

# A Probabilistic Model for Demonstrating High Path Planning Success Rate in Autonomous Capsule Robots for Bronchoscopies

Arjan Gupta  
Robotics Engineering  
Worcester Polytechnic Institute  
Worcester, MA, USA  
agupta11@wpi.edu

**Abstract**—This manuscript describes a probabilistic view of analyzing the possible paths that an autonomous robot may take when traversing through bronchi. We will consider a section of the lung, as a model that can be expanded upon using our work. Using the technique of rapidly-exploring random trees (RRT), we aim to lay down some benchmarks that can be used by clinical experts to meet the goal of the bronchoscopies they perform.

**Index Terms**—capsule robot, autonomous bronchoscope, path planning, benchmark parameters

## I. INTRODUCTION

Bronchoscopies play a vital role in diagnosing a range of pulmonary issues. For example, tumor detection is an important cause for an endoscopy performed by a pulmonologist. However, low diagnostic yields is an existing problem in transbronchial biopsies [1], [2], [3], [4]. Among the many solutions to address this problem, robotic-assisted bronchoscopy is a promising and ever-growing method [5]. In the variety of types of robotic bronchoscopies, a capsule-robot has potential and can be expanded upon in many ways. For example, the existence of the PillCam COLON capsule has shown promise in the last decade [6]. Although again in the realm of colonoscopy screenings, a capsule endoscopy has shown to be considerably more cost-effective as compared to traditional methods [7].

While rigid and fiberoptic methods remain the dominant methods of performing pulmonary endoscopies [5], they also require highly skilled operators in order to be safe [8]. However, given the forecasted shortage in the physician workforce of the United States [9], it is imperative to look for alternative solutions. Autonomous endoscopies are one such solution. In fact, autonomous capsule robots have already been trained using reinforcement learning for usage in endoscopies [10].

However, at the time of the scribing of this manuscript, the existing literature does not show any preliminary benchmark information about this type of robotic bronchoscopy. If clinical operators hope to use autonomous capsule robots for bronchoscopies, there needs to exist guidelines on tuning the

autonomous robot such that a high success rate for biopsy yield can be achieved. *The primary objective of this study is to show a simple probabilistic method to achieve over 98% success rate of an autonomous capsule robot to find a given destination in the human bronchi. We will use rapidly-exploring random trees to set up a model that can establish parameters which can be tuned to achieve a high success rate.*

## II. MATERIALS AND METHODS

### A. Set up for the model

First we use a simple figure that delineates occupied and free zones of a human pulmonary region. Such a figure is shown in Fig 1. The green region is the area where the capsule robot is free to move, and the black regions are the ‘occupied’ zones, signifying the walls of the organ.



Fig. 1. A basic model of the human lung

The capsule robot’s dimensions are roughly 10 mm long capsule with 5 mm in diameter, but for the sake of model simplicity, the dimensions are ignored in the path planning. We assume that the robot can start anywhere near the trachea and have a goal position near the end of the narrow bronchi.

### B. Occupancy grid and start/goal positions

Next, we read in the image using MATLAB’s *imread* function. Now that we have a raw byte version of the image, we convert it to a gray-scale image. By default MATLAB reads this in as the colored portion marked as black, so our next step is to invert the image, which we can do with a simple logical invert on the image matrix. At this point, we

can use MATLAB's *binaryOccupancyMap* function to create an occupancy grid, so that we can determine which regions are occupied and free. The image was given a resolution of  $10^4$ , in order to model a more realistic size of the lung.

Furthermore, the start position of the robot was randomized. In our image,  $750 \times 10^{-4}$  is roughly the y-height of the trachea. However our robot could start anywhere along the x-width of this trachea, therefore we use the *randi* function to randomly select an x-position anywhere between  $685 \times 10^{-4}$  and  $800 \times 10^{-4}$ .

To select random locations for the goal, we first randomly selected the left or right narrow airway. If the left one was selected, we selected a y-height of  $660 \times 10^{-4}$  and randomly generated the x-position between  $415 \times 10^{-4}$  and  $445 \times 10^{-4}$ . If the right narrow airway was selected, we randomly generated the x-position between  $540 \times 10^{-4}$  and  $560 \times 10^{-4}$ .

### C. Dubins and Reeds-Shepp Curves

Next, we model our state space for the path planning algorithm based on Reeds-Shepp Curves. To understand it, we first describe the Dubins path. A Dubins path analyzes a simple car model, and generates the shortest path between two 2D poses  $(x, y, \theta)$ . The paths can have three segments, right turn (R), left turn (L), straight line (S). Based on optimization techniques, the possible path can have the the following combinations,

$$\{L_\alpha R_\beta L_\gamma, R_\alpha L_\beta R_\gamma, L_\alpha S_d L_\gamma, L_\alpha S_d R_\gamma, R_\alpha S_d L_\gamma, R_\alpha S_d R_\gamma\}$$

where  $\alpha, \gamma \in [0, 2\pi)$ ,  $\beta \in (\pi, 2\pi)$ , and  $d \geq 0$ . An example of  $R_\alpha S_d L_\gamma$  is shown in Fig 2.

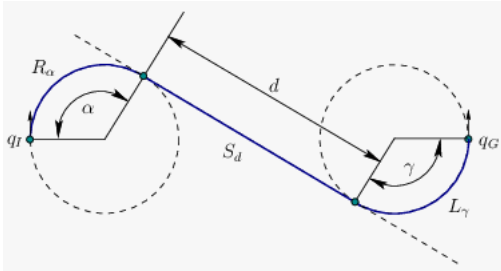


Fig. 2. An example of  $R_\alpha S_d L_\gamma$  Dubin's curve

The Reeds-Shepp curve is the same as Dubins curve, except that reverse direction movement is now allowed. Therefore, the sequences of motion primitives can be arrived at by using the Dubins combinations, and applying permutations using  $R^+, R^-, L^+, L^-$ , where the positive and negative signs denote forward and reverse motion.

In our MATLAB code, we specify that we want to use the *stateSpaceReedsShepp* object with the bounds given by the *binaryOccupancyMap*. We specify a minimum turning radius of  $20 \times 10^{-4}$ .

### D. Validation and path planning

We additionally set up a *validatorOccupancyMap* with a validation distance of  $5 \times 10^{-4}$ . Finally, we are ready to call

our rapidly-exploring random trees planner, *plannerRRT*. For the RRT algorithm, we set our maximum branch connection distance to be 10 times our validation distance. We also set our maximum iterations at  $9 \times 10^3$ .

After this, we plot the various branches of the RRT algorithm, as well as the path, if it is found. We also show our starting and goal positions in the plot.

## III. RESULTS

An example plot of our RRT algorithm is shown in Fig 3.

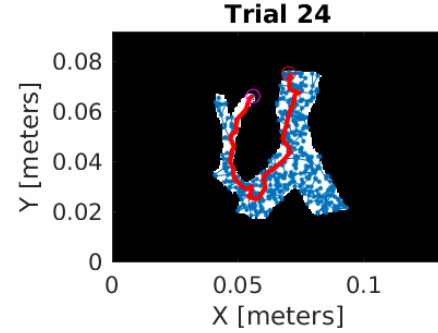


Fig. 3. An example trial of the RRT algorithm

The RRT algorithm we have written was run in batches of 24 trials, and plotted as subplots. An example of this is shown in These batches of 24 trials were then run 5 times in order to average the success rate.

## IV. DISCUSSION

In the opinion of the author, this homework problem set was insightful. The given methods can be applied to analyze the reward system for reinforcement learning agents in autonomous capsule robot navigation.

## V. APPENDIX

### A. Success rate assessment script

```
1 clear; close all; clc;
2
3 total_trials = 24;
4 solutions = 0;
5
6 for i = 1:total_trials
7     subplot(4,6,i);
8     fprintf("Running trial #%d... ", i);
9     solutionFound = capsule(i);
10    if solutionFound
11        solutions = solutions + 1;
12    end
13    fprintf("\n");
14 end
15 set(gcf, 'Position', get(0, 'Screensize'));
16
17 fprintf("Success rate: %f%%\n", ...
18     100*(solutions/total_trials), "%");
```

## B. Path planning for capsule robot

```

1 function solutionFound = capsule(trial_num)
2 solutionFound = true;
3 % Load in the airway image, transform to ...
  grayscale and invert it
4 [airway,color_map,transp] = ...
  imread("939-Oblique.png");
5 gray_airway = rgb2gray(airway);
6 inverted_airway = gray_airway == 0;
7
8 % Create occupancy grid
9 resolution = 10000;
10 occGrid = ...
  binaryOccupancyMap(inverted_airway, ...
  resolution);
11 show(occGrid)
12
13 rng shuffle
14
15 % Set start and goal poses
16 start = [randi([685,800])/resolution, ...
  750/resolution, (randi(200)/100)*pi];
17
18 leftorRight = randi(2);
19 goal = zeros(1,3);
20 if leftorRight == 1
21   goal = [randi([415,445])/resolution, ...
  660/resolution, (randi(200)/100)*pi];
22 else
23   goal = [randi([540,560])/resolution, ...
  660/resolution, (randi(200)/100)*pi];
24 end
25
26 bounds = [occGrid.XWorldLimits; ...
  occGrid.YWorldLimits; [-pi pi]];
27
28 % Use Reeds Shepp curves for defining state ...
  space
29 capsuleStateSpace = ...
  stateSpaceReedsShepp(bounds);
30 capsuleStateSpace.MinTurningRadius = ...
  20/resolution;
31
32 % Plan the path
33 capsuleStateValidator = ...
  validatorOccupancyMap(capsuleStateSpace);
34 capsuleStateValidator.Map = occGrid;
35 capsuleStateValidator.ValidationDistance = ...
  5/resolution;
36
37 planner = plannerRRT( ...
  capsuleStateSpace, ...
  capsuleStateValidator);
38 planner.MaxConnectionDistance = 50/resolution;
39 planner.MaxIterations = 9000;
40
41 planner.GoalReachedFcn = @checkIfReachedGoal;
42
43 rng shuffle
44
45 [pthObj, solnInfo] = plan(planner,start,goal);
46
47 show(occGrid)
48 hold on
49
50 % Plot entire search tree
51 plot( ...
  solnInfo.TreeData(:,1), ...
  solnInfo.TreeData(:,2), '-');
52
53 if pthObj.NumStates > 0

```

```

58 % Interpolate and plot path
59 interpolate(pthObj,300)
60 plot( ...
  pthObj.States(:,1), ...
  pthObj.States(:,2), ...
  'r-','LineWidth',2)
61 fprintf("Path found.")
62 else
63   fprintf("No path solution found.")
64   solutionFound = false;
65 end
66
67 % Show start and goal in grid map
68 plot(start(1),start(2),'ro')
69 plot(goal(1),goal(2),'mo')
70
71 % Edit title text
72 title(sprintf("Trial %d", trial_num))
73 hold off
74 end

```

## C. Callback function for goal proximity check

```

1 function isReached = ...
  checkIfReachedGoal(planner, goalState, ...
  newState)
2 resolution = 10000;
3 isReached = false;
4 threshold = 5/resolution;
5 if planner.StateSpace.distance(newState, ...
  goalState) < threshold
6   isReached = true;
7 end
8 end

```

## REFERENCES

- [1] M. P. Rivera, A. C. Mehta, and M. M. Wahidi, "Establishing the diagnosis of lung cancer: Diagnosis and management of lung cancer, 3rd ed: American college of chest physicians evidence-based clinical practice guidelines," *Chest*, vol. 143, 2013.
- [2] J. S. W. Memoli, P. J. Nietert, and G. A. Silvestri, "Meta-analysis of guided bronchoscopy for the evaluation of the pulmonary nodule," *Chest*, vol. 142, 2012.
- [3] A. C. Mehta, K. L. Hood, Y. Schwarz, and S. B. Solomon, "The evolutionary history of electromagnetic navigation bronchoscopy: State of the art," 2018.
- [4] D. E. Ost, A. Ernst, X. Lei, K. L. Kovitz, S. Benzaquen, J. Diaz-Mendoza, S. Greenhill, J. Toth, D. Feller-Kopman, J. Puchalski, D. Baram, R. Karunakara, C. A. Jimenez, J. J. Filner, R. C. Morice, G. A. Eapen, G. C. Michaud, R. M. Estrada-Y-Martin, S. Rafeq, H. B. Grosu, C. Ray, C. R. Gilbert, L. B. Yarmus, and M. Simoff, "Diagnostic yield and complications of bronchoscopy for peripheral lung lesions: Results of the aquire registry," *American Journal of Respiratory and Critical Care Medicine*, vol. 193, 2016.
- [5] M. Lu, S. Nath, and R. W. Semaan, "A review of robotic-assisted bronchoscopy platforms in the sampling of peripheral pulmonary lesions," 2021.
- [6] S. N. Adler and Y. C. Metzger, "Pillcam colon capsule endoscopy: Recent advances and new insights," 2011.
- [7] C. Hassan, A. Zullo, S. Winn, and S. Morini, "Cost-effectiveness of capsule endoscopy in screening for colorectal cancer," *Endoscopy*, vol. 40, 2008.
- [8] D. Stahl, K. Richard, and T. Papadimos, "Complications of bronchoscopy: A concise synopsis," *International Journal of Critical Illness and Injury Science*, vol. 5, 2015.
- [9] X. Zhang, D. Lin, H. Pforsich, and V. W. Lin, "Physician workforce in the united states of america: Forecasting nationwide shortages," *Human Resources for Health*, vol. 18, 2020.

- [10] M. Turan, Y. Almalioglu, H. B. Gilbert, F. Mahmood, N. J. Durr, H. Araujo, A. E. Sari, A. Ajay, and M. Sitti, "Learning to navigate endoscopic capsule robots," *IEEE Robotics and Automation Letters*, vol. 4, 2019.