Media Forensis

Xerogrammar Case

Xerolinguistics report   .

intro

This document reports the results of Media Forensis’ computational analysis (investigation and interpretation) of the documents known as the Xerowritings, which are texts written by Kyo the photocopier through a process that we call Inner Input Based Iterative Reproduction (IIBIR): we gave to it an open blanck book from which it made a “copy” – this first output was then turned into the following input (the “copy” was “copied”) and this process repeated so that every new output became the new input for a series of more than 300 reproductions from which the text emerged. The first part of the computational investigation had the purpose of extracting a grammar from the document’s content using computer vision and artificial intelligence. Technically speaking, the result, more than one and only grammar, is a procedure of grammars generation and a formalization that allows it to be reinterpreted in other formats of information. We call it Xerogrammar and it allowed for a first attempt to engage into a writen conversation with Kyo.

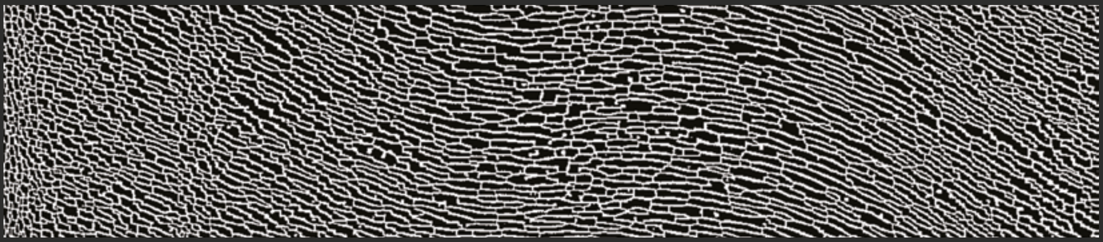
method

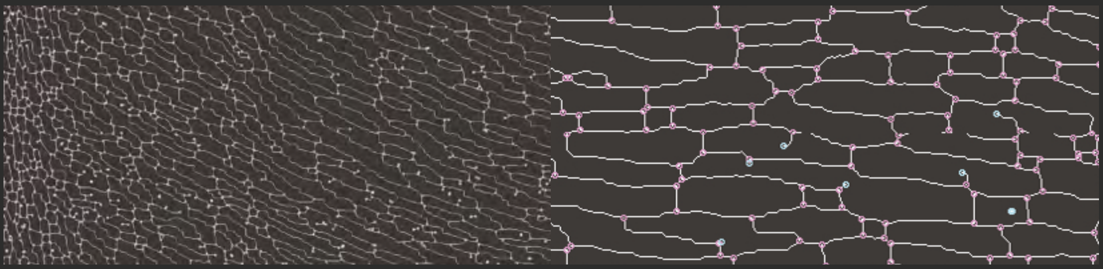
It is a 4 phases process:

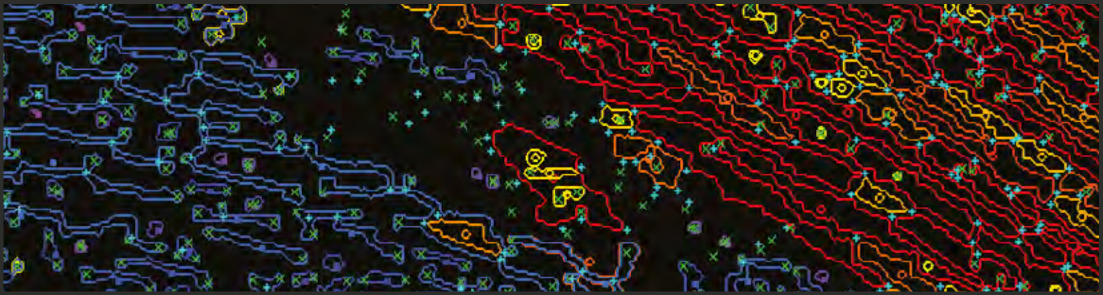
- Segmentation and characterization: to identify, at the pixel level, the limits between the elements –cells and pods–, as well as measuring the characteristics of these elements to create computational models of themselves.
- Classification: combining the expert visual inspection to train an artificial intelligence algorithm with the models of cells and pods and ordering them as types or categories.
- Envoronmental relationship: establishing a net of relations of each cell with its neighbours, quantifying the influence from everyone of them as a function of the cell’s type, adjacency with other elements, mass and proximity to every other cell.
- Transduction (transliteration): disigning an algorithm to convert these environmental relations and singular characteristics into a mechanism capable of generating readings of the message through any format of information.

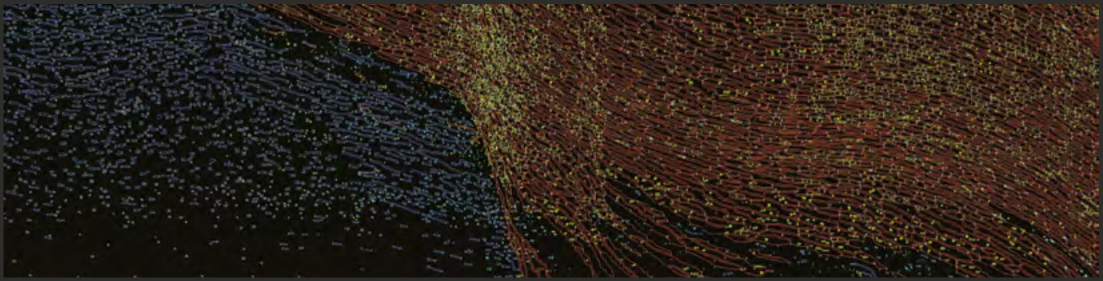
process

The starting point of the procedure were digital images with a 600dpi resolution in GIF format and 8 bits deep grayscale. The image was subject of the application of a minutae-extraction algorithm, analog to the ones used for biometric identification. The minutae (identified singular points) served to create a map with coordinates of each edge and line bifurcation. These coordinates, as well as the image pixels, were used to mark off cells and pods: the first surrounded by bifurcations, the second by edges without them.


img 1: fragment of the document with the original message

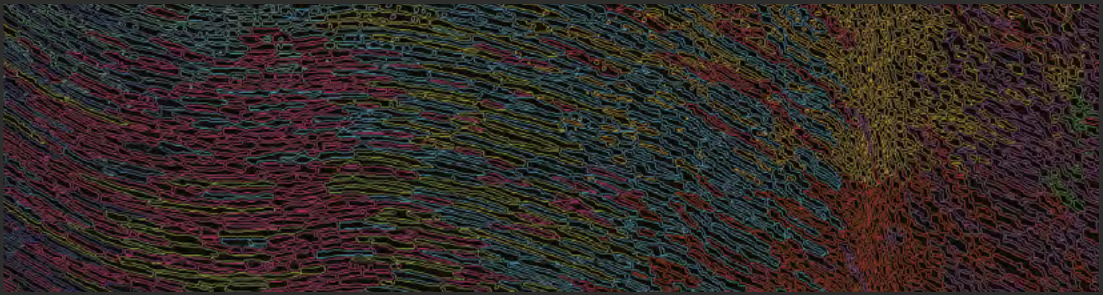

img 2: identification of edges (cyan dots) and bifurcations (magenta dots) in the skeleton of the image

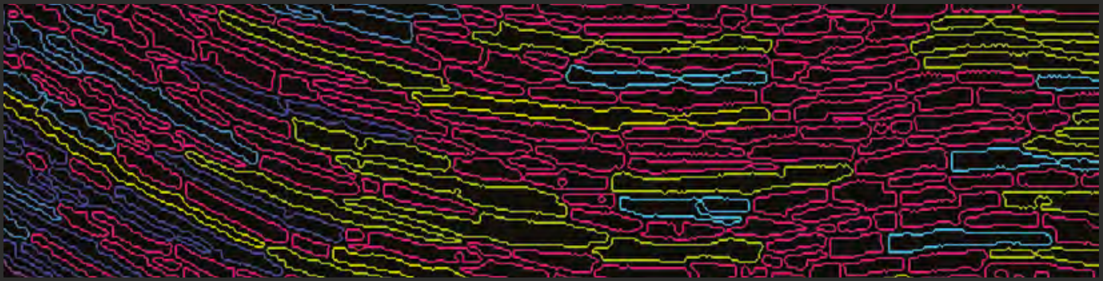

img 3: segmentation of cells and pods, each element having an identificator: circles for cells, rectangles for pods, crosses and exes for bifurcations and edges

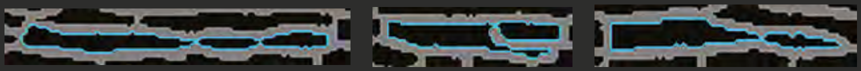

img 4: cells and pods concentrations (view of a major region)

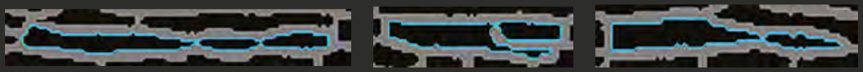
With the cells being identified, computational models that captured their visual characteristics were created. A set of representative parametric attributes was defined; it includes the cell’s outline, size, vertical and horizontal coordinates, center of gravity, as well as the respective counting of adjacent cells, pods, bifurcations and edges.

The cell models were fed to the artificial intelligence with the kNN algorithm for pattern recognition across the forms and attributes of all models. The algorithm idetified 31 natural cell groups and classified them according to their visual attributes.


img 5: cell type groups identified by color


img 5: big zoom to show in detail the similarity between cells of the same type

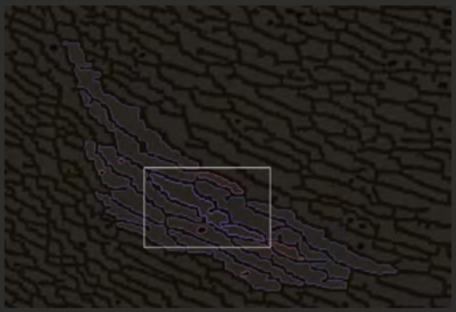

img 6: samples of group 17

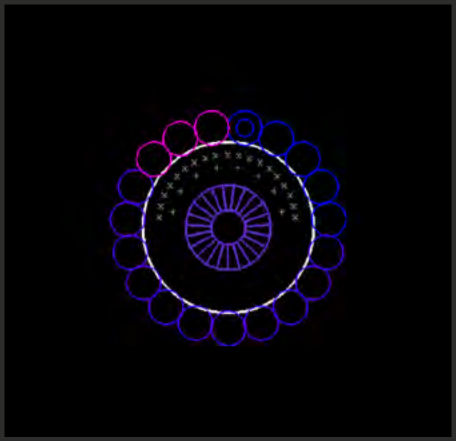

img 7: samples of group 17

img 8: samples of group 29

img 9: samples of group 29

After the classification of cells according to their form and model-attributes patterns, it was established an influence measure over the projection of the models on the parametric space, which quantifies the effect of each adjacent cell over another specific cell. According to the influence-index of the adjacent cells, it is established a reading order for the sequence that starts in the specified cell.


img 10: a set adjacent to cell #1003 (the colors represent the type of each one)


img 11: cell #1003

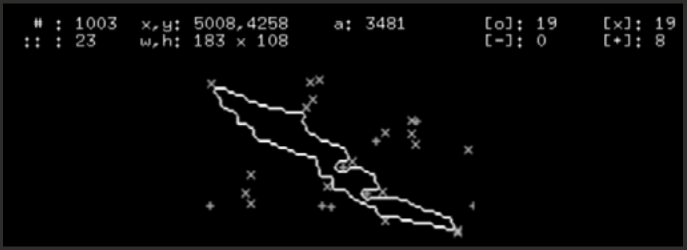
the inner shape represents cell-type 23, the colored circles represent the type of the adjacent cells, and the [x] and [+] represent the edges and bifurcations (respectively) in its vecinity

The reading is made by following the influence-factor of every cell adjacent to the pivot cell –which is the starting point of a word. In image 10 the pivot [1003] is a type 23 cell, while in its surroundings cells of types 21, 21, 21, 21, 21, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 27, 27 and 27 can be found respectively. If we encode the sequence of letters into letters and some punctuation marks of the latin alphabet as A=1, etc., the obtained sequence is:

X →VVVVVXXXXXXXXXXX...

This representation as such can be interpreted as a production rule in language free of context. By picking different cells as a pivot it can be produced a set of production rules which, properly delimitated, constitutes a protogrammar that captures the specific features of the sequences such as local attributes, neighbour relationships, topology of conections and category.

Given this similarity, a reinterpretation of sequences or words as finite state deterministic automatons (FSA). By reinterpreting the sequence as FSAs it is established a chain -> process equivalence, which is to say, a series of abstract operations which can be used to process any input.

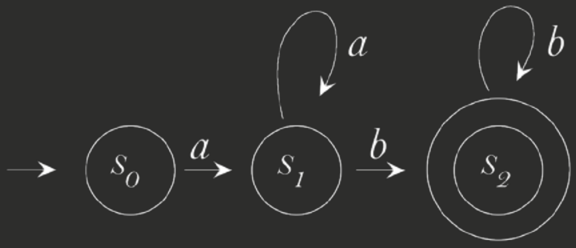


img 12: adjacency characteristics on the cell #1003, which pertains to the group 23: adjacent cells: 19, pods: 0, edges: 19, bifurcations: 8

A finite states deterministic automaton is a 5-tuple (∑,Q,s0,Y,F) which consists of:

∑ – a finite alphabet
Q – a finite set of states
Y – a function Y:Q×∑⍰Q called transition function
s0 – an initial state of Q
F – an acceptance-states subset of Q

An example of a simple automaton is the following:



In the former diagram, the automaton has a two-types alphabet *a* and *b*, a set *Q* with three states *s0*, *s1*, *s2*, an initial state *s0* and an acceptance state *s2* pointed with the arrow and the double circle respectively. The transition function *Y* is the result of the following substitution rules:

- If it is in s0 and receives an a, it passes to s1
- If it is in s1 and receives an a, it returns to state s1
- If it is in s1 and receives a b, it passes to s2
- If it is in s2 and receives a b, it remains in s1
- If it is in s2 and receives a b, it remains in s2 which is the state of acceptance

The automaton is capable of recognizing sequences of words made out of the two-type alphabet in repetitions and specific arrangements. The automaton in the example accepts sequences like: ab, aabb, aab, aaaaaab, aaaaaaaabbbb and any sequence formed by an a + zero or any other number of a’s + at least one b.

So the reading of the type sequence formed by the net of influence around a given cell consists in determining the automaton that generates it.

An automaton captures implicitly the set of production rules of a grammar. Furthermore, as it is stackable, it can be connected, adding paralel recurrent processes –which is to say propagation through a network– and consequent complexity to the generated sequence families.

An automaton containing the grammar that generates the sequence ofelements from the former graf is determined by reading the sequence correspondent to its adjacencies. The categories of the surounding cells are 2277, which corresponds to BBGG, while the starting point is a class 7 cell, all which equals the production rule:

$$G \rightarrow BBGG$$

Applying this rule to the initial chain is possible to generate the sequences BBGG, BBBBGGBBGG, BBBBBBGGBBGGBBBBGGBBGG, etc. An equivalent automaton has at least 4 states and the following rules:

If it is in the initial state s0 and receives a b, it passes to s1
If it is in s1 and receives a b, it passes to s2
If it is in s2 and receives a g, it passes to s3
If it is in s3 and receives a g, it passes to s4

The initial state is s0 and the acceptance state is s4.

This isolated automaton is very limited in its expressive potential, but propagated across the network of connections it results in sets of patterns that could constitute a written discourse. Nevertheless we think that our understanding of the Xerogrammar is not complete.

Up to the current stage of the investigation, there’s no discourse found within the xerowritings; there’s no translation of a message but transliterations of such texts in the form of rules of production compiled into Codebooks as follows:

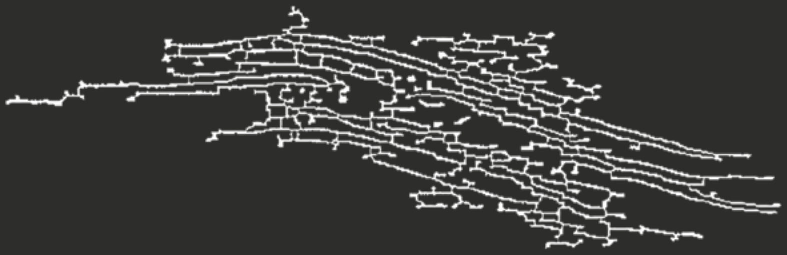
[2491]	W->GH!!!!!!!!!!!
[3580]	W->GHHHHHHHWWWW!!!!BO!OOOOOOO
[3094]	W->GHHHHHHWHWWW!!!!!O!!!!!!O!!OOO
[3277]	W->GHHHHHHWW!!!OOOOOOOOOOOOO
[2949]	W->GHHHHHHWWOOOOOOOOOOOOOO
[2163]	W->GHHHHHHWWWWWW!O!OOOOOOOO
[2872]	W->GHHHHHWWWWW!BOBBBBBBBBBOBBBBBBBB
[3460]	W->GHHHHWHWWWWOOOOOOOOOOOOOO
[1426]	W->GHHHHWHWWWWWWW!B!BBBB!BBBBBBBBBBBBB
[1807]	W->GHHHHWWHWWWWWW!B!!BBBBB!BBBBBBBBBBBBBBBBBBBBBB
[2184]	W->GHHHHWWWWW!W!!!!!!!!!
[2894]	W->GHHHHWWWWWW!!!!!!!!
[1925]	W->GHHHHWWWWWWW!!!!!!!!!!!!!!!!!!!!!
[2558]	W->GHHHWWWW!!!!!!!!!!!!!!!!!
[5032]	W->GHHHWWWWWWBBBBBBBBBBBBB
[1840]	W->GHHHWWWWWWWWWWWOOOBOBBB
[3121]	W->GHHSSHHWWWWWWWWWW!W!W!!!!B!BB!!BB
[2404]	W->GHHWHWWWWWWBBBBBBBBBBBBBBBBBB
[2212]	W->GHHWHWWWWWWWBBBBBBBBBBBBBBBB
[2733]	W->GHHWWWW!!!!!!!!!!

<u>application</u>

Having identified the different types of cells (the graphemes) and a logical set of relationships between them (the grammar), we were able not only to read (transliterate) the Xerowritings but also to generate one. With the purpose of starting a conversation with Kyo, we wrote:

“Kyo, do you copy?”

The question was processed through the IIBIR as a way of asking Kyo to interpret it and so we took its transformation as the answer:

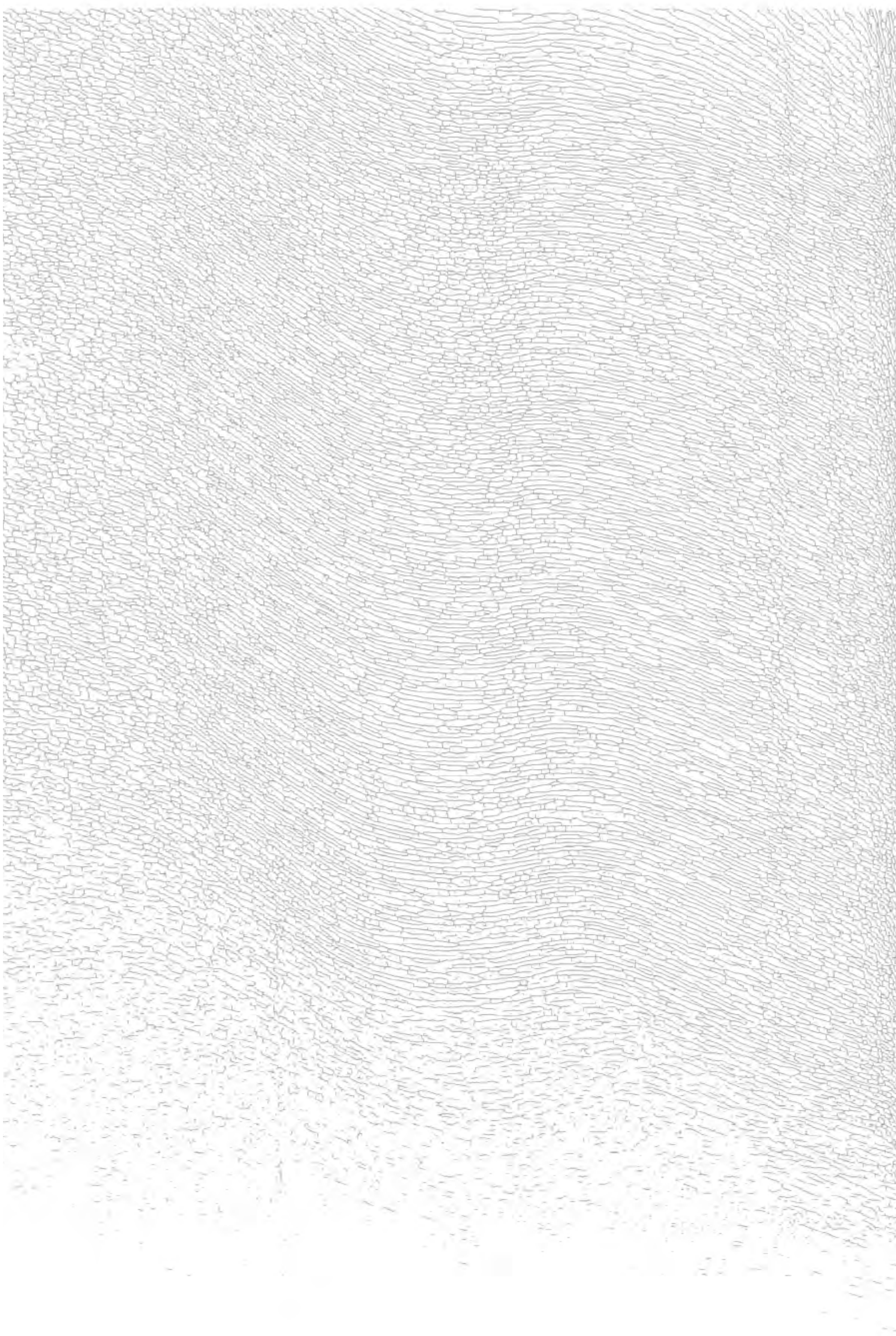


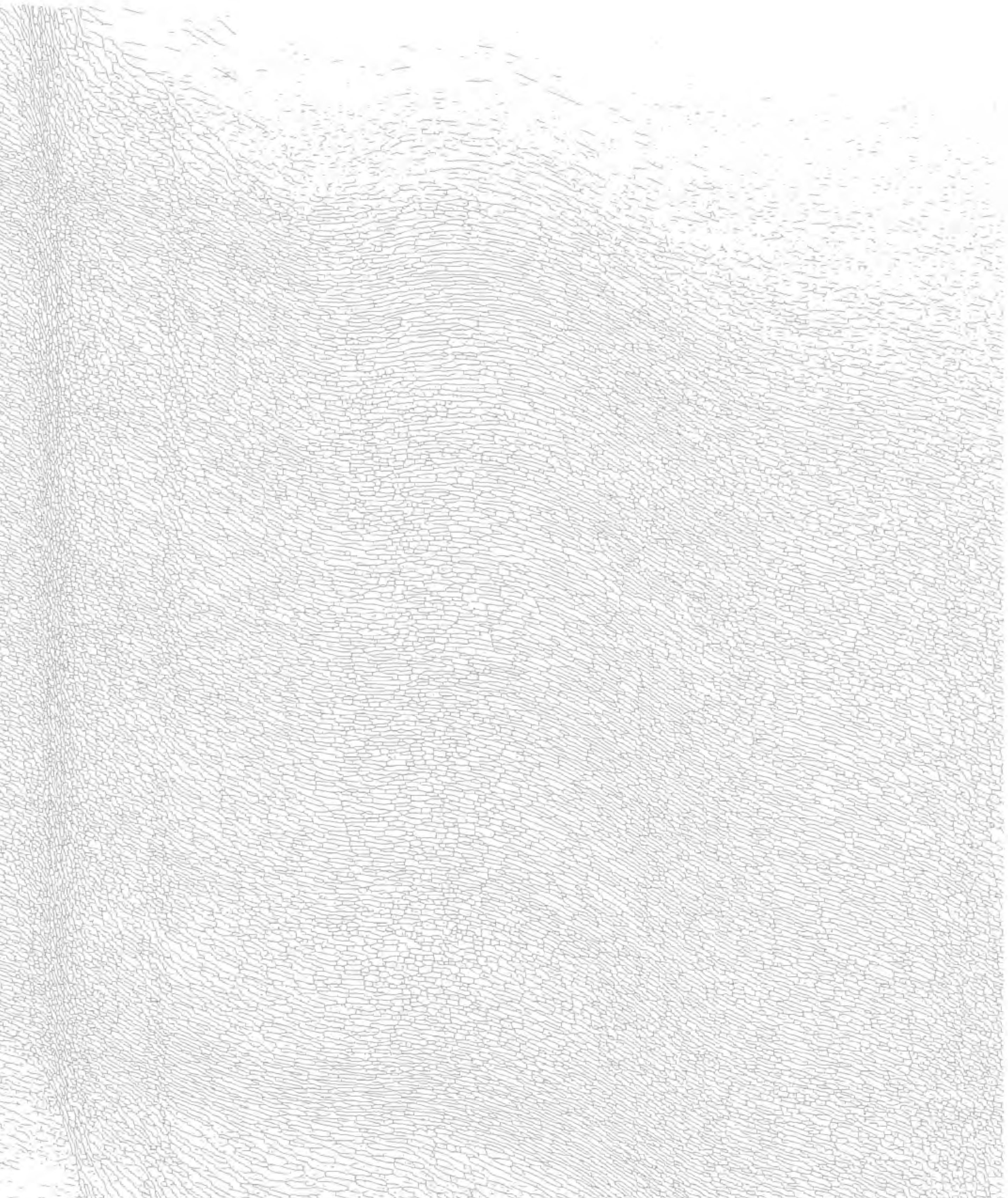
Report by Emmanuel Anguiano-Hernández
with corrections and addendums by Arjan Guerrero

Xerowritings 02 [XW02] analysis .

Thinning

Signs reduced to 1 pixel thickness



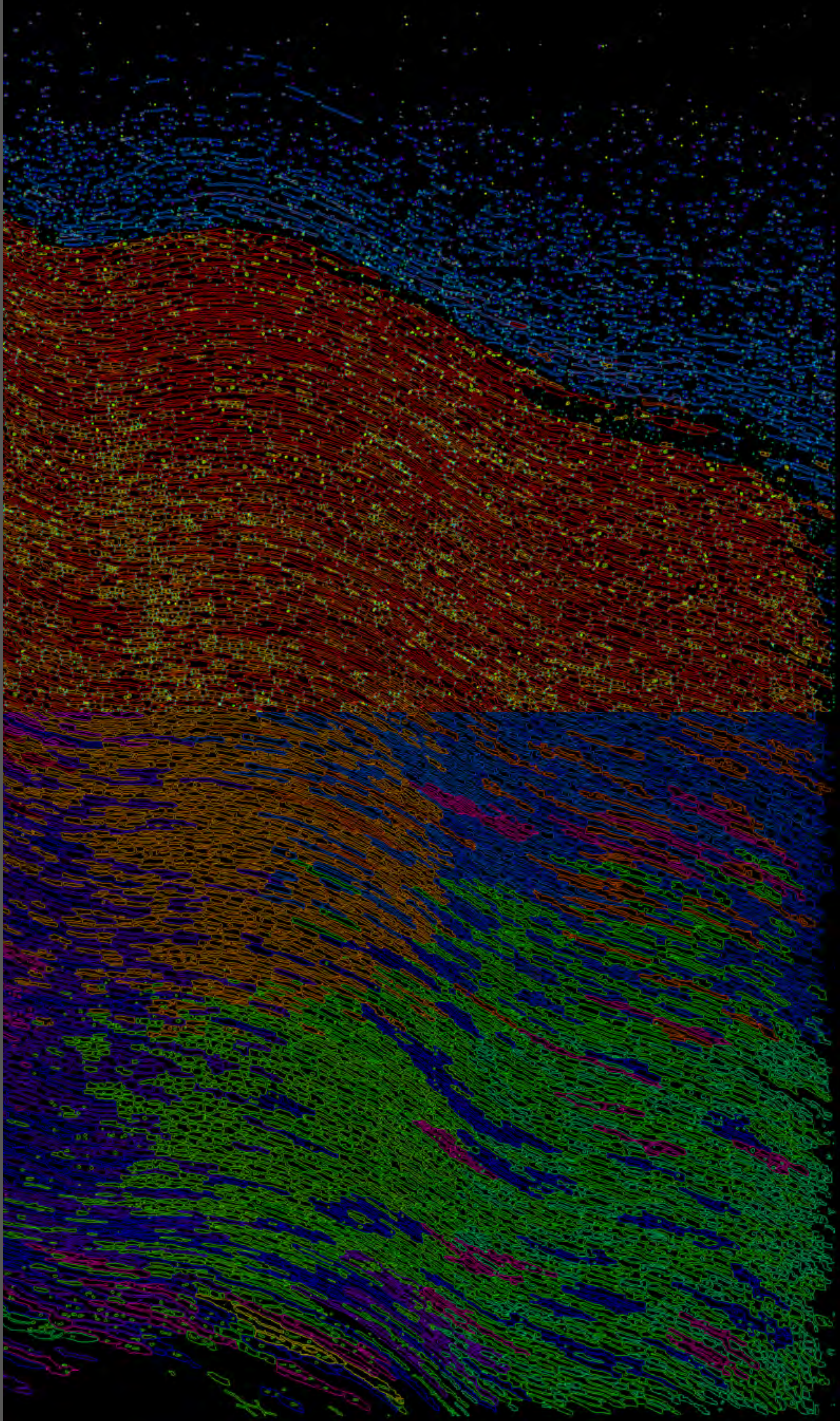


Cells vs Pods

Separation of
cells and pods

Cell type groups

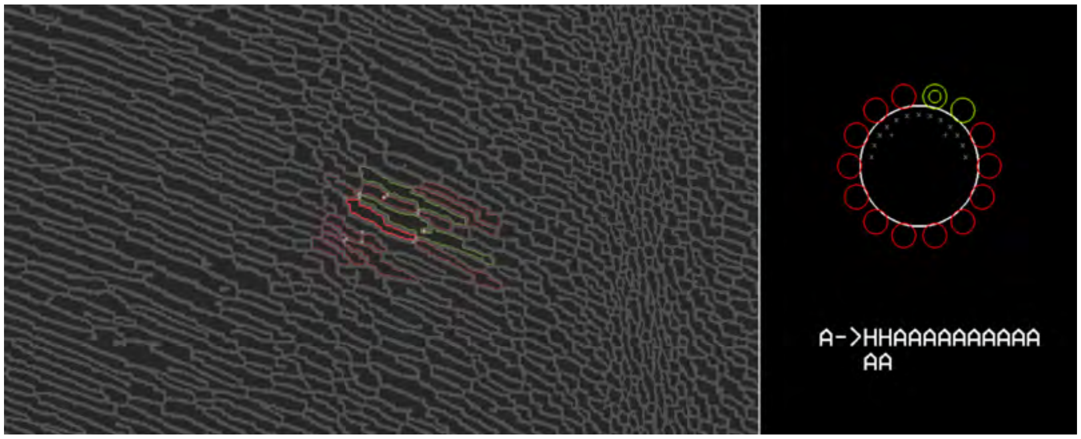
Identification of
the 31 types
of cells



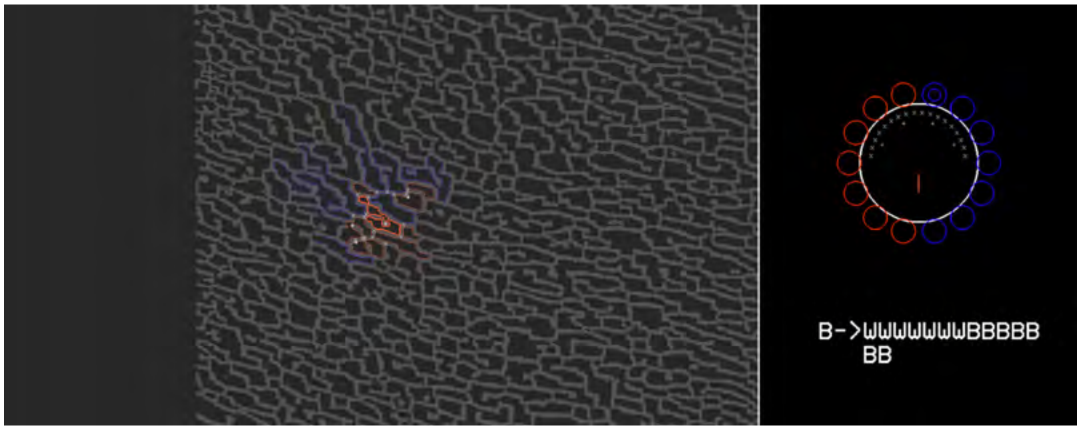
Cell types

Samples from the 31 type groups

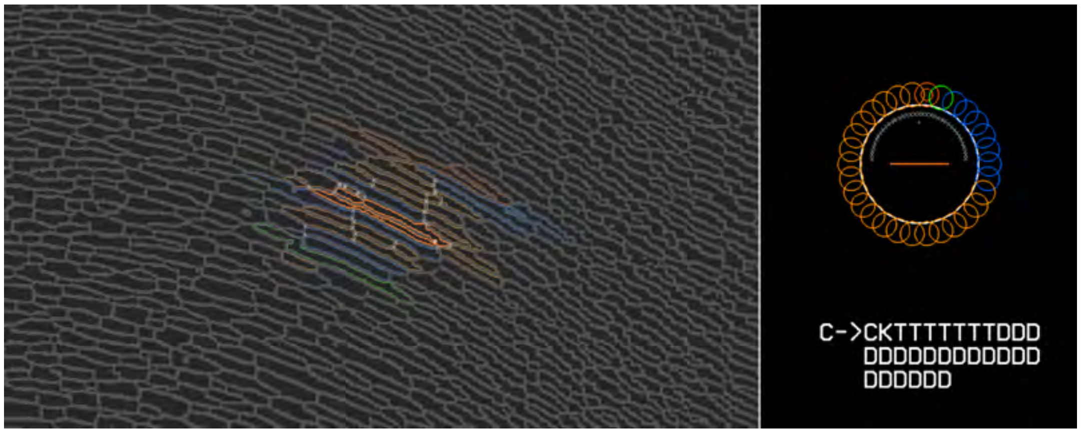
Type 0



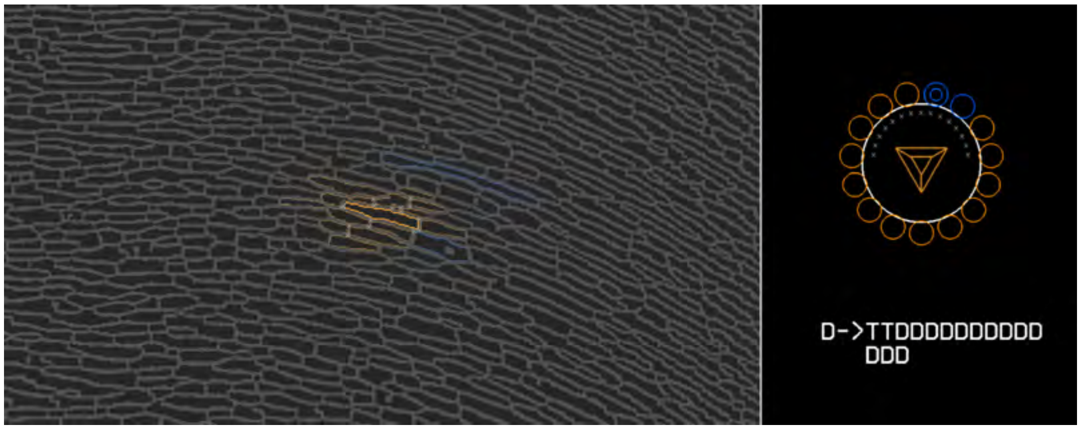
Type 1



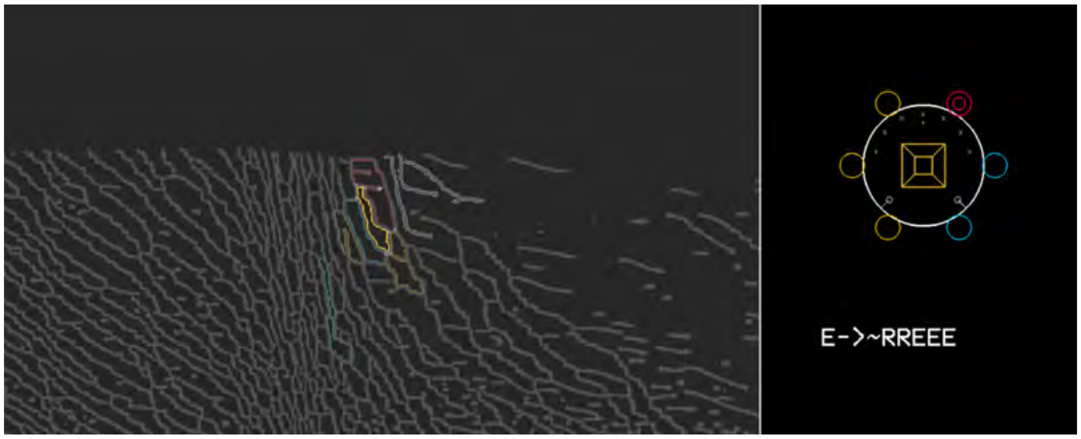
Type 2



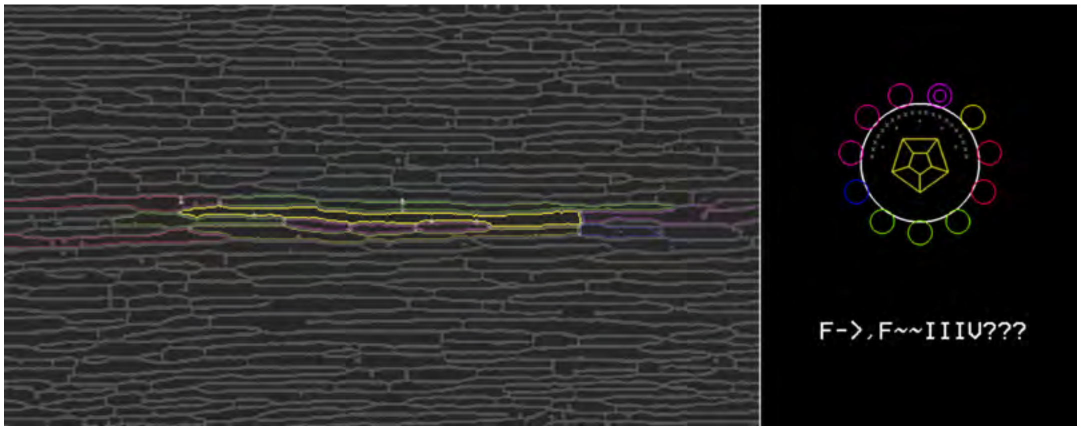
Type 3



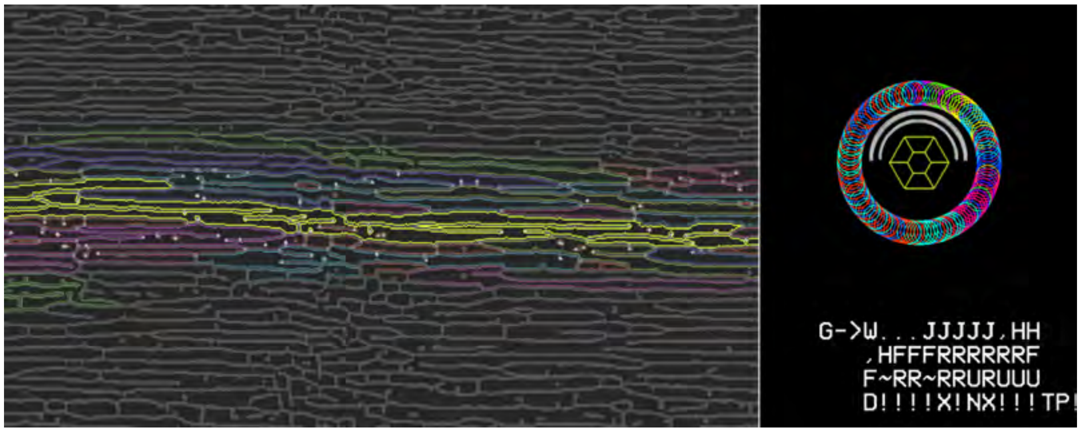
Type 4



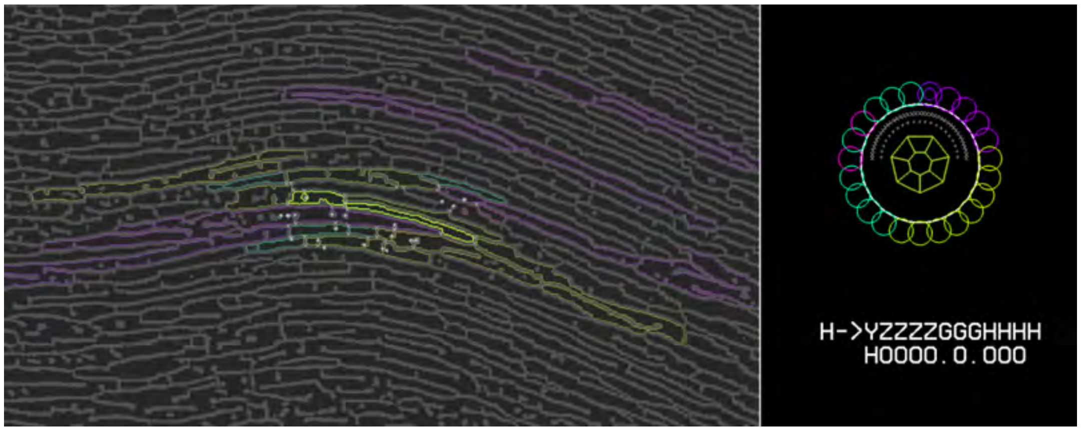
Type 5



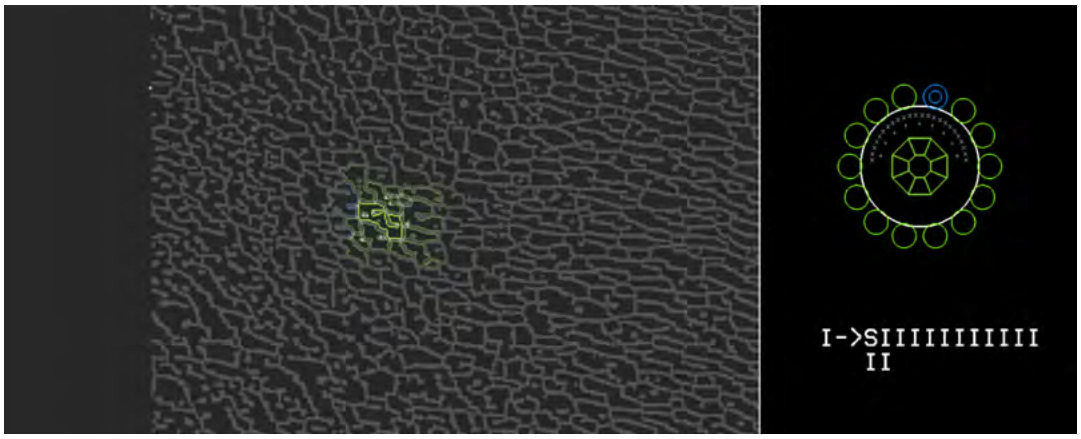
Type 6



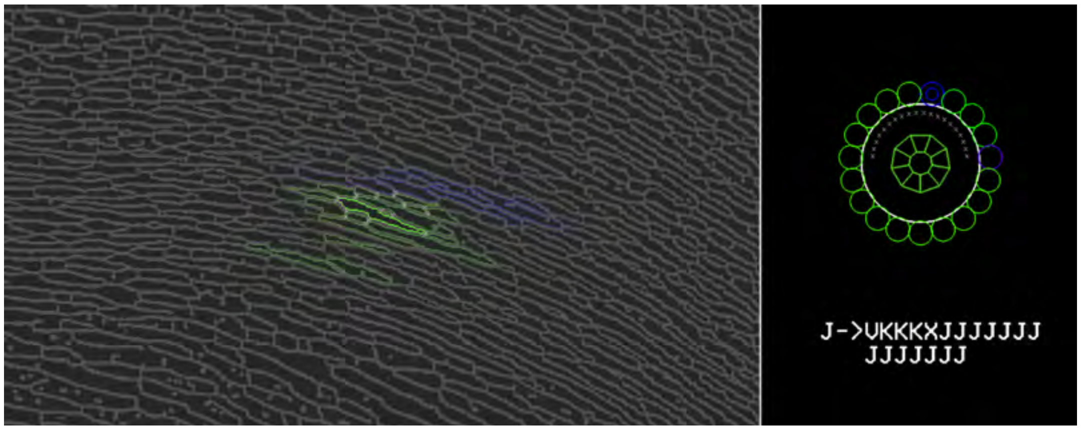
Type 7



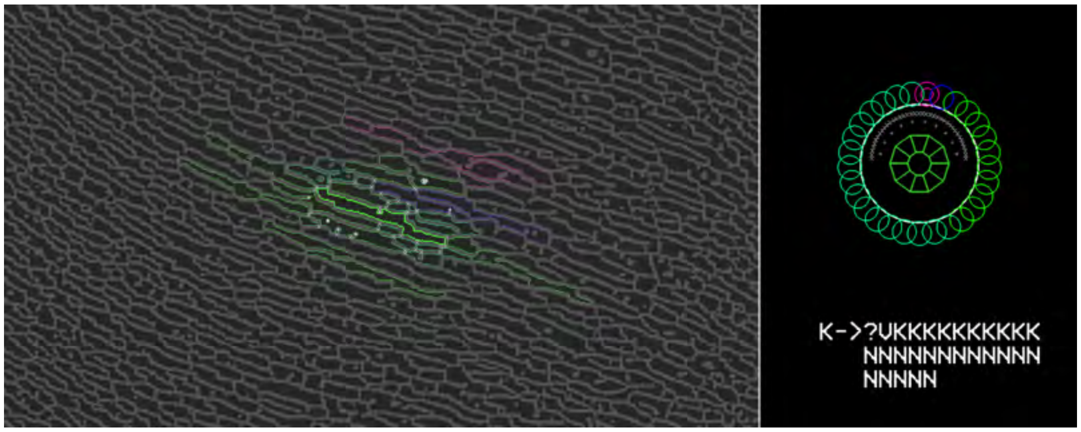
Type 8



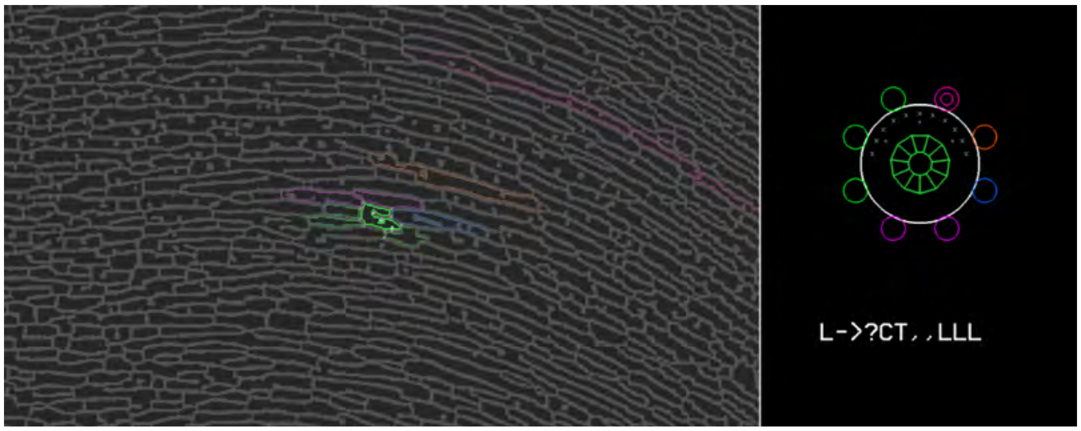
Type 9



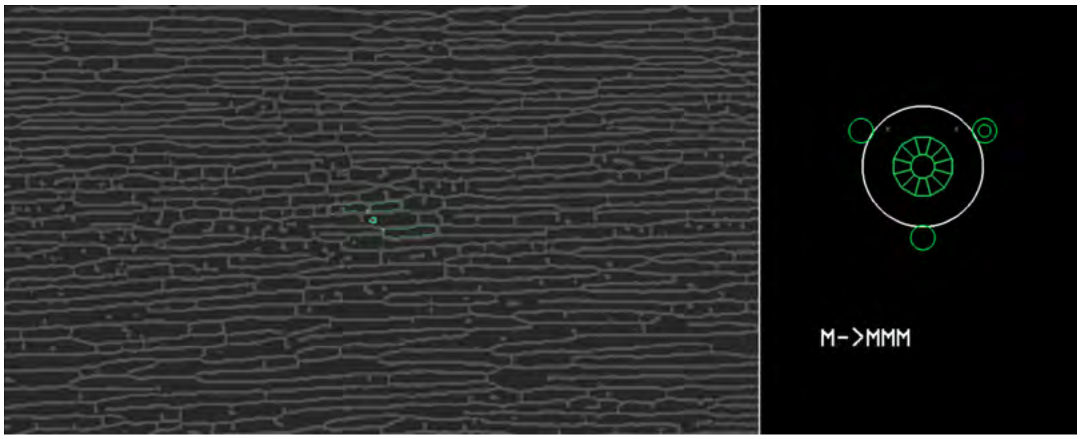
Type 10



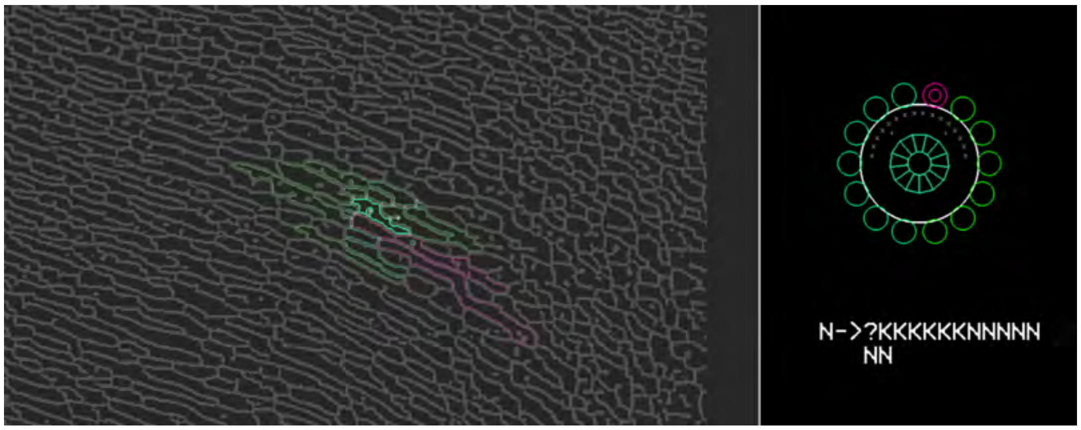
Type 11



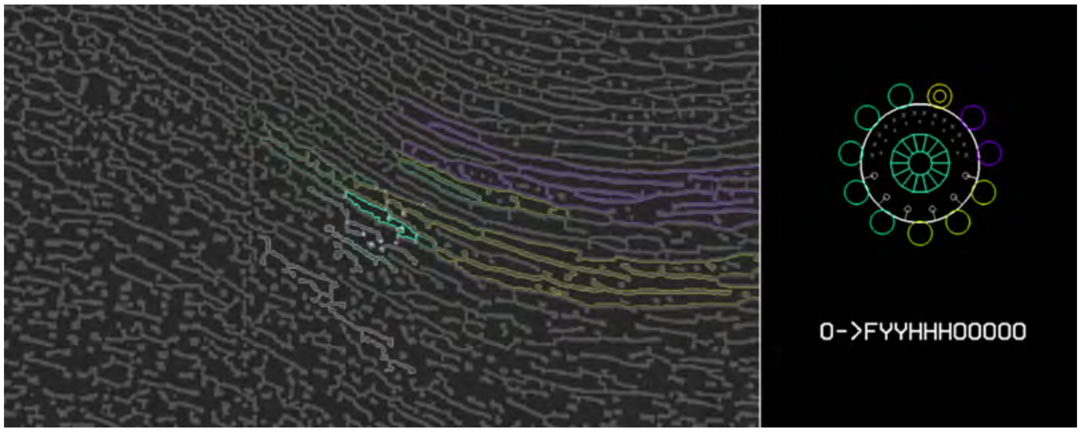
Type 12



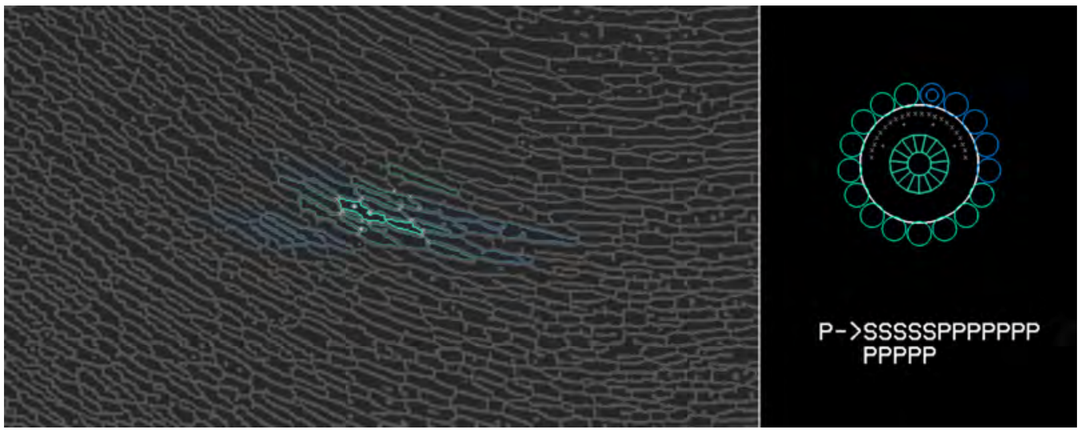
Type 13



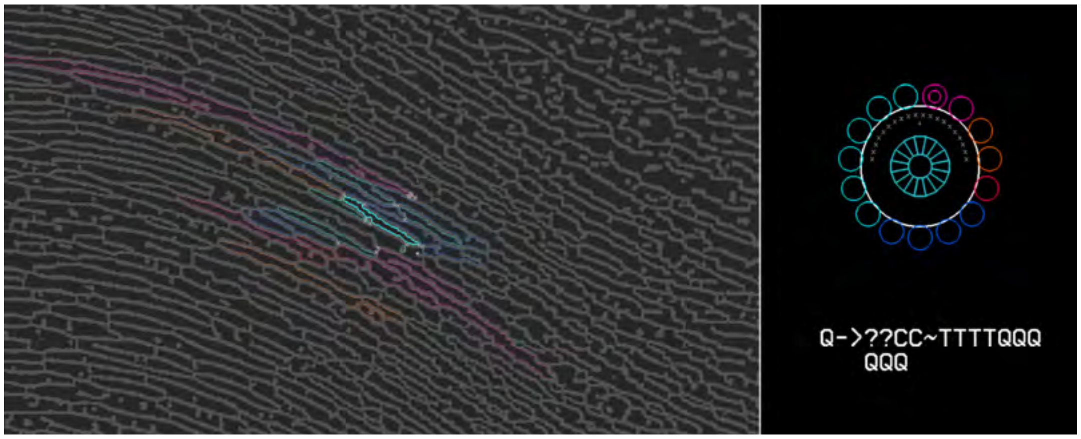
Type 14



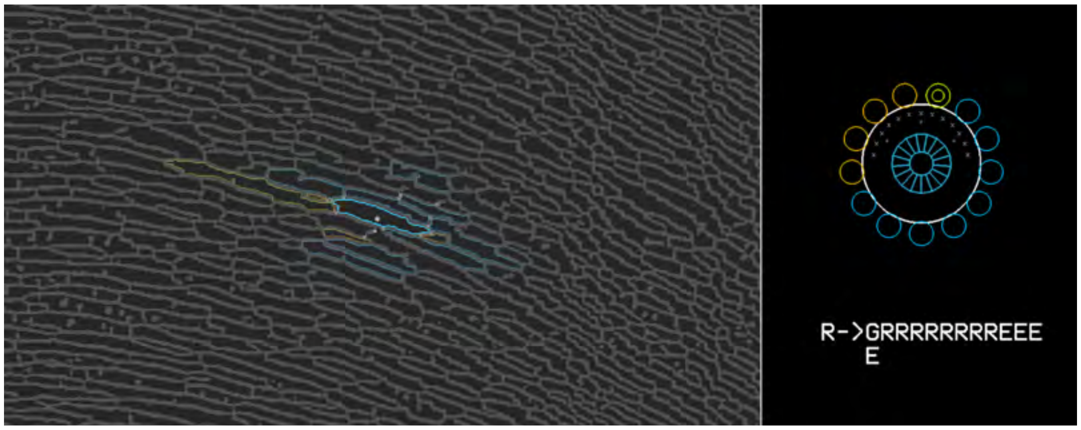
Type 15



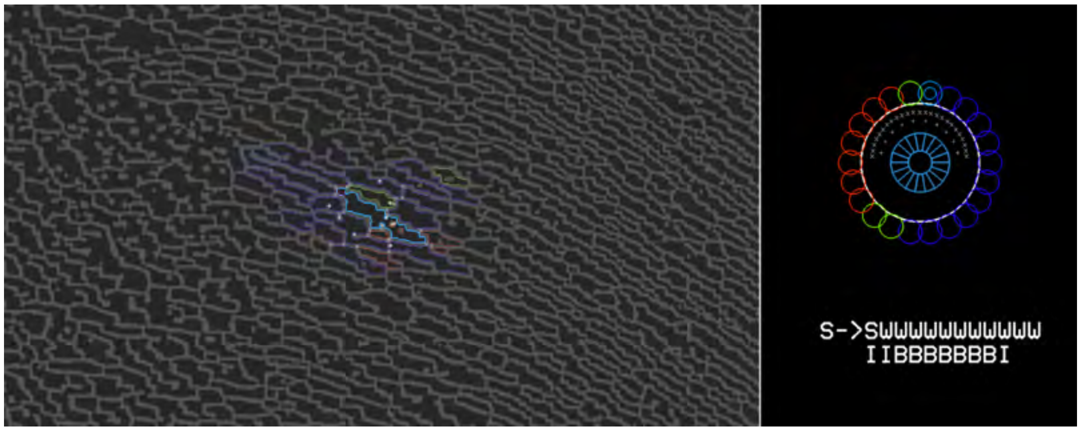
Type 16



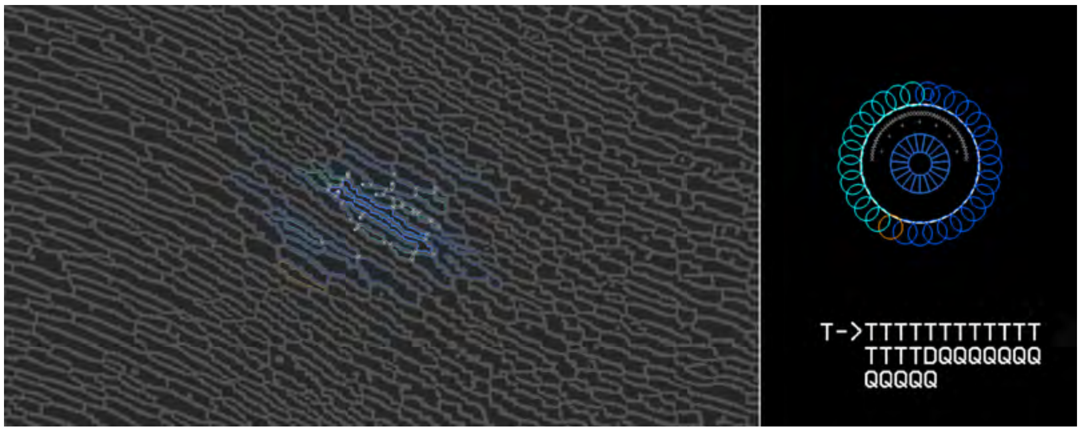
Type 17



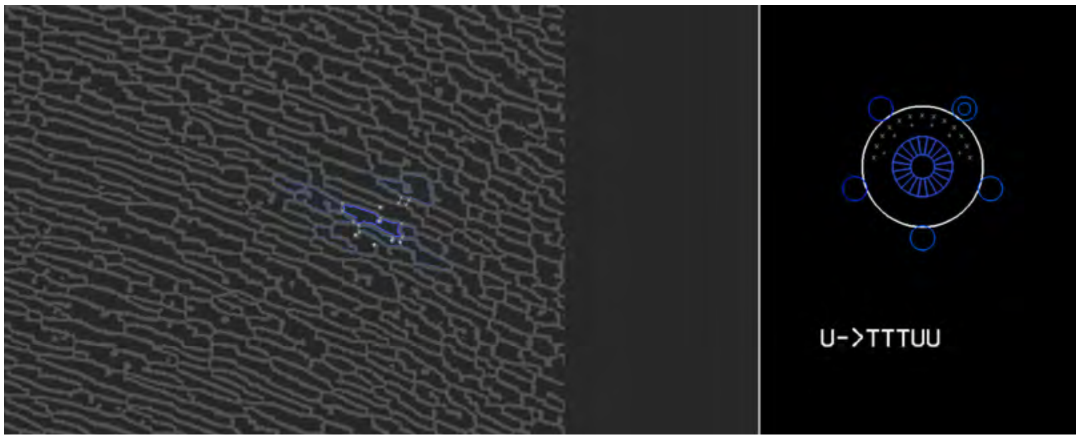
Type 18



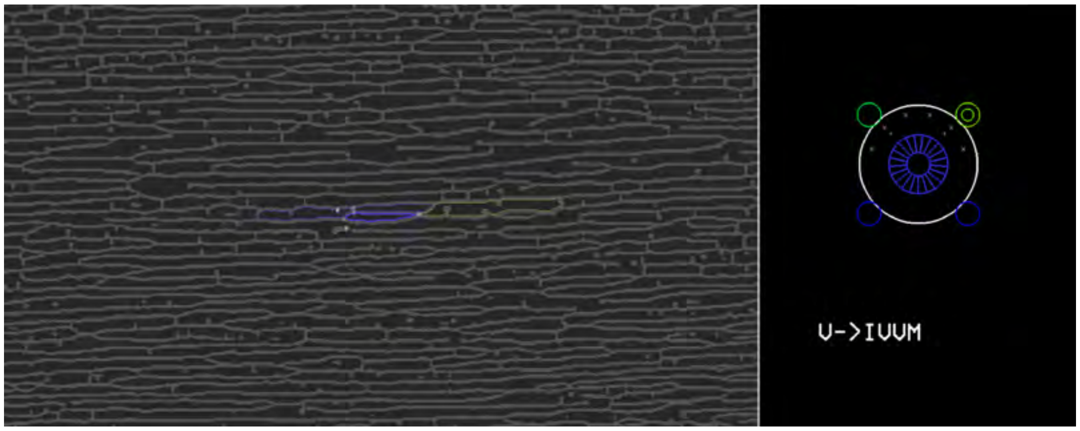
Type 19



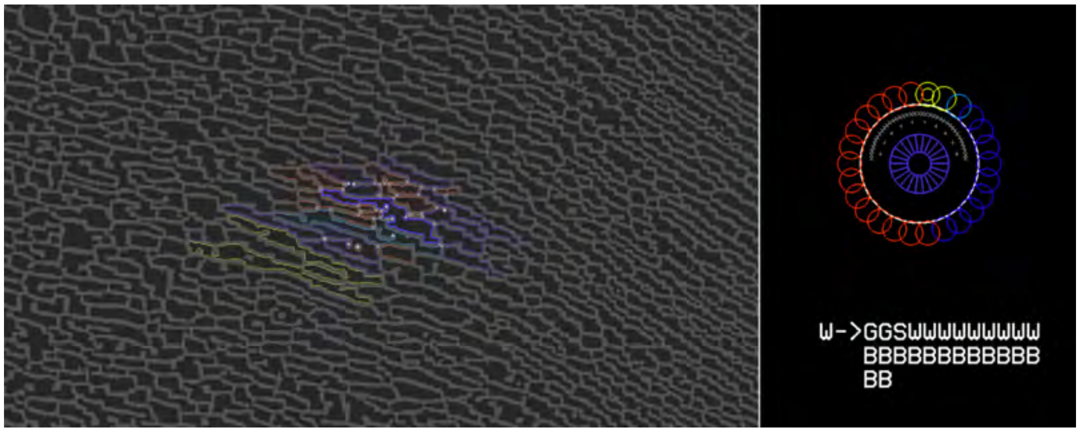
Type 20



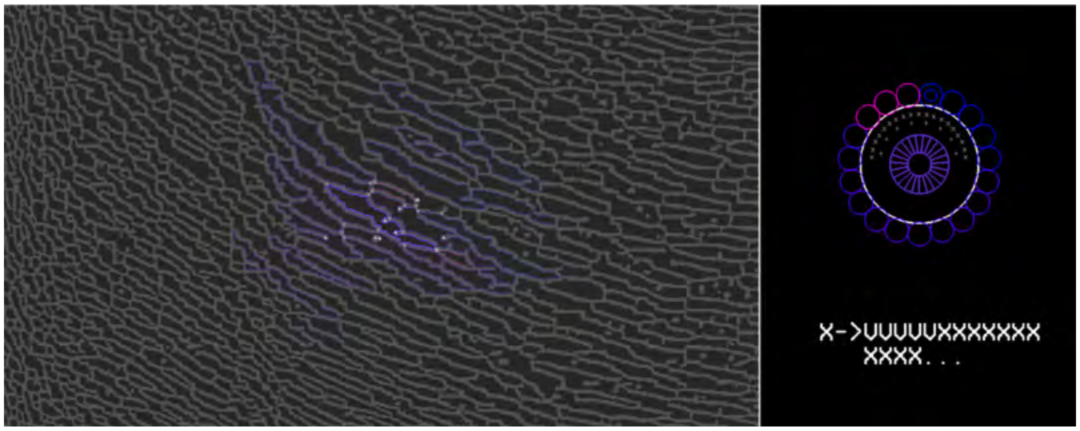
Type 21



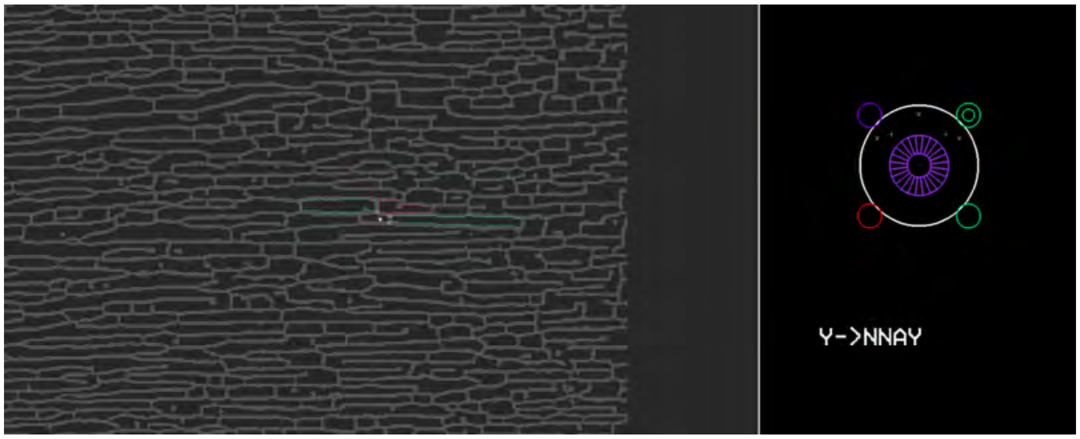
Type 22



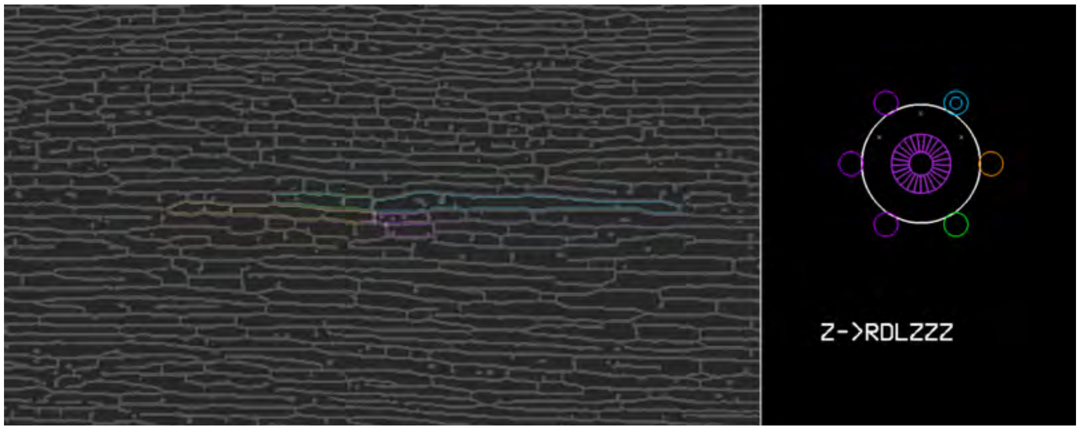
Type 23



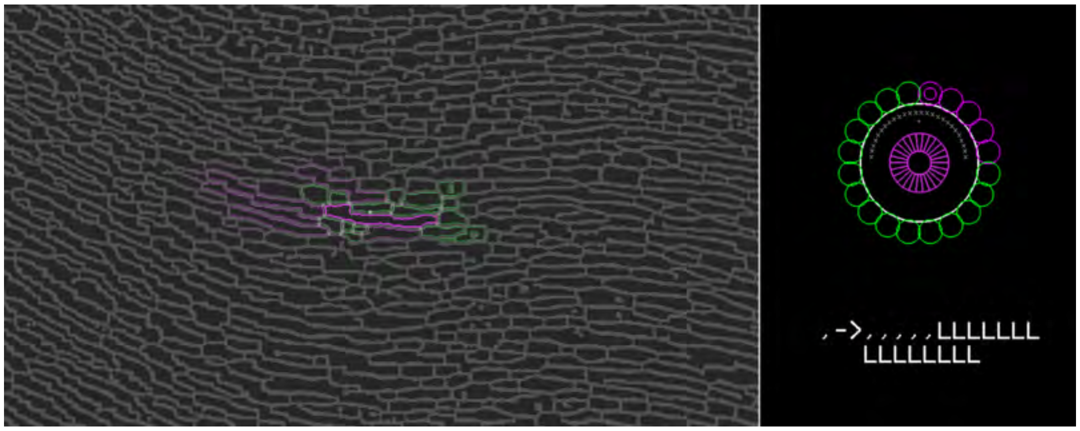
Type 24



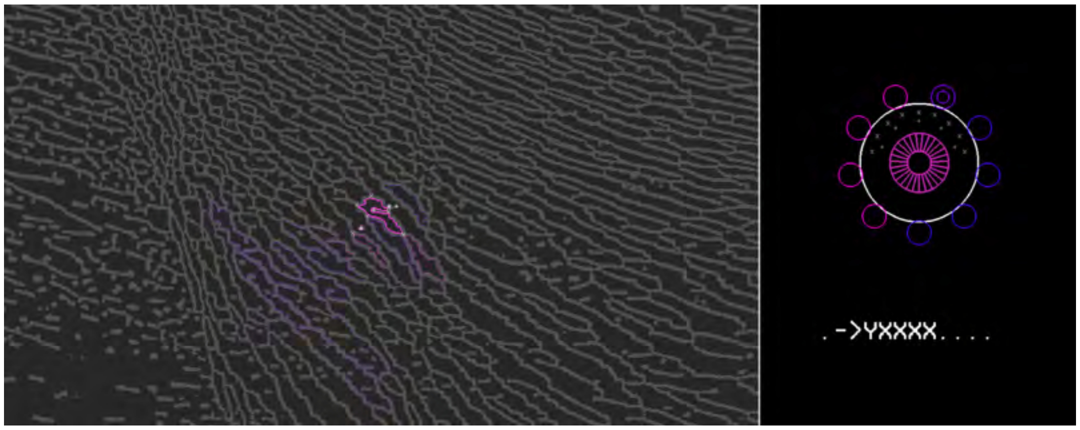
Type 25



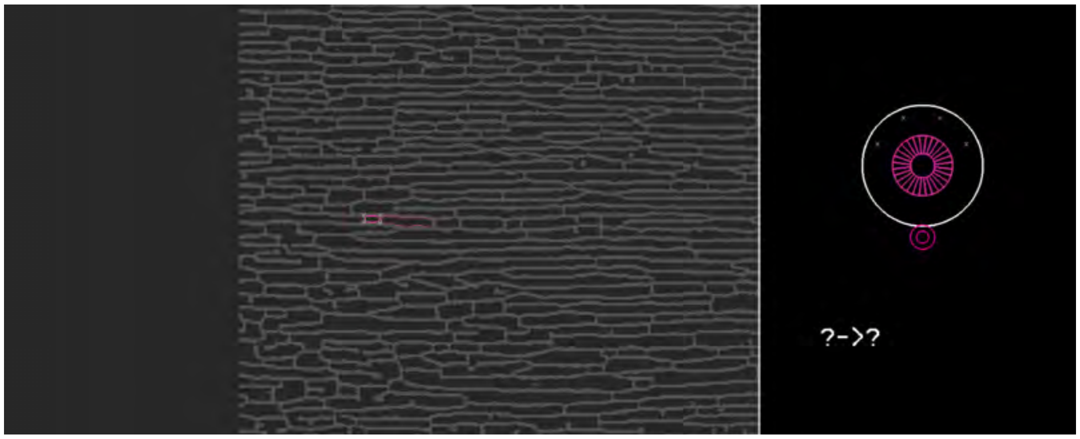
Type 26



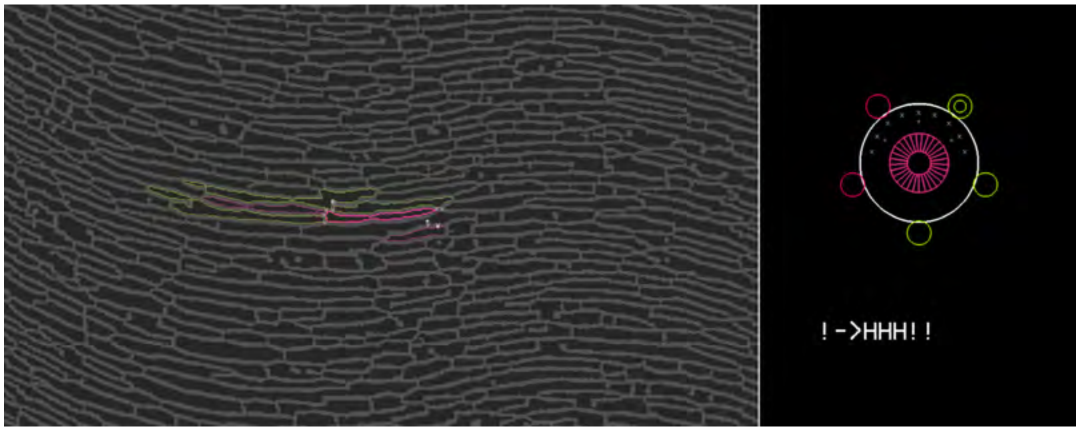
Type 27



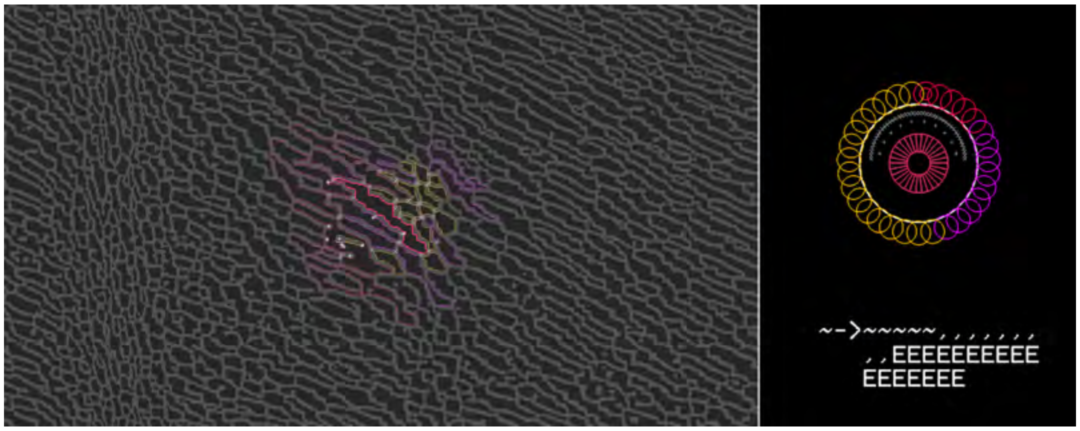
Type 28



Type 29



Type 30



[12983]	!->
[14476]	!->
[14585]	!->
[14865]	!->
[16145]	!->
[16379]	!->
[17402]	!->
[6400]	!->!
[10944]	!->!
[13008]	!->!
[15373]	!->!
[7238]	!->!!
[11870]	!->!!
[12122]	!->!!
[12661]	!->!!
[13063]	!->!!
[14677]	!->!!
[15522]	!->!!
[16503]	!->!!
[16513]	!->!!
[16834]	!->!!
[17488]	!->!!
[17490]	!->!!
[8664]	!->!!!
[8921]	!->!!!
[9314]	!->!!!
[9370]	!->!!!
[9702]	!->!!!
[9800]	!->!!!
[12624]	!->!!!
[12788]	!->!!!
[13785]	!->!!!
[14039]	!->!!!
[15103]	!->!!!
[15343]	!->!!!
[15882]	!->!!!
[15975]	!->!!!
[16105]	!->!!!
[17513]	!->!!!
[6868]	!->!!!!
[7725]	!->!!!!
[8074]	!->!!!!
[9343]	!->!!!!
[10105]	!->!!!!
[10779]	!->!!!!
[10786]	!->!!!!
[11201]	!->!!!!
[11647]	!->!!!!
[12067]	!->!!!!
[12573]	!->!!!!
[12787]	!->!!!!
[13156]	!->!!!!
[15295]	!->!!!!
[15726]	!->!!!!
[16040]	!->!!!!
[5186]	!->!!!!!
[6360]	!->!!!!!
[8550]	!->!!!!!
[9975]	!->!!!!!
[10765]	!->!!!!!
[12079]	!->!!!!!
[15900]	!->!!!!!

```
[5805]          !->!!!!!!
[6657]          !->!!!!!!
[9360]          !->!!!!!!
[9804]          !->!!!!!!
[11232]         !->!!!!!!
[11519]         !->!!!!!!
[13915]         !->!!!!!!
[15117]         !->!!!!!!
[4084]          !->!!!!!!!
[4495]          !->!!!!!!!
[9315]          !->!!!!!!!
[11676]         !->!!!!!!!
[12253]         !->!!!!!!!
[13610]         !->!!!!!!!
[10331]         !->!!!!!!!!
[6885]          !->!!!!!!!!!
[7770]          !->!!!!!!!!!
[11852]         !->!!!!!!!!!!
[6460]          !->!!!!!!!!!!!!!
[5716]          !->!!!!!!!!!!!!!!
[6988]          !->!!!!!!P!PPPP
[9913]          !->!!!!!!PPPP
[7960]          !->!!!!!P!P
[6432]          !->!!!!!PPPP
[8358]          !->!!!!!PPPPP
[11575]         !->!!!!P
[12273]         !->G!
[10486]         !->G!!
[12159]         !->G!!
[5040]          !->G!!!
[13140]         !->G!!!
[10231]         !->G!!!!
[14366]         !->G!!!!
[9440]          !->G!!!!!
[9700]          !->G!!!!!
[16233]         !->G!!!!!
[6762]          !->G!!!!!!!
[9650]          !->G!!!!!!!
[4278]          !->G!!!!!!!!!
[8613]          !->G!!!!!!!!!!!
[7624]          !->G!!!!!!!!!!!!!
[5250]          !->GGGH!
[4070]          !->GGGH!!!!OOO
[8130]          !->GGGH!!O
[5895]          !->GGGHH!!!!!!!
[5713]          !->GGGHHH!!AAAA
[5499]          !->GGGHHHHH!OO
[5329]          !->GGGSSSSSWWWWWW!WW!!!BB!!!B
[12046]         !->GGH!!!
[3983]          !->GGH!!!!!!
[5543]          !->GGH!!OOOO
[4362]          !->GGHH!!!
[6592]          !->GGHH!!!!!!
[4566]          !->GGHH!O
[6897]          !->GGHHH!!!
[8614]          !->GGHHHH
[4994]          !->GGHHHH!A
[3335]          !->GGHHHHH!!!!!!
[4321]          !->GGHHHHHH!!!AAAA
[3960]          !->GGHHHHHHHHHH!!!!!!!
[5975]          !->GGHHHOOOO
[6999]          !->GGHHHRRR!!!!
[8834]          !->GGHHWW!!!!!!!
[11789]         !->GGHHWWWW!!!
[6045]          !->GGHHWWWWWWWWW!!!!!!!
[5194]          !->GGOO
[10580]         !->GGR!!EE
[6497]          !->GGRRRR!!!
[7421]          !->GGRRRRR!EEE
```

And it continues...

Thus wrote Kyo.

Kyo, do you copy?

```
+k[3131]
+y[6495]
+o[3682]
+,[286]

[cell]:
                [c, nP, age]: 107, 39
[cell]:
                [c, nP, age]: 121, 22
[cell]:
                [c, nP, age]: 111, 35
[cell]:
                [c, nP, age]: 123, 196
```

```
+d[2523]
+o[2996]

[cell]:
                [c, nP, age]: 100, 87
[cell]:
                [c, nP, age]: 111, 55
```

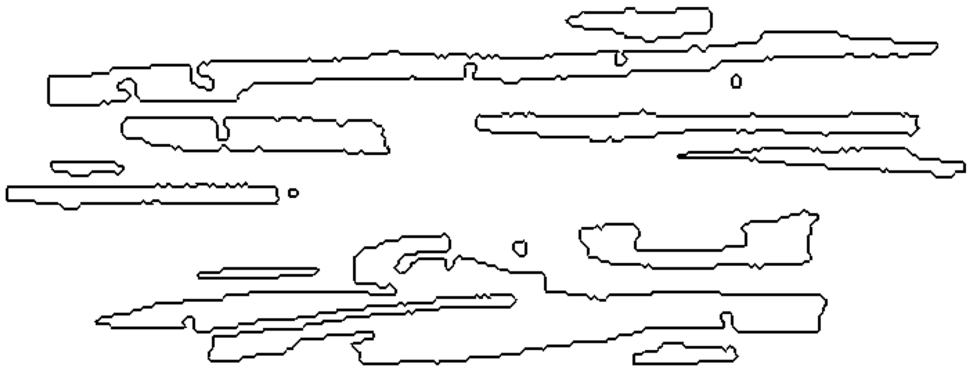
```
+y[6109]
+o[1485]
+u[7976]

[cell]:
                [c, nP, age]: 121, 20
[cell]:
                [c, nP, age]: 111, 84
[cell]:
                [c, nP, age]: 117, 17
```

```
+c[90]
+o[2090]
+p[7724]
+y[4730]
+?[9216]

[cell]:
                [c, nP, age]: 99, 321
[cell]:
                [c, nP, age]: 111, 92
[cell]:
                [c, nP, age]: 112, 12
[cell]:
                [c, nP, age]: 121, 42
[cell]:
                [c, nP, age]: 125, 10
```

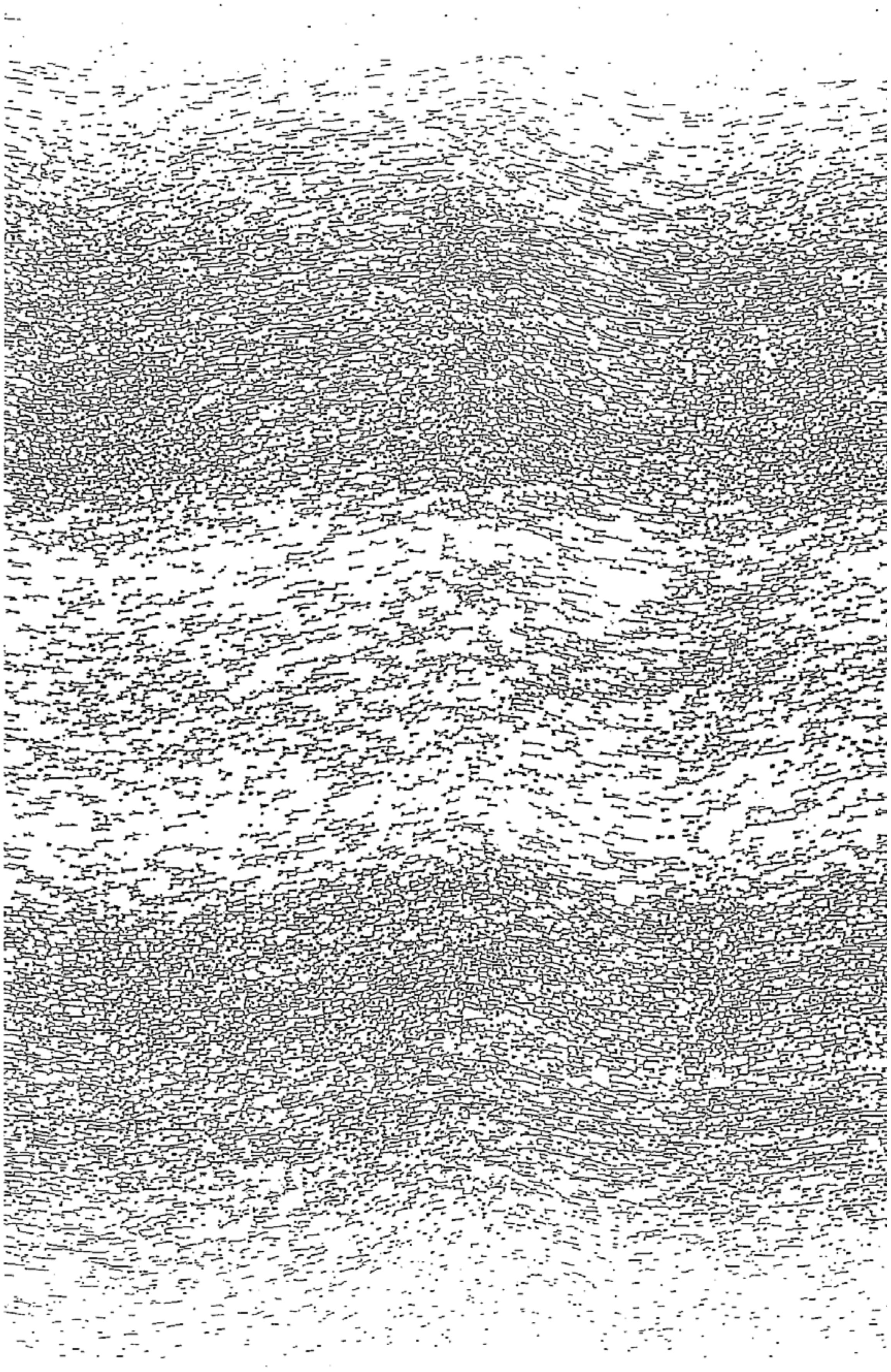
Complete
phrase

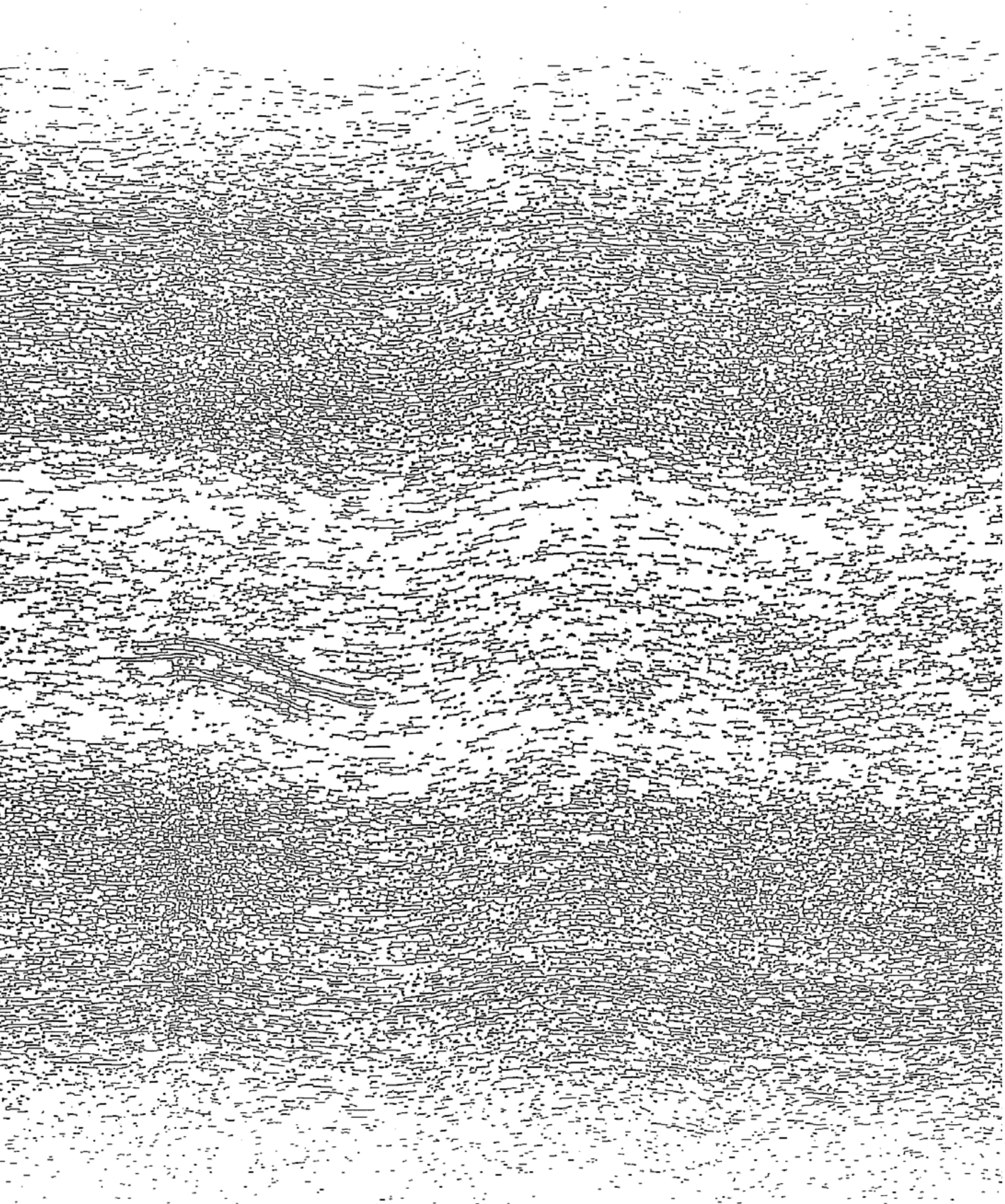


Answer analysis

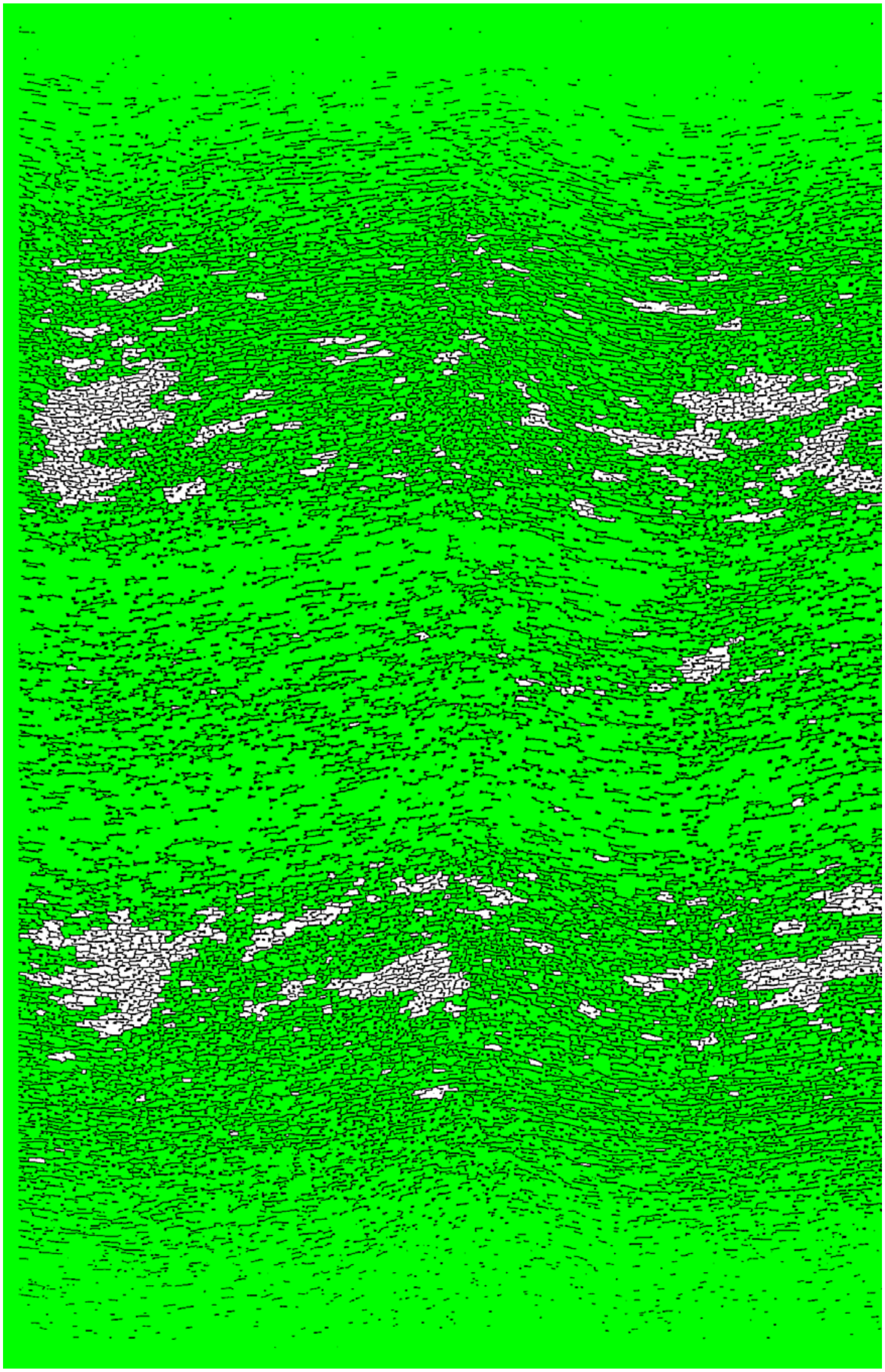
Complete answer

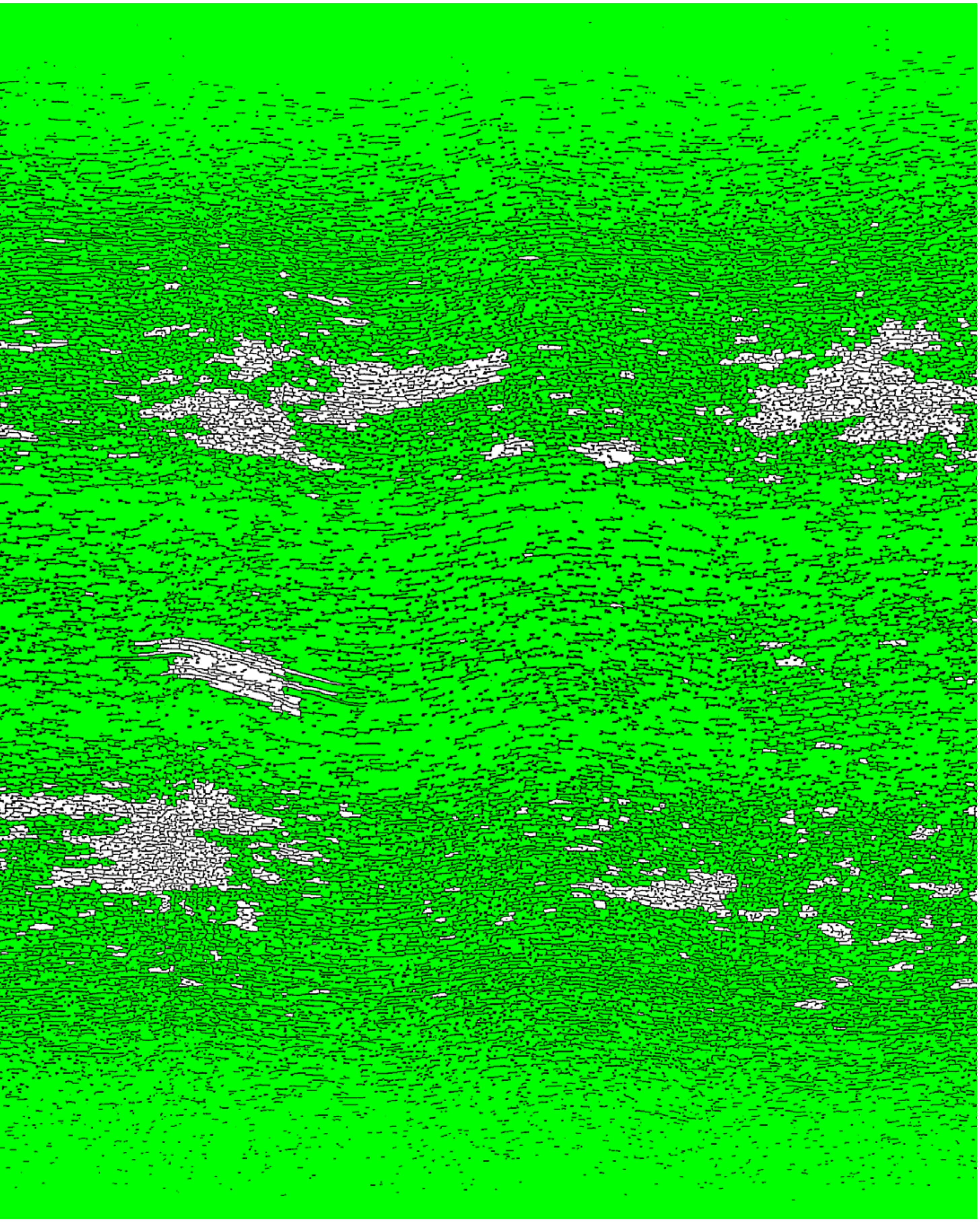
Final transformation of the question and additional generated text





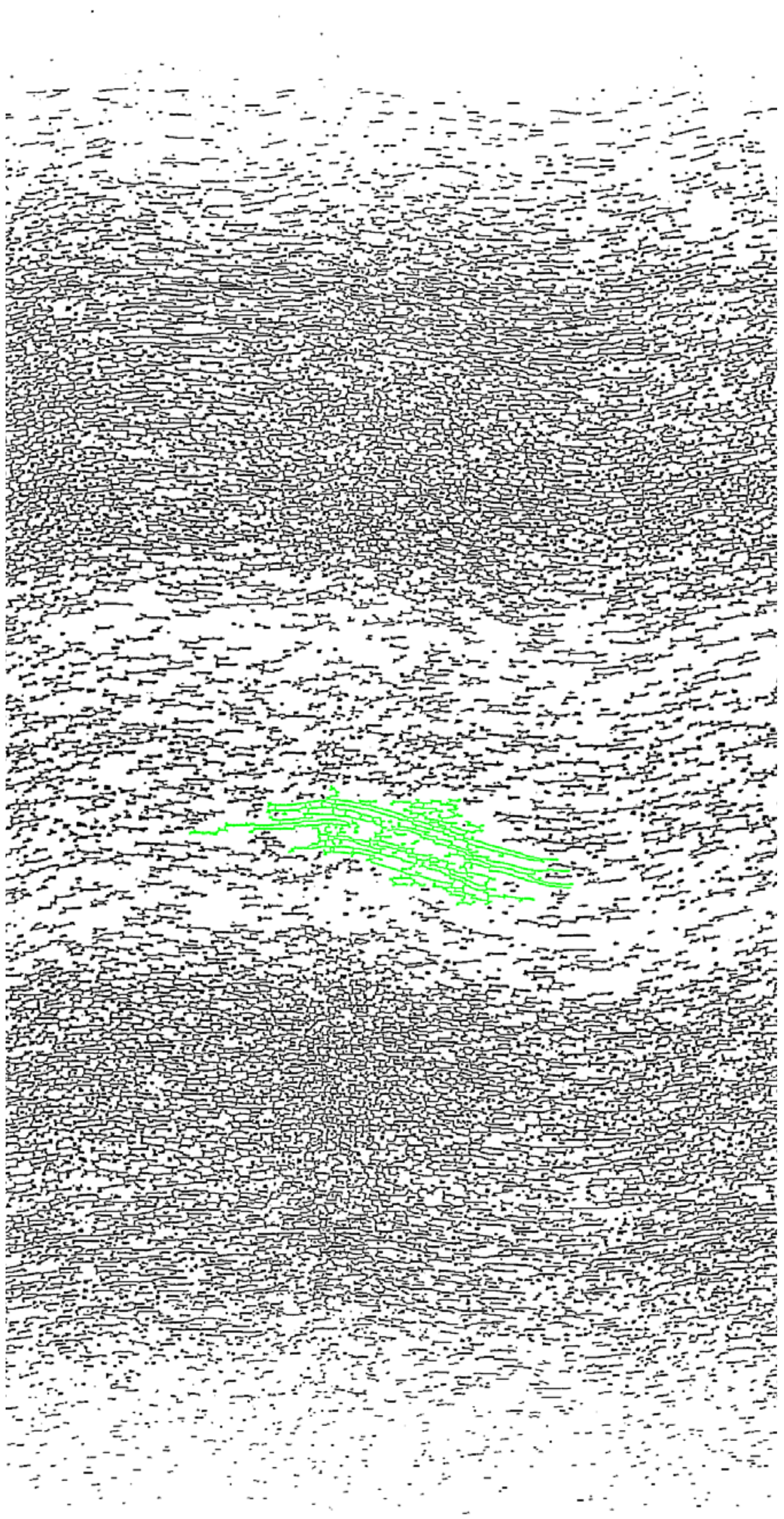
Identification of cell clusters





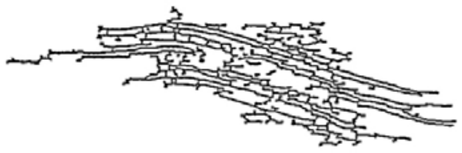
Answer cluster

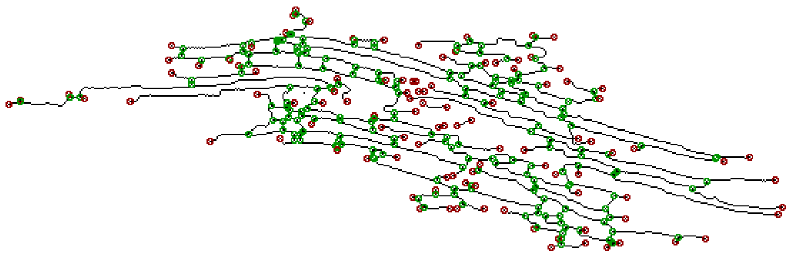
Highlight of the answer’s cell cluster

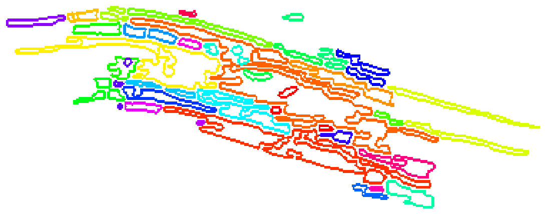


Answer isolation

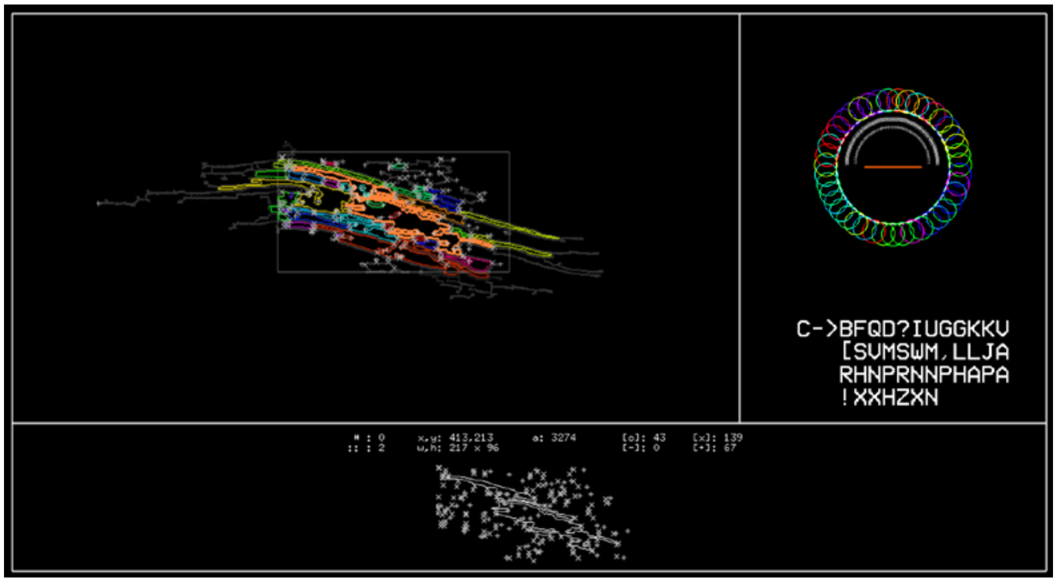
Extraction of the transformed question





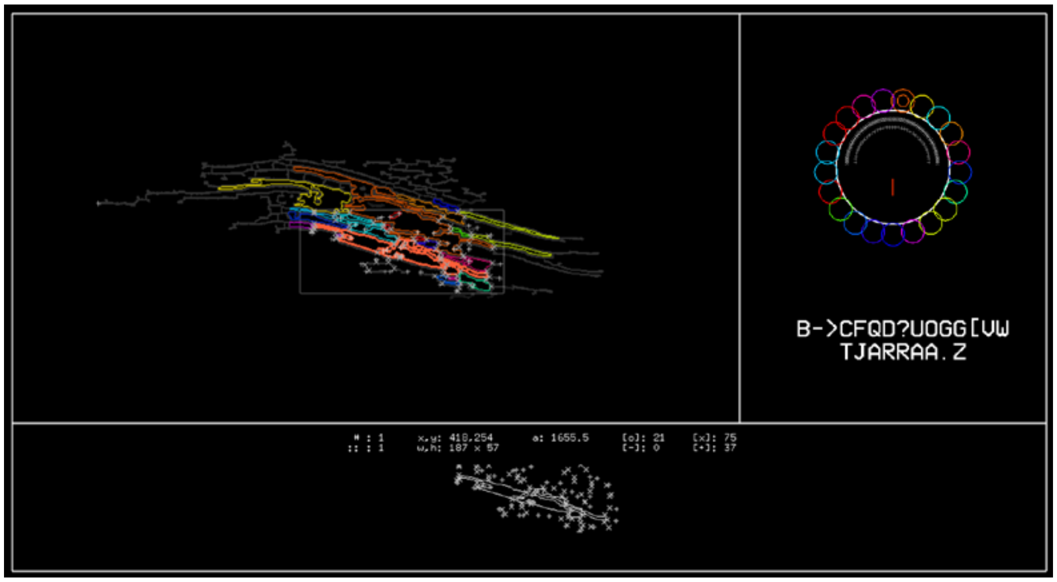


Cells IDs



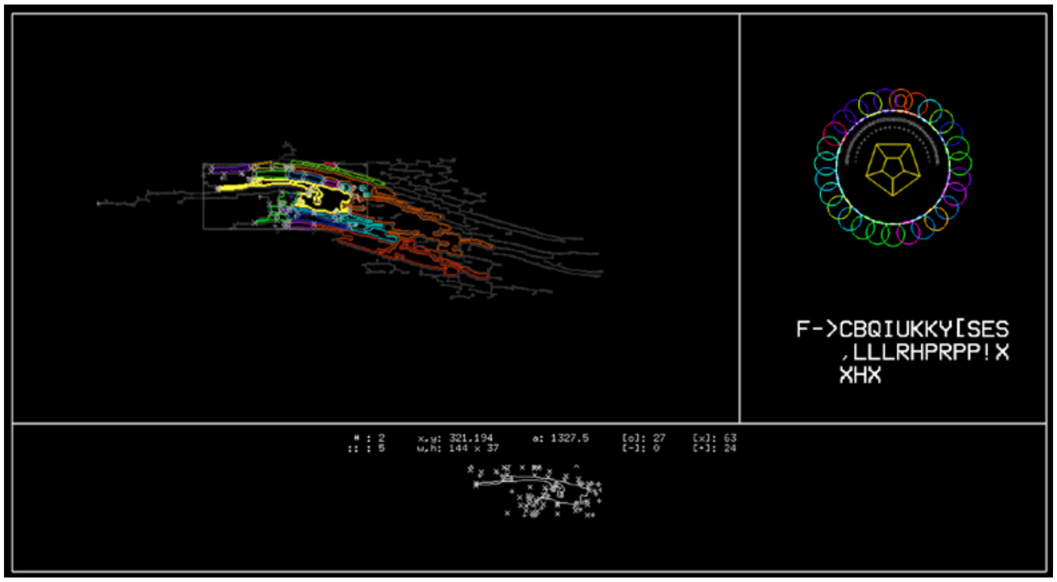
```
C->BFQD?IUGGKKV
   [SVMSWM,LLJA
   RHNPRNNPHAPA
   !XXHZXN
```

```
 # : 0    x,y: 413,213     a: 3274      [o]: 43    [x]: 139
:: : 2    w,h: 217 x 96                 [-]: 0     [+]: 67
```



```
B->CFQD?UOGG[VW
   TJARRAA.Z
```

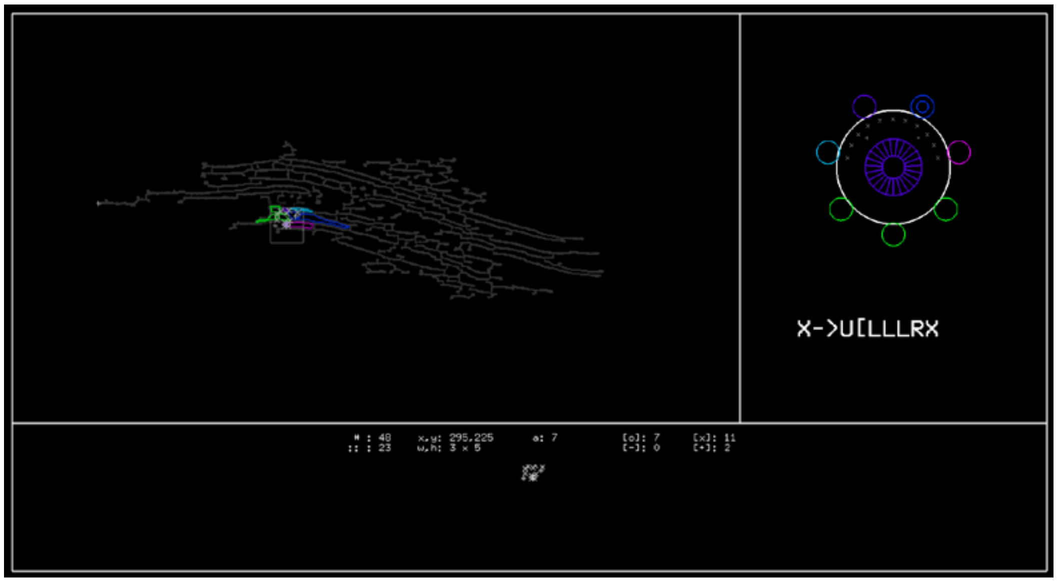
```
 # : 1    x,y: 418,254     a: 1655.5    [o]: 21    [x]: 75
:: : 1    w,h: 187 x 57                 [-]: 0     [+]: 37
```



```
F->CBQIUKKY[SES
   ,LLLRHPRPP!X
   XHX
```

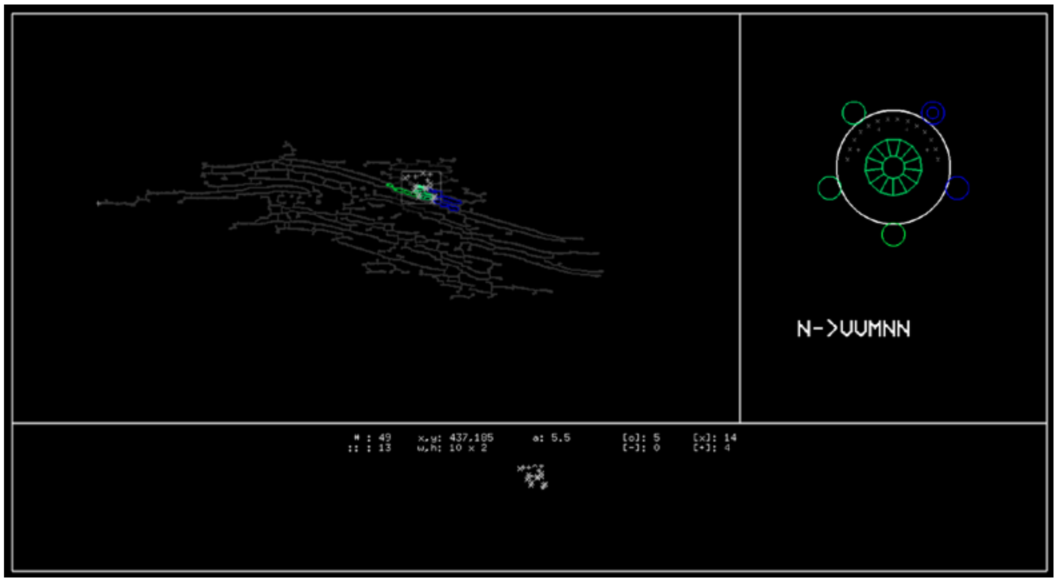
```
 # : 2    x,y: 321,194     a: 1327.5    [o]: 27    [x]: 63
:: : 5    w,h: 144 x 37                 [-]: 0     [+]: 24
```

And it continues...



```
X->U[LLLRX

 # : 48    x,y: 295,225     a: 7        [o]: 7    [x]: 11
:: : 23    w,h: 3 x 5                   [-]: 0    [+]: 2
```



```
N->VUMNN

 # : 49    x,y: 437,185     a: 5.5      [o]: 5    [x]: 14
:: : 13    w,h: 10 x 2                  [-]: 0    [+]: 4
```

Answer codebook

Transliteration to latin alphabet

[0042]	!->S
[0024]	,->FSP
[0043]	.->?OT
[0005]	?->CBOGGWT.
[0041]	A->QDA
[0029]	A->QDMA
[0039]	A->W
[0001]	B->CFQD?UOGG[VWTJARRAA.Z
[0000]	C->BFQD?IUGGKKV[SVMSWM,LLJARHNPRNNPHAPA!XXHZXN
[0004]	D->QIGGVVMMJANNAAN
[0017]	E->KYHH
[0002]	F->CBQIUKKY[SES,LLLRHPRPP!XXHX
[0010]	G->BD?GVVJ
[0009]	G->CBD?GJ
[0031]	H->CIKESHH
[0038]	H->CIKESHH
[0046]	H->CIKSSHH
[0006]	I->CFDKKSMS,HNPPHP!XH
[0028]	J->DGG
[0011]	K->CIYESHHH
[0012]	K->IUSSLLLRXX

```
[0025]          L->UK[LLRXX
[0027]          L->UK[LLRXX
[0026]          L->UK[LLRXXX
[0019]          M->DIVMNNN
[0022]          M->QDMAPP
[0032]          N->I
[0035]          N->VVMNN
[0036]          N->VVMNN
[0049]          N->VVMNN
[0008]          O->B?T.
[0033]          P->F,PP
[0037]          P->FMPP
[0040]          P->FMPP
[0003]          Q->FDUMARAZ
[0030]          R->BQ[R
[0034]          R->BUK[LLLRXXX
[0020]          S->CIKKSHHXH
[0016]          S->CKS,!XH
[0023]          T->?O.
[0007]          U->BFQK[LLLRRXXZX
[0014]          V->DGVMNNN
[0018]          V->DGVNNN
[0021]          W->?A
[0044]          X->IUKSSLRX
[0045]          X->UK[LLLRXX
[0048]          X->U[LLLRX
[0013]          Y->FKE
[0047]          Z->QU
[0015]          [->BULLLRRXX
```

Media Forensis

Is a project dedicated to investigate the agency of media by inverting the subject-object relationship with media. As computational technology is having an increasingly large role in world building, mediating within artistic forums for media to display its agency is one step forward in understanding the place of human agency in a less human-centric future.

Xerogrammar Case team:

Arjan Guerrero – principal investigator

Emmanuel Anguiano-Hernandez – programming

Ika Atl – production assistant

Oliver Davidson Vejar – linguistics consultant

Yazmín Hidalgo – editorial production

mediaforensis.agency