Computational application to circumference random parametrization: an initial approach to generate segmented nanoparticles like images

Héctor Daniel Molina Ruiz, Obed Pérez Cortés, Heberto Gómez Posos

Universidad Autónoma del Estado de Hidalgo, Área Académica de Computación y Electrónica, Mineral de la Reforma, Hidalgo, México

Abstract

Present document discusses the generation of images in image processing context, specifically focusing on creating images that approaches to realistic ones; while actual data is lacking or standard data augmentation techniques are not feasible. The complexity of real images and shapes is acknowledged, but regular geometric forms are used as a starting point to generate more complex desired shapes. The context involves segmented images obtained from scanning electron microscope (SEM) images of thin films, where a lack of available information hinders the use of conventional data augmentation techniques. To address this limitation, the document proposes an approach that involves the use of random criteria on parametric equations for circumferences to generate shapes resembling nanoparticles found in SEM images of ZnO thin films. Python V3 is used in the research to draw circles on a canvas, creating visually similar segmented SEM images as a first step in making (drawing) visually comparable segmented SEM images available or mimic segmented SEM images of thin films.

Resumen

El presente documento aborda la generación de imágenes artificiales en el contexto de procesamiento digital de imágenes, centrándose específicamente en la creación de imágenes que se asemejen a aquellas reales, cuando falta información o las técnicas estándar de aumento de datos no son factibles. Se reconoce la complejidad de las imágenes y formas reales, sin embargo, las formas geométricas regulares se pueden utilizar como punto de partida para generar formas más complejas y que puedan ser similares a las deseadas. Para imágenes segmentadas obtenidas a partir de imágenes provenientes de la técnica de microscopía electrónica de barrido (SEM) en películas delgadas, donde la falta de información disponible dificulta el uso de técnicas convencionales de aumento de datos, el documento propone un enfoque que implica el uso de criterios aleatorios en ecuaciones paramétricas para circunferencias con el fin de generar formas que se asemejen a nanopartículas encontradas en imágenes SEM (ya segmentadas) de películas delgadas de ZnO. Se utiliza Python V3 en la investigación para dibujar círculos en un lienzo, creando imágenes segmentadas SEM visualmente similares como primer paso para disponer de imágenes similares a aquellas imágenes SEM segmentadas, provenientes de películas delgadas.

Keywords: Image generation, Parametric aquations, Regular shapes, Scanning electron microscope, Segmented SEM images
Palabras Clave: Ecuaciones paramétricas, Formas geométricas regulares, Generación de imágenes, Imágenes SEM segmentadas, Microscopía electrónica de barrido

1. INTRODUCTION

Data augmentation is a technique which helps on different computational processes. However, when data bases content availability shows a lack, it become necessary to complete available data with similar unreal images (also called virtual images or synthetic images), to help some proper processing and information generation context. About that, unreal images o similar to realistic ones, can help the data accomplishment, when there exists a lack on the information availability and it is not possible to use the known data augmentation techniques.

As a matter of fact, real images and real images' shapes are more complex than those regular forms that geometry has defines along its history, however, those regular shapes conform the first approach to generate more complex desired shapes.

When talked about segmented images from thin films´ scanning electron microscope images (SEM images), there exists a clear lack of available information, which makes it not feasible to use well known data augmentation techniques in the image processing context. Due to that lack, it is necessary to figured out alternative approaches to cover blank on the computational field.

Present document integrates a method to generate segmented nanoparticles like images, taking in to consideration a computational application to deform the well-known circumference´s parametric equation, to generate similar shapes that approaches those nanoparticles that can be found in a scanning electron microscope´s segmented image, specifically for those segmented SEM images that belong to ZnO thin films (Zinc Oxide thin films).

2. THEORETICAL FRAMEWORK AND METHODOLOGY

The scanning electron microscope (SEM) is one of the most popular and user-friendly imaging tools that reveal the surface topography of a sample (Zhu & Inada, 2012). SEM is based on a focused beam of electrons that scan the sample, which interacts with the atoms in the sample to provide three-dimensional surface topography (Pallares-Rusiñol et al., 2023) in a two dissensions image (2D) or representation.

As SEM images provides a topography representation of deposited substance on the substrate, it can be identified zones or region which can provide information on the analysis and characterization of SEM images. Image processing segmentation is one of the techniques to characterize or generate information from a 2D representation (an image). Particularly in the case of SEM images, there exists a marked lack of images availability.

Generating new image from another is one of the challenging tasks in computer vision. As stated in Elasri, Elharrouss, Al-Maadeed & Tairi (2022), images generation can be done using different form as input including RGB images, videos, medical images, and text, etc., while the output, in general, is an image or a video and the type of dataset used and the target data expected as output can be a 2D image or also a 3D video. The task gets more complex and increases its computational requirements when there exist a lack of information or data bases, which can offer the needed images to perform an image generation process. Since that lack in the availability of information cannot be taken as a justification to do not perform an image generation task, it is necessary to develop alternative tools to overcome the lack (of information).

Since the parametric equations are able to characterize observed shapes (Xu et al., 2019), in some images, it is feasible to use them (parametric equations), for example, to generate similar shapes found in segmented images, overcoming a possible lack of information. The parametric equations allow a computationally efficient and an easily implementation (Pérez-Rastelli, Godoy, Villagran & Onieva, 2013), particularly when it is talked about image generation, due to the reduction in two scopes: complexity and processing time; compared with other method like generative adversarial networks, deepfake, among others.

According to Wang & Boyer (2012) to draw a circle, it can be used the parametric equations (Eq. 1):

$$\begin{cases} x = x_c + r * cos\theta \\ y = y_c + r * sin\theta \end{cases}, \quad . . . . (1)$$

Whare (x_c, Y_c) is the circle´s center, r is the radius and θ vary among $[0,2\pi)$. Last equations (Eq. 1) give the opportunity to draw a circle or circles on a canvas, due to the parametrization from the circle contours which allow the drawing (Figure 1).

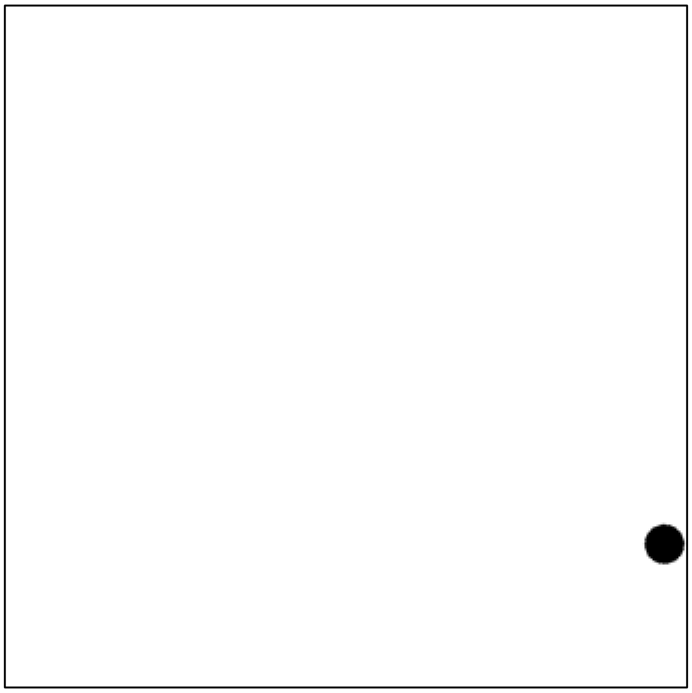

Figure 1. Circumference traced by parametric equations, with fulfillment, on a canvas. Source: Self-made with Colab® by Google® notebook based on Python V3

In last figure they were drew 1 circle thorough parametric equations tracing a lines´ series, with defined circle's center and initial radius. In the following pseudo algorithm, it is shown the drawing process (Algorithm 1).

Algorithm 1. Drawing of a circle through parametric equations

```
1:  procedure drawing_circles(lines, angle, canvas, number of circles, radius, circles center, circumference)
2:      importing libraries (numpy
                             matplotlib.pyplot
                             cv2)
3:        defining a number of lines to draw the circumference´s perimeter
4:          defining the angle of drawing coverage from 0 to $2\pi$, according to the previous number of lines
5:        generating a canvas to draw the circles (final canvas of 768 x 768 pixels)
6:        defining the number of circles to be draw on the canvas
7:        defining a value for initial radius
8:        call function Trace_circle
9:            def Trace_circle():
10:               defining initial center circle´s coordinates´
11:               defining coordinates from lines tracing along the circle´s circunmference
12:               coincide last circumference´s coordinates with initial circumference´s coordinates
13:               drawing and fulfilling circumference
14:               save generated image
15:         end Trace_circle
```

Once it is though about a regular shape, it can be said, that initial shape can be replicated on the canvas, which means more than a single regular shape (in this case a circle) can be draw in it (in the canvas). Using parametric equations (Eq. 1), it can be drawn a different number of circles on a canvas (Figure 2).

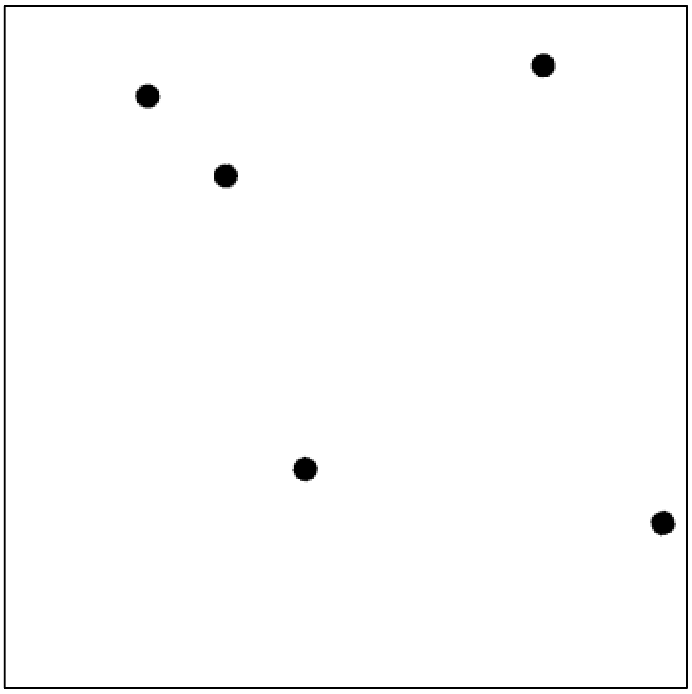

Figure 2. Circumferences traced by parametric equations, with fulfillment, on a canvas. Source: Self-made with Colab® by Google® notebook based on Python V3

In last figure they were drew 5 circles thorough parametric equations tracing a line´ series, with random circle's center and initial random radius. In the following pseudo algorithm, it is shown the drawing process (Algorithm 1).

Algorithm 2. Drawing of 5 circles through parametric equations

```
1:  procedure drawing_circles(lines, angle, canvas, number of circles, radius, circles center, circumference)
2:      importing libraries (numpy
                             matplotlib.pyplot
                             cv2)
3:        defining a number of lines to draw the circumference´s perimeter
4:          defining the angle of drawing coverage from 0 to $2\pi$, according to the previous number of lines
5:        generating a canvas to draw the circles (final canvas of 768 x 768 pixels)
6:        defining the number of circles to be draw on the canvas
7:        defining a value for initial radius
8:          for i, number of circles do
9:            call function Trace_circle
10:         end for
11:           def Trace_circle():
12:               defining center circle´s coordinates´
13:               defining coordinates from lines tracing along the circle´s circumference
14:               coincide last circumference´s coordinates with initial circumference´s coordinates
15:               drawing and fulfilling circumference
16:               save generated image
17:         end Trace_circle
18        merging generated circles
```

When it considered aleatory components, it can be enriched the scope of drawing regular shapes or fulfilled contours thorough parametric equations, by randomization of drawing criterions like lines, number of circles, radius, circles center, among others. Aleatory or random numbers, better known as pseudo-aleatory numbers or pseudorandom numbers have different applications on the computer context. One of the main characteristics of a pseudo random number it the need of a seed to start its calculation.

There exist a variety of random number generators, that can be used for computational purposes (Table 1). According to James (1988), among the desirable properties of a they can be mentioned: good distribution, long period, repeatability, long disjoint subsequences, portability and efficiency.

Table 1. List of some random number generators since 1946 to 2020. Source: Self-made bases on literature review

Generator	Year	Author	Hints on Academic Google	Latest approach or mention published
Middle-square method	*1946*	*Neumann*	*6*	*2022*
Lehmer generator	*1951*	*Lehmer*	*2*	*1996*
Linear congruential generator (LCG)	*1958*	*Thomson & Rotenberg*	*172*	*2023*
Lagged Fibonacci generator (LFG)	*1958*	*Mitchell & Moore*	*11*	*2016*
Linear-feedback shift register (LFSR)	*1965*	*Tausworthe*	*329*	*2023*
Wichmann–Hill generator	*1982*	*Wichmann & Hill*	*1*	*1993*
Inversive congruential generator (ICG)	*1986*	*Eichenauer & Lehn*	*35*	*2021*
Blum Blum Shub	*1986*	*Blum, Blum & Shub*	*88*	*2023*
Park-Miller generator	*1988*	*Park & Miller*	*6*	*2023*
ACORN generator	*1989*	*Wikramaratna*	*4*	*2023*
Subtract-with-borrow (SWB)	*1991*	*Marsaglia & Zaman*	*6*	*2017*
Maximally periodic reciprocals	*1992*	*Matthews*	*1*	*1992*
KISS	*1993*	*Marsaglia*	*1*	*2023*
Multiply-with-carry (MWC)	*1994*	*Marsaglia & Koç*	*8*	*2023*
Complementary-multiply-with-carry (CMWC)	*1997*	*Couture & L'Ecuyer*	*1*	*2018*
Mersenne Twister (MT)	*1998*	*Matsumoto & Nishimura*	*14200*	*2023*
Xorshift	*2003*	*Marsaglia*	*23*	*2022*
Well equidistributed long-period linear (WELL)	*2006*	*Panneton, L'Ecuyer & Matsumoto*	*3*	*2020*
A small noncryptographic PRNG (JSF)	*2007*	*Jenkins*	*4*	*2021*
Advanced Randomization System (ARS)	*2011*	*Salmon, Moraes, Dror & Shaw*	*6*	*2021*
Threefry	*2011*	*Salmon, Moraes, Dror & Shaw*	*1*	*2020*
Philox	*2011*	*Salmon, Moraes, Dror & Shaw*	*1*	*2020*
WELLDOC	*2013*	*Balkova, Bucci, de Luca, Hladky & Puzynina*	*1*	*2015*
SplitMix	*2014*	*Steele, Lea & Flood*	*25*	*2023*

Permuted Congruential Generator (PCG)	*2014*	*O'Neill*	*1*	*2020*
Random Cycle Bit Generator (RCB)	*2016*	*Cookman*	*1*	*2021*
Middle-Square Weyl Sequence RNG	*2017*	*Widynski*	*1*	*2017*
Xoroshiro128+	*2018*	*Blackman &Vigna*	*1*	*2019*
64-bit MELG (MELG-64)	*2018*	*Harase & Kimoto*	*2*	*2022*
Squares RNG	*2020*	*Widynski*	*1*	*2020*

One of the well-known random number generators is the Mersenne Twister, published by Matsumoto and Nishimura in 1998 (Table 2). Mersenne Twister (MT) algorithm provides a super astronomical period of [(2**19937)-1] and 624-dimensional equidistribution up to 32-bit accuracy, while using a working area of only 624 words (Matsumoto & Nishimura, 1998).

Table 2. CPU time for 10**7 generations and working area among random number generators selected by the author. Source: Adapted from Matsumoto & Nishimura (1998)

Dimension	MT19937	TT800	Taus88	Rand	Rand_array	KISS	COMBO
CPU-Time [s]	*10.18*	*9.97*	*7.95*	*9.64*	*23.23*	*9.24*	*11.14*
Working area [words]	*624*	*25*	*3*	*1*	*1000*	*5*	*4*
Period	*(2**19937)-1*	*[(2**800)-1*	*~2**88*	*~2**31*	*~2**129*	*~2**127*	*~2**61*

In present research, it was used Python V3 to generate the circles on a canvas. Python uses the Mersenne Twister algorithm as the core generator for random numbers, due to it produces 53-bit precision floats and has a period of [(2**19937)-1], being this, one of the most extensively tested random number generators in existence (Python Software Foundation, 2023).

It´s good to remark that real aleatory numbers or random numbers do not exits, it is because they are better known pseudorandom numbers, however for purposes of our study we use the expression “random numbers” as a commonly used phrase to refer to pseudorandom numbers. As a note, there are cases like random.org which calculate what they call: real random numbers, based on atmospheric noise, however, the mentioned noise is triggered by a complex number of factors once unveiled, turns in to a determined seeded number.

2.1 Methodology

The objective in present research is to state one of the first steps to overcome the lack in SEM images availability, due to the dearth of available images, which makes it complex to create a basement for studies where that kind of images are needed. It is well known in the materials field that most of the experts have their own images, and while they perform new studies their personal databases get increased, however those images resemble private. To help on the issue, it was determined the need to generate SEM images like, due to lack of SEM images´ availability, a theoretical framework was integrated to understand the problem, it was also defined the initial regular shape (the circle) to approach SEM image like considering random drawing criteria, to finally approaching the SEM image like´s generation thorough circle´s regular shapes (Figure 3).

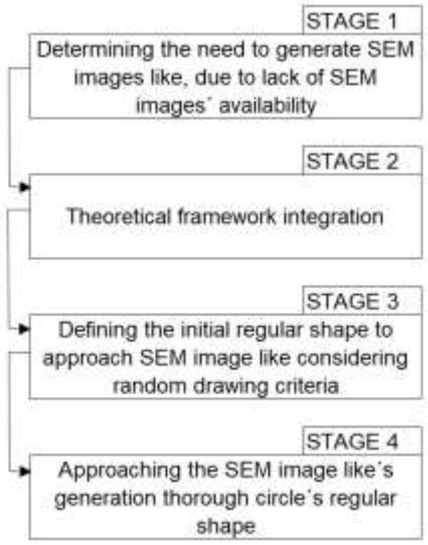
STAGE 1
Determining the need to generate SEM images like, due to lack of SEM images´ availability
STAGE 2
Theoretical framework integration
STAGE 3
Defining the initial regular shape to approach SEM image like considering random drawing criteria
STAGE 4
Approaching the SEM image like´s generation thorough circle´s regular shape

Figure 3. Representation of applied methodology

3. INITIAL APPROACH TO GENERATE SEGMENTED NANOPARTICLES LIKE IMAGES

Once taken into consideration the chance to use aleatory factors to enrich the possibility of generating similar images to those real ones, as initial step to overcome a possible lack of information or lack of images availability. It was integrated aleatory criterions to approach real images like those sourced from SEM images corresponding to thin films.

In the following figure (Figure 4), it is shown a segmented image obtained from a SEM image of a pure ZnO thin film, using the Chan & Vese (1999) segmentation method.

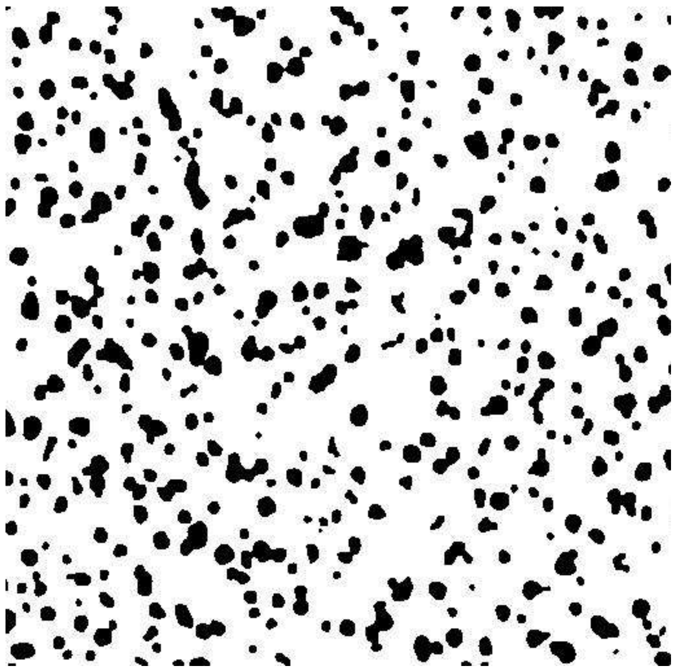

Figure 4. Segmented SEM image using Chan-Vese method. Source: Adapted from Amutha et al. (2014)

In the late image it can be identified 104 blobs (binary large objects), considering those blobs over 100 pixels (Table 3).

Table 3. Number of blobs detected depending on the minimum pixels´ size.
Source: Self-made based on blob detection and counting algorithm

Number of blobs detected	712	135	118	117	116	112	111	110	107	106	104	104	104	102
Minimum blob considered area [Pixels]	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*	*110*	*120*	*130*

As mentioned before, data augmentation in some fields, has a lack of available information, particularly in the context of scanning electron microscope images from thin films. To start overcoming this issue it was considering the circumference parametric equations and random criterions on circumference drawing and fulfillment.

In preset study it was used the function cv2.fillPoly in Python version 3, to draw fulfilled circumferences (circles) which, once merged, can approach a segmented image like. In the following algorithm it is shown the drawing process for the circles on a canvas (Algorithm 3).

Algorithm 3. Drawing of random number of circles with random criterions through parametric equations

```
1:  procedure drawing_circles(lines, angle, canvas, number of circles, radius, circles center, circumference)
2:      importing libraries (numpy
                             matplotlib.pyplot
                             random
                             PIL
                             cv2)
3:        defining an aleatory number of lines to draw the circumference´s perimeter
4:          defining the angles of drawing coverage from 0 to $2\pi$, according to the previous number of lines
5:        generating a canvas to draw the circles (final canvas of 768 x 768 pixels)
6:        defining the random number of circles to be draw on the canvas
7:          for i, number of circles do
8:            defining a random value for circle´s radius
9:            call function Trace_circle
10:         end for
11:           def Trace_circle():
12:               defining random center circle´s coordinates´
13:               defining coordinates from lines tracing along the circle´s circumference
14:               coincide last circumference´s coordinates with initial circumference´s coordinates
15:               drawing and fulfilling randomized circumference
16:               save generated image
17:          end Trace_circle
18:       merging generated circles
```

Considering a random number of drawn circles, random number of lines to draw the circumference´s perimeter, the angle of drawing coverage from 0 to 2π, random number of circles to be draw on the canvas, random value for circle´s radius and random center circle´s coordinates, it was obtained similar images to the following one (Figure 5) with 816 randomized circles drawn.

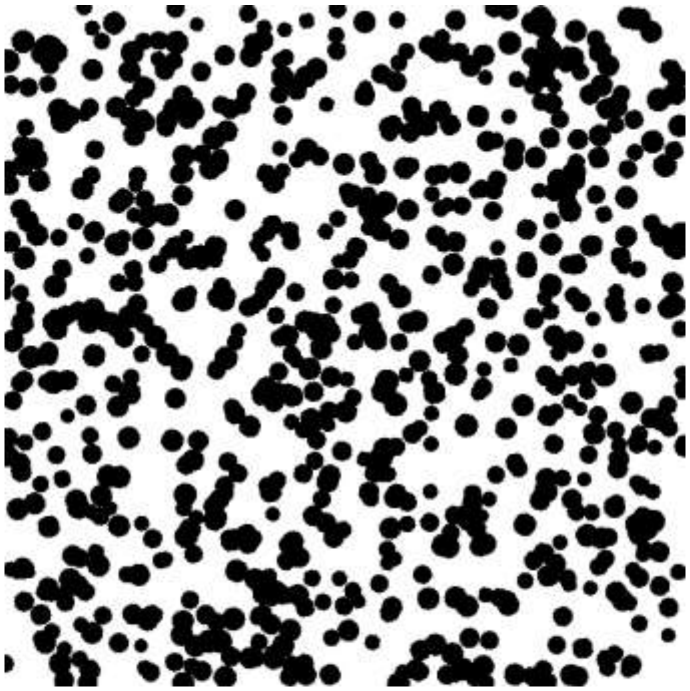

Figure 5. 816-randomized-circles´s generated image, segmented SEM image like, by drawing fulfilled circumferences. Source: Self-made with Colab® by Google® notebook based on Python V3

Additionally, when the resultant 816 randomized drawn circles´ image was eroded with an erosion matrix of 7 x 7, the following image was obtained (Figure 6).

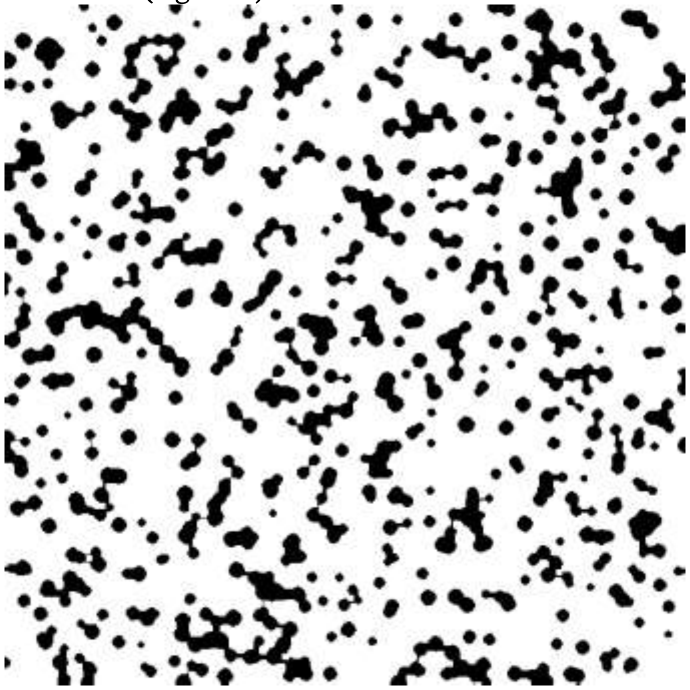

Figure 6. 816-eroded-randomized-circles´s generated image, segmented SEM image like, by drawing fulfilled circumferences. Source: Self-made with Colab® by Google® notebook based on Python V3

Comparing the resultant artificial randomized and eroded image with the Amutha et al. (2014) segmented image, it can be seen a visual similarity (Figure 7). However, the visual similarity is a subjective criterion to state a real similarity between synthetic and real images, it can be taken as a first step to develop a general criterion to overcome the lack in the availability of information thorough image generation task.

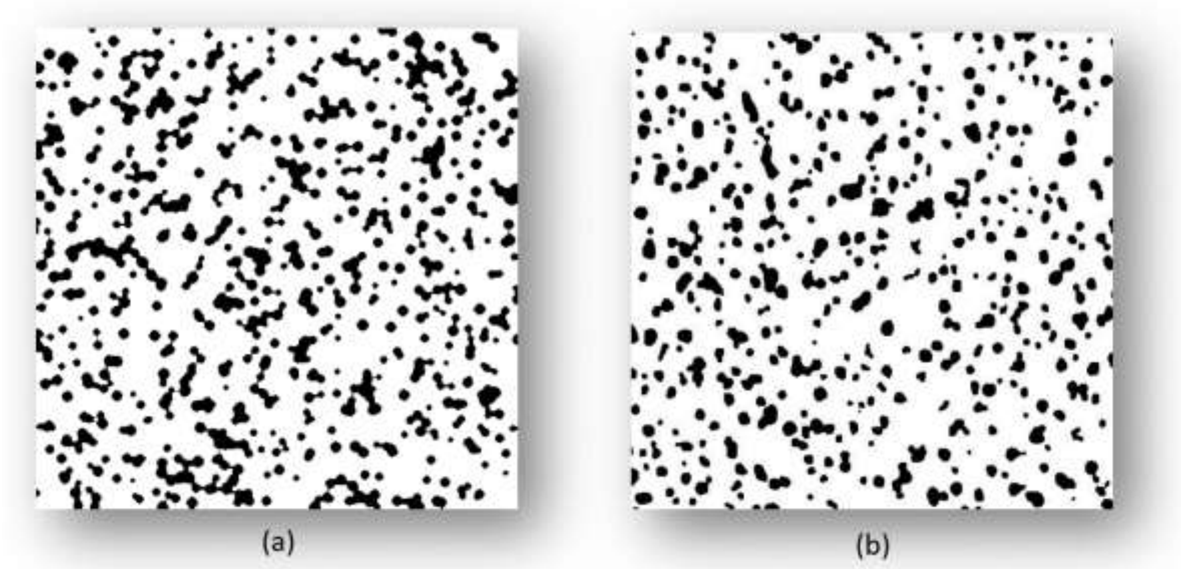

Figure 7. Visual similarity among (a) 816-eroded-randomized-circles´s generated image versus (b) Amutha et al. (2014) segmented image. Source: Self-made with Colab® by Google® notebook based on Python V3 and Amutha et al. (2014)

4. DISCUSSION AND CONCLUSIONS

Images and their shapes exhibit greater complexity compared to the simple geometric forms. Nevertheless, these regular shapes serve as an initial approach for generating more intricate and desired shapes. By introducing random criterions on drawing of regular shapes or contours through parametric equations, the drawing process can be enriched. Random or aleatory numbers, often referred to as pseudo-aleatory numbers or pseudorandom numbers, find diverse applications in the realm of computers. One crucial characteristic of a pseudo-random number is that it requires a seed to initiate its calculation.

Considering the possibility of using random criterions to enhance image´s generation resembling real-world ones, especially when facing a lack of information or a scarcity of available images, aleatory criteria can be integrated into the approach for generating images similar to those obtained from SEM images corresponding to thin films.

In this research, Python V3 was employed to draw circles on a canvas, which produced visually similar segmented SEM images as first step to make available similar SEM images. This research focused on SEM images due to the lack of that kind of images availability. The following step would be to produce similar segmented SEM images adding a random component to the circles perimeter to overpass the perimeter regularity of the so-called regular shapes.

REFERENCES

[1] Amutha, C., Dhanalakshmi, A., Lawrence, B., Kulathuraan, K., Ramadas, V. & Natarajan, B. (2014). Influence of concentration on structural and optical characteristics of nanocrystalline ZnO thin films synthesized by Sol-Gel dip coating method. Progress in Nanotechnology and Nanomaterials, 3(1), 13-18, URL: [https://api.semanticscholar.org/CorpusID:136473361].

[2] Chan, T. & Vese, L. (1999). Chapter 13: An Active Contour Model without Edges. In: Nielsen, M., Johansen, P., Olsen, O.F., Weickert, J. (eds) Scale-Space Theories in Computer Vision. Scale-Space 1999. Lecture Notes in Computer Science, vol 1682. Springer, Berlin, Heidelberg. URL: [https://link.springer.com/chapter/10.1007/3-540-48236-9_13], DOI: [10.1007/3-540-48236-9_13].

[3] Elasri, M., Elharrouss, O., Al-Maadeed, S., & Tairi, H. (2022). Image generation: A review, Neural Processing Letters, 54(5), 4609-4646, URL: [https://link.springer.com/article/10.1007/s11063-022-10777-x], DOI: [10.1007/s11063-022-10777-x].

[4] James, F. (1988). A review of pseudorandom number generators, CERN-Data Handling División, URL: [http://cds.cern.ch/record/192937/files/cer-000104530.pdf].

[5] Matsumoto M. & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul., 8(1), 3–30. DOI: [10.1145/272991.272995], URL: [https://dl.acm.org/doi/pdf/10.1145/272991.272995].

[6] Pallares-Rusiñol, A., Bernuz, M., Lima-Moura, S., Fernández-Senac, C., Rossi, R., Martí, M. & Pividori, M.I. (2023). Chapter 2: Advances in Exosome Analysis. In Advances in Clinical Chemistry, Elsevier: Amsterdam, The Netherlands, 112, pp. 69–117, ISBN 978-0-443-19284-5, URL: [https://www.sciencedirect.com/science/article/pii/S0065242322000701], DOI: [10.1016/bs.acc.2022.09.002].

[7] Pérez-Rastelli, J., Godoy, J., Villagra, J., & Onieva. E. (2013). Trajectory generator for autonomous vehicles in urban environments. IEEE ICRA - IEEE International Conference on Robotics and Automation, URL: [https://inria.hal.science/hal-00789760/file/ICRA_Perez_et_al_2360.pdf], DOI: [10.1109/ICRA.2013.6630608].

[8] Python Software Foundation (2023). random — Generate pseudo-random numbers, Numeric and Mathematical Modules, The Python Standard Library, 3.11.4 Documentation, Python Software Foundation, URL: [https://docs.python.org/3/library/random.html].

[9] Wang, Q., & Boyer, K. L. (2012). The active geometric shape model: A new robust deformable shape model and its applications. Computer vision and image understanding, 116(12), 1178-1194, URL: [https://wangquan.me/files/research/AGSM_CVIU_2012.pdf] & [https://www.sciencedirect.com/science/article/pii/S1077314212001154], DOI: [10.1016/j.cviu.2012.08.004].

[10] Xu, F., Midoux, N., Li, H., Hébrard, G., & Dietrich, N. (2019), Characterization of Bubble Shapes in Non-Newtonian Fluids by Parametric Equations. Chemical Engineering & Technology, 42: 2321-2330, URL: [https://onlinelibrary.wiley.com/doi/epdf/10.1002/ceat.201800690], DOI: [10.1002/ceat.201800690].

[11] Zhu, Y. & Inada, H. (2012). Scanning Electron Microscopy. In: Bhushan, B. (eds) Encyclopedia of Nanotechnology. Springer, Dordrecht. URL: [https://link.springer.com/referenceworkentry/10.1007/978-90-481-9751-4_110], DOI: [10.1007/978-90-481-9751-4_110].

Corresponding author´s e-mail: ***m_en_i_molina_ruiz@engineer.com; hmolina@uaeh.edu.mx***