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 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, 750×10^{-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 685×10^{-4} and 800×10^{-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 660×10^{-4} and randomly generated the x-position between 415×10^{-4} and 445×10^{-4}. If the right narrow airway was selected, we randomly generated the x-position between 540×10^{-4} and 560×10^{-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, θ). 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.

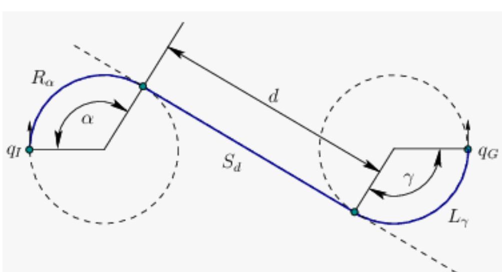

Fig. 2. An example of $R_\alpha S_d L_\gamma$ Dubin's curve

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 20×10^{-4}.

D. Validation and path planning

We additionally set up a *validatorOccupancyMap* with a validation distance of 5×10^{-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 9×10^3.

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.

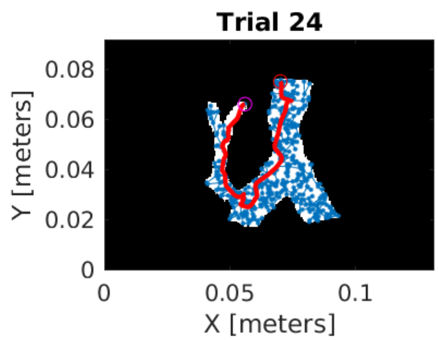

Fig. 3. An example trial of the RRT algorithm

The RRT algorithm we have written was run in batches of 24 trials, and plotted as subplots. For each path successfully found path, we incremented the total successes by 1. After all 24 trials were run, we divided the number of successes by 24 in order to calculate the success rate. An example of a perfect success rate is shown in Fig 4.

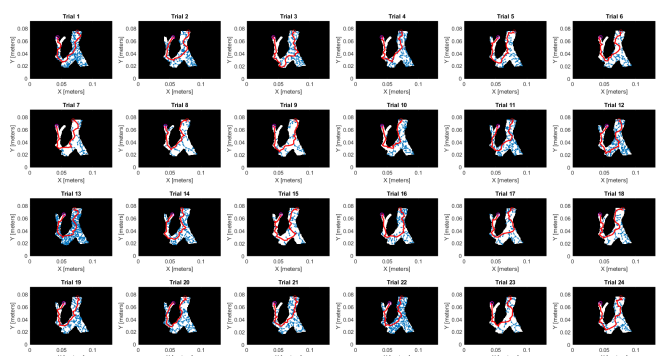

Fig. 4. An example of a batch of trials with 100% overall success rate

.

These batches of 24 trials were then run 5 times in order to average the success rate. In our final results, we obtained 2 results of 95.83% success rate, and 3 perfect success rate batches. We have formulated Table I in order to summarize the results. From this table, we can average our overall success rate as 98.33%. Although run time for each trial was not explicitly measured (since it can vary based on computational power), a regularly fast laptop can run all 24 trials in roughly 30 seconds.

Trial Batch	Success Rate
1	100.00%
2	95.83%
3	100.00%
4	95.83%
5	100.00%

TABLE I
BATCH RESULTS FOR RRT ALGORITHM

IV. DISCUSSION

The given methods of probabilistically analyzing the success rate of a capsule robot reaching its goal location in a bronchial region can be applied to improve the reward system for reinforcement learning (RL) agents in autonomous capsule robot navigation, specifically for bronchoscopy. Autonomous endoscopic robotic systems often use RL agents [10] in order to navigate in a 'maze' situation like the one we have for transbronchial navigation. The agent takes an action which is applied to the environment, and an observing interpreter replies with a state and a reward for the agent. Such a closed-loop scenario propagates until the goal is reached. A pictorial representation of this closed loop is shown in Fig 5.

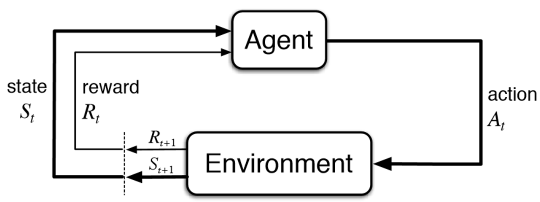

Fig. 5. A typical reinforcement learning scenario

The reward system for the agent needs to be properly trained in order to guide the agent toward the goal. For proper training, tens of thousands of example paths need to be provided to the reward system. This training data can be generated using the method described in this manuscript. Once adequately adept autonomous bronchoscopes can be developed, it can lead to a path for improving biopsy diagnostic yields in bronchoscopies.

V. APPENDIX

A. Success rate assessment script

```
clear; close all; clc;

total_trials = 24;
solutions = 0;

for i = 1:total_trials
    subplot(4,6,i);
    fprintf("Running trial #%d... ", i);
    solutionFound = capsule(i);
    if solutionFound
        solutions = solutions + 1;
    end
    fprintf("\n");
end
set(gcf, 'Position', get(0, 'Screensize'));

fprintf("Success rate: %f%s\n", ...
    100*(solutions/total_trials), "%");
```

B. Path planning for capsule robot

```
function solutionFound = capsule(trial_num)

% Default the flag to true
solutionFound = true;

% Load in the airway image, transform to ...
    grayscale and invert it
[airway,color_map, transp] = ...
    imread("939-Oblique.png");
gray_airway = rgb2gray(airway);
inverted_airway = gray_airway == 0;

% Create occupancy grid
resolution = 10000;
occGrid = ...
    binaryOccupancyMap(inverted_airway, ...
    resolution);
show(occGrid)

rng shuffle

% Set start and goal poses
start = [randi([685,800])/resolution, ...
    750/resolution, (randi(200)/100)*pi];

% Select left or right narrow airway for the ...
    goal
leftorRight = randi(2);
goal = zeros(1,3);
if leftorRight == 1
    goal = [randi([415,445])/resolution, ...
        660/resolution, (randi(200)/100)*pi];
else
    goal = [randi([540,560])/resolution, ...
        660/resolution, (randi(200)/100)*pi];
end

bounds = [occGrid.XWorldLimits; ...
    occGrid.YWorldLimits; [-pi pi]];

% Use Reeds Shepp curves for defining state ...
    space
capsuleStateSpace = ...
    stateSpaceReedsShepp(bounds);
capsuleStateSpace.MinTurningRadius = ...
    20/resolution;

% Plan the path
capsuleStateValidator = ...
    validatorOccupancyMap(capsuleStateSpace);
capsuleStateValidator.Map = occGrid;
capsuleStateValidator.ValidationDistance = ...
    5/resolution;

planner = plannerRRT( ...
    capsuleStateSpace, ...
    capsuleStateValidator);
planner.MaxConnectionDistance = 50/resolution;
planner.MaxIterations = 9000;

planner.GoalReachedFcn = @checkIfReachedGoal;

rng shuffle
```

```
[pthObj, solnInfo] = plan(planner,start,goal);

show(occGrid)
hold on

% Plot entire search tree
plot( ...
    solnInfo.TreeData(:,1), ...
    solnInfo.TreeData(:,2),'.-');

if pthObj.NumStates > 0
    % Interpolate and plot path
    interpolate(pthObj,300)
    plot( ...
        pthObj.States(:,1), ...
        pthObj.States(:,2), ...
        'r-','LineWidth',2)
    fprintf("Path found.")
else
    fprintf("No path solution found.")
    solutionFound = false;
end

% Show start and goal in grid map
plot(start(1),start(2),'ro')
plot(goal(1),goal(2),'mo')

% Edit title text
title(sprintf("Trial %d", trial_num))
hold off
end
```

C. Callback function for goal proximity check

```
function isReached = ...
    checkIfReachedGoal(planner, goalState, ...
    newState)
    resolution = 10000;
    isReached = false;
    threshold = 5/resolution;
    if planner.StateSpace.distance(newState, ...
        goalState) < threshold
        isReached = true;
    end
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 evolutional 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.