Planning of Graspless Manipulation based on Rapidly-Exploring Random Trees

Kiyokazu MIYAZAWA† Yusuke MAEDA* Tamio ARAI†

†Dept. of Prec. Eng., School of Eng.,
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-8656 JAPAN

*Div. of Sys. Res., Fac. of Eng.,
Yokohama National University
79-5 Tokiwadai, Hodogaya-ku Yokohama
240-8501 JAPAN

Abstract

Planning of graspless manipulation is a difficult problem because of the complicated mechanics involved. The authors previously developed a motion planning method for general graspless manipulation by robot fingertips, but it requires a great deal of computation. In this paper, we present an improved method based on a probabilistic technique, RRTs (Rapidly-exploring Random Trees), to accelerate planning. The method is applied to the planning of graspless manipulation where the manipulated object has three degrees of freedom, while our previous planner can deal with manipulation where the object has only one degree of freedom.

1 Introduction

Manipulation without grasping is referred to as graspless manipulation [1] or nonprehensile manipulation [2]. It includes pushing, sliding and tumbling (Figure 1). The merits of graspless manipulation include the following:

- Manipulation without supporting all the weight of the object.
- Manipulation with simple mechanisms.
- Manipulation when grasping is impossible.

Thus, graspless manipulation is important as a complement to conventional pick-and-place to enhance the dexterity of robots.

One of the demerits of graspless manipulation is the difficulty in planning. The manipulated object is in contact with not only robots but also the environment; thus, the planning of graspless manipulation requires complicated computation for mechanical analysis. Most related studies, therefore, deal with the planning of manipulation in a specific operation, such as pushing, (e.g., [3]) to simplify computation.

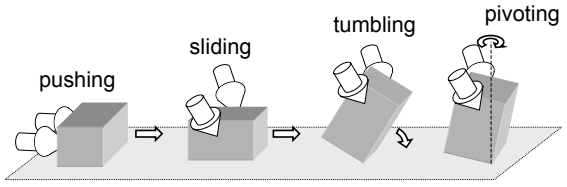

Figure 1: Graspless Manipulation

The authors previously proposed a planning method for general graspless manipulation by multiple robot fingertips [4]. The method can automatically generate various graspless operations, including pushing and tumbling. However, the computation time for planning is very long; for example, it takes 533 CPU minutes for a Pentium 4, 2.8 GHz PC to plan a simple sliding operation where the object has only one degree of freedom.

Randomized motion planning techniques such as Probabilistic Roadmaps (PRM) [5] and Rapidly-exploring Random Trees (RRTs) [6] are widely used to obtain feasible plans quickly. Some studies have used randomized techniques for manipulation planning. Using PRM, Ji and Xiao proposed a planning method of the motions of an object in contact with polyhedra [7]. Yashima et al. developed a planning method for in-hand manipulation using RRTs [8].

In this paper, we report the improvement of our previous planner using RRTs to obtain feasible graspless manipulation more efficiently. Our new method is applied to planning problems of a sliding operation on a plane where the manipulated object has three degrees of freedom. It is very difficult for our previous planner to deal with such problems because of the long computation time.

This paper is organized as follows: In Section 2, assumptions for planning graspless manipulation are introduced. In Section 3, a method of motion planning of robot fingertips for graspless manipulation is presented. Our new planning algorithm using randomized techniques is

0-7803-9080-6/05/$20.00 ©2005 IEEE.

described in Section 4. In Section 5, some examples of planned graspless manipulation are presented. Finally, this paper is concluded in Section 6.

2 Assumptions

In this paper, for graspless manipulation by multiple robot fingertips (Figure 2), we make the following assumptions:

1. The manipulated object, robot fingertips, and environment are rigid.
2. The manipulated object is a polyhedron that slides on a plane with three degrees of freedom ($x,\ y,\ \theta$).
3. Manipulation is quasi-static.
4. Coulomb friction exists between the object and the environment (or the robot fingertips). The friction coefficient on a contact surface is uniform.
5. Static and kinetic friction coefficients are equal.
6. All the contacts can be approximated by finite-point contacts [9].
7. Each friction cone can be approximated by a polyhedral convex cone [10].
8. Each robot finger is modeled as a rigid sphere and is in one-point non-sliding contact with the object. Only fingertips are considered.
9. The normal component of each finger force has an upper limit.
10. Each robot finger is either in position-control mode or in force-control mode.
    (a) Each robot finger in position-control mode can passively apply arbitrary force within its friction cone.
    (b) Each robot finger in force-control mode is in hybrid position/force control [11]; the finger can actively apply a commanded normal force and passively apply arbitrary tangential force within its friction cone.
11. Sliding and rolling of robot fingers on object surfaces is not allowed. Regrasping is required to change the location of fingertips on the object. Regrasping motions are always collision-free.

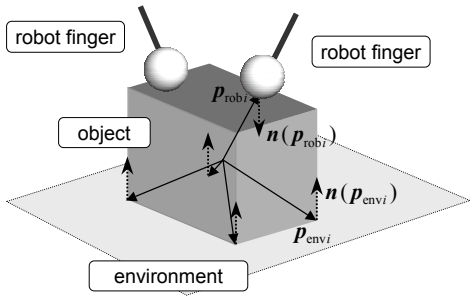

Figure 2: Object in Graspless Manipulation

The problem to be solved is to determine a sequence of fingertip positions and finger control modes to move an object from a given initial configuration to a given goal configuration by graspless manipulation. A sequence of desired normal forces is also to be obtained for force-controlled fingers.

3 Planning of Graspless Manipulation

In this section, we outline the motion planning of robot fingertips for graspless manipulation.

3.1 Configuration Space

We define a configuration space (C-Space) that represents the degrees of freedom for both the manipulated object and robot fingertips. Let us denote a configuration of the object by $\boldsymbol{X} = (x, y, \theta)$. We represent positions of the robot fingertips in manipulation as their locations on the faces of the object; we denote a configuration of the i-th fingertip by $\boldsymbol{F}_i = (x_i, y_i, f_i)$, where f_i is the ID number of the face on which the fingertip is located, and x_i and y_i are the coordinates of fingertip location on the face. The configuration of the manipulation system is expressed as $\boldsymbol{q} = (\boldsymbol{X}, \boldsymbol{F})$, where $\boldsymbol{F} = (\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N)$ and N is the number of the fingertips.

3.2 Sampling Configurations

We cannot, however, search the C-Space in the same manner used in conventional obstacle avoidance problems because graspless manipulation may be irreversible and regrasping causes a discontinuous "jump" in the C-Space [4]. Thus, we sample configurations in the C-Space and connect them by directed arcs to construct a directed graph that represents feasible graspless manipulation. Planning of graspless manipulation is transformed into constructing

```
GMRRT (X_init, X_goal);
    F_c.init (); A.init ();
    F_init ← RANDOM_FINGER (F_c);
    q_init ← (X_init, F_init);
    T.init (q_init);
    for k = 1 to K do
        X_rand ← RANDOM_STATE ();
        C_k ← EXTEND (T, X_rand, F_c);
        if (X_goal ∈ T ) then
          return Reached;
        A.update(C_k);
        if (A > R_f ) then
          F_c.add_finger_location ();
          A.init ();
    return Failed;
```

```
EXTEND (T, X_rand, F_c);
    q_near = (X_near, F_near)
        ← NEAREST_NEIGHBOR (X_rand, T);
    if (RAND() > P_regrasp) then
        q_rand ← (X_rand, F_near);
    else
        F_rand ← RANDOM_FINGER (F_c);
        q_rand ← (X_near, F_rand);
    T.connect (q_near, q_rand);
    return FAILURE_RATE (q_near, q_rand);
```

Figure 3: Planning Algorithm

this graph and finding a path from the initial configuration to the goal configuration.

In our previous planner [4], lattice points in the C-Space are sampled to construct a graph that represents feasible manipulation. On the other hand, in this paper, we adopt RRTs [6] to sample configurations and construct the graph.

We can perform adaptive sampling of configurations using RRTs, which will reduce the number of graph nodes required for motion planning. Therefore, the computation time will also be reduced.

Planning of graspless manipulation is, however, not so simple that we can adopt RRTs straightforwardly. Thus, we describe below how we incorporate RRTs into our planner.

4 Details of Planning Algorithm

Our proposed algorithm is shown in Figure 3. The main function is GMRRT(). It extends branches and constructs a search tree $\mathcal{T}$ from an initial configuration of the object, $\boldsymbol{X}_{\rm init}$. If a goal configuration of the object, $\boldsymbol{X}_{\rm goal}$, is added to $\mathcal{T}$, a path from the initial configuration to the goal configuration is found; that is, we obtain a feasible plan of graspless manipulation.

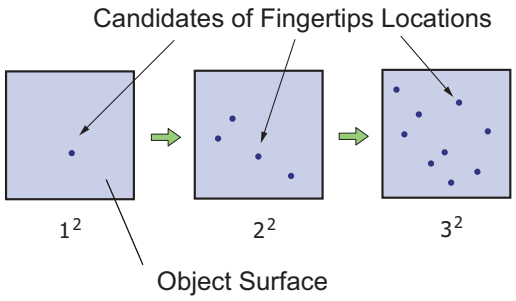

Figure 4: Fingertip Locations using Halton Sequence

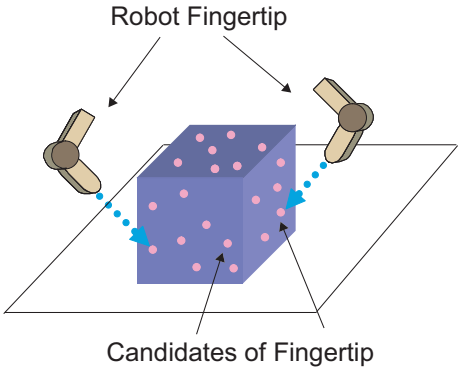

Figure 5: Candidates of Fingertip Locations

4.1 Fingertip Locations

In order to avoid regrasping many times, we restrict fingertip locations in a set of some candidate points, F_c. At first, F_c contains few candidates. As the search tree grows, we add candidates to F_c by the function add_finger_location() using the Halton Sequence [12] (Figure 4), which is well known as a low-discrepancy sequence.

4.2 Tree Extension

The function RANDOM_STATE() randomly samples a configuration of the object, $\boldsymbol{X}_{\rm rand}$, from the collision-free C-Space of the object. The function NEAREST_NEIGHBOR() selects a configuration, $\boldsymbol{q}_{\rm near} = (\boldsymbol{X}_{\rm near}, \boldsymbol{F}_{\rm near})$, nearest to $\boldsymbol{X}_{\rm rand}$ in the search tree $\mathcal{T}$

generated so far. The distance between $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ is determined in accordance with the distance function $d(\boldsymbol{X}_1, \boldsymbol{X}_2)$ as follows:

$$d(\boldsymbol{X}_1, \boldsymbol{X}_2) = \{w_x(x_2 - x_1)^2 + w_y(y_2 - y_1)^2 + w_\theta(\theta_2 - \theta_1)^2\}^{1/2}, \quad (1)$$

where

$$\boldsymbol{X}_1 = (x_1,\ y_1,\ \theta_1),\ \ \boldsymbol{X}_2 = (x_2,\ y_2,\ \theta_2),$$
$$w_x,\ w_y,\ w_\theta : \text{const.}$$

4.3 Nodes Connection

Next, we extend the tree $\mathcal{T}$ by making a new branch from $\boldsymbol{q}_{\rm near}$. If a random number (RAND()) is larger than a threshold, $P_{\rm regrasp}$, we try to make a branch from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm rand} = (\boldsymbol{X}_{\rm rand}, \boldsymbol{F}_{\rm near})$; that is, we change not the fingertip locations but the configuration of the object in order to avoid frequent regrasping. Otherwise, we randomly select new fingertip locations $\boldsymbol{F}_{\rm rand}$ from F_c and try to make a branch from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm rand} = (\boldsymbol{X}_{\rm near}, \boldsymbol{F}_{\rm rand})$.

Then we check whether we can make a connection from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm rand}$; the local feasibility of graspless manipulation is examined. When the above points can be connected, the newly sampled configuration $\boldsymbol{q}_{\rm rand}$ is added to the search tree $\mathcal{T}$. When we can make a connection only up to a halfway point, $\boldsymbol{q}_{\rm new}$, we add it to $\mathcal{T}$. Thus, repeating sampling configurations, selecting the nearest neighbors, and connecting them will extend the search tree.

Here, we explain the function connect(), which checks the feasibility of manipulation and makes a new branch. There are two possible situations according to the fingertip locations of $\boldsymbol{q}_{\rm rand}$:

Changing Object Configuration When $\boldsymbol{q}_{\rm rand}$ and $\boldsymbol{q}_{\rm near}$ have the same fingertip locations, we divide the interval between $\boldsymbol{q}_{\rm near}$ and $\boldsymbol{q}_{\rm rand}$ into n sections (Figure 6):

$$n := \left\lceil \frac{d(\boldsymbol{X}_{\rm near},\ \boldsymbol{X}_{\rm rand})}{\Delta t} \right\rceil, \quad (2)$$

$$\boldsymbol{s} := \frac{\boldsymbol{X}_{\rm rand} - \boldsymbol{X}_{\rm near}}{n}, \quad (3)$$

$$\boldsymbol{X}_i := \boldsymbol{X}_{\rm near} + i\boldsymbol{s}, \quad (4)$$

where Δt is a step width. In each step, we check collisions and calculate manipulation stability, as defined in [13]. If no collisions occur and manipulation stability z is larger than a threshold, $z_{\rm min}$, we can connect this interval. We repeat this process as far as possible. When we cannot make a connection beyond $\boldsymbol{X}_i$,

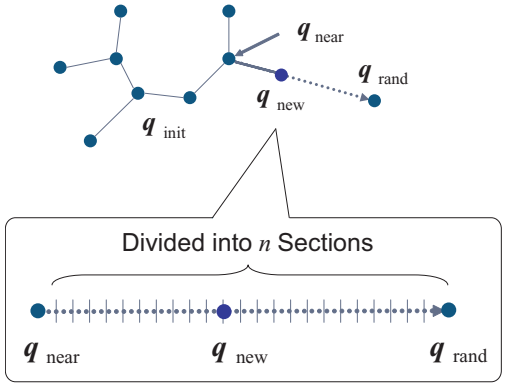

Figure 6: Connection Step

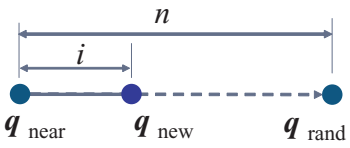

Figure 7: Failure Rate

a new configuration, $\boldsymbol{q}_{\rm new} = (\boldsymbol{X}_i, \boldsymbol{F}_{\rm near})$, is added to the search tree $\mathcal{T}$. At this point, the function FAILURE_RATE() records the connection failure rate C_k, defined as follows:

$$C_k = 1 - \frac{i}{n}. \quad (5)$$

This value is used to decide when to increase candidate points of F_c.

Changing Fingertip Location When $\boldsymbol{q}_{\rm rand}$ and $\boldsymbol{q}_{\rm near}$ have different fingertip locations, we simply connect them by a branch that corresponds to regrasping, as long as no collisions occur.

4.4 Add Candidates of Fingertip Locations

When we find a difficulty in extending the search tree, we add new candidates of fingertip locations to F_c. We calculate the arithmetic average of the connection failure rate as follows:

$$A = \frac{C_1 + C_2 + \cdots + C_m}{m}. \quad (m : \text{connection trials}) \quad (6)$$

When A is larger than a threshold, R_f, and m is also larger than a threshold, M_f, we add candidates of fingertip locations using the Halton sequence; the candidates are increased to $(i+1)^2$ per surface in the i-th addition (Figure 4). Once we add candidates, the previously recorded

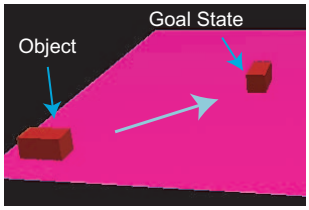

Figure 8: Plan A

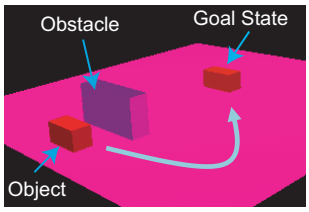

Figure 9: Plan B

connection failure rate is discarded, and we calculate the connection failure rate from the beginning again.

4.5 Connection to Goal Configuration

Because sampling is yielded randomly, there is little possibility that the goal configuration is sampled. Consequently, we sample the goal configuration with a probability of $\epsilon(\ll 1)$ in RANDOM_STATE() [6]. When the goal configuration $\boldsymbol{X}_{\rm goal}$ is connected to the search tree, the planning is finished. Then we track back the search tree from $\boldsymbol{X}_{\rm goal}$ to $\boldsymbol{X}_{\rm init}$, and a feasible path is obtained.

5 Planned Results

In this section, results of our proposed algorithm are shown. Here we present a sliding operation of a cuboid on a plane by two fingertips. Our planning algorithm is written in C++, based on Motion Strategy Library [14]. The computation time for the examples given below is measured on a Linux PC with Pentium 4 at 2.8 GHz.

Suppose a cuboid whose size is $5 \times 5 \times 10$. The mass of the object is 1, and its distribution is uniform; each robot fingertip is modeled as a sphere whose radius is 1; the gravitational acceleration is 9.8. The area of the field is 100×100. The friction coefficient between the object and the environment is 0.5, and that between the object and each finger is 0.7. Initially, F_c has four candidate points for each surface of the object. Other parameters of this planning are as follows:

$$z_{\rm min} = 0.5;\ w_x = w_y = 1;\ w_\theta = (50/\pi)^2; \Delta t = 0.1;$$
$$P_{\rm regrasp} = 0.05;\ \epsilon = 0.05;\ R_f = 0.8;\ M_f = 10.$$

We have solved two types of problems:

1. Plan A:
   sliding on a planar with no obstacles (Figure 8)
   from $\boldsymbol{X}_{\rm init} = (10, 10, 0)$ to $\boldsymbol{X}_{\rm goal} = (50, 50, \pi/2)$
2. Plan B:
   sliding on a planar with an obstacle (Figure 9)
   from $\boldsymbol{X}_{\rm init} = (50, 10, 0)$ to $\boldsymbol{X}_{\rm goal} = (50, 60, 0)$

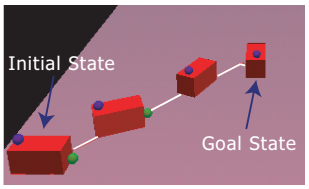

Figure 10: Good Result of Plan A

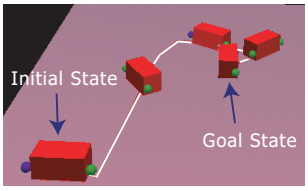

Figure 11: Bad Result of Plan A

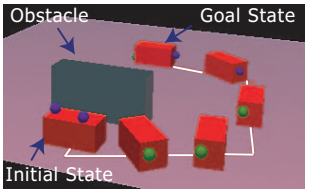

Figure 12: Good Result of Plan B

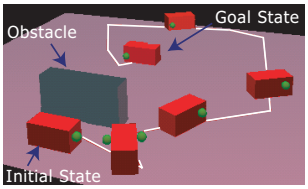

Figure 13: Bad Result of Plan B

Plan A is the sliding operation in the diagonal direction. In Plan B, there is an obstacle in the goal direction which must be avoided.

The planning results (100 trials for each) are shown in Table 1. "CPU Time" is the average computation time. "Number of Regrasps" is the average number of regrasps. "Number of Finger Locations" is the average number of candidates of finger locations for each of the object surfaces. "Number of Samples" is the average number of sampled configurations.

In the case of Plan A, computation times had a wide dispersion in 100 trials, ranging from 0.180 [min] to 78.8 [min]. Figure 10 shows an example of a good result. This plan was finished in 4.11 [min], and five configurations were sampled. There was no need to regrasp, and the trajectory was relatively straight. Thus, when a good configuration is sampled early, the computation time is small. On the other hand, Figure 11 shows an example in which it took 18.7 [min] to search the path. One regrasp was needed, and the planned path was meandering. These results indicate that feasible manipulation can be achieved quickly if a good configuration is sampled by chance. Thus, the quality of manipulation depends highly on randomly sampled configurations, as in Figure 10 and Figure 11. In particular, the finger locations are important to obtain a high quality path for graspless manipulation.

In the case of Plan B, in spite of an obstacle, the path can be searched. Plan B also has wide variations in computation time and in quality (Figure 12, Figure 13). The computation time range was from 0.936 [min] to 57.0 [min].

Table 1: Computation Time

Plan	CPU Time [min]	Number of Regrasps	Number of Finger Locations	Number of Samples
Plan A	13.0	1.50	4.08	25.8
Plan B	15.1	2.31	5.62	47.1

6 Conclusion

We developed a method of motion planning of robot fingertips for graspless manipulation using Rapidly-exploring Random Trees. The method was successfully applied to the planning of sliding operations of a cuboid with three degrees of freedom $(x,\ y,\ \theta)$ on a plane with and without obstacles, while our previous planner can deal with planning of graspless manipulation where the object has only one degree of freedom. This method can also apply for other graspless manipulation, tumbling and pivoting.

A problem in our current method is a wide variation in the computation times. Because this problem stems from random sampling of fingertip locations, it can be solved using biased sampling by heuristics. For example, sampling opposed locations of fingertips with a high probability, which leads to stable pinching, will help reduce the variation.

References

[1] Y. Aiyama, M. Inaba, and H. Inoue: "Pivoting: A New Method of Graspless Manipulation of Object by Robot Fingers," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 136–143, 1993.

[2] M. T. Mason: "Progress in Nonprehensile Manipulation," Int. J. of Robotics Research, Vol. 18, No. 1, pp. 1129–1141, 1999.

[3] K. Lynch and M. Mason: "Stable Pushing: Mechanics, Controllability, and Planning," Int. J. of Robotics Research, Vol. 15, No. 6, pp. 533–556, 1996.

[4] Y. Maeda, T. Nakamura, and T. Arai: "Motion Planning of Robot Fingertips for Graspless Manipulation," Proc. of 2004 IEEE Int. Conf. on Robotics and Automation, pp. 2951–2956, 2004.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars: "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," IEEE Trans. on Robotics and Automation, Vol. 12, No. 4, pp. 566–580, 1996.

[6] S. M. LaValle and J. J. Kuffner: "Rapidly-Exploring Random Trees: Progress and Prospects," B. R. Donald, K. M. Lynch, and D. Rus, eds., Algorithmic and Computational Robotics: New Directions, pp. 293–308, A. K. Peters, 2001.

[7] X. Ji and J. Xiao: "Planning Motion Compliant to Complex Contact States," Int. J. of Robotics Research, Vol. 20, No. 6, pp. 446–465, 2001.

[8] M. Yashima, Y. Shiina, and H. Yamaguchi: "Randomized Manipulation Planning for A Multi-Fingered Hand by Switching Contact Modes" Proc. 2003 IEEE Int. Conf. on Robotics and Automation, pp. 2689–2694, 2003.

[9] Y. Maeda and T. Arai: "A Quantitative Stability Measure for Graspless Manipulation," Proc. of 2002 IEEE Int. Conf. on Robotics and Automation, pp. 2473–2478, 2002.

[10] S. Hirai and H. Asada: "Kinematics and Statics of Manipulation Using the Theory of Polyhedral Convex Cones," Int. J. of Robotics Research, Vol. 12, No. 5, pp. 434–447, 1993.

[11] M. H. Raibert and J. J. Craig: "Hybrid Position/Force Control of Manipulators," ASME J. of Dynamic Systems, Measurement, and Control, Vol. 102, No. 2, pp. 126–133, 1981.

[12] T.-T. Wong, W.-S. Luk, and P.-A. Heng: "Sampling with Hammersley and Halton Points," Journal of Graphics Tools, Vol. 2, No. 2, pp. 9–24, 1997.

[13] Y. Maeda and T. Arai: "Automatic Determination of Finger Control Modes for Graspless Manipulation," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2660–2665, 2003.

[14] Motion Strategy Laboratory, University of Illinois: The Motion Strategy Library, `http://msl.cs.uiuc.edu/msl/`.