TERM PROJECT

Generative Art with Cellular Automata

An exploratory study for generation of Abstract Designs and Patterns with Cellular Automata

By:

Adarsh Kumar (18MA20003)
Department of Mathematics

Indian Institute of Technology Kharagpur
adarshkumar712@iikgp.ac.in

Tanishq Kishani (18IM10042)
Department of Industrial and Systems Engineering

Indian Institute of Technology Kharagpur
tanishqk@iitkgp.ac.in

Abstract

Cellular automata (CA) are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions. In the past, many researchers have demonstrated how to apply this form-making methodology on architectural design, visual design, and even music composing. Here in this project, we try to explore a few of these generative ideas to create abstract visual designs and patterns. The overall assessment is divided into three subparts namely 1D Cellular Automata, 2D Cellular Automata and GA based coordination of Cellular Automata.

Under 1D Cellular Automata, we studied and explored various 1D rules to generate 1D Patterns which were then stacked further to generate 1D Rule base 2-Dimensional lattices of intriguing patterns and texture designs.

Under 2D Cellular Automata, we identified various rules and neighborhood schemes to generate abstract patterns and designs on a 2D grid, based on the `Stepping stone` rule and `Cyclic Cellular Automata` schemes. Additionally, to explore the various possibilities in rule space, genetic crossover and mutation methods are incorporated to generate restricted neighborhood schemes to apply rules on.

Lastly, we tried to leverage the concept of evolutionary algorithms to witness how genetic algorithms GAs can evolve cellular automata CAs to perform computations that require global coordination thus contributing in an effort to search for some interesting Designs and Patterns.

The results from various schemes studied under this project try to illustrate the vast potential of using Cellular Automata in the generation of Abstract patterns and designs, an attribute long perceived to be a standalone quality of humans.

Content

1 Introduction

Abstract Art is defined as ‘the art of painting new structures out of elements that have not been borrowed from the visual sphere, but had been created entirely by the artist...it is a pure art.’

It is an art of ‘expression’ with colors, shapes. Feelings and their expression are a profound trait of humans. But there have been consistent efforts in the past to recreate this sense of abstractness through machines, through computer programs. Following a similar motivation, here we have used ‘Cellular Automata’ to generate abstract designs and patterns, following different rules and neighborhood schemes. We have also used Genetic Algorithms to study how the CAs can coordinate with each other globally, even each of the cell’s states depends on local conditions.

A ‘cellular automaton’ is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells. The rules are then applied iteratively for as many time steps as desired. The concept was originally discovered in the 1940s by Stanislaw Ulam and John Von Neumann.

Here in this project, we have tried to explore the potential of Cellular Automata to generate these abstract designs and patterns at two different levels, first at the level of 1D-Cellular Automata and second at the level of 2D Cellular Automata. Where the section of 1D Cellular Automata mainly focuses on the generation of designs and patterns through the combination of Elementary Cellular Automaton rules. In the second section for 2D Cellular Automata, the 2D cellular matrix is evolved using different rule spaces and neighborhood schemes, to generate abstract designs and patterns.

Alongside generation of art, we have also tried to leverage the concept of evolutionary algorithms to witness how genetic algorithms GAs can evolve cellular automata CAs to perform computations that require global coordination resulting into generation of some interesting Patterns and Designs. We have tried to implement this on the “$\rho_c = ½$” task, which is discussed in the third section of this project.

2 Part 1: 1D Cellular Automata

2.1 Introduction

A one-dimensional cellular automaton consists of a line of sites, with each site carrying a value 0 or 1 (or in general 0, ..., k -1). The value of the site at each position is updated in discrete time steps according to some deterministic rule depending on a neighborhood of sites around it.

Figure 2.1 Illustration of a one-dimensional, binary-state, nearest-neighbor (r=1) cellular automaton with N=11. Both the lattice and the rule table for updating the lattice are illustrated. The lattice configuration is shown at two successive timesteps. The cellular automaton has spatially periodic boundary conditions: the lattice is viewed as a circle, with the leftmost cell being the right neighbor of the rightmost cell, and vice versa.

Elementary cellular automata have two possible values for each cell (0 or 1), and rules that depend only on the nearest neighbor values. As a result, the evolution of an elementary cellular automaton can completely be described by a table specifying the state a given cell will have in the next generation based on the value of the cell to its left, the value of the cell itself, and the value of the cell to its right. Since there are `**2x2x2=8**` possible binary states for the three cells neighboring a given cell, there are a total of `$\mathbf{2^8 = 256}$`elementary cellular automata, each of which can be indexed with an 8-bit binary number (Wolfram 1983, 2002). For example, the table giving the evolution of rule 15 ($15=00001111_2$) is illustrated above.

There has been an extensive study on the results of these elementary rules, in generating one-dimensional sequences that are then stacked to generate Two-Dimensional grid patterns, Wolfram S. (1983); Eric W. et al. Some of these results are shown in the Fig 2.2. In our study, our primary focus is to explore these results further, in their potential to generate Abstract designs and patterns, with similar kinds of strategy and rule-based framework.

Figure 2.2 Results of application of Rule 60, Rule 73, and Rule 129 respectively.

2.2 Algorithms

Stand-alone rules as described in the previous section have been explored a lot as a part of various studies in the past, and most of these results are easily available on various forums and have been discussed in detail. As a part of our study, we would like to search beyond these singleton rule sets with the help of the following algorithms:

I. Multi-Rule Cellular Automata

The key motivation here is to devise an algorithm to extend the CA updation rule to more than one Deterministic rule in a systematic way to generate an Artistic Two Dimensional Image. This way, the generative aspects of more than a single rule can be combined to evolve new designs and patterns.

For this, we propose two temporal based methods, first is **Uniform Rule Application**, and second is **Randomly Sampled Sequence-Based Rule Application**. In Temporal based uniform rule application, we applied different rules as per the timesteps (or iteration step) count, where each rule is applied after uniform intervals. For example, suppose we choose

three rules to apply, namely **A**, **B**, and **C**. The uniform interval rule application can be implemented as, for i^{th} iteration step, if i%3==0, apply rule A, else if i%3==1, apply B and otherwise apply rule C, with an interval of 3 steps. This way, different rules are implemented temporally, allowing the Two Dimensional Image to evolve through application of multiple sets of rules onto One-Dimensional Sequences of 0's and 1's which are then further stacked.

Randomly Sampled Sequence-Based Rule is again temporally based, ie. the application of the rule is dependent on the timestep. However, here the rule application is not uniform.
In Randomly Sampled Sequence-Based Rule, we first allot a unique index to each of the rules. Then we randomly sample a sequence of numbers, **S (**let's say**)** ranging from 0 to N-1, where N is the number of rules being used. The length of this sequence is defined by the number of iterations we want to run the CA updation. Now, this sequence is used to define which rule applies at different timesteps. For i^{th} timesteps, rule with index S[i] is applied onto the 1-D lattice.

Another important aspect here to note is the initialization of the 1-D lattice to start with. For most of our examples and study, we used a policy we call 'center_black', in which all cells are white except the cell at the center of the lattice. We used for most of our examples the lattice size of 257 characters.

2.3 Results

Here are some of the interesting results obtained from the application of the Algorithms discussed above.

I. Multi-Rule Cellular Automata

The results obtained from the application of more than one rule strategy illustrate that by the application of multiple rules, the overall entropy of the lattice can be altered such that to allow the integration of individual rule-based aspects into the generation of an astonishing Two Dimensional Lattices of artistic patterns.

Following are some of the intriguing results, we obtained as result of our search of various combination of rules applied using two different strategies aforementioned in the algorithms section:

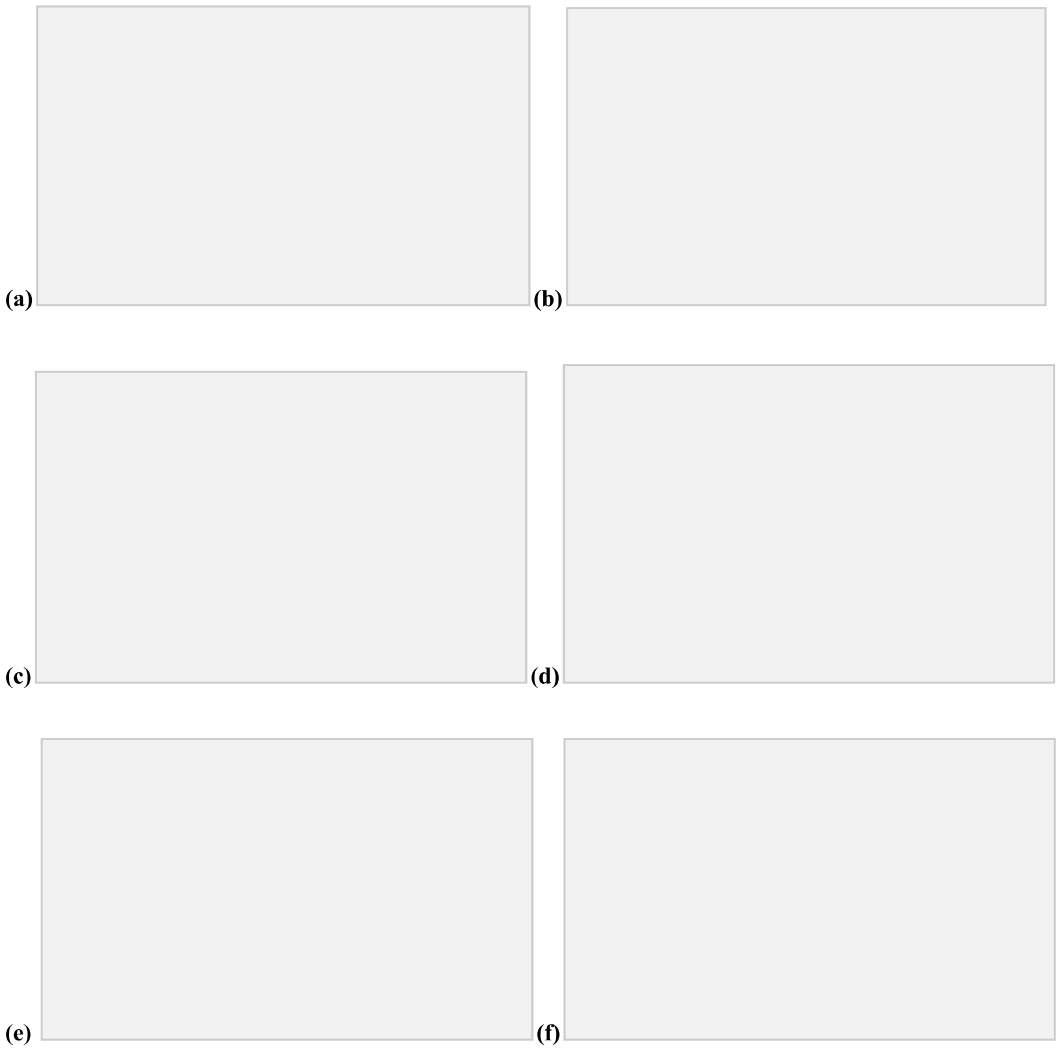

Figure 2.3 Result of application of Temporal based rules with a rule **set size of 2** (ie. two different rules were used for Lattice updation with CA). The first three images use the Uniform Rule application, while the other three use a Randomly Sampled sequence-based Rule application. In fig, (a), (b), and (c) results from a ruleset combination of Rules: (60, 73), (113, 195), and (174,185) respectively. In fig, (d), (e) and (f) results from a ruleset combination of Rules: (160, 164), (27, 56), and (174,185) respectively.

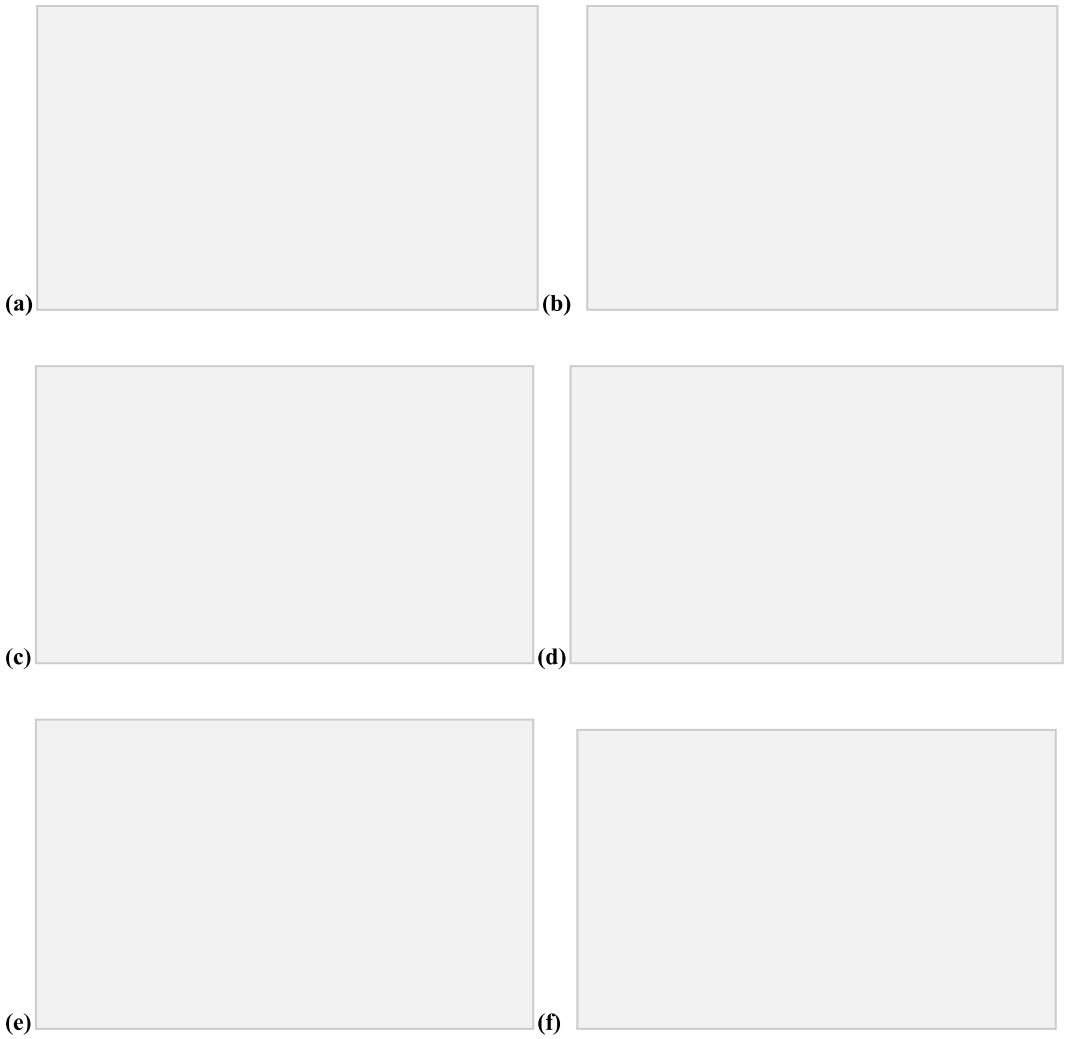

Figure 2.4. Result of temporal based rule application for rule set sizes>2 with Randomly Sampled Sequences. In fig, (a), (b) and (c) are obtained from the ruleset **combination** of **three** ,**five** and **three** rules: (49, 117, 249), (98,120,126,148,181) and (22,45,82) respectively. Fig (d) and (e) are obtained with a rule set of **three** randomly selected rules: (25,176,240) and (23,49,199). (f) is obtained with a rule set of **three** randomly selected rules: (82,92,108).

NOTE: A collection of plots generated can be found *here*.

2.4 Scope of further improvements

In this section so far, we discussed some of the ways which we can use to generate different artistic designs and patterns. We proposed some temporal based methods to evolve patterns out of a set of Multiple rules, which provided some really intriguing results with respect to our motive of generative artwork, thus establishing to some extent the very potential of Generative art using Muti-Rule Algorithms in Cellular Automata. However, there are still a large number of ways that were not included in this exploratory study, which might improve the quality of generative results we obtained so far. These improvements might be in terms of final results obtained or in terms of the search methods. Some of the suggestions for further exploration and improvements could be, the use of spatially (positional) based algorithms for implementation of Multi-Rule based Cellular Automata, the initialization of Lattice using some other policies rather than the `central_black` policy used above (see Figure 2.5, for an example on using randomly initialized lattice instead of `central_black` policy changes the result obtained after certain iterations), to employ CrossOver and Mutation rules to select rules for Updation as will be discussed in next section.

Figure 2.5 The result of the application of Rule 57, Rule 125 and Rule 73 with randomly initialized lattice.

3 Part 2: 2D Cellular Automata

3.1 Introduction

This part presents an exploratory study of two-dimensional cellular automata, along with the definition of various rules and neighborhood schemes to generate abstract designs and patterns.

A cellular automaton consists of a regular lattice of sites. Each site takes on k possible values, and is updated in discrete time steps according to a rule that depends on the value of sites in some neighborhood around it.

a) b)

Figure 3.1 Neighborhoods commonly being used in Two-dimensional Cellular Automata. In the cellular automaton evolution, the value of the center cell is updated according to a rule that depends on the values of the shaded cells. Cellular automata with neighborhood (a) are termed " five-neighbor square"; those with neighborhood (b) are termed "nine-neighbor square." (These neighborhoods are sometimes referred to as the von Neumann and Moore neighborhoods, respectively.)

In every iteration, the current state of each site is updated as per the previous state of the sites that are part of the neighborhood being used. What is being chosen as the neighborhood makes the results vary drastically. At the same time, the initial state of the sites is also an important factor in determining their state in the future. A random initialization versus a fixed predefined lattice, the results in appreciably different results. Lastly, the ‘rules’ to define the next state based on the present states plays a very crucial role in determining the ‘to be’ states in this evolutionary process. Exploring various factors and how they change the results is an important aspect of this exploratory study to generate artistic designs and patterns.

3. 2 Algorithms

There have been a large number of possible ways in which a two-dimensional lattice can be evolved based on rules being used. Hereunder this project, we explored two different Cellular Automaton Algorithms which were both the works of David Griffeath.

I. Stepping Stone Cellular Automaton

This CA rule was first designed and explored by David Griffeath on his Commodore 64 back in the mid-1980s.

The basic rule is you fill an image with random colors and set an update probability. For each pixel see if a random value is less than the probability. If it is then change the pixel to the color of one of its neighbor cells (this can be the 4 N, S, E, W Von-Neumann neighbors or all 8 closest cells in the Moore neighborhood). Griffeath's original uses a 50/50 chance of each cell being changed and uses the Von-Neumann neighborhood.

a) b)

Figure 3.2 Result of applying Stepping Stone Cellular Automata after 600 iterations. a) is the starting images, where random colors are assigned to each site. b) is the result obtained on applying the Stepping Stone updation rule on image, iteratively 600 times. From the result above it is clearly visible, the degree of randomness has reduced over iterations and the image has taken the shape of a splash of colors, which looks artistic with its inherent abstractness.

The above rule is very effective in bringing about artistic touch to the images. In some cases, it also brings a sense of abstractness into the image as we will see some examples ahead.

Algorithm 1 Stepping Stone Cellular Automata rule: Single Iteration

Input: *Image*
Output: *Updated Image*

```
height, width, channels ← size(Image)
for each i ∈ [0, height -1] and j ∈ [0, width-1] do
    r ←rand(0,1)
    if r < 0.5 then
        x ← randint(0,4)
        if x = 1 then
          Image[(i-1) % height, j, :]  ← Image[i, j, :]
        else
          if x=2 then
            Image[i, (j-1) % width, :]  ← Image[i, j, :]
          else
            if x=3 then
              Image[(i+1) % height, j, :]  ← Image[i, j, :]
            else
              Image[i, (j+1) % width, :]  ← Image[i, j, :]
            endif
          endif
        endif
    endif
endfor
```

II. Cyclic Cellular Automata

Cyclic Cellular Automata were first developed by David Griffeath at the University of Wisconsin. In this system, each cell remains unchanged until some neighboring cell has a modular value exactly one unit larger than that of the cell itself, at which point it copies its neighbor's value.

Basic Rules are as follows:

1. Take a 2D grid of cells.
2. Select the maximum number of states each cell can have.
3. Select a threshold value.
4. Fill the cells with random state values between 0 and (maximum states-1).
5. At each step of the simulation, every cell follows these rules;
   a) Count how many neighboring cells (Moore or Von Neumann neighborhoods) surround the cell with a value of the current cell's state + 1.
   b) If the count is greater or equal to the threshold value then the cell state is increased by 1.
6. Repeat this process as long as you want to.

Figure 3.2 "*Medley of Spirals*" Result of applying Cyclic Cellular Automata rule on a randomly generated 2D Grid. The first image is the starting images, where each site is given some random state, and each of these states is mapped onto a randomly generated color palette. The second image is the result after 250 iterations and the Third image is the result after 500 iterations.

For most of our project we used the threshold value to be equal to `one`, ie. if any of the neighboring values is greater than the current cell's state by 1, then increment the values. However, we did perform a lot of experimentation with the Cell Neighborhoods, which will be discussed further. The spirals so obtained are indeed a subject of interest to us as these are the 'abstract' arts generated as an evolutionary consequence of Cyclic Cellular Automaton Rule, out of some randomly sampled two-dimensional grids, which is the main motivation of our exploratory study. Importantly, the whole process of evolution of abstract spiral patterns is in itself an art in process, what we can call 'Moving Art' or 'Evolving Art'.

Algorithm 2 Cyclic Cellular Automata rule: Single Iteration with threshold value 1

Input: *2D grid Canvas*
Threshold_value = 1
Output: *Updated 2D Canvas*

```
newCanvas ←grid(ofsize(Canvas))
height, width ← size(Canvas)
for each i ∈ [0, height -1] and j ∈ [0, width-1] do
    newCanvas [ i, j ] ← canvas[ i, j ]
    nextValue ← canvas[ i, j ] + 1
    if any(Neumann_Neighbors) = nextValue then
        update newCanvas [ i, j] ← nextValue
    endif
endfor
Canvas ← newCanvas
```

III. Cyclic Cellular Automata with Restricted Neighborhoods

For a 3 x 3 neighborhood, there is a total maximum of 9 neighbors (including the cell itself). Where the standard neighborhood schemes like Neumann or Moore take into consideration only some specific neighborhood configurations, but an important point to note is that the Cyclic Automaton rule can be applied to any definition of the neighborhood in any dimension. This gives rise to an idea to explore the results with different sub-neighborhoods of 3 x 3 neighborhoods, so-called 'Restricted Neighborhoods'.

'Restricted Neighborhood' is the term we are using to define the custom-defined neighborhood with respect to a particular cell, represented by **P** followed by the **indexes given to the neighborhood cells taken into consideration** while evaluating for the next value in Cyclic Cellular Automata. For example, if we consider Moore's Neighborhood, we will call it a **P12346789** neighborhood scheme. Similarly, a Neumann Neighborhood would be a **P2468** neighborhood. For better understanding, refer to Fig 3.3.

a) b) c)

Figure 3.3 Definition and example of 'Restricted neighborhoods', term being used to refer to revised restrictions being put onto the standard neighborhood schemes, to define new ones. a) shows the labeling of indexes under the aforementioned strategy. It also depicts the P123456789 neighborhood scheme. b) represents the P234579 neighborhood. c) represents a P15679 neighborhood. These are examples to explain how we will be referencing these neighborhoods afterward.

Once we have defined the terminology, next the question comes, how are we going to explore these neighborhoods? This is an important aspect because here we are talking about exploring various possibilities for the neighborhoods and the corresponding design and patterns they would generate. Just for an instance, if we talk about a 3x3 neighborhood, there are a total of $\mathbf{2^9}$ **(=512)** possibilities, and if we talk about 5x5 neighborhoods then the number of possibilities jumps up to $\mathbf{2^{25}}$!!! which is quite large. However, one must note not all these neighborhoods will be a topic of interest to us. So in order to shell out the degree of randomness in this sample space of neighborhoods, we introduce a technique to evolve neighborhoods using CrossOver and Mutation. The intuition behind the idea is that just like in case of Genetic Algorithms we use to CrossOver and Mutation techniques to explore the sample space to find an optimal solution, in the very same manner, we will use the CrossOver and Mutation techniques to explore our sample space of various possible neighborhoods to find some good neighborhoods that generate some fascinating abstract patterns and Arts that are the main motivation of our project.

We represented an NxN neighborhood as an NxN matrix consisting of zeros and ones. Each element of the matrix is filled with `1` if that neighbor value is to be considered for comparison as per the Cyclic Cellular Automata rule and `0` if that value is not considered. This way, for each possible neighborhood, we have a corresponding 0-1 matrix representation. The rest of the part was the same as we did for the Cyclic Cellular Automata.

In order to generate the neighborhoods incorporating CrossOver and Mutation, we followed the following procedure:

1. First, we randomly generated 5 neighborhood matrices (randomly assigning `0` or `1` at each position) as our first generation.
2. Then we ran 100 iterations of `Cyclic CA updation rule` for each of the matrices and displayed the results along with matrices generating them.
3. Now, to create the next generation, we randomly selected 2 parents from the first generation, performed `**Uniform CrossOver**` between them, with a mutation percentage of **0.1,** and selected one of the two children for the next generation. We followed this procedure 5 times, and we had the second generation ready.
4. Then we repeated the **Step 1, 2,** and **3,** as many times as wanted.

This way, we generated a variety of artistic patterns in the form of spirals following different neighborhood schemes, so-called “MEDLEY OF SPIRALS”.

Figure 3.4 Result of first-generation after 100 iterations of updation for each neighborhood scheme. These are the results obtained for neighborhoods: *P389, P123457, P123679, P347,* and *P23689* respectively. These neighborhood patterns can be further iterated to develop a specific stable form of the pattern as in the case of first and third patterns, because these patterns need more iterations to reach a visually stable state.

As observed above, the aforementioned Multi-Generation Updation technique was run for multiple generations to search for different abstract designs and patterns across our search domain of Neighborhood schemes. Through this process, various intriguing Artistic patterns and designs were obtained as a result of our exploratory study.

3.3 Results

From different algorithms described above, a variety of results were obtained that were of particular interest to us with respect to our objective to explore algorithms that generate abstract designs and patterns. Here are some of the results we obtained out of our exploratory study.

I. Stepping Stone Cellular Automaton: <u>Impressionist Paintings</u>

In all these examples an important aspect of the result obtained is the initial images we are starting with. An important aspect of the art generated using this algorithm is its similarity with that of **Impressionist paintings,** an art style contemporary to the late nineteenth century, which defines drawing as an outcome of brushstrokes of right value and color

Figure 3.5 Result of application of stepping stone rule onto some random clicked pictures taken from google images after 0, 100, and 200 iterations. As in the above figures, iterating the stepping stone rule has the effect of fostering competition between colors, with larger and larger planes of color formed. Also as in a) and b), the idea of abstractness has been introduced into the well-defined shapes and objects of the natural setting. The image so formed, has an increased degree of freedom with respect to color expression.

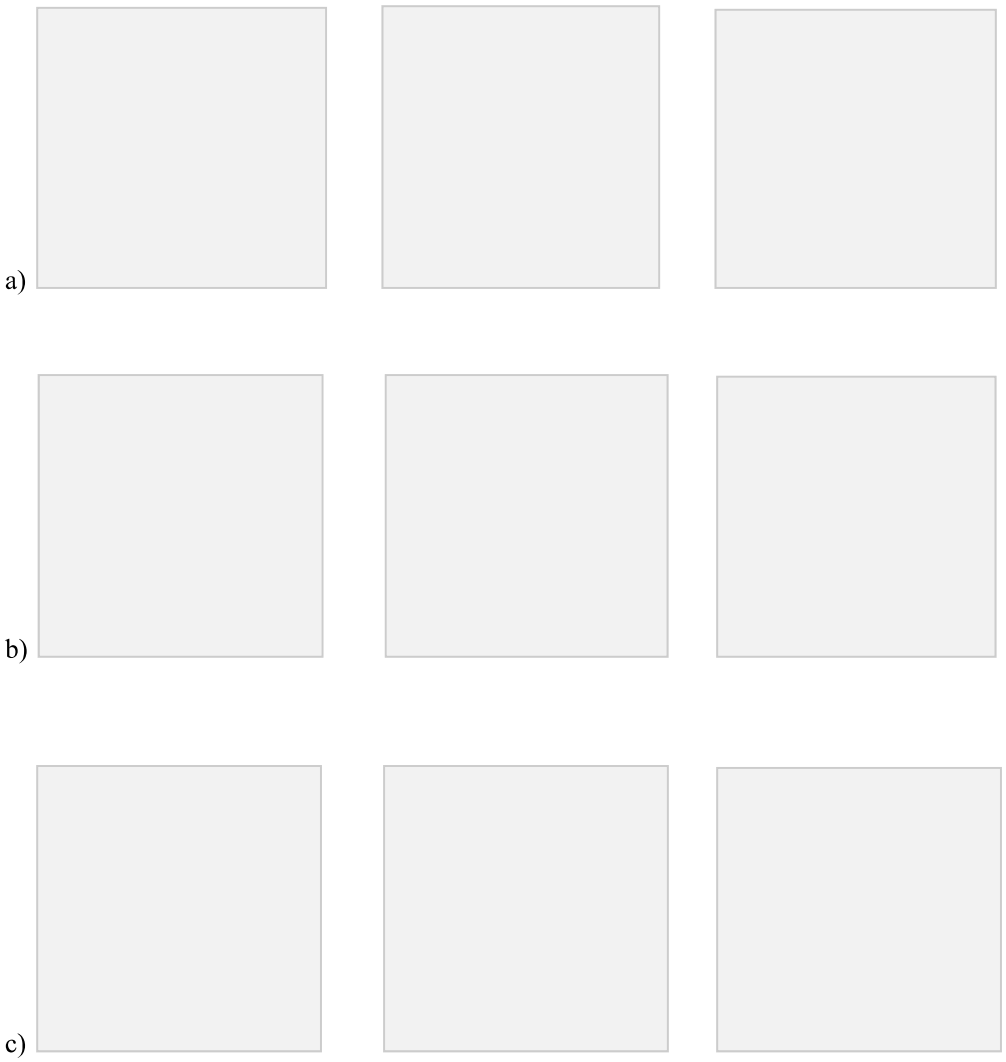

Figure 3.6 Result obtained on application of stepping stone rule onto some simple drawing/paintings take from google. The updated rule successfully inculcates the `painting techniques of Impressionists` into these paintings with the dispersion of colors out of their boundaries, the kind of artwork that generates the abstractness beyond the outlines of mere object-based drawings.

II. Cyclic Cellular Automaton

The results obtained by the application of this updation rule are mostly driven by the type of neighborhood scheme chosen. For most part of our exploratory study, we applied the rule onto a randomly generated color canvas. The key aspect of the results is the **cyclic design and patterns** generated out of mere randomness based on some specific updation rule. These results also include the results obtained using 'Restricted neighborhoods'.

a)

b)

c)

Figure 3.9 Result from the three different generations of randomly sampled 3x3 Neighborhood Schemes. Fig a) is obtained in the first generation on 1000 iterations of the following neighborhoods: *P123457, P234578, P234567, P234569,* and *P12345678* respectively. Fig b) is obtained in second generations on 1000 iterations of the following neighborhoods: *P123457, P23567, P234567, P12345,* and *P134567* respectively. Fig c) is obtained in the third generation for the following neighborhoods: *P123457, P23457, P23567, P1235679,* and *P134679* respectively.