Fei PENG, Yao ZHAO

Huazhong University of Science and Technology

Triangle Sampling Path Planning in Environment with Danger Zones for Assembly/Disassembly

Abstract. *To plan a feasible path in the environment with danger zones for assembly/disassembly, a RRT-based path planning algorithm was presented. The algorithm addressed a new density avoided sampling function based on kurtosis coefficient for planning in the free configuration space, and introduced a random sampling function which obtains random state from the triangles that describe the danger zones. The local planner connecting the samples with RRT trees is also introduced. Experimental results show that the algorithm much more effectively searches the paths between danger zones and obstacles.*

Streszczenie. *Zaprezentowano metodę wyznaczania możliwej ścieżki ruchu w środowisku z zagrożeniami. Algorytm bazuje na systemie PRM (probabilistic roadmap) i RRT (rapidly-exploring random trees).* (***Planowanie drogi w środowisku z zagrożeniami przy próbkowaniu trójkątnym***)

Keywords: Path planning; Random Sampling; Rapidly-exploring Random Trees; Configuration Space; Motion Constraints
Słowa kluczowe: planowanie drogi, próbkowanie losowe, próbkowanie trójkątne.

Introduction

Assembly maintainability and mechanical part disassembly are important problems in manufacturing engineering. Its solutions would allow CAD systems to provide better feedback to designers, in order to help them create products that are more cost-effective to manufacture, but the manual generation of detailed disassembly or assembly paths can be tedious and time consuming, particularly in environments prone to frequent design changes. The recent trend has been towards developing automated algorithmic solutions for such design problems that can automatically compute a collision-free, global path [1]. These simulation technologies are significant aids in a rapid prototyping environment. The goal of PLM (product lifecycle management) is to provide efficient software solutions to problems that were traditionally solved using costly physical mockups.

The simulation of assembly maintainability attempts to remove a particular part from an assembly. Similarly, part disassembly simulation boils down to computing collision-free trajectories for objects through tight spaces. Many of these problems reduce to motion planning of robots, where collision-free paths need to be computed for rigid objects with six degrees of freedom (DOF) among stationary obstacles [2].

Sample-based motion planning, such as probabilistic roadmaps (PRM) [3] and rapidly-exploring random trees (RRT) [4] have been successfully used to solve high degree-of-freedom motion planning problems arising in different applications [5, 6, 7].

The basic constraints on solution paths are to avoid collision with obstacles, which are called hard constraints. Many techniques have been developed in this field for automatically generating (dis)assembly plans with hard constraints [2]. Prior approaches for rigid objects can be classified into exact algorithms based on algebraic formulation, local techniques that use potential fields or sample-based planning algorithms. In terms of CAD/CAM applications, most of practical planners are based on randomized sampling. These algorithms generate collision-free samples and attempt to connect these samples using local planning in some complex environments, however, in addition to obstacles that must be avoided, there are some areas that must be avoided as much as possible. That is, a path going through these areas is not highly desirable, but would be acceptable if no better path exists or can be computed efficiently. Such constraints on desirable zones and danger zones are called soft constraints [8]. Danielle developed an extended PRM approach to deal with both obstacles and danger zones by sampling the configuration space and throwing away forbidden and unwanted configurations [8]. But the sampling scheme does not work well when the configuration space contains narrow passages between obstacles and danger zones through which the robot must pass. This is because the random sampling scheme tries to distribute nodes with constant density and the volume spanned by the narrow regions makes up only a small fraction of the total volume. Thus few nodes will end up in narrow regions, making it difficult for the planner to find a feasible path through the narrow passage. Some researches show tree-based planners, such as RRT, can use local information to more effectively explore these regions than PRM planner [9].

In this paper, one method DRRT (Danger zones RRT) for extending a traditional RRT planning approach to deal with a set of danger zones is discussed. More specifically, we propose a new path planner that builds RRT tree, by extending existing techniques for danger zones sampling and local planning. The rest of the paper is organized as follows. In section 2, the previous work is briefly introduced. In section 3, a RRT-based planner is showed to find paths in the environment with danger zones by an appropriate local planner and triangles random sampling method. In section 4, some experiments with the developed planner is discussed, the results show that the method indeed work and lead to the required paths. Finally, section 5 concludes this paper and discusses the future work.

Problem definition

The Danger Zones

Let configuration space C be a bounded connected open subset of Rd, where $d \in N, d \geq 2$. Figure 1 shows the issues about the path planning in the environment with obstacles and danger zones. The square body must move from the given start to the given goal position. It must avoid the black obstacles and should preferably avoid the grey danger zones. This is not completely possible but the robot should try to stay as much as possible in the free space.

The problem studied in this paper is an extension of the traditional problem of motion planning. The workspace not only contains a set of obstacles, but also a set of danger zones. This space can be subdivided into a number of subspaces [8]:

- The free configuration space Cfree that consists of those configurations in which the robot does not intersect any obstacle or danger zone.

- The forbidden configuration space Cobs that consists of those configurations in which the robot R does intersect an obstacle.
- The undesirable configuration space Cundes that consists of those configurations in which the robot R completely lies in a danger zone.
- The semi-desirable configuration space Csemi-des that consists of those configurations in which the robot R does not intersect an obstacle and at least one point of the robot lies in a danger zone and at least one point does not.

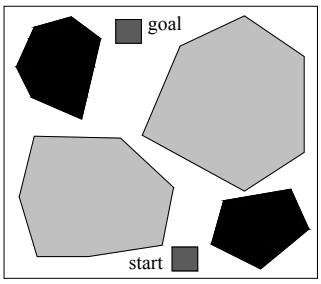


Fig. 1. Two danger zones (grey) and two obstacles (black) that the moving square should try to avoid

It is easy to see that these four subspaces are disjoint and that their union is the full configuration space C.

Danger zones are explicitly modeled in the environment (so they are e.g. not implicitly defined by an area around an obstacle). Obstacles and danger zones are assumed do not intersect each other. Obstacles and danger zones to be open, that is, the robot is allowed to touch them.

Problem Definition

A sequence on a set A, denoted as $\{a_i\} i \in N$, is a mapping from N to A with $i \to a_i$. Given $a,b \in R$, the closed interval between a and b is denoted by [a, b]. The Euclidean norm is denoted by $\|\cdot\|$.

Let Cgoal called the goal region, be open subsets of C. Let the initial state, xinit, be an element of Cfree. In the sequel, a path in Cfree ∪ Csemi-des is said to be a feasible path. A path that starts at xinit and ends in the goal region is said to be a feasible path, i.e., a collision-free path $\sigma : [0, s] \to C_{free} \cup C_{semi-des}$ is feasible if and only if $\sigma(0) = x_{init}$ and $\sigma(s) \in C_{goal}$. The feasibility problem of path planning is to find a feasible path, if one exists, and report failure otherwise.

Problem (Feasible planning): Given a bounded connected open set $C \in R^d$, an initial state $x_{init} \in C_{free}$, and a goal region $C_{goal} \in C_{free}$, find a path $\sigma : [0, s] \to C_{free} \cup C_{semi-des}$ such that $\sigma(0) = x_{init}$ and $\sigma(s) \in C_{goal}$, if one exists. If no such path exists, then report failure.

For the configuration space with obstacles and danger zones, reference [8] proved that there exists a path that lies completely in Cfree ∪ Csemi-des if between two free configurations s and g there exists a path, that possibly intersects on obstacles or more danger zones. This theorem shows that it is feasible to look for paths from start configuration to goal configuration, where the robot will never be completely in the danger zone. It will always have at least one point at the border of the zone. This paper will use this fact in the sampling approach to avoid that the robot completely penetrates a danger zone.

Path planner

The Basic RRT Algorithm

The basic RRT algorithm presented in [4] is shown in algorithms 1. The RRT algorithm explores the free space by randomly sampling and building a tree. RRT is used to search high dimensional spaces. The basic RRT algorithm is as follows. Starting with a tree T with a root node, the algorithm iteratively adds more nodes to the tree. During each iteration, a configuration qrand is randomly generated by RANDOM_STATE() function, the nearest node qnear in the tree T to qrand is generated by NEAREST_NEIGHBOR() function, and the basic RRT algorithm attempts to connect qnear to qrand by a straight line by the EXTEND() function in the configuration space. If the configuration qrand and qnear can be connected via a collision free path, the tree is extended from qrand to qnear and grows as shown in Figure 2. Otherwise, the planner computes qn, the first in-contact configuration (a configuration in the contact space) on the straight line from qnear to qrand. The tree then extends to qn. We refer to this way of growing the tree as the standard RRT extension.

Algorithm 1 Basic RRT expansion method

```
RRT((qinit,qgoal)
1   Ta.init(qinit); Tb.init(qgoal);
2   For k=1 to K do
3     qrand ← RANDOM_STATE();
1     qnear ← NEAREST_NEIGHBOR(qrand,T);
2     qnew← EXTEND(qnear,T);
4     if (qnew can connect to qnear)   then
3         T.add_vertex(qnew);
4         T.add_edge(qnear,qnew);
5   End for
6   Return T
```

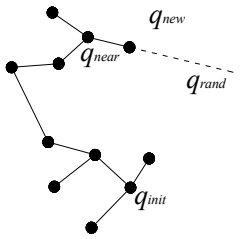


Fig. 2. The basic RRT algorithm

Density Avoided Sampling in the Cfree Space

To find a feasible path in Cfree ∪ Csemi-des, firstly, sampling in Cfree and trying to find a free path, if a path between start and goal can be found there, the danger zones can be avoided.

One of the challenges for RRT planners is to generate samples in narrow passages of the free space. Moreover, though the basic RRT planner can perform a biased search towards regions not yet visited, such bias does not take into account the obstacles in the environment. Therefore, the basic RRT planner can have difficulty growing out of narrow passages in cluttered environments [2]. If trying to grow two trees from the start and goal configurations, one tree is extended in each iteration and an attempt is made to connect the nearest vertex of the other tree to the new vertex. The basic RRT is converted into RRT_DUAL. If instead of attempting to extend an RRT by an incremental step, the process can be iterated until the random state or an obstacle is reached. This method is called CONNECT. If CONNECT function is used to replace EXTEND in RRT_DUAL and yields a RRT that grows quickly [4], this method is called RRT_ConCon.

In this section we will use an algorithm which will sample

in the configuration space by avoiding the high density areas of tree nodes. The RRT_DUAL-based procedure can be mimicked by a simple sampling technique to improve the expansion process in Rapidly-exploring Random Trees. This method should bias the tree to have a tendency to escaping from high density areas [10].

To estimate the density of tree nodes, an additional density parameter is defined to each node. This parameter can be considered as an estimation of each node's ambient free space. Reference [10] introduced a density parameter for each node which is defined as the total sum of the reciprocal of its distances to other nodes.

$$D(i,j)=\begin{cases}\dfrac{1}{\left|q_i-q_j\right|_2} & \left|q_i-q_j\right|>\varepsilon \\ D_{\max} & \left|q_i-q_j\right|\le\varepsilon\end{cases} \tag{1}$$

The density of each node:

$$TD(i)=\sum_{k=0}^{n-1}D(i,k) \tag{2}$$

where, q is the index specified node of the tree and n is the total number of nodes existing in the tree. Also a maximum value (Dmax) should be defined and applied when the distance of two particular nodes is less than a specified value (ε) in order to prevent division by zero.

The density parameter TD is based on the second moment which can be used to indicate the "density" in some times, but the "density" is equivalent while the standard deviation is different. As shown in Figure 3, two nodes are very close to qnear, but most of the nodes are far away from qnear, the above density parameter can't be used to estimate the density of qnear exactly.

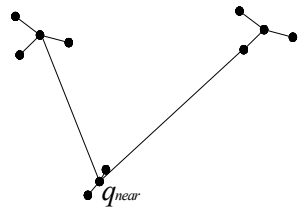


Fig. 3. The distribution of tree nodes

To estimate the density of tree nodes more exactly, kurtosis parameter based on fourth moment is defined which is a measure of the "peakedness" of the probability distribution of a real-valued random variable. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations, so the fourth moment is practicable to estimate the density of tree nodes. Formula (3) is the definition of kurtosis.

$$Kurtosis(i)=\frac{\mu_4}{\sigma^4}=\frac{\sum_{k=0}^{n}(x_k-\bar{x})^4 f_k}{ns^4} \tag{4}$$

where μ_4 is the fourth moment about the mean and σ is the standard deviation. The x_k is a sample point, $\bar{x}$ is the mean of sample points and fk is the frequency of sample point k. This definition is used so that the standard normal distribution has a kurtosis of zero. In addition, with the second definition positive kurtosis indicates a "peaked" distribution and negative kurtosis indicates a "flat" distribution.

The density parameter of node i can be defined by formula (4).

$$TD(i)=1/Kurtosis(i)=\frac{ns^4}{\sum_{k=0}^{n}\left|q_k-q_i\right|^4} \tag{4}$$

Where, s is the standard deviation. If

$$\sum_{k=0}^{n}\left|q_k-q_i\right|^4=0$$

Then TD (i) =Dmax;

The density value can be iteratively updated whenever a new node is generated.

Now we can perform sampling based on this density values. During the density updating procedure, the node with the lowest density, will be a candidate with maximum interest for expansion. This means that the node which is in the freest space will be chosen to have another successor node or child. Then instead of a uniform sampling in the entire space, a little neighborhood will be chosen for sampling.

We chose this neighborhood, in a circle around the minimum density node (which will be spherical volume around the node in higher dimensions). Radius of this circle may be chosen such that the sampling task can be considered as local.

With a constant radius for this circle there is only one parameter (angle) to be determined by the random variable generator. Its value can change between 0 and 2π Radians.

After this stage, rest of the work is just like the basic RRT. Although with a large probability, the nearest neighbor of the sampled point must be the freest node, but some times (especially when two major branches are approaching together) it is not so desired. Hence the nearest neighborhood search should be carried out globally. After that, a new node will be generated and added to the tree.

Although we have some additional computations to do in this algorithm, the results are outstanding. Density Avoided RRT expands with a fairly equal density in the whole space and around the obstacles, especially when the space is very constrained and highly restricted with the obstacles.

If a feasible path does not exist in Cfree, the Csemi-des need to be sampled. To sample in the environments with danger zones, we apply a triangle random sampling algorithm which includes a sampling approach that samples Cfree ∪ Csemi-des and a local planner which is used to extend the tree.

Triangle Random Sampling in Cfree ∪ Csemi-des

An easy approach would be to simply generate a random configuration and then test whether the robot at this configuration intersects an obstacle or lies completely within a danger zone. But testing whether the robot lies completely within a danger zone is a relatively expensive operation.

After finishing the sampling for Cfree, the possible path should grow along the boundary of danger zones, because there exists a path that lies completely in Cfree ∪ Csemi-des if there exists a path between two free configurations. To improve the search speed in the boundary of danger zones, we use triangle surfaces obtained from the danger zones to assist in selecting a direction for the tree to try and grow. Random points obtained from a 3-dimensional danger zone are taken from the triangles (as that is how we model our environments) that describe the danger zone. Given a triangle selected from danger zones randomly, a direction for the tree to grow can be obtained from the vertices of the triangle. A triangle T in three dimensions is described by

three vertices pt1, pt2 and pt3 in the three dimensional space(pti∈R3, i=1,2,3).The random point obtained from the triangle T can be chosen randomly from the area of the triangle, as shown in Figure4.

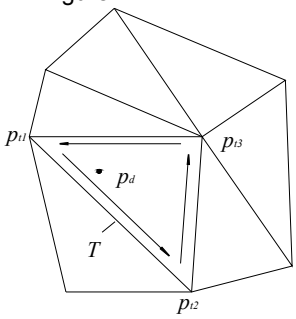


Fig. 4. Possible random point obtained from a random triangle of danger zone polygon

The random point Pd∈R3 in the closed triangle T can be selected by the barycentric coordinates. pt1, pt2, pt3 are the three vertices (corners) of the random triangle T. Let t1, t2, t3 be real numbers that sum to 1, with each between 0 and 1:

$$t_1 + t_2 + t_3 = 1, \qquad 0 \le t_i \le 1$$

One way of selecting a random point is to generate random t1, t2, t3 independently and uniformly in [0,1]

$$p_d = p_{t1}t_1 + p_{t2}t_2 + p_{t3}t_3$$

where, $t_i \sim R_{[0,1]} \quad (i = 1,2,3)$ and Pd is uniformly distributed in the triangle T.

To sample Csemi-des space completely, we select a random point pr inside the robot. The robot R with point pr is placed on position Pd. If there are rotational degrees of freedom, the rotation values are set randomly and a random configuration qrand is generated. Next we test whether the robot intersects any obstacle. Because Pd lies on the surface of a danger zone, pr lies inside the robot, the resulting configuration cannot lie completely inside a danger zone, as shown in Figure 5.

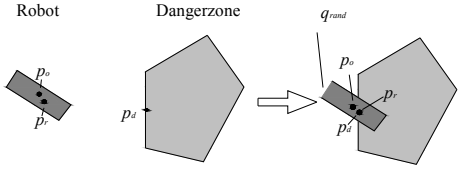


Fig. 5. Generation a new configuration qrand on the boundary of danger zone.

Because the pd is sampled completely in the boundary of danger zones and pr is sampled completely in the robot, it is easy to see that the planner samples in the complete Csemi-des space.

Local Planner in Cfree∪Csemi-des

The goal of the local planner is to efficiently connect two samples that are close together with a feasible path. In this paper, the local planner is based on the method in [8]. A configuration qrand was created through sampling in the boundary of danger zones, then find the closest vertex qnear of RRT tree to qrand in terms of distance metric ρ. But if the qnear and qrand lie on opposite sides of a danger zone, the local planner will generate an invalid path that goes through the danger zone. The selected random triangle can be used to choose a proper qnear and guide the direction for the tree to grow which is consistent with the normal direction of the triangle T. Assuming the translation part of qnear is pnear∈R3 and the angle between the direction pnear pd and the normal direction of the triangle be θ, the planner selects the nearest convex qnear which ensures θ <90.

n is the unit vector in the normal direction of triangle T.

$$n = \frac{(p_{t3} - p_{t1}) \times (p_{t2} - p_{t1})}{\left\| (p_{t3} - p_{t1}) \times (p_{t2} - p_{t1}) \right\|}$$

Then if :

$$n \cdot (p_d - p_{near}) > 0$$

The projection of vector （pd pnear）on the normal direction of triangle T is consistent with the vector n.

After obtaining the qnear, We check whether the line segment pnearpd lies completely in the free workspace. po is the origin of the robot's local coordinate system, pdpo lies completely inside the robot. Otherwise, we interpolate a point in the robot between qnearqrand and move this interpolated point over qnearqrand. At the same time we interpolate the remaining rotational degrees of freedom , get a new input u with the step length ε and ensure that the state remains in Cfree∪Csemi-des. As shown in Figure 6, When the state u is not in Cfree∪Csemi-des, the previous state is set to qnew which will be added into the tree.

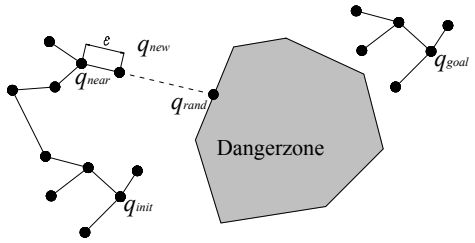


Fig. 6. Possible random point obtained from a triangle before perturbing

Collision detection with obstacles can be performed by an incremental method while collision detection with danger zones can be performed through checking whether a point is in a danger zone polygon. One simple way of finding whether the point is inside or outside a danger zone polygon is to use the ray casting algorithm to test how many times a ray starting from the point intersects the edges of the danger zone polygon. If the point in question is not on the boundary of the polygon, the number of intersections is an even number if the point is outside, and it is odd if inside.

To illustrate the algorithm, we present the description of DRRT algorithm for sampling in the Csemi-des which is based on RRT_ConCon algorithm.

Algorithm 2 Extend function of DRRT

```
DRRT(qinit,qgoal)
5    Ta.init(qinit); Tb.init(qgoal);
6    For k=1 to K do
7      qrand ← RANDOM_TRI_STATE() or
     RANDOM_STATE;
8      If not (CONNECT(Ta, qrand) = Trapped) Then
9        If (CONNECT(Tb,qnew) = Reached ) then
10           Return PATH(Ta,Tb);
11     SWAP(Ta,Tb);
12   Return Failure;

   EXTEND(T,q)
```

```
7    qnear ← NEAREST_NEIGHBOR(q,T);
8    if NEW_STATE(q,qnear,qnew,unew)  --- then (check
     whether q is in CfreeUCsemi-des)
9      T.add_vertex(qnew);
10     T.add_edge(qnear,qnew,unew);
11     If qnew = q then
12         Return Reached;
13     Else
14         Return Advanced;
15   Return Trapped;

CONNECT(T,q)
1    repeat
2      S ← EXTEND(T,q);
3    Until not (S = Advanced)
4    Return S;
```

Probabilistic Completeness

The RRT method is probabilistically complete for many different types of motion planning problems.

In this section, the feasibility problem in danger zones is considered. It is proven that the DRRT algorithm inherits the probabilistic completeness as well as the exponential decay of the probability of failure (as the number of samples increase) from the RRT. These results imply that the RRT and DRRT algorithms have the same performance in producing a solution to the feasibility problem as the number of samples increase.

Sets of vertices and edges of the graphs maintained by the RRT and the DRRT algorithms can be defined as functions from the sample space to appropriate sets. More precisely, let $\{V_i^{RRT}\}_{i\in N}$ and $\{V_i^{DRRT}\}_{i\in N}$, sequences of functions defined from subsets of Cfree $\cup$ Csemi-des, be the sets of vertices in the RRT and the DRRT, respectively, at the end of iteration i. By convention, we define $V_0^{RRT} = V_0^{DRRT} = \{x_{init}\}$. Similarly, let ε_i^{RRT} and ε_i^{DRRT}, defined for all $i \in N$, denote the set of edges in the RRT and the DRRT, respectively, at the end of iteration i. Clearly, $\varepsilon_0^{RRT} = \varepsilon_0^{DRRT} = \varnothing$.

An important lemma used for proving the equivalency between the RRT and the DRRT algorithms is the following.

Lemma 1: For all $i \in N$ and all $\omega \in \Omega$, $V_i^{RRT}(\omega) = V_i^{DRRT}(\omega)$ and $\varepsilon_i^{RRT}(\omega) \subseteq \varepsilon_i^{DRRT}(\omega)$.

Lemma 1 implies that the paths discovered by the RRT algorithm by the end of iteration i is, essentially, a subset of those discovered by the DRRT by the end of the same iteration.

An algorithm is said to be probabilistically complete if it finds a feasible path with probability approaching one as the number of iterations approaches infinity.

Note that there exists a collision-free path starting from xinit to any vertex in the tree maintained by the RRT, since the RRT maintains a connected graph on Cfree that necessarily includes xinit. Using this fact, the probabilistic completeness property of the RRT is stated alternatively as follows.

Theorem 1 (see [11]): If there exists a feasible path in Cfree, then

$$\lim_{i\to\infty} P(\{V_i^{RRT} \cap C_{goal} \neq 0\}) = 1 \tag{8}$$

With Lemma 1, the following theorem is immediate.

Theorem 2: if there exists a feasible solution to Problem 1, then

$$\lim_{i\to\infty} P(\{V_i^{DRRT} \cap C_{goal} \neq 0\}) = 1 \tag{9}$$

So the DRRT algorithm is probabilistically complete.

Experimetns and results

In this section, we present experimental results of the DRRT planner on 3D rigid robots. All the timings reported here were taken on a Windows PC with 2.8GHz of CPU and 2GB of memory.

We implemented DRRT on 3D rigid robots. A rigid robot needs to plan through some narrow passages in the 3D environment. Our implementation consists of two parts: the implementation of the sampling step and the integration with an RRT planner. The PQP algorithm is used for collision detection. We integrate the random triangle sampling retraction algorithm into the basic RRT planner. During each step, a sequence of configurations in the triangle surfaces of danger zones, are generated. Path planner DRRT then attempts to extend the tree to each configuration using the modified extension scheme.

This example corresponds to the problem illustrated in Figure 7. The green blocks are obstacle and the yellow blocks are danger zones. The red object is a robot. A path can be searched in the environment without danger zones, but it is impossible to find a path in Cfree if the danger zones are contained by the environment. Csemi-des space must be sampled to find a feasible path.

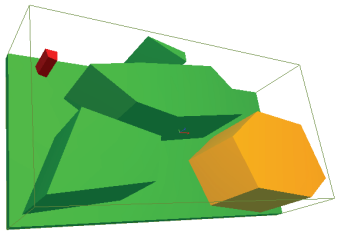
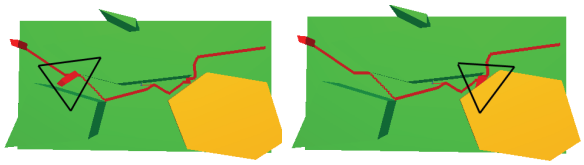


Fig. 7. The environment with obstacle and danger zones

Because the planner is based on sampling randomly, the results are the average values of 100 experiments. To test the efficiency of DRRT, we compared DRRT with the DPRM (PRM-based path planning in the environment with danger zones) [8]. Table 1 displays the computing time and the number of nodes in the search trees generated for solving the problem with DRRT and DPRM. The maximum number of samples for the Cfree and Cfree $\cup$ Csemi-des is 1000.

Table 1. Results on the problem

	Nodes	Collision Detection	Time(s)
DRRT	794	39763	34.87
DPRM	947	45008	61.56



(a). Path generated by DRRT (b). Robot on the boundary of danger zones

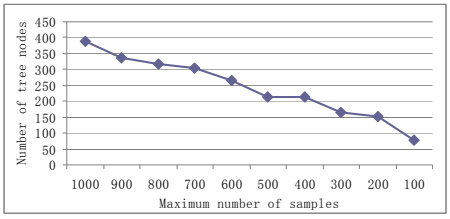
Fig. 8. The results of experiments

As shown in Figure8 (a), the robot does not collide with obstacles and danger zones and lies in the Cfree. In Figure8 (b), the robot does not lie in the danger zones completely. A part of the robot lies in the danger zones while the other part is outside of the danger zones, which means some segments of the path is in Csemi-des. Table 1 shows that, in the first stage for sampling Cfree, both DRRT and DPRM algorithms fail to find a feasible path in Cfree. But in the second stage for sampling Cfree $\cup$ Csemi-des, compared with the idea of sampling in the whole Cfree space, the idea of sampling in the boundary of danger

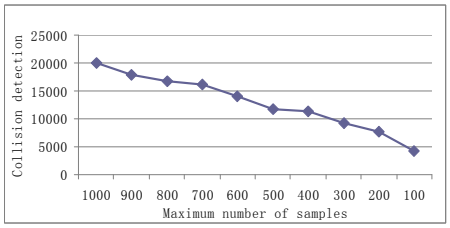
zones gains efficiency by filtering out those samples which are far away from danger zone boundaries. The time of the first planner is so less than the second one.

Because the planner is divided into two stages, the first stage is to sample in Cfree space and the second stage is to sample in Cfree ∪ Csemi-des, the efficiency of planning will be affected by the maximum number of samples in two stages.

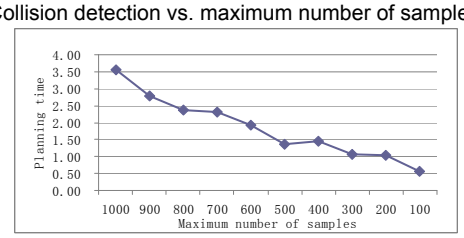
Figure 9 shows the relationship between the maximum of samples in the first stage and the planning time, success rate, collision detection and tree nodes. The maximum number of samples in the first stage is from 100 to 1000 and the number in the second stage remains 1000.



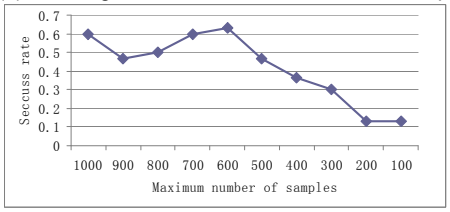
(a) Number of tree nodes vs. maximum number of samples



(b) Collision detection vs. maximum number of samples



(c) Planning time vs. maximum number of samples



(d) Success rate vs. maximum number of samples

Fig. 9. Path planning results with different maximum number of samples

As shown in Figure9, while reducing the maximum number of samples in the first stage, the total planning time, the times of collision detection and the number of tree nodes decrease because of the more efficiency sampling on the boundary of danger zones in the second stage. However, the success rate of planning did not decline. The highest success rate appeared when the maximum number of samples is 600. Therefore, selecting the proper maximum number of samples should take into account the balance between the success rate and planning time.

Conclusions

The example above has shown that the planner is able to find paths between danger zones and obstacles. During the motion the robot always has at least one point in the free space. Touching danger zones can be dealt with by explicitly sampling on the boundary of the danger zones.

Our approach has a few limitations. The DRRT optimization algorithm does not take account into smoothing the path, which results in a rough path.

Danger zones are just one aspect of improving the quality of the paths. Many other constraints exist on paths, like staying away from obstacles, using as little degrees of freedom as possible. We plan to study such quality issues further in the near future.

Acknowledgment

This paper is an updated and revised version of an award winning paper previously presented at The Fourth International Conference on Measuring Technology and Mechatronics Automation, Changsha, China, 2010.

REFERENCES

[1] Chang, H. and Li, T.-Y, "Assembly maintainability study with motion planning", Proceedings of the 1995 IEEE International Conference on Robotics & Automation, pp1012-1017, 1995.

[2] L. Zhang, X. Huang, Y. Kim and D. Manocha, "D-Plan: Efficient Collision-Free Path Computation for Part Removal and Disassembly", In Journal of Computer-Aided Design and Applications, vol.15, No.5 pp1209-1212, 2008.

[3] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in highdimensional configuration spaces", IEEE Trans. Robot. Automat, vol.12, No.4, pp566–580, 1996.

[4] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning", In Proc. IEEE International Conference on Robotics and Automation, vol.35, No.5, pp1806-1812, 2000.

[5] H.-L. Cheng, D. Hsu, J.-C. Latombe, and G. S´anchez-Ante, "Multi-level free-space dilation for sampling narrow passages in PRM planning", In Proc. IEEE Int. Conf. on Robotics & Automation, pp1255–1260, 2006.

[6] S. Redon and M. Lin, "A fast method for local penetration depth computation", Journal of Graphics Tools, vol.11, No.2, pp37–50, 2006.

[7] M. Saha, J. Latombe, Y. Chang, Lin, and F. Prinz, "Finding narrow passages withprobabilistic roadmaps: the small step retraction method", Intelligent Robots and Systems, vol.19, No.3, pp301–319, Dec 2005.

[8] Danielle. Sent & Mark. H. Overmars, "Motion planning in environments with danger zones", Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, pp1488-1493, 2001.

[9] LaValle S M, "Rapidly-exploring random trees: a new tool for path planning", Technical Report TR98-11, Computer Science Dept, Iowa State University, 1998.

[10] Sohrab Khanmohammadi and Amin Mahdizadeh, "Density avoided sampling: an intelligent sampling technique for Rapidly-exploring Random Trees", Eighth International Conference on Hybrid Intelligent Systems, pp672-677, 2008.

[11] LaValle S M. and J. J. Kuffner, "Randomized kinodynamic planning", International Journal of Robotics Research, vol.20, No.5, pp378–400, May 2001

***Authors**: prof. dr Fei Peng, School of Naval Architecture & Ocean Engineering, Huazhong University of Science and Technology Wuhan, China, E-mail: Pengfeia@188.com;.*