The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Trajectory planning with task constraints in densely filled environments

Bogdan Maris, Debora Botturi, and Paolo Fiorini

Abstract— **In this paper the problem of computing a rigid object trajectory in an environment populated with deformable objects is addressed. The problem arises in Minimally Invasive Robotic Surgery (MIRS) from the needs of reaching a point of interest inside the anatomy with rigid laparoscopic instruments. We address the case of abdominal surgery. The abdomen is a densely populated soft environment and it is not possible to apply classical techniques for obstacle avoidance because a collision free solution is, most of the time, not feasible. In order to have a convergent algorithm with, at least, one possible solution we have to relax the constraints and allow collision under a specific contact threshold to avoid tissue damaging. In this work a new approach for trajectory planning under these peculiar conditions is implemented. The method computes off-line the path which is then tested in a surgical simulator as part of a pre-operative surgical plan.**

I. Introduction

A common robotic task is to plan a robot trajectory from an initial configuration to a desired configuration. In its standard form, the solution of the motion planning problem requires the computation of a collision free path for a moving body between start and goal positions.

Depending on the nature of the problem, we may be interested in any collision-free trajectory, or one that provides the minimum (or close to minimum) overall cost, where the cost of a trajectory may be a function of several factors. The most common factor to be minimized are time for traversal, traversal risk and visibility. Several approaches exist for generating such trajectories, and in the following some of them are reviewed. In recent years motion planning has been increasingly used in virtual environments and games [1], where contacts and deformations need to be taken into account.

In this paper, we are interested in computing a path for a rigid body moving in an environment densely populated by soft objects that can deform and that can be damaged by an excessive penetration. This requires that environment objects to be constrained and their reaction forces to be always balanced by the forces exerted by the moving body during its contacts with the objects.

Our research is motivated by the possible application of motion planning to surgery, where the environment is the patient's anatomy and the robot is a surgical instrument.

In particular, our scenario is Minimally Invasive Robotic Surgery (MIRS), where small incisions are used to introduce special instruments at the distal end of a long rod through a cannula into the body of the patient. The operation area is small and sometimes difficult to reach, and the freedom to move the instruments is limited. Moreover the visibility is restricted to the small field of view of the endoscopic camera. Tactile and contact sensations is totally different from open surgery. This makes very hard to choose and perform a proper trajectory without planning and motion indication.

This work was supported by the European Commission's Seventh Framework Programme under contract no. 248960 as part of the SAFROS (Patient Safety in Robotic Surgery) project.

Bogdan Maris, Debora Botturi, and Paolo Fiorini are with the Department of Computer Science, University of Verona, Italy. bogdan.maris@metropolis.sci.univr.it, debora.botturi@ieee.org, paolo.fiorini@univr.it

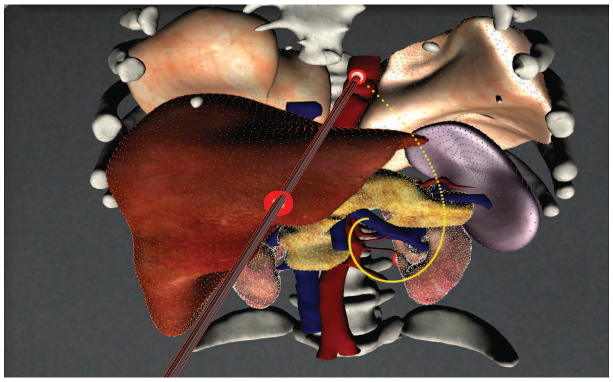

Fig. 1. Trajectory between deformable objects in a very populated environment.

In this paper we present a framework to compute the most comfortable trajectory for a surgeon that has to move a probe. Such a path has collisions and penetrations, because the obstacles are unavoidable in this environment, but it should damage the organs as little as possible. In our approach we represent the obstacles with a geometric model and we compute a trajectory that minimizes the collisions by considering both the geometric shape and the stiffness of each deformable object. We use a planning algorithm based on the minimization of the tool penetration (a rigid object) into the soft obstacles, while it is moving. The method computes off-line the path which is then tested in a virtual environment by using a surgical simulator developed in our laboratory (Fig. 6). The surgical simulator shows the feasibility of the trajectory to the surgeon.

After the introduction of the previous work done in the field of planning and modeling with deformable objects (Section II) we present in Section III the computation of the penalty function used in Section IV to find the trajectory. Section V describes the experimental results and tests we have carried out. Finally, Section VI presents the conclusions and an outlook of a possible continuation of this work.

II. Related Work

A. Planning with deformable objects

Although research on motion planning for rigid bodies has produced many practical results, there is not an equivalent body of work in the area of motion planning among deformable objects. One difficulty facing motion planning with deformable objects is having a (deformation) model that accurately represents the object physical properties, while preserving the efficiency of the planner. In fact, a planner that uses a physically correct deformation model can be very slow [2] and a planner that uses only geometric deformations can compute unnatural motions [3].

Workspaces studied in most standard planning algorithms, generally include only static and rigid obstacles.

A very good study and implementation of motion planning in medical interventions is done in [4]. In this book the problem of computing deformation to compensate for errors caused by soft tissue displacement during needle insertion is addressed. Their planner uses 2D images of the organs and the physical simulation of the deformations is done by using the finite elements method (FEM). The same book describes also the computation of a path for a steerable needle in deformable environments using again the FEM modeling on 2D images.

Frameworks such as Probabilistic Roadmap Planner and Rapidly-Exploring Random Tree (RRT), [3], [5], describe path computation in totally dynamic and deformable environments. [2], [6] deal with planning and compute physically correct deformations models. These approaches are not suitable to our problem because the main goal of those algorithms is to completely avoid collisions, and mainly they consider a deformable robot [7] that deforms to avoid compenetration. Moreover such expensive computations for solving mechanical models and generating collision detection data structures are too time consuming and are not needed for our goal. Generally they are applied to simple objects, such as a sheet of metal or a pipe-like robot. In [8] deformed distance fields are used to control the amount of deformation between non-penetrating simple flexible bodies.

A combination of probabilistic roadmaps with physical simulation of object deformations to determine a path that optimizes the trade-off between the deformation cost and the distance to be traveled using finite element theory for calculating the deformation cost is described in [9].

B. Surfaces modeling and the surgical simulator

The environment of our planner is based on a virtual model of the abdomen, implemented in a surgical simulator built in our laboratory.

The surgical simulator uses data from a computed tomography process to create the organs as clouds of points linked by linear springs. Applying a displacement to one or more points of the model results in changes in the length of the model springs, and this generates internal forces that deform the model. Mass spring models (MSM) have been used in medical simulations to simulate skin, fat or muscle [10] [11] [12]: their reduced computational complexity makes them a good choice for interactive simulation with haptic feedback. The main limitation of mass spring models is their lack of physical background that makes them difficult to calibrate. Another issue that arises using MSM is the realism of the simulation: it is known that mass spring formulation leads to correct simulations when the magnitude of deformations stays below 10% of the model size. Using the graphic processing unit present on recent graphic cards, we are able to provide to the user a virtual environment in which he/she can use a haptic device to move a probe along a given trajectory and feel the force exerted by the virtual tissue on the probe. Particular attention has been paid to the implementation to obtain update frequencies that are suitable to realistic visual and haptic feedback.

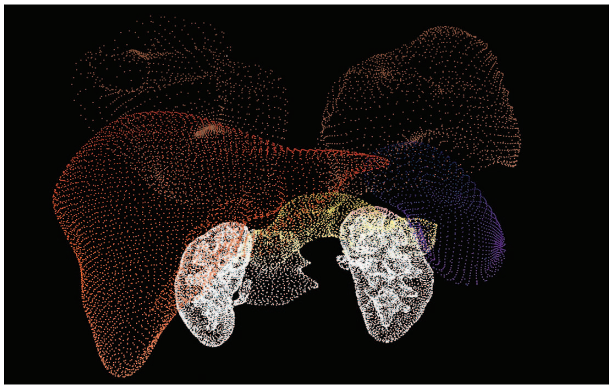

Fig. 2. The meshless representation of the environment.

The MSM model is described by a mesh of springs connecting a set of surface points, called surflets, and a set of internal points, called phyxels, where every point has a mass. The surflets and the phyxels describe the geometry of the objects.

Since the surgical simulator could be used to test a trajectory (once the trajectory is computed) and, while computing the trajectory it is difficult to simulate the deformation, we chose to use a meshless model instead of the finite elements method, considering only the set of surflets points.

In the computation of the trajectory, we will consider only statical deformations, quantified by the penetration depth, therefore, in our approximation, the contacts are local. Each state of the problem consists of the position of the tool and the penetration depth function at that point. The path is feasible when a collision does not damage the obstacle and providing that the obstacle deforms of an appropriate amount.

The point based representation of the objects (the meshless model) allows us reconstructing the scene in a more automatic and straightforward way, to do fast model resampling, and to easily compute collisions and penetration depth.

Following the method described in [13] point clouds can be easily generated from different 3D representations (CAD models, triangular surfaces, segmented medical images, implicit functions) thus our method can be generalized to many scenarios represented by real data sets. Fig. 2 shows a screenshot of the meshless model we have considered.

III. Collision Detection and Response

This section describes the collision detection module used by the planning algorithm to compute the penetration depth function, $PD(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}$. PD is the objective function to be minimized in the optimization process.

The detection of collisions in a virtual environment has been studied in different fields (among others: planning, computer animation) with the goal of making it as fast and accurate as possible [14]. The main task of these algorithms is to detect a collision and define the penetration depth (PD) among two objects.

The penetration depth function gives a measure of the amount of the collisions along a given discrete trajectory. To have the total amount of collision we added the value of the collision of the probe with every single object of the environment.

The first step of our approach was to compute the collision between the probe and only one organ for a given position of the tool.

To detect the collision and to compute the penetration depth function we used an algorithm derived from *expanding-polytope algorithm* (EPA) [15] and from [16].

The metric used by EPA algorithm to compute the penetration depth is in terms of Minkowski difference of two objects (or the translational configuration space obstacle TCSO). The Minkowski difference (Fig. 3 and 4) between two sets A and B is defined as:

$$A \ominus B = \{\mathbf{a} - \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\} \tag{1}$$

where $\mathbf{a} - \mathbf{b}$ is the vector difference of the position vectors $\mathbf{a}$ and $\mathbf{b}$.

Without loss of generality, let us assume that polytopes A and B are defined with respect to the global origin O. Thus, if the two polytopes intersect, then the origin O is inside of $A \ominus B$ and $PD(A, B)$ corresponds to the minimum distance from O to the surface of the Minkowski difference $A \ominus B$ [17]. Also we notice that if A and B do not intersect then O is outside of $A \ominus B$ and the distance between A and B corresponds to the minimum distance from O to the surface of A and B [16]. Therefore the unified computational framework based on Minkowski difference provides a continuum of the distance measure between the two objects as they alternate between separation and inter-penetration configuration. This characteristics will be very useful in the minimization process (next section).

The penetration depth of two inter-penetrating objects A and B is defined as the minimum translation distance that one objects undergoes to make the interiors of A and B disjoint. Formally, $PD(A, B)$ is defined as:

$$PD(A, B) = min\{\|\mathbf{t}\| : interior(A - \mathbf{t}) \bigcap B = \phi\} \tag{2}$$

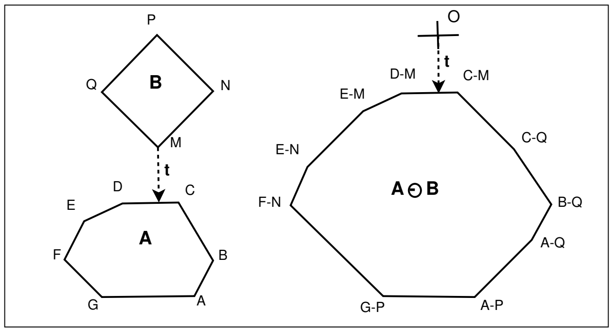

Fig. 3. The Minkowski difference of two disjoint convex objects in the two-dimensional Euclidean space. The norm of the vector $\mathbf{t}$ gives the Euclidean distance between the objects.

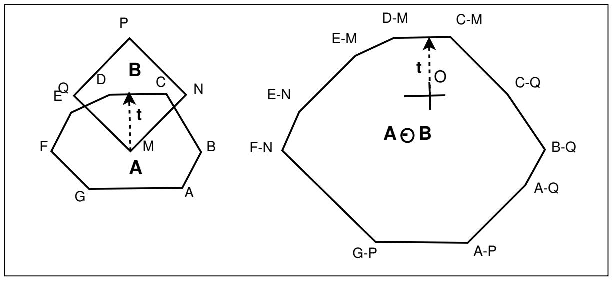

Fig. 4. The Minkowski difference of two intersecting convex objects in the two-dimensional Euclidean space. The vector $\mathbf{t}$ norm is the penetration depth between the objects.

Both algorithms we have cited and used do not need to compute explicitly the Minkowski difference of the two objects, but they iterate on convex sets formed with vertices from the Minkowski difference sampling through a *support mapping* function on demand. The *support mapping* function returns the farthest point of the Minkowski difference in a given direction. The support mapping function is defined for objects made of points or for some classes of nonpolytopal convex sets (see [18]). This types of objects gives the applicability of our algorithm to polytopes and more general convex sets. The first algorithm detects collisions by searching a separating plane between the Minkowski difference and the origin and the second algorithm computes the minimum norm point of the boundary of the Minkowski difference (see [15] for more details). The objects are approximated by their convex hull and this approximation suits very well our planner since we are interested in touching the objects as little as possible.

In our environment we have represented the organs as unions of convex meshless models given only by the points on their surfaces (surfels) and the probe as decagonal right prism (it approximates well enough, for our purposes, a cylindric probe, which is normally used).

A parameter e_o models the stiffness of each object o or may be used to implement constraints such as objects that must be avoided along the path.

With this set-up, $PD(\mathbf{p})$ (the value of the penetration depth in a given position $\mathbf{p}$) is the sum of all contributions PD^o to the penetration depth:

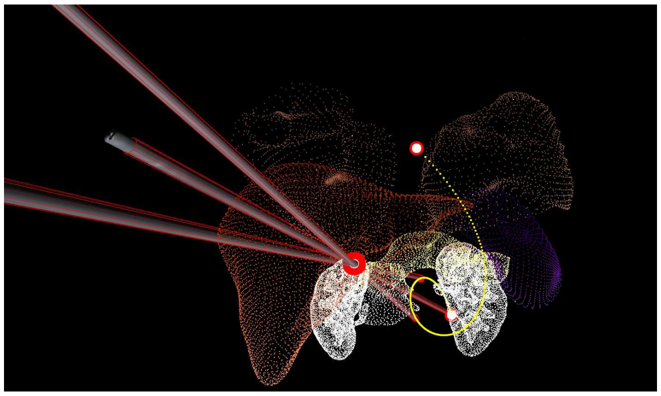

Fig. 5. Three positions of the tool along a computed trajectory. The red point represents the entering point and is always aligned with the tool. The red-white points are the starting position and the end position of the tool along the trajectory (yellow).

$$PD(\mathbf{p}) = \sum_{o \in O} e_o PD^o(\mathbf{p}) \tag{3}$$

where O is the set of all the objects of the scene. Therefore $PD(\mathbf{p})$ represents a penalty function which depends on the global collision relations, once the probe is placed at position $\mathbf{p}$.

IV. PLANNING THE TRAJECTORY

A. The trajectory

We can formulate the problem of computing the minimum penetration trajectory among deformable obstacles as a minimization problem, where the optimization variables are the trajectory parameters and the performance index is a measure of how much the probe collides with the objects on its path.

The mobile probe is constrained to move inside the abdominal environment and it has only four degrees of freedom, since it passes always through a fixed point (Fig. 5).

In our work we decided to focus on trajectories described by polynomials C of degree d, computed from an initial point P_0 to a final point P_f, $C : [0, 1] \rightarrow \mathbb{R}^3$, with $C(0) = P_0$ and $C(1) = P_f$.

For the polynomial function C we have choosen the Bèzier curve of degree d. The control points $P_0, P_1, ..., P_{d-1}, P_f \in \mathbb{R}^3$ represent, excluding the initial and the final point, our parameters.

The Bèzier curve lies within the convex hull of the control points, this providing a simple way of bounding the trajectory by just imposing constraints on the control points.

The minimization problem can be stated as follows:

$$min\ G(x_1, x_2, ..., x_{3d-3}),\ G : \mathbb{R}^{3(d-3)} \rightarrow \mathbb{R} \tag{4}$$

where $x_1, x_2, ..., x_{3d-3} \in \mathbb{R}^{3d-3}$ are the coordinates of the interior control points $P_1, P_2, ...P_{d-1}$,

$$P_i(x_i, x_{i+d-1}, x_{i+2d-2}) \in \mathbb{R}^3, i = 1, ..., d-1 \tag{5}$$

$$C(u) = \sum_{i=0}^{d} B_i^d(u) Pi \tag{6}$$

$$, u \in [0, 1]$$

$$B_i^d(u) = \binom{d}{i} (1-u)^d u^{d-i} \tag{7}$$

Equation (7) describes the Bernstein polynomials of degree d and

$$G(x_1, ..., x_{3d-3}) = \sum_{j=1}^{s} PD(C(u_i)), \tag{8}$$

is the objective function to be minimized, where $PD(C(u_i))$ is the function described by equation (3) and

$$u_i \in \{0, \frac{1}{s-1}, \frac{2}{s-1}, ..., 1\},$$

s represents the number of discrete time steps that we use to sample the movement of the probe. For each parameter u_i, $C(u_i) \in \mathbb{R}^3$ represents a position of the probe along the trajectory.

Note that we try to minimize the global collisions so we are looking for an optimum choice of the $d-1$ control points, which are our variables. At every step of the iteration we will compute the penetration depth along the entire trajectory described by the points $P_1, .., P_{d-1}$.

B. The minimization algorithm

To minimize the objective function given by (8) we need a minimization algorithm that can find the minimum of a function when the computation of the derivative is not feasible. The dimension of the problem depends on the number of the interior control points along the trajectory, therefore it depends on the polynomial degree it was chosen. For instance a 4 degree polynomial trajectory has two control points so the objective function will have the dimension 6 (each point has 3 coordinates).

We have chosen and implemented the downhill simplex method [19] to minimize our function, since it requires only function evaluation and matches very well with our geometric model.

The dimension of the Euclidean space where the downhill simplex algorithm operates is given by the dimension of the domain of the objective function. A d degree polynomial trajectory yields $3d-3$ coordinates of the interior control points; to construct a simplex in this space we need $3d-2$ points.

A simplex is the convex hull of a set of (n + 1) affinely independent points in some Euclidean space of dimension n or higher.

The downhill simplex method iterates on simplices. On every step the algorithm uses a $n+1$ dimensional simplex, where n is the dimension of the problem , reflecting, expanding or contracting this simplex by moving just one point, in

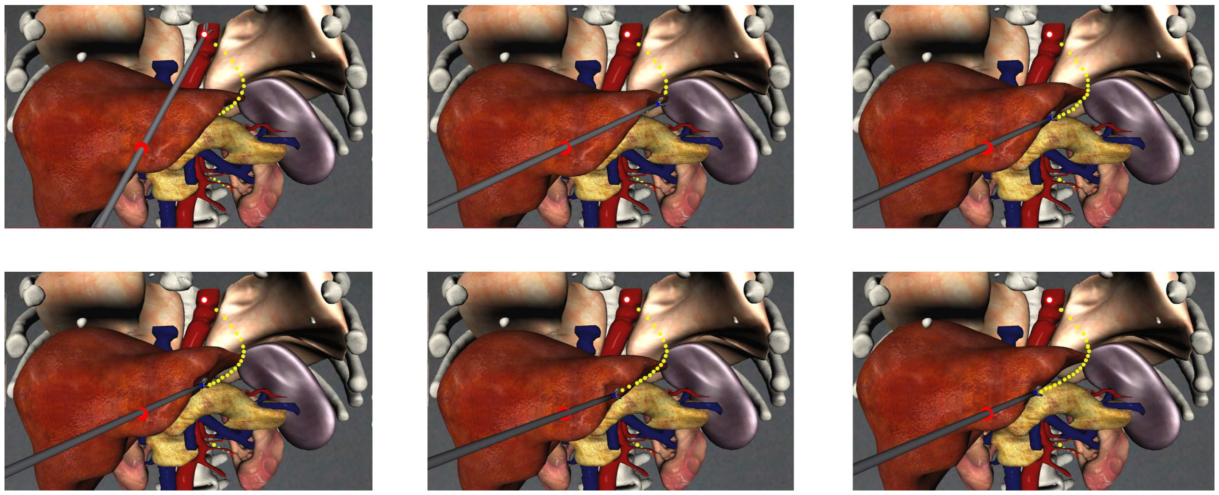

Fig. 6. A simulation of the trajectory with the MSM surgical simulator. The organs are moved and deformed while the probe is moving along the trajectory.

order to find a nearest position to the local minimum. The termination criteria must be imposed using a threshold on the magnitude of the distance vector that moves the point of the simplex.

Every point of the downhill simplex method represents a trajectory in our case. The algorithm starts with a $n+1$ arbitrary points. The initial $n+1$ points must be affinely independents to form a simplex. A method to chose these points, once we have a starting point $\mathbf{T_0}$ (or an initial trajectory), is to take other n points to be

$$\mathbf{T_i} = \mathbf{T_0} + \lambda \mathbf{e_i} \tag{9}$$

where $\mathbf{e_i}$ are n unit vectors and λ is a constant according to the problem's characteristic length scale (or it may be different for each vector direction).

The main advantage of this method is that we can decide to continue our search when we find a local minimum $\mathbf{T_{min}}$ by just reinitializating the simplex using $\mathbf{T_{min}}$ and other n values from the domain of our objective function as in (9).

We have used for our tests, as the initial trajectory, a random configuration of the control points inside a bounding box containing the environment. Some better initial guess could be chosen with the help of a physician.

This method moves the interior control points toward a configuration where the objective function is minimal.

Since the domain of our function is spatially limited inside the abdomen (we do not want to have a trajectory passing outside the abdomen), we had to limit the search inside a bounding box containing the organs.

V. Experimental results and simulations

We developed our test code in C++ and compiled with gcc (GNU Compiler Collection).

As the trajectory function, we used polynomials of degree from 3 to 7 in order to assure smoothness but also the capability to overcome more than one obstacle.

In our scenario each of the organs has about 3000 vertices.

Function PD is null outside the objects and because of this, the algorithm pushes the computed trajectory to the border of the obstacles, but not farther.

Table I shows a comparison of the performance results of different degree polynomial functions, using different numbers of discrete steps. We have used the same entering point (red in Fig. 5) and the same initial and final points (red and white) to compare the performance. The comparison is based on the minimal cost function and the computation time. According to the number of steps considered, the initial and the final values of the penetration depth function was scaled.

The initial PD value (4-th column) is the minimum value of the PD function when the initial trajectory $\mathbf{T_0}$ is as a random set of control points and the other n values compose an affinely independent set of points, as described by the equation (9).

As we expected, increasing the polynomial degree we increase the computation time but, in most of the cases, we achieve better results. There are no problems in handling an increasing number of objects, because the tests are only between the probe and single objects; of course, depending on the complexity of the environment, the optimization process may need more time to find a good solution, but the algorithm should scale well to bigger environments. However, we can decide a priori the degree, considering the total number of the objects and the computational time we want to achieve.

The feasibility of each trajectory was then checked manually with the surgical simulator (Fig. 6). Since the penetration depth values were small, the amount of the deformations perceived by the user was also small and, in this case, the geometric model fits well with the physical model implemented by the simulator. By having the force feed-back implemented in the simulator, the tests could be done by virtually guiding the user along the precomputed trajectory.

TABLE I

PERFORMANCE COMPARISON BETWEEN TRAJECTORIES OF DIFFERENT POLYNOMIAL DEGREES.

num	polynomial degree	discrete steps	initial PD value	PD value after minimization	number of function evaluation	time(msec)
1	3	50	290.702	165.344	326	160312
2	4	50	282.760	153.460	257	130047
3	5	50	252.268	135.461	363	175454
4	6	50	243.555	105.708	591	214266
5	7	50	229.885	107.961	755	242125
6	3	100	280.403	163.300	246	349328
7	4	100	288.315	140.787	436	419657
8	5	100	255.329	120.066	520	516609
9	6	100	285.796	92.348	681	670547
10	7	100	280.060	80.218	719	783171
11	3	150	291.246	169.687	340	523031
12	4	150	283.398	137.283	466	639047
13	5	150	275.463	128.130	329	392094
14	6	150	249.794	97.001	444	756500
15	7	150	236.260	86.485	558	980313

VI. CONCLUSIONS

In this paper we presented an approach to path planning in environments densely filled with non-rigid objects.

Using point based representation of the environment we are able to detect collision and rapidly compute the penetration depth and an associated penalty value, used by the minimization algorithm as a measure when searching for the best path.

An optimization method to obtain trajectories that minimizes the sum of the penalties is used. The planner developed computes collision free trajectories or trajectories with soft object interaction if the free trajectory is not feasible.

In our simulation we imposed constraints regarding the movement of the mobile probe (it passes always through a fixed point) and constraints on the domain of the objective function (the interior control points move only inside the bounding box surrounding the organs), together with constraints on the objects themselves (each penetration depth value is multiplied with a parameter representing the stiffness of an organ).

The main contribution of this work is the use of a geometric description of the environment that goes straight to the penetration function used by the optimization algorithm to compute a feasible path.

Optimization is a powerful framework for formulating and computing motion plans that maximizes the probability of successfully achieving clinical goals while minimizing tissue damage and other negative side effects.

Our results encourage the use of such method for the special problem class we are addressing, i.e. MIRS (Minimally Invasive Robotic Surgery) scenario.

REFERENCES

[1] M. C. Lin and S. Gottshalk, "Collision detection between geometric models: a survey," in *IMA Conference on Mathematics of Surfaces (San Diego (CA))*, vol. 1, 1998, pp. 602–608.

[2] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki, "Deformable volumes in path planning applications," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 2290–2295.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Probabilistic roadmap motion planning for deformable objects," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2002.

[4] R. Alterovitz and K. Goldberg, *Motion Planning in Medicine: Optimization and Simulation Algorithms for Image-Guided Procedures*. Springer Publishing Company, Incorporated, 2008.

[5] S. Rodriguez, J.-M. Lien, and N. M. Amato, "Planning motion in completely deformable environments," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006.

[6] F. Lamiraux and L. Kavraki, "Planning paths for elastic objects under manipulation constraints," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[7] R. Gayle, M. Lin, and D. Manocha, "Adaptive dynamics with efficient contact handling for articulated robots," in *Proceedings of Robotics: Science and Systems*, 2006.

[8] S. Fisher and M. C. Lin, "Deformed distance fields for simulation of non-penetrationg flexible bodies," in *IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2001.

[9] B. Frank, M. Becker, C. Stachniss, W. Burgard, and M. Teschner, "Efficient path planning for mobile robots in environments with deformable objects," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2008.

[10] D. Terzopoulos and K. Waters, "Physically-based facial modeling, analysis, and animation," *Journal of Visualization and Computer Animation*, vol. 1, no. 1, pp. 73–80, 1990.

[11] K. Waters and D. Terzopoulos, "Modeling and animating faces using scanned data," *Journal of Visualization and Computer Animation*, vol. 2, no. 2, pp. 123–128, 1991.

[12] J. Mosegaard, P. Herborg, and T. S. Srensen, "A gpu accelerated spring mass system for surgical simulation." *Studies in health technology and informatics*, vol. 111, pp. 342–348, 2005.

[13] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Computer Graphics and Visualization*, vol. 8, no. 4, 2002. [Online]. Available: citeseer.ist.psu.edu/alexa02computing.html

[14] F. Thomas and C. Torras, "3d collision detection:a survey," *Computer and Graphics*, vol. 25, pp. 269–285, 2001.

[15] G. van den Bergen, *Collison detection in interactive 3d environments*. Morgan Kaufmann Publishers, 2003.

[16] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three space," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 4, Mar 1987, pp. 1883–1889.

[17] S. Cameron, "Enhancing gjk: Computing minimum and penetration distances between convex polyhedra," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 1997, pp. 3112–3117.

[18] E. Gilbert and C.-P. Foo, "Computing the distance between general convex objects in three-dimensional space," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 53–61, Feb 1990.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.