Online Evasive Strategy for Robotic Exploration Problem using Sierpinski Knoop curve

Ashay Wakode[1], Arpita Sinha[2]

Abstract— **The paper deals with the robotic exploration problem using the Sierpinski Knoop curve. A simply closed region is divided into triangles, and Sierpinski Knoop curve is used to explore each triangular region. The entire region can have one or more obstacles. An algorithm is proposed to avoid obstacles online and explore the desired region. Simple geometric observations were used to formulate the algorithm. The non-uniform coverage and multiple obstacle problems are also dealt with in brief.**

I. Introduction

In 1878, George Cantor demonstrated that an interval [0,1] can be mapped bijectively onto $[0,1]^2$. The next question asked was whether there exists a curve that passes through every point of an area. G.Peano discovered one such curve, and hence curves are called Peano curves or Space-filling curves. More Space-filling curves were discovered later on by E. Moore, H. Lebesgue, W. Sierpinski, and G. Polya [1]. W. Sierpinski discovered the Sierpinski curve, which maps a unit interval onto a unit square (fig 1a). The Sierpinski curve for a unit square can be thought of as made of two similar triangular Space-filling curves (fig 1b), each of which is known as Sierpinski Knoop Curve, after Konrad Knoop, who described the curve mathematically [2]. The Sierpinski Knoop Curve is referred to as SKC in short in the paper.

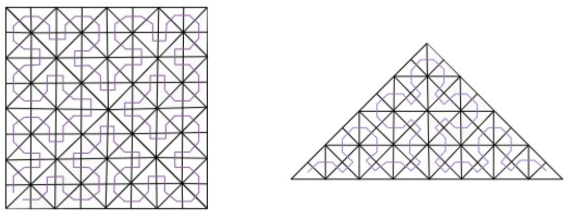

(a) Sierpinski Curve (b) Sierpinski Knoop Curve

Fig. 1: Space-Filling Curves for Square and Triangle

Space-filling curves are the maps from a 1-D interval to a higher dimension. Space-filling curves have properties like locality preservation, exhaustive coverage, fractal nature, and many more. It provides a novel solution to the *robotic exploration* problem, in which a robotic system travels to explore a given area entirely, avoiding obstacles. The exploration conducted by a collaborative robotic system is robust in terms of the guarantee of completion and communication failures when using a Space-filling curve [3]. Exploration task using collaborative aerial robotic systems without the presence of an obstacle is studied in [4], the zig-zag path is used to transverse a convex polygonal area. Many times, a refined search of a part of the area is intended, since the probability of discovery of the object of interest may be known to be high beforehand, or simply a more rigorous search is intended [5]. Building upon this [6] uses Hilbert's Space-filling curve for traversing the area with parts having a non-uniform probability of discovery of an object of interest or parts which need to be searched with dissimilar rigor. The robotic exploration problem is a part of the Coverage Planning Problem (CPP). CPP is the determination of the path for a robotic system, such that it performs tasks like cleaning, searching, painting, and others over the given area, avoiding obstacles. Various ways to generate such a path for single/multiple robotic systems for coverage of area/volume by decomposing them into smaller regions are summarised in [7]. [8] proposes a solution to CPP without prior knowledge of the static environment using neural-neighbourhood analysis; the presented algorithm can be extended to dynamic systems with knowledge of the changes taking place in the environment. Robotic exploration of Hilbert's Space-filling curve in the presence of an obstacle blocking a single way-point on the curve is studied in [9]; The paper proposes an alternate path for evasion without prior knowledge of the location of the obstacle (Online Evasive Strategy). [10] builds upon [9] and proposes an alternate path for an obstacle sizable enough to occupy two adjacent way-points. Alternate path generation for exploration of a square area using the Sierpinski curve with obstacles is presented in [11]. However, the solution presented needs knowledge about the location of obstacles ahead of time (Offline Evasive Strategy).

This paper deals with the robotic exploration of a flat and static environment using SKC. An algorithm is proposed which suggests the required change in the path to evade the obstacle and get back on the original route. The obstacle is assumed to cover only a single way-point. The proposed solution can be used for multiple disjoint obstacles (except the first three and last three way-points of SKC used for exploration).

The preliminaries required for generating SKC for isosceles right triangle are discussed initially, which are extended to a general closed curve in the following sections. Generation of SKC requires area decomposition and connecting the smallest units (cells) in a particular manner. The area of the cells is such that the robotic system can sense the entire cell once placed at a specific location. In subsequent sections, lemmas are proved geometrically to ensure the applicability of the algorithm. Applicability of the algorithm to multiple obstacles and non-uniform coverage situations is discussed

[1]Undergraduate Student, Mechanical Engineering; [2]Associate Professor, Systems and Control Engineering; Indian Institute of Technology Bombay, India {ashaywakode,arpita.sinha}@iitb.ac.in

towards the end.

II. Preliminary on Sierpinski Knoop Curve

A. Mathematical Description

In this paper, generation of SKC which maps from unit interval $\mathbf{I} = [0,1]$ to $\mathbf{T}$, where $\mathbf{T}$=$\{(x,y)$: $y \geq 0, x-y \geq 0$, $x+y-2 \leq 0\}$ is introduced. Building on this, SKC generation is extended to map from an unit interval $\mathbf{I}$ to a scalene triangle.

k^{th} ($k \in \mathbb{Z}^{0+}$) iteration of SKC is constructed by dividing $\mathbf{I}$ into 2^{k+1} equal smaller interval and mapping them to 2^{k+1} isosceles triangular cells created by dividing $\mathbf{T}$. Here, k is called the order of the SKC. The divisions generated on $\mathbf{I}$ are mapped to the triangular cells on $\mathbf{T}$. A piece-wise straight line curve is generated by joining centers of triangular cells in the order of sub-intervals on $\mathbf{I}$ to which they are mapped; thus SKC is generated.

A geometric way of generating SKC is presented. A higher order SKC is generated by further dividing a lower order SKC. An increase in the order corresponds to an increase in the number of triangular cells. Simple geometric patterns along with production rules are used to generate SKC. The patterns and the production rules are specific to the parity of the order of the SKC, a particular set of patterns and production rules can generate SKC with even numbered order, while a different set of patterns and production rules are needed for the generation of SKC with odd numbered order. [2] presents a way to generate odd iterations (iteration with an order which is an odd number) of SKC, which is explained first, building upon the same concepts a way to generate even iterations (iteration with an order which is an even number) is presented.

Consider patterns $\mathbf{S}$, $\mathbf{Z}$, $\mathbf{R}$ and $\mathbf{P}$ as shown in fig 2. Here, S is the first iteration of SKC ($Order = 1$). The following production rules are used to generate further odd iterations:

$$\mathbf{S} \Rightarrow \mathbf{S} \nearrow \mathbf{R} \rightarrow \mathbf{P} \searrow \mathbf{S}$$

$$\mathbf{R} \Rightarrow \mathbf{R} \nwarrow \mathbf{Z} \uparrow \mathbf{S} \nearrow \mathbf{R}$$

$$\mathbf{Z} \Rightarrow \mathbf{Z} \swarrow \mathbf{P} \leftarrow \mathbf{R} \nwarrow \mathbf{Z}$$

$$\mathbf{P} \Rightarrow \mathbf{P} \searrow \mathbf{S} \downarrow \mathbf{Z} \swarrow \mathbf{P}$$

Here, the intervening arrows ($\rightarrow, \leftarrow, \uparrow, \downarrow, \nwarrow, \nearrow, \swarrow, \searrow$) between $\mathbf{S}$, $\mathbf{Z}$, $\mathbf{R}$ and $\mathbf{P}$ represents the direction from the end of the previous pattern to start of the succeeding pattern.

The above-mentioned strategy is used to generate the odd iterations of SKC. This is shown in fig 3.

Next, a strategy for generating even iterations of SKC is presented. Consider patterns $\mathbf{P}$*, $\mathbf{Z}$*, $\mathbf{R}$* and $\mathbf{S}$* as shown in Figure 4. The following production rules are used:

$$\mathbf{P}* \Rightarrow \mathbf{P}* \rightarrow \mathbf{Z}* \nearrow \mathbf{S}* \uparrow \mathbf{P}*$$

$$\mathbf{Z}* \Rightarrow \mathbf{Z}* \downarrow \mathbf{R}* \searrow \mathbf{P}* \rightarrow \mathbf{Z}*$$

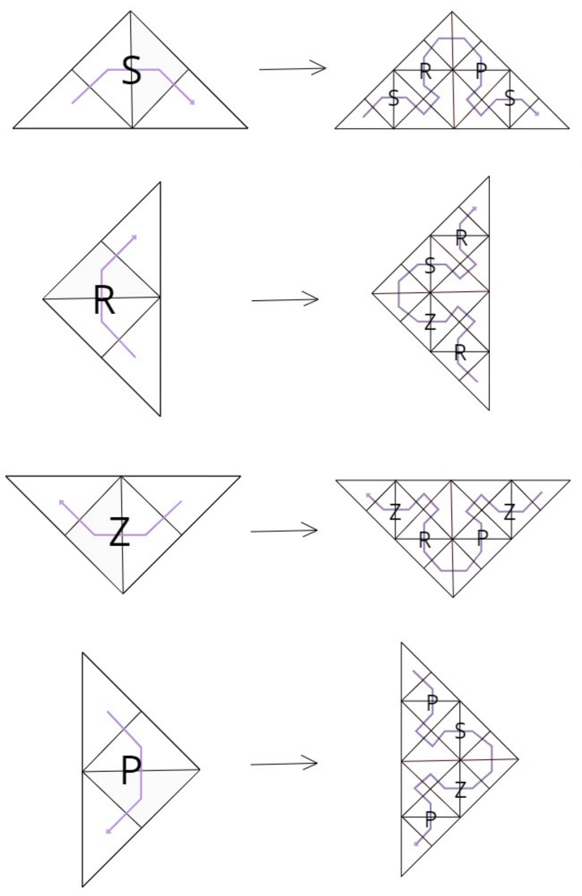


Fig. 2: Simple Patterns for odd iterations of SKC

$$\mathbf{R}* \Rightarrow \mathbf{R}* \leftarrow \mathbf{S}* \swarrow \mathbf{Z}* \downarrow \mathbf{R}*$$

$$\mathbf{S}* \Rightarrow \mathbf{S}* \uparrow \mathbf{P}* \nwarrow \mathbf{R}* \leftarrow \mathbf{S}*$$

The production rules can be used for generating even iterations, similarly as done in the odd case. **P*→Z*** is first even iteration of SKC ($Order = 2$).

B. Extension to Scalene Triangle and General Closed curve

The SKC generated in the previous subsection spans an isosceles right triangle. However, in real-life applications, the area to be explored may not be a triangle, polygon, or level surface. In this paper, a way to span SKC onto an area enclosed by a planar simple closed curve is worked out. If a way for SKC to span such an area with an obstacle is worked out, then the evasive strategy presented in section III can be applied to it.

Firstly, SKC for $\mathbf{T}$ is generated. Then SKC in $\mathbf{T}$ is transformed to an arbitrary triangle whose corner coordinates are known using projective transformation; this is shown in Fig 5.

Following transformation equation is used:

$$\begin{bmatrix} x\prime \\ y\prime \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & \gamma \\ \epsilon & \zeta & \eta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

Here, x′ and y′ are the coordinates of the point enclosed in the arbitrary scalene triangle corresponding to point (x,y)

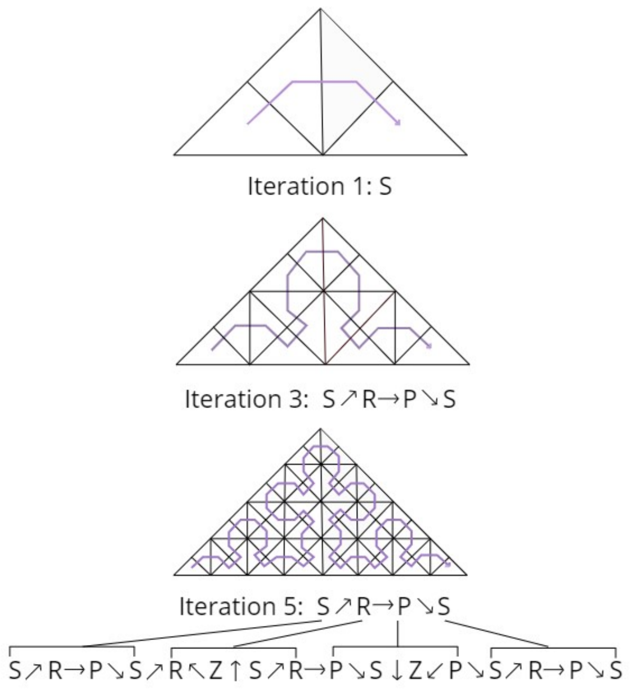


Fig. 3: Using production rules for generating further iterations

inside $\mathbf{T}$. The coordinates of corners of $\mathbf{T}$ and the scalene triangle (which is being mapped) are known. Putting those in eq. (1) would yield 6 equations, which can be solved for α, β, γ, ϵ, ζ and η. Once the elements of the transformation matrix are known, the transformation matrix can be used to find the points on SKC spanning scalene triangle corresponding to points on SKC spanning $\mathbf{T}$.

While generating SKC for $\mathbf{T}$, the region was divided by drawing perpendiculars from the right angle, and the same operation was iteratively performed on the smaller triangles several times depending upon the number of iteration aimed to be generated. The perpendiculars drawn are also the medians due to the geometry of $\mathbf{T}$. The lines remain median even after performing projective transformation due to the linearity of the operation. So, it can be concluded that, area decomposition for generating SKC for scalene triangle is constructed using medians. [2] points out that area decomposition for any arbitrary triangle can be done by any cevian (a line segment starting on one of the vertexes of the triangle and ending on the opposite side of the triangle). However, using medians for area decomposition can be helpful. The cells (smallest triangular element of area decomposition) in the area decomposition form 4 groups of congruent triangles. This property of area decomposition by medians is helpful in deciding the iteration of SKC needed for a particular, given the sensing radius of the robotic system. Therefore, area decomposition using medians is considered in this paper.

The boundary of any simple closed curve can be approximated by straight lines; the polygonal region formed may not be unique. Also, it is always possible to triangulate a polygon [12]. Each triangle formed after triangulation can be spanned by an SKC, as already discussed. So, a way to explore the polygonal region and therefore a region enclosed

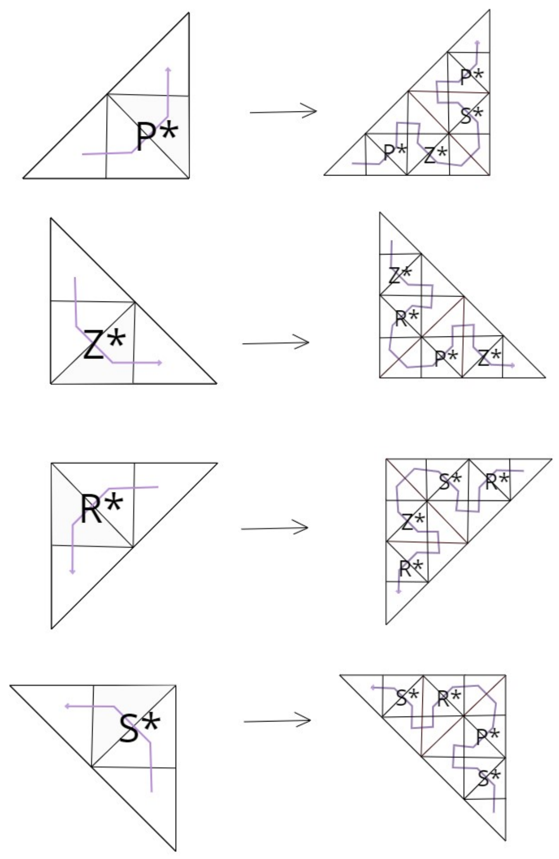


Fig. 4: Simple Patterns for even iterations of SKC

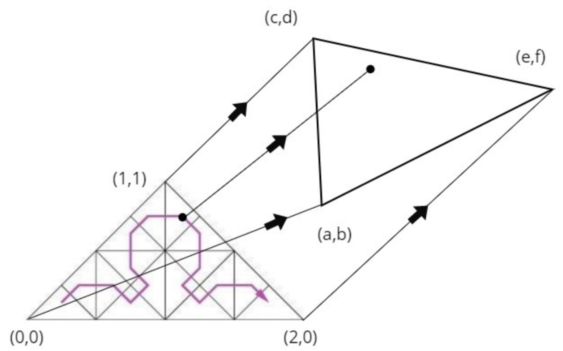


Fig. 5: Mapping point in $\mathbf{T}$ to a point in arbitrary triangle

by a planar simple closed curve using SKC is devised.

III. Main Result

A. Problem Formulation

The problem of robotic exploration is considered on an arbitrary triangle with a single obstacle. The obstacle is assumed to occupy only one cell. It may or may not cover the cell completely, but it is assumed that the cell cannot be visited by the robot. Call the center of the triangular cell "node". The number of nodes present in the n^{th} iteration of SKC is 2^{n+1}. The nodes are numbered as shown in Fig 6 for iteration 2. It is assumed that the first 3 nodes (1,2,3) and the last 3 nodes (2^{n+1}-2, 2^{n+1}-1, 2^{n+1}) are not blocked by an obstacle, since no evasive path is possible for these cases.

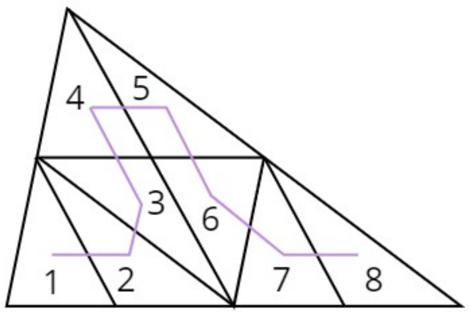


Fig. 6: Iteration 2 SKC with nodes numbered

When an arbitrary triangle is to be explored, it is assumed that coordinates of corners are known. With this information, the coordinates of corners of every triangular element can be obtained by using the projective transformation operation. The sensing radius of the robotic system is already known to us. The iteration of SKC is chosen such that the robotic system can sense the entire triangle. It is also assumed that sensing radius is sufficient to sense an obstacle in any triangle sharing side with the triangle in which the robotic system is present.

In an exploration problem, it is intended to explore the entire region while avoiding obstacles. Due to the circular sensing area, the obstacle (assumed to be present at $i+1^{th}$ node) can be sensed much before arriving at the i^{th} node. Nevertheless, for exploring the entire region, the robotic system travels up to the i^{th} node and then follows an alternate path to the $i+2^{th}$ node.

So, the problem boils down to planning a path from the i^{th} node to the $i+2^{th}$ node avoiding obstacle at $i+1^{th}$ node. There are five possible geometries of three consecutive triangles (fig 7) present in the area decomposition of SKC.

Lemma 1: Cases $1-5$ are the only possible geometries of 3 consecutive triangles in the area decomposition of SKC.

Proof. As shown by eq.1, SKC mapped onto an isosceles right triangle can be converted to SKC mapping onto a scalene triangle using a linear transformation. So, the collinearity of points and median property (Median from a vertex divides opposite side into equal parts) are retained. Therefore, the geometric properties of cases 1-5 will remain the same before and after transformation. SKC mapping onto $\mathbf{T}$ (isosceles right triangle) is used to prove the lemma.

The lemma is proved for odd iterations using production rules introduced in section II. The lemma can be proved similarly for even iteration.

Consider iteration 1 from fig 3. Iteration 1 ($\mathbf{S}$) has Case 3 and Case 5 geometries, as shown in the Fig 8.

As one proceeds from iteration 1 to iteration 3, patterns $\mathbf{S}$, $\mathbf{R}$, $\mathbf{P}$ and $\mathbf{S}$ are joined in order to get higher iterations, again shown in fig 3. Iteration 3 has 3-triangle geometries from Cases 1-5 of patterns $\mathbf{S}$, $\mathbf{R}$, $\mathbf{P}$ and $\mathbf{S}$, as well as new intervening geometries as shown in Fig 9.

Now, any area decomposition of odd iteration can be represented by series of $\mathbf{S}$, $\mathbf{R}$, $\mathbf{P}$ and $\mathbf{S}$ along with intervening arrows. Here, all the possible doublets (a pair of two patterns)

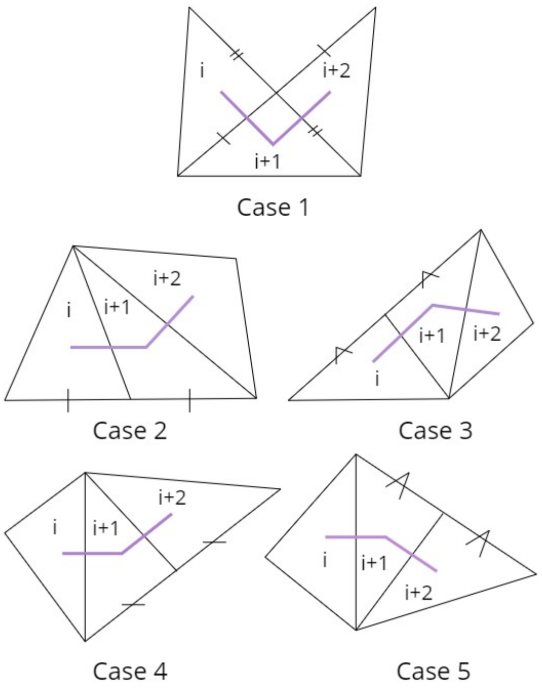


Fig. 7: Possible geometries

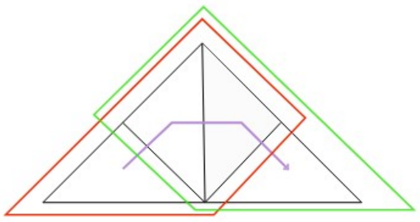


Fig. 8: Case 3 (Red) and Case 5 (Green) geometries in pattern $\mathbf{S}$

that can be present in any such series are,

$$\begin{array}{cccc} \mathbf{S} \nearrow \mathbf{R} & \mathbf{R} \rightarrow \mathbf{P} & \mathbf{P} \searrow \mathbf{S} & \mathbf{Z} \uparrow \mathbf{S} \\ \mathbf{S} \downarrow \mathbf{Z} & \mathbf{R} \nwarrow \mathbf{Z} & \mathbf{P} \leftarrow \mathbf{R} & \mathbf{Z} \swarrow \mathbf{P} \end{array}$$

There are 12 (4×3) possibilities for doublet, but after crossing out geometrically infeasible doublets, only the above-mentioned doublets remain. Each of these doublets is used in one of the production rules (can be easily verified by looking at production rules in the previous section), and hence each one will be used to produce a series that will build an odd iteration of SKC.

In table I, it can be seen that each doublet has only Cases 1-5 of 3-triangle geometry. All the doublets that will even-

Doublet	3-triangular Geometries present (Bold: Geometries created after joining the patterns)
$\mathbf{S} \nearrow \mathbf{R}$	3 5 **1 1** 3 5
$\mathbf{S} \downarrow \mathbf{Z}$	3 5 **1 1** 4 2
$\mathbf{R} \rightarrow \mathbf{P}$	3 5 **3 5** 3 5
$\mathbf{R} \nwarrow \mathbf{Z}$	3 5 **1 1** 4 2
$\mathbf{P} \searrow \mathbf{S}$	3 5 **1 1** 3 5
$\mathbf{P} \leftarrow \mathbf{R}$	3 5 **4 2** 3 5
$\mathbf{Z} \uparrow \mathbf{S}$	4 2 **4 2** 3 5
$\mathbf{Z} \swarrow \mathbf{P}$	4 2 **1 1** 3 5

TABLE I: Doublets and 3-triangle geometries present in them for odd numbered iterations of SKC

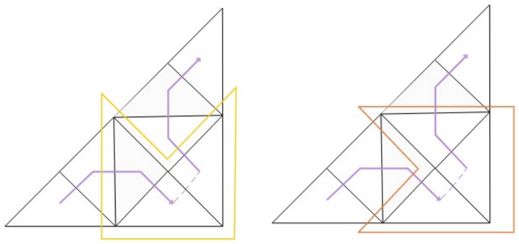


Fig. 9: Case 1 geometries after joining pattern **S** and **P** while constructing iteration 3

Doublet	3-triangular Geometries present (Bold: Geometries created after joining the patterns)
$\mathbf{P}* \rightarrow \mathbf{Z}*$	2 4 **1 1** 2 4
$\mathbf{P}* \nwarrow \mathbf{R}*$	2 4 **5 3** 5 3
$\mathbf{Z}* \downarrow \mathbf{R}*$	2 4 **1 1** 5 3
$\mathbf{Z}* \nearrow \mathbf{S}*$	2 4 **2 4** 5 3
$\mathbf{R}* \searrow \mathbf{P}*$	5 3 **2 4** 2 4
$\mathbf{R}* \leftarrow \mathbf{S}*$	5 3 **1 1** 5 3
$\mathbf{S}* \uparrow \mathbf{P}*$	5 3 **1 1** 2 4
$\mathbf{S}* \swarrow \mathbf{Z}*$	5 3 **5 3** 2 4

TABLE II: Doublets and 3-triangle geometries present in them for even numbered iterations of SKC

tually generate an area decomposition for SKC have only Cases 1-5 geometries. Therefore, every area decomposition of odd iteration SKC will only have Cases 1-5 geometries.

It can be similarly proven that every area decomposition of even iteration SKC will only have Cases 1-5 3-triangular geometries. In this case, patterns **P***, **Z***, **S*** and **R*** are used. The possible doublets and corresponding geometries are presented in table II. ■

Cases 2 and 3 are similar since i^{th} and $i+1^{th}$ triangles are divided by a median, while the $i+2^{th}$ triangle merely shares a side with $i+1^{th}$ triangle. Going on the same line, in Cases 4 and 5, $i+1^{th}$ and $i+2^{th}$ triangles are divided by median and i^{th} triangle merely shares side with $i+1^{th}$ triangle. The strategy presented works the same for Case 2,3 and Case 4,5. Name Case 1 as **a**, Cases 2,3 as **b** and Cases 4,5 as **c**. Tag i^{th} node as **a** if the geometry of i^{th}, $i+1^{th}$ and $i+2^{th}$ triangles comply with Case 1. Similarly, tag nodes as **b** and **c**. The tag for each node entirely depends upon i, for a particular iteration of SKC.

From here on, indicate the coordinates of the node for the i^{th} triangle in SKC area decomposition of order n as i_n. Represent the coordinates of the corner of the i^{th} triangle which does not lie on the side shared with the $i+1^{th}$ triangle as $D_{i,i+1}$. Therefore, $D_{i+1,i}$ represents the coordinates of the corner of the $i+1^{th}$ which is not shared with i^{th} triangle. Observe that all the three triangles in every case have a common vertex, let's refer it as C. Further, denote the vertex common to triangles i and $i+1$ only as $B_{n,n+1}$. Define a function $\mathbf{A}(U,V,W)$, which gives the area of the triangle with U, V and W as corners. Algorithm 1 is proposed to tag nodes as **a**, **b** and **c**.

Algorithm 1 Algorithm for tagging node as a, b or c

```
1: Input : $i$, $C$, $D_{i,i+1}$, $D_{i+1,i}$, $B_{i,i+1}$
2: Output : Tag for $i$
3: if $A(D_{i,i+1}$, C, $D_{i+1,i}) = 0$ then
4:     Tag $i$ as a
5: else if $A(D_{i,i+1}$, $B_{i,i+1}$ , $D_{i+1,i}) = 0$ then
6:     Tag $i$ as b
7: else
8:     Tag $i$ as c
```

B. Proposed Solution

Evasive strategies are proposed for all the tags in table III. In each strategy, it is assumed that the robotic system is present on the i^{th} node, and an obstacle at the $i+1^{th}$ triangle has been detected. The strategies can be used for any node except the first three and the last three nodes of SKC. The strategies and algorithm presented do not use the fact that medians are used for area decomposition; therefore, the strategies are applicable for other types of area decomposition as well. One of the two possible cases for evasion in **b** and **c** are decided by the smallest distance (normalized distance) required for the detour. The normalized distance for the detour is denoted by L. L gives an idea about which of the two proposed detours requires more distance for evasion.

$$L = \frac{\text{Path Length}}{\text{Straight line distance between Node } i \text{ and Node } i+2}$$

An algorithm for online implementation is discussed in section V. Algorithm 2 takes care that at no point in the detour, the search agent is on the boundary or outside the entire triangle.

C. Validity of the strategy

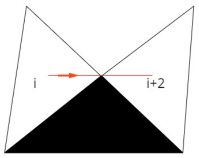
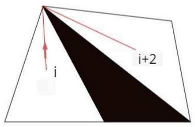
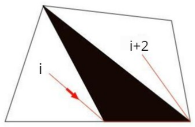
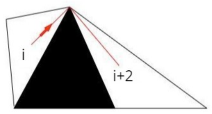
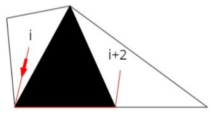
Lemma 2: Only the first two and the last two triangles in SKC area decomposition has all the three vertices on the boundary of the entire triangle. Also, only 2^{nd} and 3^{rd} triangles, both from start and end of SKC, share two vertices, both of which lie on boundary.
Proof. The lemma is proved for SKC on an isosceles right triangle. The transformation of an isosceles right triangle to any scalene triangle will not change the fact that a certain point lies on a particular line. So, proving the lemma for an isosceles right triangle is sufficient.

Mathematical induction is used to prove the lemma. Look at iteration $n = 2$ of SKC (fig 10). It can be seen that all the vertices of the first two and last two triangles in the area decomposition lie on the boundary. Also, 2^{nd} and 3^{rd} triangles from the start and end share vertices that lie on the boundary. This is also true for iteration $n = 3$ (fig 3).

It can be easily deduced that the triangular cells having all their vertices on the boundary should be located near the triangular region's corners since a triangle cannot have all of its vertices on a single line.

Consider, iteration $n = k$ (where k is an odd number) area decomposition for SKC on an isosceles right triangle (fig 11). Here, it can be seen that the lemma is satisfied. $k+1^{th}$ iteration of SKC can be generated by joining two

TABLE III: Evasive Strategies

Sr. No.	Evasive Maneuver	Change of Path	L
1		$i_n \rightarrow$ C $\rightarrow i+2_n$	$L_1 = \frac{\lvert C-i_n\rvert+\lvert i+2_n-C\rvert}{\lvert i+2_n-i_n\rvert}$
2		$i_n\rightarrow$ C $\rightarrow i+2_n$	$L_2 = \frac{\lvert C-i_n\rvert+\lvert i+2_n-C\rvert}{\lvert i+2_n-_n\rvert}$
3		$i_n \rightarrow B_{i,+1} \rightarrow B_{i+1,i+2} \rightarrow i+2_n$	$L_3 = \frac{\lvert B_{i,i+1}-i_n\rvert+\lvert B_{i+2,i+1}-B_{i,+1}\rvert+\lvert i+2_n-B_{i+2,i+1}\rvert}{\lvert i+2_n-i_n\rvert}$
4		$i_n \rightarrow$ C $\rightarrow i+2_n$	$L_4 = \frac{\lvert C-i_n\rvert+\lvert i+2_n-C\rvert}{\lvert i+2_n-i_n\rvert}$
5		$i_n \rightarrow B_{i,i+1} \rightarrow B_{i+1,i+2} \rightarrow N_{i+2}$	$L_5 = \frac{\lvert B_{i,i+1}-i_n\rvert+\lvert B_{i+2,i+1}-B_{i,i+1}\rvert+\lvert i+2_{n+2}-B_{n+2,n+1}\rvert}{\lvert i+2_n-i_n\rvert}$

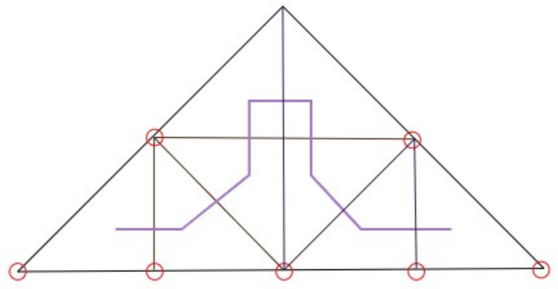


Fig. 10: Iteration 2 with vertices on boundary marked

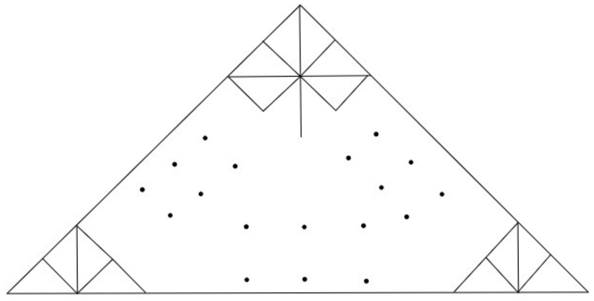


Fig. 11: k^{th} iteration with only corner triangles

k^{th} iterations of SKC, as shown in Figure 12. Here, again it is observed that only the first two and last two triangles have all their vertices on the boundary. Again, the 2^{nd} and 3^{rd} triangles from the start and end share vertices that lie on the boundary. So, the lemma is proved when k is odd. The lemma can be proved in a similar way when k is even. ■

Lemma 3: Using the evasive strategies enlisted in Table III, a robotic system can evade any obstacle lying entirely inside a cell, except when the obstacle is present inside the first 3 or last 3 cells of SKC.

Proof. Table III lists evasive techniques for nodes with tags **a**, **b** and **c**. The tags are given to geometries of cases 1 to 5 (refer fig 7). The cases 1-5 are the only possible geometries of three consecutive triangles in the area decomposition for SKC, as proved in lemma 1. The geometries of cases 2 and 3 are similar; they are water-images. The evasive strategies enlisted in table III are for cases 1, 2 and 4. The strategies for case 2 can be used without modification for case 3. Similarly, the strategies for case 4 can be used for case 5. So, it remains to be shown that strategies enlisted in table III are always possible.

Case 1: Case 1 is impossible if C (refer fig 13) lies on the boundary. If C lies on the boundary, then there will be two line segments originating from C with an interior angle between them greater than 180°, which is infeasible, and hence C will never lie on the boundary. Therefore, the evasive strategy for Case 1 will always be possible.

Case 2 and 3: Let the triangle with obstacle be denoted as ΔDEF. The possible boundary lines which can stop the robotic system from taking a detour in Case 2 and 3 are labeled as m, n, and o in fig 14.

Firstly, assume only lines o and m are present. Then the triangle preceding ΔDEF will have all of its vertices on

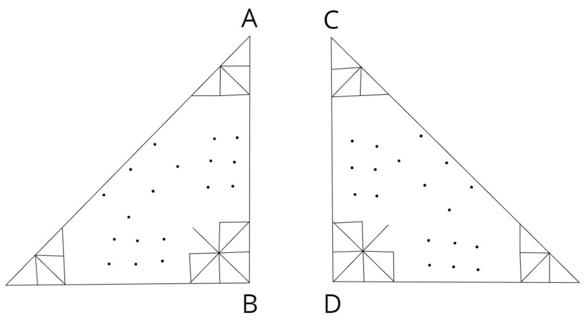


Fig. 12: $k+1^{th}$ iteration generated by superimposing line segment AB over CD

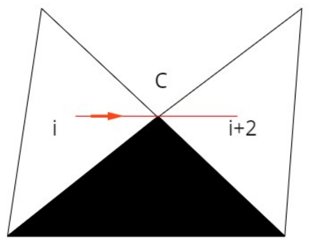


Fig. 13: Case 1

the boundary, and hence it will be either 1^{st} or 2^{nd} triangle from start or end, which implies ΔDEF will be 2^{nd} or 3^{rd} triangle; this contradictions the assumption that the triangle (or cell) with an obstacle is not one of the first or last three triangles in area decomposition.

Secondly, assume only lines o and n are present. In this situation, the robotic system can follow evasive strategy no. 2.

Thirdly, assume only line m and n. Here, ΔDEF and succeeding triangle share two vertices that lie on boundary. This implies ΔDEF is either 2^{nd} or 3^{rd} triangle from start or end. Therefore, a contradiction.

Lastly, assuming the presence of all lines o, m, and n will imply that ΔDEF is 2^{nd} triangle from start or end. Again, a contradiction.

Case 4 and 5: Proceeding on the same lines, Let the triangle with obstacle be denoted as ΔHIJ (refer fig 15). The possible boundary lines which can stop the robotic system from taking a detour in Case 4 and 5 are labeled as x, y, and z as shown in fig 15.

Firstly, assume only lines x and z are present. The triangle succeeding ΔHIJ will have all of its vertices on the boundary, and hence it will be either 1^{st} or 2^{nd} triangle from start or end, hence ΔHIJ will be 2^{nd} or 3^{rd} triangle. A contradiction.

Secondly, assume only lines z and y are present. In this situation, the robotic system can follow evasive strategy no. 4.

Thirdly, assume only lines x and y are present. Here, ΔHIJ and the preceding triangle share two vertices that lie on the boundary. This implies ΔHIJ is either 2^{nd} or 3^{rd} triangle from start or end, respectively. Therefore, a contradiction. Lastly, assuming the presence of all lines x, y, and z will imply that ΔDEF is 2^{nd} triangle from start

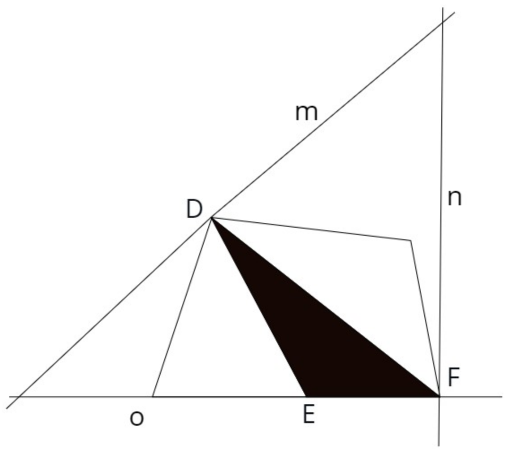


Fig. 14: Case 2 and 3

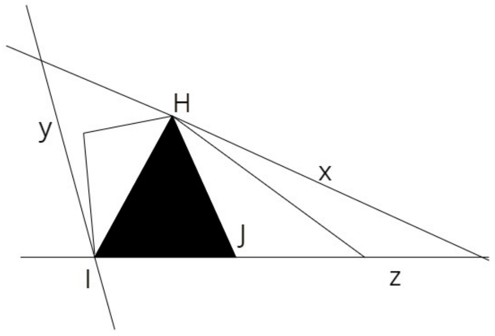


Fig. 15: Case 4 and 5

or end. Again, a contradiction.

Therefore, it can be asserted that using the evasive strategies enlisted in table III, a robotic system can evade any obstacle with size enough to cover a triangular cell, except when the obstacle is present on the first three or last three triangular cells of SKC. ■

Lemma 4: Using the evasive strategies enlisted in table III, a robotic system traverses every node except the node with the obstacle of an SKC if the obstacle is present on any node except first three and the last three nodes of the SKC.
Proof: The robotic system travels to the node preceding the obstacle on SKC. Then it performs the detour to the node succeeding the node with an obstacle. It then again follows SKC until the end of the search maneuver. So, every node on SKC is visited except the obstacle node. If the obstacle is present at the first or last three nodes, then the path is blocked entirely, and no evasive strategy is possible. ■

IV. Multiple Obstacles and Non-Uniform Coverage

The strategy presented can be extended to situations wherein multiple obstacles are present. The alternate path for evading a single obstacle depends only on the triangular cell preceding and subsequent to the triangle with an obstacle. So, if j^{th} (some positive integer denoting node number) triangle has an obstacle, then the strategy presented will not work if $j+1^{th}$ triangle also has an obstacle. Nevertheless, the strategy can be applied in a usual way if the $j+2^{th}$ triangle has an obstacle. It can be concluded that strategy will work for any number of disjoint obstacles (not present on the first three and last three triangles) with size enough to occupy a single triangular cell. This is shown in Figure 16.

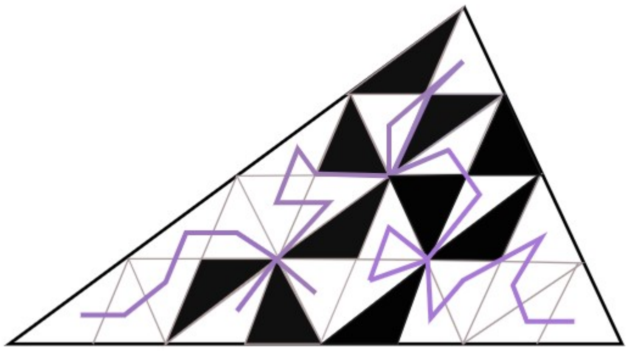


Fig. 16: Robotic Exploration with multiple obstacles

In some robotic exploration situations, it is already known that some parts of the entire region of exploration have more chances of discovery of an object of interest, and therefore it is wise to search such parts more rigorously. For example, a robot deployed to search for water in a particular region should increase the rigor of search in parts with green patches of grass. Such situations are known as non-uniform coverage searches. [6] suggests strategies for non-uniform coverage using Hilbert's space filling curve. [9] proposes an online evasive algorithm for an alternate path when a single obstacle is present in the trajectory defined by [6]. [5], [6], and [9] assume an aerial robotic system with sensor footprint to be square for identifying parts with a greater chance of discovery of objects of interest. The non-uniform coverage search can be performed by another aerial robotic system or a ground robot. For further analysis of non-uniform coverage scenarios, it is assumed that aerial robotic system is used for inspection from a height.

The algorithm presented can be applied to the non-uniform coverage problem. Given a simply closed area, boundaries are approximated by straight lines to form a polygon. The Polygon (P) is triangulated (T_P) such that each triangle has uniform "interestingness". SKC is followed in each of the triangles with iteration determined by order of "interestingness" of the patch. The more is the probability of finding the object of interest, the higher the iteration of SKC used. Each SKC path can have multiple disjoint obstacles, but they need to follow the assumptions to follow the presented strategy.

Every triangle in T_P is to be traveled. A scheme to travel the triangles in order, such that triangles in succession have a side shared, is required. A Dual graph of T_P $(G(T_P))$ is constructed. A Dual graph of triangulation is constructed by joining the nodes of triangular cells, given the triangular cells share side [12]. If $G(T_P)$ is a Linear graph then there is a trivial way of sequencing the triangles. A Linear graph (also known as Path) is a connected graph with exactly two vertices with degree 1 and the remaining vertices with degree 2. Here, it is assumed that $(G(T_P))$ is a linear graph. This is an easy case in which the robotic system starts at one terminal vertex and travels all the vertices until another terminal vertex. However, a more meticulous strategy needs to be planned for traveling between the vertices of the graph if the dual graph is not a linear graph. Consider an example, Area A is triangulated into triangles 1, 2, 3, and 4. The Dual graph of the triangulation is a Linear graph, and the sequence of triangles for traversal is $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, as shown in fig 17.

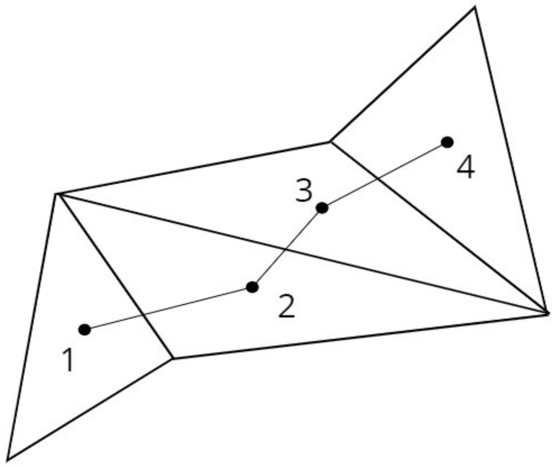


Fig. 17: Dual graph for triangulation of Area A

The order of "interestingness" to be $3 > 2 > 1 > 4$ is assumed. Therefore, the order of iteration of SKC used is also $3 > 2 > 1 > 4$ (fig 18). Shortcut Heuristic (SH) strategy [5] is used (fig 19). In SH strategy, a robotic system visits all the nodes at the lowest level of a patch, then visits the nearest node of the next patch and goes on visiting all the nodes at the lowest level in that patch; this is done recursively to cover the entire region.

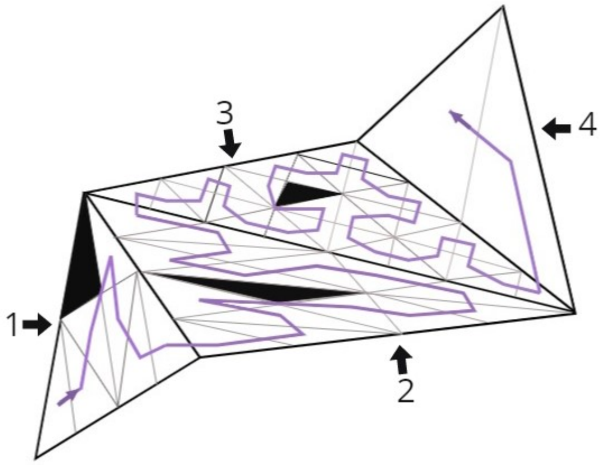


Fig. 18: SKC in non-uniform coverage scenario with obstacles

V. Implementation and Simulation

The presented algorithm can be implemented online since no prior knowledge of the obstacle's location is required. The non-uniform coverage situation needs knowledge about the region, which can be made possible by collaboratively working with an aerial robot. The prescribed algorithm can

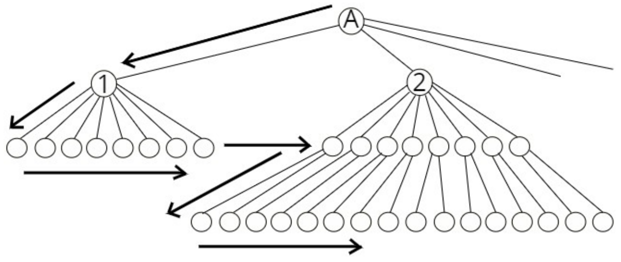


Fig. 19: Shortcut Heuristic

be used for multiple obstacle situations, wherein multiple disjoint obstacles of size enough to fill the entire triangle are present (Except at the first three and last three triangular cells in the area decomposition). The implementation of evasive strategies enlisted in table III are summarised in Figure 20. The strategy presented is summed up in the algorithm 2.

Assume the equations of the boundary of the triangle to be $P(x, y) = 0$, $Q(x, y) = 0$ and $R(x, y) = 0$. Let, $Y = P \times Q \times R$. Function Y is used to ensure that the robotic system does not cross the boundary and goes out of the exploration region.

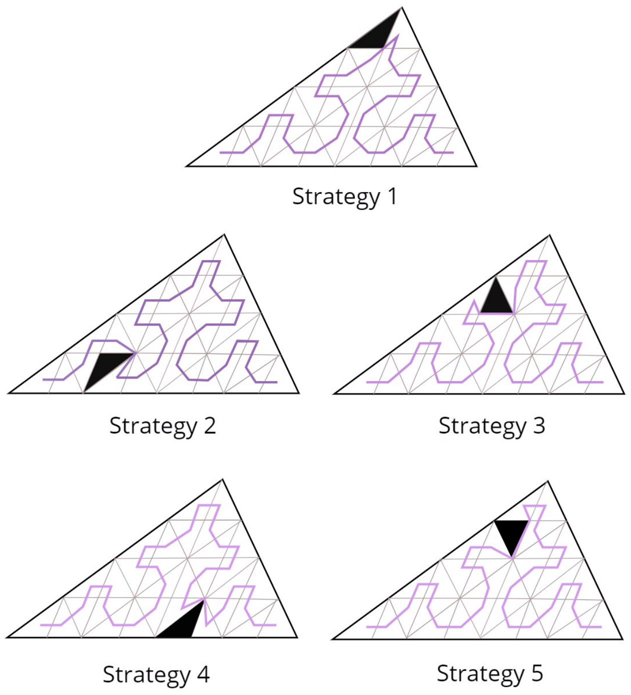


Fig. 20: Implementation of Evasive strategies

VI. Conclusion

The Robotic Exploration problem using the Sierpinski Knoop curve has been considered, a strategy to evade a single obstacle online is presented. Simple geometric properties were used to devise evasive strategies. The algorithm introduced is proven to be exhaustive. The presented strategy can be extended to a simple closed curve with multiple disjoint obstacles. The evasion plan is also shown to be helpful in non-uniform coverage situations. The presented strategy cannot be used for obstacles covering two or more arbitrary triangular cells, but this can be a good starting point, and hence further work demands consideration of arbitrary sized and shaped obstacles.

Algorithm 2 Algorithm for online implementation

```
1: Input : $n$ (Order of SKC), $i + 1^{th}$ (Obstacle detected at $i + 1^{th}$ node), $C$, $B_{i,i+1}$, $B_{i+1,i+2}$
2: Output : Evasive strategy to be followed
3: **if** $Tag(n) = a$ **then**
4:     Follow strategy no.1 from Table III
5: **if** $Tag(n) = b$ **then**
6:     **if** $Y(C) \neq 0$ **then**
7:         **if** $L_2 < L_3$ **then**
8:             Follow strategy no.2 from Table III
9:         **else if** $Y(B_{i,i+1}) \times Y(B_{i+1,i+2}) \neq 0$ **then**
10:            Follow strategy no.3 from Table III
11:    **else**
12:        Follow strategy no.3 from Table III
13: **if** $Tag(n) = c$ **then**
14:    **if** $Y(C) \neq 0$ **then**
15:        **if** $L_4 < L_5$ **then**
16:            Follow strategy no.4 from Table III
17:        **else if** $Y(B_{i,i+1}) \times Y(B_{i+1,i+2}) \neq 0$ **then**
18:            Follow strategy no.5 from Table III
19:    **else**
20:        Follow strategy no.5 from Table III
```

References

[1] H. Sagan, *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[2] M. Bader, *Space-Filling Curves: An Introduction with Applications in Scientific Computing*. Springer-Verlag Berlin Heidelberg, 2013.

[3] S. V. Spires and S. Y. Goldsmith, “Exhaustive geographic search with mobile robots along space-filling curves,” *Collective Robotics*, pp. 1–12, 1998.

[4] I. Maza and A. Ollero, “Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms,” in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 221–230.

[5] S. A. Sadat, J. Wawerla, and R. T. Vaughan, “Recursive non-uniform coverage of unknown terrains for uavs,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1742–1747.

[6] S. A. Sadat, J. Wawerla, and R. Vaughan, “Fractal trajectories for online non-uniform aerial coverage,” in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2971–2976.

[7] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[8] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, “A solution to vicinity problem of obstacles in complete coverage path planning,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 612–617.

[9] S. H. Nair, A. Sinha, and L. Vachhani, “Hilbert’s space-filling curve for regions with holes,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 313–319.

[10] A. A. Joshi, M. C. Bhatt, and A. Sinha, “Modification of hilbert’s space-filling curve to avoid obstacles: A robotic path-planning strategy,” in *2019 Sixth Indian Control Conference (ICC)*, 2019, pp. 338–343.

[11] A. Tiwari, H. Chandra, J. Yadegar, and J. Wang, “Constructing optimal cyclic tours for planar exploration and obstacle avoidance: A graph theory approach,” in *Advances in Cooperative Control and Optimization*. Springer, 2007, pp. 145–165.

[12] M. Van Kreveld, O. Schwarzkopf, M. de Berg, and M. Overmars, *Computational geometry algorithms and applications*. Springer, 2000.