Space Filling Curves on 2D & 3D geometrical shapes: A multi-robot path planning strategy

Teerthaa Parakh*
teerthaaparakh10@gmail.com

Ankita Humne*
ankita.humne@epfl.ch

Arpita Sinha*
arpita.sinha@iitb.ac.in

***Abstract*— A space filling curve maps a one dimensional space to a multi-dimensional space, for instance ,a unit interval to a unit square. Space filling curves are used widely in data storage and retrieval, image processing etc. However, the application in robotics is comparatively less dwelled into. This paper addresses the use of SFCs for finding systematic way of searching, exploring or covering a region of interest while ensuring complete coverage in finite time with any desired resolution. An interesting question that arises is whether the curves could be modified in any way to cover regions of other shapes like polygons, circles and ellipses. The aim is to find out any such possible modifications. We have also looked at coverage of a region with multiple robots travelling along SFCs. A swarm of cooperating robots traversing a SFC provides an energy frugal approach for exploration of a given geographic region, thereby covering a large region, and at the same time provides a collision free method. SFCs also find application in inspection and mapping of structures such as bridges, railways, buildings etc. So, after establishing that simple 2D figures without obstacles can be covered using SFCs, we have looked at ways to cover surfaces of 3D bodies using SFCs. For validation, a simulation based testing has been done in ROS and Gazebo with Turtlebot3 for 2D surfaces without obstacles and Parrot AR.Drone 2.0 for 3D surfaces. A simple position controller is employed to get the desired performance.**

I. INTRODUCTION

A space filling curve or SFC is a way of mapping the multidimensional space into a one dimensional space. It can be thought of as a thread that passes through each cell in the multidimensional space so that every cell is visited exactly once. There exist different space filling curves, each having its own way of mapping to the one-dimensional space. Since SFC provides systematic approach for covering a region, therefor it ensures complete coverage in finite time. Various attempts have been made to use SFCs for motion and path planning of mobile robots. We aim to extend the Hilbert curves and Sierpinski curves to cover geometric shapes beyond squares and right angled isosceles triangles respectively in 2D and 3D. These curves will be the potential paths for robot motion. When there are multiple robots, it is advisable that the robots meet frequently enough for better communication. We design the curves satisfying this condition. The communication is necessary, because if one robot fails, then another robot is required to cover the area, that was supposed to be covered by the failed robot.

*The authors are with the Department of Aerospace Engineering, Indian Institute of Technology, Bombay

II. LITERATURE REVIEW

Several attempts have been made to slightly modify the Hilbert curve to achieve desired results or coverage and also for obstacle avoidance and collision free coverage. **[2]** aims to cover a region efficiently using multiple robots and ensure optimal coverage in finite time with minimal configuration energy. A Hilbert Curve is seen to satisfy their needs when the robots are placed at equal lengths along the curve and given m robots, the region can be searched in time proportional to $\frac{2^{2n}}{m}$. They connect the starting and end points of the curve to the beginning so as to make it a closed curve. This also helps in case any robot dies and the other robots are required to finish its task. The report also mentions that a Hilbert curve eradicates the need of a dynamic collision avoidance. In **[3]**, the authors have tried to modify the Hilbert curve to achieve various combinations of its starting and ending point, inspired by Moore. They have also presented the modified transformations to achieve the given Hilbert-types curves. However, such modifications violate the adjacency condition which says that the adjacent sub-intervals correspond to adjacent sub-squares.In **[4]**, the authors present the implementation of the Hilbert curve to achieve coverage of a region having specific sub-regions of interest. When a region of interest is discovered at the highest point, the robot moves lower where the region is divided into grids of higher resolution to better inspect the region of interest. The coverage of a square by a Hilbert curve depends on the minimum size of the grid cell and the size of the square. To over come this, the paper **[5]** presents addition and modification to the Hilbert curve to cover a square of any region and also rectangles of different sizes. In **[6]**, a Moore's space filling curve is used to achieve surveillance of a given region using a portable mini Unmanned Aerial Vehicle. The task requires detection of wireless devices and since an SFC moves through each point in a square, a Moore curve based flight route can detect all the the devices.Chapter 9 of Michael Bader's Space Filling Curves: An introduction with Applications in Scientific Computing **[7]** speaks about adaptive Hilbert curve orders to cover the boundary of a region of any arbitrary shape. Whenever a given sub-square encounters a boundary, it is further refined until all sub-squares are entirely inside or outside the domain or region whose boundary is to be covered.In **[8]**, a review of the most successful coverage and path planning methods in the past decade is presented. It includes various offline and online methods, 3D coverage methods and also methods for

multiple robots.

The rest of the paper is divided into six sections. Section III introduces Hilbert curve and Sierpinski curve. Section IV presents methods to cover different 2D shapes with single robot using SFCs. Section V extends these methods to multiple robots. Section VI presents methods to cover 3D surfaces using SFCs. Section VII provides algorithm for Hilbert and Sierpinski curve with simulation results from ROS and Gazebo followed by concluding remark and future work in sectionVIII

III. PRELIMINARY ON HILBERT SPACE-FILLING CURVE

A. *Hilbert Curve*

Although, it was Peano who discovered the first space filling curve, it was Hilbert who recognized a general geometrical generating procedure that allowed construction of an entire class of space filling curves named after its creator. The procedure for creating a Hilbert curve can be described as follows:

- I is to be mapped surjectively and continuously to Q. We begin by partitioning I into 4 equal intervals and also divide Q in 4 congruent sub-squares. Then each sub-interval can be mapped to one sub-square each. We repeat the procedure of sub-dividing each sub-interval into 4 sub-intervals and each sub-square into 4 sub-squares until the desired order or resolution is obtained.
- If a square maps to an interval, then its sub-squares must map to the sub-intervals of the interval. This ensures that a mapping of the nth iteration preserves mapping of the (n-1)th iteration
- For an nth order Hilbert curve, I and Q are partitioned into 4n congruent replicas and Hilbert demonstrated that the sub-squares can be arranged so that adjacent sub-intervals correspond to adjacent subsquares with an edge in common.

We define $f_h : I \rightarrow Q$ which maps the interval to the square, such that every parameter t ϵ I corresponds to a unique sequence of nested closed squares that shrink into a point Q. The process of generation of the Hilbert curve becomes easier if the parameter t is represented in the quaternary base as follows.

$$t = 0._4 q_1 q_2 q_3 ... = \frac{q_1}{4} + \frac{q_2}{4^2} + \frac{q_3}{4^3} + ... \tag{1}$$

t lies in the (q1 + 1)th sub-interval of the first partition of I into 4 sub-intervals. Hence, its image lies in the (q1 + 1)th square of the first partition of Q. It lies in the (q2 + 1)th partition of the second partition within the (q1 + 1)th subsquare of the first partition and so on. The following set of transformations were defined by Hilbert which act recursively on the set Q to achieve the desired map

$$T_0 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2}$$

$$T_1 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3}$$

$$T_2 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4}$$

$$T_3 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{5}$$

The map f_h then, can be written as

$$f_h = \lim_{n \rightarrow \infty} T_{q_1} T_{q_2} T_{q_3} ... T_{q_n} Q \tag{6}$$

Note that any finite quaternary represents the starting of an interval and

$$f_h(0._4 q_1 q_2 q_3 ... q_n) = T_{q_1} T_{q_2} ... T q_n \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7}$$

The polygonal line that connects the image points of the sub-interval edges of I of the nth iteration is termed as the nth approximating polygon of the Hilbert curve.

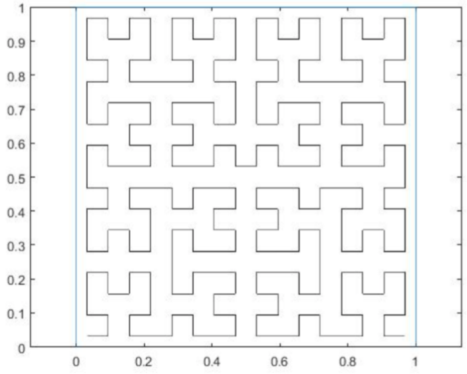

Fig. 1. Hilbert Curve of order 4

B. *Sierpinski Curve*

The Sierpinski curve results from a recursive substructing of a right, isosceles triangle. The original triangle is successively divided into congruent sub-triangles by splitting the hypotenuse. Every t in I corresponds to a unique sequences of nested closed triangles that shrink into a point of T, the image $f_s(t)$. The map is $f_s : I \rightarrow T$. A set of transformations can be specified which act upon the set T recursively to generate the required space-filling curve.

$$T_0 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8}$$

$$T_1 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{9}$$

$$T_2 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{10}$$

$$T_3 \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{11}$$

Given these transformations and the quaternary form of t, the map fs can be defined as

$$f_h = \lim_{n \rightarrow \infty} T_{q_1} T_{q_2} T_{q_3} ... T_{q_n} Q \tag{12}$$

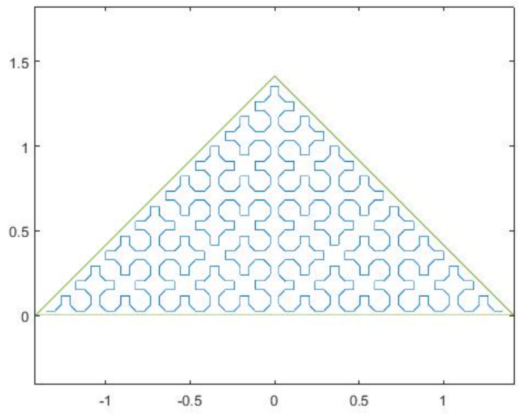

Fig. 2. Sierpinski Curve of order 4

IV. 2D SURFACES WITH SINGLE ROBOT

A. *Rectangle*

In this case we begin by creating a Hilbert curve of size equal to the breadth of the rectangle. For the remaining rectangle, the shorter side is renamed the breadth and a Hilbert curve of the size of the breadth is traversed. The process is continued till the remaining rectangle or square has a size smaller than or equal to the range of the robot. The order is chosen such that the minimum cell size is as close to the range as possible; order = floor(log(b/range)).

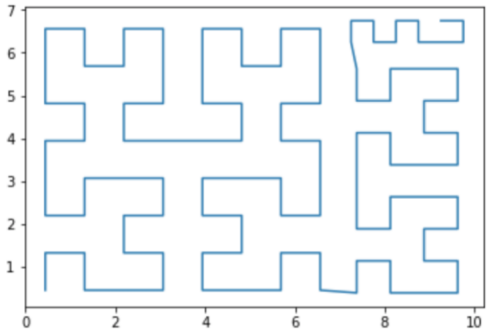

Fig. 3. Hilbert Curve in a rectangle of 7 x 10

B. *Circle*

To cover a circle using a Hilbert curve, a co-ordinate transformation can be done from a unit square to a unit circle. A point in the square is represented by (x,y) and its map in the circle is represented by (u,v). The following mappings are used.

1) FG-squircular mapping

$$u = \frac{x\sqrt{x^2+y^2-x^2y^2}}{x^2+y^2}, v = \frac{y\sqrt{x^2+y^2-x^2y^2}}{x^2+y^2} \tag{13}$$

2) Elliptical grid mapping

$$u = x\sqrt{1-\frac{y^2}{2}}, v = y\sqrt{1-\frac{x^2}{2}} \tag{14}$$

These mappings are described in [9].

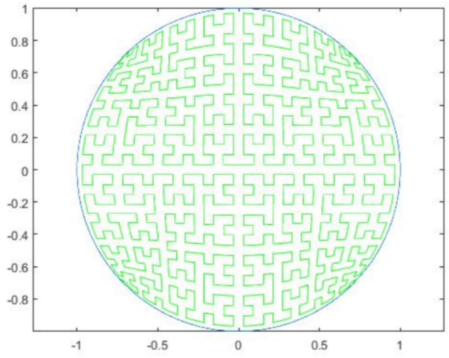

Fig. 4. Hilbert curve in a unit circle

C. *Ellipse*

To cover an ellipse with minor axis a and major axis b we can simply use the coordinate transformation for a circle and scale the coordinates as $(x, y) \rightarrow (ax, by)$.

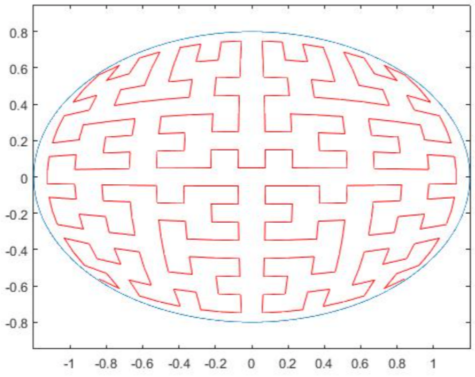

Fig. 5. An ellipse covered by a Hilbert curve

D. *Regular Polygon*

To make use of a space filling curve to fill any n-sided polygon, we use Sierpinski curves which use triangular grids. This requires dividing the polygons into triangles that can then be covered by single or multiple robots tracing Sierpinski curve based paths.

When we are dealing with polygons, it may not always be possible to subdivide them into right isosceles triangles, as is generally required by Sierpinski curves. To use the curve in polygons, we can slightly modify the splitting of triangles such that the triangle is divided by drawing a median to the requisite side depending on which way we want the robot to travel. This way we will not get a right isosceles triangular grid but an equal area triangle grid.

For a regular polygon, when we have a single robot we can have triangles subtended at the centre which can then be covered by the robot while maintaining continuity of path.

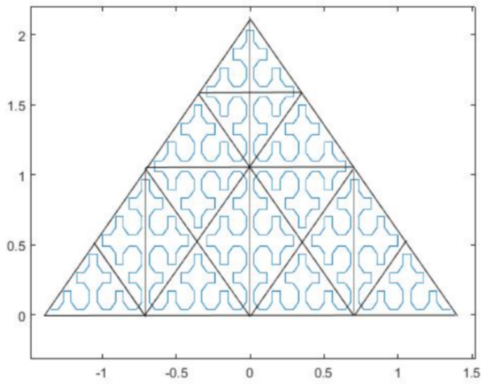

Fig. 6. Sierpinski curve in a skewed triangle

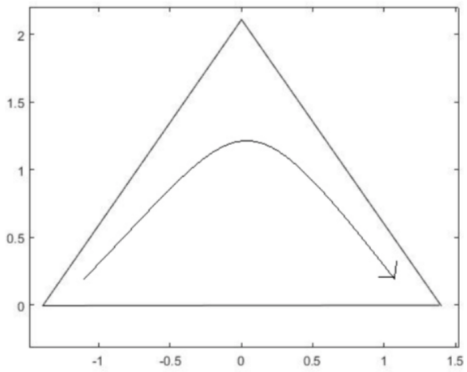

Fig. 7. Simplified representation of Sierpinski curve showing the direction of motion

V. 2D SURFACES WITH MULTIPLE ROBOTS

While working with multiple robots, we wish to ensure that the robots come in frequent contact which can help to:

- Detect deviations from planned path.
- Check if any robot has failed and ensure at least one back up agent is available.
- Transfer information about the environment.

Multiple robots configuration needs communication between robots or robot and a central control system, in the method proposed, long range communication is avoided as the robots come within each other's range often. A space filling curve also provides a collision free path for each robot.

A. *Rectangle*

A good way to ensure communication between robots is to make them trace water images or mirror images of each other, shown in Figure 9. As the robots start from the position marked by the arrows, they meet at the sides marked in green as they reach those positions together, assuming that they start together at the same speed. As the robots trace the water images of each other as in Figure 9, they tend to finish at adjacent points. If one robot dies mid way and does not reach the cell or any of the green regions, the other can detect this and go on to cover the region that was to be covered by the dead robot. If the robots move along the curves that are mirror images of each other, they come in each other's range along the green lines, when they can exchange information.

Fig. 8. Motion along Sierpinski curves in pentagon

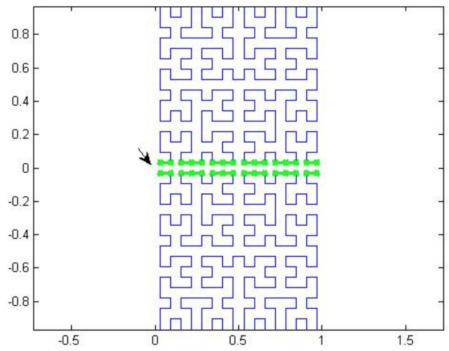

Fig. 9. Hilbert curves as water images of each other.

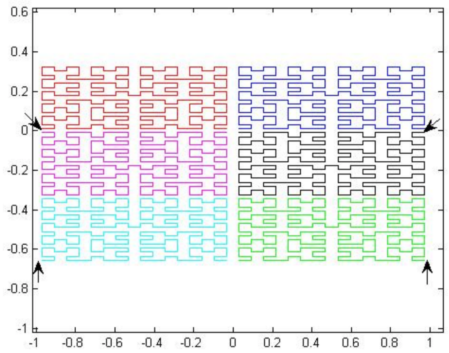

Fig. 10. Rectangular Region to be covered by 6 agents.

B. *Circle*

To cover a circle of radius R using m robots, we divide the circle into sectors with each sector angle of 360/m degrees. We then cover a triangle of two sides of length R and an angle of 360/m degrees between them using a Sierpinski curve and use coordinate transformations, to map the paths into the sector

C. *Regular Polygons*

If n is the number of sides of the polygon and m is the number of agents, then we first subtend triangles at the centre

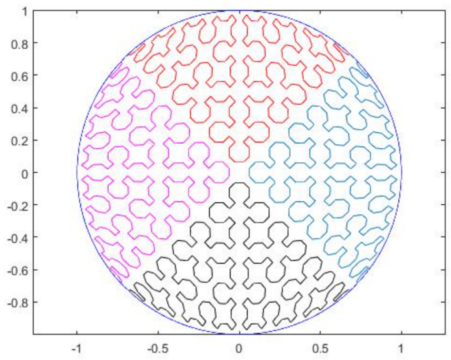

Fig. 11. A circle covered by 4 agents.

from each side and the divide the polygon into lcm(n,m) triangles such that each triangle is divided into lcm(n,m) equal triangles which are equally divided among the agents. An assumption here is that the robots take equal time to cover triangles of equal areas and so the robots meet occasionally at the centre of the polygon.

Fig. 12. A hexagon (n=6) covered by 4 agents (m=4).

VI. SURFACES OF 3D OBJECTS

A. *Cylinder*

A circular cylinder can be formed by folding a rectangle such that two of its sides coincide. We have seen how Hilbert curves can be used to cover a rectangle. For a cylinder of radius r and length l, we need to cover a rectangle of length $2pi$r and height l as given in section IV(A). For multiple agents, the rectangle should be covered using method from Section V(A).

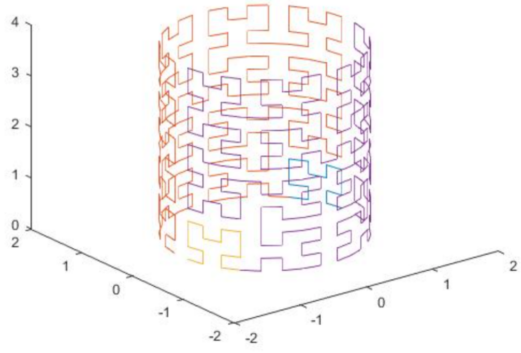

Fig. 13. Hilbert curve on a cylinder of radius 1.27 and height 8 units with single robot.

B. *Cone*

To cover a cone of base radius r and height h, using Hilbert curve, we first cover a circle of radius r using Hilbert curve as described in Section IV(B) and then map the points from the circle to the cone such that

$$x_{cone} = x_{circle}, y_{cone} = y_{circle}, z_{cone} = h - \frac{h\sqrt{x^2+y^2}}{r} \tag{15}$$

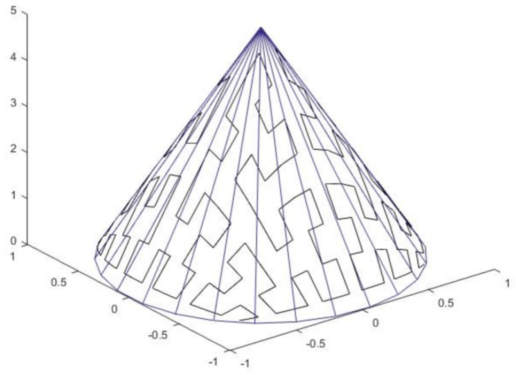

Fig. 14. Cone covered using Hilbert curve.

VII. IMPLEMENTATION AND SIMULATIONS

We have tested the algorithms using ROS, an open source library which provides a communication infrastructure for programs and processes .ROS also provides open source libraries and tools which are helpful in software development of robotic applications. For simulating robots we have used Gazebo, a simulation software corroborated to ROS, for providing a virtual environment, thus making it possible to conduct experiments without the dependence on a physical platform in a faster and inexpensive way.For simulating 2D figures we have used Turtlebot3 which is a differential drive mobile robot and its movement is based on the relative rate of rotation of the two independently driven wheels placed on its either side. We have deployed in-place rotation of the bot, which has increased the accuracy upto 1 % of cell size in position .

For "simulating 3D figures" we have used parrot AR Drone. It uses inertial sensors (accelerometer and gyrometers) for stabilization (low level control) along with downward looking camera which provides velocity estimates of the quadrotor in X-Y plane of the body-fixed frame.

Position controller is defined by:

$$velocity = constant(DesiredPosition - CurrentPosition) \tag{16}$$

$$AngularVelocity = constant(DesiredAngle - CurrentAngle) \tag{17}$$

If there were curved paths, in order to have more accuracy, it would have required implementing some complex controller, leading to increased computational time and hence a decrease in the efficiency of exploration task. Here, using

hilbert curves proves to be advantageous : in hilbert curve, path to be traced consists of straight lines and position controller works very well for this purpose.

Algorithm 1 Implementation Of Hilbert Curve

procedure HILBERT($order$)
  $interval = 0 + 0.5/4^{order} : 1/4^{order} : 1-.5/4^{order}$
  $tQuat = DecToQuat(interval, order)$
  $k = size(tQuat)$
  for `i<length(interval)` **do**
    x=[0,0]
    for `j<k` **do**
      $x = H(tQuat(i, end-(j-1)), x)$
    $curve(i,:) = x^t$
  $plot(curve(:,1), curve(:,2)$
procedure DECTOQUAT(x,N)
  q = zeros(length(x),N+1)
  for `i<length(x)` **do**
    for `j<N+1` **do**
      $x(i) = x(i) * 4$
      $q(i,j) = floor(x(i))$
      $x(i) = x(i) - q(i,j)$
  return q
procedure H(i,P)
  if $i == 0$ **then**
    $Pnew = [0, 1/2; 1/2, 0]P$
  else if $i == 1$ **then**
    $Pnew = [0, 1/2; 0, 1/2]P + [0; 1/2]$
  else if $i == 2$ **then**
    $Pnew = [1/2, 0; 0, 1/2]P + [1/2; 1/2]$
  else if $i == 3$ **then**
    $Pnew = [01/2; 1/20]P + [1; 1/2]$

Algorithm 2 Implementation Of Sierpinski Curve

procedure HILBERT(n)
  $a = 1 + 1i$
  $b = 1 - 1i$
  $c = 2sqrt(2)$
  $z = c$
  for `k<n` **do**
    $w = 1iz$
    $z = [z + b; w + b; aw; z + a]/2$
  $z = [z; 1iz]$
  plot(z)

A. *Hilbert Curve with single Turtlebot*

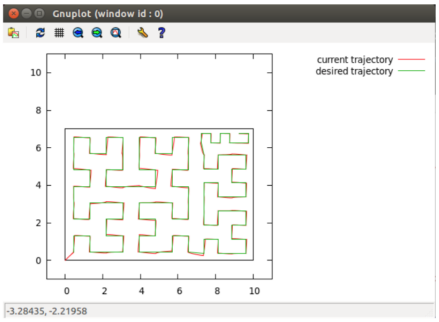

Fig. 15. A Hilbert curve of order 4 in a square of length 8.Green line shows the desired trajectory and red line shows actual trajectory. Bot starts from (0,0)

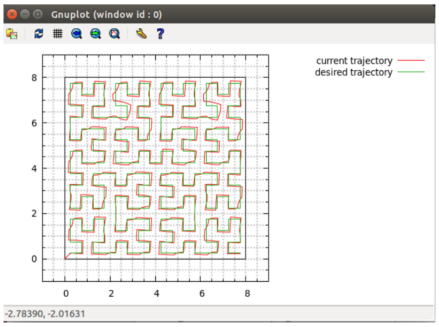

Fig. 16. Turtlebot travelling along hilbert curve .Here range sensor is taken to be .5m. So first it travels along hilbert of order 3, then order 2,then again order 2 , then order 1 ,again order 1 and it completes its path with order 1.

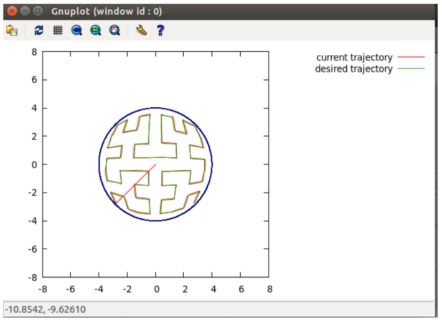

Fig. 17. Turtlebot travelling along hilbert of order 2 inside a circle of radius 4. Green line shows the desired trajectory and red line shows actual trajectory. Bot starts from (0,0).

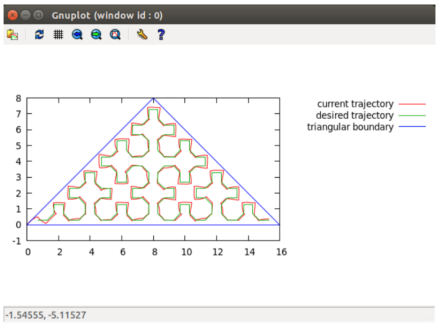

Fig. 18. Turtlebot travelling along sierpinski curve of order 3 inside a right isosceles triangle with hypotenuse 16m in length. Green line shows the desired trajectory and red line shows actual trajectory. Bot starts from (0,0).

B. *Hilbert Curve with multiple Turtlebots*

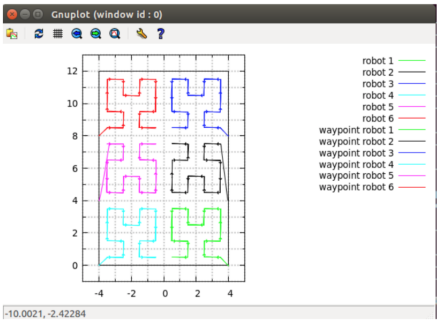

Fig. 19. Region to be covered by 6 agents. Arrows shows the direction of movement of turtlebots.

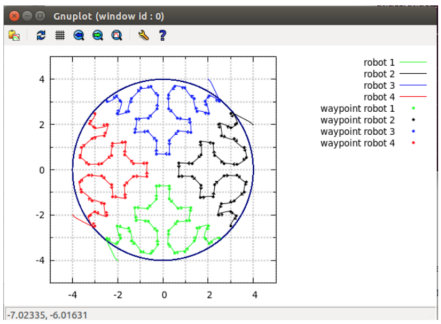

Fig. 20. Region to be covered by 4 agents. Arrows shows the direction of movement of turtlebots.

C. *Hilbert Curve on 3D Surface*

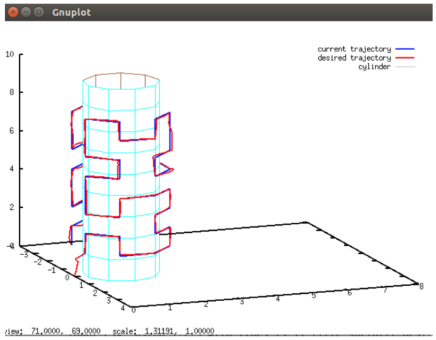

Fig. 21. Region to be covered by Parrot AR.Drone. Hilbert Curve of order 3 on cylinder .

VIII. CONCLUSIONS AND FUTURE WORK

A space filling curve allows for coverage of a region with any desired resolution. We have devised ways of covering 2D and 3D surfaces using space filling curves. We have modified the curves to cover shapes for which they were not originally designed. We have also worked on the cooperative motion of multi-robot systems which involves periodic meeting of robots while ensuring the no-collision condition.The method of coverage of simple geometrical shapes can be further used to cover slightly complicated figures and 3D surfaces which are a combination of these. These algorithms can be further modified to examine unknown or dynamic environments with obstacles . And as we have looked at very basic geometrical shapes. Following this, we can look at more complicated surfaces such as those of a bent rod, more complicated 3D surfaces which are a combination of multiple 3D surfaces.

REFERENCES

[1] Hans Sagan. Space-filling curves. Springer Science Business Media, 2012.

[2] Shannon V Spires and Steven Y Goldsmith. Exhaustive geographic search with mobile robots along space-filling curves. In Collective robotics, pages 1–12. Springer, 1998.

[3] Nicholas J Rose. Hilbert type space filling curves. North Carolina State University, Chicago,2001.

[4] Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. Fractal trajectories for online non- uniform aerial coverage. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 2971–2976. IEEE, 2015.

[5] Jian Zhang, Sei-ichiro Kamata, and Yoshifumi Ueshige. A pseudo-hilbert scan algorithm for arbitrarily-sized rectangle region. In Advances in Machine Vision, Image Processing, and Pattern Analysis, pages 290–299. Springer, 2006.

[6] Zhongli Liu, Yinjie Chen, Benyuan Liu, Chengyu Cao, and Xinwen Fu. Hawk: an unmanned mini-helicopter-based aerial wireless kit for localization. IEEE Transactions on Mobile Com- puting, 13(2):287–298, 2014.

[7] Michael Bader. Space-filling curves: an introduction with applications in scientific computing, volume 9. Springer Science Business Media, 2012.

[8] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. Robotics and Autonomous Systems,61(12):1258–1276, 2013.

[9] Chamberlain Fong. Analytical methods for squaring the disc. arXiv preprint arXiv:1509.06344, 2015.

[10] H. Dai and H. Su, “On the locality properties of space-filling curves,” Algorithms and Computation, vol. 2906, pp. 385–394, 2003.
[11] J. Kuffner and S. M. LaValle, “Space-filling trees: A new perspective on motion planning via incremental search,” Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), 2011.