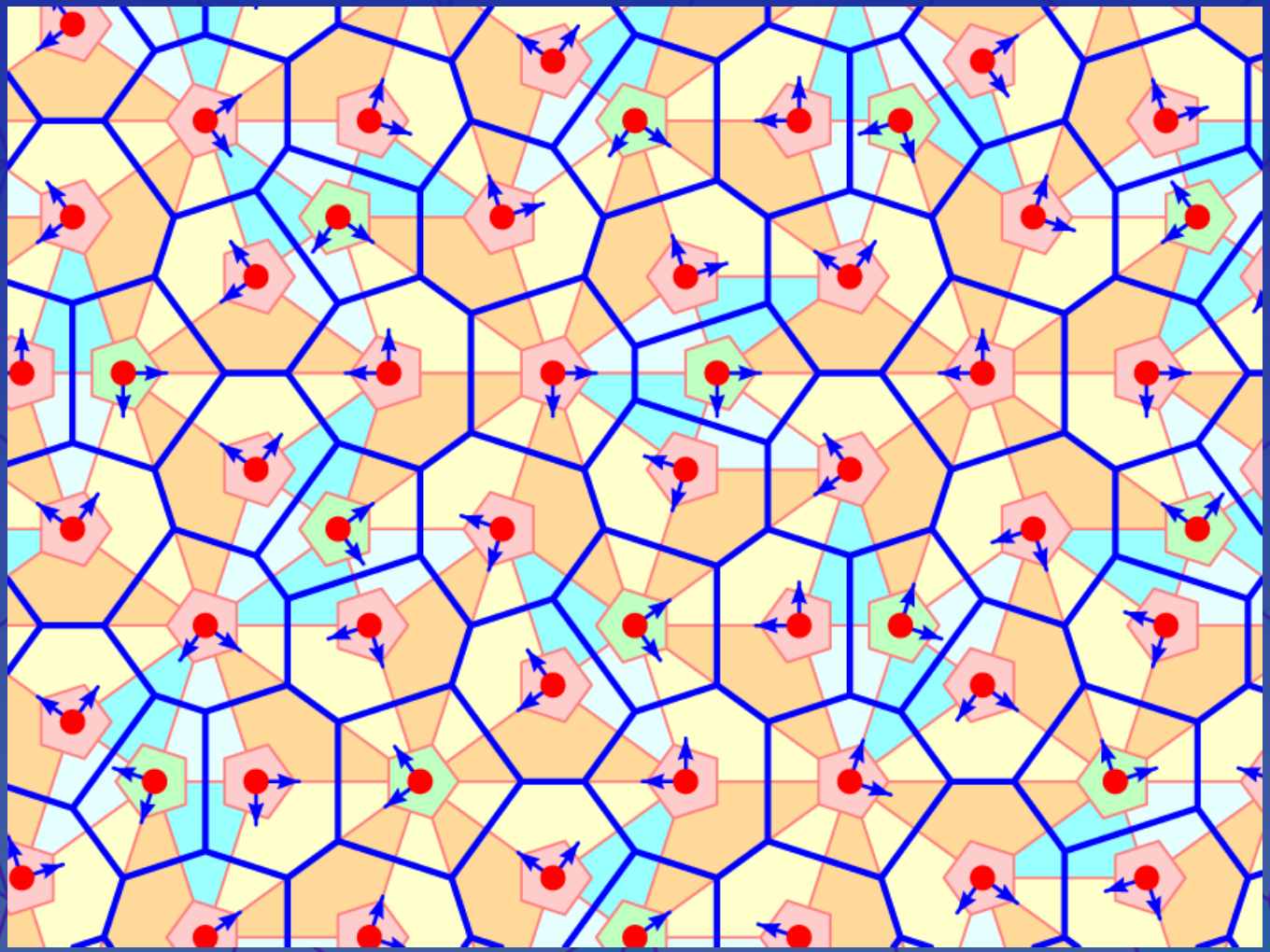
After Correction



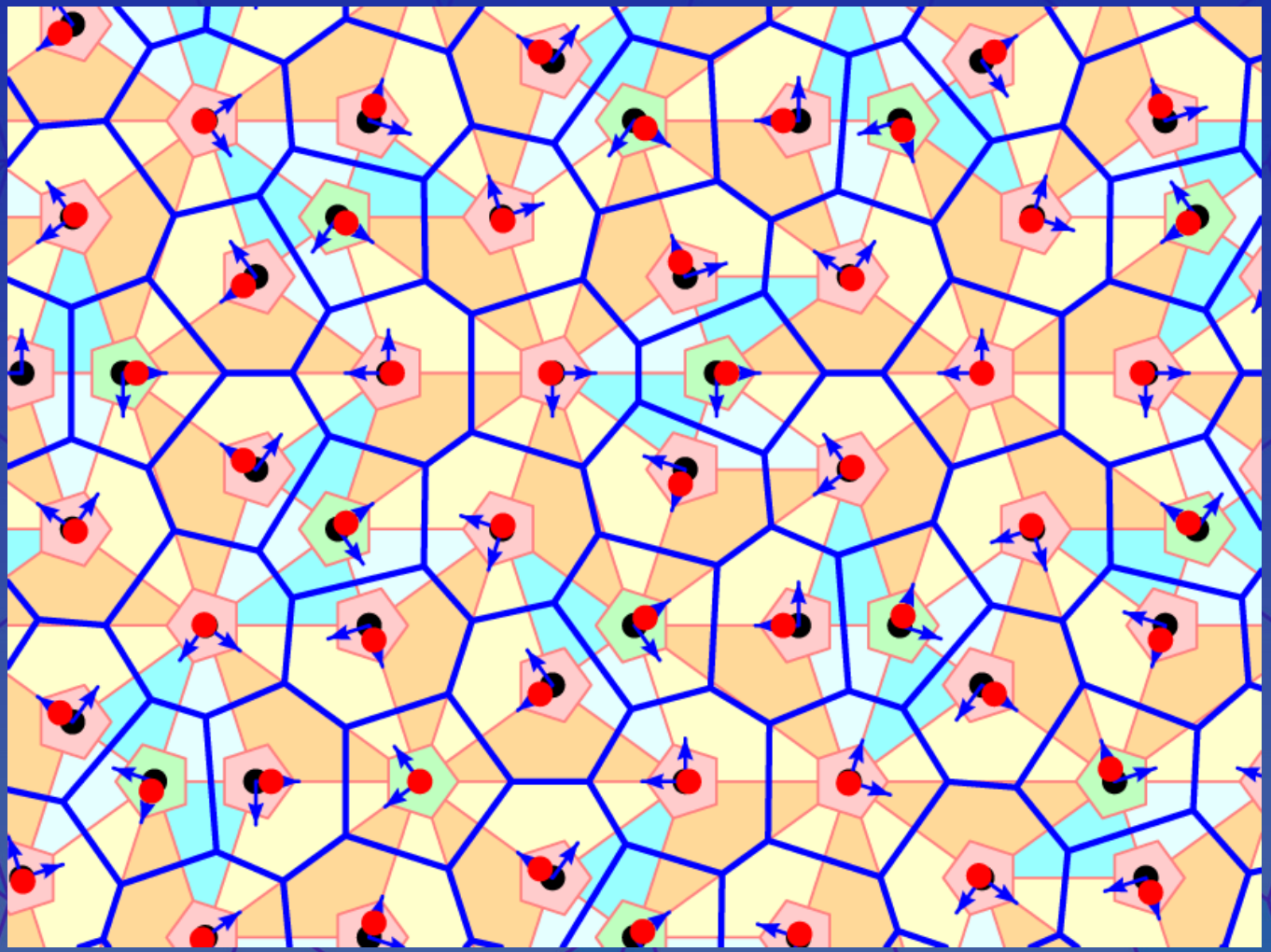
Offline Lloyd's Relaxation: init pts + basis frames



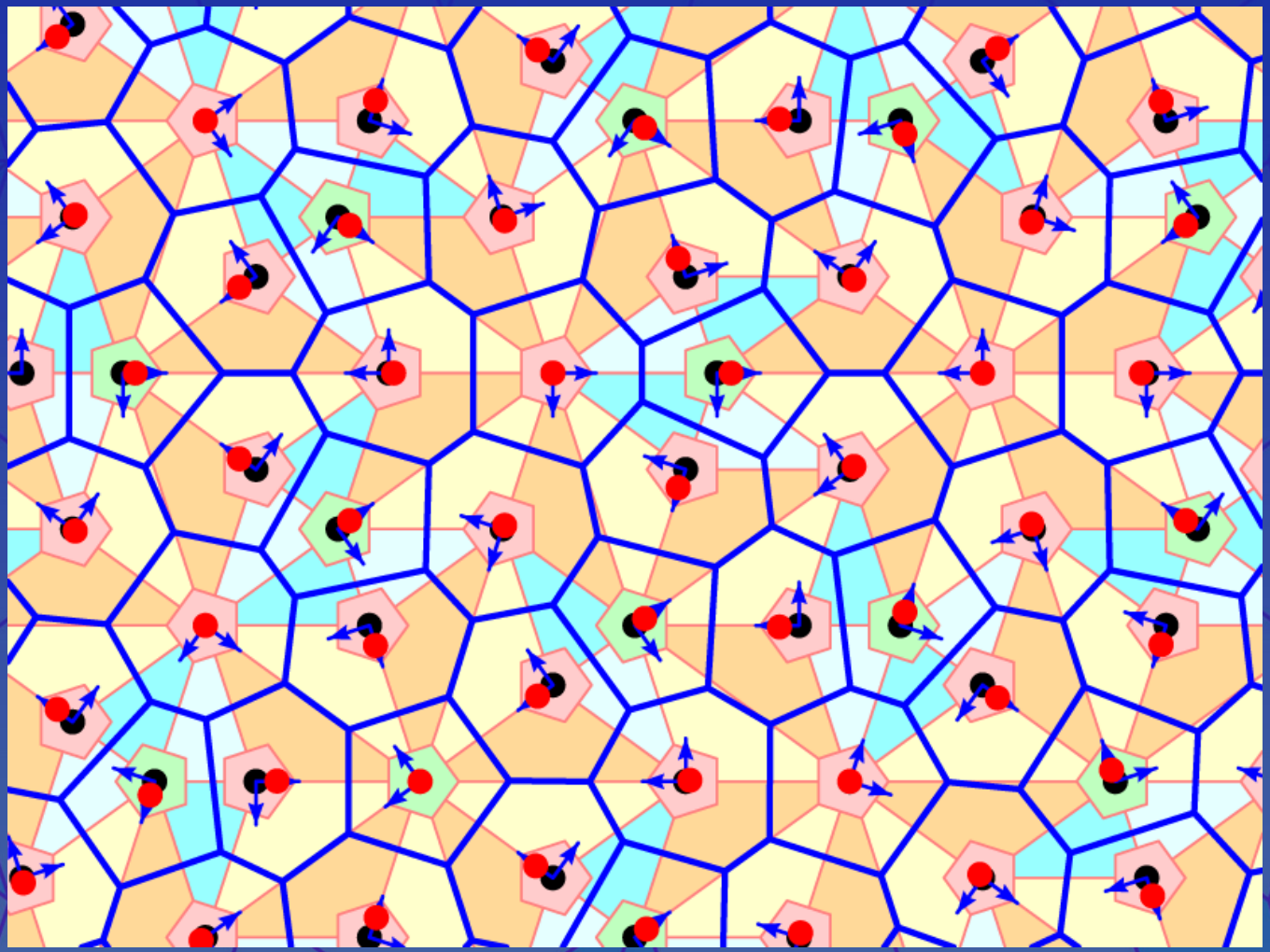
Offline Lloyd's Relaxation: init pts + basis frames



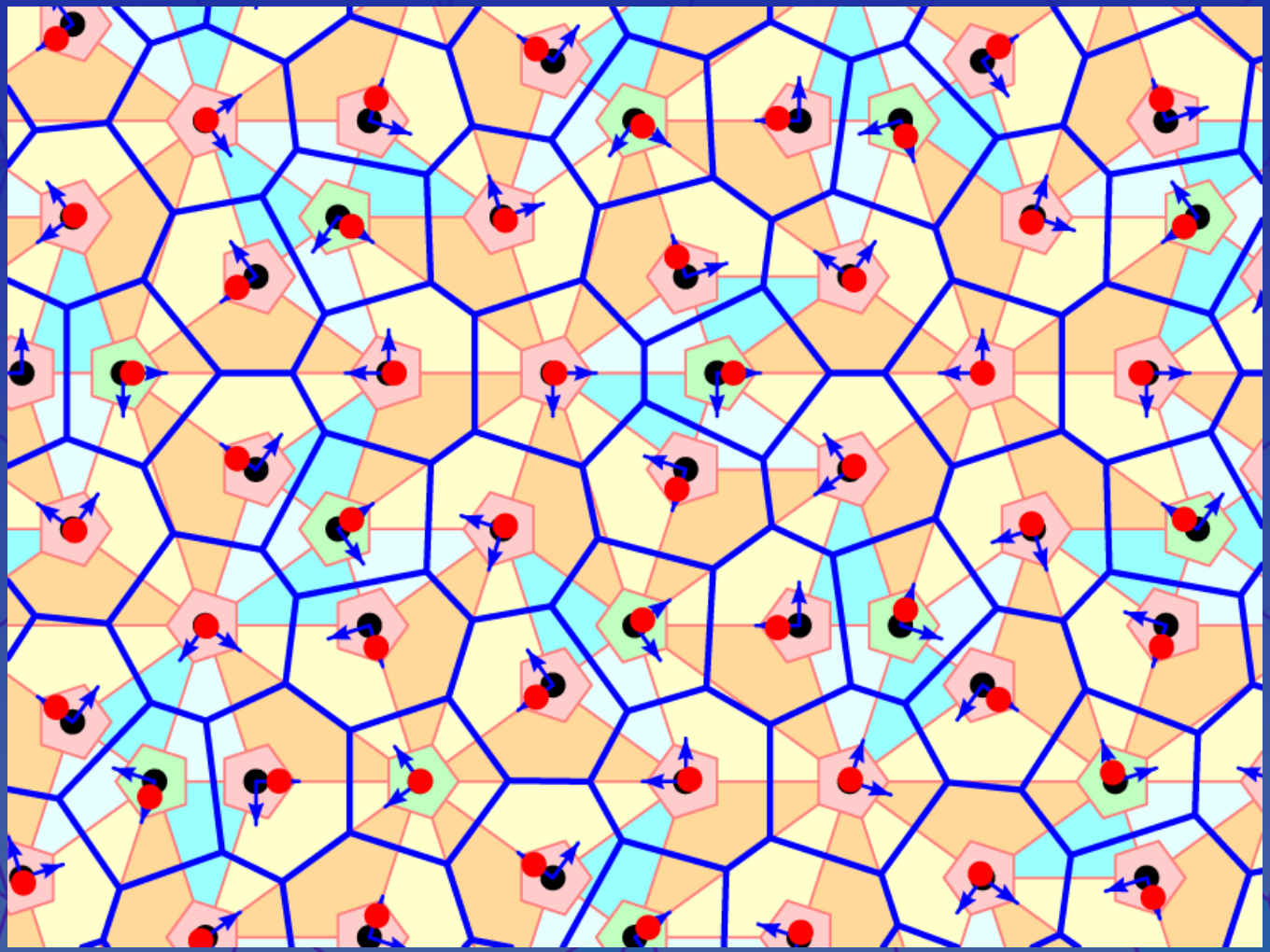
Offline Lloyd's Relaxation: iter 1



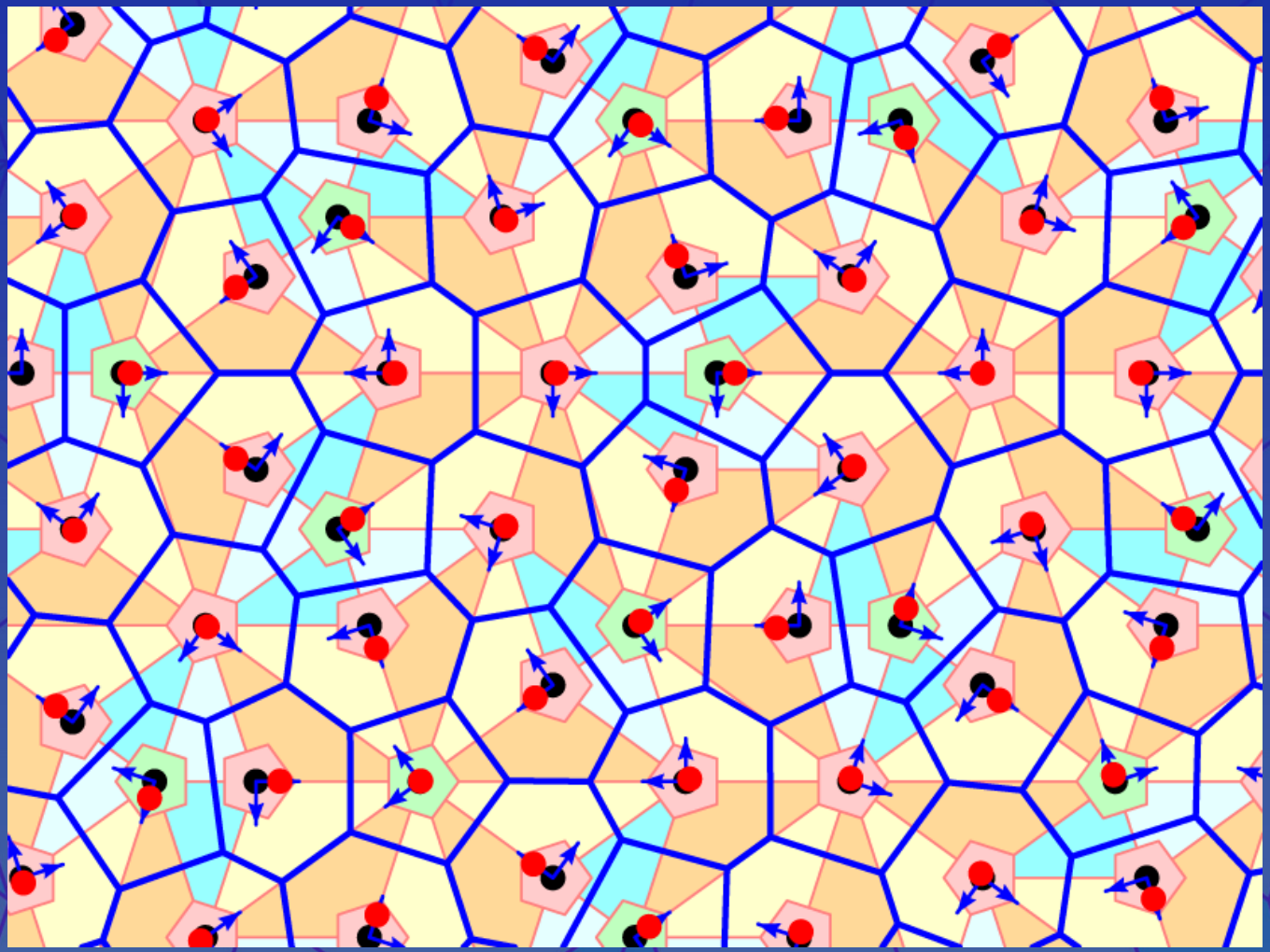
Offline Lloyd's Relaxation: iter 2



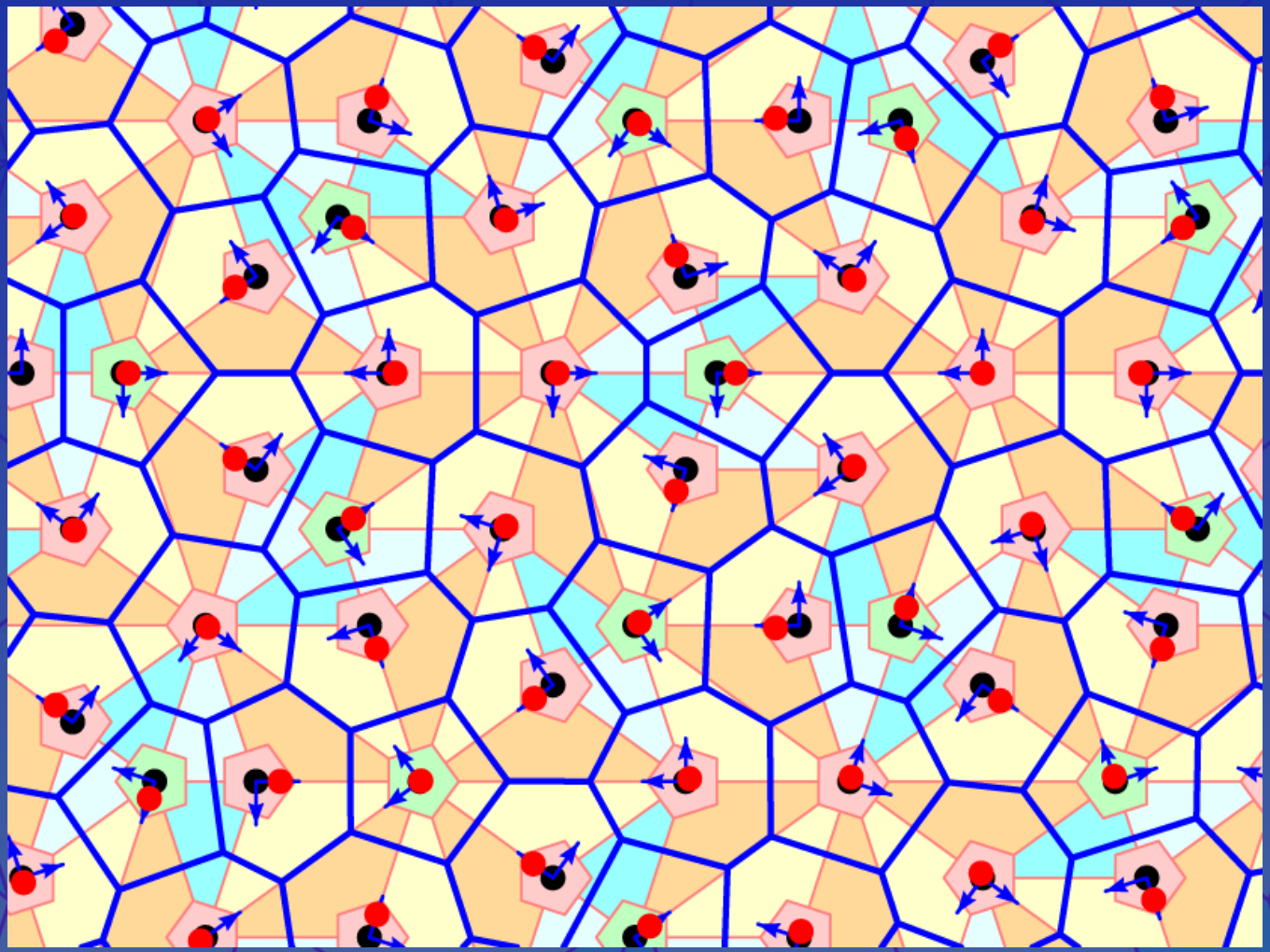
Offline Lloyd's Relaxation: iter 3



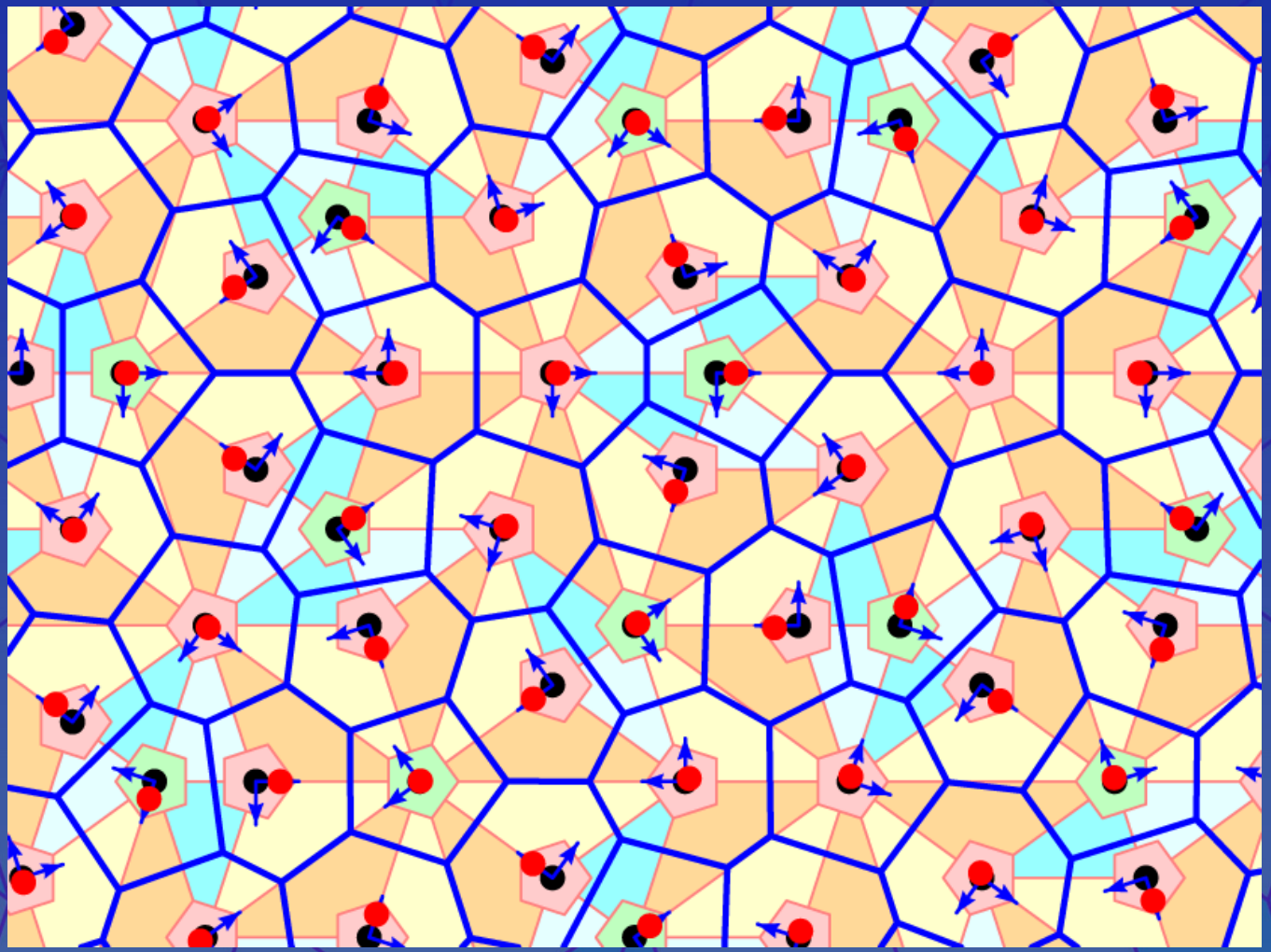
Offline Lloyd's Relaxation: iter 4



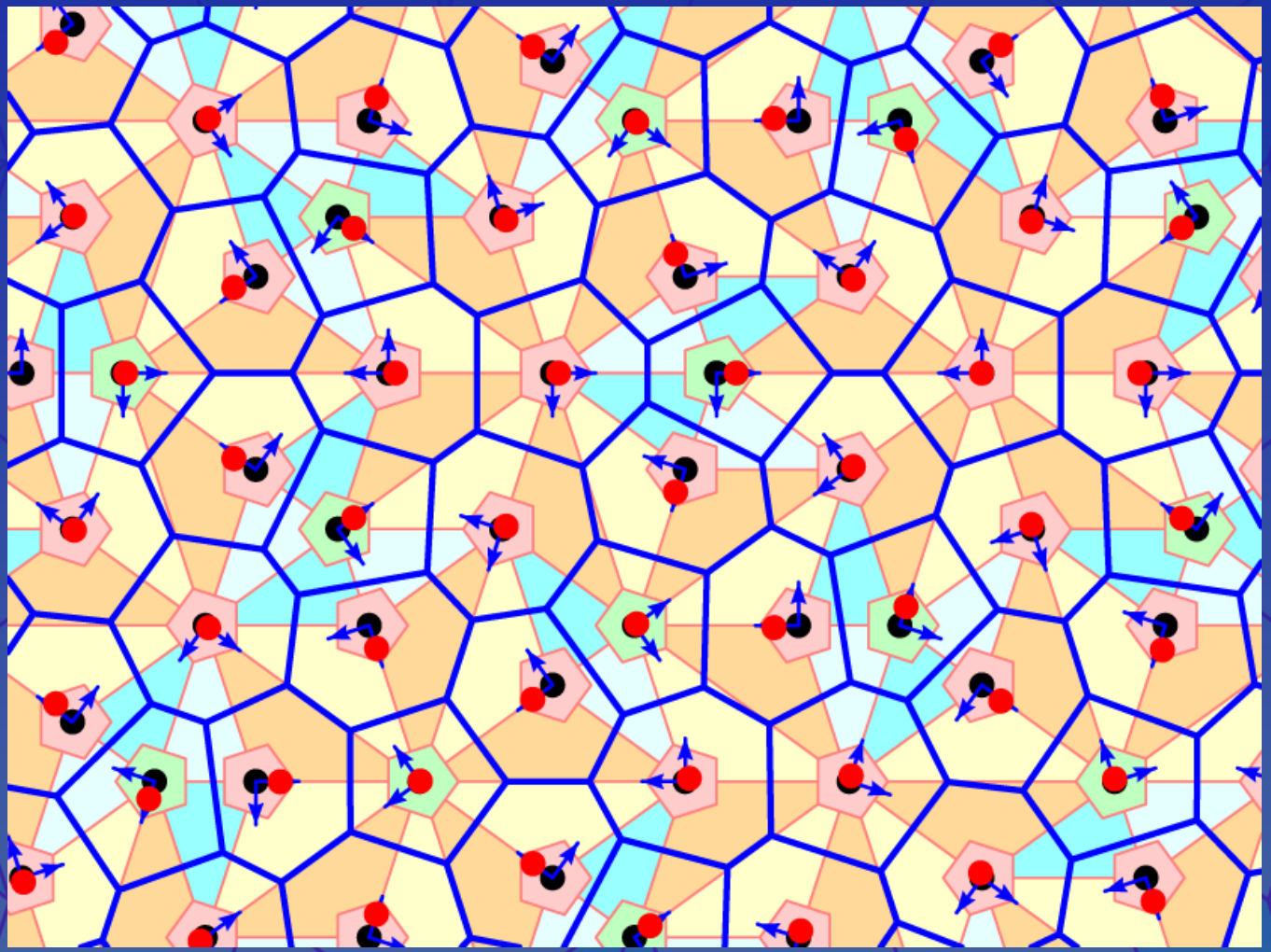
Offline Lloyd's Relaxation: iter 5



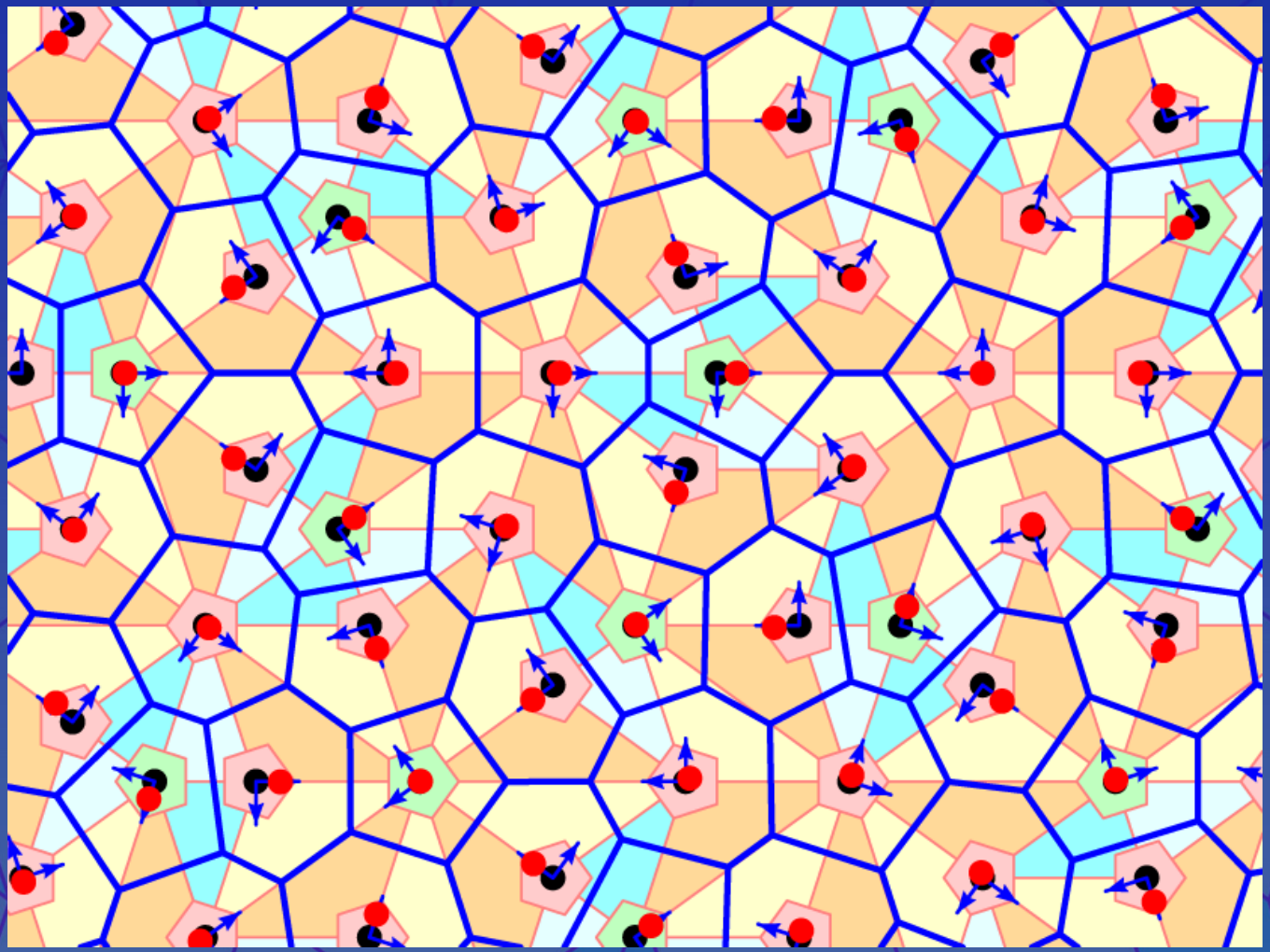
Offline Lloyd's Relaxation: iter 6



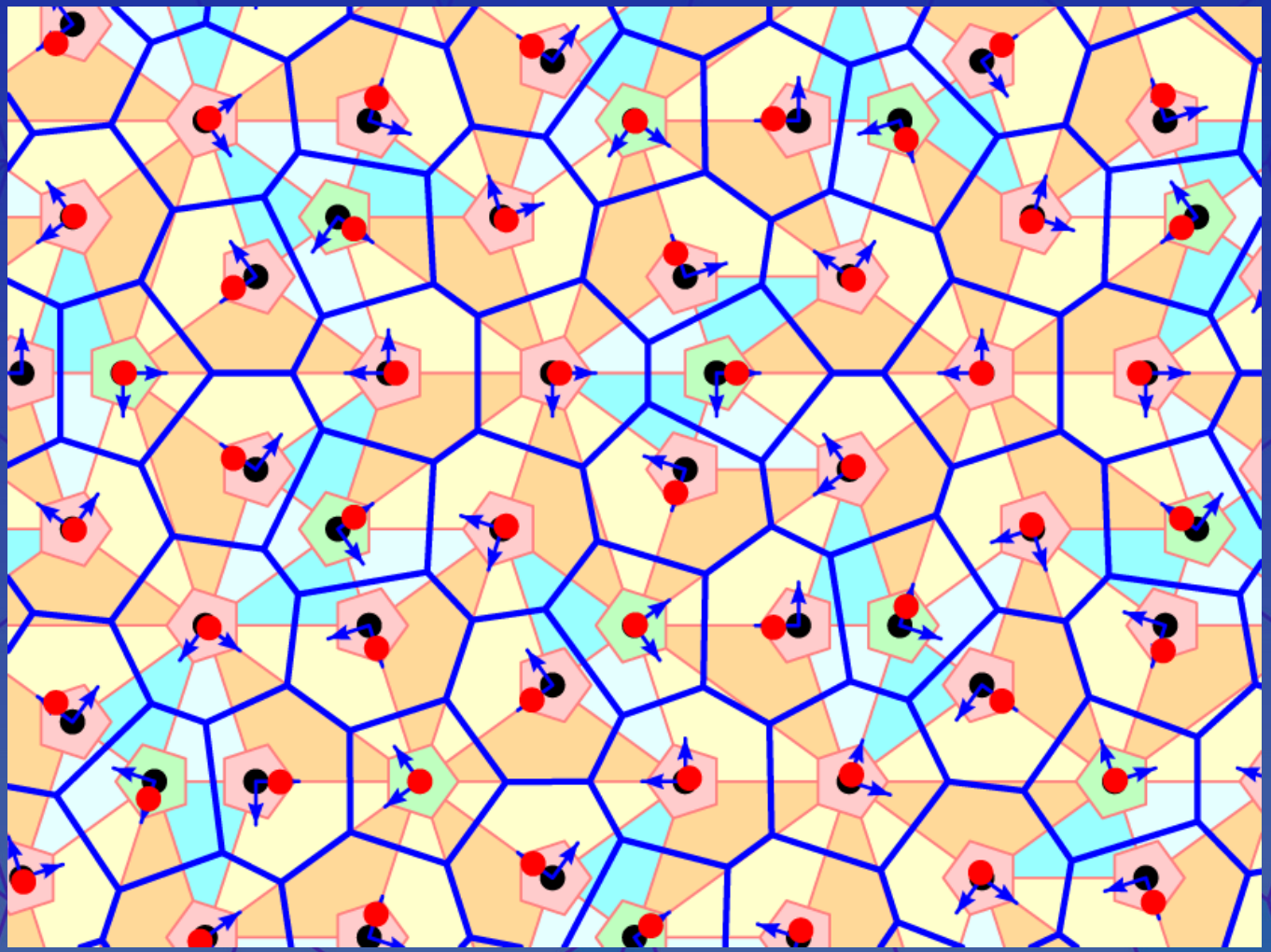
Offline Lloyd's Relaxation: iter 7



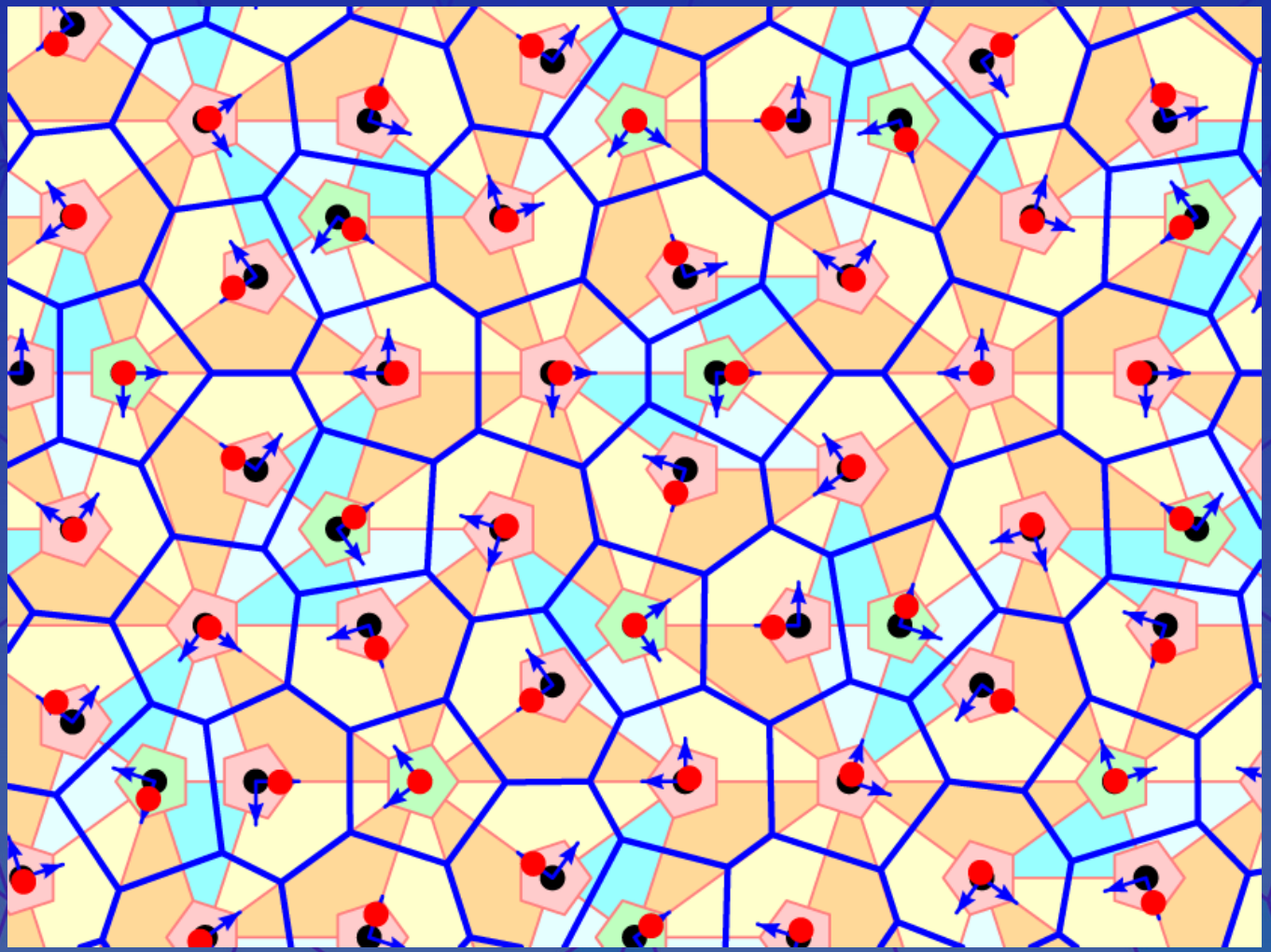
Offline Lloyd's Relaxation: iter 8



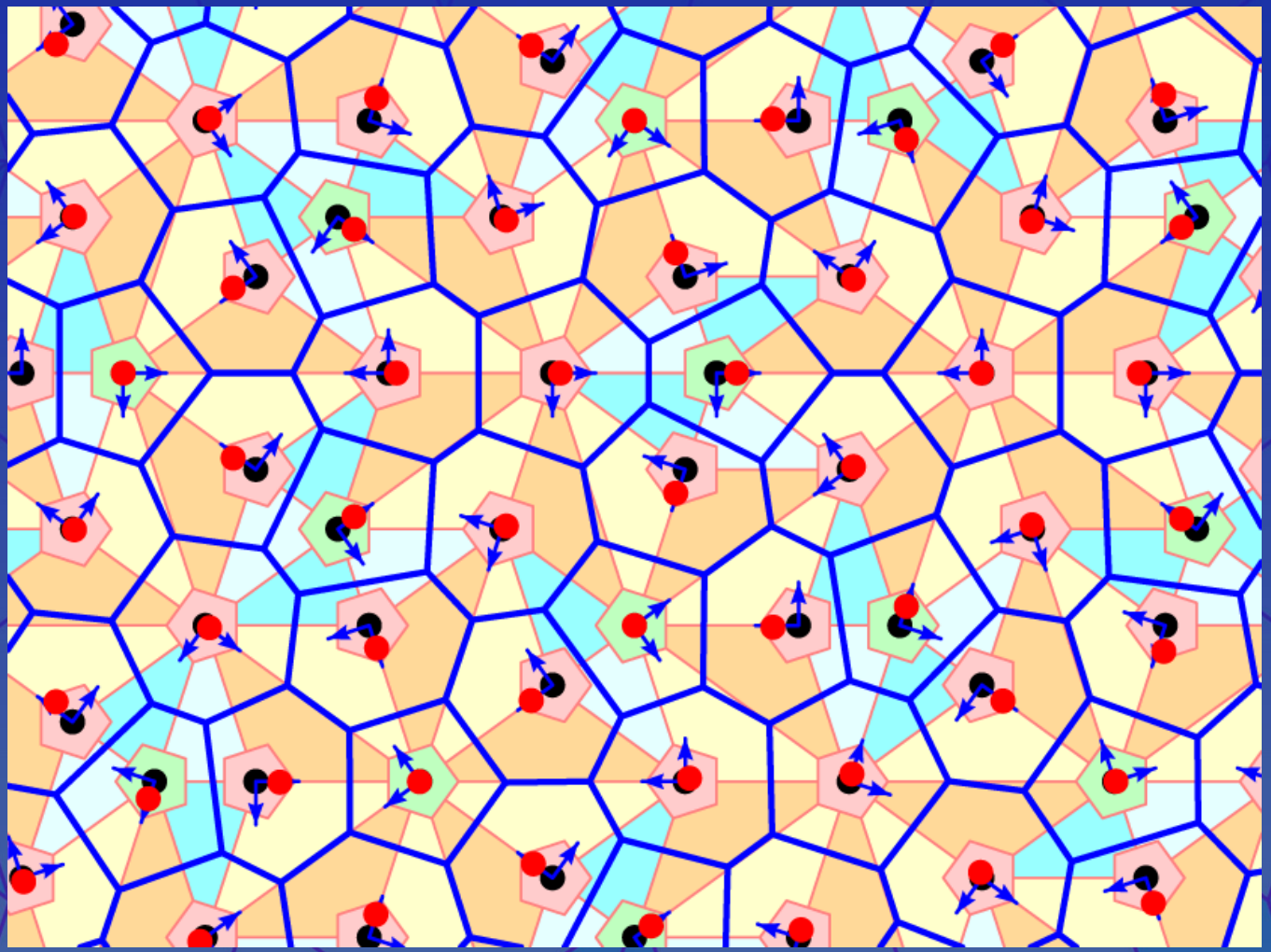
Offline Lloyd's Relaxation: iter 9



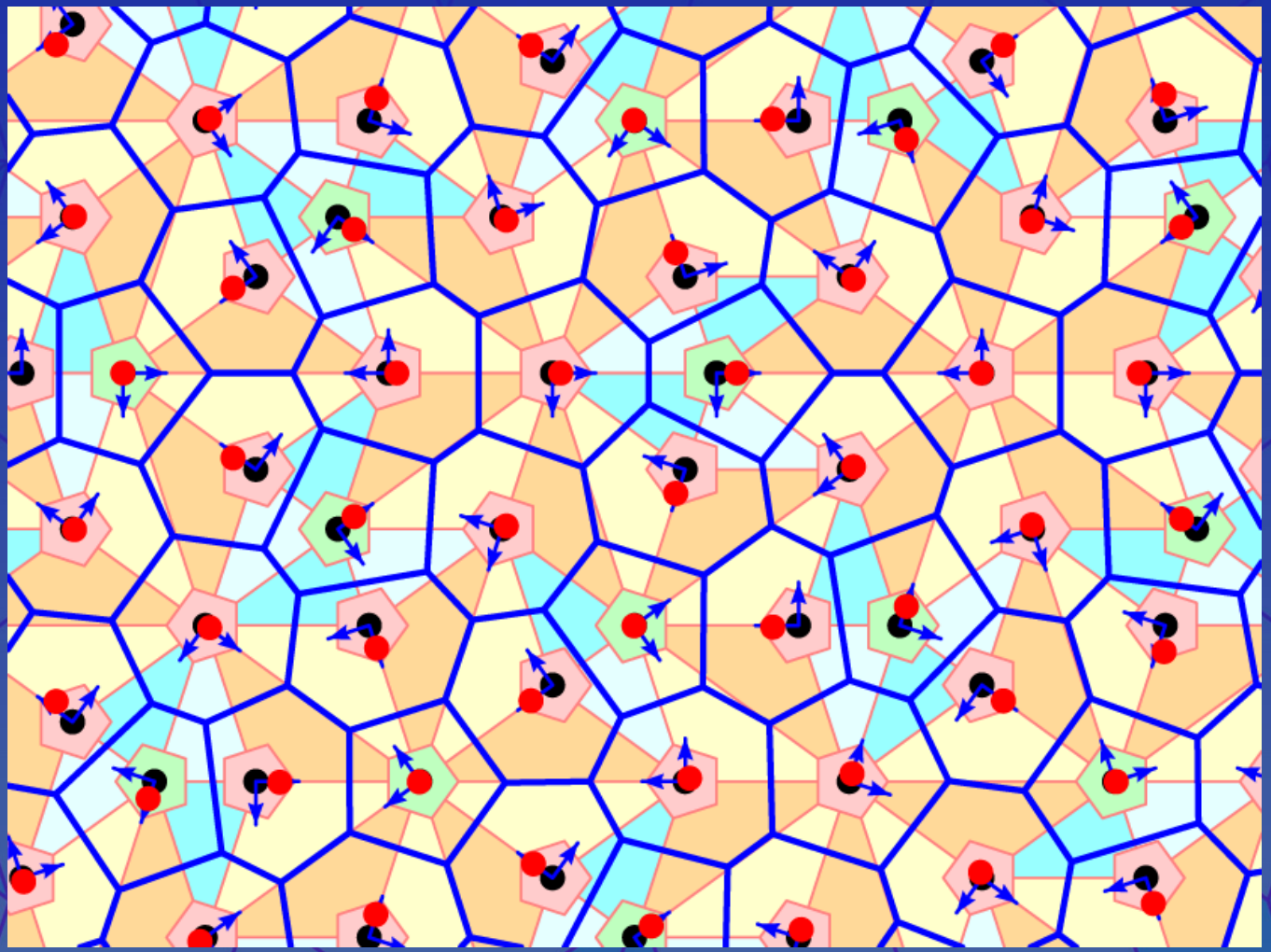
Offline Lloyd's Relaxation: iter 10



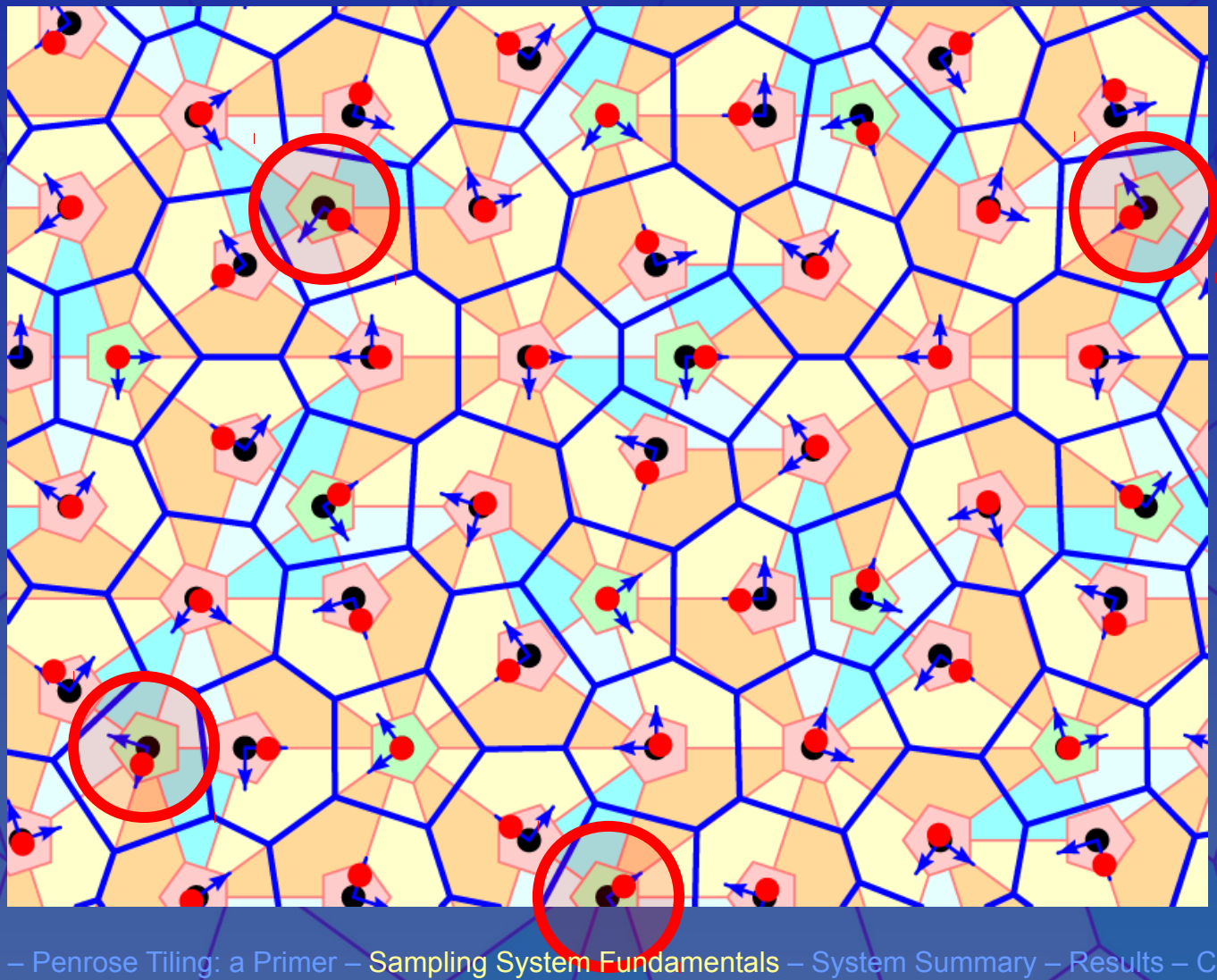
Offline Lloyd's Relaxation: iter 11



Offline Lloyd's Relaxation: iter 12



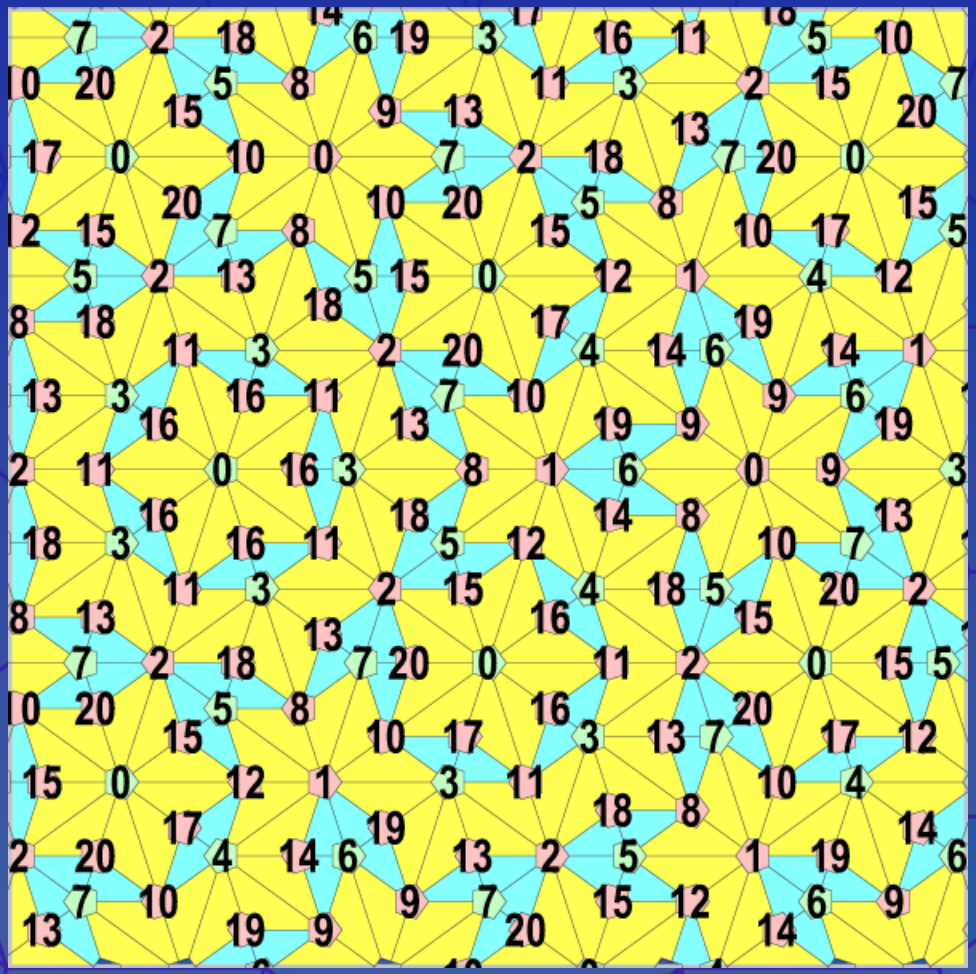
Corrective Vectors Lookup Table Lloyd's Relaxation: iter 12



Corrective Vectors Lookup Table

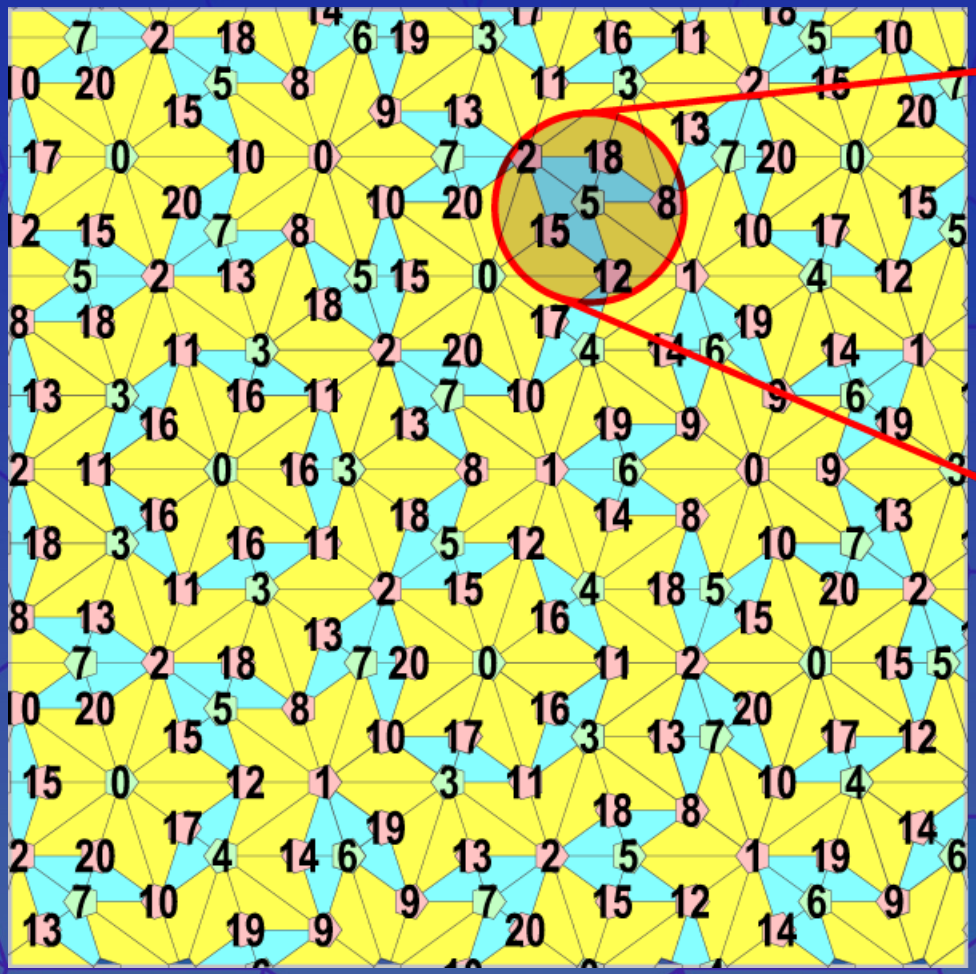
Structural Index i_s \ Intensity Index i_v	0	1	2	3	4	5	6	7
0	(-.014,.018)	(.025,-.082)	(.049,-.097)	(.055,-.090)	(.038,-.074)	(-.006,-.058)	(-.027,-.028)	(-.018,.012)
1	(.013,-.007)	(-.133,-.292)	(-.208,-.358)	(-.197,-.380)	(-.129,-.323)	(-.066,-.266)	(-.029,-.218)	(-.004,-.204)
2	(.011,-.343)	(.079,-.563)	(.095,-.604)	(.072,-.583)	(-.053,-.504)	(-.144,-.383)	(-.170,-.265)	(-.124,-.172)
3	(.014,-.197)	(.085,-.233)	(.148,-.106)	(.236,.143)	(.317,.307)	(.345,.364)	(.296,.330)	(.198,.233)
4	(.036,.044)	(.180,-.014)	(.278,.078)	(.330,.222)	(.348,.358)	(.350,.386)	(.300,.367)	(.215,.301)
5	(.227,.365)	(.363,.061)	(.342,-.094)	(.268,-.156)	(.152,-.091)	(.004,.082)	(-.060,.256)	(-.039,.392)
6	(.171,.446)	(.301,-.089)	(.243,-.224)	(.132,-.299)	(-.052,-.258)	(-.228,-.155)	(-.266,-.013)	(-.200,.140)
7	(-.235,.391)	(-.580,.059)	(-.651,-.114)	(-.709,-.313)	(-.720,-.432)	(-.694,-.411)	(-.593,-.245)	(-.473,-.057)
8	(-.002,1.019)	(.081,.846)	(.118,.731)	(.168,.560)	(.182,.381)	(.221,.263)	(.216,.253)	(.197,.294)
9	(0,0)	(-.125,.256)	(-.108,.273)	(-.065,.249)	(.006,.171)	(.064,.086)	(.084,.026)	(.072,-.013)
10	(0,0)	(-.047,.030)	(-.104,.082)	(-.146,.159)	(-.173,.219)	(-.173,.221)	(-.151,.144)	(-.134,.077)
11	(0,0)	(-.070,.251)	(-.147,.532)	(-.177,.710)	(-.199,.735)	(-.187,.607)	(-.160,.366)	(-.114,.206)
12	(0,0)	(.001,.045)	(.003,.124)	(.009,.290)	(.012,.354)	(.011,.324)	(.010,.182)	(.019,.065)
13	(0,0)	(0,0)	(.020,.012)	(.063,.041)	(.148,.096)	(.198,.131)	(.218,.131)	(.212,.120)
14	(0,0)	(0,0)	(.017,.026)	(.048,.074)	(.107,.166)	(.124,.194)	(.117,.190)	(.099,.165)
15	(0,0)	(0,0)	(0,0)	(.012,.025)	(.038,.073)	(.092,.168)	(.124,.199)	(.138,.203)
16	(0,0)	(0,0)	(0,0)	(-.009,.009)	(-.024,.035)	(-.051,.094)	(-.044,.158)	(-.033,.203)
17	(0,0)	(0,0)	(0,0)	(0,0)	(-.008,.018)	(-.025,.058)	(-.051,.136)	(-.059,.184)
18	(0,0)	(0,0)	(0,0)	(0,0)	(.004,.010)	(.003,.031)	(-.011,.069)	(-.049,.090)
19	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(-.016,.013)	(-.045,.030)	(-.100,.067)
20	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(-.010,.034)	(-.029,.078)	(-.064,.174)

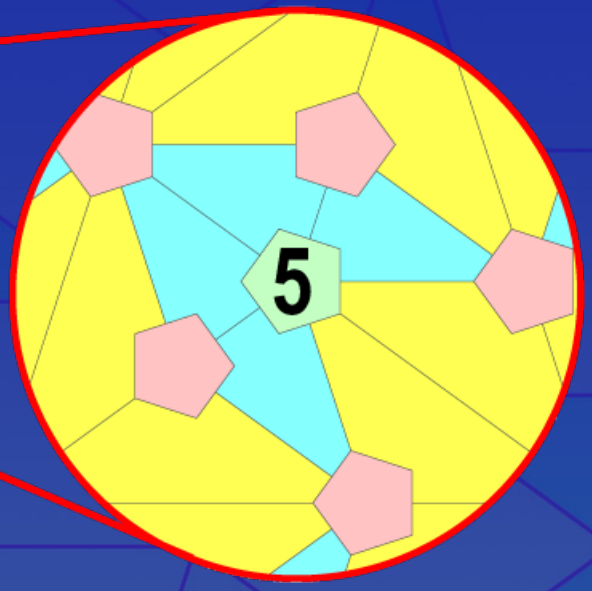
Corrective Vectors Lookup Table Structural Indices (i_s)



6 most-significant bits in F-code

Corrective Vectors Lookup Table Structural Indices (i_s)



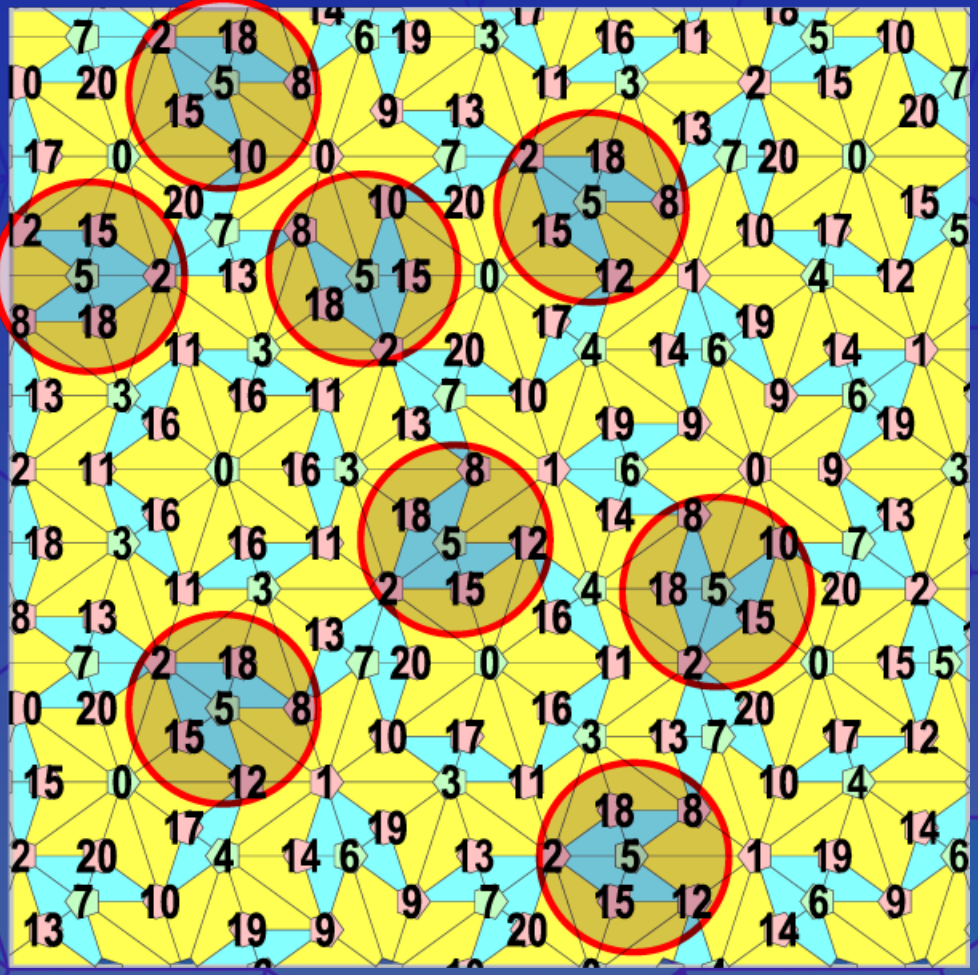


F-code:
0010000000010

6 most-significant bits:
00100000000010

Decimal value: 5

Corrective Vectors Lookup Table Structural Indices (i_s)

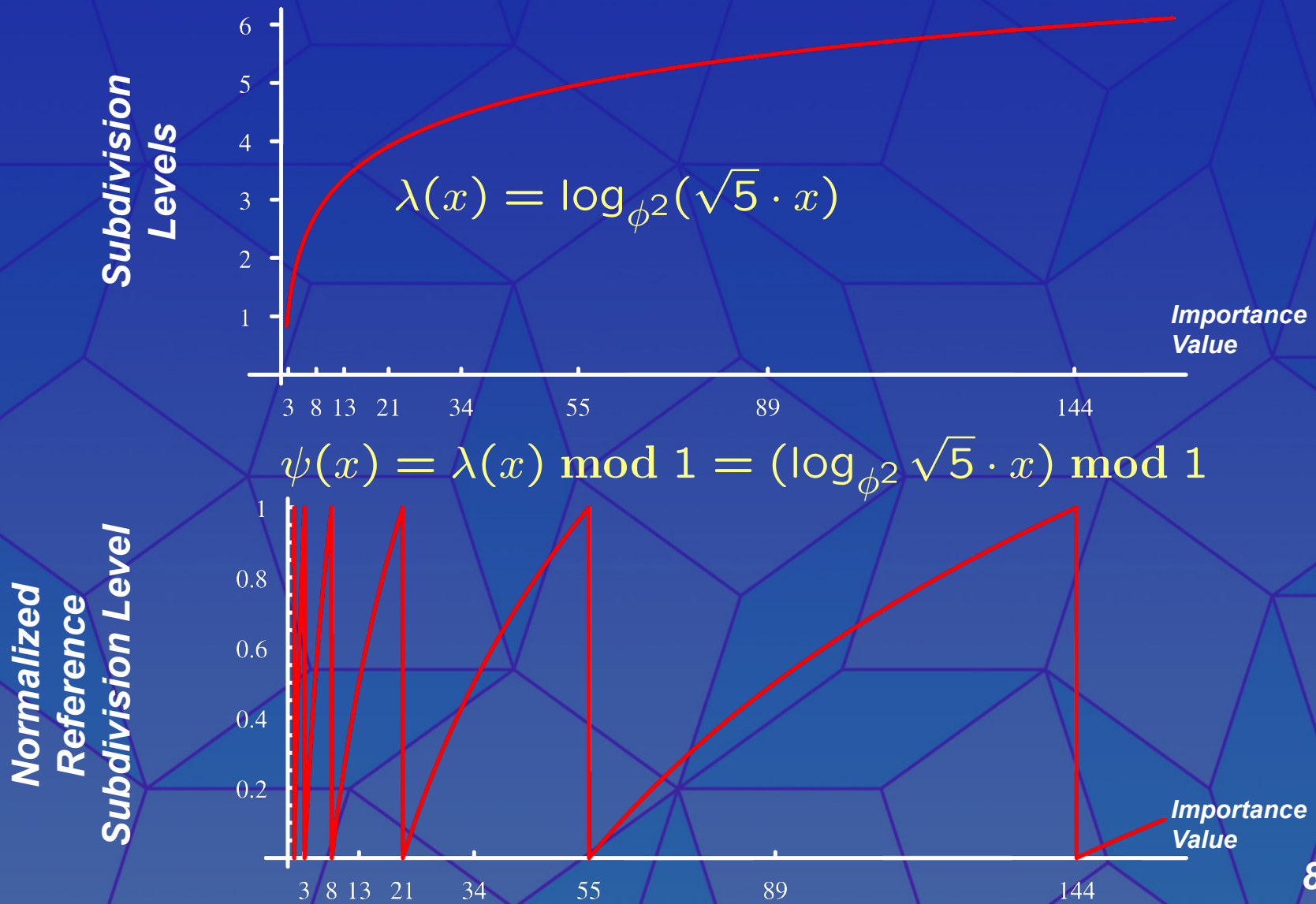


$i_s = 5$

Corrective Vectors Lookup Table

Structural Index i_s \ Intensity Index i_v	0	1	2	3	4	5	6	7
0	(-.014,.018)	(.025,-.082)	(.049,-.097)	(.055,-.090)	(.038,-.074)	(-.006,-.058)	(-.027,-.028)	(-.018,.012)
1	(.013,-.007)	(-.133,-.292)	(-.208,-.358)	(-.197,-.380)	(-.129,-.323)	(-.066,-.266)	(-.029,-.218)	(-.004,-.204)
2	(.011,-.343)	(.079,-.563)	(.095,-.604)	(.072,-.583)	(-.053,-.504)	(-.144,-.383)	(-.170,-.265)	(-.124,-.172)
3	(.014,-.197)	(.085,-.233)	(.148,-.106)	(.236,.143)	(.317,.307)	(.345,.364)	(.296,.330)	(.198,.233)
4	(.036,.044)	(.180,-.014)	(.278,.078)	(.330,.222)	(.348,.358)	(.350,.386)	(.300,.367)	(.215,.301)
5	(.227,.365)	(.363,.061)	(.342,-.094)	(.268,-.156)	(.152,-.091)	(.004,.082)	(-.060,.256)	(-.039,.392)
6	(.171,.446)	(.301,-.089)	(.243,-.224)	(.132,-.299)	(-.052,-.258)	(-.228,-.155)	(-.266,-.013)	(-.200,.140)
7	(-.235,.391)	(-.580,.059)	(-.651,-.114)	(-.709,-.313)	(-.720,-.432)	(-.694,-.411)	(-.593,-.245)	(-.473,-.057)
8	(-.002,1.019)	(.081,.846)	(.118,.731)	(.168,.560)	(.182,.381)	(.221,.263)	(.216,.253)	(.197,.294)
9	(0,0)	(-.125,.256)	(-.108,.273)	(-.065,.249)	(.006,.171)	(.064,.086)	(.084,.026)	(.072,-.013)
10	(0,0)	(-.047,.030)	(-.104,.082)	(-.146,.159)	(-.173,.219)	(-.173,.221)	(-.151,.144)	(-.134,.077)
11	(0,0)	(-.070,.251)	(-.147,.532)	(-.177,.710)	(-.199,.735)	(-.187,.607)	(-.160,.366)	(-.114,.206)
12	(0,0)	(.001,.045)	(.003,.124)	(.009,.290)	(.012,.354)	(.011,.324)	(.010,.182)	(.019,.065)
13	(0,0)	(0,0)	(.020,.012)	(.063,.041)	(.148,.096)	(.198,.131)	(.218,.131)	(.212,.120)
14	(0,0)	(0,0)	(.017,.026)	(.048,.074)	(.107,.166)	(.124,.194)	(.117,.190)	(.099,.165)
15	(0,0)	(0,0)	(0,0)	(.012,.025)	(.038,.073)	(.092,.168)	(.124,.199)	(.138,.203)
16	(0,0)	(0,0)	(0,0)	(-.009,.009)	(-.024,.035)	(-.051,.094)	(-.044,.158)	(-.033,.203)
17	(0,0)	(0,0)	(0,0)	(0,0)	(-.008,.018)	(-.025,.058)	(-.051,.136)	(-.059,.184)
18	(0,0)	(0,0)	(0,0)	(0,0)	(.004,.010)	(.003,.031)	(-.011,.069)	(-.049,.090)
19	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(-.016,.013)	(-.045,.030)	(-.100,.067)
20	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(-.010,.034)	(-.029,.078)	(-.064,.174)

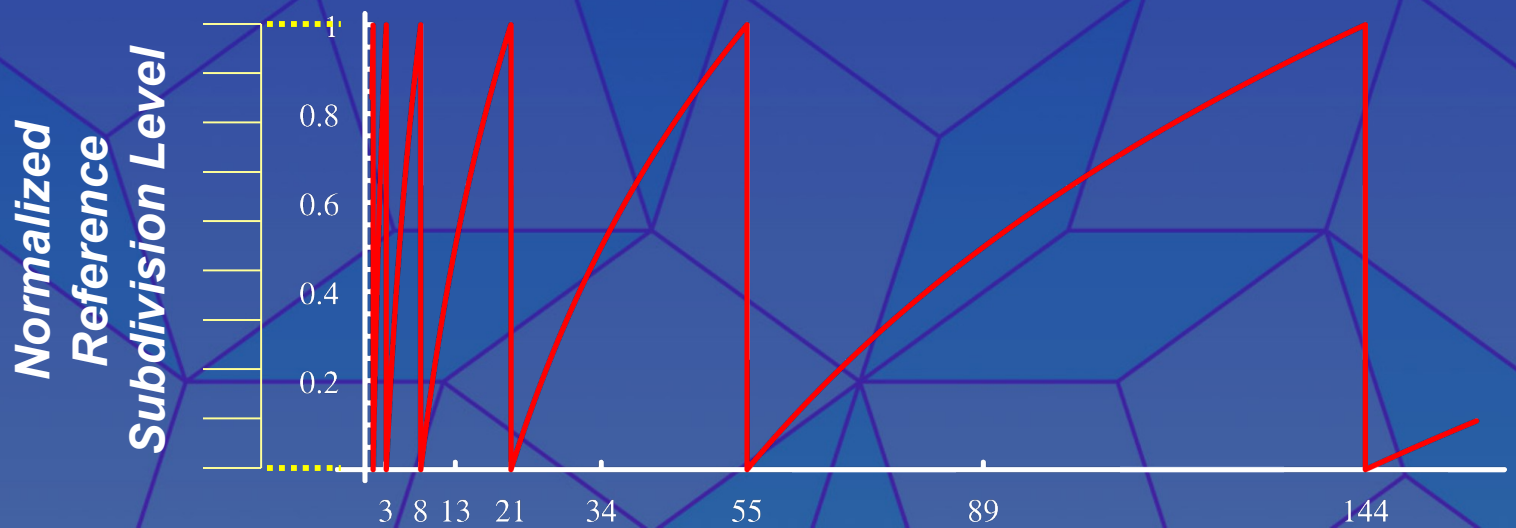
Corrective Vectors Lookup Table Importance Indices (i_v)



Corrective Vectors Lookup Table Importance Indices (i_v)

$$i_v = \lfloor n \cdot \psi(mag \cdot I(x,y)) \rfloor$$

$$\psi(x) = (\log_{\phi^2} \sqrt{5} \cdot x) \bmod 1$$



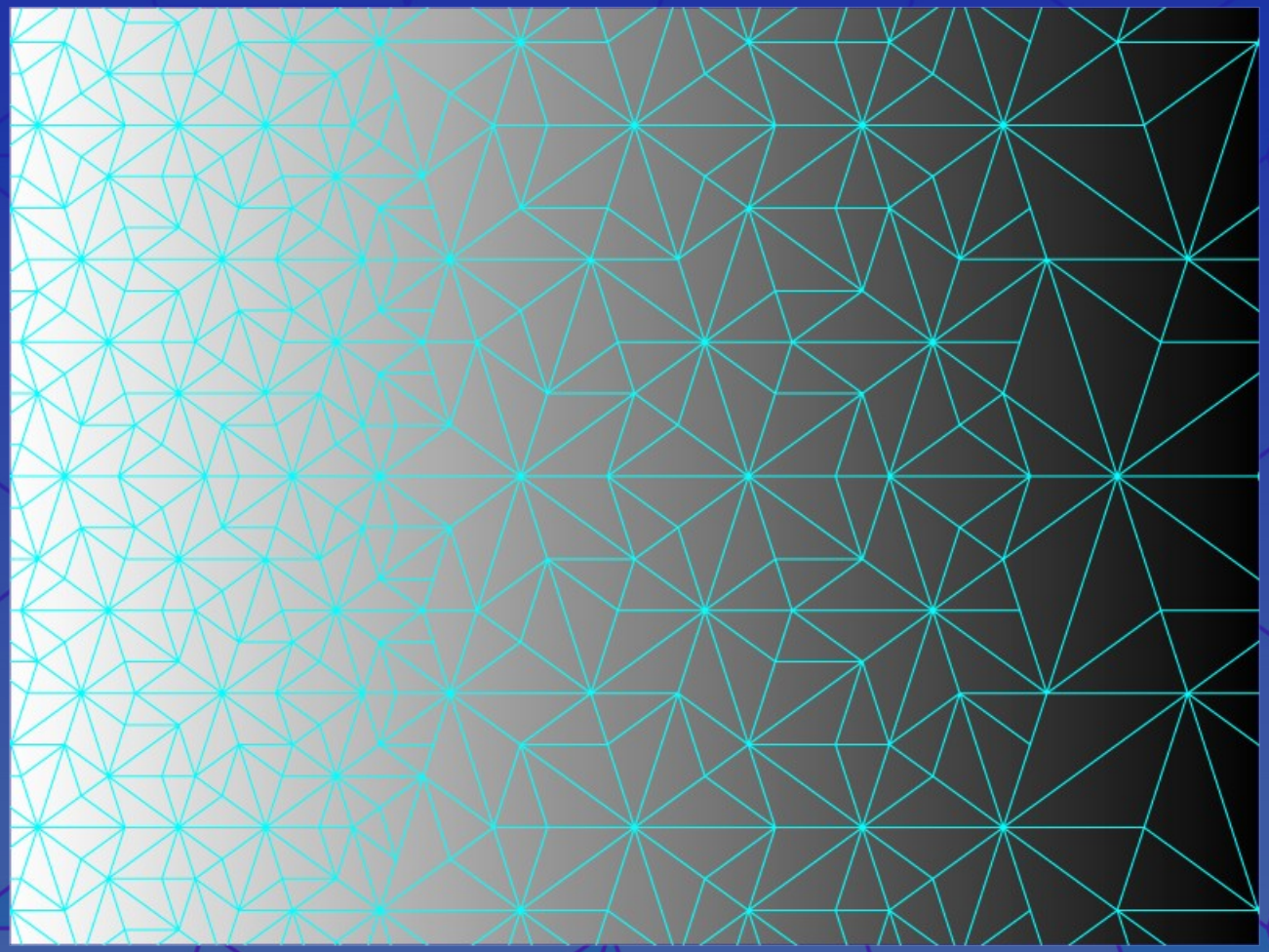
System Summary

System Summary

Sampled density function

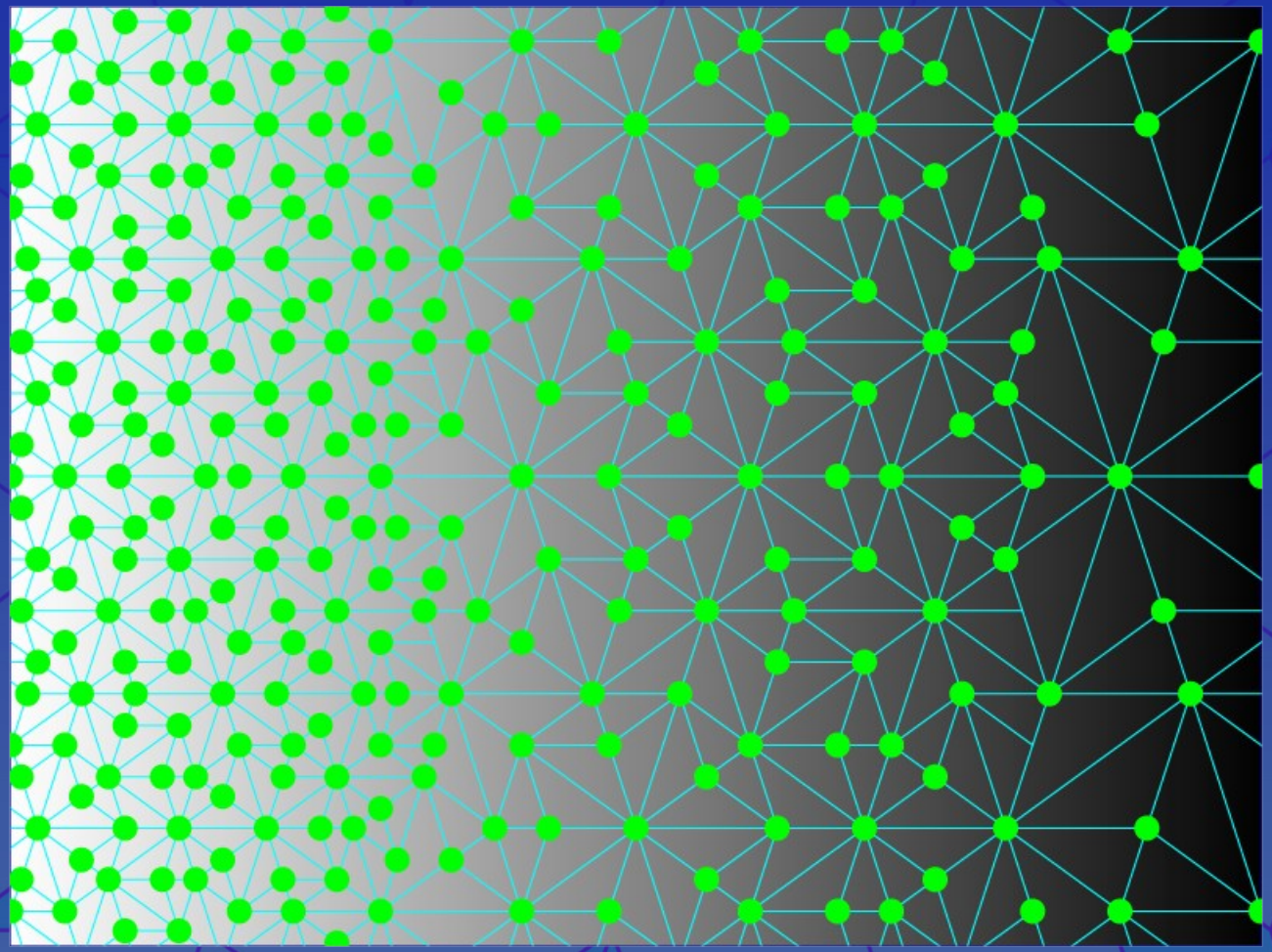
System Summary

Adaptive subdivision



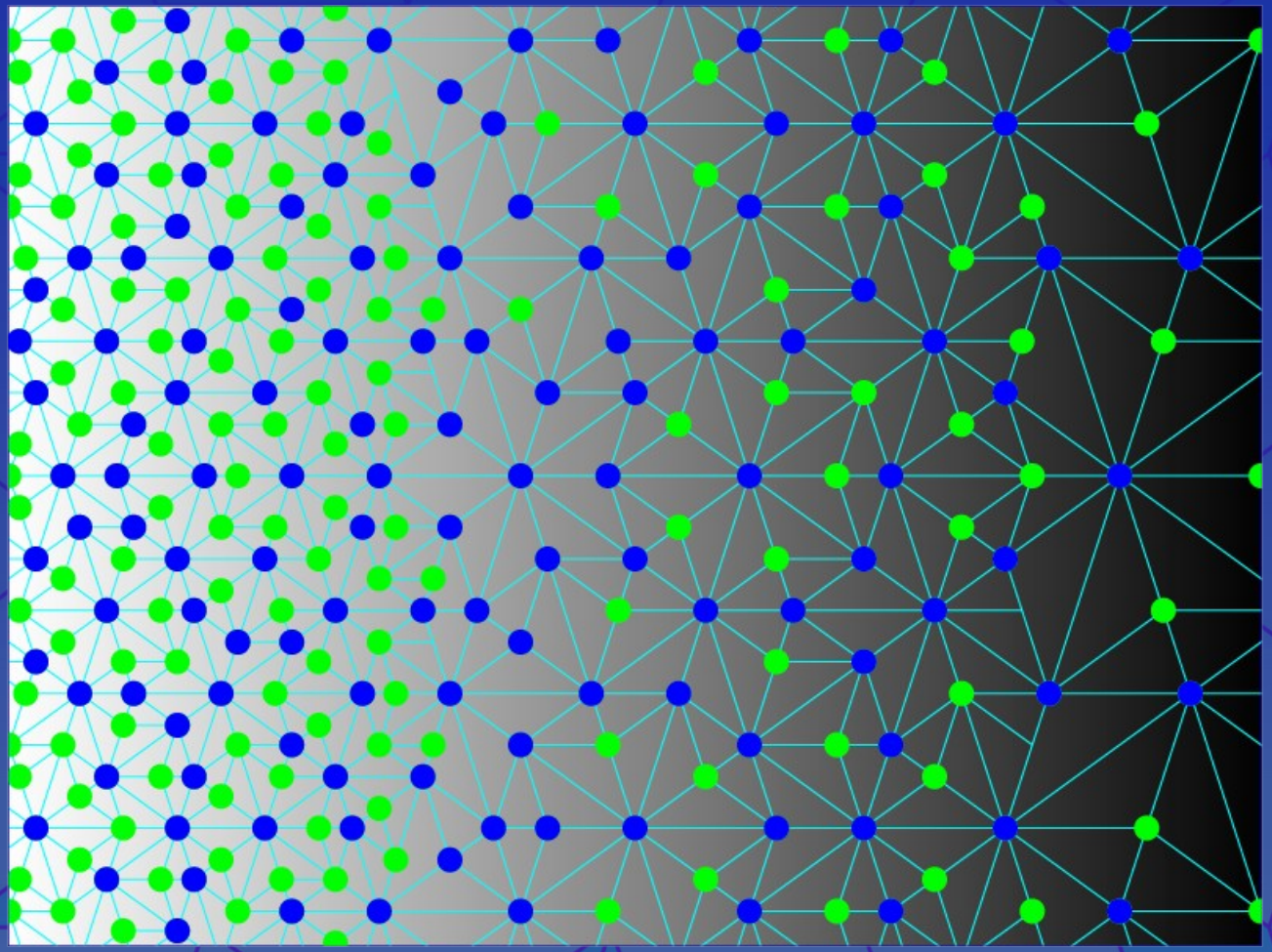
System Summary

'Sampling' tiles



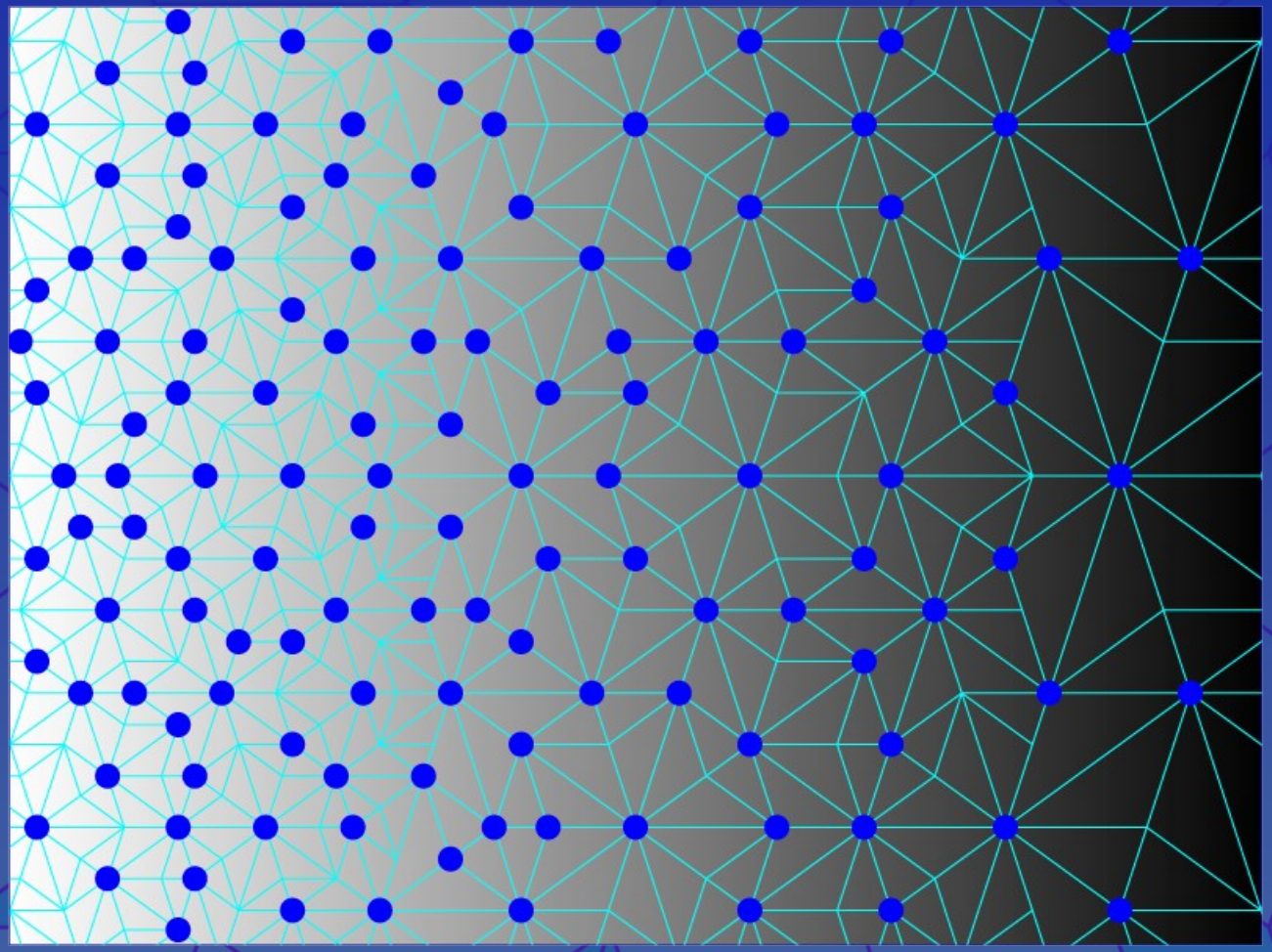
System Summary

Threshold with F-codes



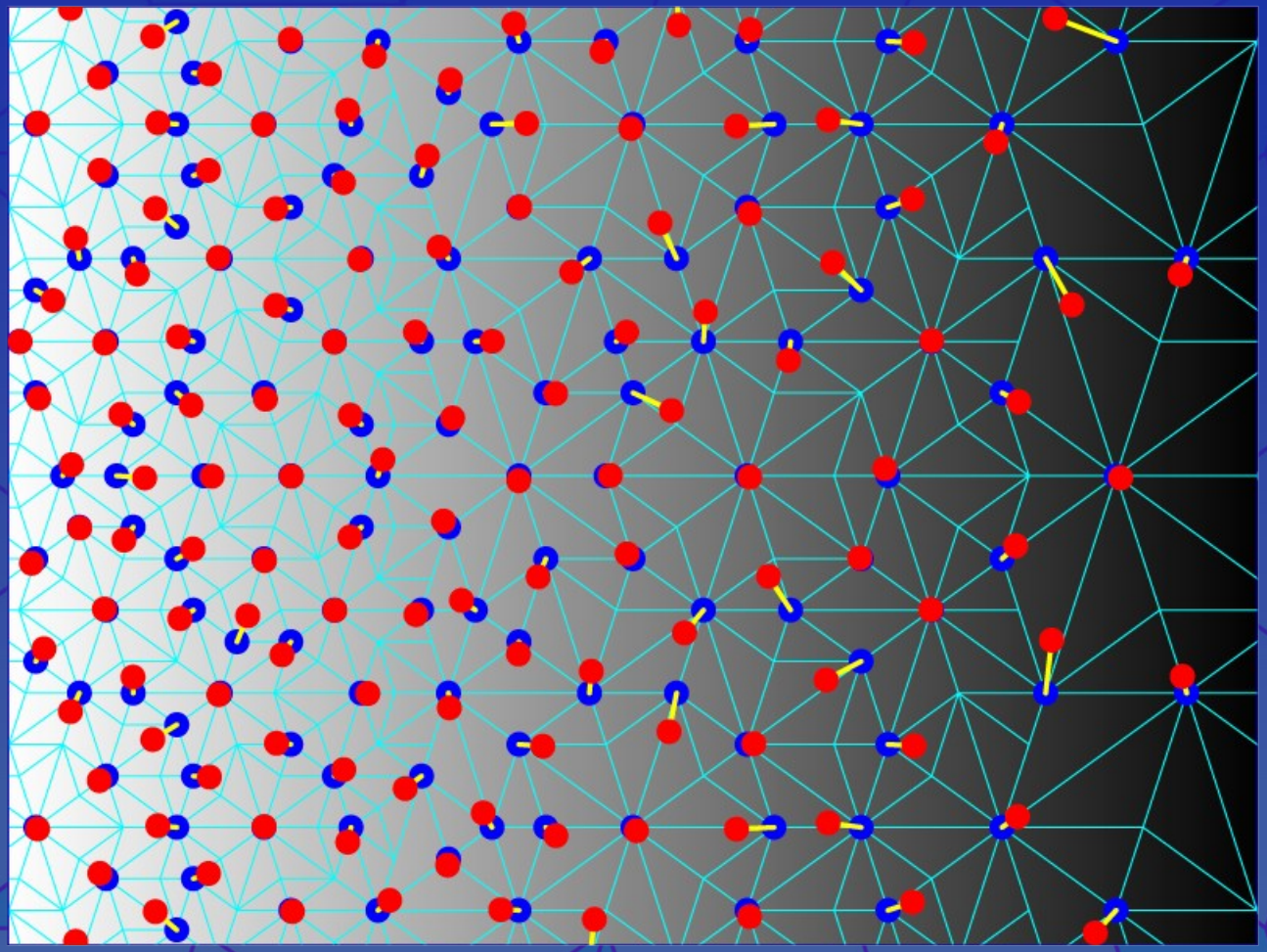
System Summary

Threshold with F-codes



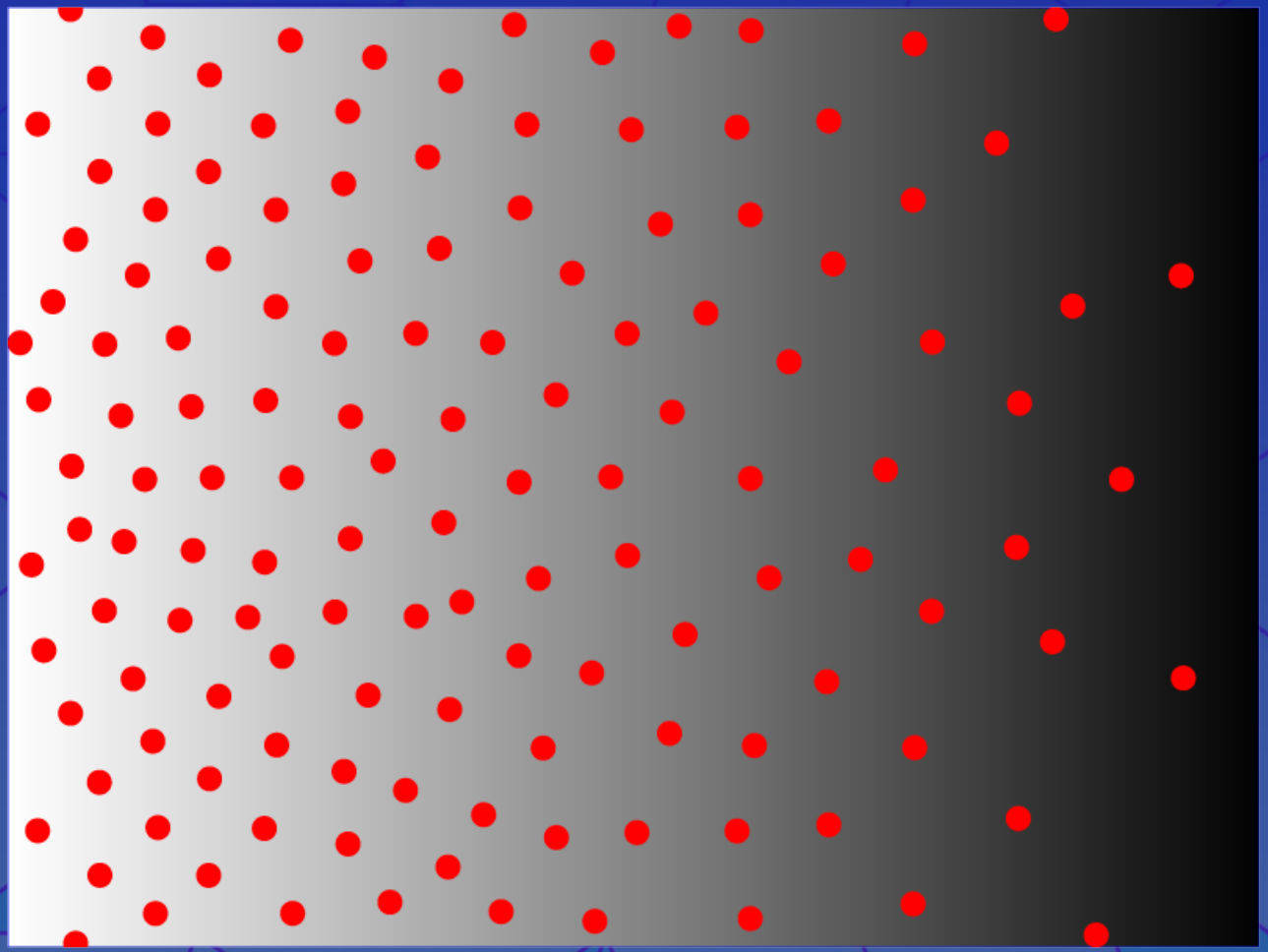
System Summary

Corrective Vectors



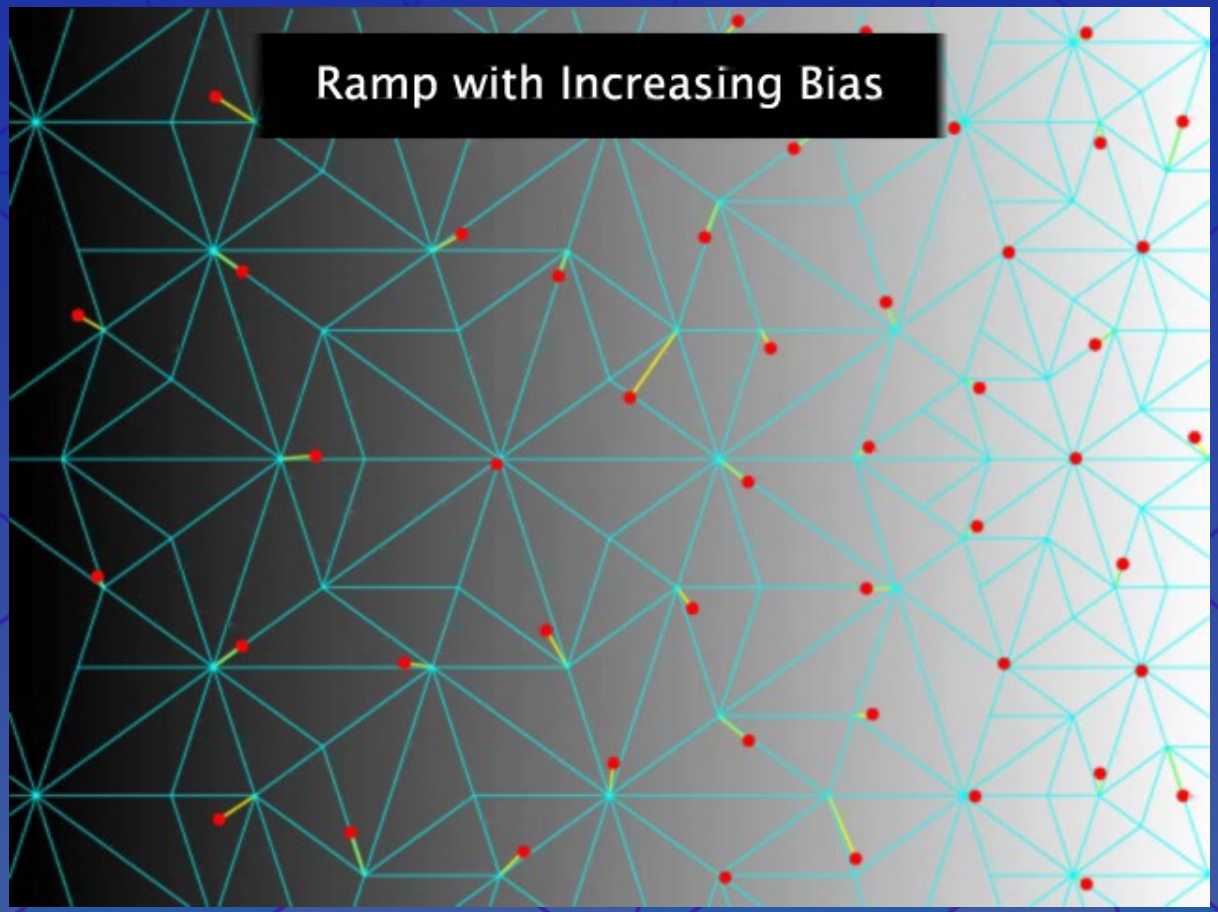
System Summary

Final Sampling



Results

Results

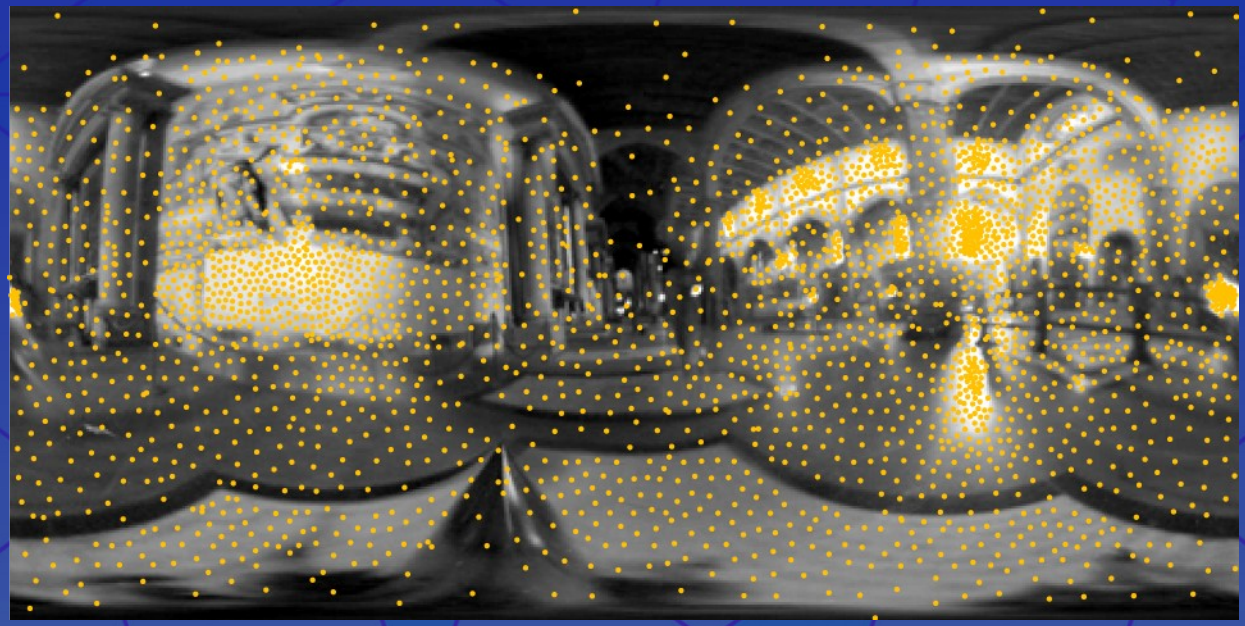



SIGGRAPH2004
Case Study:
Environment Map Sampling

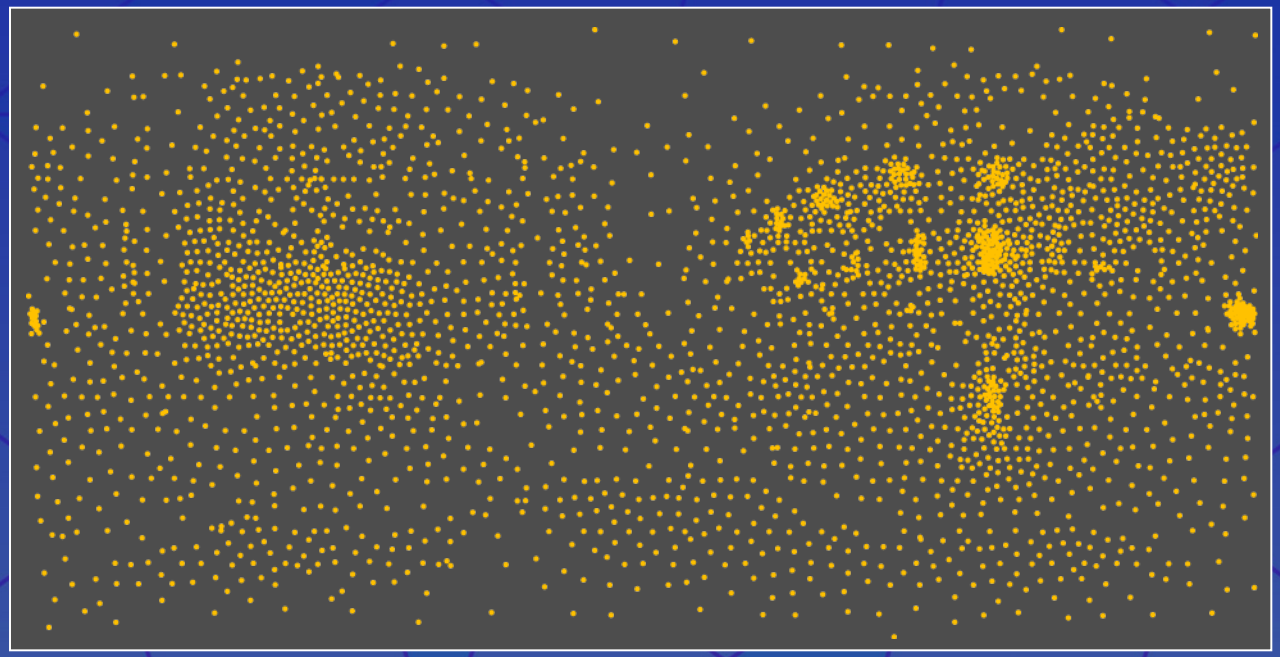
Results

Image credits: Paul Debevec

Results

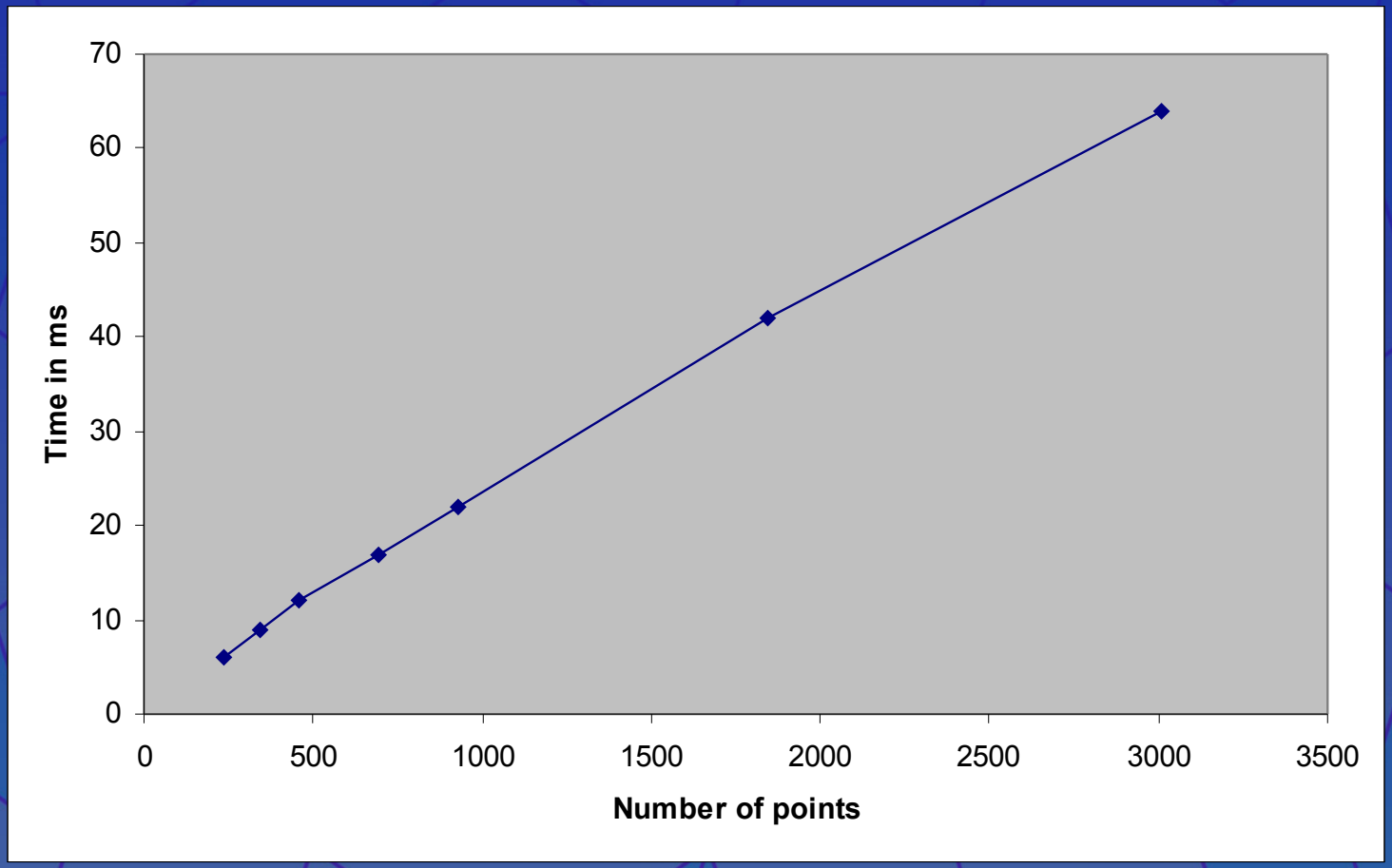


Results



Results

Timings



Polyominoes-based Sampling System [SIGGRAPH'2007]

Penrose tiling vs. Polyomino-based sampling

Penrose tiling

- Deterministic production rules
- F-code attributes: integers in Fibonacci Number System
- Structural indices: local geometrical neighborhoods
- Simple lookup table-based run-time algorithm

D

Idem

Improve-ment

Polyominoes

- Deterministic production rules
- Digital code attributes: integers in base $\mathcal{A}$
- Structural indices: local geometrical neighborhoods
- Simple lookup table-based run-time algorithm

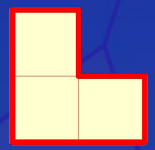
Advantages

- Less residual picks in Fourier amplitude spectrum
- Structure aligned with square grid

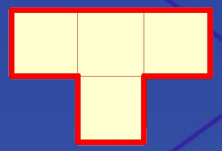
Polyominoes

Monomino

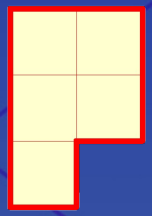
Domino



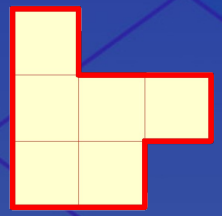
L-tromino



T-tetromino

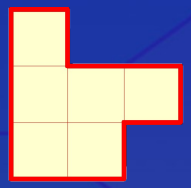


P-pentomino

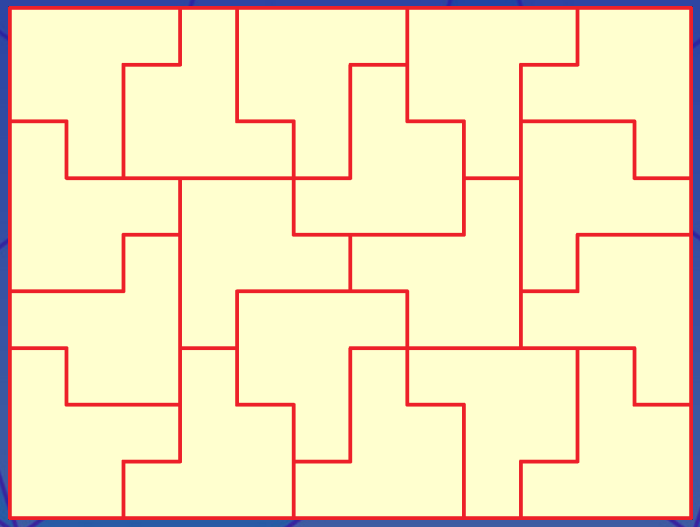


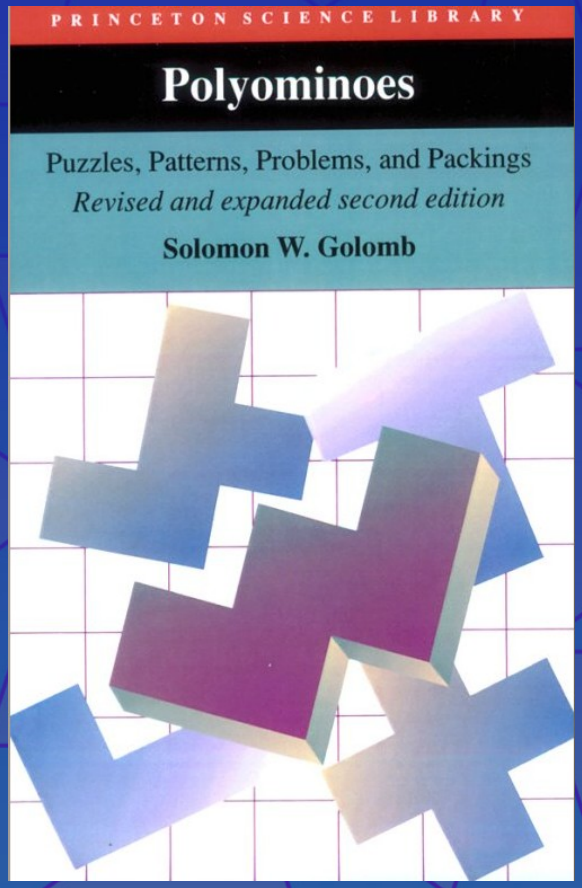
G-hexomino

Rectifiable polyominoes



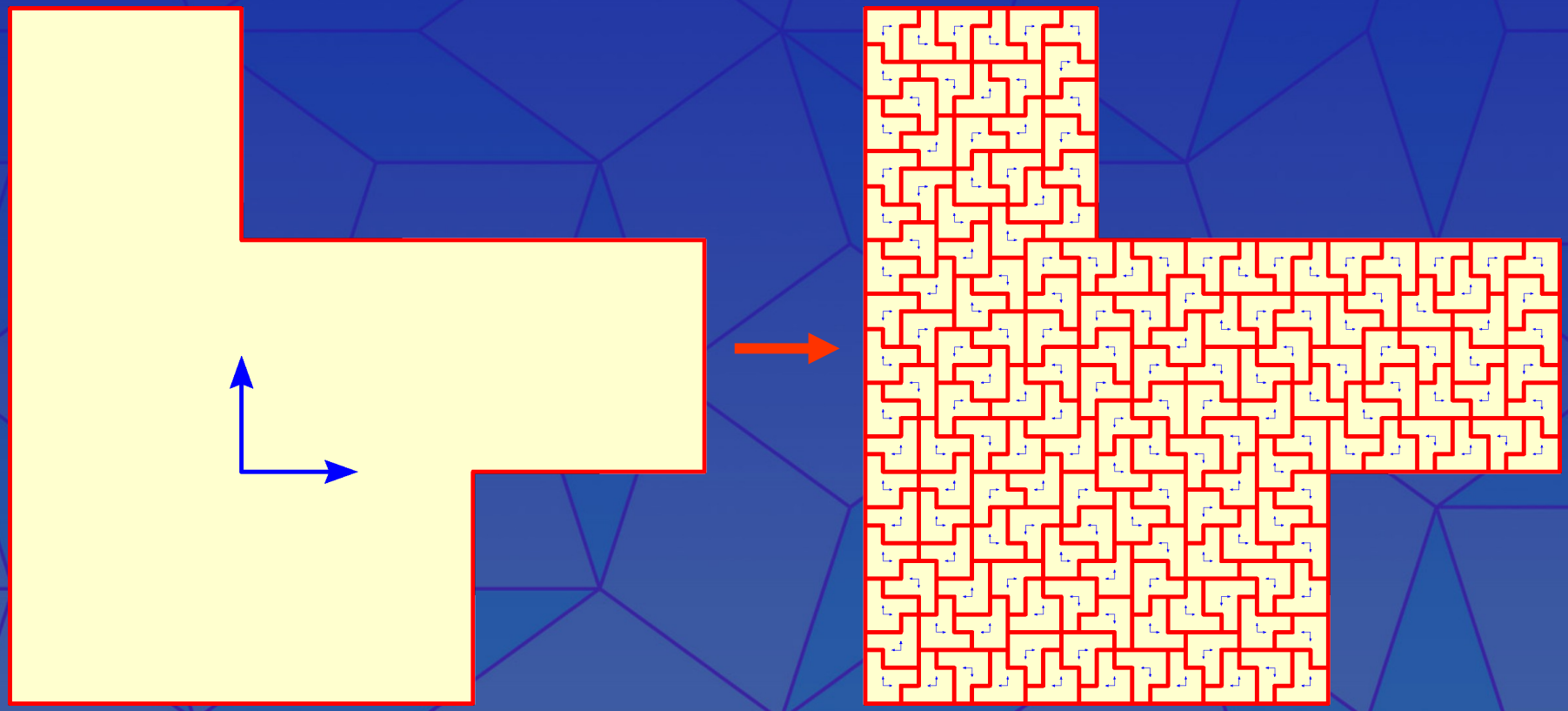
G-hexomino



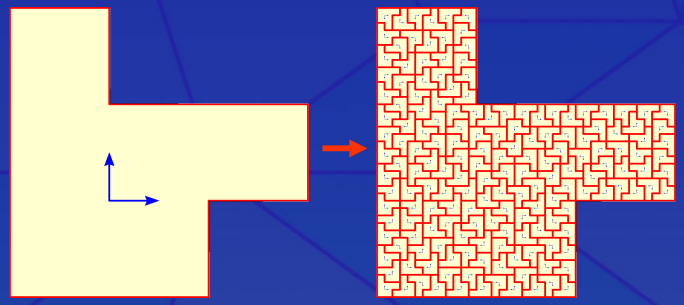


Self-similar ($\mathcal{A}$-rep) rectifiable polyominoes, or reptiles

9^2-rep or 81-rep G-hexominoes
Area scaling factor $\mathcal{A} = 9^2 = 81$



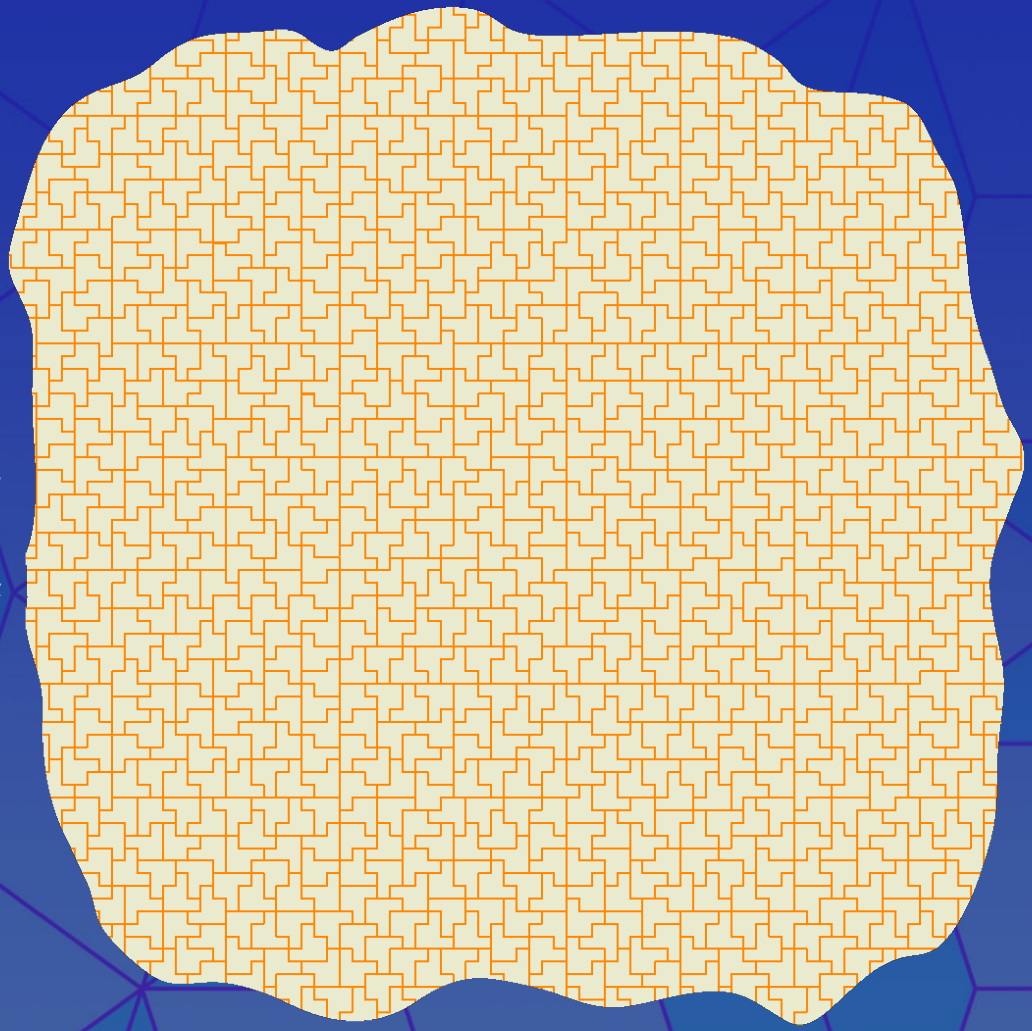
$\mathcal{A}$-rep rectifiable polyominoes, or reptiles



→ *Unique production rule*

According to "Tilings and Patterns" [Grunbaum & Shephard '86]

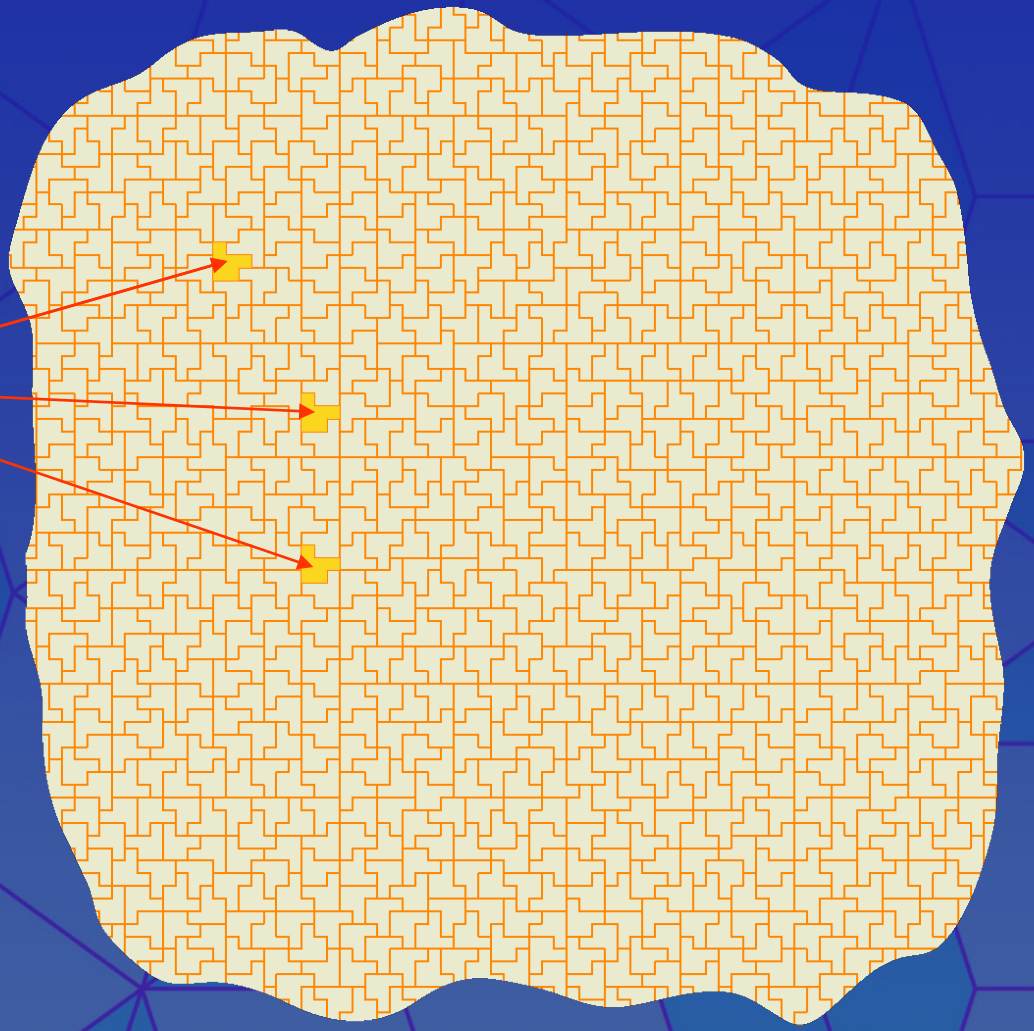
- *Fills the entire plane non-periodically*
- *Self-similarity*



Infinite plane

81-rep G-hexominoes

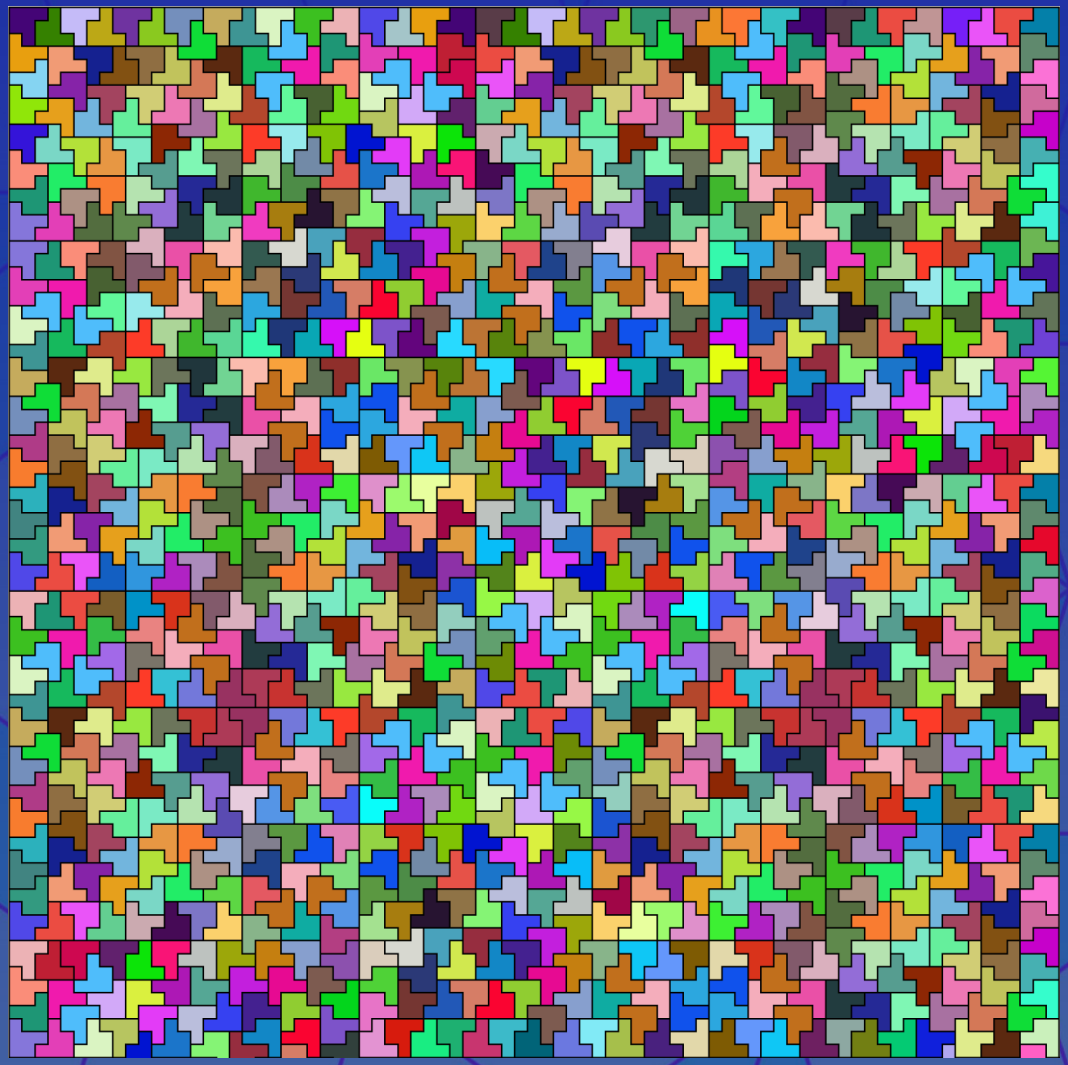
Different neighborhoods



Infinite plane

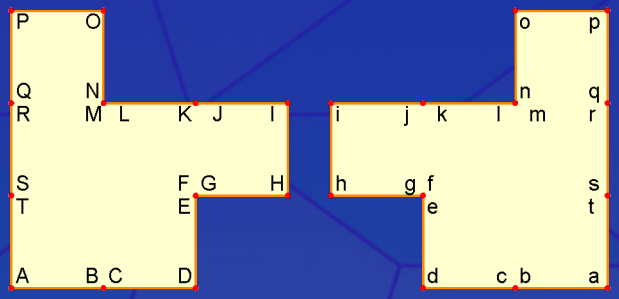
➔ *Problem of finding geometrical neighborhoods (structural indices)*

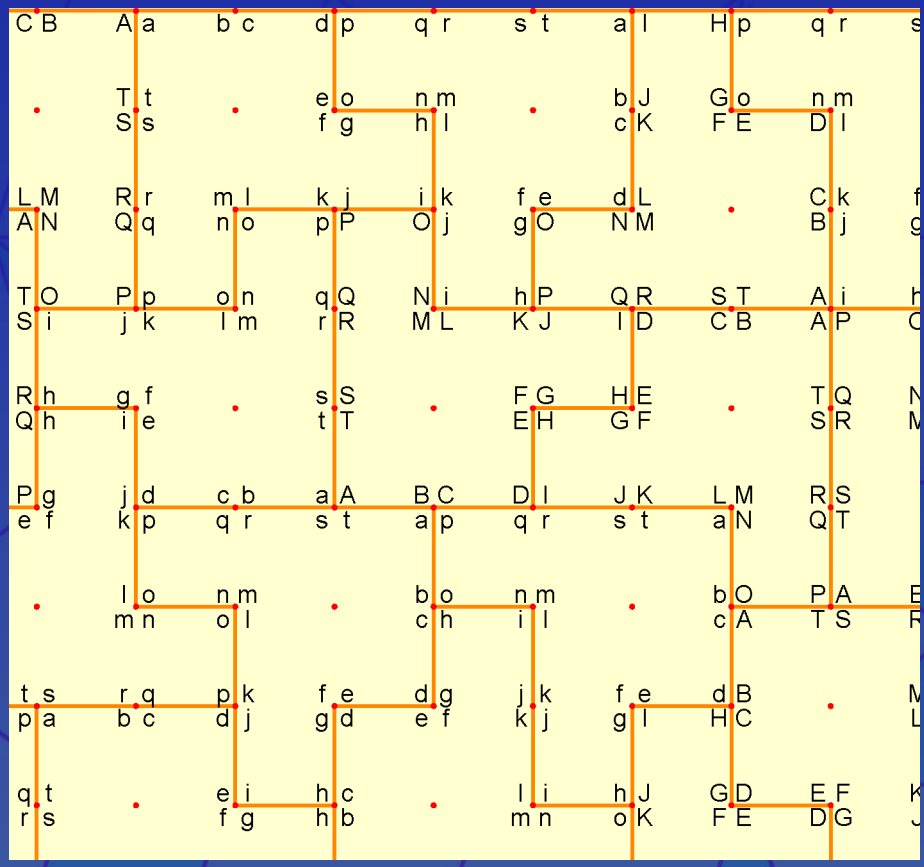
Apparent Chaos



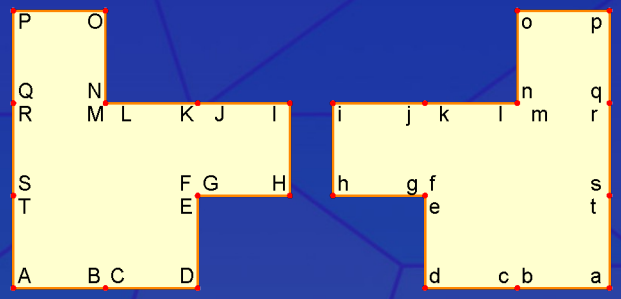
Different colors =
Different neighborhoods =
Different structural indices

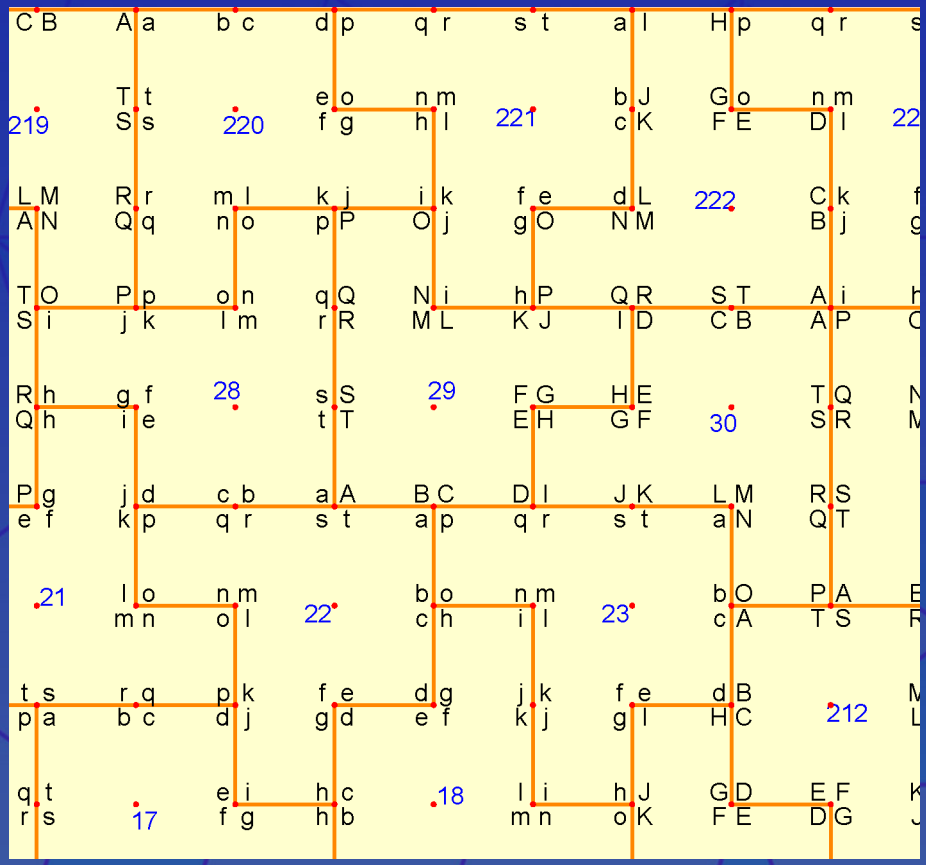
Finding structural indices



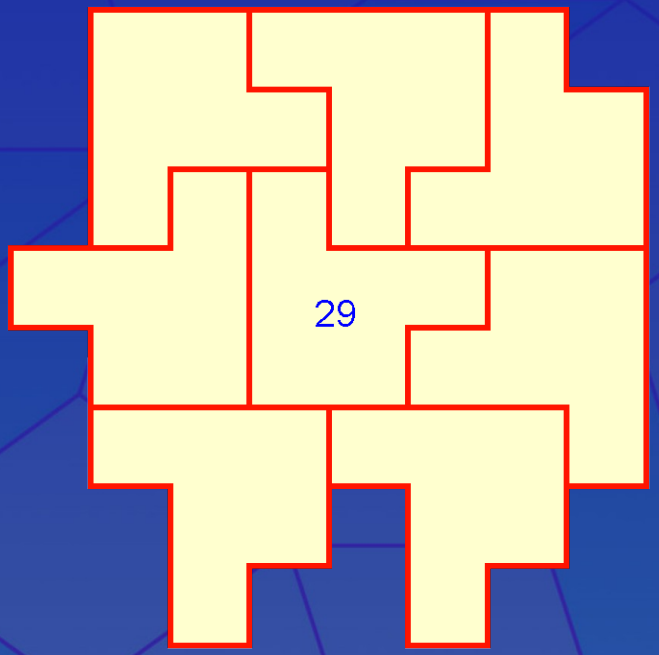


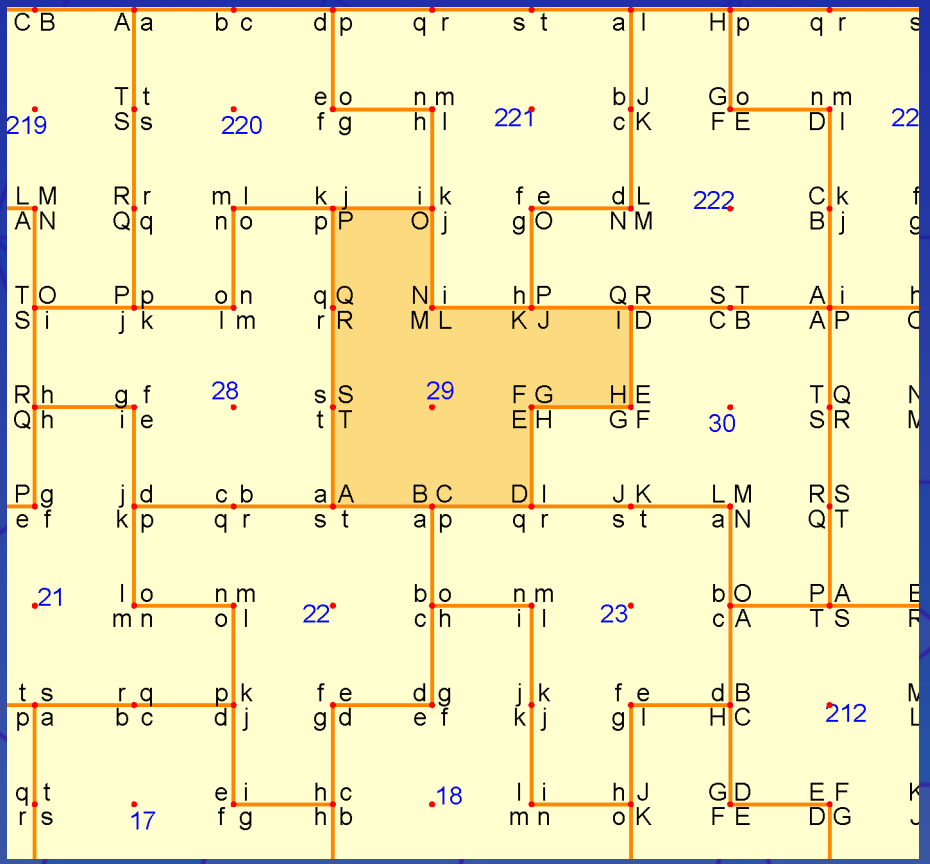
Finding structural indices



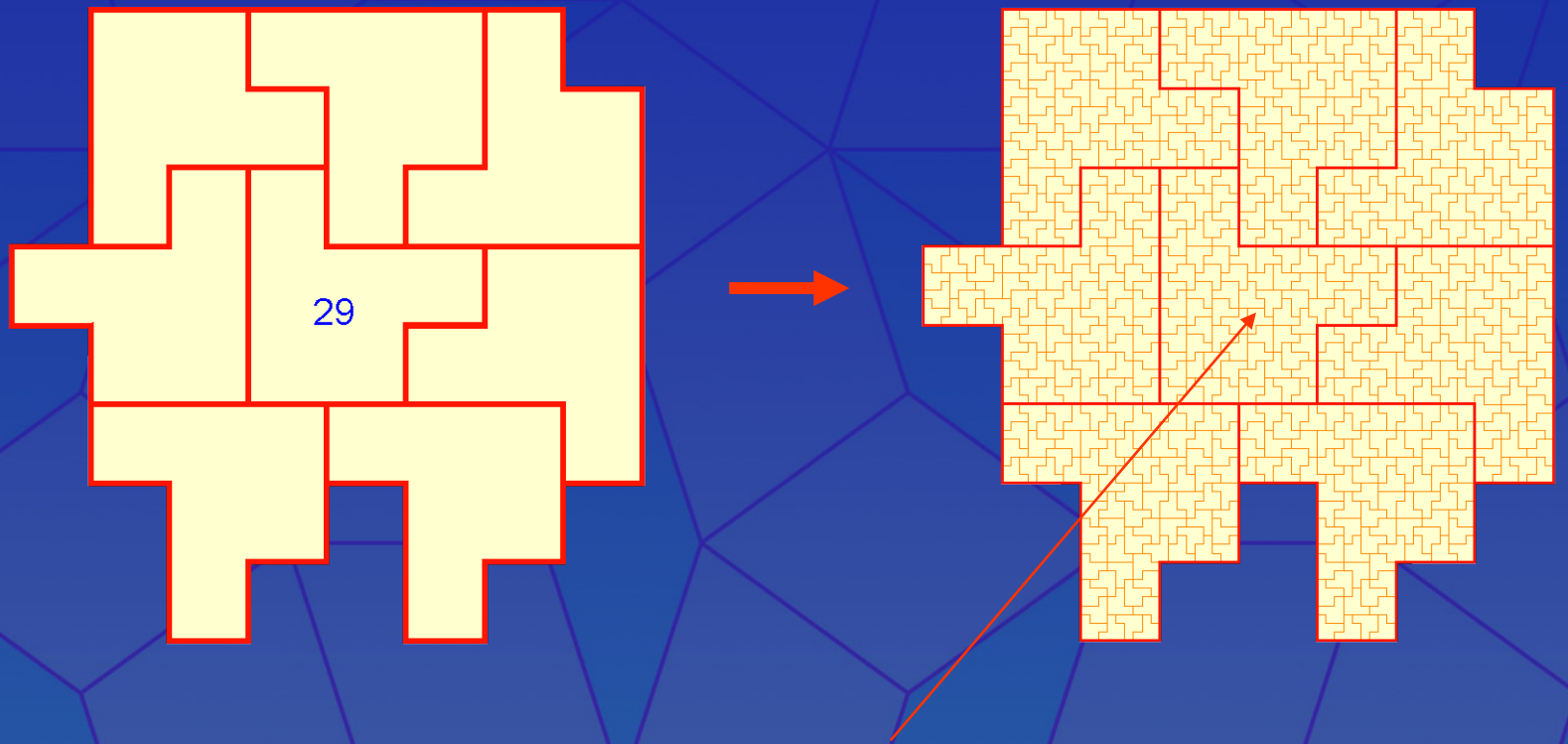


Finding structural indices

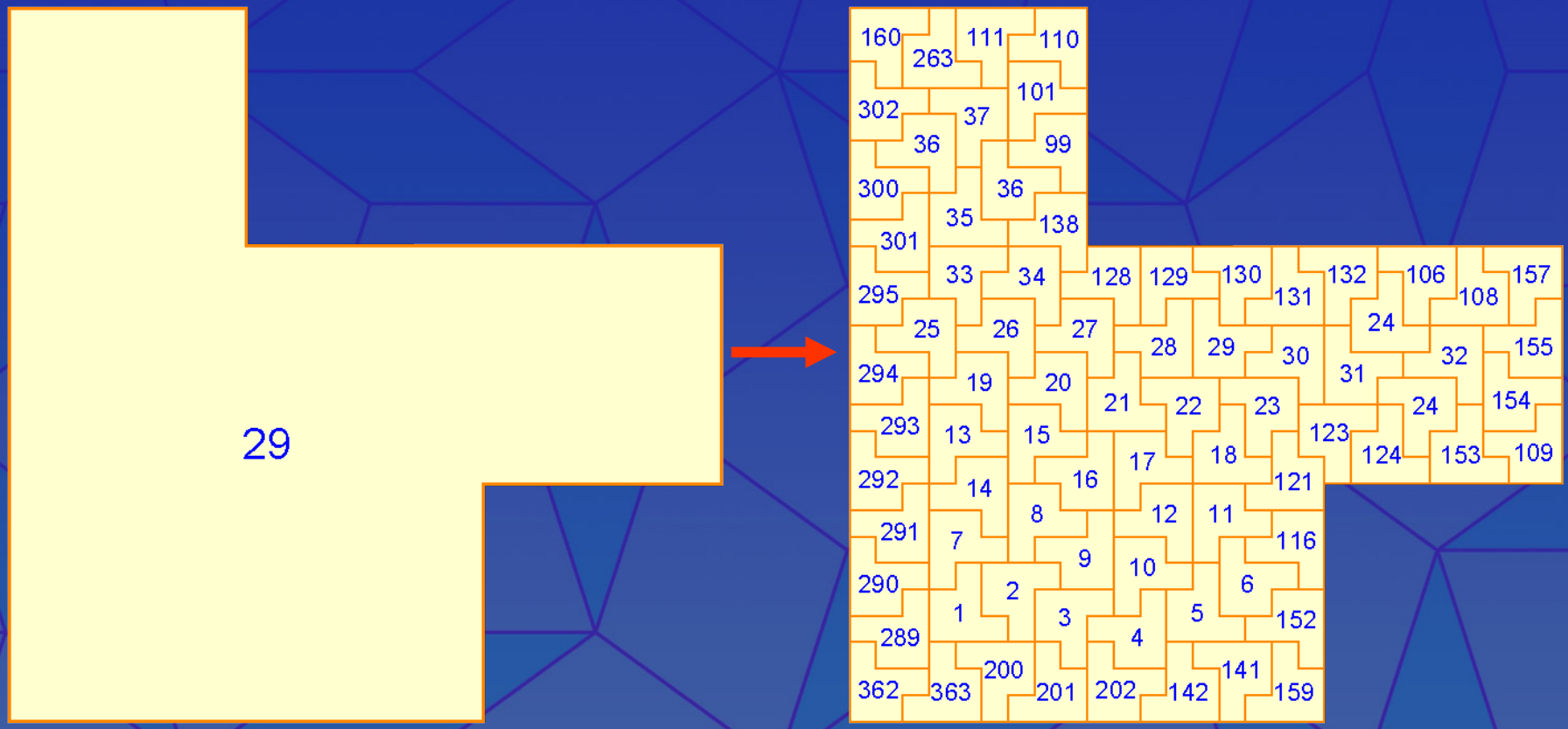




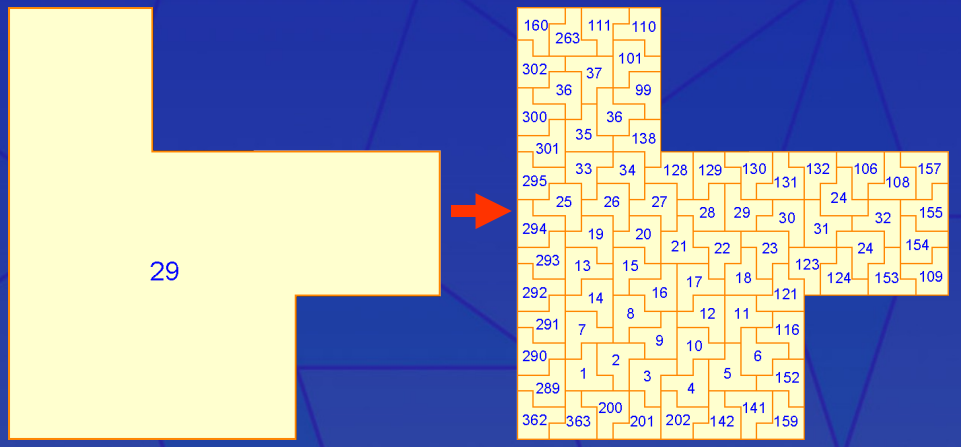
Finding structural indices



Finding structural indices



Optimization



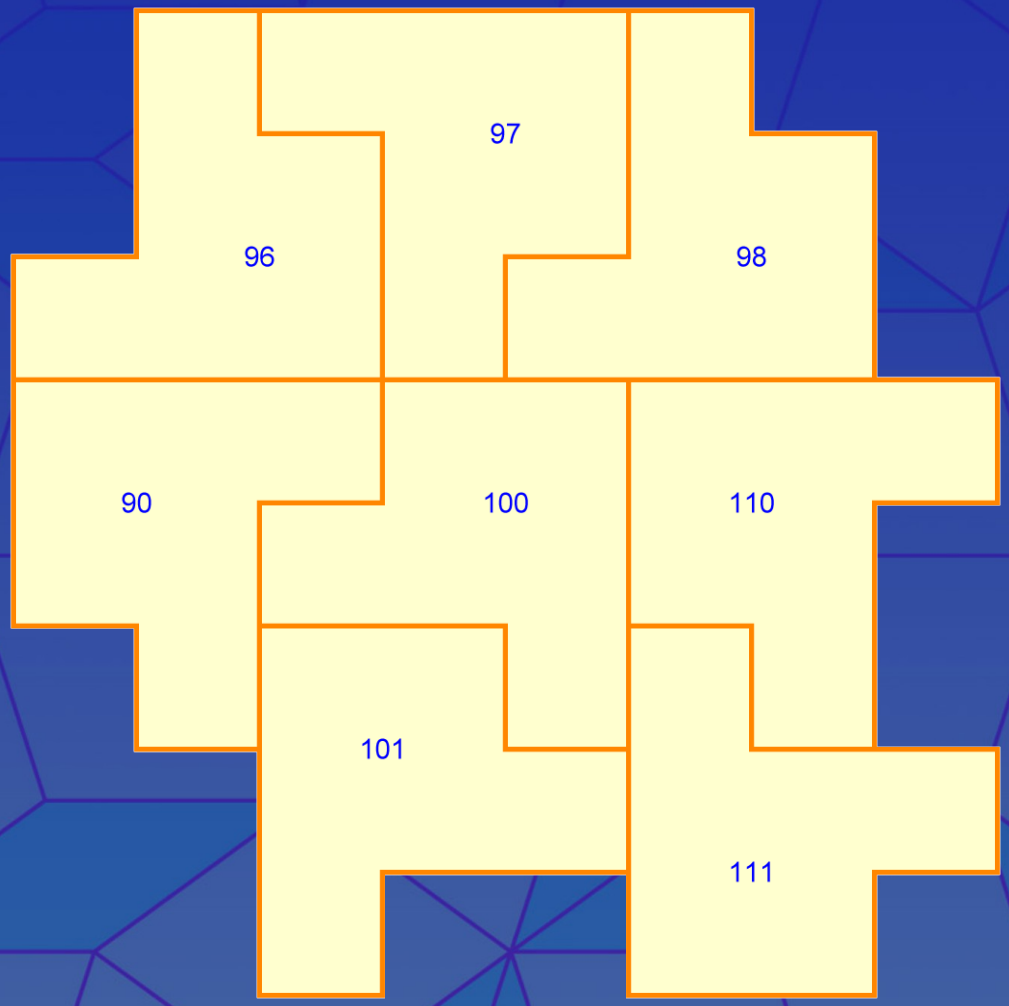
Decomposition process:

- *Production rules*
- *Structural indices characterize local geometrical neighborhoods*
- *Mapping of structural indices (finite number)*

➔ *Tiles with similar geometrical neighborhoods (structural indices) behave similarly*

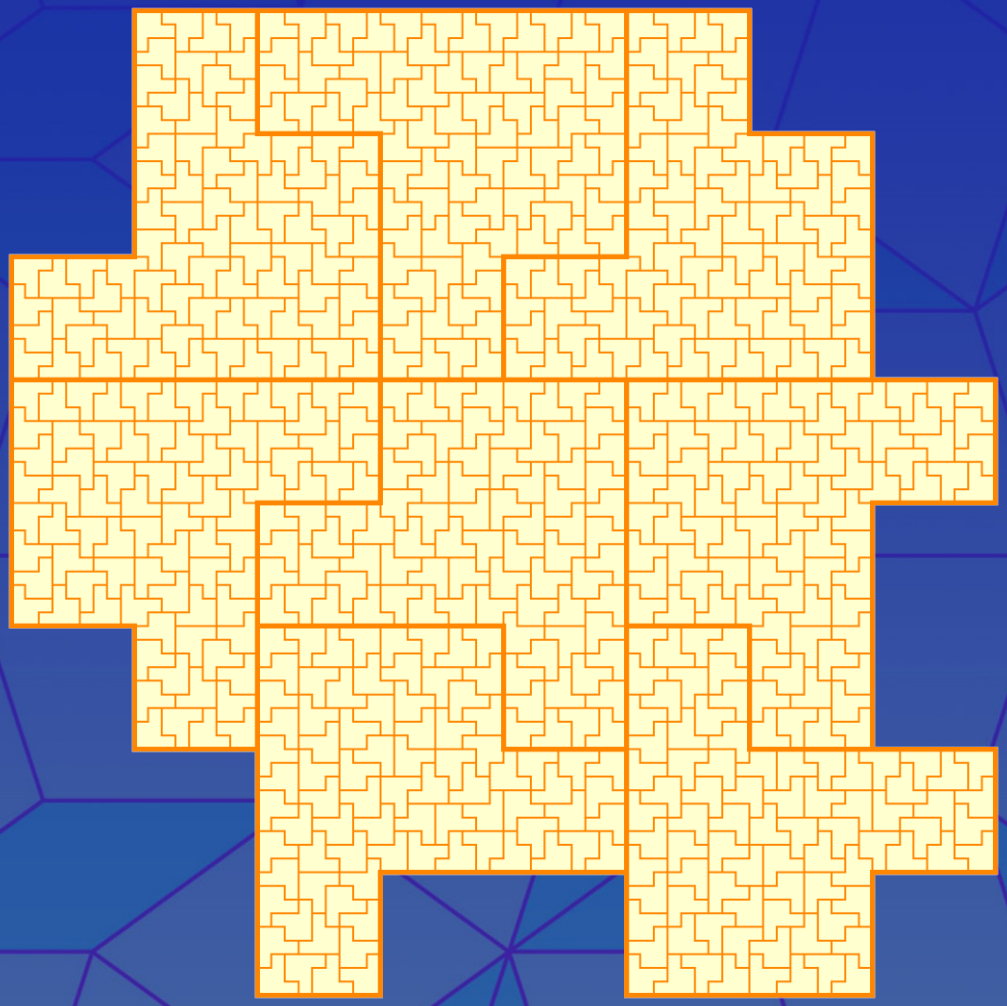
Optimization I

Patch



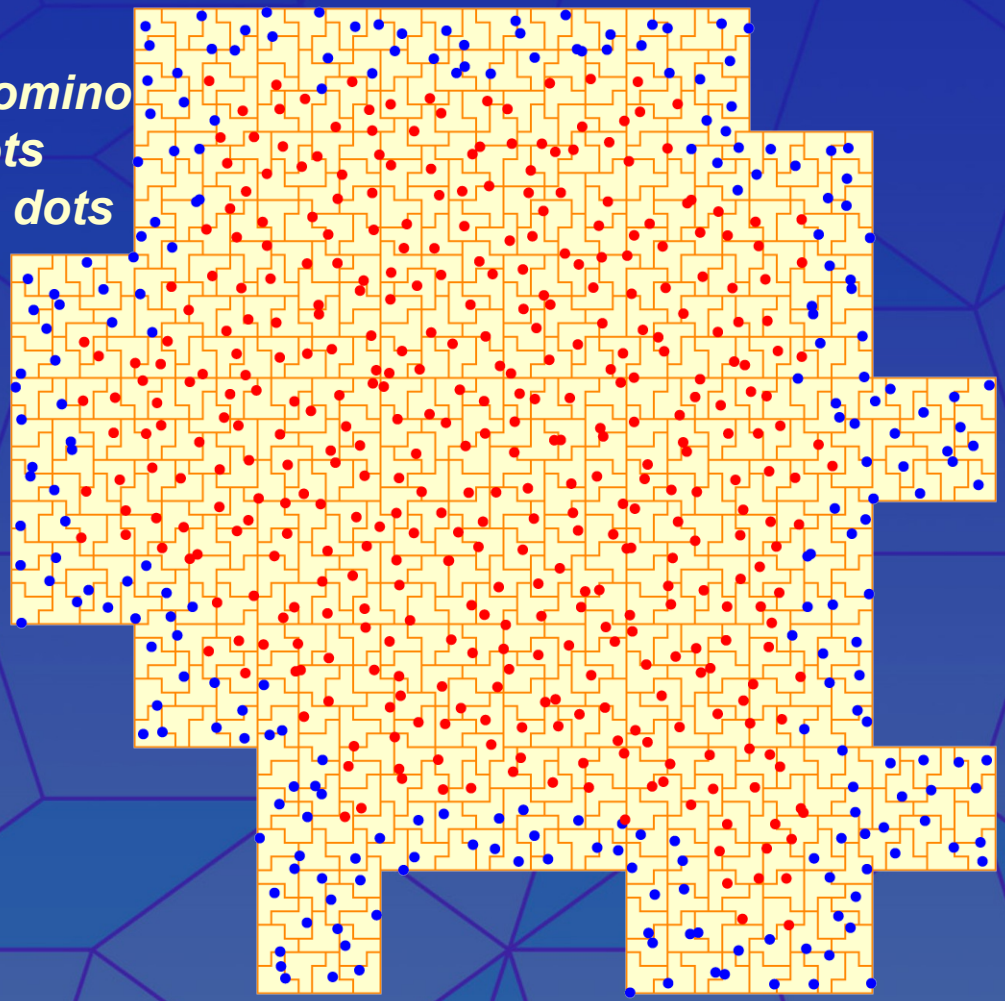
Optimization I

Subdivision



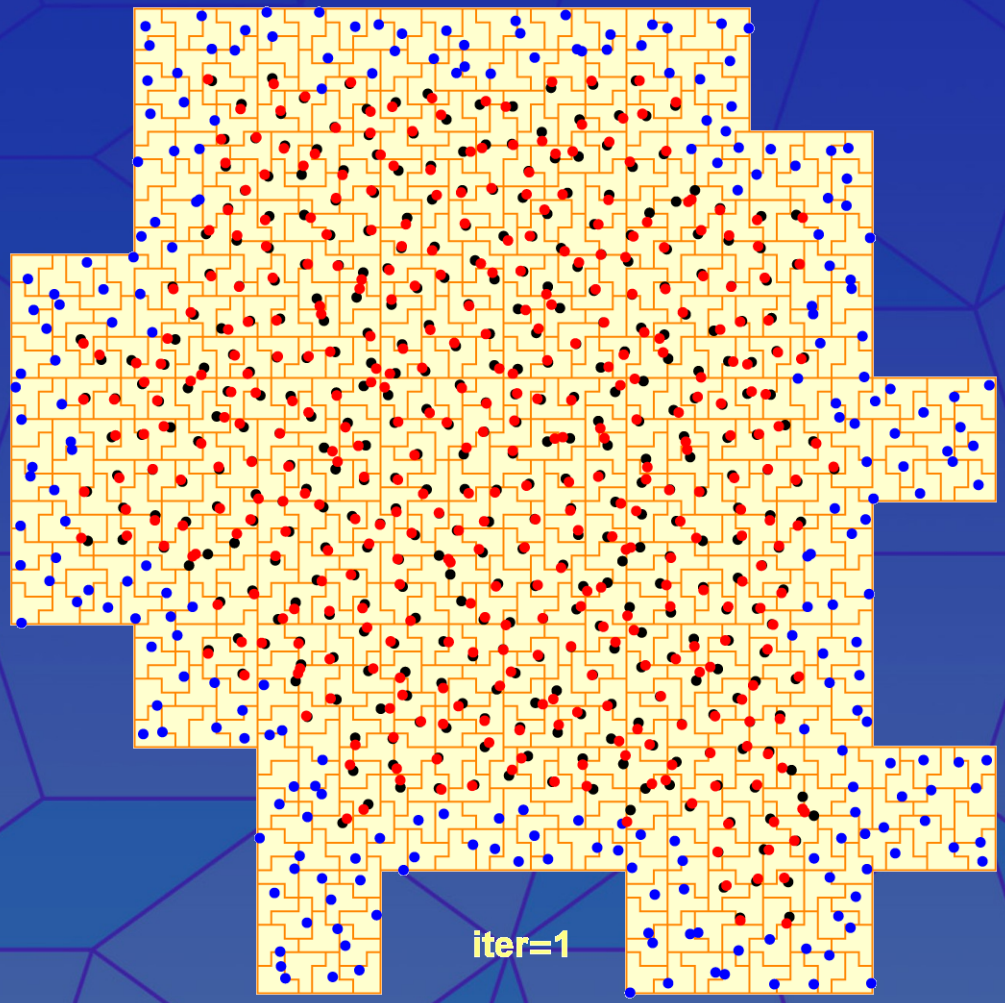
Optimization I

Random dots
within each polyomino
Red: movable dots
Blue: immovable dots



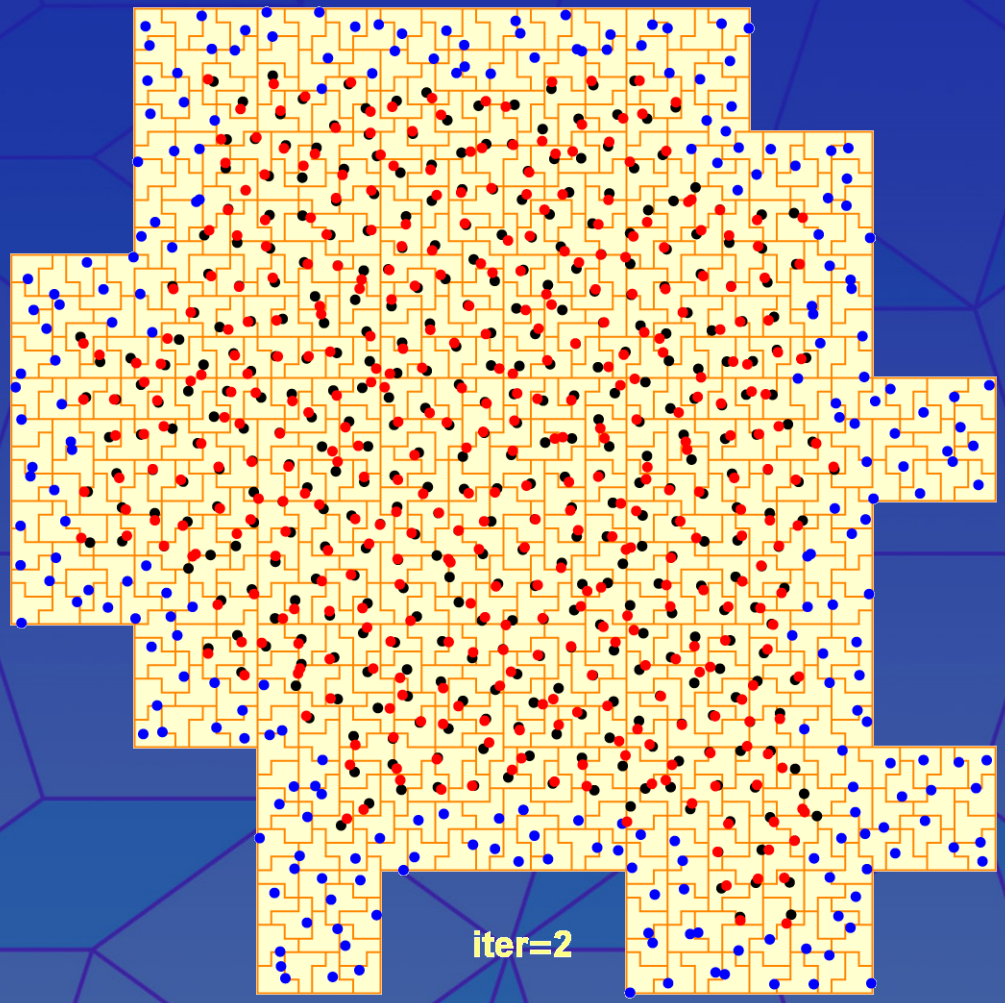
Optimization I

Relaxation



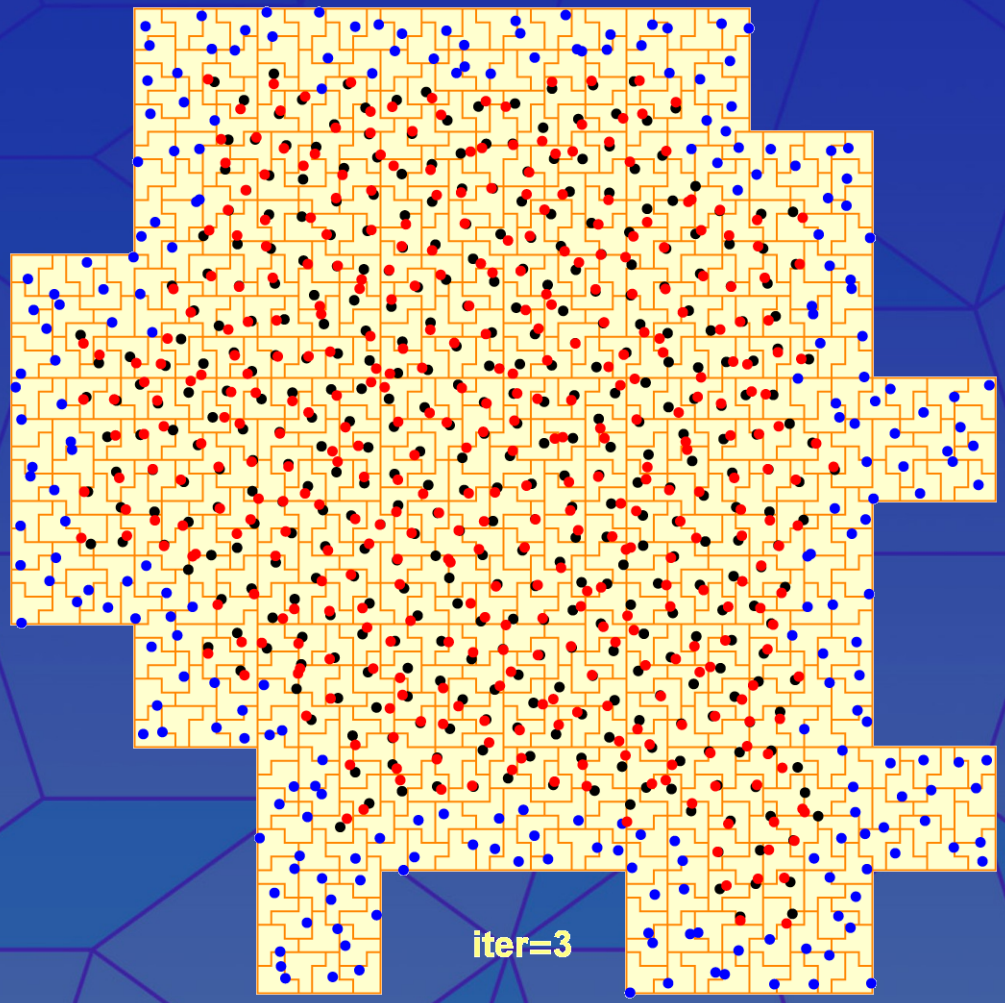
Optimization I

Relaxation



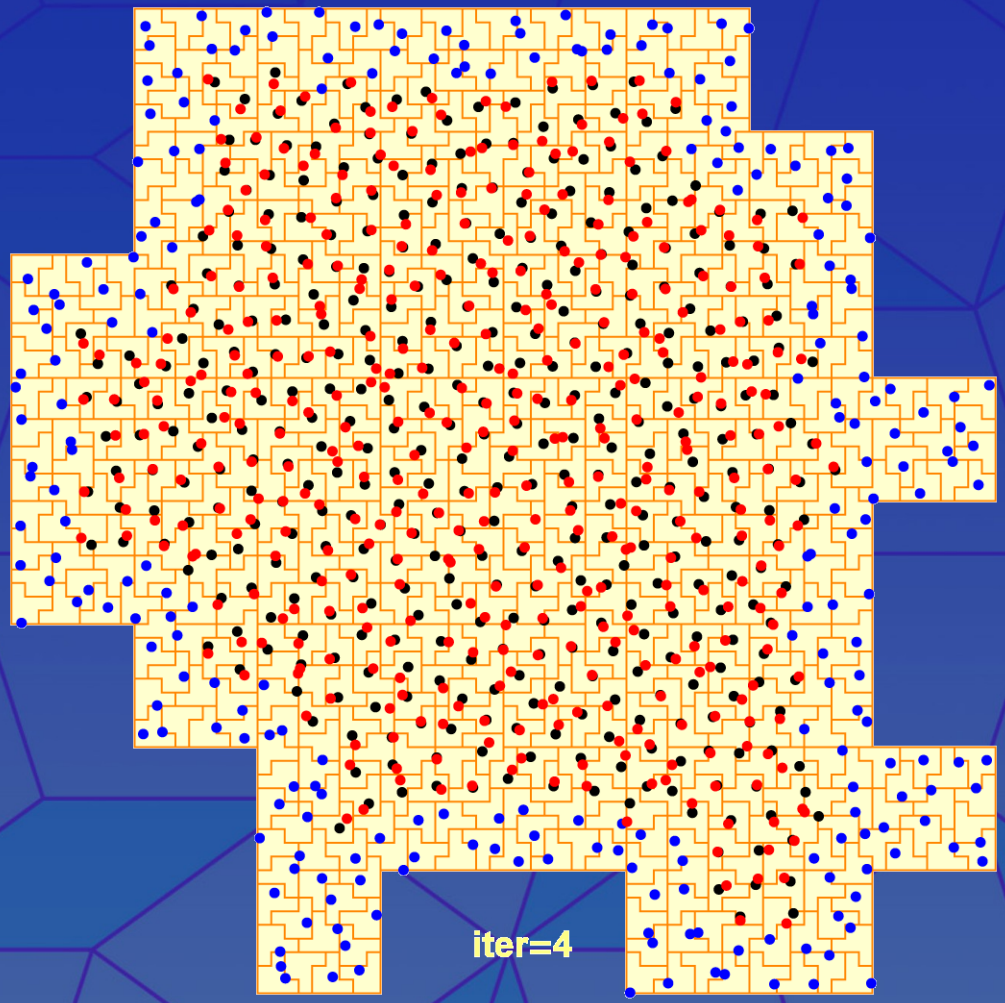
Optimization I

Relaxation



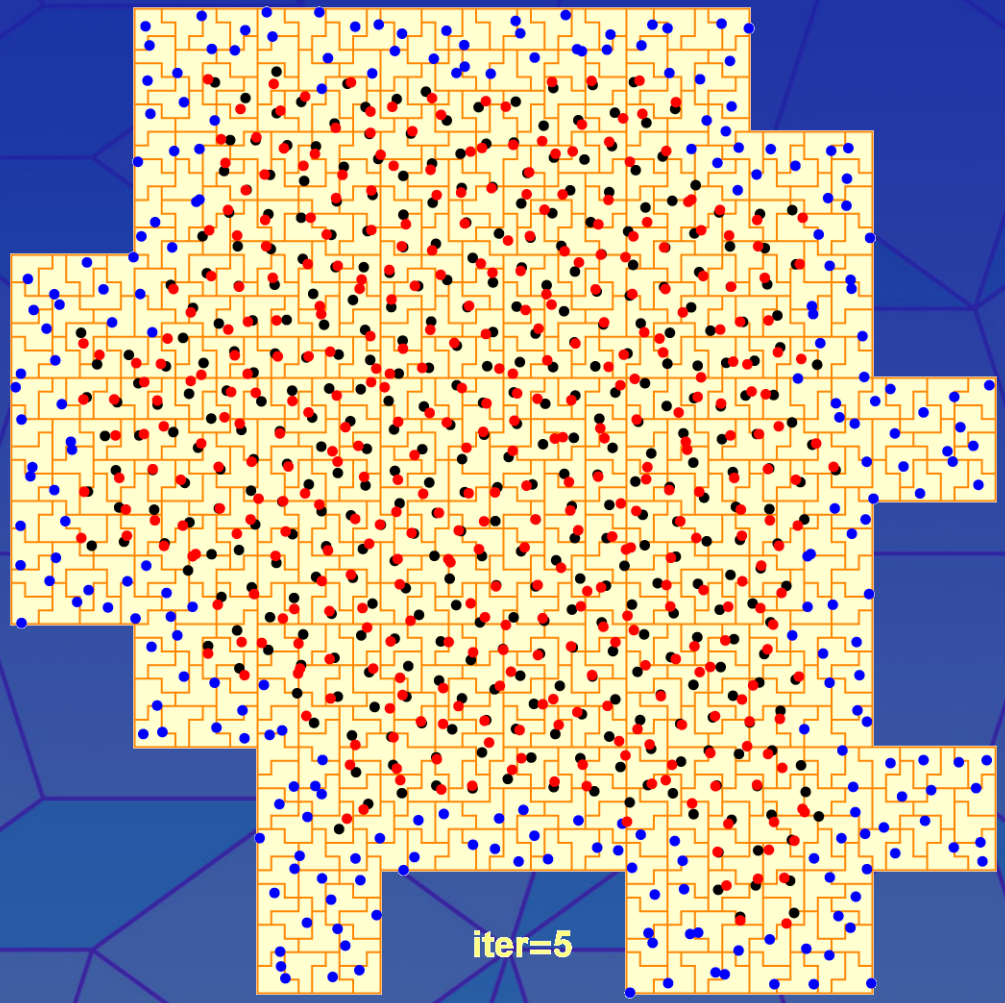
Optimization I

Relaxation



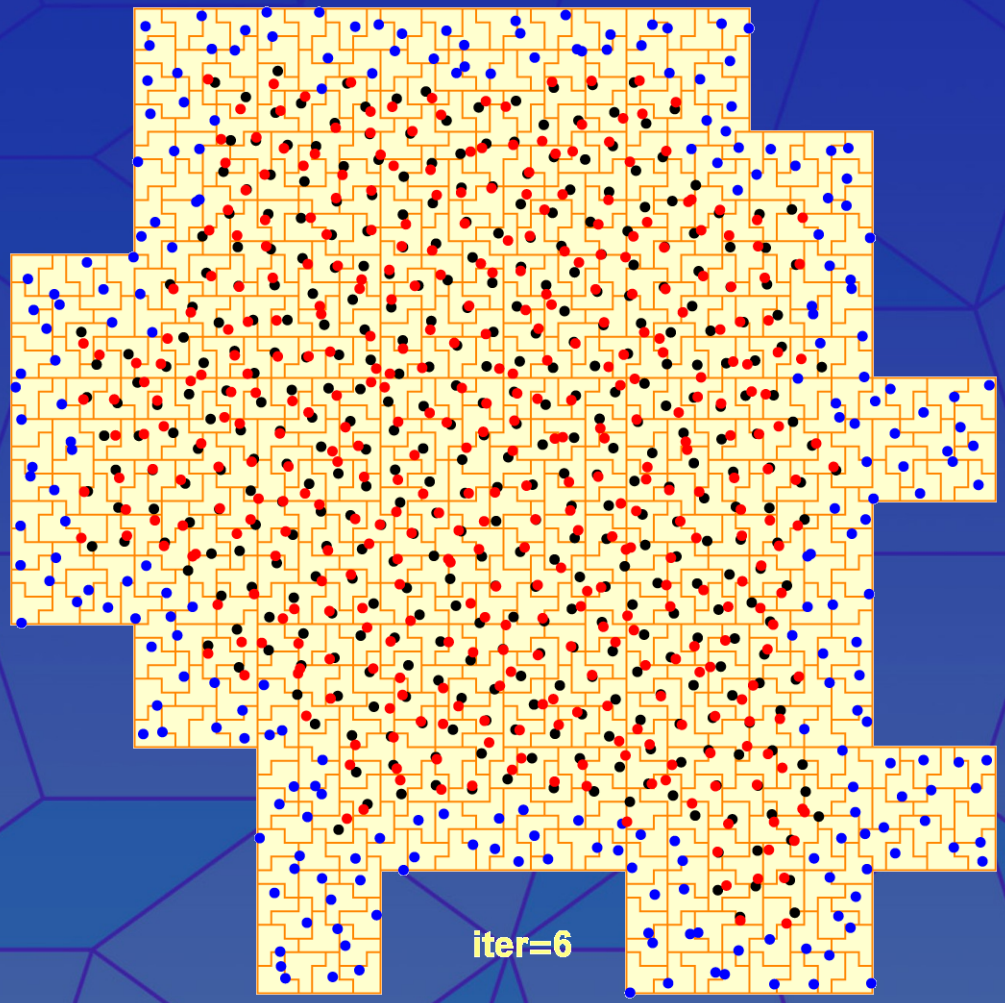
Optimization I

Relaxation



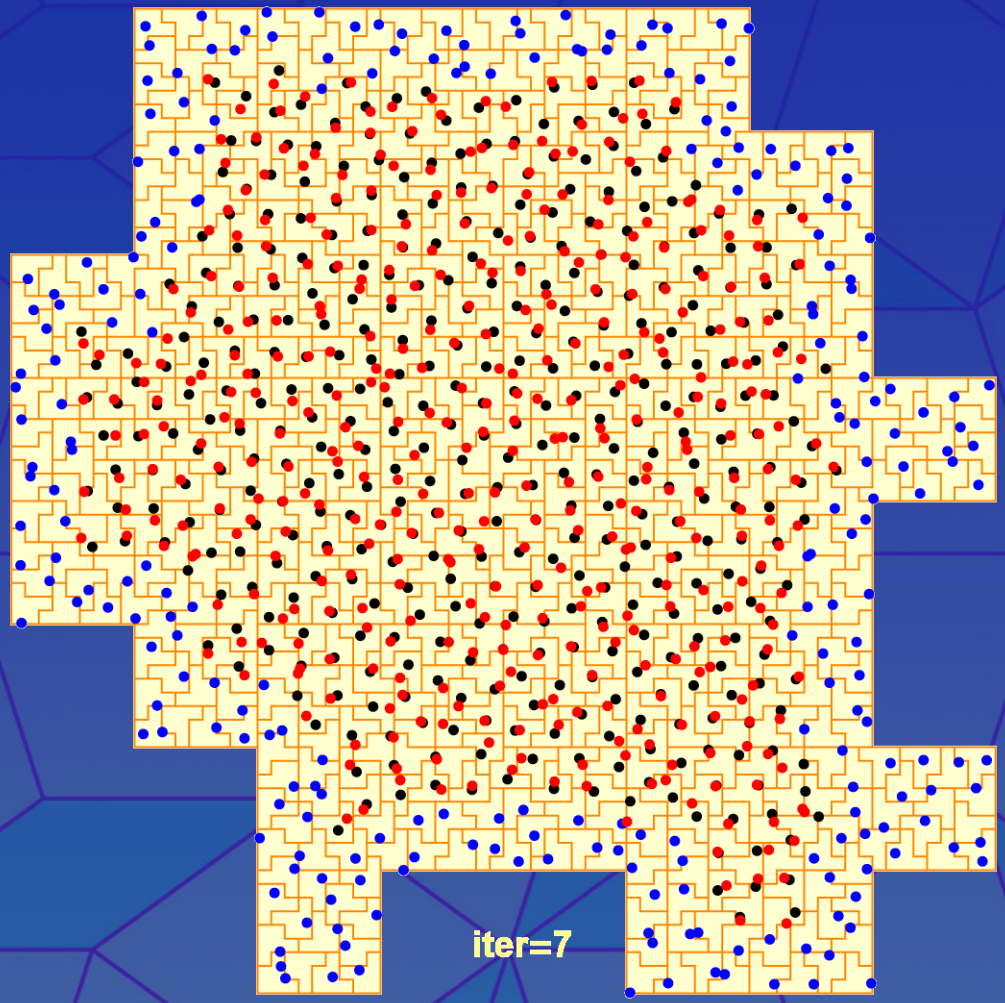
Optimization I

Relaxation



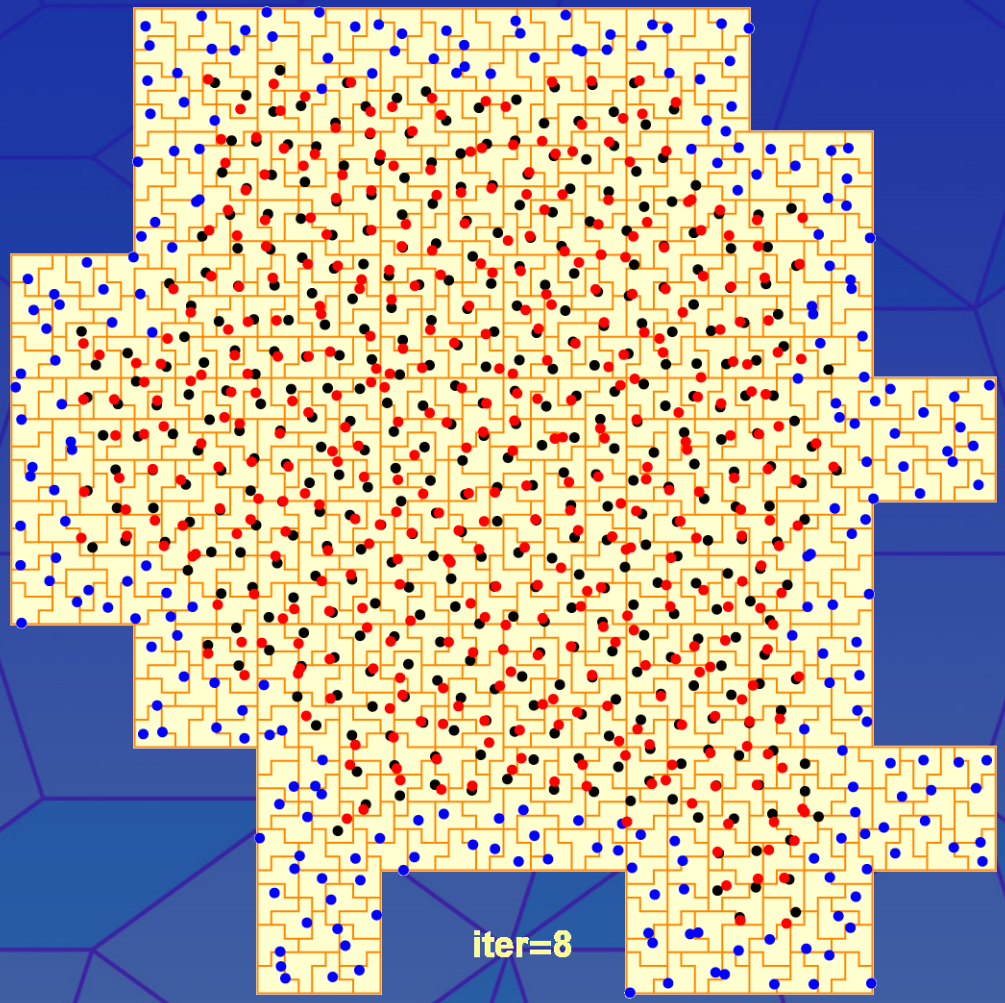
Optimization I

Relaxation



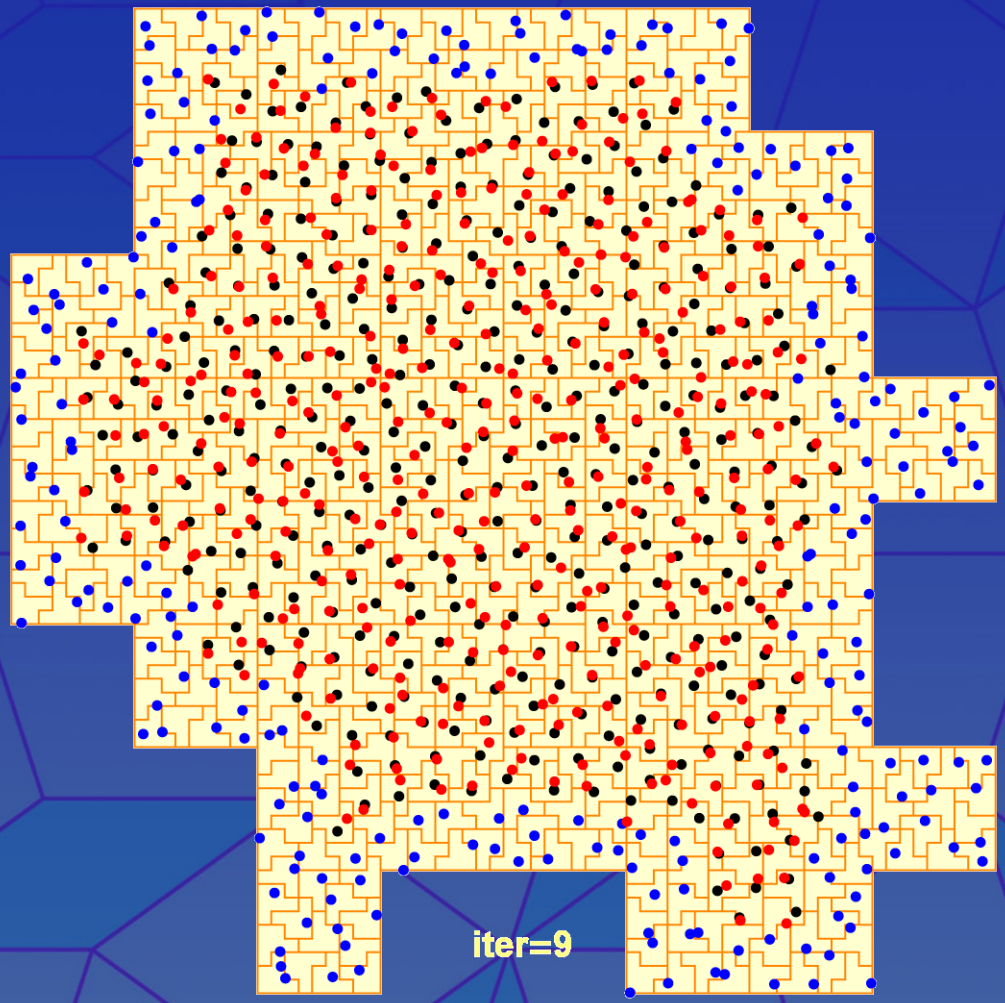
Optimization I

Relaxation



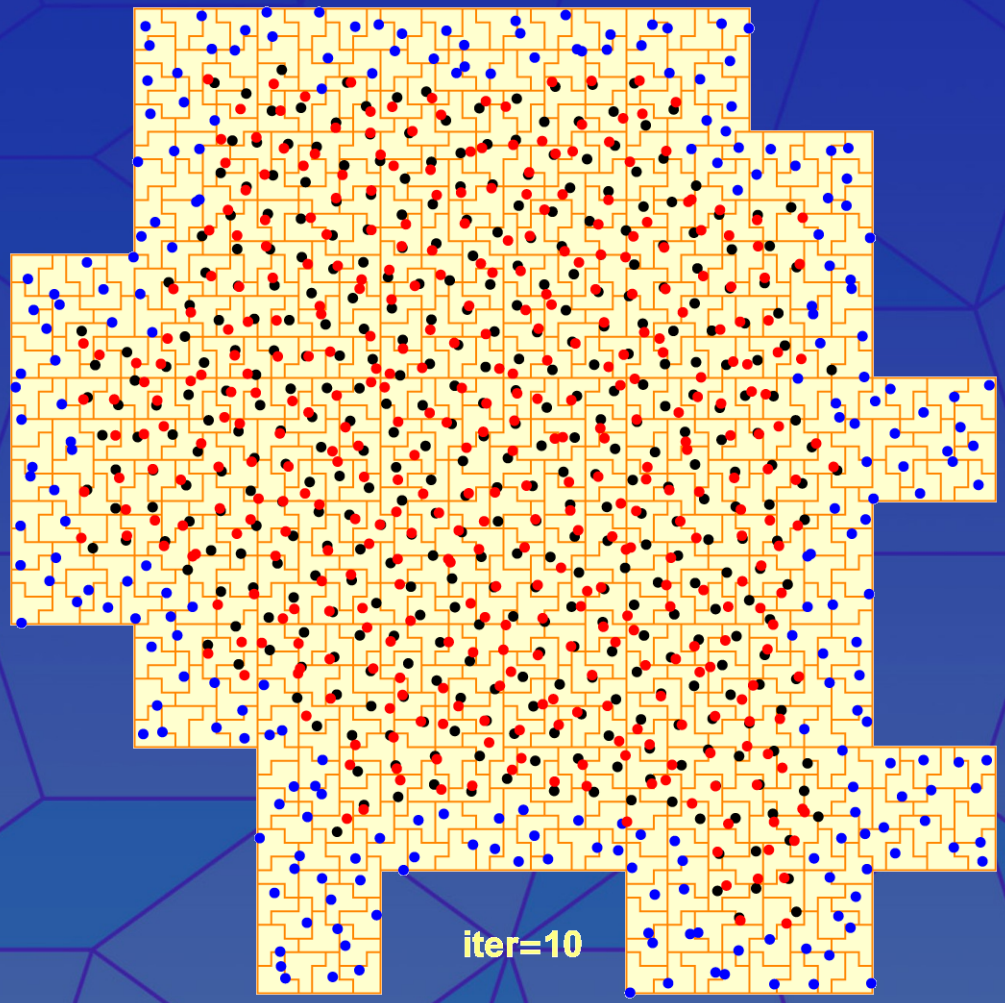
Optimization I

Relaxation



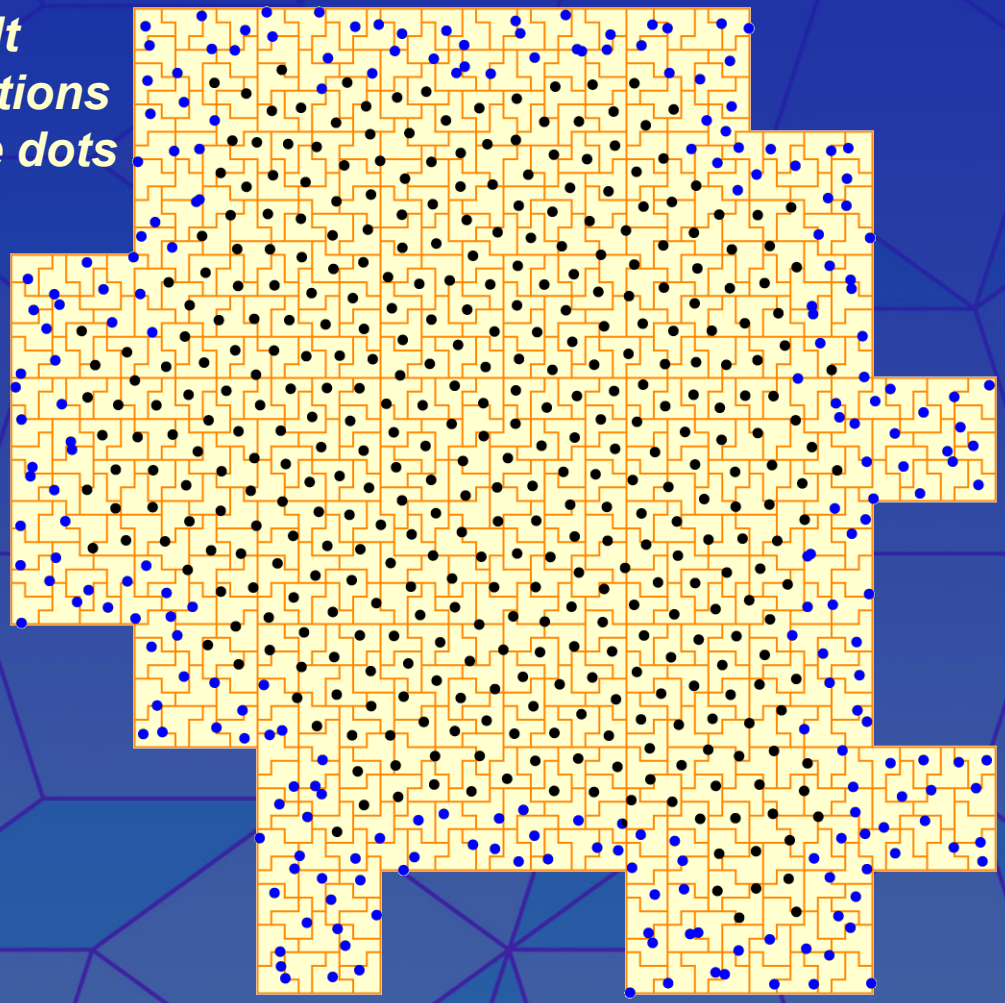
Optimization I

Relaxation



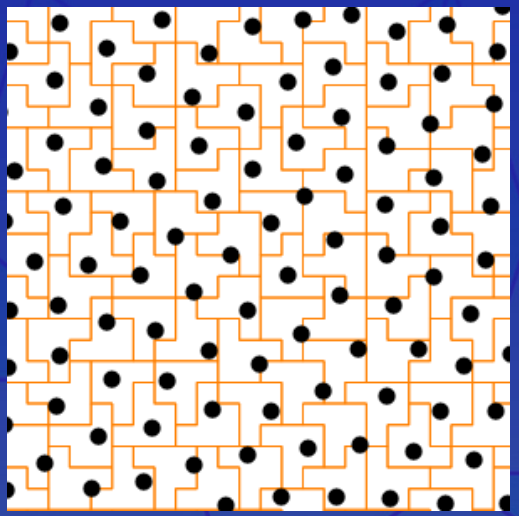
Optimization I

Black: final result
after 10 relaxations
Blue: immovable dots

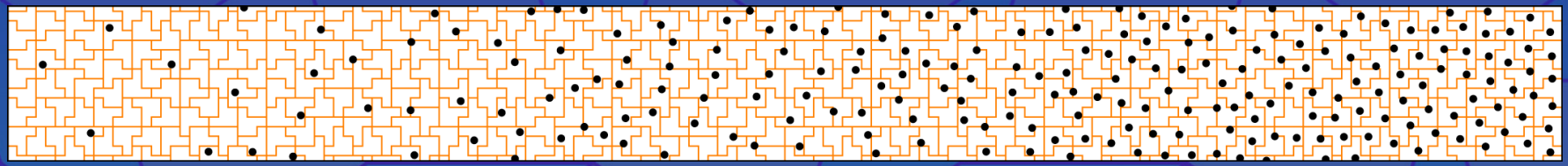


Optimization in two phases

- *Optimization I: finding optimal position of the sampling point within each polyomino, for constant density*



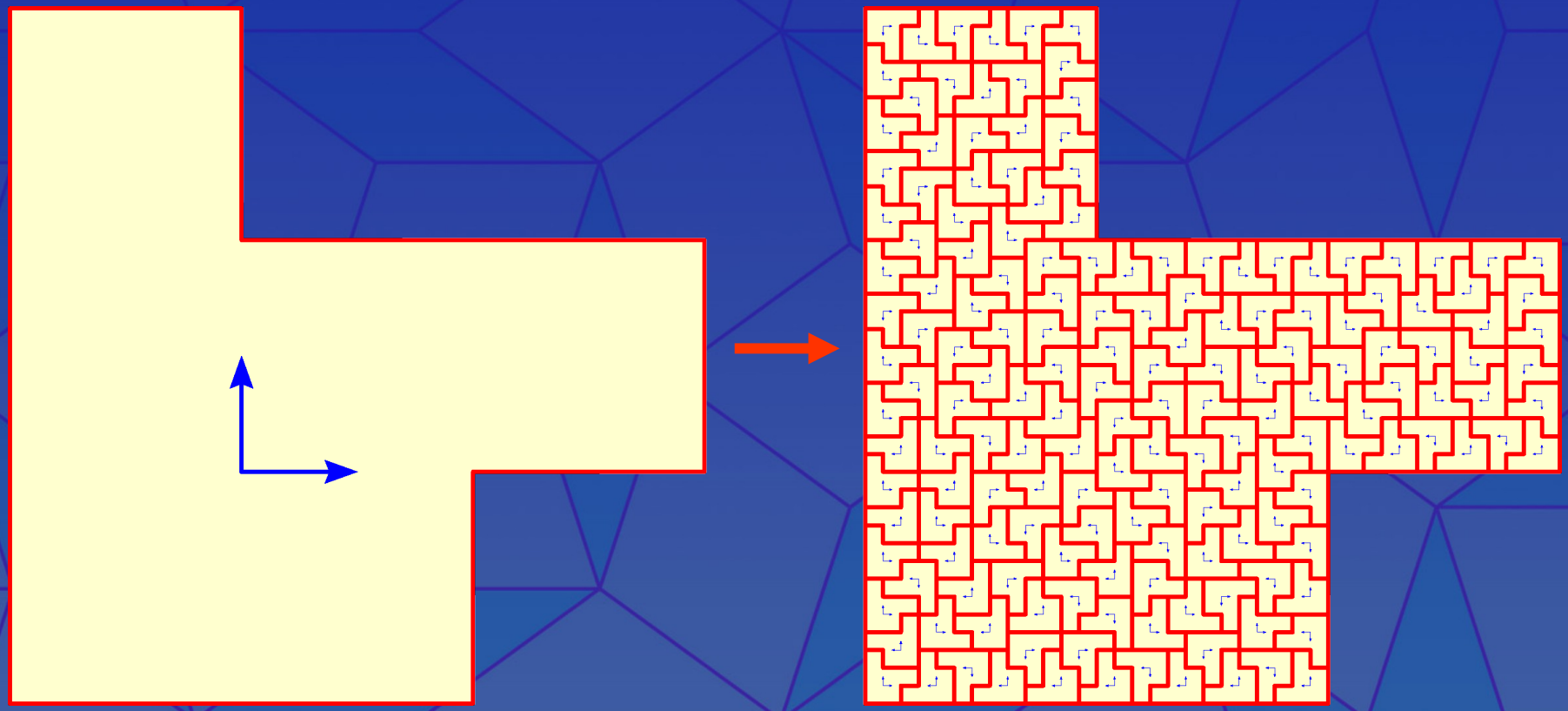
- *Optimization II: ranking and relaxation*



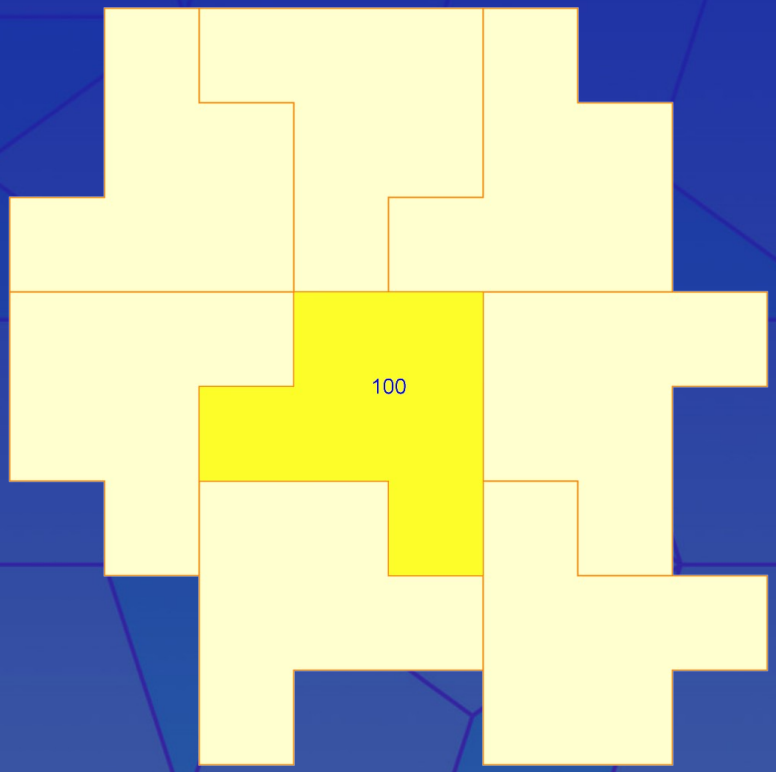
Ref: "The Void-and-cluster Method for Dither Array Generation" [Ulichney'93]

Optimization II-ranking

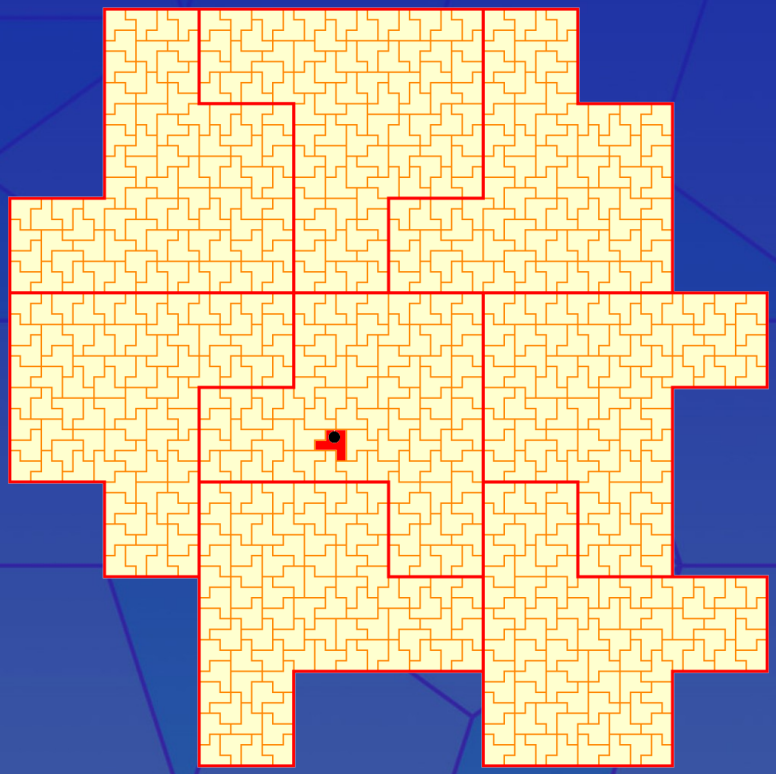
81-rep G-hexominoes



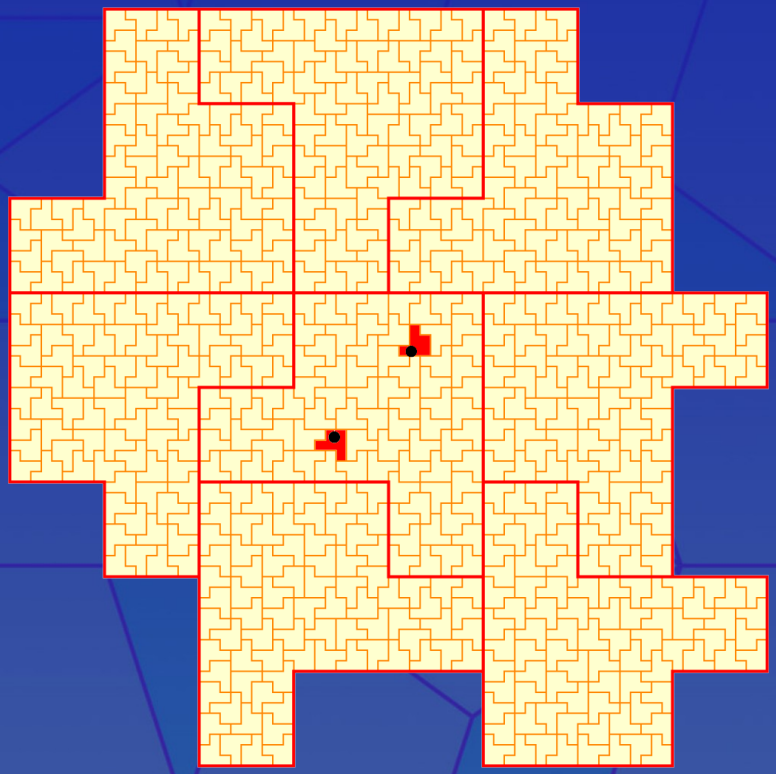
Optimization II-ranking



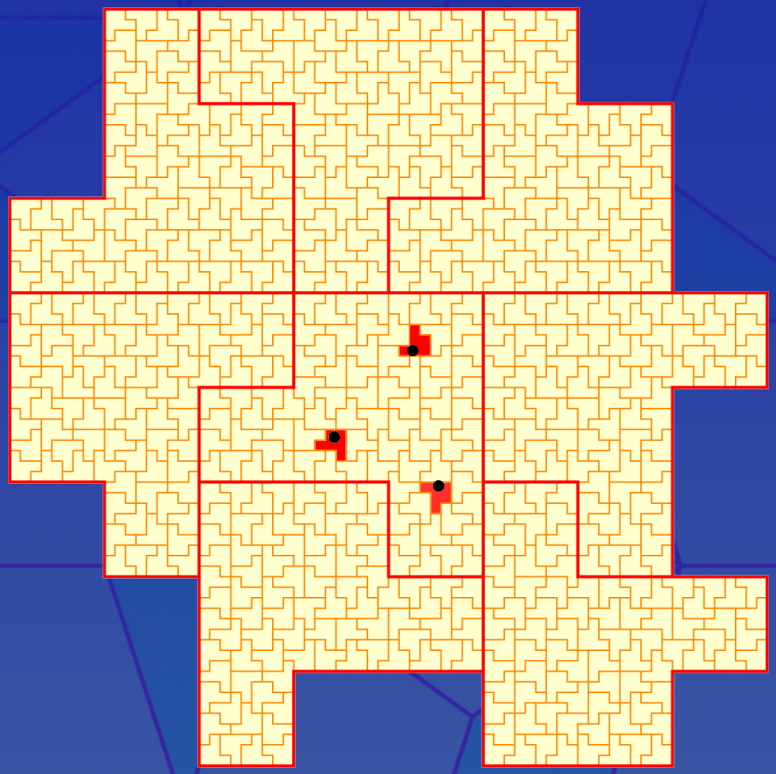
Optimization II-ranking



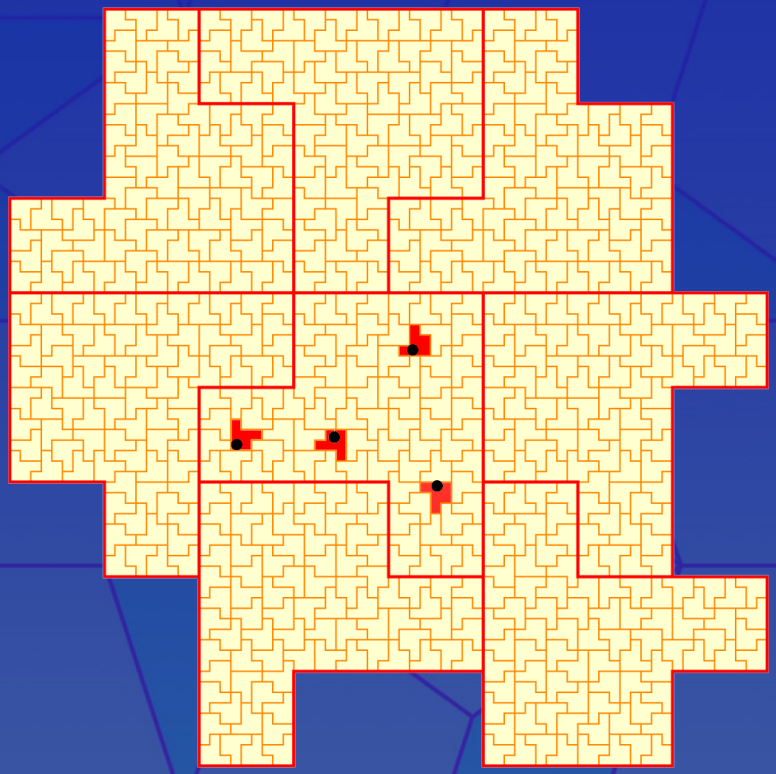
Optimization II-ranking



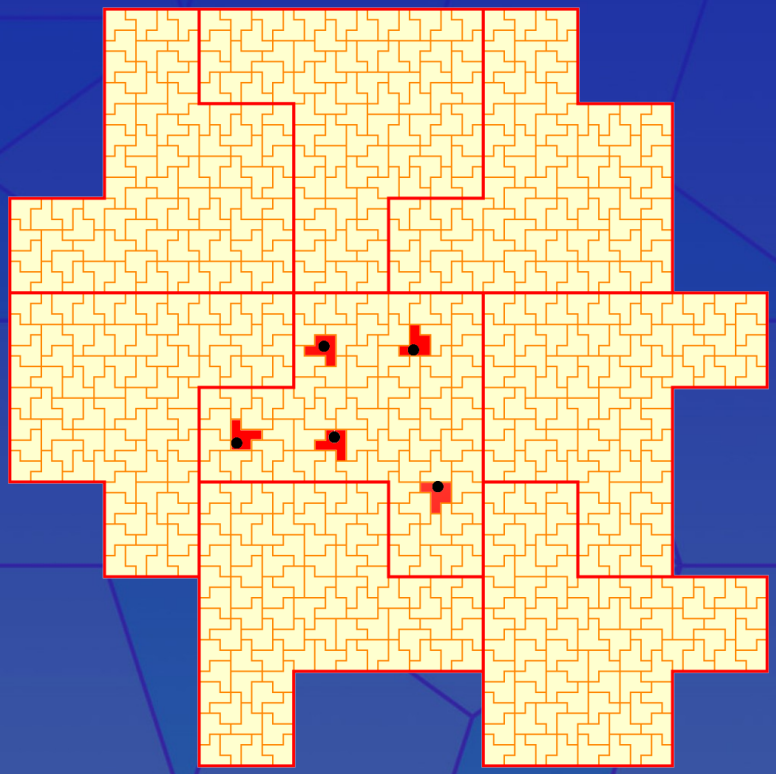
Optimization II-ranking



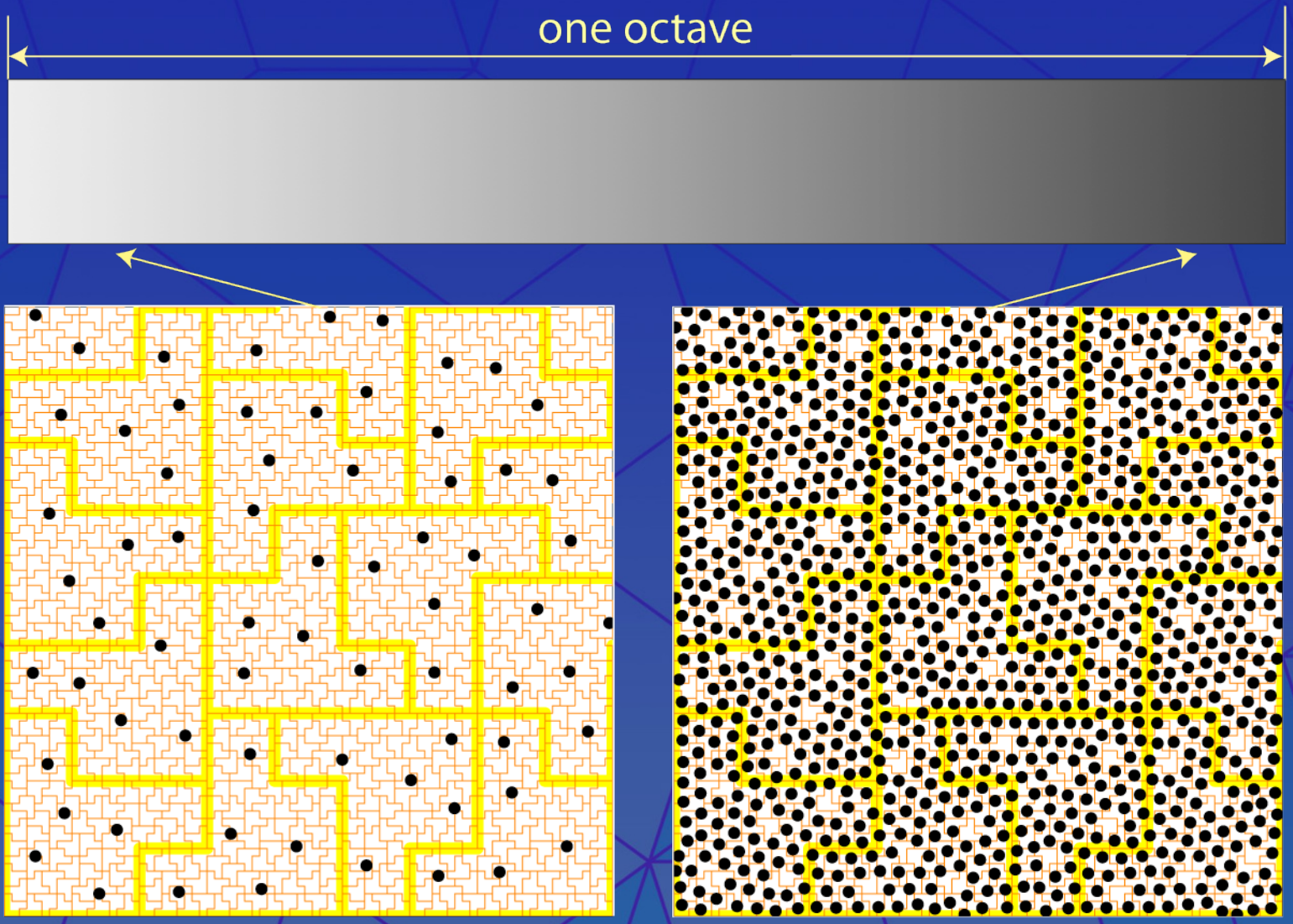
Optimization II-ranking



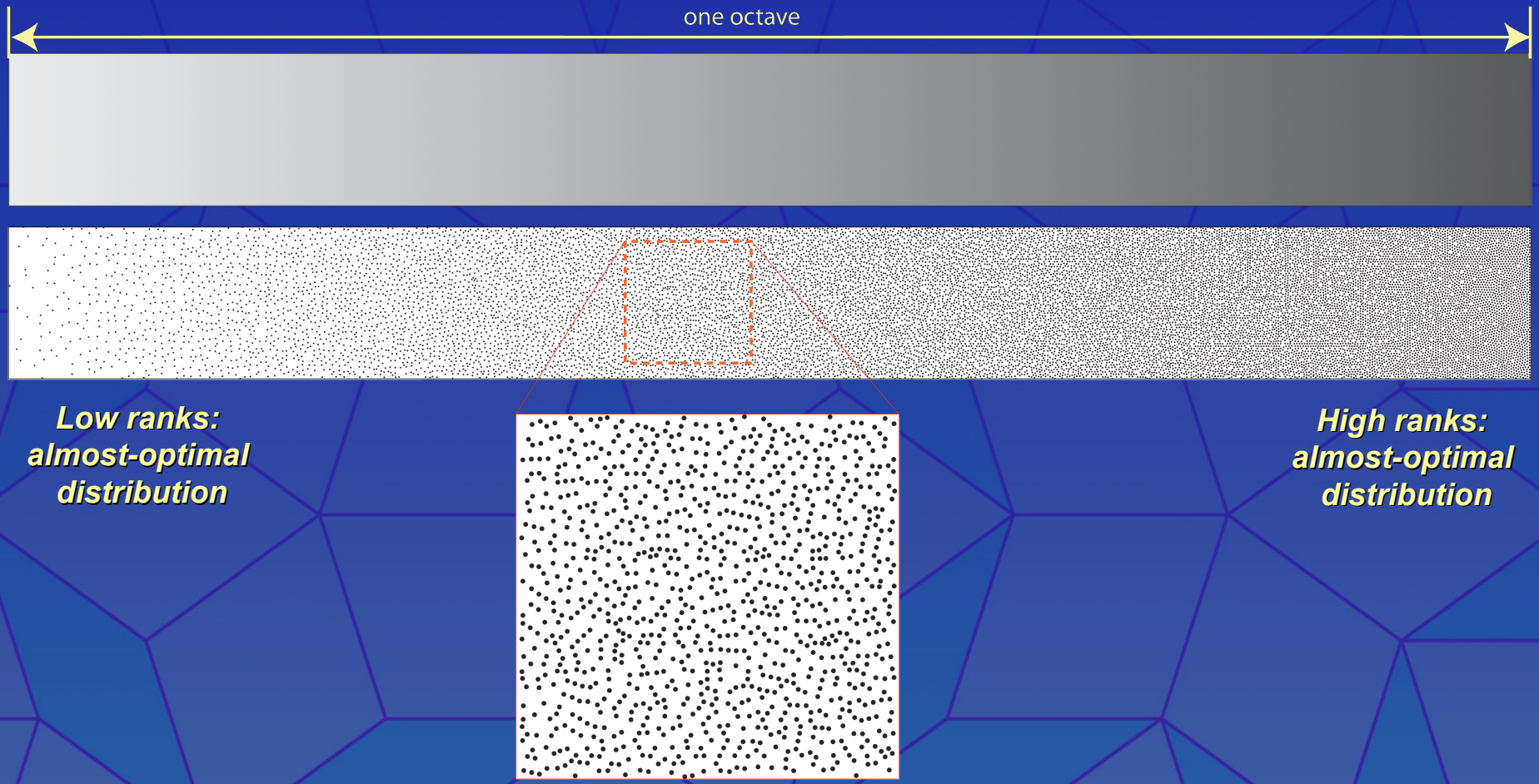
Optimization II-ranking



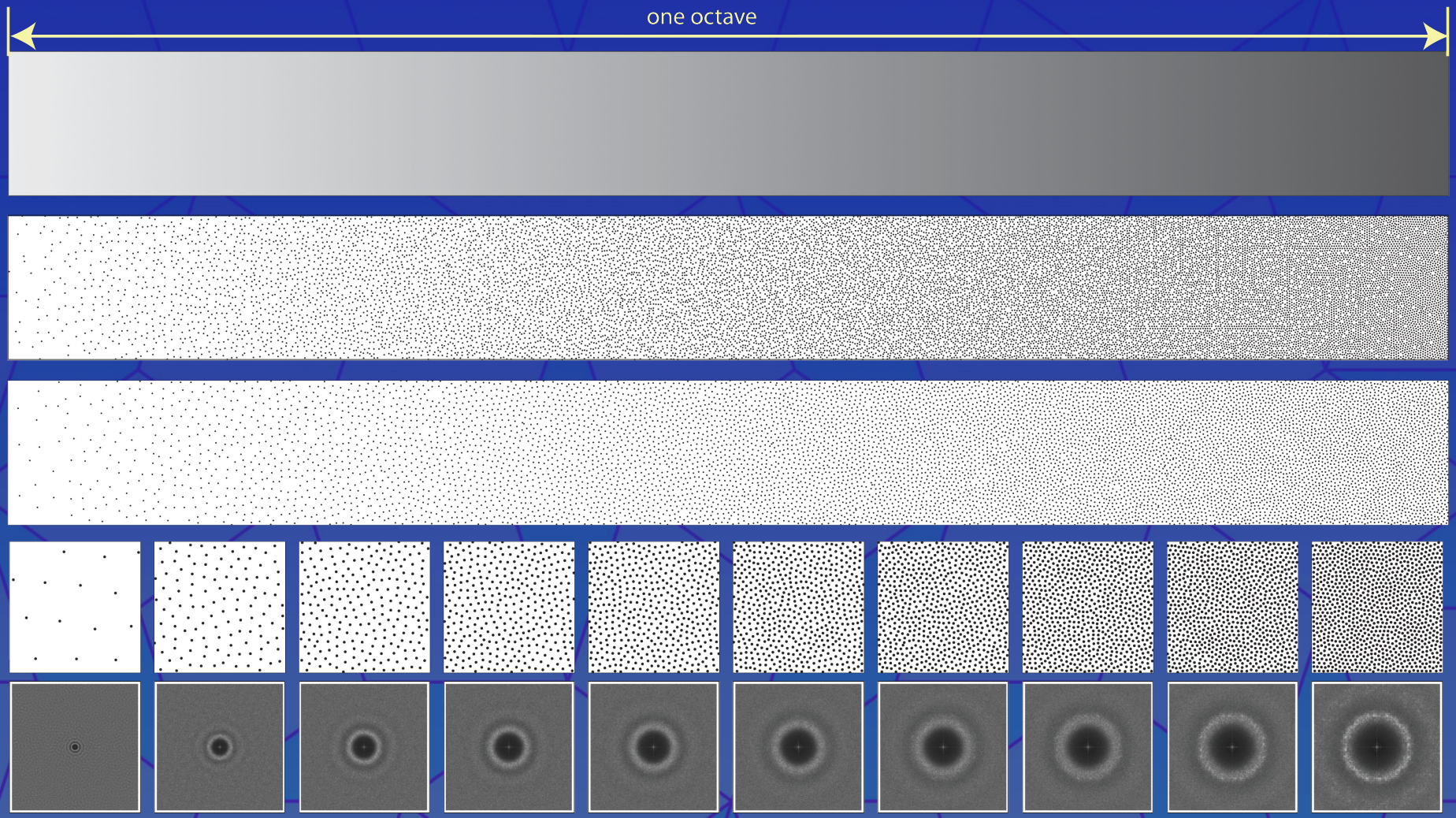
Optimization II-ranking



Optimization II-ranking



Optimization II-relaxation



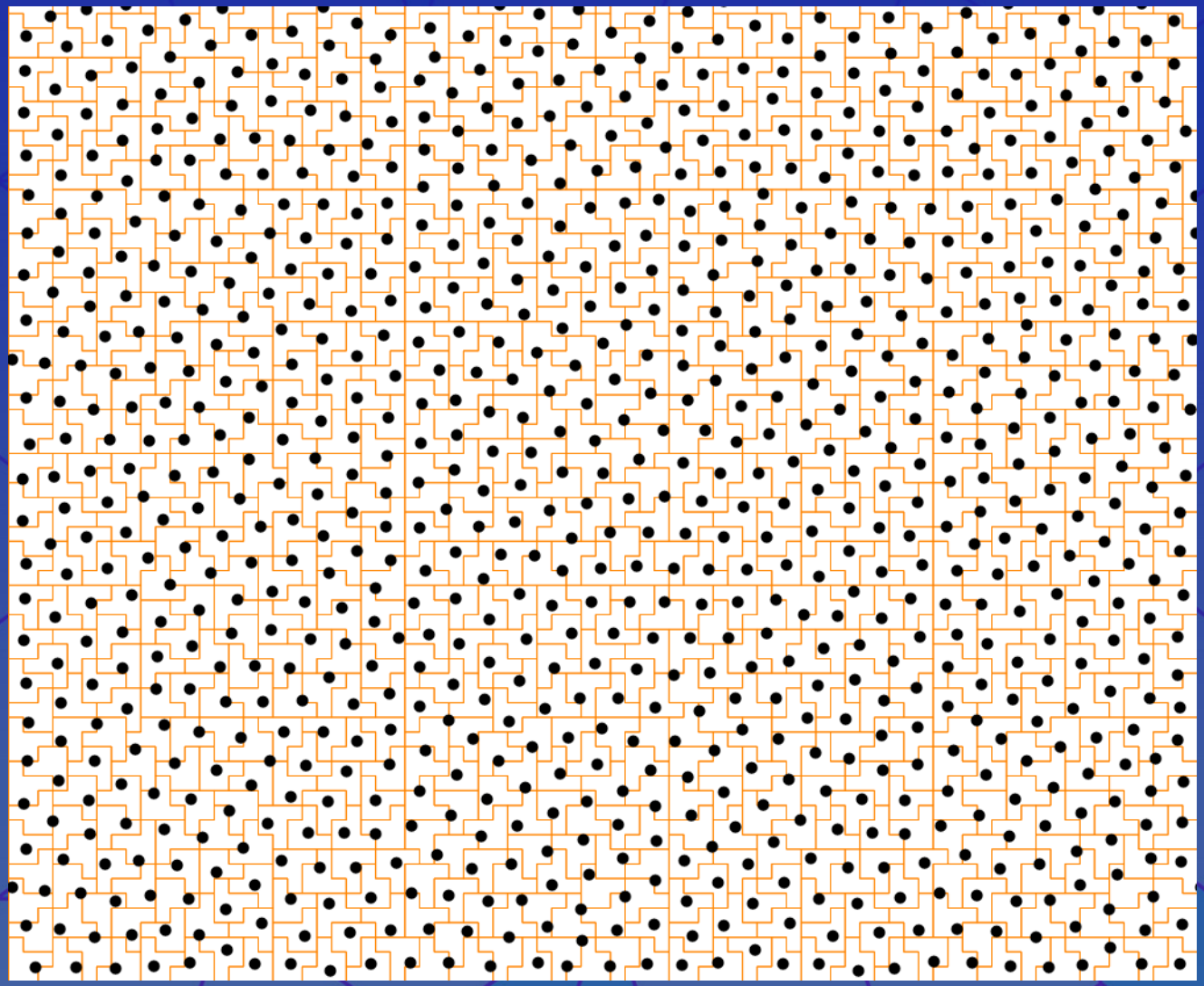
Run-time pseudo-code

```
ADAPTIVESAMP(p of type polyomino)
 1          ▷ Structure polyomino contains the fields:
 2          ▷    structuralIndex
 3          ▷    LOS: level of subdivision
 4          ▷    refPoint
 5          ▷    v1, v2: polyomino's basis vectors
 6          ▷    code: used for computing threshold
 7  local_LOS ← GETMAXLOSWITHINPOLYOMINO(p)
 8  if p.LOS ≥ local_LOS
 9     then          ▷ Terminal: no need for more subdivisions
10          local_importance ← GETLOCALIMPORTANCE(p)
11          threshold ← INTEGERINBASE𝒜(p.code)
12          if local_importance ≥ threshold
13             then          ▷ Output selected sample
14                  i_imp ← GETIMPORTANCEINDEX(local_importance)
15                  {k1, k2} ← lut[p.structuralIndex, i_imp]
16                  position ← p.refPoint + k1 * p.v1 + k2 * p.v2
17                  OUTPUTSAMPLE(position)
18          return
19     else          ▷ Need more subdivisions
20          {p1, ..., p𝒜} ← SUBDIVIDEUSINGPRODUCTIONRULES(p)
21          return {ADAPTIVESAMP(p1), ..., ADAPTIVESAMP(p𝒜)}
```

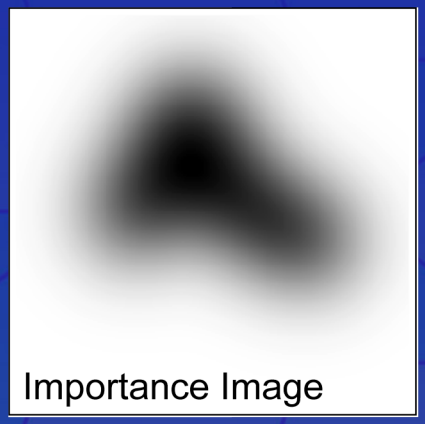
Run-time pseudo-code

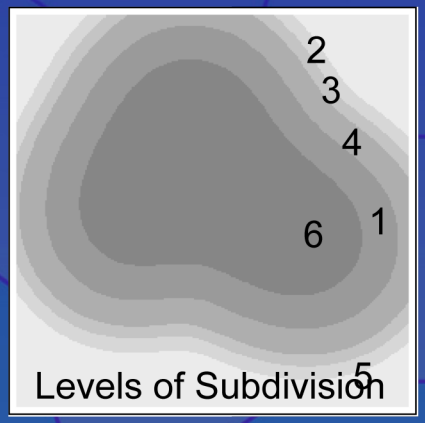
```
ADAPTIVESAMP(p of type polyomino)
 1          ▷ Structure polyomino contains the fields:
 2          ▷    structuralIndex
 3          ▷    LOS: level of subdivision
 4          ▷    refPoint
 5          ▷    v1, v2: polyomino's basis vectors
 6          ▷    code: used for computing threshold
 7  local_LOS ← GETMAXLOSWITHINPOLYOMINO(p)
 8  if p.LOS ≥ local_LOS
 9     then          ▷ Terminal: no need for more subdivisions
10          local_importance ← GETLOCALIMPORTANCE(p)
11          threshold ← INTEGERINBASE𝒜(p.code)
12          if local_importance ≥ threshold
13             then          ▷ Output selected sample
14                  i_imp ← GETIMPORTANCEINDEX(local_importance)
15                  {k1, k2} ← lut[p.structuralIndex, i_imp]
16                  position ← p.refPoint + k1 * p.v1 + k2 * p.v2
17                  OUTPUTSAMPLE(position)
18          return
19     else          ▷ Need more subdivisions
20          {p1, ..., p𝒜} ← SUBDIVIDEUSINGPRODUCTIONRULES(p)
21          return {ADAPTIVESAMP(p1), ..., ADAPTIVESAMP(p𝒜)}
```

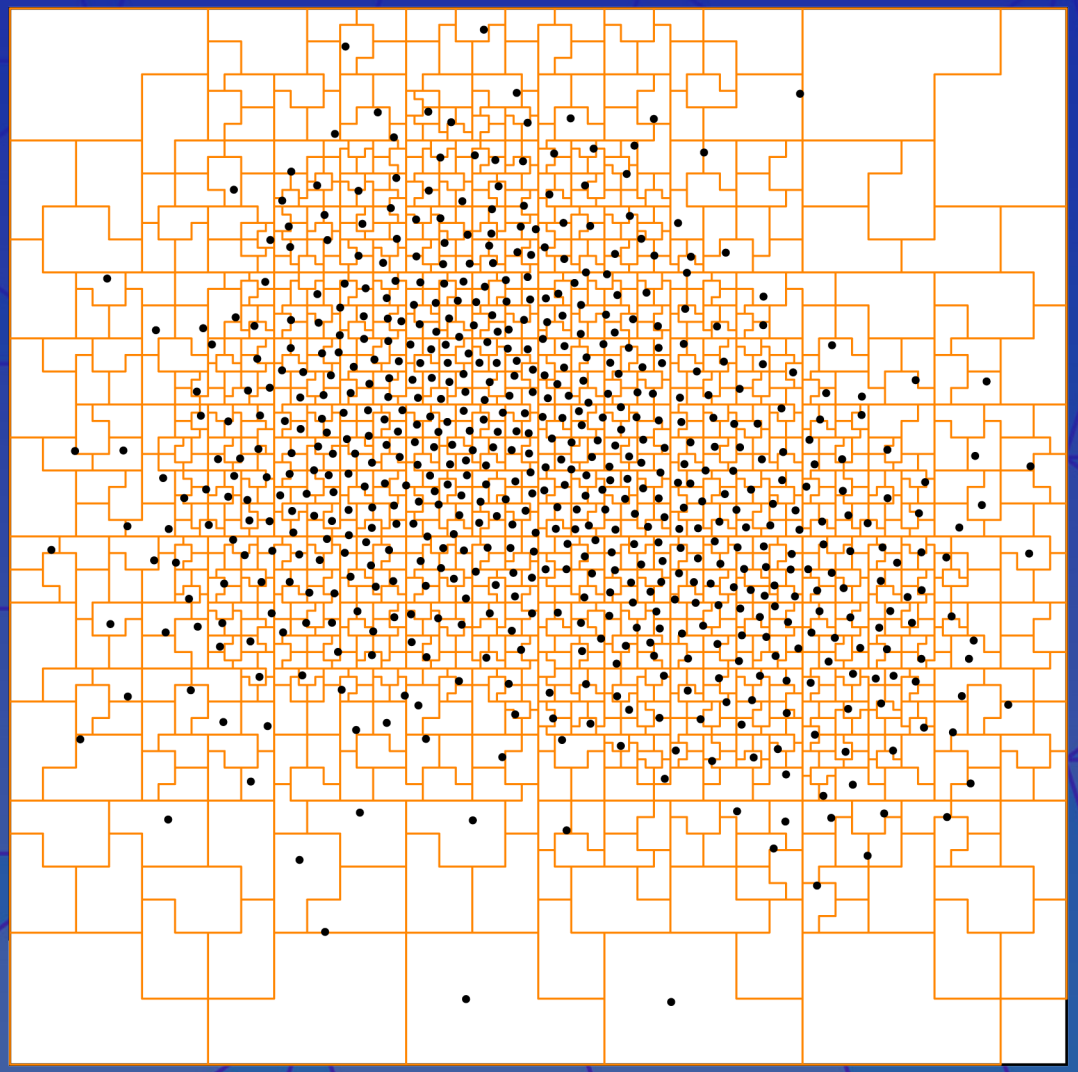
Results – constant density



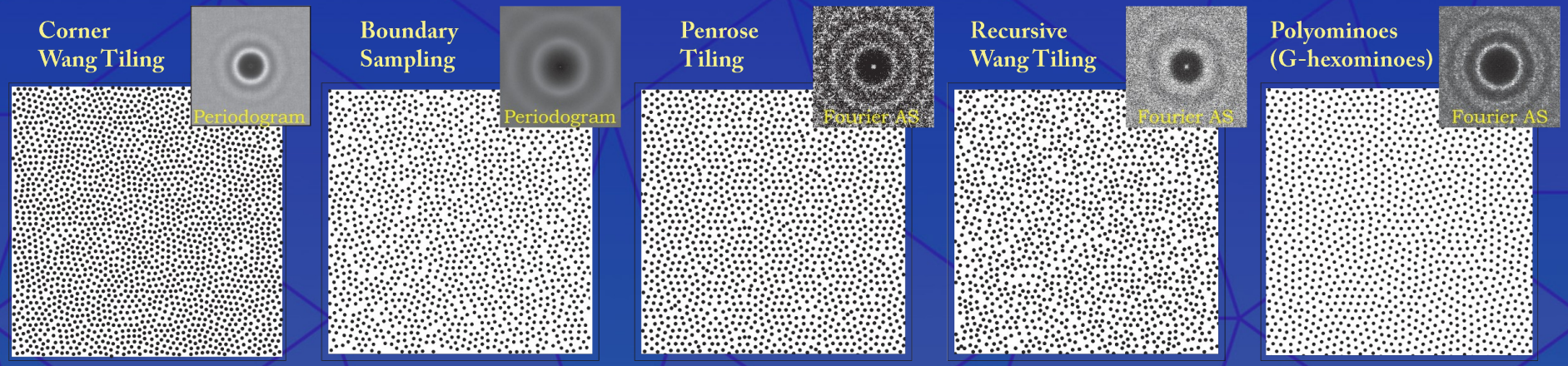
Results – variable density







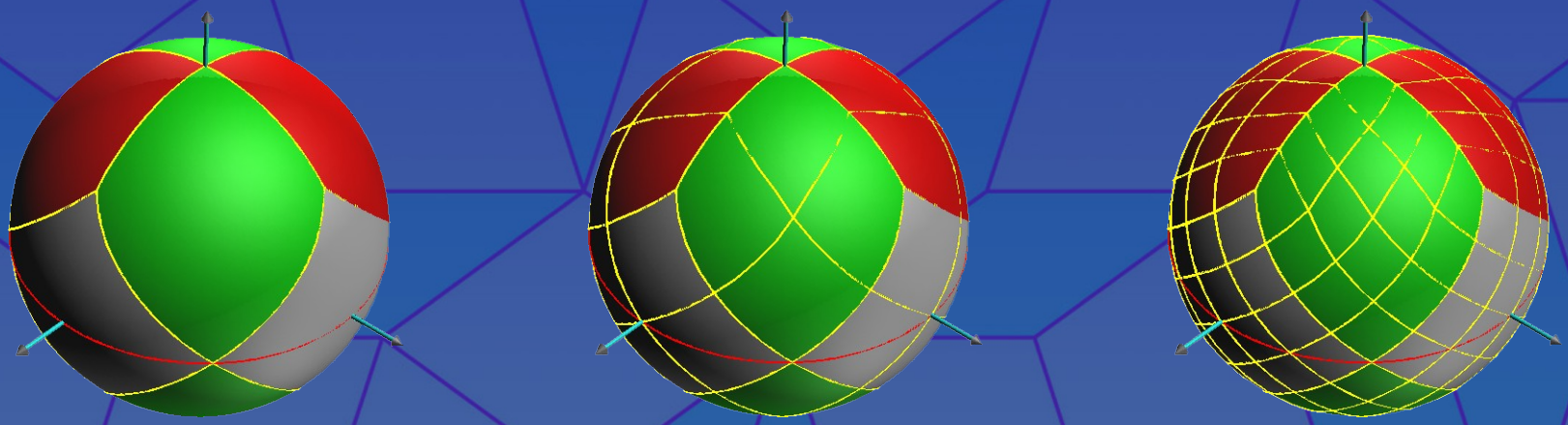
Results – comparison with other methods



Sampling on a sphere

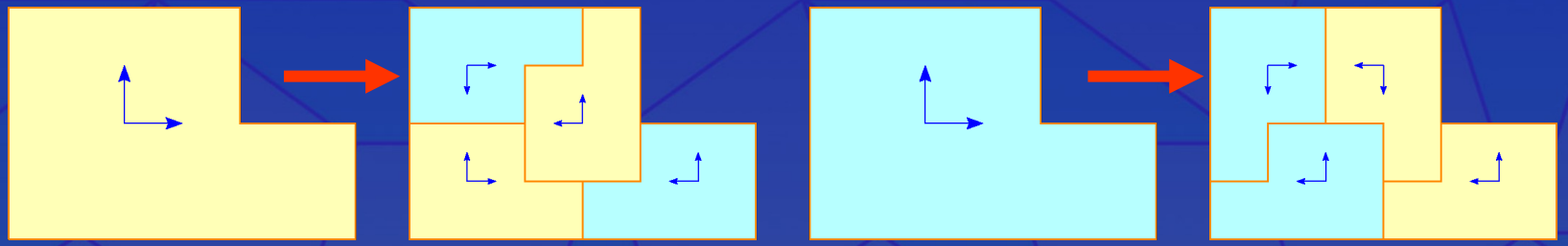
HEALPix spherical mapping [Gorski et al. 2005]

- 12 Equiareal high-level quads
- Jacobian-preserving mapping
  - Subdivision into equiareal quads
- Low distortion

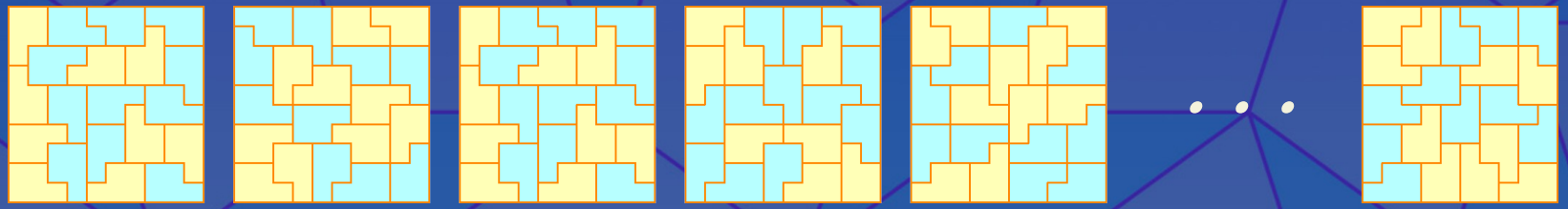


Sampling on a sphere

Pentominoes with two subdivision rules

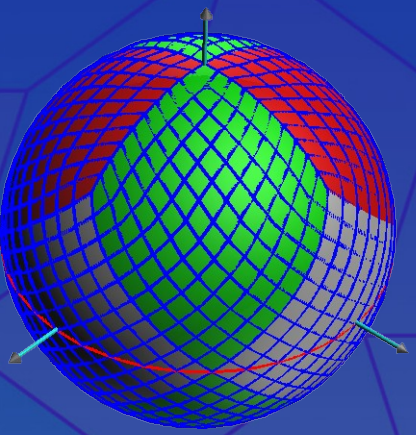


Square patches of 20 pentominoes on 10x10 squares

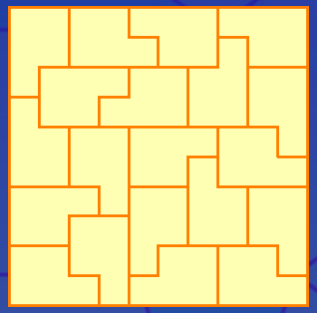


Sampling on a sphere

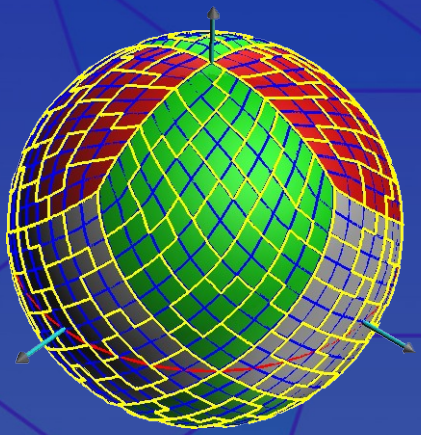
Pentominoes on 12 HEALPix quads

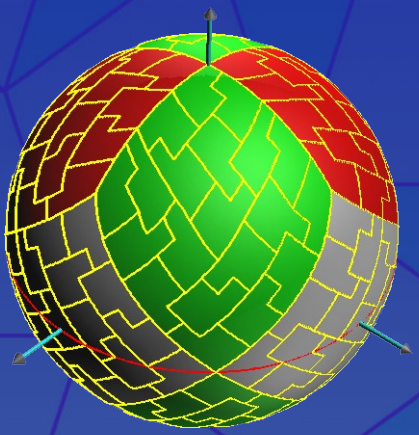


10x10 grid
on each quad



20 pentominoes
on a 10x10 square

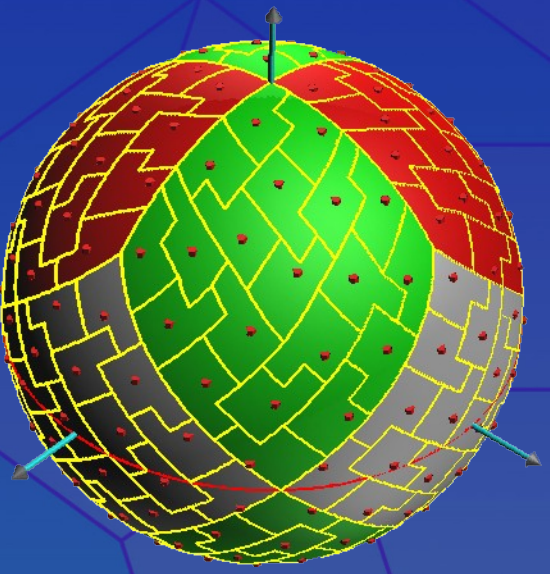




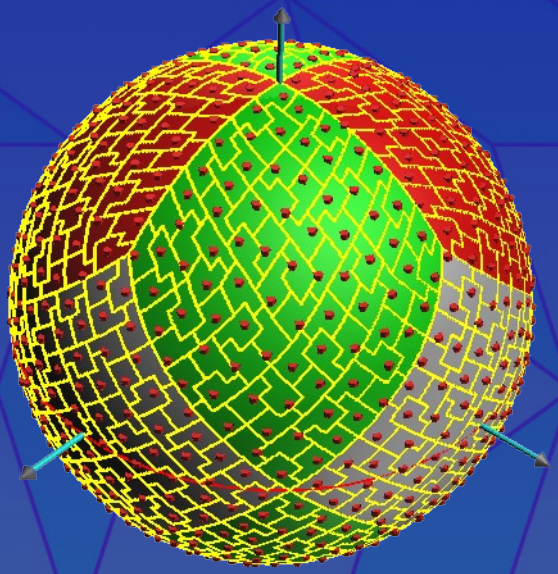
20 pentominoes
on each quad

Sampling on a sphere

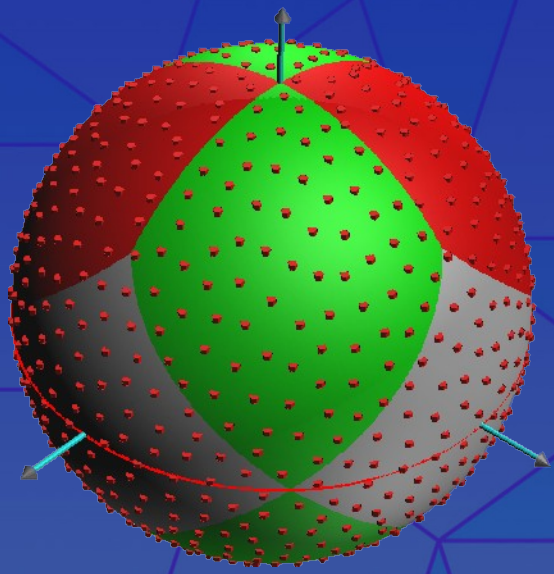
Subdivision and optimization



Level 1



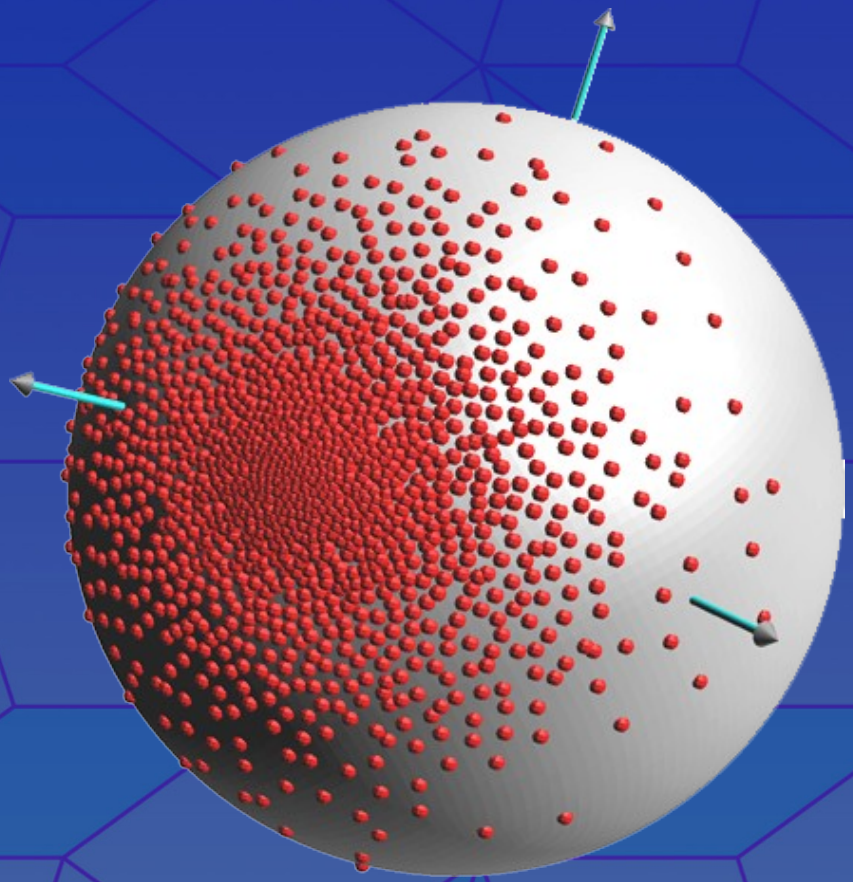
Level 2
(after one subdivision)



Level 2
(dots only)

Sampling on a sphere

An arbitrary importance (spot) function



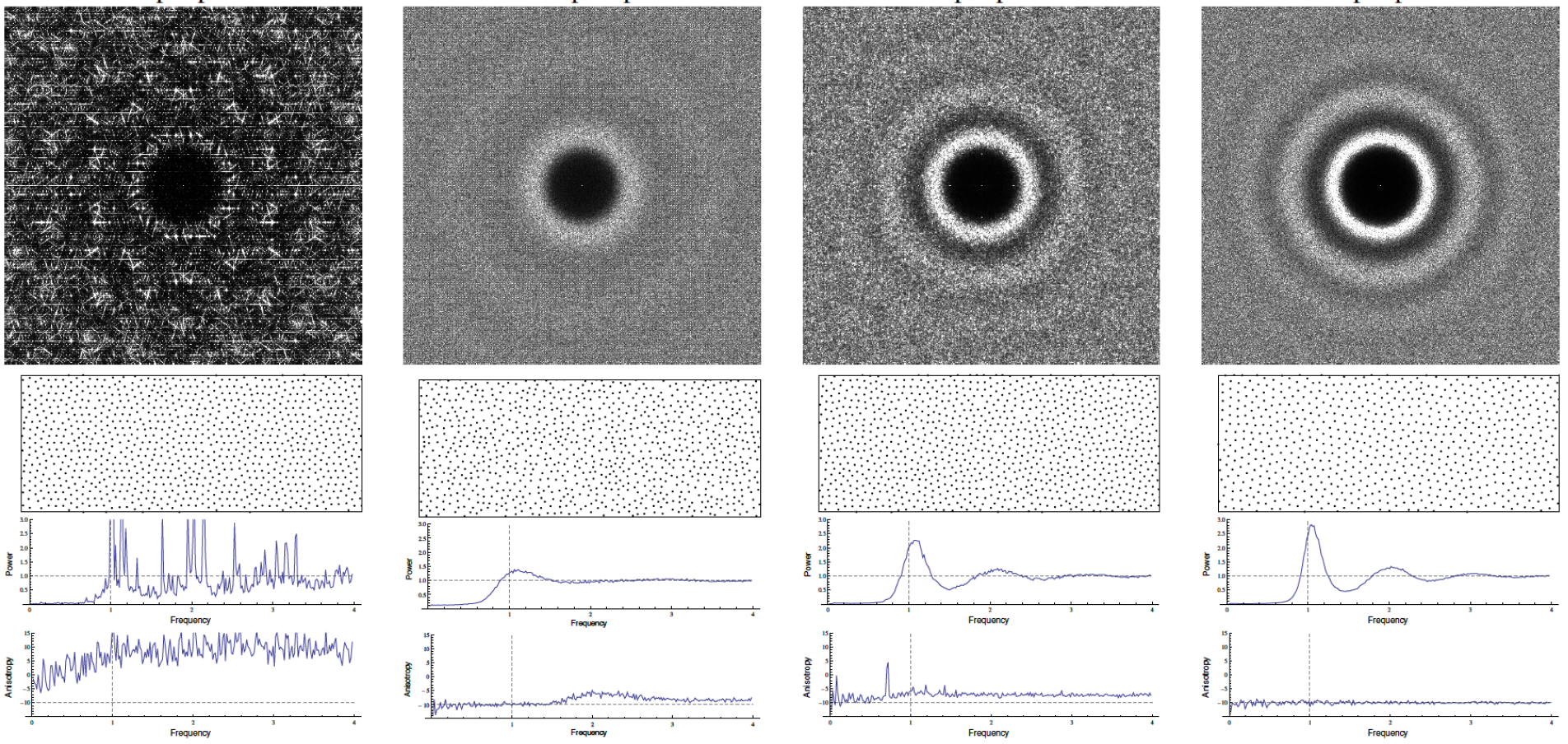
Further Improvement of the Spectrum (WIP)

[Ostromoukhov et al. 2004]
1 sample per tile

[Kopf et al. 2006]
1024 samples per tile

[Ostromoukhov 2007]
1 sample per tile

Ours
1 sample per tile



Further Improvement of the Spectrum

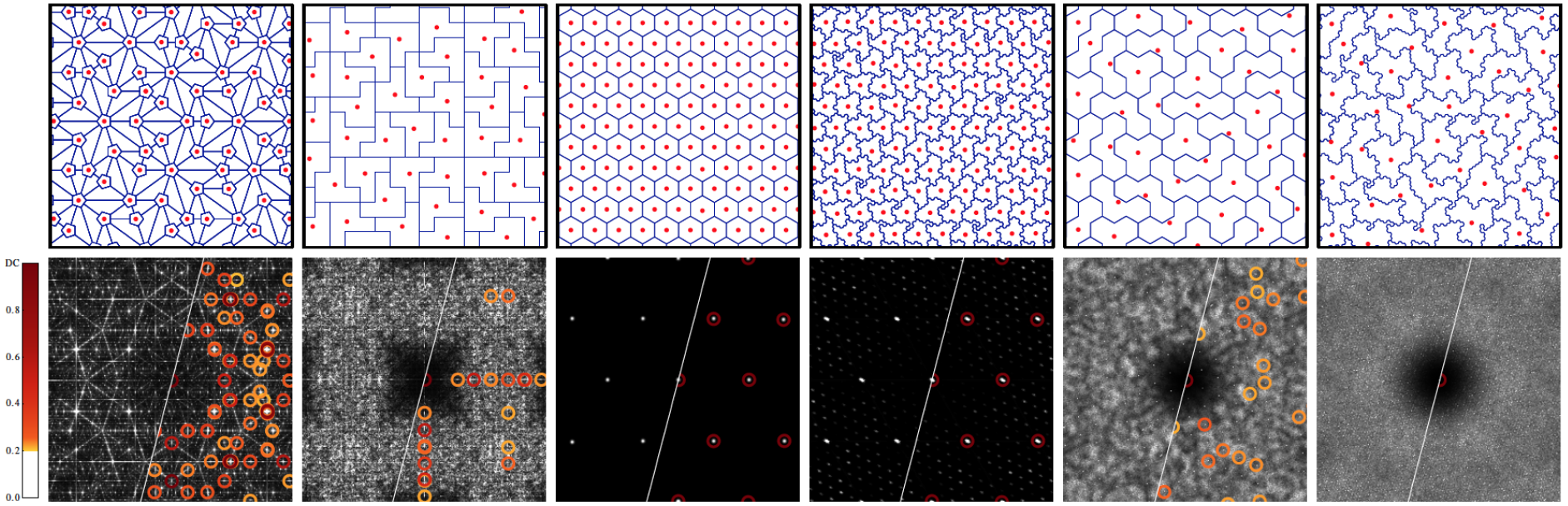


Figure 1: *Comparison of tiling used in tile-based sampling methods. From left to right: Penrose tiles, polyominoes, regular hexagonal tiles, irregular hextiles* $\tilde{h}$ *(§ 2.2), regular trihexes* t *(§ 3.1), and our choice of tile (irregular trihexes on* $\tilde{t}$*, § 3). Top row: tiling with their tile centroids, bottom row: Fourier spectra of centroid distributions. The right part of each spectrum contains circled marks to point out peaks of size* 0.2 *and above (for a DC value of* 1*).*

Building Blocks

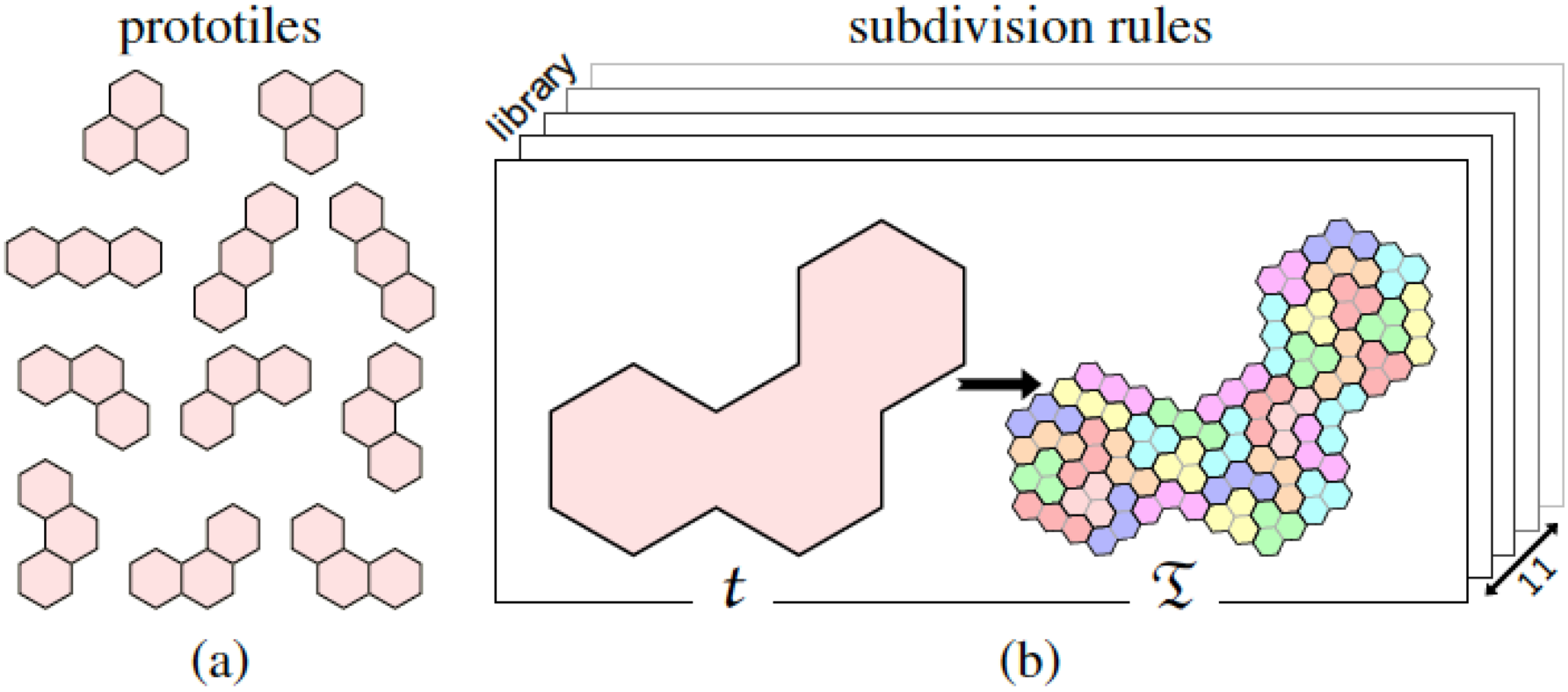


Figure 4: (a) *11 configurations of trihexes on* $\mathbb{H}$. (b) *The library of subdivision rules* $\{t \to \mathfrak{T}\}$, *where the quasi-trihex associated with* t *after subdivision is partitioned into a set of* λ *trihexes.*

Building Blocks

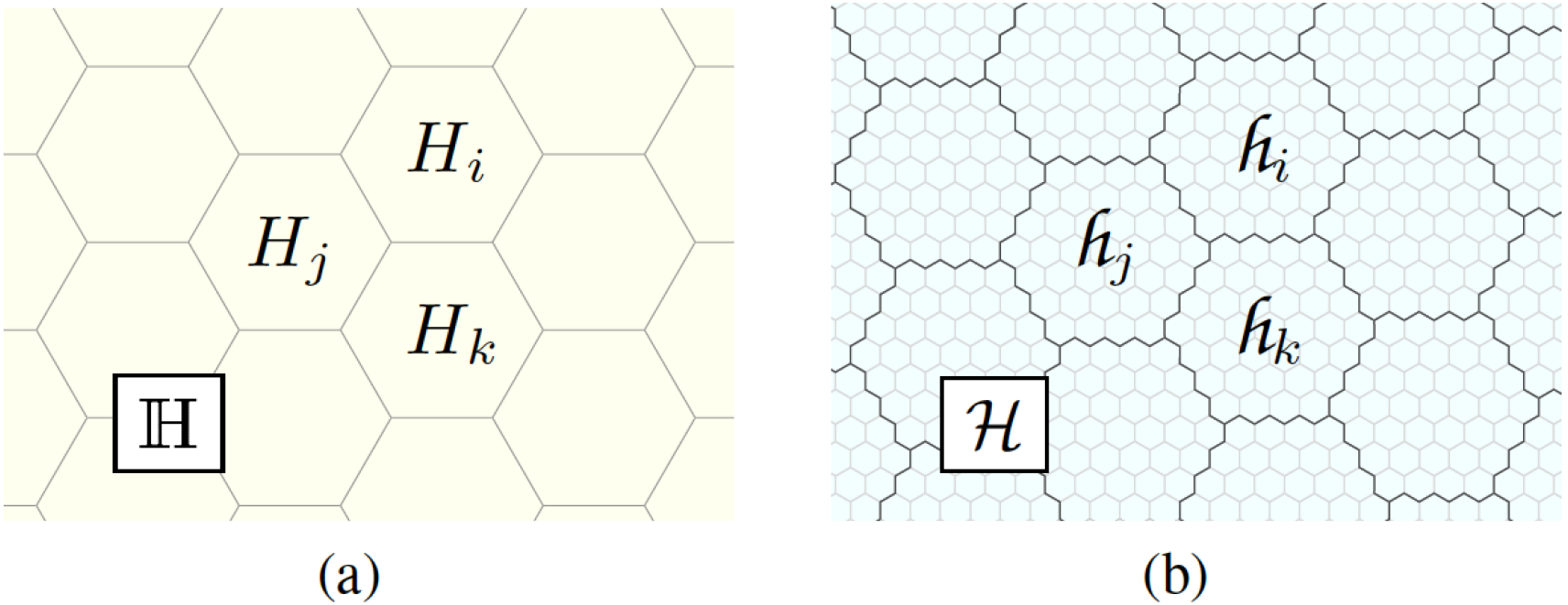


Figure 2: *Quasi-hexagonal tiles. (a) The unbounded, regular hexagonal lattice* $\mathbb{H}$. *(b) The unbounded, quasi-hexagonal lattice* $\mathcal{H}$. *Mention* λ *and* $\mathbb{h}$.

Building Blocks

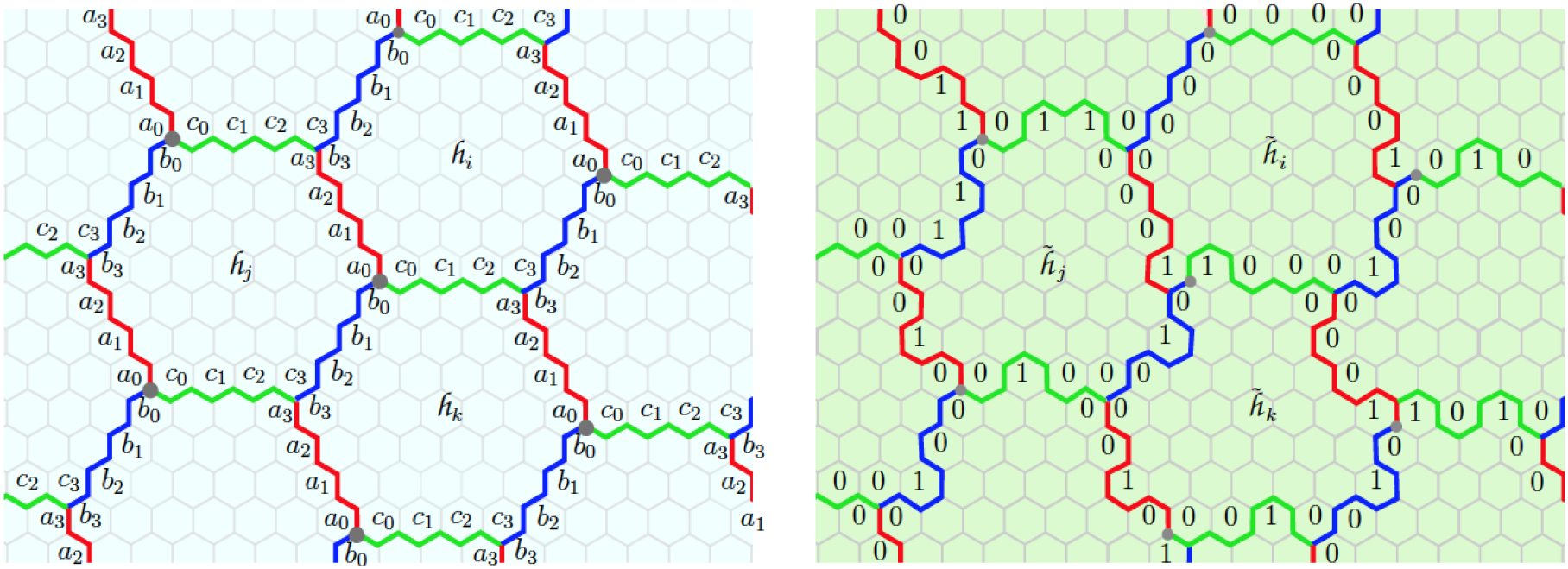


Figure 3: *Effect of edge modifiers. From a set of hextiles $\mathcal{h}_i$ and random valid edge modifiers (a), a set of irregular tiles $\tilde{\mathcal{h}}_i$ in $\tilde{\mathcal{H}}$ is obtained (b) which maintains the topology of the initial tiling as well as each tile's area.*

Building Blocks

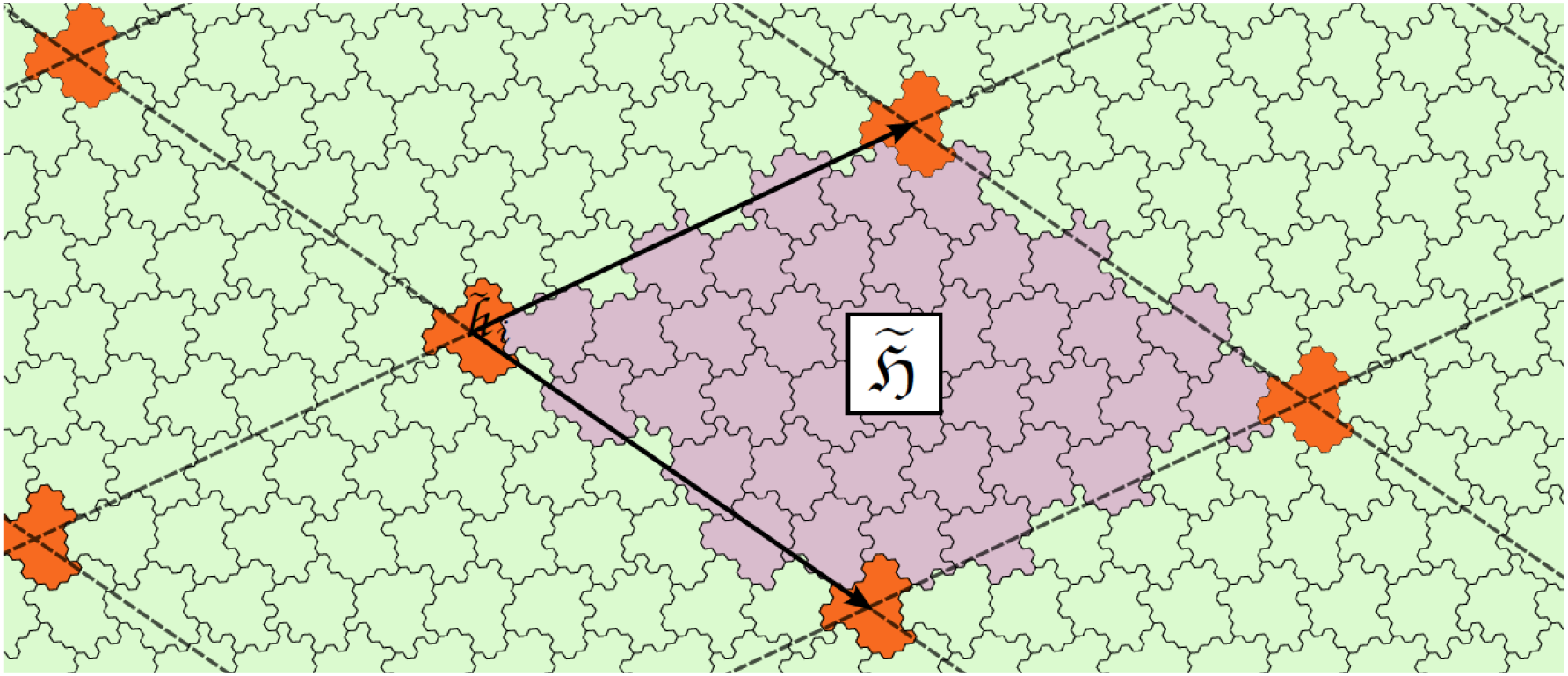


Figure 5: *Periodic tiling construction based on two integer basis vectors, creating a small set $\widetilde{\mathfrak{H}}$ of λ irregular hextiles $\{\tilde{h}_i\}$ (highlighted).*

Building Blocks

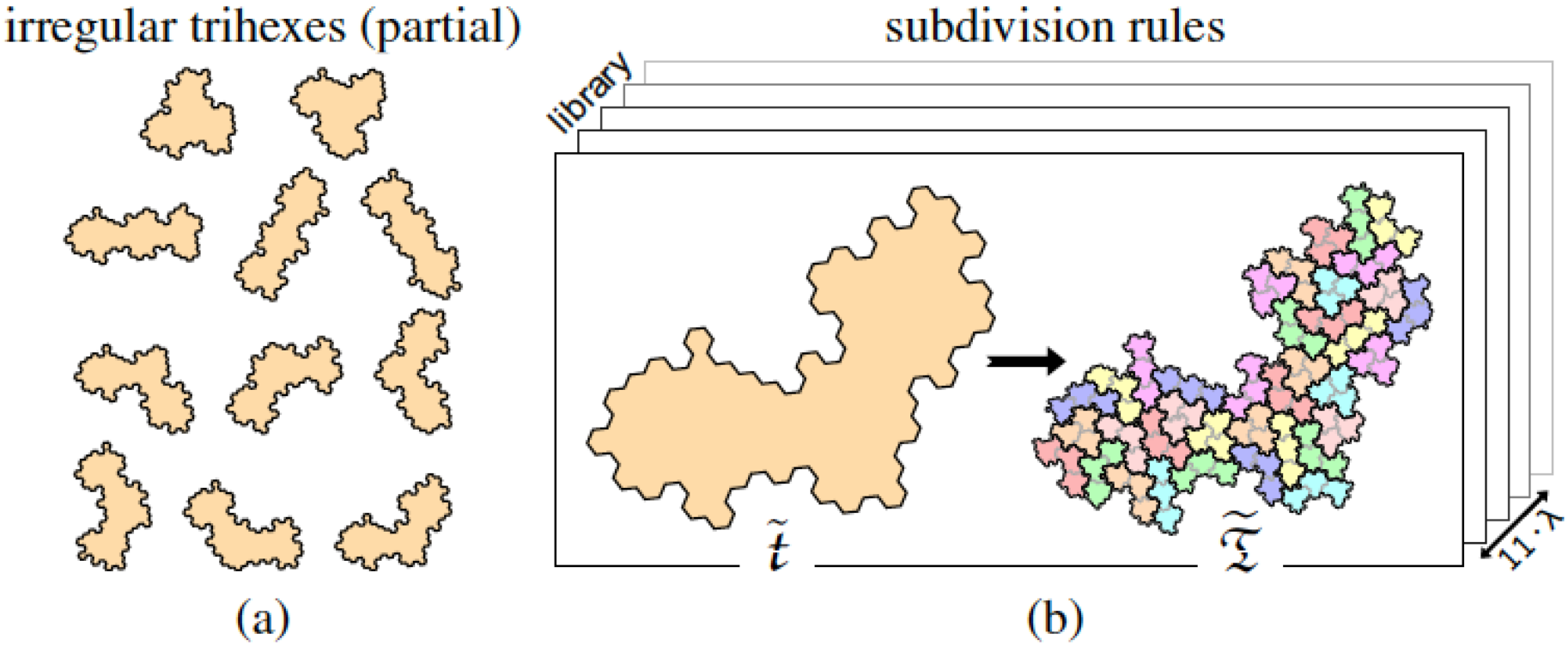
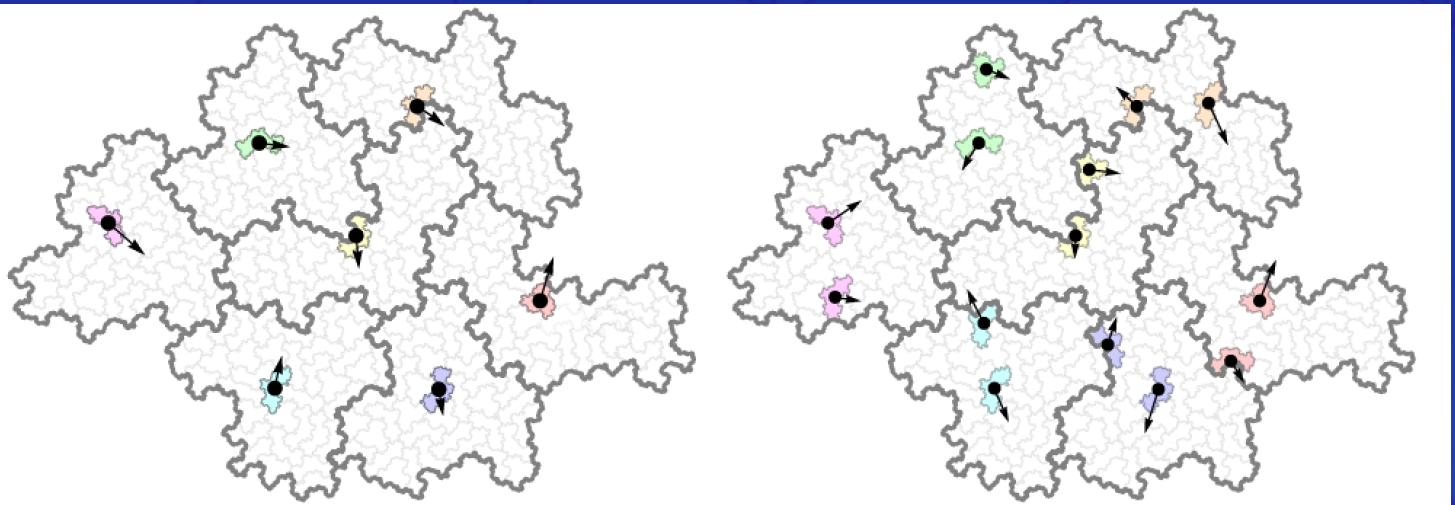
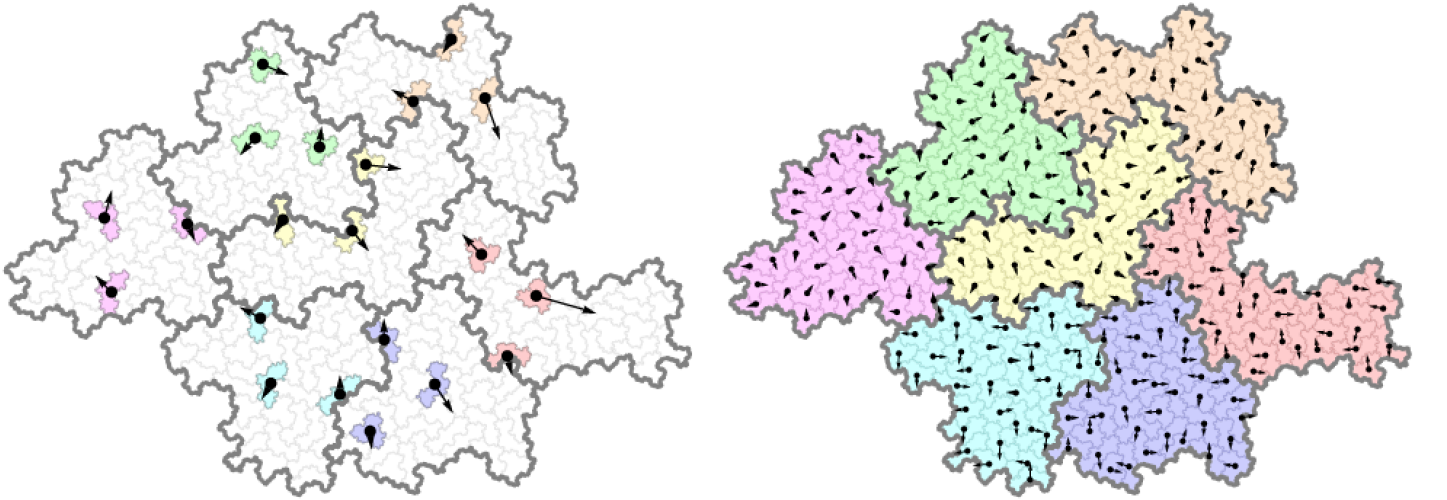


Figure 7: (a) *Examples of irregular trihexes in* $\mathcal{T}$ (11 *out of* $11.\lambda$ *are shown).* (b) *The library of subdivision rules* $\{\tilde{t} \rightarrow \tilde{\Sigma}\}$ *for all irregular trihexes.*

Ranking



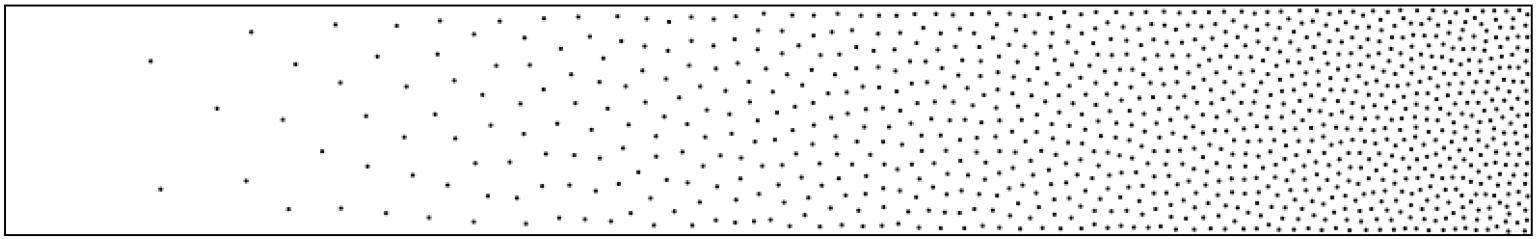
(a) R_1 (b) R_2



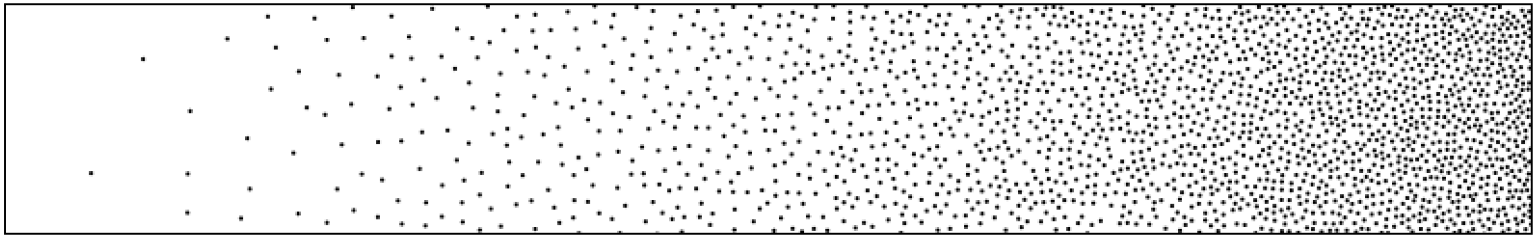
(c) R_3 (d) R_{37}

Results

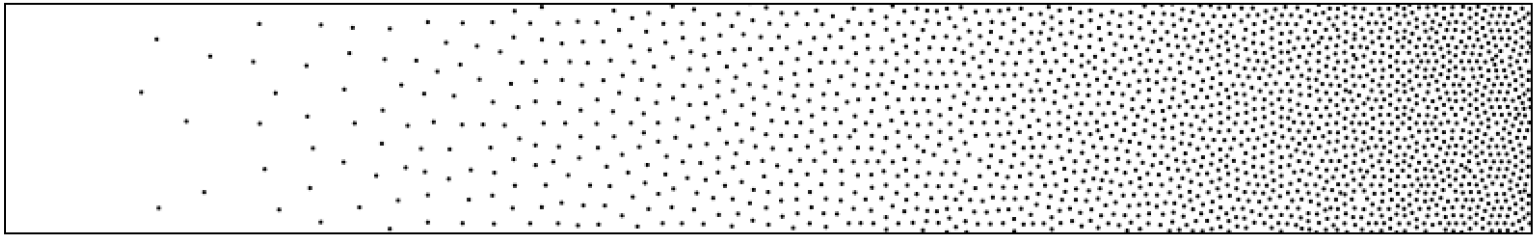
[de Goes et al. 2012]



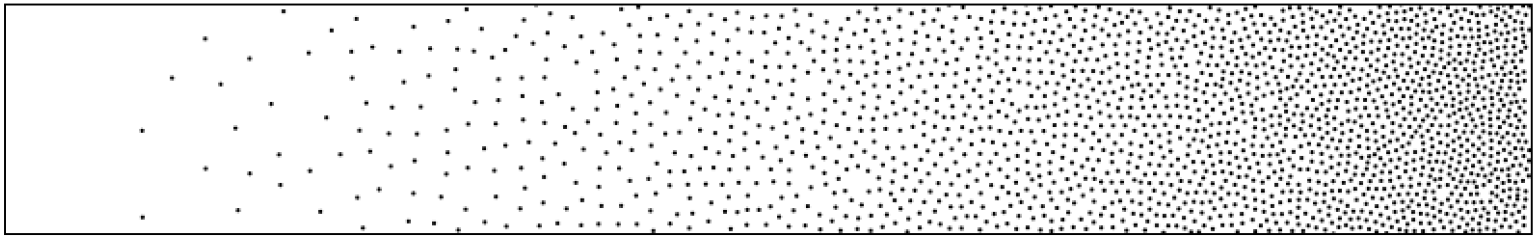
[Kopf et al. 2006]



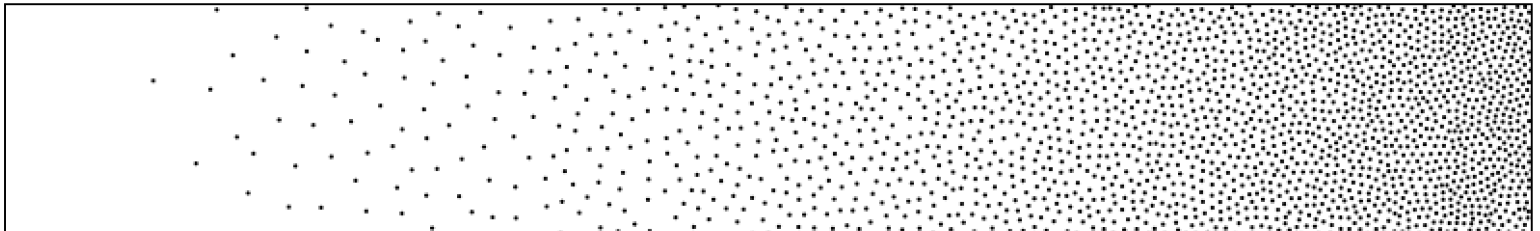
[Ostromoukhov et al. 2004]



[Ostromoukhov 2007]



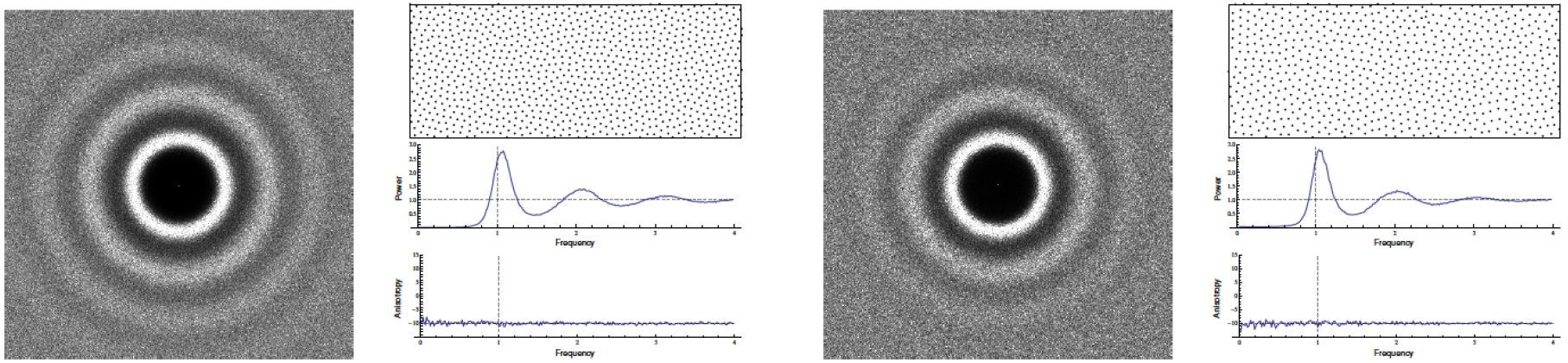
Ours



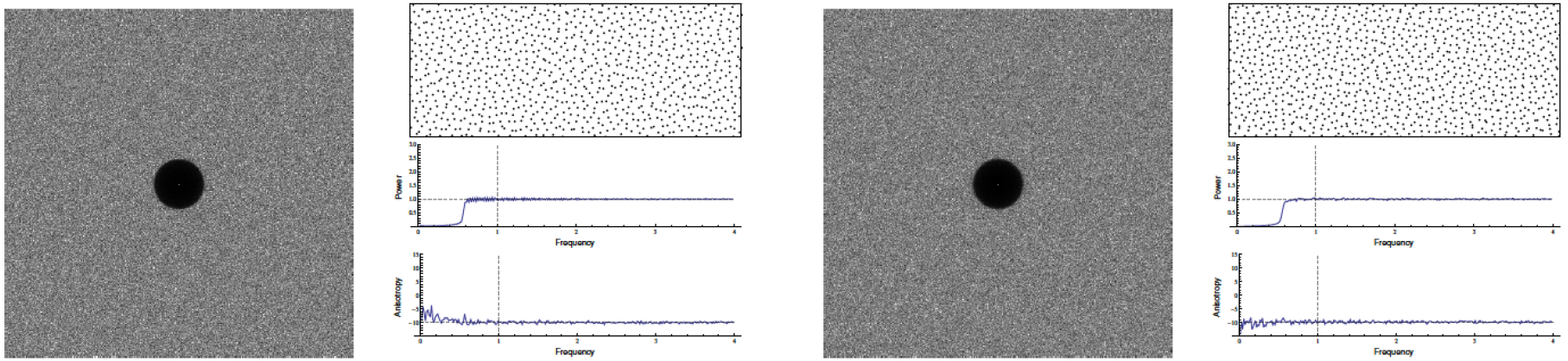
Original method

Results through our method

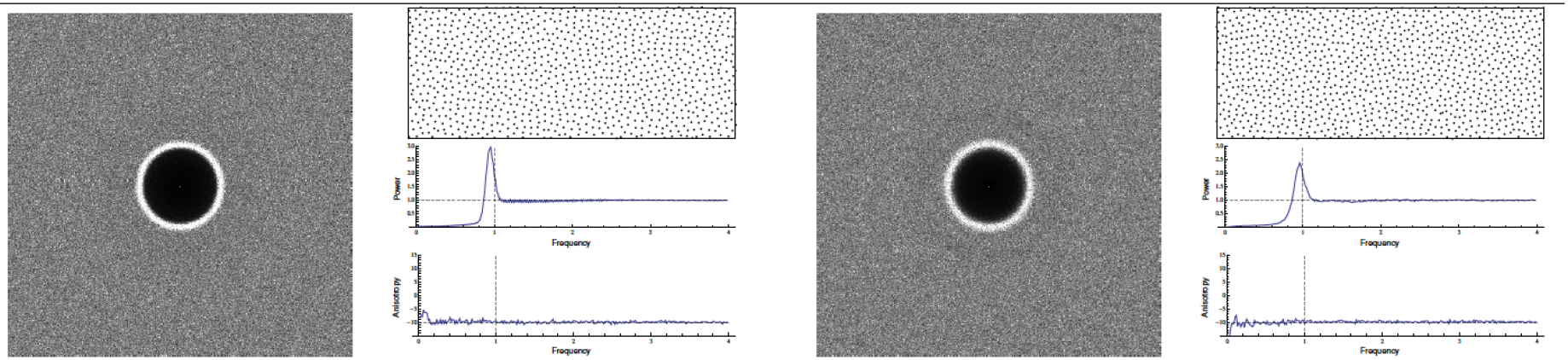
[de Goes et al. 2012]



Step



Ulichney



Results: Qualitative Analysis

Methods	Qualitative Analysis			
	Speed	Memory footprint	Distribution Quality	Spectral Control
[Schlömer et al. 2011]	poor	good	good	✗
[de Goes et al. 2012]	poor	good	excellent	✗
General Noise (blue noise target)	very poor	good	good	✓
[Ostromoukhov et al. 2004]	excellent	poor	satisfactory	✗
[Kopf et al. 2006]	excellent	poor	good	✗
[Ostromoukhov 2007]	excellent	poor	good	✗
Ours (blue noise target)	excellent	poor	excellent	✓
Ours (general noise target)	excellent	poor	good	✓

Future Challenges

- Reduce Memory Footprint
- Extend to Higher Dimensions

- Be Able to Build an Irregular Tiling System Approaching Required Spectral Properties (Penrose-like Tiling)
  - Compact
  - High (Or Without) Rotational Symmetry
  - With Appropriate Number System

Thank You!

Space-Filling Polyhedra (Plesiohedra)