Journal of Intelligent and Robotic Systems manuscript No.
(will be inserted by the editor)

Using Fractal Dimension to Assess Robot Operator Skill in a Search Task

Jeffrey Craighead

Received: 21 March 2010 / Accepted: 21 December 2010

Abstract This paper discusses a new real-time fractal path analysis (RTFPA) algorithm and its use in assessing the skill of robot operators. Twenty-five volunteers participated in an experiment to evaluate the use of the RTFPA algorithm as a metric for robot operator skill in a search task. The algorithm was used to estimate the fractal dimension of a path taken by a simulated, tele-operated Inuktun Extreme VGTV within a game-based training environment. The results show that within the sample population there exists a curvilinear relationship between fractal dimension and score. This relationship seems to indicate that when used along with score, fractal dimension can be used as a measure of an operator's search strategy and ability to maintain situational awareness. An additional analysis confirms the results of prior work showing that there is no evidence of a relationship between fractal dimension and task completion time. Additionally, a detailed description and pseudo code for the RTFPA algorithm are presented and the accuracy of RTFPA is compared to existing fractal path analysis algorithms.

Keywords Fractal dimension · Human-robot interaction · Teleoperation · Training · Video games

Mathematics Subject Classification (2000) 68T40 · 68U20 · 68W27

1 Introduction

When training robot operators for search and rescue tasks it may be beneficial to have a measure of the effectiveness of a trainee's search strategy. A typical metric for

Jeffrey Craighead, PhD
James A. Haley Veterans Hospital
HSR&D/RR&D Center of Excellence
8900 Grand Oak Circle
Tampa, Florida 33637
USA
Tel: +001-813-558-3903
Fax: +001-813-558-3994
E-mail: Jeffrey.Craighead@va.gov

measuring an operator's skill and/or search strategy is simply the number of objects identified while conducting a search (a score). While this is a useful metric, it is naive in that it does not indicate how a particular search strategy covers an open space. This paper proposes the use of the fractal dimension (D) of the robot's path as an indicator of search strategy and ability to maintain situational awareness.

The term fractal as it appears in this work is defined as an abbreviation of "fractional dimension", coined by Benoit Mandelbrot in his 1967 article "How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension" [8]. A fractional dimension is used to describe the variation in a surface or line when examined at varying spatial scales (magnification levels). A seemingly smooth surface may be very rough when magnified or a rough surface may also appear similarly rough on a smaller scale (self similarity). For instance, Mandlebrot's article posits that the coastline of Britain will be significantly longer if measured in millimeters versus kilometers due to the self-similar nature of coastline. This intuitively makes sense, by using millimeters as the unit of measure significantly more of the variation in the surface can be included in the measurement. The fractal dimension is computed by comparing the measurement of a line or surface at multiple scales. A truly straight line has a fractal dimension of 1, while a line that crosses a plane enough times to completely cover the surface (i.e. follows a Brownian motion) has a fractal dimension of 2, thus when describing the fractal dimension of a line $1 <= D <= 2$. Fractal dimensions can also be applied to the three dimensional analysis of surfaces such that the fractal dimension $2 <= D <= 3$. Figure 1 shows an example of several paths and the associated fractal dimensions. The limits only hold for fractals that are generated to be perfect recursive patterns and that the step length is a multiple of both minimum and maximum dividers. It should be noted that the estimated fractal dimension can increase past the theoretical upper limit and dip below the lower limit depending on the size difference between the minimum and maximum divider; the greater the difference, the more likely that a very tortuous path will yield $D > 2$.

This paper discusses the implementation of a real-time fractal path analysis (RTFPA) algorithm which was used to examine the relationship between fractal dimension and a robot operators skill level in a training video game. It was hypothesized that a more skilled operator would have higher situational awareness and a better understanding of how to navigate the robot in a given environment; and that a skilled operator would cover more of the search area within the given time limit. An experiment was conducted in which 25 participants played a robot training game (SARGE) [4] for one hour. During their training the RTFPA algorithm recorded their paths and fractal dimensions. The results show that D is curvilinearly related to a participants score, that D seems to indicate how well an area was searched and that D is not correlated with task completion time.

2 Related Work

Fractal dimension has been used recently to study the behavior of animals [7, 9, 15]. By classifying their path tortuosity (how convoluted the animal's movement is over time) with a fractal dimension it becomes easier to compare the territorial behaviors of different species and individual animals. Voshell, Phillips, and Woods [11, 12, 13] used path tortuosity measured by fractal dimension to compare the effectiveness of two robot user interfaces. Bae, Voyles, et al. [1, 14] used fractal dimension to compare

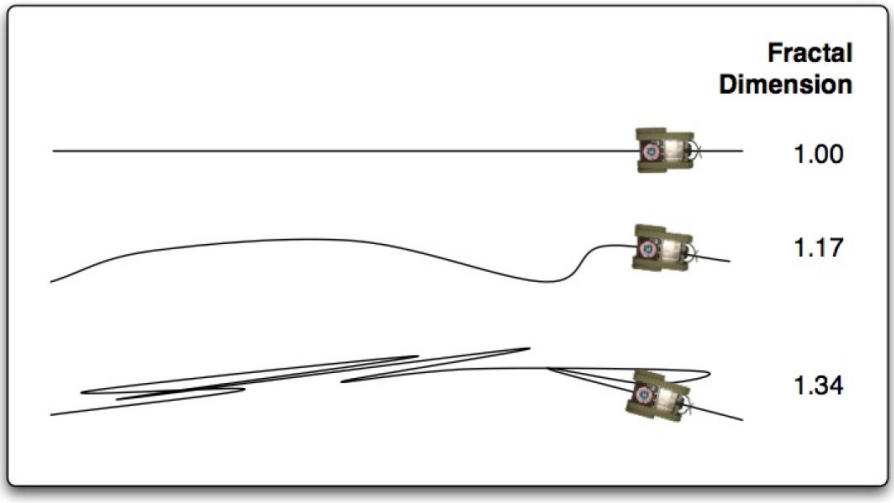

Fig. 1 Example of Path Tortuosity. This figure was adapted from [11].

the effectiveness of standard computer input devices to a wearable glove input device. In the Voshell studies, operators of a robot with a folded perspective display were shown to navigate the robot through a simulated environment towards a specific point more smoothly than operators with a standard video display. The smoother paths resulted in lower fractal dimension values. The papers suggest that the lower fractal dimension (tortuosity) of the path indicates that the operators had higher situational awareness when using the folded perspective display, thus were able to reach the goal quicker with less searching. The papers did not indicate what spatial scales were used when calculating the fractal dimension of a path, nor did they state the number of participants involved in the study. The Bae and Voyles studies used a total of 43 participants and show that path tortuosity as measured by fractal dimension tends to be positively correlated with the task completion time metric for a specific movement task. The participants were directed to drive a robot between two visible points without hitting an obstacle placed between the two points using the various input devices. This is in contrast to the work presented in this paper which directed the participants to search an unknown area for important objects.

A similar measure of path complexity, the Lyapunov exponent, has been used to show how path complexity is related to the performance of humans searching a maze [3]. While fractal dimension is a scaleless measure of path complexity in the spatial domain, the Lyapunov exponent is a scaleless measure of the rate of divergence of two trajectories in a dynamical system. Generally the Lyapunov exponent is used to assess the predictability of a system, in this case it is used to measure path tortuosity in the time domain (i.e. measure if the path likely to stay on the same course). A positive Lyapunov exponent indicates a chaotic path while a negative exponent indicates a smooth path. Clarke, et al.'s experiment involved 120 participants who wore tracking equipment and searched a 5m x 7m maze constructed in a lab. The results showed that the participants who's paths were most chaotic (positive Lyapunov exponent) were

able to find all of the hidden items in the maze in a much shorter period of time than the participants who's paths were smooth (negative Lyapunov exponent).

At first glance it seems that the works of Voshell and Clarke contradict each other, one shows that a less complex path is related to higher performance and the other shows that a more complex path is related to higher performance. However, the works actually support each other. In the Voshell study the operators were directed to go from point A to point B within the environment; the Clarke study asked participants to completely search a maze for small tags. These goals are two sides of the same coin, one rewarded traveling in a smooth path (direct travel between points) while the other rewarded chaotic movement (complete coverage in a search).

3 RTFPA Algorithm

Fractal path analysis was chosen over the Lyapunov exponent because it is conceptually simpler yet measures the same physical property, path tortuosity. As noted in Section 1 Mandelbrot introduced a fractal path analysis algorithm in 1967 which is commonly referred to as the divider method. Since then, there have been several works that have made incremental improvements to the estimates generated by the divider method. In the following subsections the reader will be introduced to the divider method (Section 3.1) before discussing improvements to the divider method that have been introduced in the literature (Section 3.2). Finally a the reader will find a discussion of the RTFPA algorithm (Section 3.3), a pseudo code example (Section 3.4), a discussion of the author's implementation (Section 3.5), and a step-by-step example (Section 3.6).

3.1 Divider Method

The following subsection describes the original divider method as proposed by Mandelbrot to give the reader knowledge of the algorithm on which RTFPA is based. The easiest way to understand this method is to imagine you are given a sheet of paper with a squiggly line drawn on it. Imagine you are also provided a dividing compass (see Figure 2) marked such that you can adjust the radius to a known "divider size" (also called "spatial scale"). Your task is to estimate the length of the squiggly line using the compass. You do this by placing one end of the compass at the start of the line and rotating it around until the other end intersects a point further down the line. This point of intersection becomes the new starting point. You continue to walk the compass in this manner, summing the radii, until you reach a point where the compass no longer intersects the line. The sum of the radii is the estimate of the path length.

To calculate the fractal dimension (D) of the line you must estimate the length of the squiggly line at least two times, using different spatial scales. Intuitively, the estimate obtained using the shorter spatial scale is more accurate and correspondingly longer spatial scales tend to be less accurate, underestimating the actual path length. It is this error in the estimate that allows us to calculate D. After you obtain two or more path length estimates, plot the estimates and corresponding spatial scales on a log-log graph. Next, calculate the slope of the log-log plot using Equations 1,2 and 3. Finally, the estimated D value for the line is given by Equation 4. Figure 5 shows how dividers of different sizes are used to estimate the length of a line and provides an

example of path length underestimation. It is clearly seen in the figure that a larger spatial scale can significantly underestimate the length of a path when compared to a smaller spatial scale.

$$\delta Distance = log(shortEstimate) - log(longEstimate) \tag{1}$$

$$\delta Divider = log(shortDivider) - log(longDivider) \tag{2}$$

$$slope = deltaDistance/deltaDivider \tag{3}$$

$$D = 1 - slope \tag{4}$$

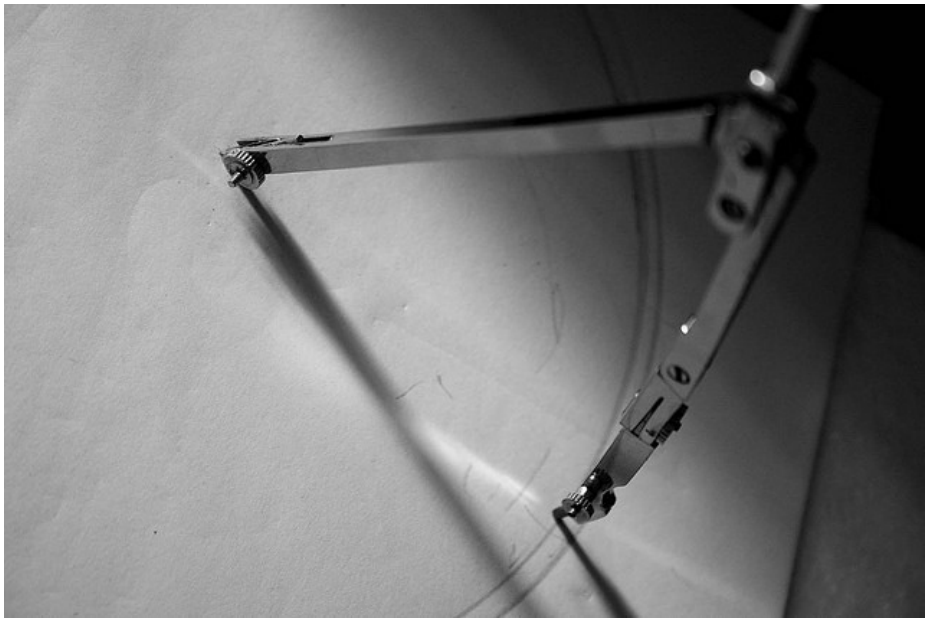

Fig. 2 A drafting compass. Photograph provided by vivekrajkanhangad via flickr.com under a Creative Commons license.

3.2 Improving the Accuracy of the Divider Method

With [15] introduced the Averaging method in 1994, which averages the D values found using the divider method from several different starting locations along a path. For example, if the path consisted of a series of waypoints visited by a robot, the Averaging method would run the divider method over the path starting from the first, second and third waypoints and then return the average of those results.

Nams [9] has made several contributions, including two methods he termed DMean and AdjustedD. DMean performs one forward path analysis and one reverse path

analysis using the divider method and averages the results. The reverse path analysis simply uses the final point of the path as the starting point and walks towards the first point of the path. The AdjustedD method can be used in conjunction with any divider based analysis. The AdjustedD method adds the remaining distance from the last intersection of the walk to the last point on the path. This extra step slightly enhances the path length estimate. In the same article Nams showed that both With's Averaging method as well as the DMean and AdjustedD methods provide better estimates of D than the original implementation of the divider method. A combination of DMean and AdjustedD provided the most accurate estimate and least variable estimates.

3.3 Discussion of RTFPA

The RTFPA algorithm has two distinct advantages over the previously published algorithms and applications. The primary advantage is that it calculates D in real time with $O(1)$ space complexity. The only data required by the algorithm is the previous location of the object being tracked, number of points in the path, the current path length, and several additional placeholder variables are also kept in memory, Table 1 in the following section provides a description of each variable used in the algorithm. This is in contrast to Nams' Fractal application and DMean algorithm which must keep the entire path in memory so that a reverse traversal can be performed. If DMean were to be used in a real time system it would have $O(n^2)$ time complexity, as every new point would require the entire path to be traversed again in the reverse direction. As the path length grows, updating D with every new point would quickly become too costly to be performed in real time.

To make up for the accuracy lost by doing a forward only traversal, the RTFPA algorithm makes use of a step-wise feedback loop to adjust the divider sizes that will be used for the future points of the path. Remember, the fractal dimension of a path is basically the slope of the line between plotted points of the estimated path length. The idea behind adjusting the divider sizes is to increase the accuracy of the path length estimate. RTFPA calculates two divider sizes using a user specified minimum and maximum multiplier value and the mean distance between points on the path. The mean distance between points is the actual mean distance, as calculated using a GPS or other sensing system, not an estimate. These divider sizes are adjusted with every new point on the path. In practice this allows RTFPA to have a much more accurate estimate of the path length using dividers that are a fixed multiple apart. By estimating a suitable divider size, RTFPA can be used for people, animals, and vehicles without any user input.

Preliminary work has shown this method to be superior to DMean in most cases. Figure 3 shows the advantage of RTFPA's automatic adjustment of spatial scales. Each of the values on the horizontal axis belong to one of 16 zig-zag paths that were generated to have a known D value. These paths were analyzed using both RTFPA and DMean algorithms. The RTFPA analysis was performed using the tracking system implementation as explained in Section 3.5 while the DMean analysis was performed using Nams' "Fractal" application. DMean was tested with various ranges of spatial scales, number of dividers and with the automatic scale estimation enabled. The key indicates the algorithm, spatial scale settings and number of dividers used for each test. For example: MeanD0.5-10:3Div indicates the DMean algorithm with the spatial scales set to 0.5 and 10 using 3 dividers. RTFPA does not indicate number of dividers

or spatial scales since RTFPA only uses 2 dividers and automatically estimates the spatial scales.

To generate the paths used for this test, Equation 5 was used which was originally presented by Mandelbrot in [8]. Given a line segment of fixed length, one can divide the segment into a number of pieces, this is the *ratio*. Each of these pieces has the same fractional length of the original line segment, placed end to end they create a line segment equal to the original. However, if additional pieces of the same fractional length are added the line segment grows in length. The number of pieces in the new line is defined by *segments*. If the start and end of the line segment are fixed on a plane then, to accommodate the additional pieces, the line must become more tortuous. Figure 4 shows the generation of a path with a ratio set to five (5) and segments set to six (6).

$$D = log(segments)/log(ratio) \tag{5}$$

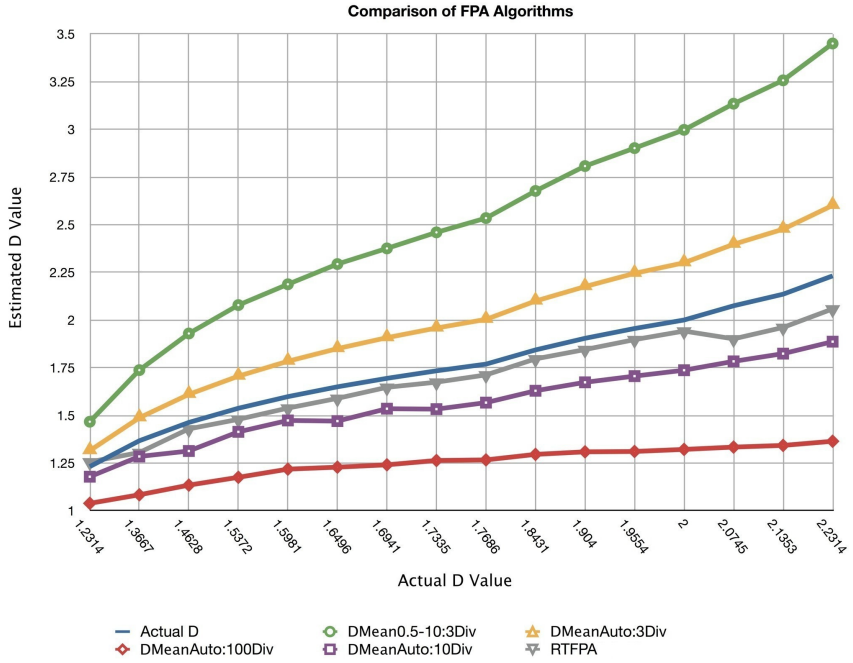

Fig. 3 A comparison of the accuracy of RTFPA vs. Nams' DMean as implemented in his "Fractal" application. For the range of paths tested, the estimate generated by RTFPA (inverted triangle) is closest to the actual D of the path (no marker).

3.4 RTFPA Pseudo Code

The following pseudo code example shows the basic implementation for RTFPA. Variables are defined in Table 1. Because it is trivial to extend this code to average two

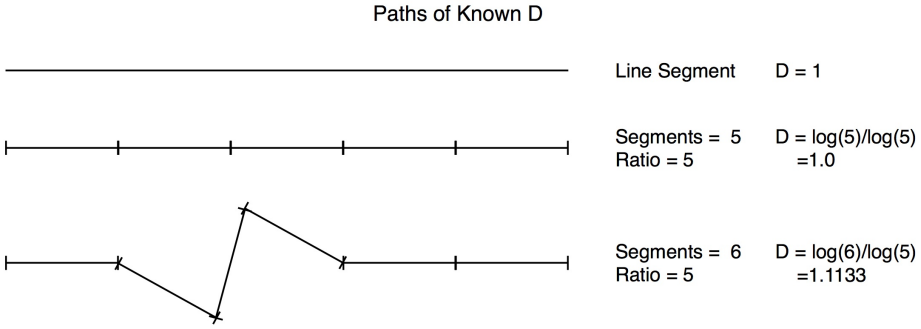

Fig. 4 An example of generating paths using Equation 5.

or more D estimates and to enhance readability, the pseudo code example does not include the steps needed to implement With's Averaging method. The example code is broken into four sub components. In the first block, the global parameters that define the two multiplier values are set, 0.5 and 10 have proved to be good choices on several studies.

The second block shows the data structure that holds the intermediate variables needed as the algorithm walks along the path. In this example -1 is used to indicate an initial value, however this is implementation dependent, there is no significance to this choice.

The third block defines the function that actually calculates D. HandleNewPosition takes a new position as an argument. From there the code first checks if the new position is the first position on the path. If it is the first position in a path, the position and number of steps values are set appropriately and HandleNewPosition returns, waiting for a second position. Once a second position is passed to HandleNewPosition the distance between this second position and the first position is checked to ensure that duplicate position readings are not counted. Assuming the position is a valid new position, the number of steps in the path is updated, followed by the actual path length as determined by summing the distance between readings. These two values allow the divider sizes used during the walk along the segment to be calculated. Once these values are calculated, WalkPath is called which estimates the distance from the end of the last walk to the new position. Following the path length estimation, the slope of the Log-Log plot of the minimum and maximum path length estimates is found. This value is then plugged into Equation 4. Finally, the position variable is updated with the new position.

The fourth block defines the WalkPath function that is called from HandleNew-Position. This function handles the walk from a starting point to an end point using a divider of a specified radius. The function is a simple while loop that performs successive line/sphere intersections and updating the starting point value as the walk continues. The walk terminates when the distance from the starting point to the end point is less than the defined radius. The code for the line/sphere intersection has been omitted as this is fairly trivial to implement and can be found online. It should be noted that the starting position is passed to this function by reference, thus the position updates are propagated back to the *minSphereCenter* and *maxSphereCenter* variables that are defined in the second block.

```
/******** SET GLOBAL PARAMETERS ********/
```

```
double minMultiplier = 0.5
double maxMultiplier = 10


/******** INITIALIZE VARIABLES ********/
point3d minSphereCenter = {-1,-1,-1}
point3d maxSphereCenter = {-1,-1,-1}
point3d position = {-1,-1,-1}
double minPathLength = -1
double maxPathLength = -1
double realPathLength = -1
double minStepSize = -1
double maxStepSize = -1
double meanStepSize = -1
double fractalD = -1
int numberOfSteps = -1


/******** HANDLE NEW POSITION ********/
function HandleNewPosition(point3d newPosition)
    if (numberOfSteps < 0)
        position=newPosition
        minSphereCenter = position
        maxSphereCenter = position
        numberOfSteps++
        return

    double distanceToLastPosition = Distance(position,newPosition)
    if(distanceToLastPosition == 0)
        return

    numberOfSteps++
    realPathLength += distanceToLastPosition
    meanStepSize = realPathLength / numberOfSteps
    minStepSize = meanStepSize * minMultiplier
    maxStepSize = meanStepSize * maxMultiplier

    minPathLength += WalkPath(minSphereCenter,newPosition,minStepSize)
    maxPathLength += WalkPath(maxSphereCenter,newPosition,maxStepSize)

    double logPathLengths = Log10(minPathLength) - Log10(maxPathLength)
    double logDividerSizes = Log10(minStepSize) - Log10(maxStepSize)
    D = 1.0 - logPathLengths / logDividerSizes;

position = newPosition


/******** WALK ALONG SEGMENT ********/
function WalkPath( start byref,end,radius)
```

```
double distance = 0
while(Distance(start,end) > radius)
    point3d intersection = LineSphereIntersection(start,end,radius)
    if(intersection != {0,0,0})
        distance += radius
        startPoint = intersection
return distance
```

Table 1 Variable descriptions for the following pseudo code.

Variable Descriptions	
Name	**Description**
minMultiplier	The user set value used to determine the minimum divider size.
maxMultiplier	The user set value used to determine the maximum divider size.
minSphereCenter	The position of the minimum divider as it walks the path.
maxSphereCenter	The position of the maximum divider size as it walks the path.
position	The most recent position handed to RTFPA (the current end of the path).
numberOfSteps	The number of steps taken from the start of the path. This is not the number of position updates, updates for which the position does not change are not counted as a step.
minPathLength	The path length as estimated using the minimum divider.
maxPathLength	The path length as estimated using the maximum divider.
realPathLength	The actual path length. Calculated by summing the distances between position updates.
meanStepSize	The realPathLength divided by the numberOfSteps.
minStepSize	The meanStepSize multiplied by the minMultiplier.
maxStepSize	The meanStepSize multiplied by the maxMultiplier.
fractalD	The current estimate of the fractal dimension of the path.

3.5 RTFPA Implementation & Benchmarking

The RTFPA algorithm as described in this article was implemented by the author in C#. Two applications have been developed that make use of this code. One application is the SARGE simulator described in Section 4, the other application is a tracking system that monitors the locations of residents within an assisted living facility (ALF). RTFPA is not built upon any pre-existing code. To the authors knowledge, there is no open code base for fractal path analysis, however Nams provides a closed source application [10] for fractal path analysis. The most current C# code for the RTFPA algorithm is available from the author upon request. Unmaintained RTFPA code can be obtained by downloading SARGE from the repository on SourceForge.com.

To benchmark the RTFPA code the author used the ALF position monitoring application which has the capability to process position log files from a previous monitoring application. A function call timer was added to the relevant class within this application such that the time required to process a new location reading could be measured. The timer used the .NET System.DateTime.Now.Ticks property (each Tick represents 100 nanoseconds) to record the system time just before the function call for

RTFPA to handle a new reading and just after that function returns. By placing the timer at this point in the code, disk access time is removed from consideration.

Using a 2010 15" 2.66 GHz Core i7 MacBook Pro with 8GB of memory the application processed a 197MB log file containing over 4 million entries, each representing a location. The average time required to process a single location was 0.027 milliseconds.

3.6 Step-by-Step Example

This section provides the reader with a step-by-step walk through of the RTFPA algorithm as it is used to analyze a simple raster path (Figure 6 (A)). After reading this section one should understand each step in RTFPA algorithm and the reasons why each step is taken. This will allow the reader to integrate the RTFPA algorithm into their own work.

The RTFPA algorithm begins with an empty point set, *path length* equal to zero, *average step size* equal to zero, *minimum divider path length estimate* equal to zero, *maximum divider path length estimate* equal to zero, and *step count* equal to zero. The multiplier values are set a priori by the user or programmer, in this example we will use 0.5 for the minimum multiplier and 3 for the maximum multiplier as was used in the experiment discussed in the previous sections. When the first position reading is available it is stored in a *current position* variable as well as in a *minimum sphere center* variable and *maximum sphere center* variable which are used when walking the divider along the path.

At this point there is not enough information to estimate any path lengths so the algorithm waits for the next sensor reading. As the second position is available *current position* is moved to *last postion* and the new position is stored in *current position.* At this point RTFPA increments the *path length* and *step count* variables, adding the distance between *current position* and *last position* at each new position to *path length* and increasing the *step count* by 1. In this example the robot has moved 1 meter, thus *path length* now equals 1, additionally the *average step size* also equals 1. The *average step size* is multiplied by the *minimum multiplier* and the *maximum multiplier* to get the *minimum divider* and *maximum divider* sizes respectively. At this point *minimum divider* equals 0.5 and *maximum divider* equals 3. Using these divider sizes RTFPA walks along the path from positions indicated in the *sphere center* variables to *current position* if the distance between the *sphere center* and *current position* greater than or equal to the divider size. An estimate of the path length is generated using the minimum and maximum dividers as they are walked along the path. These estimates are stored in *minimum divider estimate* and *maximum divider estimate.*

As shown in Figure 6 (B) the *minimum divider* (0.5m) is small enough to walk along the current path segment, however the *maximum divider* (3m) is larger than the distance between *maximum sphere center* and *current position.* Without a path length estimate generated by the maximum divider D cannot be calculated, so at this point along the path D equals zero (0).

Once the robot reaches the next waypoint along the path (Figure 6 (C)) all variables are updated such that *path length* equals 1.25 and *average step size* equals $(1+0.25)/2 = 0.625$ thus *minimum divider* equals 0.3125 and *maximum divider* equals 1.875. and a walk from both the last minimum and maximum *sphere centers* is attempted. As shown in the figure, neither the minimum or maximum dividers are smaller than the new path

segment, thus new path length estimates cannot be updated and D remains equal to zero.

In Figure 6 (D) the *minimum divider* is 0.375m and the *maximum divider* is 2.25m. The minimum divider is able to walk along the path from the last *minimum sphere center* to *current position*, however the walk does not reach *current position* as the minimum divider is too large to intersect with the path segment that ends at *current position*. The path length estimate using the minimum divider is 1.74m and the estimate using the maximum divider is still 0m. Due to limited space the this example will skip intermediate points on the path up to where the maximum divider intersects a path segment.

Figure 6 (E) shows the minimum divider path estimate (indicated by the blue, short-dashed line) which is 6.371m in length, the maximum divider path estimate (indicated by the red, long-dashed line) is 1.875m in length. The current divider sizes at this point are 0.3125m and 1.875m for the minimum and maximum dividers respectively. Having a minimum and maximum path length estimate, D can be calculated using the 1-slope formula which results in a value of 1.683.

For the final step this example will skip to the end of the path showing the remaining steps taken by the minimum and maximum dividers and provide estimates of the path length and D. The *minimum divider* is 0.32m and the *maximum divider* is 1.92m as the robot reaches the end of the path. As shown in Figure 6 (F) the maximum divider does not intersect the path at any point past the initial intersection shown in Figure 6 (E), therefore the maximum divider path length estimate remains 1.875m. On the other hand, the minimum divider has been able to walk along the path and nearly reaches the end point. The minimum divider path length estimate is 12.17m. These estimates result in a D equal to 2.04.

Had the path been long enough for the maximum divider to intersect twice or had the multipliers been different the D estimate would change significantly. For instance if the minimum and maximum multipliers had been 0.1 and 0.25 respectively the D estimate for the line would be 1.0; thus it is important to pick the multiplier sizes carefully. The author is currently researching methods to automatically estimate multiplier sizes on the fly to eliminate the need to set any parameters manually when using the RTFPA algorithm.

4 Experiment

The experiment discussed in this work was conducted at both the main Texas A&M University (TAMU) campus in College Station, Texas as well as the adjacent Disaster City training facility over a four day period in May and June 2009. The SARGE multiplayer robot simulation game [5, 6] was used to train the participants in the operation of an Inuktun Extreme VGTV for search tasks. Twenty-five volunteers participated in the training sessions which took on average one hour per group of participants.

The training was divided into two parts. The first part introduced the player to basic operation of the robot through a short scripted tutorial (Figure 7). The tasks included basic robot control using the joystick, operation of the raise and camera tilt mechanisms, obstacle traversal, and the use of the robots headlights lights and camera zoom. The tutorial covered these topics in a series of mini-missions that presented both internal and external views of the robot. (Figure 7) Bottom shows the robot interface. In the top left corner are indicators for the raise and tilt of the robot. The real

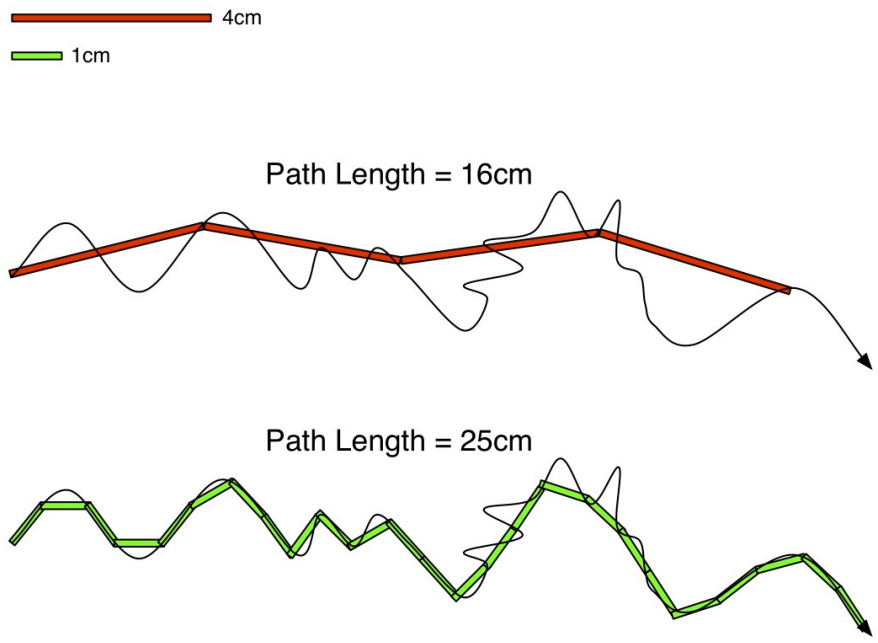

Fig. 5 In this example there are two measuring sticks, one is 4cm (red) and the other is 1cm (green). If the path is measured using the 4cm stick by walking it along the path the measured length is 16cm. If you estimate the end segment of the path that is shorter than 4cm the path length is maybe 18.5cm. However, if the path is measured using the 1cm stick the path length is 25cm. As is seen in the lower figure, even using the 1cm measuring device underestimates the path length because it is not able to capture the fine details of the squiggly path.

Inuktun VGTV interface is similarly sparse, with the addition of a battery indicator. Following the scripted tutorial the participants played two games that required them to one, maneuver the robot towards waypoint markers and and avoid plastic cups while following a zig-zag path (Figure 8), and two, navigate a series of steps to push scattered plastic cups into a goal area (Figure 9). The games were timed and the score and penalties were recorded for each game. The participants played each of the games for a maximum of 15 minutes or until they chose to continue to the next game. After a participant completed the first two games, they proceeded to play a search game (Figure 10). The search game required the participants to locate ten objects related to a human presence in a confined space within 20 minutes. The goal was to maneuver the robot near each object, select it using the mouse, and then provide a description using the keyboard. The objects that the player was required to find were: work boot, clipboard, hard hat, rubber mallet, safety glasses, screw driver, soda can, tool box, wrist watch, and work glove. The search game was completely dark except for the robot head light and arcing sparks from downed power lines that flashed periodically. The version of SARGE available on SourceForge.net includes these levels and source materials.

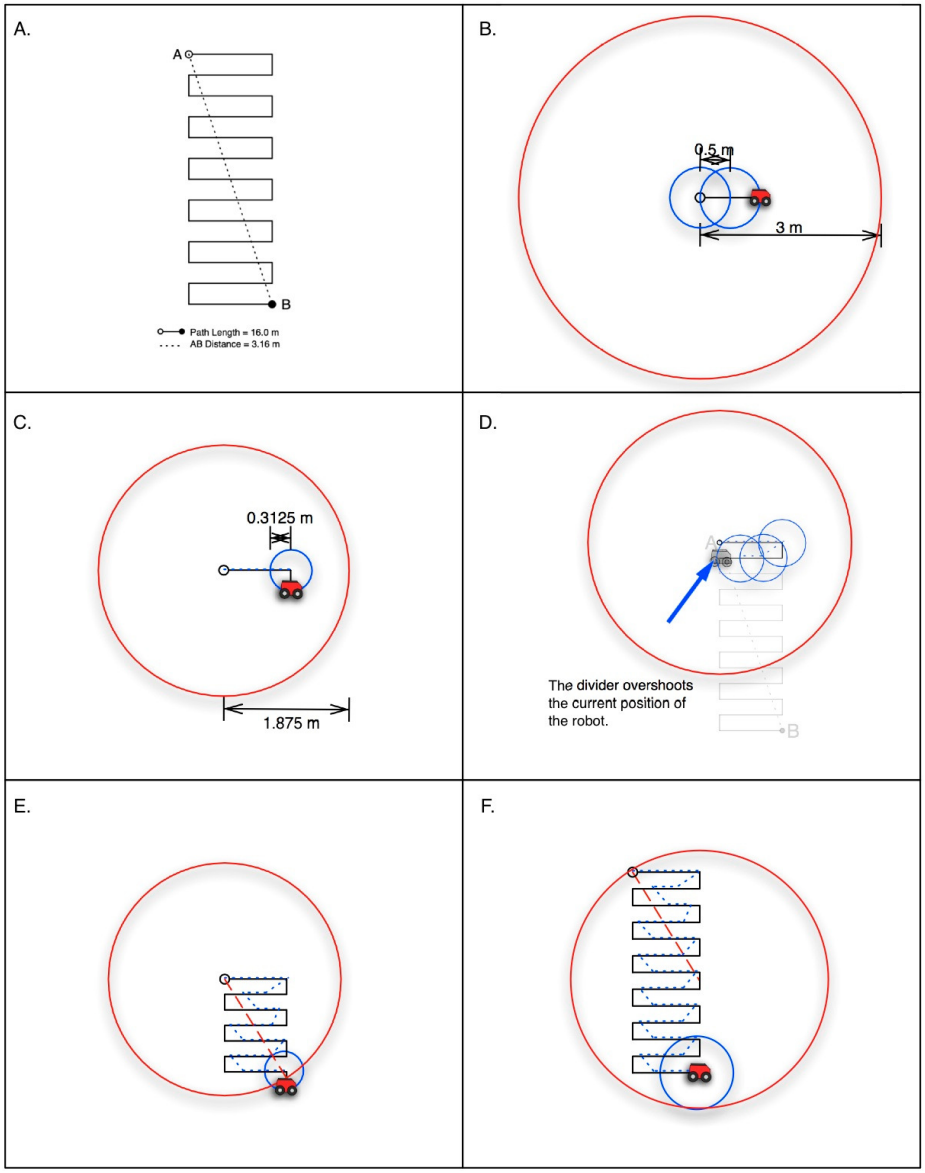

Fig. 6 A raster path with an actual D of 1.22 calculated using the divider method (A). A robot has localized itself at the start of a path (B). The robot has moved to the first waypoint on the path. The blue circles indicate the minimum divider walking along the line. The large red circle indicates the size of the maximum divider (C). As the robot moves along the raster path RTFPA adjusts the divider sizes to better estimate the remaining path length (D-F).

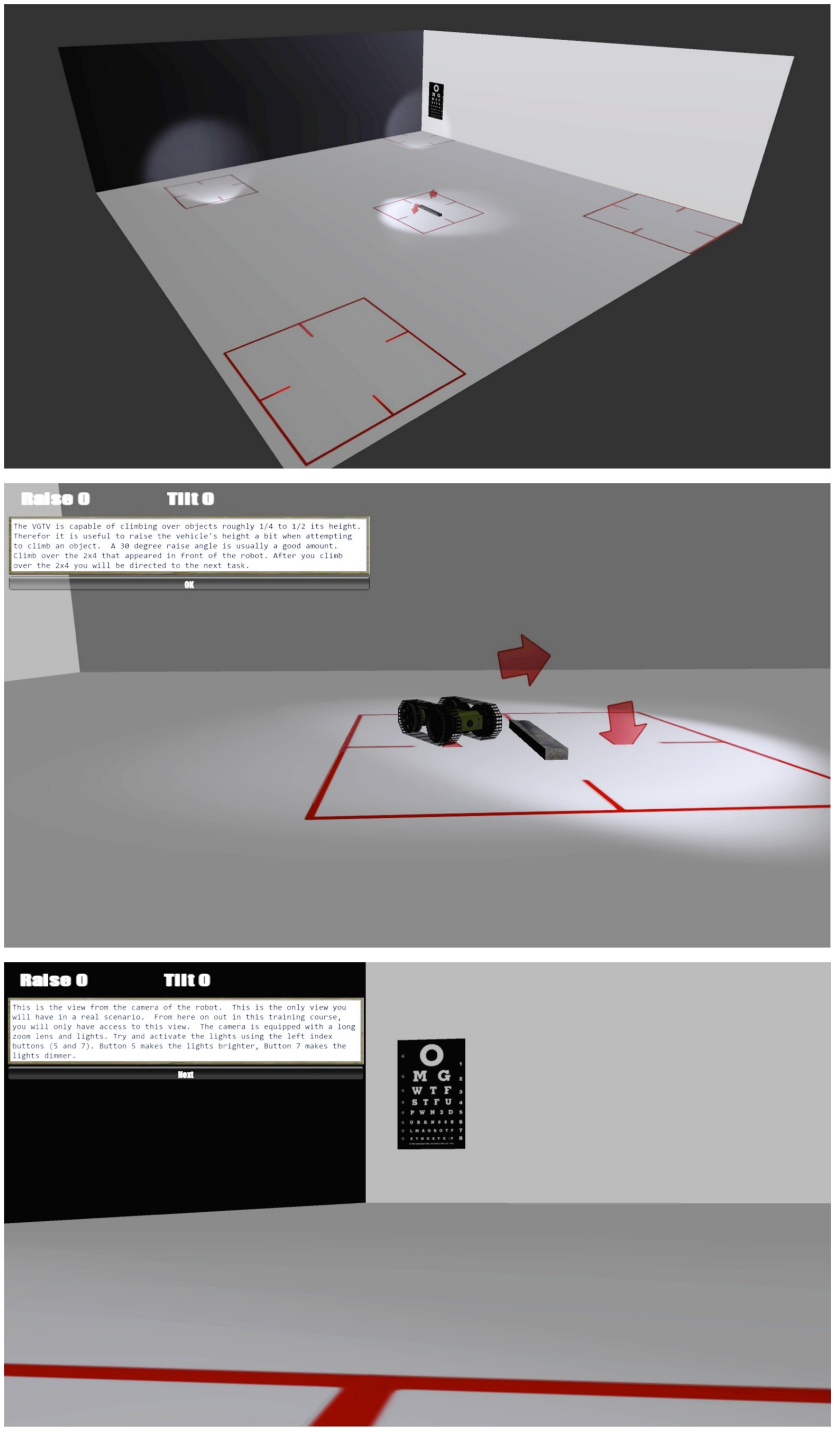

Fig. 7 This figure presents three screen shots of the SARGE VGTV tutorial. *Top*: A wide angle shot of the entire course. Each target prompted the player with a different task and instructions for accomplishing this task using the robot. *Middle*: An example task, the player was directed to drive the robot over the 2x4. *Bottom*: At the end of the tutorial the player is introduced to the robot's camera system. This is the view that is used for the remainder of the simulation.

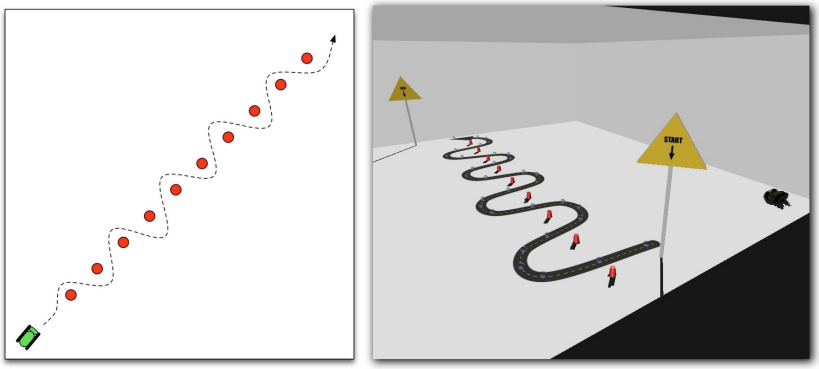

Fig. 8 This figure presents a plan view (left) and a screen shot (right) of the SARGE slalom course. The player was required to navigate the robot on the road, collecting translucent markers without hitting the cups.

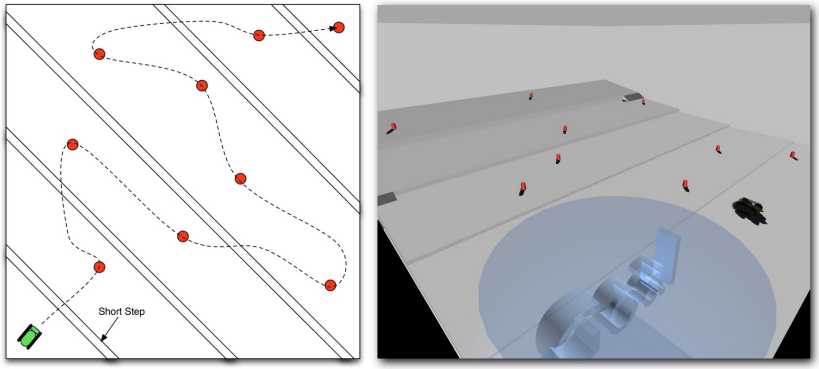

Fig. 9 This figure presents a plan view (left) and a screen shot (right) of the SARGE step room course. The player was required to push the cups into the goal area (bottom left of screen shot) using the robot.

5 Results

To demonstrate the relationship between D and performance in practice, Figure 11 shows the paths for two players in one of the training game's environments. In this environment players were required to push plastic cups (red dots) towards the goal area (the blue area in the upper right of the figure) while navigating the multilevel steps and ramps. The robot's starting location is shown (at position 0,3.5). The blue path, belonging to User 14, has an estimated D value of 1.066 while the green path, belonging to User 62 has an estimated D value of 1.20. Notice the area around the goal, the green path is much more tortuous than the blue path; as well, in the area around the cups the green path appears more tortuous.

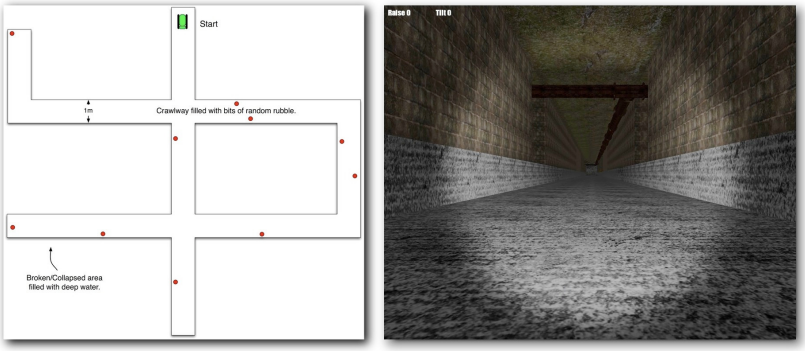

Fig. 10 This figure presents a plan view (left) and a screen shot (right) of the SARGE tunnel search course. The player was required to find and identify ten items scattered through the tunnel system. The red dots on the plan view indicate the placement of hidden objects. Note that the brightness of the screen shot has been enhanced for print.

5.1 Fractal D's Curvilinear Relationship with Score

The question answered by the experiment discussed in this paper is: Does the fractal dimension of a tele-operated robots path indicate the skill of its operator when conducting a search task? The hypothesis was that a the fractal dimension would have a negative relationship with an operators skill. That is, D would decrease as an operators skill increased. This hypothesis was based on the work of Voshell and Woods. The results of the experiment showed that within the sample population there is a curvilinear relationship between score and fractal dimension, not a linear relationship. Figure 12 shows the curvilinear relationship between fractal dimension and the score earned in the game. This relationship was tested using a non-linear regression with $F(2,23)=4.125$, $p=0.029$ and $R^2=0.264$. This curvilinear relationship supports both work by Voshell and Woods [12] that indicates that a higher situational awareness may be related to lower fractal dimensions (straighter paths) and additionally supports findings by Clarke and Goldiez [3] which indicates that a chaotic path, measured using the Lyuoponov exponent, is related to higher search performance in a maze. At the low to mid end of the scale increasing fractal dimension appears to be related to an improvement in search score; however, as can be seen in the figure, a mean D value greater than 1.125 is related to decreasing search score.

The relationship between fractal dimension and search score suggest that the data supports the initial hypothesis and indicates that a lower fractal dimension is associated with a drivers skill in operating a robot but that this relationship is non-linear; at fractal dimensions below a certain point, search score increases which may indicate that as fractal D peaks a driver becomes proficient at operating the vehicle and they search more of the area; this effect is consistent with the findings of Clarke and Goldiez. At $D > 1.12$ perhaps D continues to increase as the operator drives in an erratic pattern due to the inability to maintain situational awareness. This would most likely cause their score decreases because of the decreased understanding of the environment; this effect is consistent with the work of Voshell and Woods.

5.2 Fractal D's Relationship with Prior Gaming Experience

A regression analysis of participants that were drivers in the training game was used to test the relationship between fractal dimension and prior video gaming experience. The regression shows that a drivers prior experience with games has an increasing linear relationship with mean fractal dimension with F(1,22)=9.140, p=0.006 and R^2=0.294. That is as the participants' self rated prior experience with video games increased the mean fractal dimension for their search paths increased as well. Gaming experience was rated on a scale of 1 to 5: 1 represented no experience, 5 represented expert. Figure 13 shows the relationship between mean fractal dimension and experience with video games. The peak fractal dimension at experience level 4 is roughly 1.13; this is very close to the peak shown in Figure 12 (1.125). It should be noted that there were only 2 participants that rated themselves as expert game players.

During the experiment, the author observed that the teams that generally followed the Localize the robot, Observe the surrounding environment, look for Victims, Report findings (LOVR) strategy scored the highest [2]. This finding is based on the visual observation of the teams while they operated the robot. This strategy was not explained or taught to the participants in any way, yet it appears that the more experienced game players may have already learned a similar strategy. It is hypothesized that a successful use of the LOVR strategy would produce a D value near the peak of the curve shown in Figure 12.

5.3 Fractal D's Independence of Task Completion Time

Two regression analyses were performed to test the relationship between task completion time and fractal dimension, one analysis compared the two measures for all participants in the study while another analysis examined only those participants who completed the task under the time limit. The results confirm the findings of Voyles, et al. [14] that the fractal dimension metric is independent of task completion time. In contrast to Voyles' closed-ended task of maneuvering a robot between two points, the search task assigned for the experiment conducted in this paper is open-ended. The participants were unaware of both the layout of the environment and the location of the items of interest. Figure 14 plots the D value versus the task completion time for all 25 participants in the study. 9 of the participants completed the task in under 20 minutes, which was the maximum time allowed for the task. The regression that included all of the participants had the following results: F(1,25)=0.012, p=0.914 and $R^2 < 0.001$. If the analysis is performed on only the 13 participants who completed the task within the time limit, the results remain the same with F(1,8)=0.027, p=0.874 and R^2=0.003. These analyses clearly show that there is no indication of a relationship between the time to complete a task and fractal dimension.

6 Discussion and Conclusion

This article has presented the reader with a brief overview of the divider method for fractal path analysis followed by a detailed explanation at a new real time fractal path analysis algorithm (RTFPA) and its use in assessing the skill of robot operators.

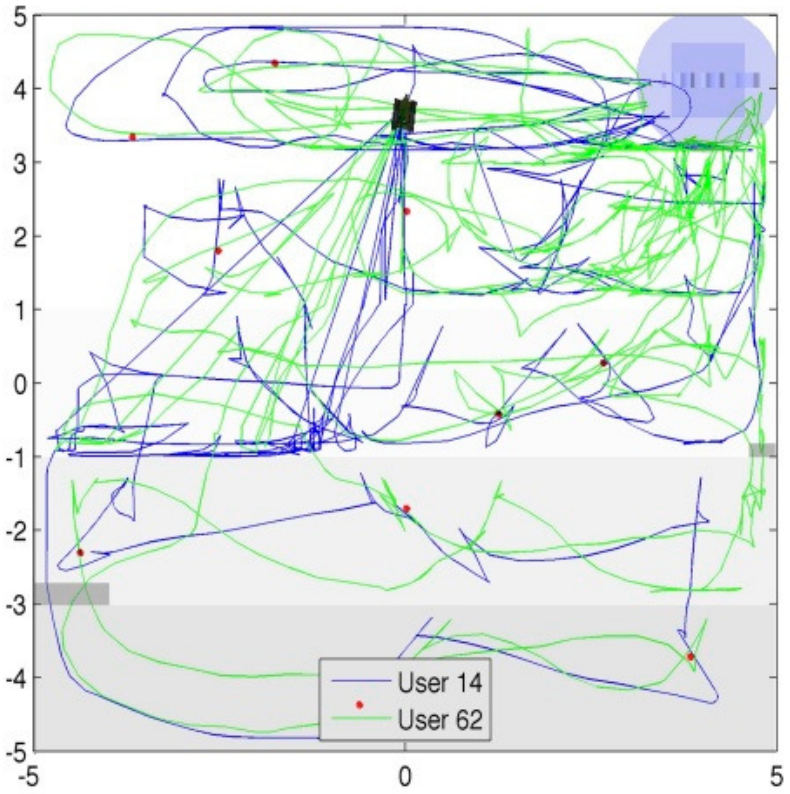

Fig. 11 This figure shows the paths for the participants with the lowest and highest fractal dimension in one of the training levels of the game. The path for User 14 has a fractal dimension of 1.066 while the path for User 62 has a fractal dimension of 1.20. The top right corner of the figure is the goal area and the red dots are the initial locations of the objects that the users pushed into the goal. The initial position of the robot can be seen at position (0,3.5).

The finding that the fractal dimension of a robot's path does seem to be a useful metric for measuring an operators search strategy within the training game suggests that fractal D could be used as a qualifying metric in future training applications. D could easily be used as an input to an intelligent tutoring system that teaches specific robot operating strategies. Each strategy would likely have an optimal execution path within an environment and this path would have an associated D value. The tutor could then use this D value as a target for the trainee to reach, adjusting the intermediate lessons to encourage the optimal operation in a specific sub-task.

Additionally, the non-linear relationship of fractal dimension to performance may be indicative of the operating teams cognitive ability if measured over time. This information could be used as a possible indicator of fatigue along with some threshold value which would allow an incident commander to relieve a team whos performance is deteriorating. This assumes that the position of the robot can be tracked with sufficient accuracy in the environment. In the case of unmanned ground vehicles (UGVs) that operate in a collapsed structure location systems tend to be very inaccurate or

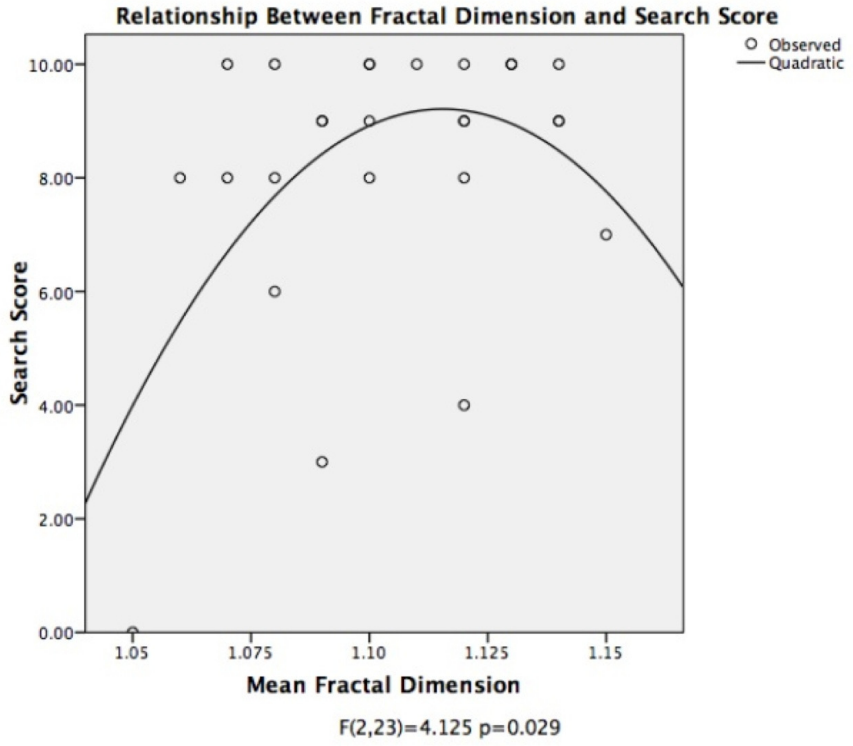

Fig. 12 Mean Fractal Dimension vs. Training Search Score. This graph shows the non-linear relationship ($SearchScore = \beta_0 + \beta_1 D + \beta_2 D^2$) between a participants mean fractal dimension (D) and their score in the training game. This relationship may indicate that there is a peak D that indicates a thorough search strategy. Values of D higher than the peak may indicate that an operator has decreased situational awareness and is struggling to localize themselves. Further experimentation is needed to verify this theory.

completely unavailable, these robots will need to have sensors onboard that can localize them within the environment in order to calculate an accurate D value.

Fractal path analysis (FPA) has the potential to provide additional information for researchers working with spatial information. In the robotics domain FPA could be used to compare various search and navigation algorithms, while RTFPA could potentially be used to identify errors in navigation systems. Additionally, FPA could be extended into the third dimension and be used for terrain analysis as part of a path planning system.

The authors future work on the RTFPA algorithm will focus on an automated method of selecting multipliers such that the min and max spatial scales will provide D values that maximally differentiate monitored individuals and robots.

7 Acknowledgement

This work was supported by the Center for Robot Assisted Search and Rescue. The author would like to thank the Texas Engineering Extension Service (TEEX) team

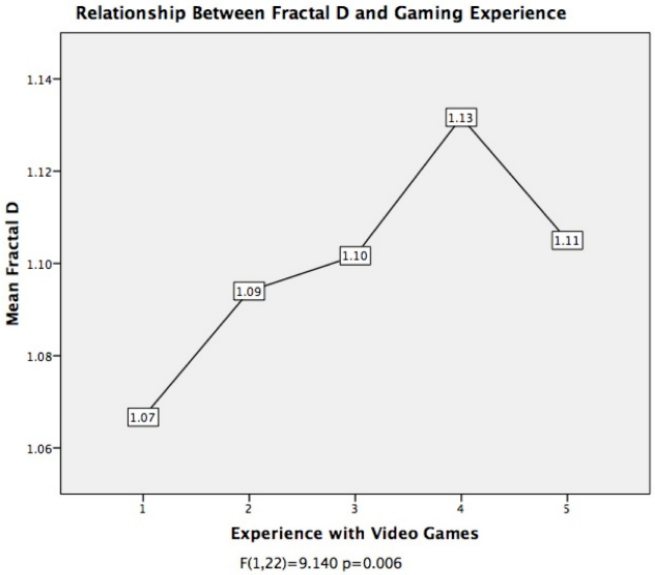

Fig. 13 Mean Fractal D vs. Prior Gaming Experience. This graph shows the relationship between a trainees prior experience with video games and the mean fractal dimension calculated during their training. The markers indicate the mean fractal dimension for each experience level.

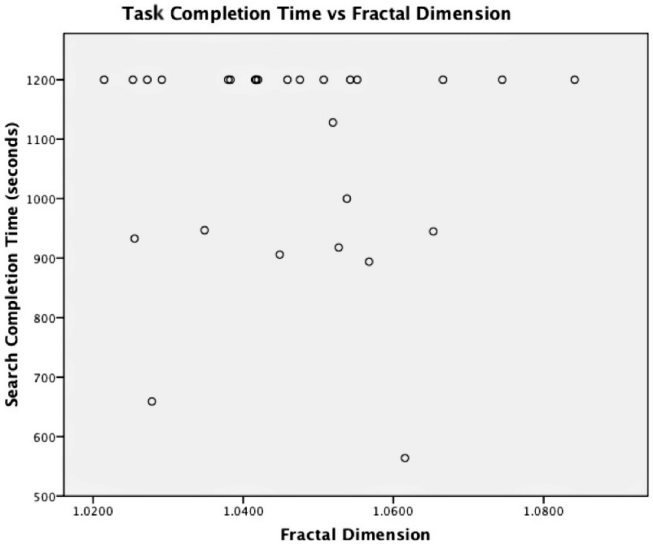

Fig. 14 This plot shows the relationship between fractal dimension and the time to complete a search task. A linear regression shows that there is no evidence of a relationship between these two metrics.

and Dr. Robin Murphy for assisting with the experiment and providing the facility at which the experiment was conducted.

References

1. Bae J, Voyles R, Godzdanker R (2007) Perliminary usability tests of the wearable joystick for gloves-on hazardous environments. In: Proceedings of the International Converence on Advanced Robotics, pp 514–519
2. Burke J, Murphy R (2004) Situation Awareness and Task Performance in Robot-Assisted Technical Search: Bujold Goes to Bridgeport. Tech. rep., University of South Florida
3. Clarke T, Goldiez B (2007) Collaboration on the edge of chaos. In: Proceedings of the 2007 International Symposium on Collaborative Technologies and Systems (CTS), pp 122–128, DOI 10.1109/CTS.2007.4621747
4. Craighead J (2008) Distributed, game-based, intelligent tutoring systems - the next step in computer based training? In: Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS 2008), pp 247–257
5. Craighead J, Burke J, Murphy R (2008) Using the unity game engine to develop sarge: A case study. In: Proceedings of the 2008 International Conference on Intelligent Robots and Systems (IROS 2008)
6. Craighead J, Gutierrez R, Burke J, Murphy R (2008) Validating the search and rescue game environment as a robot simulator by performing a simulated anomaly detection task. In: Proceedings of the 2008 International Conference on Intelligent Robots and Systems (IROS 2008)
7. Dicke M, Burrough PA (1988) Using fractal dimensions for characterizing tortuosity of animal trails. Physiological Entomology (13):393–398
8. Mandelbrot B (1967) How long is the coast of britain? statistical self-similarity and fractional dimension. Science 156(3775):636
9. Nams VO (2006) Improving accuracy and precision in estimating fractal dimension of animal movement paths. Acta Biotheoretica 54(1):1–11
10. Nams VO (2009) Fractal - a program to analyze movement paths of animal movement paths. URL http://nsac.ca/envsci/staff/vnams/Fractal.htm
11. Phillips F, Voshell M (2006) A novel metric for evaluating human-robot navigation performance. In: RTA/HFM Symposium: Human Factors of Uninhabited Military Vehicles as Force Multipliers
12. Voshell M, Woods D (2005) Breaking the keyhole in human-robot coordination: Method and evaluation. In: Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting
13. Voshell M, Woods DD, Phillips F (2007) Overcoming the keyhole in human-robot coordination: Simulation and evaluation. In: Proceedings of the Human Factors and Ergonomics Society 51st Annual Meeting, Cognitive Engineering and Decision Making, Human Factors and Ergonomics Society
14. Voyles R, Bae J (2008) The gestural joystick and the efficacy of the path tortuosity metric for human/robot interaction. In: Proceedings of Performance Metrics for Intelligent Systems (PerMIS)
15. With K (1994) Ontogenetic shifts in how grasshoppers interact with landscape structure: an analysis of movement patterns. Functional ecology(Print) 8(4):477–485