Sweeper's Algorithm and its Application on Image Clustering

[1]Utkaleswar Padhan, [2]Sagar Kumar Tripathy, [3]Sudhakar Sahoo,
[4]Birendra Kumar Nayak and [5]Om Prakash Jena
[1, 2, 5]School of Mathematics, Statistics and Computer Science, Utkal University, Bhubaneswar, India
[3]Institute of Mathematics and Applications, Bhubaneswar, India
[4]P. G. Department of Mathematics, Utkal University, Bhubaneswar, India

ABSTRACT
Using Two Dimensional Hybrid Cellular Automata (2-D HCA) rules image shifting, image copying, zooming in and out, thickening and thinning of an image etc. were possible and reported in [8]. A new searching algorithm called Sweeper's algorithm was proposed on binary images using 2-D HCA and was the basis for solving various problems like migration of initially distributed organisms in a space towards a single point destination, density classification problem of CA etc. [9]. Due to the wide scope of Sweeper's algorithm it was mentioned in [8] that the problem areas that can be solved by this algorithm are text and image compression, informed search in Artificial Intelligence, clustering problem, Cryptography, and pattern classification etc.

Here in this paper we have taken up the challenge and tried to solve the clustering problem using Sweeper's algorithm. Basically our study is based on various color images consisting of the combinations of Red, Green, and Blue (RGB) colors as well as selecting different destination points in the search space. First we have taken red color and one of the two color from blue or green as cluster points and took it as input and by applying Sweeper's algorithm we found two cluster regions one with red color and other with blue or green color. Subsequently we took all the three (RGB) color as input and by applying the algorithm we found three clusters. We took one color between red, blue, green and some other color like magenta, cyan, yellow etc. as input and found different clusters. We have also studied the intersection region of different colors using this algorithm and found interesting color patterns.

Keywords
Cellular Automata, Sweeper's Algorithm, Image Clustering

1. INTRODUCTION
1.1 Cellular Automata
A cellular automaton [3] consists of a regular grid of cells and each cell is in one of a finite number of states. The grid can be any finite number of dimensions. For each cell a set of cells called its neighborhood (usually including the cell itself) is defined relative to the specified cells. An initial state (time t=0) is selected by assigning a state to each cell. A new generation is created (advancing t by 1) according to some fixed rules that determines the new state of each cell in terms of current state of the cell and state of the cells in its neighborhood.

In 2-D Nine Neighborhood CA [1, 2, 4, 6, 8, 9] the next state of a particular cell is affected by the current state of itself and eight cells in its nearest neighborhood also referred as Moore neighborhood.

The central box represents the current cell (i.e. the cell under consideration) and all other boxes represent the eight nearest neighbors of that cell. The number within each box represents the rule number characterizing the dependency of the current cell on that particular neighbor only. Rule 1 characterizes dependency of the central cell on itself alone whereas such dependency only on its top neighbor is characterized by rule 128, and so on. These nine rules are called fundamental rules. In case the cell has dependency on two or more neighboring cells, the rule number will be the arithmetic sum of the numbers of the relevant cells.

Uniform and hybrid CA
If we apply the same rule to each entry of the problem matrix, it is called as Uniform CA and if we apply different rules to different entries at the same time then it is called as Hybrid CA.

Example (Uniform CA)
In the figure, Rule 170 (2 + 8 + 32 + 128) is applied uniformly to each cell of a problem matrix of order (3 x 4) with null boundary condition (extreme cells are connected with logic-0 states).

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow{Rule\ 170} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Example (Hybrid CA)
Let us consider an example of a hybrid CA in null boundary condition where 3 rules (Rule 2, Rule 3, and Rule 4) are applied in 3 different rows (1st, 2nd and 3rd rows respectively) in the above problem matrix.

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \xrightarrow{Rule\,2,3,4} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Table 1: Application of 2-D CA in image processing

Translated image in different Direction	Rules to be applied
Top	8
Bottom	128
Left	2
Right	32

Top-Left	4
Top-Right	16
Bottom-Left	256
Bottom-Right	64

2. SWEEPER'S ALGORITHM: REVIEW

Using 2-D Hybrid CA rules a new searching algorithm has been proposed called Sweeper's algorithm [8] and found to be applicable in many inter-disciplinary research areas. This algorithm is called Sweeper's algorithm because the technique used in this algorithm is very similar to the way a sweeper sweeps haphazardly and put it in one corner of a room. According to this algorithm we will apply different CA rules of Table 1 in different regions by rotating a fixed axis about a fixed point. An important point is that at every iteration of the algorithm, the combinations of all the nine fundamental rules were used.

Sweeper's algorithm has been used to show the different steps of migration of organisms in a particular environment towards a single destination. The organisms can be unicellular or multi-Cellular. In case of unicellular organisms it is considered as the phenomenon of Chemo taxis.

2.1 Simulation Result

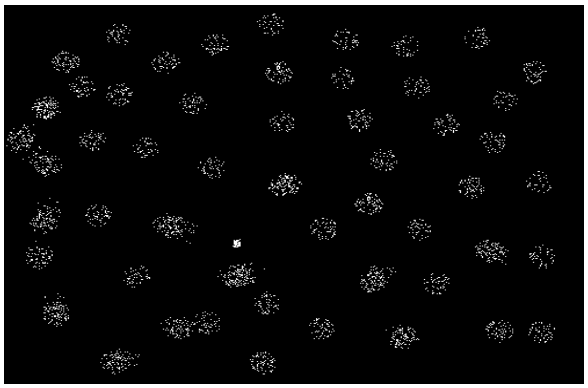

Fig 1: "sys.bmp" shows the initial position of organisms in a 2-D space of size (500X400)

Fig 2: Shows the image "res1.bmp" (After 100 iterations)

3. MODIFIED SWEEPER'S ALGORITHM

In the modified Sweeper's algorithm we have taken the number of iteration t and at a stretch two complementary rules are used for t number of times along a particular axis of rotation. In this way all the nine rules are applied to an image one after another with respect to horizontal axis, left-diagonal axis (\), vertical axis and right-diagonal axis (/). Here we have shown how the result of this algorithm is different from the original Sweeper's algorithm.

3.1 Modified Algorithm

Step1: A Monochromatic ".bmp" file is taken as the input where t is the number of iteration and the size of the matrix is (X x Y), the point of destination = (x, y), the angle of rotation = 45 (in degree), the number of rotations in one iteration, N is [360/2*45] = 4.

Step2: Input number of time: t

for r←1 to t

{

for (i = 1 to x & j = 1 to Y) // Two CA rules are applied with respect to Horizontal Axis.

Apply Rule 128

for (i = x+1 to X & j = 1 to Y)

Apply Rule 8

}

for r←1 to t

{

For (i-j <= x-y) // Two CA rules are applied with respect to Diagonal Axis (\ )

Apply Rule 16

for (i-j > x-y)

Apply Rule 256

}

for r←1 to t

{

for (i = 1 to X & j = 1 to y) // Two CA rules are applied with respect to Vertical Axis

Apply Rule 32

for (i = 1 to X & j = y +1 to Y)

Apply Rule 2

}

for r←1 to t

{

for (i+j <= x+y) // Two CA rules are applied with respect to Diagonal Axis (/)

Apply Rule 64

for (i+j > x+y)

Apply Rule 4

}

Step3:- Store the resultant matrix in "res.bmp" as the output.

3.2 Simulation Results

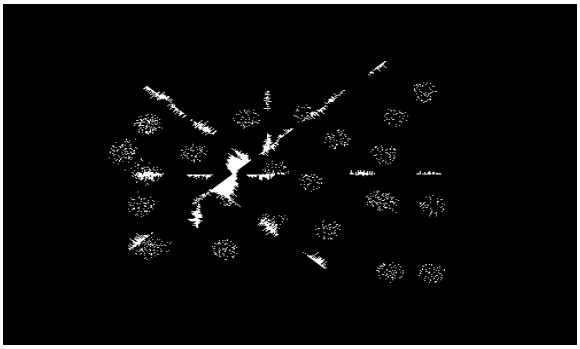

Fig 3: Shows the image "res2.bmp" (After 30 iterations)

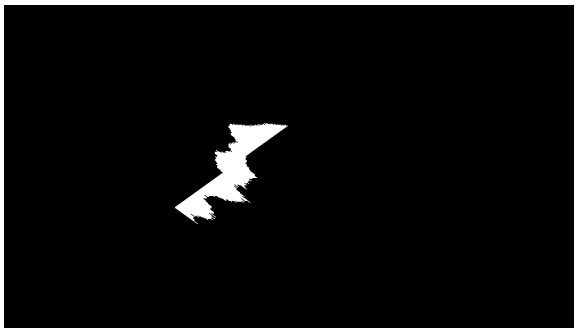

Fig 4: Shows the image "res3.bmp" (After 100 iterations)

3.3 Sweeper's algorithm for multiple destinations

Fig 5 and Fig 6 shows the simulation result of sweeper's algorithm applied to Fig 1 where two points are chosen as destination instead of one.

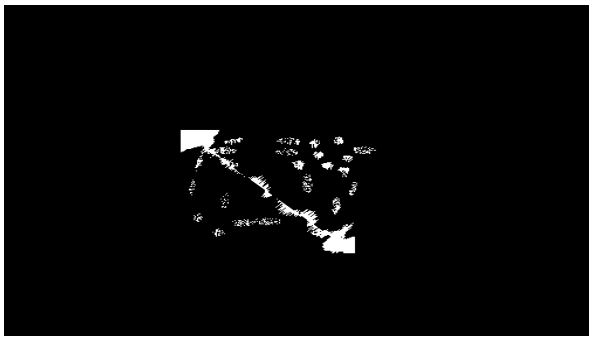

Fig 5: Shows the image "res4.bmp" (After 50 iterations)

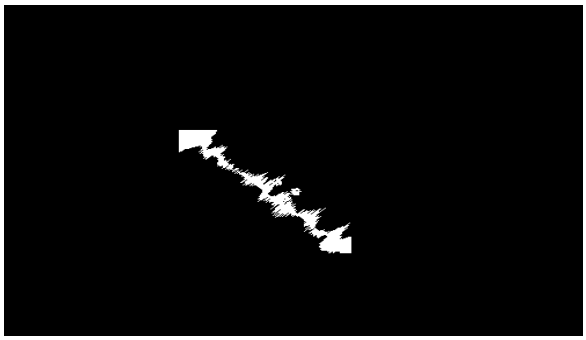

Fig 6: Shows the image "res5.bmp" (After 100 iterations)

4. SWEEPER'S ALGORITHM FOR COLOR IMAGE

This section shows the implementation of Sweeper's algorithm for color images and this is used for image clustering. Basic information for color images which will be used for implementation purpose is discussed below.

4.1 True color RGB

A true color red-green-blue(RGB) image is represented as three-dimensional (M X N X 3) matrices. Each pixel has red, green, blue components along with the third dimension with values in the interval [0, 1]. For example the color components of pixel(m, n) are MyImage(m, n, 1)=red, MyImage(m, n, 2)=green, MyImage(m, n, 3)=blue.

If each of these components has a range 0 to 255 such an image is a "stack" of three matrices; representing the red, green and blue values for each pixel. This means that for every pixel there correspond to 3 values. True color image may be of class Integer or Double.

4.2 RGB Components of an image

MATLAB has the ability to find out exactly how much Red, Green, and Blue content is there in an image. One can find this by selecting only one color at a time and viewing the image in that color. The following code allows viewing the Red content of an image.

redimage = Myimage; % Create a new matrix equal to the matrix of the original image.

redimage (:, :, 2) = 0;

redimage (:, :, 3) = 0;

Since the colors are mapped using RGB, or columns with values of Red, Green and Blue, so we can selectively nullify the green and blue columns to obtain only the red part of the image.

imshow(redimage); % show the red image that was just created. Similarly, we can do the same with the green and blue components of the image. Just keep in mind the format of the array

Red : Green : Blue

1 2 3

Table 2: RGB values of different colors

RGB Value	Short Name	Long Name
[1 1 0]	Y	Yellow
[1 0 1]	M	Magenta
[0 1 1]	C	Cyan
[1 0 0]	R	Red
[0 1 0]	G	Green
[0 0 1]	B	Blue
[1 1 1]	W	White
[0 0 0]	K	Black

4.3 Image clustering using Sweeper's algorithm

A *cluster* is a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called **cluster**) are more similar to each other than to those in other groups (clusters). So Clustering can be considered as the most important *unsupervised learning* problem.

For us the clustering means to arrange similar categories of items and put them into their respective places. The solution of this type of clustering has many applications and can simulate how a ROBOT will automatically arrange a list of Books in a library? Or, can be a model from chaotic

phenomena to a systematic arrangement or simulation for separating iron, gold or other metals from their corresponding metal ores etc. Here image clustering problem has been solved by the help of Sweeper's algorithm. The Image which is used here is True color (RGB) image.

4.3.1 Image clustering Algorithm

Step 1: A ".png" file having two color cluster points is taken as input. Here the input file is named as "c30.png". The size of the corresponding image file is (X x Y). The point of destination is (x x y) where, *t* is the number of iteration. Here the size of the input image is (500,400).

Step 2: The matrix of the file "c30.png" can be obtained by disp( ) function of MATLAB.

Step 3: The matrix is divided into two matrices.

Step4: Corresponding images of the two matrixes can be displayed using imshow( ) function. Let the two individual images are "c31.png" and "c32.png". "c31.png" contains red color clusters and "c32.png" contains blue color clusters.

Step 5: Apply Sweeper's algorithm to "c31.png"file

Do the following for *t* times

{

for (i = 1 to x & j = 1 to Y) //Two CA rules are applied with respect to Horizontal Axis

Apply Rule 128

for (i = x+1 to X & j = 1 to Y)

Apply Rule 8

for (i-j <= x-y) // Two CA rules are applied with respect to Diagonal Axis (\ )

Apply Rule 16

for (i-j > x-y)

Apply Rule 256

for (i = 1 to X & j = 1 to y) // Two CA rules are applied with respect to Vertical Axis

Apply Rule 32

for (i = 1 to X & j = y +1 to Y)

Apply Rule 2

for (i+j <= x+y) // Two CA rules are applied with respect to Diagonal Axis (/)

Apply Rule 64

for (i+j > x+y)

Apply Rule 4

}

Step 6: Store the resultant matrix in "res1.png"

Step 7: Apply Sweeper's algorithm to "c32.png"file.

Do the following for t times

{

for (i = 1 to x & j = 1 to Y) // Two CA rules are applied with respect to Horizontal Axis

Apply Rule 128

for (i = x+1 to X & j = 1 to Y)

Apply Rule 8

for (i-j <= x-y) //Two CA rules are applied with respect to Diagonal Axis (\ )

Apply Rule 16

for (i-j > x-y)

Apply Rule 256

for (i = 1 to X & j = 1 to y) // Two CA rules are applied with respect to Vertical Axis

Apply Rule 32

for (i = 1 to X & j = y +1 to Y)

Apply Rule 2

for (i+j <= x+y) // Two CA rules are applied with respect to Diagonal Axis (/)

Apply Rule 64

for (i+j > x+y)

Apply Rule 4

}

Step 8: Store the resultant matrix in "res2.png" as the output.

Step 9: Finally "res1.png" and "res2.png" can be combined in one window using either cat command in MATLAB or simply adding the two images.

Step 10: After combining "finalres.png" is the final output image having two different color cluster regions.

4.3.2 Procedure to draw color cluster points using MATLAB and its output image

First thing we have to generate different color cluster points using MATLAB. Here we have generated cluster points having two colors i.e. one red color cluster points and other is blue color cluster points using plot function of MATLAB. We can also use scatter function.

The MATLAB code to draw color cluster points using plot function is given below.

```
n = 100;
cluster1 = randn([n,4]); % 100 2-D coordinates
cluster2 = randn([n,4]); % 100 2-D coordinates
hold on;
plot(cluster1(:,1),cluster1(:,2),'ro','MarkerSize',10,'MarkerFaceColor','r'); %plotting cluster 1 pts
plot(cluster2(:,1),cluster2(:,2),'bo','MarkerSize',10,'MarkerFaceColor','b'); %plotting cluster 2 pts
```

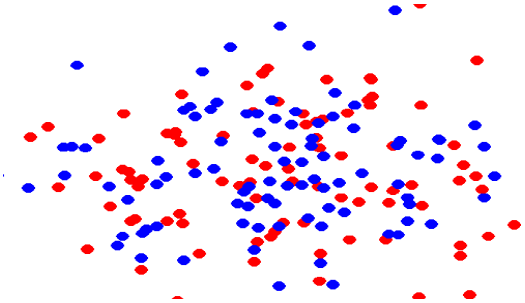

Fig7: Output image obtained from the above code

After changing the background to black using paint the above image is again shown in Fig 8.

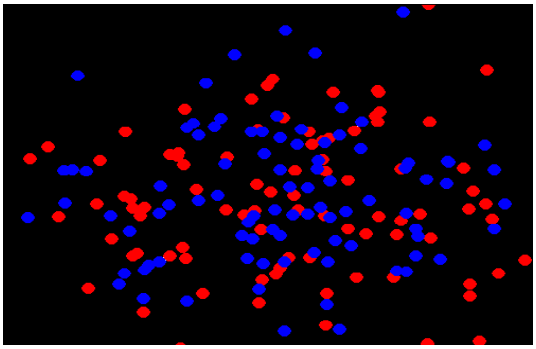

Fig 8: "c30.png" an image containing red and blue color cluster points

This image will be taken as input to implement the image clustering algorithm. The size of the image is taken as (500x400).

4.3.3 Result after applying Sweeper's algorithm to two individual images

After we get two individual images one containing red color cluster points and another containing blue color cluster points we can apply Sweeper's algorithm to the two individual images. We can vary the number of iterations from 1 to 100 and check the result after certain iterations. We have noticed that after 100 iterations we got two different cluster regions in two different images are shown below.

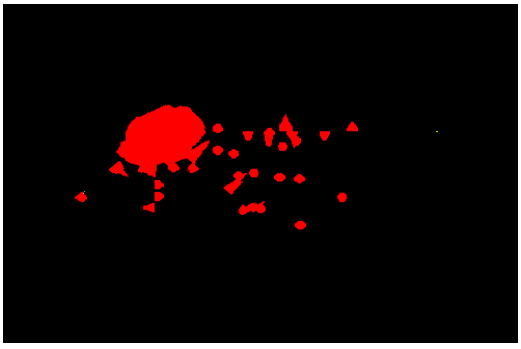

Fig 9: Shows the image containing red cluster points after 50 iterations

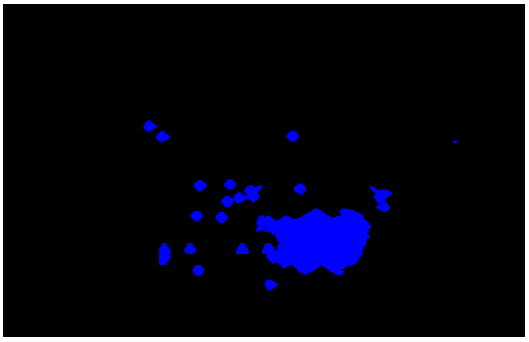

Fig 10: Shows the image containing blue cluster points after 50 iterations

Fig 11: Shows the red cluster image "res1.png" after 100 iterations and the destination point at (150, 150)

Fig 12: Shows the blue cluster image "res2.png" after 100 iterations and the destination point at (300, 300)

4.3.4 MATLAB code to combine two different images in a single window and its output result

After getting two color cluster regions i.e. red color and blue color in two different figure or window, we can combine them in a single window using built in MATLAB function or by simple addition of two images. The code is given below.

```
a=imread('res1.png','png');
b=imread('res2.png','png');
finalimage =cat(2,a,b);
or we can write
a=imread('res1.png','png');
b=imread('res2.png','png');
imshow(a+b)
```

Also we can write subimage(a+b) in place of imshow(a+b).

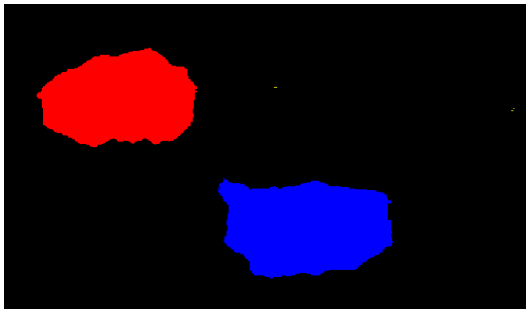

Fig 13: image "finalres.png" After combining "res1.png" and "res2.png" in a single window.

Fig 14: image "finalrb.png" shows the overlap of red and blue color cluster image

Here we have studied using computer program that the combination of red and blue color gives magenta Color. For red color cluster the destination is at (150,150) and for blue color cluster the destination is at (200,200).

4.3.5 Input and Output Result for other types of images

In the above algorithm we had considered only red and blue color points and after applying Sweeper's algorithm we got two different color cluster regions. Now we have considered color combinations other than red and blue and applied Sweeper's algorithm and different results are obtained.

A) Input image and output result to get the magenta and green color cluster in one window

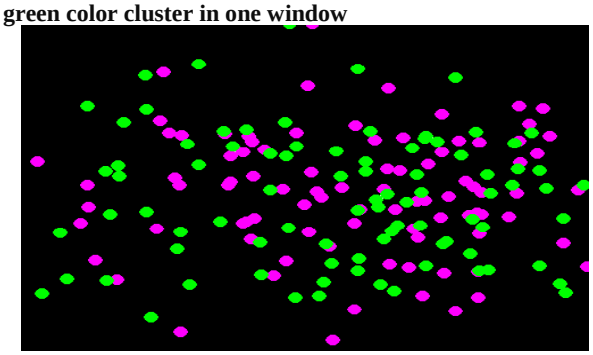

Fig 15: Shows the original image "m10.png"

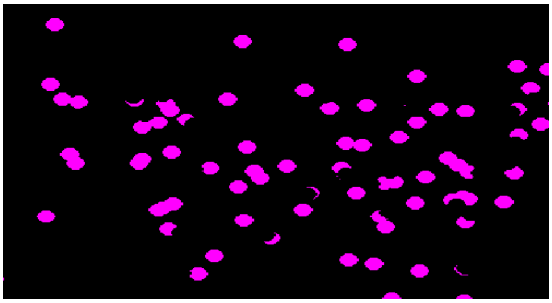

Fig 16: image "m11.png" containing magenta color cluster points

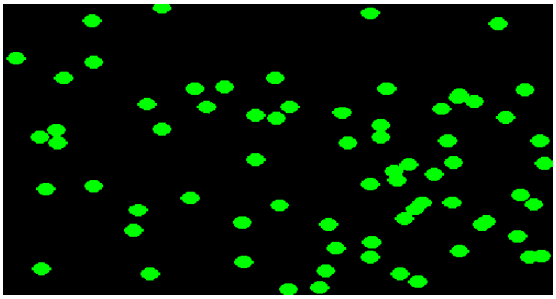

Fig 17: image "m12.png" containing green color cluster points

Fig 18: shows the green color cluster image "mgres1.png" after 100 iterations and the destination is at (150,150)

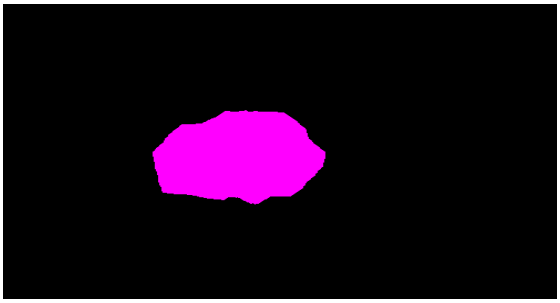

Fig 19: shows the magenta color cluster image "mgres2.png" after 100 iterations and the destination is at (200,200).

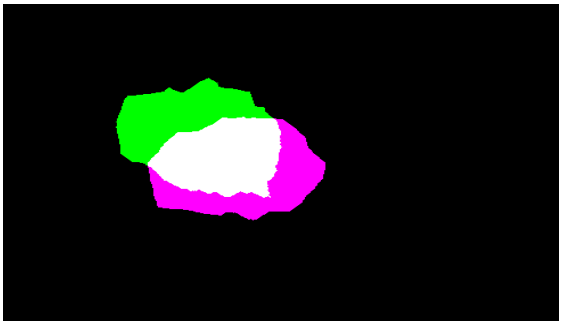

Fig 20: image "finalgm.png" shows the overlap of green and magenta color cluster image

The computer simulation shows that the combination of green and magenta color gives white Color therefore it also mimics the natural color phenomena. For green color cluster the destination is at (150,150) and for magenta color cluster the destination is at (200,200).

B) Input image and output result to get the combination of red, blue and green color cluster in one window

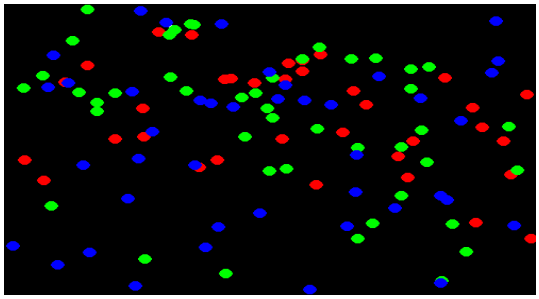

Fig 21: Shows the original input image "rgb10.png"

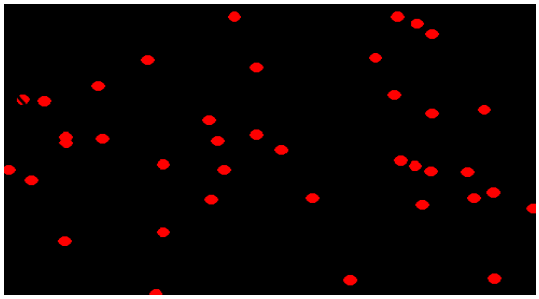

Fig 22: Shows image "rgb11.png" containing red color cluster points

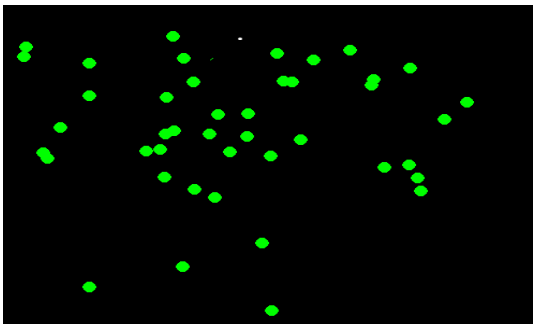

Fig 23: image "rgb12.png" containing green color cluster points

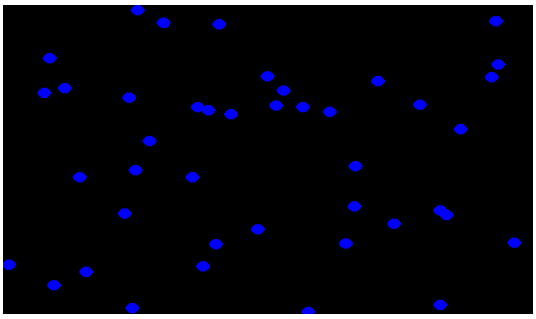

Fig 24: image "rgb13.png" containing blue color cluster points

Fig 25: shows the red color cluster image "rgbres1.png" after 100 iterations and the destination is at (170,180).

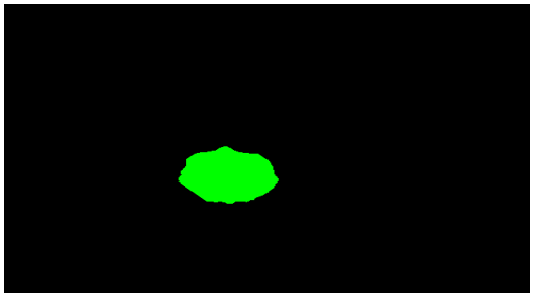

Fig 26: Shows the green color cluster image "rgbres2.png" after 100 iterations and the destination is at (200,180).

Fig 27: shows the blue color cluster image "rgbres3.png" after 100 iterations and the destination is at (210,160).

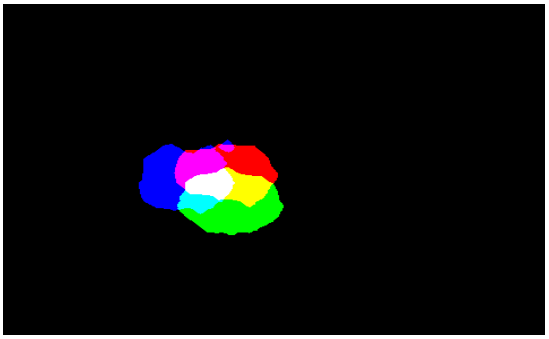

Fig 28: image "finalrgb.png" shows the combination of red, green and blue color

The intersection point of red, blue and green color is white. For red color the destination is at (170,180).For green color the destination is at (200,180).For blue color the destination is at (210,160).

So by taking different color combination we got a different color by the intersection of two particular colors. In this way we can study set theory by assigning a set to a particular color and study their intersection region.

5. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

We have studied broadly the Sweeper's algorithm which was initially designed and developed by the authors in [8]. At first we have implemented the existing Sweeper's algorithm for binary images. Later on we have modified the internal architecture of the Sweeper's algorithm by changing the original iterative procedure and implemented both the algorithms (original and modified versions), choosing multiple destination points in the search space and found different clusters both in binary as well as in color images. If destination points are close to each other then we found that the intersection region of two different color clusters giving a new color. For example intersection of red and blue is giving magenta, similarly intersection of red and green is giving

yellow, and intersection of blue and green giving cyan color. Also we take one of the colors from RGB and another color like yellow or cyan or magenta and observe that combination of blue with yellow, red with cyan and green with magenta gives white color.

Due to its wide scope we feel that the work can be extended for any complicated images containing RGB component and various patterns can also be generated using Sweeper's algorithm with little modification. Except image clustering, this algorithm spans many other inter-disciplinary research areas and can be studied extensively like informed search in artificial intelligence, text and image compression [2], cryptography, morphology [5], pattern classification [7] etc. which is our immediate future research goal.

6. REFERENCES

[1] A. R. Khan, P.P. Choudhury, K. Dihidar, S. Mitra and P. Sarkar, VLSI Architecture of Cellular Automata Machine, *Computers Math. Applic.* Vol. 33, No. 5, 79-94, (1997).

[2] A. R. Khan, P. P. Choudhury, K. Dihidar and R. Verma, Text compression using two-dimensional cellular automata, *Computers and Mathematics with Applications* 37 (1999), 115-127.

[3] J. von. Neumann, The Theory of Self- Reproducing Automata, (Edited by A.W. Burks) *Univ. of Illinois Press Urbana* (1996).

[4] K. Dihidar, P. P. Choudhury, Matrix Algebraic formulae concerning some special rules of two-dimensional Cellular Automata, *International journal on Information Sciences, Elsevier publication*, Vol. 165, 91-101, (2004).

[5] Ikenaga, T. Ogura, T., Real-time morphology processing using highly parallel 2-D cellular automata CAM/sup 2/, *IEEE Transactions on Image Processing*, Vol. 9, Issue: 12, pp 2018- 2026, (2000).

[6] P. Chattopadhyay, P. P. Choudhury, Characterisation of a Particular Hybrid Transformation of Two-Dimensional Cellular Automata, *Computers and Mathematics with Applications*, Vol. 38, 207-216, (1999).

[7] P. Maji, C. Shaw, N. Ganguli, B. K. Sikdar, and P. Pal Chaudhuri. Theory and Application of Cellular Automata For pattern Classification, *Fundamenta Informaticae 58, IOS Press,* pp 321-354 (2003).

[8] P. Pal Choudhury, B. K. Nayak, S. Sahoo, S. P. Rath, Theory and applications of Two-dimensional, Null-boundary, Nine-Neighborhood, Cellular Automata Linear rules, CoRR abs/0804.2346 (2008).

[9] S. Sahoo, P. Pal Choudhury, A. Pal, B. K. Nayak, Solutions on 1-D and 2-D Density Classification Problem Using Programmable Cellular Automata. *J. Cellular Automata 9(1):* 59-88 (2014).