Implementation of the Coverage Path Planning for Robot-assisted Damage Control Surgery with Remote Center-of-motion Constraints

1st Jiawei Ge
Mechanical Engineering Department
University of Maryland, College Park
College Park, USA
jge0@umd.edu

2nd Michael Kam
Mechanical Engineering Department
University of Maryland, College Park
College Park, USA
mkam@umd.edu

3rd Reza Monfaredi
Sheikh Zayed Institute
for Pediatric Surgical Innovation
Children's National Medical Center
Washington, D.C., USA
rmonfare@childrensnational.org

***Abstract*—This paper reports a robotic system for implementing coverage path planning on robot-assisted damage control surgery. The system consists of a lightweight robotic arm, a customized tool for delivering self-expanding foam, a planner for coverage planning, and control algorithms for guiding the robot to complete coverage motion. The system first computes the trajectory for covering a target area and then operates remote center-of-motion through an entry port to travel the entire path. We simulated the motion and implemented on a 7 degree-of-freedom lightweight arm to evaluate the results.**

***Index Terms*—coverage path planning (CPP), Morse decomposition, remote center-of-motion (RCM), robot-assisted surgery (RAS), damage control surgery (DCS)**

I. Introduction

The first hour after a traumatic injury is considered the Golden Hour in which timely surgical intervention is essential for survival of critically injured patients [1]. Damage control surgery(DCS) is a technique of surgery used to saves lives [2]. In some scenario, such as on the battlefield, instead of performing an operation on a wounded soldier in the first place, the DCS is used to control hemorrhage and minimize contamination until the soldier is sent to a hospital for further treatment. While performing the DCS in the battlefield, current approaches rely heavily on the surgeon experience and the environmental condition, leading to high variability operating the procedures. However, Robot-assisted surgery(RAS) system incorporates highly dexterous tools, hand tremor filtering, and motion scaling to increase the surgery accuracy and consistency in DCS [3].

Recently, a self-expanding foam used in DCS becomes a promising new procedure to rapidly control hemorrhage minimally invasively with reduced risk of contamination [4]. Once the foam is injected into a body, it quickly expands to fill the abdominal cavity and applies life-saving pressure stabilizing the patient. When performing the foaming injection, our assumption is that the DCS will be performed better if the foam is covered entire target area. Therefore, if the procedure is guided by a coverage path planner, which travels all over the target area, the foam will be dispersed evenly and without much inter-lap area. In addition, due to the constraint of the single port, the robot will need to use a programmable remote center-of-motion mechanism to achieve positions along the path. Comparing with a simple injection at a single location, foaming procedure using coverage path planning will cover more target area and control hemorrhage effectively.

The goal of this project is to implement a coverage path planning algorithm on single-port foaming procedure in robot-assisted surgery. The result will be evaluated through a simulation in Gazebo with robot-operating-system(ROS) using an iiwa KUKA robotic arm. We develop algorithms including i) a coverage path planner to identify the path first based on a provided map, generating the path to cover the target area. ii) a control strategy for maneuvering the robotic arm to complete trajectory through a single port constraint. Finally, we perform tests for evaluating the trajectory of the robot traveled through the target region on a cardboard with a specific shaped cut.

II. METHOD

A. Morse Decomposition for Coverage

The boustrophedon decomposition approach [5] inspires our implementation of the cell decomposition. However, instead of analyzing the vertices in the workspace, we find the critical points, which cause the connectivity changes of the slice, to locate the cell boundaries. The upcoming subsection 1 explains how we find the critical points in the configuration space. Then subsection 2 mentions the cell decomposition method we implement. Once the decomposition is completed, the planner determines a coverage path in two steps. We need to find an inter-cell exhaustive walk throughout all cells and then plan an explicit robot motion inside every cell. The two steps will be covered in the subsection 3 and 4.

1) Critical Points:

The critical points are found by different methods in smooth boundaries and non-smooth boundaries situations.

For smooth configuration space edges, we know that a function takes its extrema values at its critical points where its first derivative vanishes or the boundaries of its domain. A critical point p is non-degenerate if and only if its Hessian is non-singular. If all the critical points of a function are nondegenerate, then the function is a Morse function. [6] In our experiments, the Morse function are chosen to be $h(x) = \lambda$. Therefore, it's apparent that the gradient of the Morse function $\nabla h(x)$ should be a horizontal direction which is perpendicular to the sweeping line in the 2d case. And we denote the surface normal to be $\nabla m(x)$. From the above, the following Corollary immediately follows. [7]

Corollary1. The point x in a smooth boundary is a critical point of the free configuration space if and only if $\nabla h(x)$ is parallel to $\nabla m(x)$.

Or we can present it more intuitively,

Corollary2. The slice is tangent to an obstacle boundary at critical points.

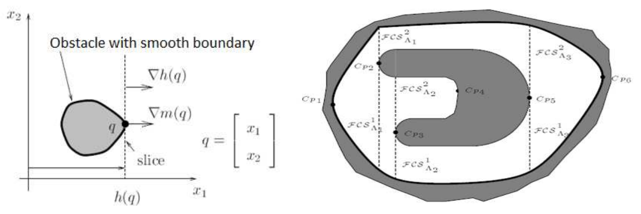
x_2
Obstacle with smooth boundary
$\nabla h(q)$
$\nabla m(q)$
q
slice
$q = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
$h(q)$
x_1
$\mathcal{FCS}^2_{\Lambda_1}$
$\mathcal{FCS}^2_{\Lambda_3}$
C_{P2}
C_{P6}
$\mathcal{FCS}^2_{\Lambda_2}$
C_{P4}
C_{P1}
C_{P5}
$\mathcal{FCS}^1_{\Lambda_1}$
C_{P3}
$\mathcal{FCS}^1_{\Lambda_3}$
$\mathcal{FCS}^1_{\Lambda_2}$

Fig. 1. Critical points for smooth boundaries

Fig. 1 left helps to understand the corollaries well. For the point p, the $\nabla h(x)$ is parallel to $\nabla m(x)$ and the slice is tangent to the boundary as well. Fig. 1 right is another example shows different situations of critical points in the smooth points. C_{p2}, C_{p3}, C_{p4} and C_{p5} are all critical points created by the central obstacle. There is a minor difference of the point C_{p4} because the slice there doesn't intersect with the free space directly, thus making the upright cell dividing line vanished. C_{p1} and C_{p6} are critical points caused by the outer boundary of the free space. They are similar to the C_{p4}.

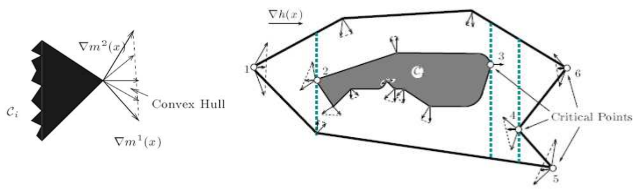

Fig. 2. Critical points for non-smooth boundaries

For non-smooth configuration edges, the surface normal is undefined, thus making the method above not compatible because we have nothing to compare with the gradient of the Morse function. However, we can use Clarke's generalized gradient to define the surface normals at these non-smooth points. [8] If we denote $\nabla m^1(x)$ and $\nabla m^2(x)$, which are shown in the Fig. 2 left, as the surface normal for adjacent smooth surfaces that are subsets of a piecewise smooth surface. The generalized gradient is the set of vectors within the convex hull of $\nabla m^1(x)$ and $\nabla m^2(x)$. Therefore, at a non-smooth point, if the gradient of the slice function $\nabla h(x)$ is contained within the convex hull of the generalized gradient, we can locate a non-smooth critical point. Note that at the smooth boundary points, the generalized gradient reduces to the conventional gradient. [7] From the example shown in the Fig. 2 right, points 1-6 are all critical points because the horizontal gradient of the slice falls into the convex hulls.

2) *Cell Decomposition:*

Definition1. Morse Decomposition is an exact cellular decomposition whose cells are the connected components of free configuration space with critical points.

The critical points can be found using the methods listed in the previous subsection. Using the critical points found in the free configuration space, we can find the dividing lines of the cells. The processor is working regarding the configuration space as a grayscale image, the pattern recognition method presented by Haralick [9] can be implemented to recognize different cells by the dividing lines and assign them with order by column sweeping from left to the right.

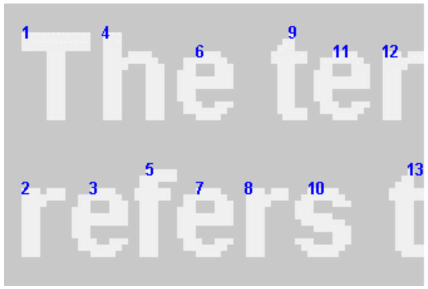

Fig. 3. Matlab function bwlabel clarification

Fig. 3 is a very good example to show the concept of the Matlab function we use for implementation, bwlabel. The algorithm tries to covering the area column-wisely, from left to right, by defining the connected 0 valued area of an image matrix to be a separated cell. The searching will start from the top-left pixel. Since the left margin of the letter 'T' and 'r' is the same, 'T' will be defined as the cell #1 and 'r' is the cell #2. Because the left margin of letter 'e' is relatively left to the left margin of letter 'h', 'e' will be defined as cell #3 and then assign a cell number to letter 'h', etc.. The grayscale value of the free space and obstacle will be assigned to 0 and 255, respectively, in the first place. Then finding the upright separating lines based on some of the critical points which connect to free space directly along the slice. Replacing the grayscale value to be 255 in these lines rather than previous 0. Finally, the bwlabel can be called and the cells will be divided and labeled with an order.

3) *Inter-cell Exhaustive Walk:*

The depth-first search algorithm for a graph data structure is one of the solutions to the inter-cell path planner. We can start from a randomly selected root in the graph and explore along each branch until it traverses all the cells. For complicated cases, this method will always give us a solution but it'll vary from time to time depending on which cell is detected and labeled first, which means the result might (actually in most cases) not be the optimal solution. Since the configuration space in our experiments is quite simple and without any obstacle in the center, it won't effect our result - the path we acquired is still optimal.

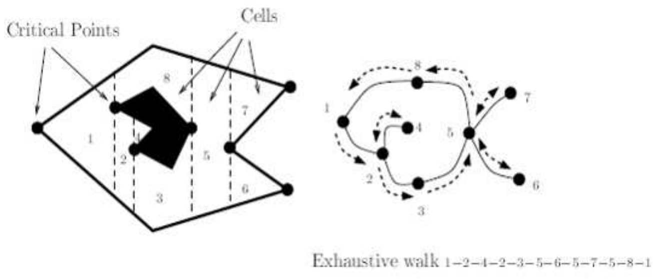

Fig. 4. Depth-first search

Depth-first search for graphs should be solved by a stack or first-in last-out concept. Assume there's a heuristic cost between connected cells and we start from the cell #1 in Fig. 4 example. The cell #1 will go into the stack first. Then we compare the cost-to-do between cell #2 and #8, choose #2 in that the cost is smaller and put it into the stack. Then compare between cell #3 and #4, choose #4 first with smaller cost. Because there's no unvisited cell connected to cell #4, we need to pop out it from the stack and go back to the previous cell to find other unvisited options, which makes the cell #3 the next in the stack. By doing so, we can generate the final path between cells as shown in the bottom-right of the Fig. 4.

What's more, Since the adjacency graph is connected, the union of the cells corresponding to the nodes fills the free space and coverage within each cell is trivial, complete coverage reduces to finding an exhaustive walk through the adjacency graph. In general, each cell in the exhaustive walk may not be unique. When a robot passes through an already covered cell, instead of covering it again, it simply plans a path to the next cell in the exhaustive walk list using a conventional path planner. For instance, we will traversal cell #2 right after traversal cell #1 in the Fig. 4 example. However, the path contains the cell #2 again after #4. At this time, we'll move the robot from cell #4 to the center of cell #2 and then next cell rather than traversal cell #2 again.

4) In-cell Coverage Path Planning:

To cover each cell, the boustrophedon motion is selected. The robot follows the "mowing the lawn" patterns, which can be separated to motion along the slice and motion along the boundaries in detail. It's demonstrated in the Fig. 5 below.

These motions are easy to determine because of the simple structure of the cells of which Morse theory assures us. One step to decide start and goal point in each cell and another step to realize the boustrophedon motions will be enough.

B. Remote center-of-motion

Since the foaming tool can only be maneuvered based on the incision point, we introduce applying remote center-of-motion (RCM) to complete moving the tool inside of the

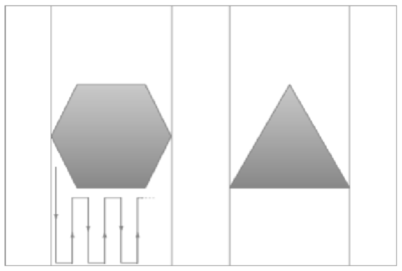

Fig. 5. Boustrophedon pattern

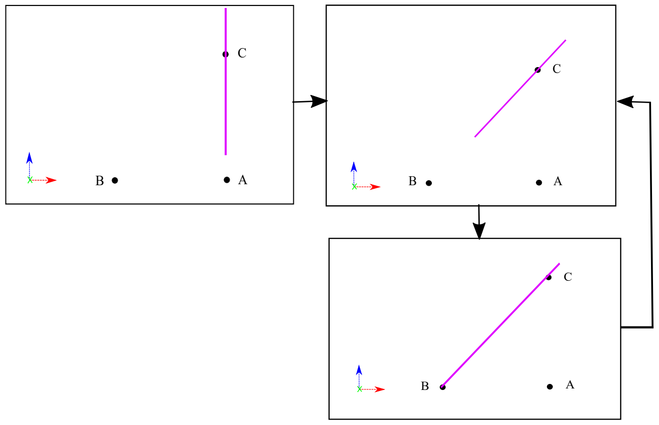

Fig. 6. Remote center-of-motion for maneuvering the foaming tool, indicated with purple color.

patient. Fig 6 shows a block diagram of how the remote center-of-motion works. Assuming the robot is going to move the tip of the foaming tool from point 'A' to 'B', and the tool is inserted through point 'C'. Moving the tip from point to point are separated into two steps: rotating the tool on the RCM point 'C', and translating to the target location, point B. First, we calculate an orientation from point 'C' to point 'A'. Then we rotate the end-effector based on the entry point 'C' by solving inverse kinematics of the orientation. Next, since the tool is rotated toward to the target location 'B', we only need to move the tip on the tool along this direction to reach the target point. This can be simply done by deriving the inverse kinematics of the tip on the tool. Lastly, we update the tool offset from the end-effector flange to the entry point because we will use it for the next rotating movement. The entire process will be done until the robot has done traveling all the point within the trajectory generated by a coverage path planner.

C. Control loop

Fig. 7 shows the block diagram of the control loop for moving the robot. The robot is first sent to an entry point and use it to operate remote center motion. The controller will read the path file generated by coverage path planner, loading all the points within trajectory into the program. Both translation and orientation for all point are computed before controlling the robot. In the control loop, as mentioned in section II-B, the robot first rotates on the entry point then does translation along its direction. Kinematics and Dynamics Library (KDL) in Open Robot Control System (OROCOS) is used to transform the task-space to robot joint space for both movements. For the translation part, based on the distance of end-effector moved, we update the tool offset, In other words, we update a new location for operating next remote center motion. IIWA stack helps to control the robot to follow a certain trajectory. A position feedback control is applied to makes sure the robot is moving to an exact location. The entire loop will continue until the robot completes traveling all the points within the path.

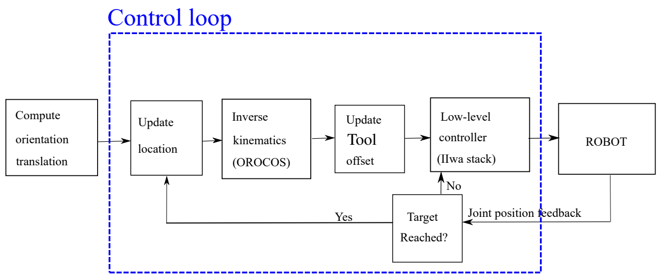

Fig. 7. Control loop.

III. Innovations

Current DCS is mainly performed by combat medics to control hemorrhage. The outcome relies heavily on the operating medics and it varies by their experience, physical and mental state on the battlefield.

Our key innovation is the realization of RAS for DCS in the battlefield to maintain a high-quality outcome of hemorrhage control. By an implementation of coverage path planning combined with the remote center of motion, the foam injection through the belly button which covers the whole bleeding abdominal area inside soldiers' body can be realized, which will have a significant positive influence on the 30% of trauma-related deaths. [10]

IV. Experiments

A. Experimental setup

The Experimental setup for implementing our robotic system is shown in Fig. 8. A 7 degree-of-freedom(DOF) lightweight arm (KUKA) is used as the surgical robot. A foaming tool is mounted on the end-effector of the robotic arm. A laptop served as a position controller for computing coverage path planning and sending position command for moving the robot. The laptop is connected with a KUKA sunrise controller with Ethernet cable, controlling the robot. Finally, a cardboard with a specific cut shape to be our test subject is placed on a desk. The length of the tool is 0.397 m, and the entry point is set at a location (0.4, -0.55, 0.35) m in the robot frame. In other words, the robot will be maneuvered on the location to perform coverage path movement on the target area.

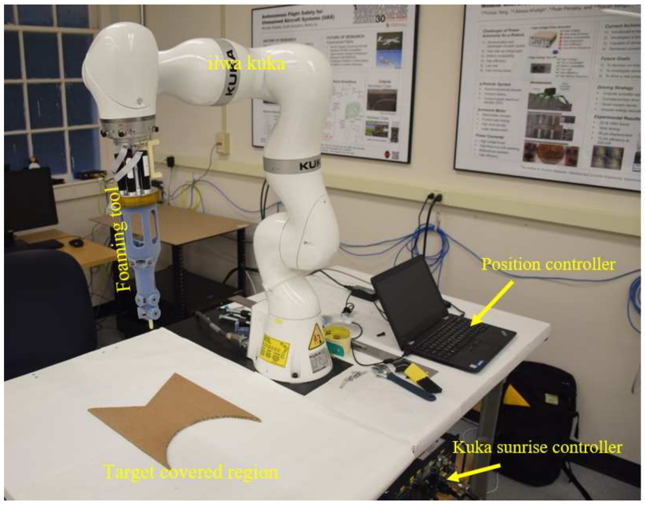

Fig. 8. Experimental setup.

B. Experiments description

The dimension of the target cardboard is 320×220 mm. It has linear upper and lower edges. The left margin is shown in the 8 is formed by 2 intersected straight lines to represent the nonsmooth critical point's situation and the left margin is formed by a half circle to simulate smooth critical point's situation, which makes the cardboard a minimum complexity demo to verify our algorithms. The procedures of the experiments can be divided into three parts: i) Generate path to covering the target area using coverage path planning in MATLAB (MathWorks, Natick, Massachusetts, The USA.) ii) Simulate the robot motion in GAZEBO to verify the coverage path with performing remote center-of-motion using C++ in ROS. iii) Implementing on the KUKA arm to further evaluate the simulation results.

C. Experiments results

1) Coverage path planning:

The Fig. 9 below demonstrates the generated final path by applying our CPP algorithms. There are 6 critical points found in the configuration space, which are marked in red in the Fig. 9. Only the right one point in the central area is derived from a smooth boundary. And all other 5 are non-smooth critical points. The cells are divided and ordered following the column-wise sweeping from left to right. Note that the orders of cell #4 and #5 are correct because the Matlab plot function will reverse the positive direction of the y-axis. You can regard the Fig. 9 as an upside-down figure and the original figure is the one applied to the algorithms rather this one. It does follow the right consequences. The full path of the inter-cell exhaustive walk is $\{1, 3, 2, 3, 4, 3, 5\}$, which meets our expectation. The green dots are the preserved positions in the zigzag-shaped boustrophedon in-cell paths from 1 out 15 in column and row, combined with the linear movements between cells with a step motion that equals to 20 mm. The physical meaning of this path is the trajectory the tip on the tool, which is mounted on the ending flange of the KUKA arm, will follow.

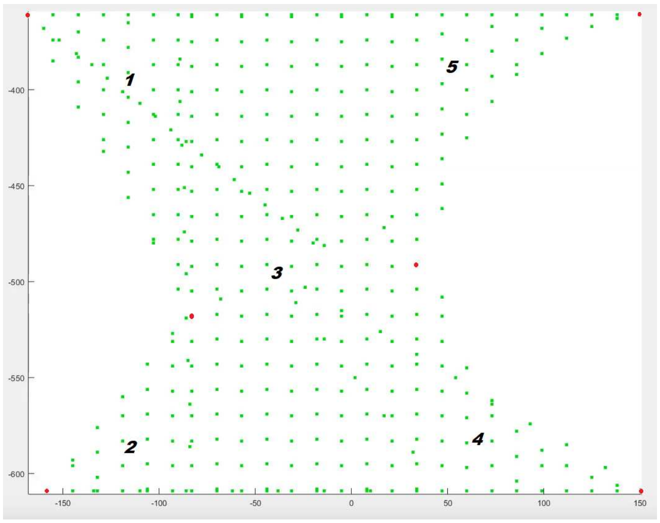

Fig. 9. CPP result in Matlab.

2) *Simulation:*

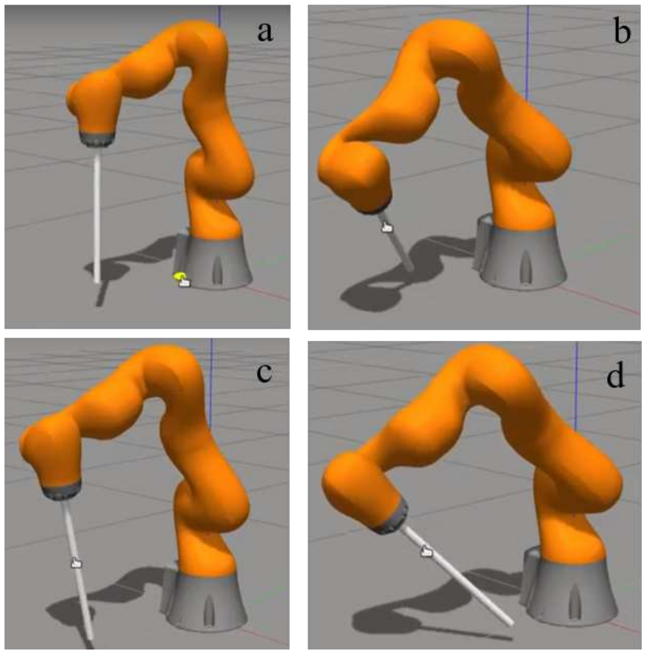

Fig. 10. Simulation result in GAZEBO.

Fig. 10 shows the simulation scene. A cylindrical rod, 0.397 m, was attached to the end-effector as the foaming tool. The robot in the GAZEBO was controlled by the program through ROS. Fig. 10.a. shows the initial position of the robot, inserting the foaming tool into the entry point. Fig. 10.b. shows the robot performing coverage path movement for the first cell. Fig. 10.c. shows the coverage path motion on the third cell. Fig. 10.d shows the robot covering last cell area. It took 13:43 minutes to travel entire coverage path with 10 Hz in the control loop. Notice that the location of a cursor in Fig. 10.b is the entry point. By performing the coverage path movement, the robot always maneuvers the rod on that location, namely, performing remote center-of-motion.

3) *Robot implementation:*

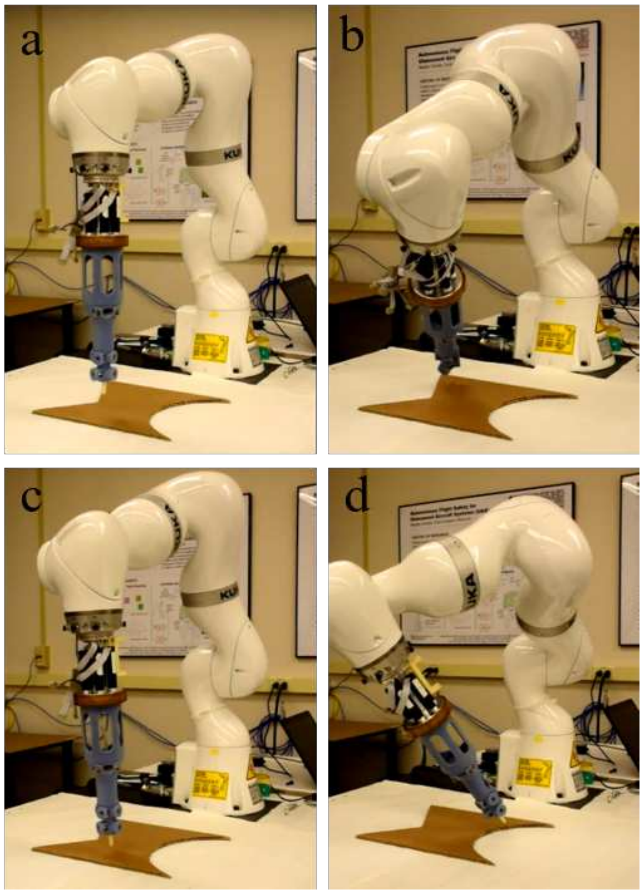

Fig. 11. Implementation on KUKA robot.

Fig 11 is the result of an implementation on a lightweight robotic arm. Since we only implemented position control, We downsampled the coverage path in section IV-C2 in order to save time for traveling all the points in the original path. In addition, the frequency of the control loop was set to 2 Hz. Fig 11.a shows the initial position of the robot, inserting the foaming tool into the entry point. Fig 11.b shows the robot performing coverage path movement for the first cell. Fig 11.c shows the coverage path motion on the third cell. Fig 11.d shows the robot covering last cell. It took 8:16 minutes to finish the process. We evaluated the path by observing the movement of the tip on the tool, checking whether the tip follows the command moving inside the cardboard.

V. Discussion and Future Work

The CPP results shown in section IV are optimal for our minimum complexity case. However, the final path will definitely become not optimal when we increase the free spaces complexity. If we implement DFS algorithm based on the cell number given by MATLAB bwlabel function without considering the distance-to-go, the tip of the tool may detour a lot. In our case, the distances between every 2 cell are similar. But if we stretch the cell #2 to the left by 1000 mm, the optimal path should be a path went to cell #2 last rather than be the 3rd to cover in our solution. What's more, if we shrink the right margin of cell #2 to 20 mm, the inter-cell movement for the trivial cases will lead the tip of the tool to intersect too much to the obstacles by applying a center to center linear motion between cells. In the future, we'll add a heuristic cost function for the DFS algorithm to optimize the distance of final path and ask the tip of the tool to move following a center-shared edge-center path between cells rather just a center-center linear path in order to avoid collision with obstacles.

Our main target patients for this project are wounded soldiers on the battlefield. Some cavities inside their body are bleeding heavily. The shape of our cardboard in the experiment is designed because we need to confirm our algorithms for the smooth and non-smooth critical points, but it should be circular shapes at most cases in the real world in that most cavities inside the body are ball-shaped. When it comes to Morse function selection, $h(x,y) = \sqrt{x^2+y^2}$ is a better choice than $h(x) = \lambda$ because it's apparent that the spiral-curved path traversal a circular area better than linear path. The curve can be better understood by watching Fig. 12 below.

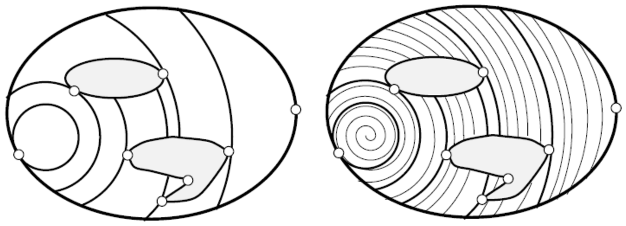

Fig. 12. Morse function: $h(x,y) = \sqrt{x^2+y^2}$

We don't have any feedback source like an endoscopic camera so all the configuration spaces are predefined. In reality, there should be less self-expanding foam above the organs than the pure cavity space inside the body to generate an even pressure, which means the foam injection speed should be smaller when foaming the organ area. For general CPP problems, these areas are regarded as obstacles while we cannot treat them in the same way. In the future, the foam injection speed will be a variable rather than a constant according to the organs distribution in the target cavity.

In the experiment, we only maneuvered the robot with a point to point command, without linear interpolation between two points. In other words, the movement between two points can't guarantee to be stray. The more distance between two points, the more likely the path isn't stray. For the translation movement in remote center-of-motion, even though we expected the tip on a tool will move along the direction to the target point, we observed a curve movement on the tip. This indicates that a linear interpolation is needed in the translation movement in order to operate remote center-of-motion.

We applied only position control in the experiment, meaning we did not consider velocity profile when the robot moved. Since the controller did not plan for acceleration and deceleration moving between two points, the robot can only use the default velocity planner in the sunrise system, producing a non-smooth movement on the tool. This might affect the accuracy of the trajectory as well. In the future, a velocity planner for controlling the robot is needed. In addition, our program could not control the robot speed so the robot can only move relatively slow during the experiment; therefore, the time cost for overall travel through the coverage path is much longer than the GAZEBO simulation. In clinical usage, the speed is crucial since it indicates how fast the foam will cover entire target region to stop a hemorrhage. If the robot can only operate at low speed, the foam can't cover the area before the patient loses too much blood.

Lastly, in the experiment, we did not have an external device to measure the trajectory nor the motion on the entry point. Therefore, whether the movement was correct could not be evaluated in this experiment. Even though we could see the tool was moving across the target region, we did not know the accuracy of the path and how much trajectory error was in our system. In addition, we assumed our program operated remote center-of-motion in the entry point; however, we could not tell the motion was always fixed at the specific point during the experiment. In the future, we will need to have the measurement system to verify the accuracy of the trajectory in our system.

VI. Conclusion

In this work, we developed a control strategy for a foaming robotic system on damage control surgery. Using coverage path planning and remote center-of-motion, we demonstrated how the system maneuvered the foaming tool and travel trajectory through an entry point. The results indicate that this study exhibits the potential to improve the coverage of the target region of a robotic-assisted surgery system in a foaming task. Our future work will include enhancement of the performance in the low-level controller, as well as evaluation of the deviation of the coverage trajectory.

References

[1] L. A. Melniker, E. Leibner, M. G. McKenney, P. Lopez, W. M. Briggs, and C. A. Mancuso, "Randomized controlled clinical trial of point-of-care, limited ultrasonography for trauma in the emergency department: the first sonography outcomes assessment program trial," *Annals of emergency medicine*, vol. 48, no. 3, pp. 227–235, 2006.

[2] B. H. Waibel and M. M. Rotondo, "Damage control surgery: it's evolution over the last 20 years," *Revista do Colegio Brasileiro de Cirurgioes*, vol. 39, no. 4, pp. 314–321, 2012.

[3] G. Mercante, P. Ruscito, R. Pellini, G. Cristalli, and G. Spriano, "Transoral robotic surgery (tors) for tongue base tumours," *Acta otorhinolaryngologica italica*, vol. 33, no. 4, p. 230, 2013.

[4] A. P. Rago, U. Sharma, M. Duggan, and D. R. King, "Percutaneous damage control with self-expanding foam: pre-hospital rescue from abdominal exsanguination," *Trauma*, vol. 18, no. 2, pp. 85–91, 2016.

[5] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*. Springer, 1998, pp. 203–209.

[6] J. Milnor, *Morse Theory.(AM-51)*. Princeton university press, 2016, vol. 51.

[7] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, 2002.

[8] A. Manitius, "Optimization and nonsmooth analysis (frank h. clarke)," *SIAM Review*, vol. 27, no. 2, pp. 288–291, 1985.

[9] R. M. Haralick and L. G. Shapiro, *Computer and robot vision*. Addison-wesley, 1992.

[10] M. Saleh, "Management of uncontrolled hemorrhagic trauma: State of the art," *Ain-Shams Journal of Anaesthesiology*, vol. 8, no. 1, pp. 10–13, 2015.