**RELATED WORKS**

*Path Planning in 3D space*

Obtaining optimum path in 3D space is much costly than in the 2D space, optimum path searching algorithms in 2D space such as A star is no more feasible in 3D space due to its unacceptable computational time. Since we assume that we have the pre-known information of the whole workspace, that is, a mathematic representation to describe the workspace. We chose the sampling-based method in our work since it gives us a moderate result with low computational cost. This kind of method usually samples the environment as a set of nodes, or cells, or in other forms. Then map the environment or just search randomly to achieve a feasible path.

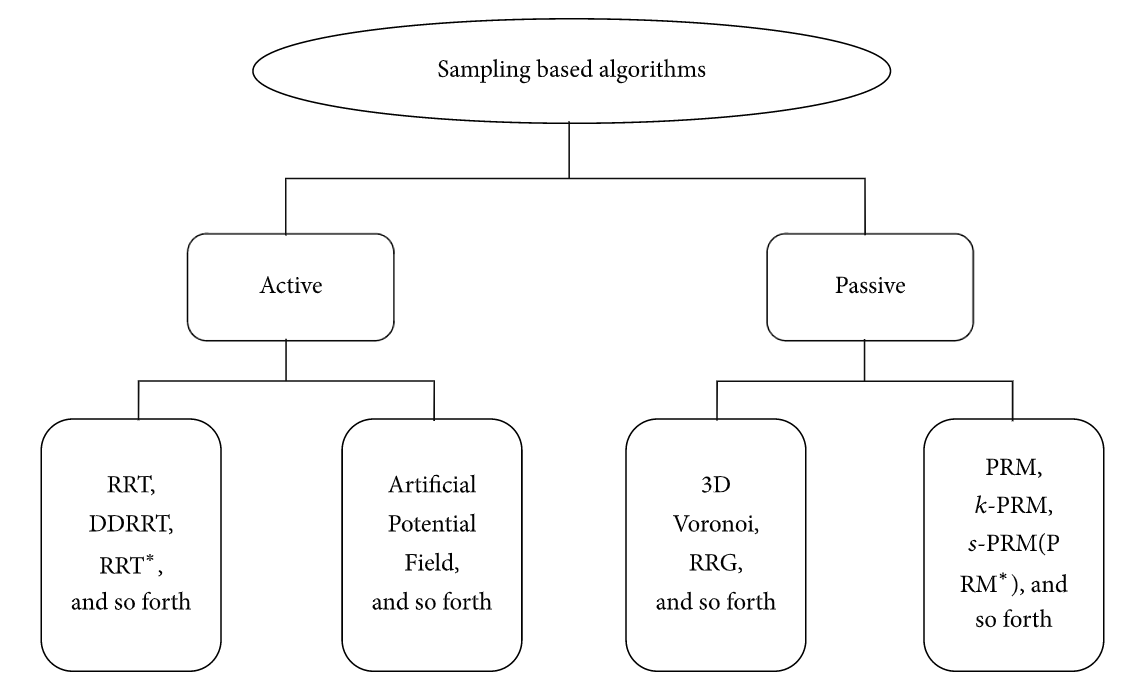


Fig. 1 Sampling-based path planning [1]

Fig. 1 showed the main classifies of the sampling-based algorithms. The sampling-based algorithms could be divided into two more detailed categories, active and passive. Active means algorithm such as rapidly exploring random trees (RRT) [2] which can achieve the best feasible path to the goal all by its own processing procedure. Passive means algorithms such as Probabilistic Road Maps (PRM) [3] only generate a road net from start to the goal, thus a combination of search algorithms to pick up the best feasible path in the net map where many feasible paths exist.

3D Voronoi [4] forms a 3D obstacle free network based on pre-known knowledge of the whole environment. Voronoi algorithm works in an optimization way to follow the varying edges to ensure equal distance. Artificial Potential Field algorithms [5] also needs the whole workspace sampling information to escape from local minima.

In this work, we chose the RRT star algorithm as our path planner, RRT star is a variation of RRT algorithm which provides acceptable path while keep the computational cost low.

**METHOD**

The method we used is based on a lumped model for the needle. The lumped model considers the needle as a discrete structure composed of several rigid bars connected in series by helical springs, referred to as virtual joints. The model will not only concern a needle in free space but also the contact with tissue.

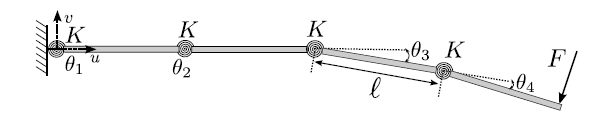


Fig. 2 lumped needle model

Fig. 2 showed the lumped needle model in free space is shown. The needle of length *L* is discretized into *n* undeformable weightless bars, each of which has the length *l* so that . In Fig. 1, *n* = 4. isthe angle by which the bar *i* rotatesrelative to the bar *i−1* along an axis perpendicular to the plane. *Fr* is the force applied to the needle tip along an axis normal to the unbent needle which means all. At each joint, two bars are connectedthrough a helical spring. All helical springs have the samestiffness of K.

*Forward Kinematics*

For the example in Fig. 2, the torque  is generated by a force at each joint *i*, where *Fu* and *Fv* are the horizontal and vertical components of *F* respectively. The torque could be represent as following:

(1)

And thus, for the needle model composed of n links, the torque at the joint *i* could be calculated by:

(2)

where 1 ≤ i ≤ n and the Jacobian matrix is:

(3)

The torques obtained by (2) are then divided by the joint stiffness K to calculate the angular displacements of the joints, i.e., . The Cartesian position of joint *i* in the normal and axial directions with respect to the straight needle are denoted by and respectively. And we can obtain them as following:

(4)

is defined to be the rotation angle of link *i* relative to link *i−1*. Equations (3)-(4) stated that the Cartesian position of joint *i* is the position of joint *i−1* plus the displacement caused by link *i*. And the deflection of the needle tip will be .

*Determining the Joint Stiffness*

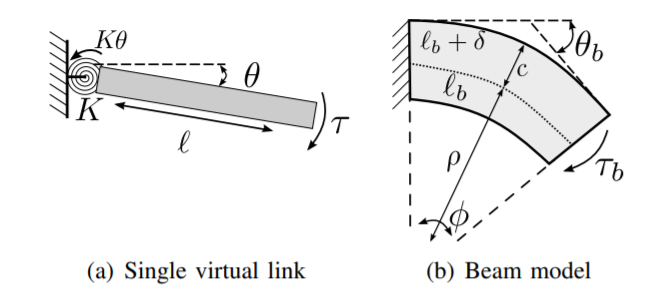


Fig. 3 beam model for determining the joint stiffness

The second step is to determine the stiffness *K* of the joints that are used in the lumped needle model. In Fig. 3(a), the torque τ is applied at the end point of this rigid link with length *l*, and it is connected to a fixed wall through a helical spring of stiffness *K*. The reaction torque *Kθ* at the clamping point (*θ* is the deflection angle) is equal to *τ*, and we could get:

(5)

For Fig. 3(b). The dotted line means its neutral axis with a radius of *ρ*, the length of short beam is the same as that of the virtual link *l* in Fig. 3(a). *c* denotes the radial distance between the neutral axis and the outer radius. is the outer arc length. The bending angle of the beam is when it is under a constant bending moment , which is its angle of curvature φ. The strain is the amount of elongation divided by the initial length. And it could be calculated as with a radial distance *y* to the neutral axis, where E is the beam Young’s modulus and I is its second moment of inertia. At the outer arc, the strain is by the definition, and we could obtain that:

(6)

(7)

Combining (6) and (7) and known that , we can now get the relation between the bending moment and the resultant bending angle:

(8)

Therefore, when the moments applied on the virtual link in Fig. 2(a) and on the beam model in Fig. 2(b) are the same (). The stiffness that gives the same bending angle () in the two models with beam length could be defined as:

(9)

*Tissue Interaction Modeling*

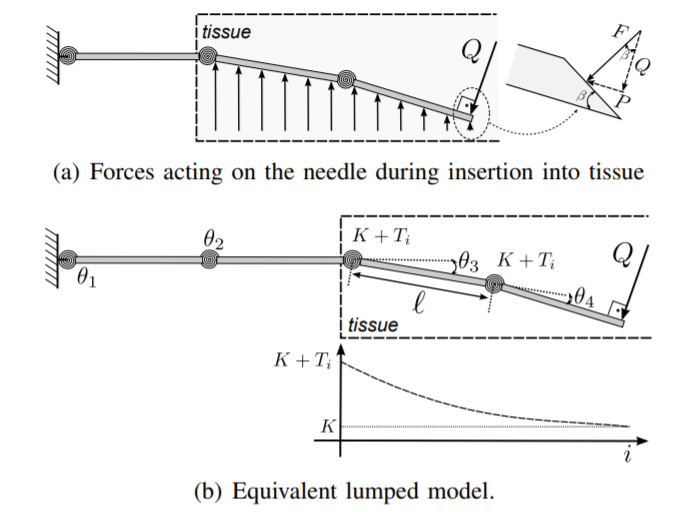


Fig. 4 tissue interaction model

The reaction of tissue is modelled by increasing the stiffness of the virtual joints in the needle model that act against the deflection of the needle. Needle insertion into tissue means a force F applied perpendicularly to the needle beveled tip as shown in Fig. 4(a). The force F can be decomposed into a transverse force Q which is perpendicular to the last bar of the last virtual joint of the needle, and a force P parallel to the last virtual joint of the needle depending on the tip bevel angle β. And Q causes the needle to bend. If the needle will not substantially deviate from a straight line, the effect of P on deflection will be neglected. These forces are typically assumed constant during the insertion and are insensitive to the overall bending of the needle. The force applied at the needle tip in (2) could be calculated by:

(10)

In Fig. 4(a), the needle bends and increasingly compresses the part of tissue under the needle as it is inserting into the tissue. The compressed tissue acts as a support and applies reaction forces against the needle. For an elastic tissue, the forces are proportional to the tissue compression. Due to the increasing tissue compression forces, it is reasonable to assume that the tissue applies the highest reaction forces at the needle entry point and the least reaction force at the needle tip. Therefore, Fig. 4(a) shows the tissue reaction force to be at its maximum at the needle entry point and at its minimum at the needle tip.

In our model, the tissue reaction forces above act as helical springs of stiffness  added to the initial nominal joint stiffness *K* calculated by (9). Thus, the joints that are inside the tissue have stiffness . For a joint *i* outside the tissue, . For simplicity, we assume the needle bending outside the tissue to be negligible. The stiffness  is calculated as:

(11)

where and depend on the mechanical properties of the needle and tissue. From (11), the angular displacements of the joints become:

(12)

The radius of curvature ρ of the needle shape at the joint i is:

(13)

In (11),  depends on the stiffness of the tissue. It determines the amount of needle deflection for a given load applied at the tip: the higher is, the less the needle deflection is; can be tuned to obtain a specific radius of curvature. Since *l* (n − i + 1) < L ∀n and L < 1, the stiffness of the tissue is inversely proportional to . Therefore, the radius of curvature is proportional to . In other words, for low values of , the shape of the needle is highly curved. This means that the needle tip can reach different deflections at the same depth by following different paths.

*Simulation Insertion Loop*

This loop simulates the needle insertion for a given set of *Q*, and . Q should be given by measurement and and could be given by empirical or model. The insertion begins with the straight needle placed outside the tissue, thus = 0 and = 0, where is the number of virtual joints that are inside the tissue. The needle is pushed by *l* units of length into the tissue (= + 1). At each increment of , the joint *i* that is inside the tissue acquires a stiffness of *K* +. The load Q is subsequently applied at the end of the last virtual joint and the Cartesian position of the virtual joints is calculated through the forward kinematics. The process is repeated until ≥ where = is the number of joints that need to be inserted to reach a desired insertion depth *d*. Once the needle reaches d, the insertion loop is finished, and another interaction of the numerical search loop begins.

*RRT\* Path Planner*

Our path planner is based on the RRT star [6] algorithm. Finding an obstacle-free path in 3D space is usually costly. The reason we choose RRT star algorithm is that it could provide us an acceptable result while runs fast. RRT\* is an optimized version of RRT. The basic principle of RRT\* is the same as RRT, but there are two key differences. First, RRT\* records the distance each vertex has traveled relative to its parent vertex. Second, RRT\* adds is the rewiring of the tree. These additions enable RRT\* to have the potential to obtain a shortest obstacle-free path from the starting point to the target point with the cost of computational time.

The first step of the planner is to Sample N points inside the tissue area. In our experiment, we set N to 300, it’s a moderately value that could give us an acceptable result while keeping the planning speed fast. Higher value of N enables the planner to output a more accurate path with lower cost of moving. The larger the number of N, the higher the accuracy of the planning result. Theoretically, the RRT star algorithm could generate the global optimum path when N goes infinity. However, when N goes up, the computational time will be significantly increased. If the N is not adequate, the planner may fail to output an obstacle-free path.

When sampling a new point, the new point needs to be detected whether this point is inside an obstacle area, if it is, it should be given up and sample a new point. At the same time, the calculation of the cost of the potential path by each new point is being executed. In order to reduce the computational cost, the RRT star algorithm will not estimate the cost of each node until the sampling step end. Instead, the planner will evaluate the new node once it's been generated.

In order to evaluate the cost of the new point, the planner will firstly find the nearest neighbors of this point. The nearest neighbors of a point mean the points located within a threshold distance to that point. In our experiment, we set the threshold to 50mm. Higher threshold could slightly increase the accuracy of the result, while the computational speed will decrease.

For each point in the nearest neighbors of the new point, the planner will execute an obstacle detection which is executed by evaluating whether the line from the new point to this point collided with the obstacles. And this is implemented by a sampling-based way in our implementation, we will evenly sample one thousand points on the line to see whether these points lie on the obstacle areas. If not, the planner will calculate the total cost of the path from the starting point to this near point. The way of calculating the cost of a new point is to add the cost of this near point and the distance between the new point and this near point. By doing this calculation in each iteration, the cost of all points could be obtained. However, in order to obtain a smoother path, when the distance from the new point to the near point is higher than a threshold, we will choose a point between the near point and the new point as the next point in this path.

After the above steps, we will get the moving cost from the starting point to all the new points. And the potential paths could be established by connecting each point to the target points, however, if obstacle existed between the point and the target point, this path will be given up. At last, the planner will evaluate the moving cost of all potential path. The cost of a path could be calculated by adding the cost of the point and the distance between the point and the target point. Finally, the planner will choose the path with the lowest cost as the output.

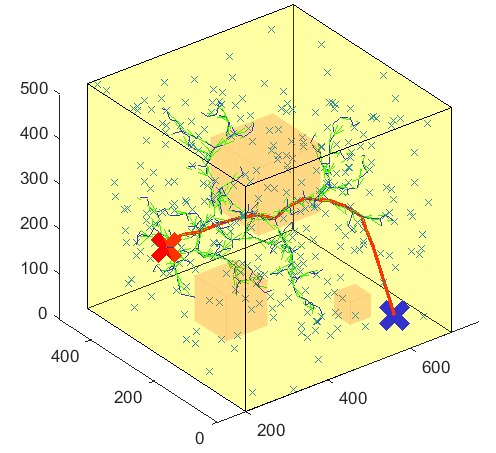
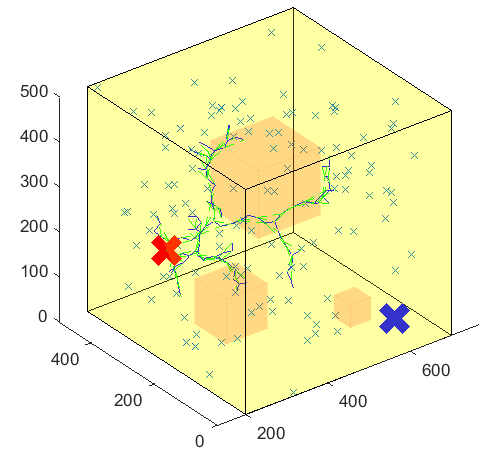
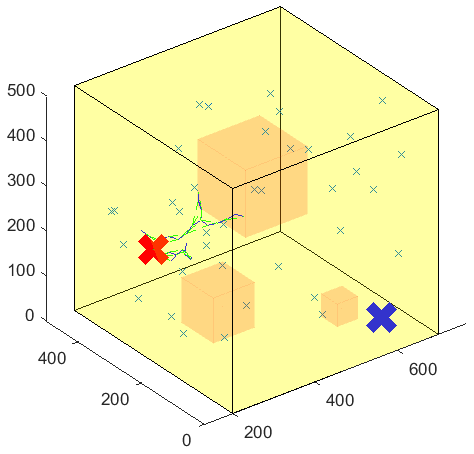


Fig. 5 example of execution of the RRT\* planner

Fig. 5 showed an example of the execution process of our RRT\* planner. The red cross is the starting point and the blue cross marked the target point. The little crosses are the newly sampled points. It could be seen in the figures that the number of points is keep increasing as the planner searching the path. At the same time, the green lines marked the paths of the nearest neighbors of each point being evaluated, and the black lines mean the potential paths being chosen by the planner. At last, an optimum path in all potential paths, the red line, is being chosen as the output of the planner.

[1] Liang Yang, Juntong Qi, Dalei Song, Jizhong Xiao, Jianda Han, and Yong Xia, “Survey of Robot 3D Path Planning Algorithms,” Journal of Control Science and Engineering, vol. 2016, Article ID 7426913, 22 pages, 2016.

[2] LaValle, S. M., ‘Rapidly-Exploring Random Trees: A New Tool for Path Planning’, TR 98-11, Computer Science Department, Iowa State University, Oct. 1998.

[3] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high dimensional configuration spaces,” IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.

[4] V. A. Luchnikov, N. N. Medvedev, L. Oger, and J.-P. Troadec, “Voronoi-Delaunay analysis of voids in systems of nonspherical particles,” Physical Review E, vol. 59, no. 6, pp. 7205–7212, 1999.

[5] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, Book Section 29, Springer, New York, NY, USA, 1990.

[6] Karaman, Sertac, and Emilio Frazzoli. "Incremental sampling-based algorithms for optimal motion planning." Robotics Science and Systems VI 104, 2010.