MEAM 520 Final Project Report: Non-Cooperative Locally Dynamic Trees (NCLDT)

Shrenik Muralidhar and Jay Anjankar

December 11, 2017

1 Abstract

Our project aims to create a novel probabilistic path planning algorithm as an extension to RRT. While RRT samples points in the free configuration space and rejects them if there is no straight line path from it to the nearest node, our algorithm grows a tree from the sampled node. This ensures that even if the sampled point is not completely favorable (in terms of location relative to nearby obstacles), it still has a chance to dynamically move around and find a no-collision path to the target or some other tree.

Each tree has a specific target node that it tries to connect to and specific directions of growth. The directions of growth depend on the end node q_{start} and the position of the node itself. An initial batch of trees are uniformly or radially sampled in the configuration space at the beginning of the search to make up for localized searches. Each tree has a set of parameters and rules by which it grows. It also has an energy level which is used to decide whether to grow or decay the tree. New trees are grown and old trees decayed so as to keep the energy of the system constant. This energy is a parameter of the algorithm and must be chosen before hand like some of the other parameters. The decay (deletion) of trees is another factor that distinguishes our algorithm from existing methods (INSERT REFERENCES HERE).

The trees continue to grow and decay till a path to the target is found. The algorithm is probabilistically complete as in the worst case scenario (All the sample points are bad), it reduces to a regular RRT.

NCLDT aims to make it easier to find effective solutions to spaces where usual sampling based methods find it hard to find solutions (such as the narrow passage problem). Such problems only get worse as the number of dimensions increase and the obstacles start occupying more volume in the space.

2 Method

In this section, we elucidate the specifics of our model and its parameters. The following are the steps our algorithm takes to solve the search problem:

1. **Initial batch sampling:** The entire space is uniformly sampled initially. This can be done in a grid, radially or any other pattern. Each sample then forms a root node for a particular tree which would be grown by a governing algorithm described in the next subsection.

2. **Grow/Decay trees:** Individual trees are grown from the root nodes. Each tree is given an energy value that changes depending on how much it has grown. Stagnating trees would have their energies reduced and will eventually be candidates for deletion.

3. **Biased Sampling:** The concentration of trees provides a good measure of effective places to sample in the configuration space. This information can now be used to make more informed sampling decisions.

4. **Connecting trees:** Trees are connected either to the end configuration point or other trees depending on certain conditions. Once a tree connected path is formed from the start to the end configuration, the search problem is complete.

The ability for localized trees to grow around obstacles and connect to other trees is possible in our model because we inhibit omni-directional growth. Each tree in our model has a specific target node that it tries to attach itself to and specific directions along which it can search and grow. Hence the crux of our algorithm is the model that governs the growth of individual trees. The following subsection outlines all the details regarding the single tree growth model.

2.1 Single Locally Dynamic Tree

2.1.1 Growth Directions

As described earlier, each tree tries to grow towards a particular target node, while growing in specific directions which must be chosen. To make this decision process effective, the directions are based on the target configuration q_{end}, start configuration q_{start} and current configuration q_{root} of the sampled root node for the tree.

Two directions are defined for each tree. The first direction ω_t is the direction towards the target node. For now, the target node is assumed to be the end configuration q_{end} (Later while dealing with multiple trees, the target node can also be set as a node in a nearby connected tree). This direction is given as a unit vector that points from the start node to the end node.

$$\omega_t = \frac{q_{end} - q_{root}}{\|q_{end} - q_{root}\|}$$

Each tree is given a second direction ω_s which acts as a fall back direction of growth if growing along ω_t is unfavorable. It is given as the direction from q_{start} to q_{root} (Away from q_{start}).

$$\omega_s = \frac{-(q_{start} - q_{root})}{\|q_{start} - q_{root}\|}$$

Hence the direction of growth for a tree can be thought of as being between the two unit vectors ω_t and ω_s.

ADD TWO IMAGES HERE. ONE TO SHOW WT AND WS. THE OTHER TO SHOW OVERLAPPING GROWTH REGIONS.

The figures above show the direction for a particular start, end and root node configurations, as well as overlapping growth regions in the case of multiple trees. Hence the disadvantages of localizing growth directions may be negated by getting enough initial tree roots such that all the localized growth and search regions when summed up accounts for most of the space. Even if this is not the case, the algorithm would sample new trees which would start covering more of the space.

2.1.2 Growing trees by sampling

Once the root node has been set up, the tree will have to be grown by adding nodes to the root node. This is done by sampling along the current direction of growth (ω_{tc} which is either equal to ω_t or ω_s) at a specified distance within certain angle limits. To make it more probabilistically flexible, nodes are sampled along ω_{tc} at a distance between two values ϵ_{min} and ϵ_{max}. The samples are also constrained to be within an angle α in this region. The sampling radius is denoted by ρ. The initial value of ρ is given by rho_0 and is a parameter that will have to be decided beforehand. The radius keeps getting updated every iteration and is given by ρ_c (Current value). Each iteration, the values of ϵ_{min} and ϵ_{max} are updated depending on ρ_c. These are detailed in section ENTER SECTION HERE.

FIGURE OF SAMPLING

The figure above shows the sampling procedure for growing trees with parameters ϵ_{min}, ϵ_{max} and α. Every iteration, m nodes are added to each tree to grow them. As the value of m increases, the computational complexity also increases as all the m nodes are taken into account while updating parameters for the growth algorithm. A large value of m also ensures enough samples which leads to a more effective tree path.

In the next iteration, from all of the sampled nodes, the node closest to the root node is chosen to be the pivot node q_{pivot}. Samples in this iteration are now taken about q_{pivot} along w_{tc} parameterized by updated values of ϵ_{min}, ϵ_{max} and α. Every iteration, the value of q_{pivot} is updated and new samples are computed from it. This way, the tree is grown in the required direction.

There are multiple ways of sampling nodes for tree growth. The most trivial way is to keep sampling a random point in the configuration space till the sampled point meets the required restrictions. This method needed to sample a large number of points to get a valid sample, especially if the sampling parameters have small magnitudes. Hence for our model, we switched to a method using generalized spherical coordinates. ADD REFERENCE.

Given a radius r, we can directly get a sampled point by solving for the components of the coordinates as:

$$\begin{aligned}
x_1 &= r\cos(\phi_1) \\
x_2 &= r\sin(\phi_1)\cos(\phi_2) \\
x_3 &= r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\
&\vdots \\
x_{n-1} &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\
x_n &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1})
\end{aligned}$$

To account for the range of values of sampling radius, we randomly select r as a value in $[\epsilon_{min}, \epsilon_{max}]$. The angles ϕ_1 to ϕ_{n-1} are also picked randomly. ENTER DETAILS AFTER CHANGING ALGORITHM.

Hence using this method to sample, saves a lot of computation time as this process will have to be repeated every iteration for growing tree.

2.1.3 Updating current direction of growth

The direction of growth (direction of sampling) will need to be updated depending on the location of obstacle in its vicinity. If there are no obstacles along a straight line path from its current set of m sampled nodes to the target node, then the path connects directly. If there are a lot of obstacles surrounding the current set of m nodes, it would be better to just decay the tree and spend more computation time on a newly sampled tree root that would probably be at a better location. Our algorithm focuses on making this decision (On the favorability of the tree growth, given the current set of m nodes) and updating ω_{tc} to enable the tree to find its way around obstacles in its vicinity (Provided the number of obstacles in its vicinity is not too high).

Implementation Details

Initial Batch Sampling

Our algorithm makes full use of an initial uniform sampling due to the way the trees are set up to search. Each tree searches in a localized region that is partly determined intrinsically by the position of its root node in the configuration space. A good initial uniform sample thus ensures that our model has enough trees at the start to cover all search directions.

Grow/Decay Trees

Unlike RRT's, the individual trees in our algorithm do not have the capability of growing in any direction to any extent. Each tree is characterized by two directions of growth: The target direction ω_t which is the direction of the target node it is trying to reach. This may be the end configuration or some other connected tree. The second direction is an intrinsic direction ω_s that is defined by the position of the root node of the tree. This direction is in the direction from the start configuration to the root node. Hence for a single tree, the growth direction would be between these two directions.

From the root node, other nodes are sampled every iteration at a radius between $\epsilon_m in$ and $\epsilon_m ax$. The sampling is also carried out in an angular spread of α. A total of m nodes are sampled in each tree every iteration. The nearest node is taken as the pivot node and all further samples are taken from this node. We classify all the individual nodes to make a decision regarding the direction of growth. The exact mathematical details are given in the next section.

The obstacle search radius is also varied dynamically. If the tree grows without any change in direction, the radius is increased to make the tree more distance sensitive to obstacles, whereas if an obstacle is encountered, the obstacle search radius is reset to ensure that the growth is slowed down till it acquires some momentum again.

Each tree is given an energy value depending on its spread. Its spread is computed by the Euclidean distance between the current and previous pivots. Every iteration, the energy value either remains unchanged or reduces. Once the energy value goes below a certain threshold, the growth of the tree is stopped. More trees are then spawned so as to keep the total sum of energies of all trees equal to a specified energy value.

Biased Sampling:

Our algorithm also gives us information regarding good and bad sampling areas in the configuration space. This is obtained by analyzing the spread and congestion of trees. As we do a uniform sampling initially, we can assume that each section of the workspace has some tree population. For areas in the configuration space that are not very favorable (Lots of obstacles, etc), trees grown would be dense and would then be decayed. This gives us valuable insights on which part of the space to sample additional trees in when required, something that was not available in RRTs.

Connecting Trees

ω_t for the trees is updated using a probabilistic model. This ensures that the trees can connect to other connected trees as well and not just the target node configuration. This is the final piece in putting together locally dynamic trees. The mathematical basis for making the growth direction update decision is given in the next section.

Mathematical Analysis

Computation of ω_t and ω_s

ω_t is defined as the vector pointing from a root node to the target node. Initially, this vector is directed from the root node to the target. However, as more and more trees connect to the target node, ω_t is dynamically changed to grow either towards the target node or connected nodes.

$$\omega_t = \frac{q_{target} - q_{root}}{\|q_{target} - q_{root}\|}$$

As seen, ω_t is a unit vector in the direction specified. q_{target} can either be the end configuration q_{end} or one of the nodes in a connected tree.

ω_s is the vector that gives the intrinsic direction of every root node. This is a vector directed from the root node to the sampled point.

$$\omega_t = \frac{q_{root} - q_{start}}{\|q_{root} - q_{start}\|}$$

As we want our trees to be dynamic and move around obstacles to a limited exted, the direction of growth of the tree must be varied accordingly. As mentioned earlier, each tree focuses on growth in a localized direction, given by the angle between ω_t and ω_s. It cannot grow in a direction angled greater than these two end vectors. Every iteration, the direction of growth is computed as $\omega_{current}$ and is in between ω_t and ω_s. Right now, we implement a naive approach where the current direction of growth is switched from being either ω_t or ω_s. This ensures that the model is more robust as needing to grow around obstacles while computing a real valued direction between the two closed limits would require more complex growth algorithm.

To decide on the direction of growth, each of the m sample points are put in either or both of the sets η and μ. η consists of all nodes (out of the m) that would not collide with any obstacle around a obstacle search radius of ρ in the direction of ω_t. μ consists of a similar set but now in the direction ω_s. By weighing the relative sizes of these sets, we decide if the tree has to be grown or decayed, as well as the direction of growth.

ϵ_{min} and ϵ_{max}

$\epsilon_m in$ and $\epsilon_m ax$ determine the extent of growth for a tree from its root node. These parameters perform sampling along a probabilistic radius to ensure a good even spread of sampled points along the angle spread α as well.

Computation of ρ

ρ is the obstacle search radius which defines the sensitivity of obstacle search. A low value means that it is less sensitive in the sense that it requires the tree to be closer to the obstacle to detect it and change directions. Hence our approach is to constantly increase ρ if the direction of the tree does not change. Once it does change, its value is reset to the low initial value ρ_0. ρ is thus computed by taking into account $\|\omega_{current} - \omega_t\|$ as this defines the changed in direction each iteration.

Sampling

For an n-dimensional space, sampling is done in an ϵ ball in $\mathbf{R}^n$ along a radius $r \in [\epsilon_{min}, \epsilon_{max}]$ within an angle α. This is done using generalized spherical coordinates in n-dimensions, saving us from having to sampling points and then deciding if it satisfies the necessary conditions.

q_{pivot} is taken as the node nearest to the previous pivot node from among the m sampled nodes and the next sample is taken with q_{pivot} as the center of the ϵ ball. -

Direction change

The direction of ω_t will have to be changed depending on the connectivity of other trees. If a lot of other trees are connected and the tree under observation is close to one of them, it makes sense to have this tree connect to the connected tree instead of the target. Hence ω_t is chosen using a probabilistic approach that takes into account the number of non-connected trees, as well as the Euclidean distance of the tree from other connected trees. If the distance is less or if most other trees have connected to q_{end}, then there is a higher probability for the tree to move towards and connect to the closes node in the connected tree.

This direction update is carried out for all trees every iteration.

Results and Analysis

1. **Narrow Passage:**NCLDT performs well with respect to the narrow passage problem. The concept of uniformly sampling points across the configuration space ensures that the effect of different obstacles wouldn't adversely affect the results. As the narrow passage problem becomes more difficult to solve as the dimension of the space increases, NCLDT would become more effective.

2. **Sampling points:** NCLDT provides a very good estimate of good sampling points in the space which can then be used to make the search increasingly efficient as the algorithm runs.

3. **Dynamic Parameters:** Dynamic parameters like the number of trees, search radius among others ensure that the path search can be modified to provide better results as and when required.

4. **Probabilistically complete:** NCLDT is an extension to RRT and hence, probabilistically complete. If a path exists between the start and target nodes, NCLDT will find it.

Future work

1. **Decay trees:**Future implementation would incorporate the concept of decaying trees after their energy drops below a certain minimum threshold to ensure computation to grow such trees is avoided.

2. **Sampling bias:** Using favorable sampling regions provided by NCLDT to better sampling efficiency and speed up the search.

3. **Extend the algorithm for higher dimensions:** NCLDT has been designed for 2D configuration space. However, the same concepts can also be used for higher dimensional configuration space.