HOMOTOPY CONSCIOUS ROADMAP CONSTRUCTION BY FAST SAMPLING OF NARROW CORRIDORS

Rahul Kala
Robotics and Artificial Intelligence Laboratory,
Indian Institute of Information Technology, Allahabad,
Allahabad, Uttar Pradesh, India
Email: rkala001@gmail.com
Web: rkala.in

Citation: R. Kala (2016) Homotopy conscious roadmap construction by fast sampling of narrow corridors, Applied Intelligence, 45(4): 1089-1102.

Full Version Available At: https://link.springer.com/article/10.1007/s10489-016-0808-9

Abstract: Probabilistic Roadmaps are increasingly being used for robot motion planning. The method makes use of an offline construction of a roadmap. Even though the method is offline, it needs to be initially constructed as quickly as possible for efficient and near-real time initial motion of the robot. The challenge lies in sampling of multiple narrow corridors wherein the probability of samples is very low. It is important to discover all homotopic groups very early to make good initial decisions from the roadmap. Missing out of even a single homotopic group can lead to no solution or poor solutions. The proposed method uses a multi-strategized approach to sampling of the initial points and then intelligently constructs edges between the points in a multi-strategized manner. The aim is to increase sampling at the narrow corridors and then to facilitate edge connectivity of nodes inside the corridor with the rest of the roadmap, so as to lead to discovery of all possible homotopies between any pair of sources and goals. The approach results in better performance as compared to uniform sampling and obstacle based sampling.
Keywords: *Robot Motion Planning, Homotopy, Sampling based Approaches, Probabilistic Roadmaps.*

1. Introduction

Probabilistic Roadmaps (PRM) [1, 2] are extensively used for motion planning of a single or multiple mobile robots. The methods first construct a roadmap ($\zeta<V,E>$) of the obstacle-free configuration space (C^{free}) by sampling of points which prospectively become the vertices of the roadmap (V) and then connecting the neighboring vertices by using a local search algorithm using trajectories which become the edges of the roadmap (E). A popular method is to use a straight line collision checking algorithm as the local planner. An all-connected approach checks for collisions between all pairs of vertices. The approach however has a very high computational complexity making it worthless when the initial roadmap needs to be computed in a small computational time. It is common to limit the collision-checking to only k-nearest neighbors [3-4] or vertices within a radius of k [3,5] to limit connectivity and reduce the computation time. The roadmap (ζ) is then used for the online planning. Since the number of vertices and edges are limited, a near-real time response to the motion planning queries is realizable.

The algorithm comes under the class of sampling based approaches which try to generate a representation of the collision-free configuration space by generation of random samples. The sampling based approaches face the problem of narrow corridor, wherein the probability of generation of random samples inside the narrow corridors is very low which cannot be located easily. Several common techniques have been used for solving the narrow corridor problem. In obstacle-based sampling [6] a sample generated inside an obstacle is promoted to the nearest obstacle-free region. The samples generated inside the corridors hence include the samples generated in the nearby obstacles which make the narrow corridor, the probability of generation of which is significantly high. Gaussian sampling [7] is another popular method wherein the probability of acceptance of a sample diminishes in a Gaussian manner from the obstacle boundary. By encouraging generation of samples near the obstacle boundary, the possibility of discovering the narrow corridor increases significantly.

Bridge-test [8] specifically aims in sampling inside the narrow corridor. In this method every sample is checked whether it lies inside a narrow corridor. For every sample the test generates two more samples in opposing directions. If both happen to be placed inside obstacles, while the initial sample is obstacle-free, the sample is said to be placed inside a narrow corridor. A variety of sampling techniques may be generated by their hybrids or adaptive combinations [8, 9]. The hybrids certainly benefit from the combined advantages of the base approaches, however getting a mix of narrow corridor biased sampling and a general obstacle-prone sampling is always hard. Besides, discovery of a narrow corridor doesn't necessarily redundantly and adequately connect it to the rest of the roadmap, a challenge largely left in the literature.

A large and complicated configuration space may have multiple narrow corridors amidst numerous wide open spaces. The narrow corridors are initially hard to cite and connect in a roadmap architecture. While the roadmaps are intended to be initially constructed in an offline manner, requirements of rapidly changing environments, fast initial motion of the robot or robotic team, etc. may demand a fast and near-real time construction of the roadmap. Discovering all the narrow corridors and redundantly connecting them with the roadmap plays an important role towards the optimality and completeness of the solution. The roadmap must consist of the least number of vertices and edges (for minimizing roadmap construction and query time), while leading to discovery of all possible homotopic groups of paths between all possible sources and goals. Two paths are said to be in the same homotopy [10] if one can be produced from the other by multiple infinitely small deformations, such that all intermediate paths are collision-free. The concept is shown in Figure 1. Discovery of all homotopic groups guarantees a path for every pair of valid sources and goals, or in simple words the approach is complete. A deformation retract [11] is the roadmap formed by shrinking all open spaces and reducing them to simple curves. Even though a deformation retract of the configuration space serves as an ideal roadmap, some redundant nodes and edges may be acceptable and even preferable to cater to optimality.

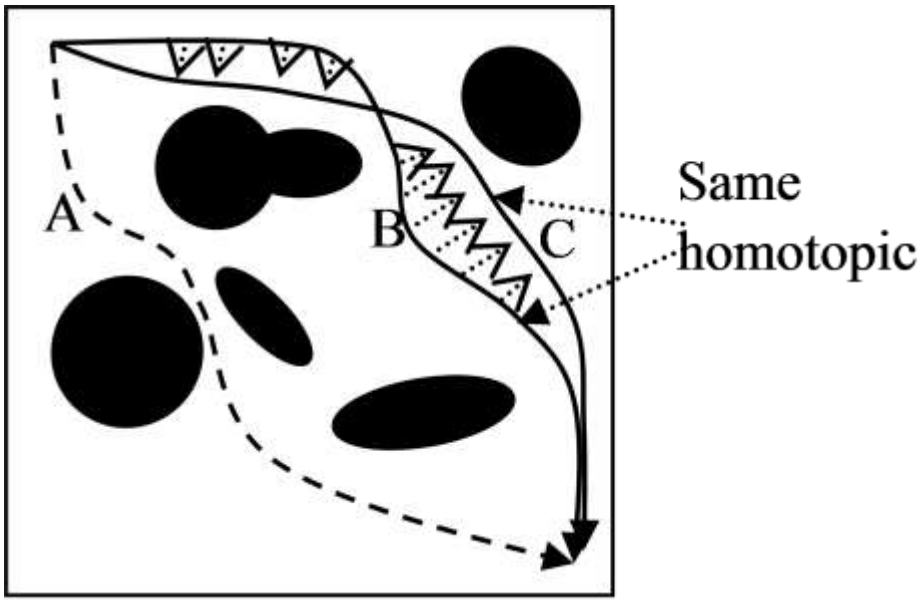

Figure 1. Homotopies. Trajectory *B* and *C* belong to the same homotopic group, while *A* does not belong to the homotopic group of *B* and *C*.

Elimination of redundant edges and cycles is one of the most primitive ways of reducing the roadmap construction as well as the query time. Visibility PRM and related techniques [12-13] sample out the configuration space with the same fundamental, however redundancy is needed to mine out all possible homotopies ultimately affecting the roadmap optimality. Redundant edges may later be admitted [14-15]. An alternative to PRM is the Rapidly-exploring Random Tree (RRT) [16,17] algorithm which are single-query algorithms and attempt to quickly compute a path between the source and the goal without necessarily trying to explore the complete search space. RRT* [3] algorithm can use excess computation time to iteratively improve the solution. Attempts are further made to extend the computational ability of RRTs for the production of roadmap, like the Sampling based Roadmap of Trees (SRT) [18] and Rapidly-exploring Random Graphs [19]. In an earlier work

by the author, RRT style exploration was used to construct a redundant roadmap in a time-efficient iterative manner [20]. These techniques however are not well suited for narrow corridor biased sampling.

Approaches like graph spanners [21-22] restrict the number of edges by removing redundant edges if they do not result in getting paths worse by a factor or more than *k*. However they do not solve the problem of timely roadmap construction at the first instance. Given an initial roadmap, approaches like Adaptive Roadmap [23-24] and Elastic Roadmap [25-26] can be used for maintenance and extension of the roadmap for a dynamic environment. If initially a good roadmap can be obtained, it becomes possible to model the environment dynamics to adapt the roadmap as the environment changes. The adaptations may even be applied to only the trajectory being followed by the robot (e.g. [27]), obtained from some roadmap based approach.

The key contributions of the work are: (a) A narrow corridor sampling strategy is proposed which combines the obstacle based sampling strategy along with a bridge-test sampling strategy. The resulting sampling strategy is a single hybrid strategy and not adaptive independent calls to different strategies in an ensemble architecture popular in the literature. (b) The narrow corridor sampling strategy is called in conjuncture with other strategies to suit diverse environment types including the ones with numerous narrow corridors, no narrow corridors and wide open spaces. (c) It is proposed to add relevant and strategized vertices in the edge connectivity mechanism to aid in redundant roadmap connectivity. (d) A multi-strategized edge connectivity mechanism is proposed to aid in redundant connectivity between all vertices of the roadmap, thereby also leading to complete homotopy discovery. (e) The proposed approach is seen to be significantly better than obstacle based and uniform sampling.

The paper is organized as follows. Section 2 presents the algorithm framework. This includes the roadmap generation phase (sections 2.1-2.3) as well as the query phase (section 2.4). The roadmap generation phase includes the addition of vertices (section 2.1) and edges (section 2.2), followed by the pseudo-codes (section 2.3). The results are given in section 3. Section 4 gives the conclusions.

2. Algorithm

The first task associated with the algorithm is to construct a roadmap ($\zeta<V,E>$) consisting of the vertices V and edges E, which can then be used for robot path planning. Let the configuration space be given by C and let the free configuration space be given by C^{free}. Let C^{obs} be the obstacle prone configuration space, which gives $C^{free}=C\backslash C^{obs}$. Non-strictly, first the vertices are computed. Once a set of vertices are available, they are connected by adding some edges.

2.1 Computing Vertices

The first task is computation of the vertices that make up the roadmap. Three strategies are proposed to generate samples which become the vertices of the roadmap:

- *Narrow Corridor Sampling Strategy*, which attempts to sample out all narrow corridors;
- *Obstacle based Sampling Strategy*, which produces samples around obstacles which are critical for a roadmap; and
- *Uniform Sampling*, to add samples for optimality in case of wide open spaces.

The first strategy is known as *Narrow Corridor Sampling strategy*, using which the emphasis is only to find the samples which lie inside some narrow corridor. Narrow corridors are important to sample out almost completely in order to expect discovery of all possible homotopic groups in the generated roadmap between all possible sources and goals.

In this strategy, we first sample out a point in the obstacle prone configuration space ($q_o=U(C^{obs})$). The sampled point is advanced to the nearest free configuration space given by equation (1). The algorithm uses a bi-resolution strategy to search for q_o^{free} by first searching at a higher resolution, and then searching the obtained boundary using a finer resolution.

$$q_o^{free} = q_o + d(\hat{\theta}) : q_o + d(\hat{\theta}) \in C^{free},$$
$$q_o + d_1(\hat{\theta}) \in C^{obs} \forall d_1 < d \quad (1)$$

Here $\hat{\theta}$ is a randomly sampled direction. If $\hat{\theta}$ does not result in a point in the free but bounded configuration space, a new direction is chosen.

To test whether the point q_o^{free} represents a point inside a narrow corridor, a bridge-test is used. A point q_D is generated at a distance of D from q_o^{free} (that is $q_D = q_o^{free} + D(\hat{\theta})$). If q_D is collision-prone, and knowing that q_o^{free} lies on an obstacle boundary at direction of $-\hat{\theta}$, it can be stated that q_o^{free} does represent a narrow corridor of minimum width D. The attempt is to sample out the point in the middle of the narrow corridor so as to facilitate easier connection of the sample to the rest of the roadmap, for which a point in the other corridor boundary is required given by equation (2). The sampled point inside the narrow corridor is given by equation (3).

$$q_D^{free} = q_D - d(\hat{\theta}) : q_D - d(\hat{\theta}) \in C^{free}, q_D - d_1(\hat{\theta}) \in C^{obs} \forall d_1 < d \, . \quad (2)$$

$$q_C = \left(q_o^{free} + q_D^{free}\right)/2 \, . \quad (3)$$

The least distance between any two vertices of a roadmap is taken to be Δ ($\leq D$). This disallows placing vertices too close to each other, thus resulting in too many vertices. In order for the vertex q_C to be admissible in the roadmap with vertices V, this distance condition must hold (that is $\|q_C - v\| \geq \Delta \forall v \in V$).

The maximum attempts of finding a narrow corridor are restricted to n_C. A good narrow corridor sampling strategy must find all narrow corridors, if they exist, while exit gracefully after a small computation if the environment is free of narrow corridors. The proposed approach uses a combination of obstacle-based sampling strategy along with a bridge-test to quickly discover the narrow corridor. The use of bridge-test alone results in poor probability of finding a narrow corridor (which is still larger than a uniform sampling), while obstacle based sampling alone can result in generation of most of the samples around obstacles and not necessarily the narrow corridors, if the narrow corridor is a small part of the otherwise cluttered environment.

Samples inside the narrow corridors are of significant value in quick generation of roadmaps. However the strategy is supplemented by two other strategies, which also suite cases of low or no occurrence of narrow corridors; and spaces with sparsely occupied obstacles. The second strategy of use is an *obstacle based sampling* strategy where random samples in the obstacle prone configuration space are generated ($q_o = U(C^{obs})$) and then moved to the nearest point in the free configuration space(q_o^{free}) as indicated in equation (1). For a sample to be admissible as per this strategy, it is required to have at least a clearance of Δ, which gives enough scope for the edge connection algorithm to connect the sample redundantly to the rest of the roadmap. For this reason the q_o^{free} sample obtained equation (1) is further moved in the same direction by a magnitude of Δ to obtain a sample q_B given by equation (4).

$$q_B = q_o^{free} + \Delta(\hat{\theta}). \quad (4)$$

The sample is admitted in the roadmap if it is collision-free and at a distance of more than Δ from any other vertex in the roadmap (that is $q_B \in C^{free} \wedge \|q_B - v\| \geq \Delta \forall v \in V$). Failure of admissibility due to lack of maintenance of the minimal clearance is an indication of a narrow corridor which is admissible as per the narrow corridor sampling strategy. The strategy is applied for a maximum of n_B attempts.

The third and the last strategy is to *uniformly sample* the free configuration space ($q=U(C^{free})$). The strategy is applied for a very small number attempts (n_U) as it is only of value in sparely occupied scenarios while in some cases it may also result in connecting some challenging samples to the rest of the roadmap.

2.2 Computing Edges

Fundamentally the aim of this step is to connect the vertices generated by the strategies mentioned in section 2.1 into a roadmap structure using edges (E). However the process of connecting the different vertices may involve adding new vertices. At any time the roadmap may have vertices in groups of disconnected components. Two vertices a and b lie in the same connected component if there is a path from a to b and vice versa as per the current roadmap. The edge connectivity function is forcibly made symmetric and hence all edges are non-directed and a reverse path always exists. Initially all vertices are into different disconnected components. Some of the connections are very easy and less time consuming to make. Hence, stating from an all-disjoint graph, the algorithm initially adds up all simple connections, resulting in only a few-disjoint components. The addition of edges then aims at making sure all the components are connected, to aid in completeness and discovery of all homotopic groups; as well as to add redundant connections for the sake of optimality. The former is done by focusing on means to add vertices and edges in pursuit of connections between otherwise disconnected components. The latter is done by exploration through different strategies, namely leaf exploration, hash based exploration and random exploration.

2.2.1 Terminology

Let $\kappa(a)$ denote the collection of connected vertices to which a belongs. Initially all vertices are disconnected as no edges exist and hence $\kappa(a)=\{a\}\forall a \in V$. Since connectivity of the roadmap is of importance and disconnected components are discouraged, we maintain the following data structures throughout the edge connectivity procedure:

- $|\kappa(a),\kappa(b)|$: The distance between the connected components $\kappa(a)$ and $\kappa(b)$, defined as the minimum distance between any vertex x in $\kappa(a)$ with any vertex y in $\kappa(b)$, or equation (5).

$$|\kappa(a),\kappa(b)| = \min_{x\in\kappa(a),y\in\kappa(b)}\|x-y\|. \quad (5)$$

- $x[\kappa(a),\kappa(b)]$: The vertex x in $\kappa(a)$ which records the minimum distance of $|\kappa(a),\kappa(b)|$ with any vertex y in $\kappa(b)$, or equation (6).

$$x[\kappa(a),\kappa(b)] = \arg\min_{x\in\kappa(a),\forall y\in(b)}\|x-y\|. \quad (6)$$

- $y[\kappa(a),\kappa(b)]$: The vertex y in $\kappa(b)$ which records the minimum distance of $|\kappa(a),\kappa(b)|$ with any vertex x in $\kappa(a)$, or equation (7).

$$y[\kappa(a),\kappa(b)] = \arg\min_{y\in\kappa(b),\forall x\in(a)}\|x-y\|. \quad (7)$$

- κ: The set of distinct non-components in the roadmap. Initially all components are non-connected and hence $\kappa=V$. In general, $\kappa=\bigcup_a \kappa(a)$.
- *Failed*($\kappa(a)$, $\kappa(b)$): The number of failed attempts in connecting two disjoint sets vertices $\kappa(a)$ and $\kappa(b)$.
- $\Psi(z)$: The configuration space is mapped to a lower dimensional space $\Psi(.)$ using a hash function and every cell z of the reduced space stores a count of the total number of vertices which lie within that cell. The maximum occupancy of the cell z, denoted by $|\Psi(z)|$, is given as the volume of the free configuration space (C^{free}) covered by that cell. The ratio $\Psi(z)/|\Psi(z)|$ is known as the desirability ratio of the cell and corresponds to the degree of under-occupancy of the cell.

2.2.2 Initial Edge Connection

A limited connectivity roadmap is aimed, wherein connections are attempted between any vertex to the neighboring *k* vertices. Initially there are no edges and hence an initial attempt to get some connected roadmap is made by trying an *initial edge connection strategy*. The roadmap (not necessarily completely connected) is then extended by adding edges (and vertices) using a variety of other strategies.

Initially an attempt is made to connect every vertex to the neighboring *k* vertices. Initially only straight line connections between the vertices are attempted. Suppose a connection attempt between a vertex v_1 and a neighboring vertex v_2 fails, as the straight line from v_1 to v_2 intersects some obstacle at point o_1 while travelling from v_1 to v_2. A point (o_1^{Δ}) at a distance of Δ from o_1, towards v_1, is added as a new vertex, given by equations (8-9).

$$o_1 = v_1 + d(\hat{\theta}) : v_1 + d_1(\hat{\theta}) \in C^{free} \forall d_1 < d, v_1 + d(\hat{\theta}) \in C^{obs} . \quad (8)$$

$$o_1^{\Delta} = o_1 - \Delta(\hat{\theta}). \quad (9)$$

The vertex o_1^{Δ} can only be added if the distance between points o_1^{Δ} and v_1 is at least 2Δ apart from the usual admissibility conditions for vertices. Distances smaller than this magnitude reflect that the points o_1^{Δ} and v_1 are nearly the same and hence an additional vertex is not preferable. The notations are shown in figure 2.

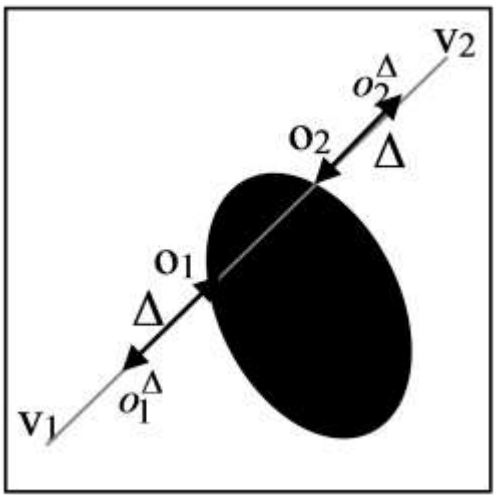

Figure 2.Adding new vertices in initial edge connectivity

v_1 and v_2 may be far apart and in many cases belonging to disconnected components of the roadmap. Absence of a straight line connection necessitates the use of local search methods to aim a connection. In this implementation redundant connections or useful cycles are also of value and hence attempting such connections is of value. Local search algorithms usually take a significant time, especially if the participating vertices are far away. A better way of attempting connections is by trying to add vertices which result in a connection between the two vertices. Searching for such bridging vertices in the entire region may be too costly with not necessarily a good probability of connection and a single vertex may not be capable of connecting the two vertices.

Hence a different strategy is used in the proposed algorithm. With a single connection attempt from v_1 to v_2 an obstacle prone region around o_1^{Δ} has already been identified. Similarly in the connection attempt from v_2 to v_1, obstacle region around o_2^{Δ} would be identified and a vertex added. In a couple

of connection attempts which is not cost intensive, two points o_1^{Δ} and o_2^{Δ} are easily obtained. The difficult problem of connecting v_1 and v_2, which may be far apart or separated with complex and unknown obstacle grids, has now been reduced to the problem of connecting points o_1^{Δ} and o_2^{Δ}, which may be close and separated by just an obstacle corner, building a small (non-straight) trajectory around o_1^{Δ} and o_2^{Δ} may be much easier. This connection is done using the regular set of edge connection strategy.

2.2.3 Connect Disconnected Vertices Strategy

After some initial edges have been quickly added using the strategy detailed above, the regular connection strategies continue. The first strategy is the *connect disconnected vertices* strategy. This strategy aims at taking two disconnected vertex sets (κ(*a*) and κ(*b*)) and connecting them by adding a suitable bridging vertex (*c*) that directly connects to both of them. The first set (κ(*a*)) is chosen randomly out of the available set κ. It is evident that vertex sets closer to κ(*a*) are more likely to be easily connected by addition of suitable bridging vertices. However even if two sets are very close to each other, connecting them may not always be possible and hence other alternatives also need to be explored. The second set κ(*b*) is chosen randomly from a Roulette Wheel selection where any set *i* had a probability of selection as given by equation (10).

$$P(\kappa(i)) = \frac{1 \Big/ |\kappa(i), \kappa(a)|}{\sum_i 1 \Big/ |\kappa(i), \kappa(a)|}. \tag{10}$$

The attempt is made to connect the closest located vertices given by $x[\kappa(a), \kappa(b)]$ and $y[\kappa(b), \kappa(b)]$. This is done by adding a new vertex *c* at a distance of δ in a random direction from $x[\kappa(a)]$ given by equation (11).

$$c = x[\kappa(a), \kappa(b)] + \delta(\hat{\theta}). \tag{11}$$

The vertex *c* is added only if the path from $x[\kappa(a)]$ to *c* is collision free, in which case the corresponding edge is also added. The path from *c* to $y[\kappa(b)]$ may not be collision-free and is not a criterion for admissibility of *c*. If the path comes out to be collision-free, the corresponding edge is also added and the two disjoint vertex sets are said to be merged and the updated metrics are computed. However if a feasible *c* cannot be computed, the failed attempts between the pair (*Failed*(κ(*a*), κ(*b*))) is added by 1. The selection of pair κ(*a*) and κ(*b*) can only happen if the failed attempts of the particular pair are less than a threshold of η. If multiple vertices are needed to connect the two disjoint sets, each such vertex can be added in separate iterations. Addition of a new vertex *c* also means that the connectivity of *c* with the nearest *k* nodes will also be checked, however here no new vertex may be added.

2.2.4 Leaf Expansion Strategy

The next strategy is the *leaf vertex expansion* strategy, wherein a random leaf vertex is selected from the entire roadmap in pursuit of its expansion. The leaf node in a roadmap is the one which has a single parent linking it to the rest of the roadmap. Such leafs are normally found at the edges of the expansive cover of the roadmap and are good candidates for extension of the cover of the roadmap. A random leaf node (q_L) is selected. The node is expanded in a random direction by a distance of δ to create a new vertex *c* similar to equation (11). The vertex *c* is added if a straight line collision-free path from *q* to *c* is possible along with the other necessary conditions. On addition of the vertex connections with the neighboring *k* vertices are also checked.

2.2.5 Hash Expansion Strategy

Even though a complete coverage of the entire configuration space is not the aim of the work, as it contradicts the aim of a quick initial roadmap generation, a small attempt is made in the direction

for the cases where prolonged computation is possible using the *hash expansion strategy*. A lower dimensional hash map $\Psi(z)$ is taken to store the number of vertices at every region. A poor desirability ratio $\Psi(z)/|\Psi(z)|$ means a need to expand in that region. Hence a region z is randomly selected using a Roulette Wheel Selection with the probability of selection of the region z given by equation (12).

$$P(z) = \frac{1 - \psi(z) \big/ |\psi(z)|}{\sum_{q} \left(1 - \psi(z) \big/ |\psi(z)|\right)}. \tag{12}$$

A random sample c is generated which maps to the cell z of the map. The closest node in the roadmap (v_1) is computed. If a collision-free path from v_1 to c exists along with the necessary admissibility criterions of c, the node c and the corresponding edge are added. Connectivity to the nearest k neighboring edges is checked.

2.2.6 Random Expansion Strategy

The last strategy is the *random expansion strategy*. A random sample ($q=U(C)$) is generated. If the sample is collision-prone ($q \in C^{obs}$), it is promoted to the nearest free configuration space using equation (1) to generate a new sample q_o^{free}. The sample (q_o^{free}) is used to attract the roadmap towards itself. The nearest vertex in the roadmap (v_1) is extended towards the sample to create a vertex (c), which is added as per the necessary admissibility criterions. This is another way of expanding the roadmap to other areas.

The strategies are implemented with probabilities p_κ, p_L, p_Ψ and p_R such that $p_\kappa+p_L+p_\Psi+p_R=1$. Initially, if the roadmap has disconnected components, the first strategy is dominant. However once the entire roadmap has been connected or some disconnected components have reached the failed attempts threshold, this strategy simply vanishes. The other strategies are for utilizing excess computation, making redundant connections and useful cycles, and trying new ways to connect disconnected components.

2.3 Pseudo Code

The algorithm is presented as pseudo code 1 to 10. The different components correspond to the concepts discussed in sections 2.1 and 2.2.

Pseudo Code 1: GenerateVertices(C)
$\Psi(z) \leftarrow 0 \; \forall$ z, V$\leftarrow$NIL
for n_C failed attempts (narrow corridor sampling)
    calculate q_o, q_o^{free}, q_D and q_C from equations (1), (2) and (3)
    if $\|q_C - v\| \geq \Delta \forall v \in V$, V $\leftarrow$ V $\cup$ q_C, $\Psi(\text{hash}(q_C)) \leftarrow \Psi(\text{hash}(q_C))+1$
for n_B failed attempts (obstacle based sampling)
    calculate q_o, q_o^{free} and q_B from equations (1) and (4)
    if $\|q_B - v\| \geq \Delta \forall v \in V$ V$\leftarrow$V $\cup$ q_B, $\Psi(\text{hash}(q_B)) \leftarrow \Psi(\text{hash}(q_B))+1$
for n_U failed attempts (uniform sampling)
    q$\leftarrow$U(C^{free})
    if $\|q - v\| \geq \Delta \forall v \in V$, V$\leftarrow$V $\cup$ q, $\Psi(\text{hash}(q)) \leftarrow \Psi(\text{hash}(q))+1$

Pseudo Code 2: GenerateEdges(C,V)

E←NIL, $\kappa(v) \leftarrow \{v\} \forall v \in V$

$|\kappa(a), \kappa(b)| \leftarrow \|a-b\|, x[\kappa(a), \kappa(b)] \leftarrow a, y[\kappa(a), \kappa(b)] \leftarrow b,\ failed(a,b) \leftarrow 0, \forall a, b \forall V$

for v ∈ V, addKEdges(v) (initial edge connection strategy)

do until stopping criterion

with probability p_κ, ConnectDisconnectedVertices()

with probability p_L, LeafVertexExpansion()

with probability p_Ψ, HashExpansion()

with probability p_R, RandomExpansion()

Pseudo Code 3: ConnectDisconnectedVertices()

if |κ|≤1 return null

κ(a) ←U(κ)

calculate κ(b) using roulette wheel selection using equation (10)

if failed(κ(b), κ(b))<η

calculate c using (11)

if addVertexAndEdge(x[κ(a), κ(b)],c)

if $q \in C^{free} \forall q \in Path(c, y[\kappa(a), \kappa(b)])$

E←E ∪ (c, y[κ(a), κ(b)]) ∪ (y[κ(a), κ(b)], c)

κ(a)← κ(a) ∪ κ(b), κ← κ-b

addKEdges(c)

else

failed(κ(b), κ(b)) ← failed(κ(b), κ(b))+1

Pseudo Code 4: LeafVertexExpansion()

q_L←U(V): q_L is a leaf node, addVertex(q_L)

c← $q_L + \delta(\hat{\theta})$ for random $\hat{\theta}$

if addVertexAndEdge(q_L,c), addKEdges(c)

Pseudo Code 5: HashExpansion()

calculate z using equation (11)

q←U(C^{free}): hash(q)=z

if $q \in C^{free} \wedge \|q-v\| \geq \Delta \forall v_1 \in V$

v_1← arg $\min_v \|v-q\|$, c← $q + \delta(\hat{\theta})$

if addVertexAndEdge(v_1,c), addKEdges(c)

Pseudo Code 6: RandomExpansion()

q←U(C), if q ∈ Cobs, move q using equation (1)

if $q \in C^{free} \wedge \|q-v\| \geq \Delta \forall v_1 \in V$

v_1 ← arg $\min_v \|v-q\|$, c← $q + \delta(\hat{\theta})$

if addVertexAndEdge(v_1,c), addKEdges(c)

Pseudo Code 7: addVertexAndEdge(q,c) (adds a new vertex c and the edge from q to c)

if $\left(c \in C^{free}\right) \wedge \left(\|c-v\| \geq \Delta \forall v \in V\right) \wedge \left(q_1 \in C^{free} \forall q_1 \in Path(q,c)\right)$

V←V ∪ c, Ψ(hash(c)) ← Ψ(hash(c))+1,

E←E ∪ (q,c) ∪ (c,q), update(κ,c)

return true

else

return false

Pseudo Code 8: addKEdges(v) (checks nearest k vertices and adds edges if feasible)

$V_N \leftarrow k \arg\min_{v_1 \neq v} \|v - v_1\|$

for $v_1 \in V_N$

if $(v, v_1 \notin E \wedge (q \in Path(v, v_1)))$,

$E \leftarrow E \cup (v,v_1) \cup (v_1,v)$

Update(κ,v,v_1)

else if initial edge adding strategy $\wedge \|v, o_1\| > 2\Delta$

calculate o_1^{Δ} by equations (8) and (9), $\Psi(\text{hash}(o_1^{\Delta})) \leftarrow \Psi(\text{hash}(o_1^{\Delta}))+1$

$E \leftarrow E \cup (v, o_1^{\Delta}) \cup (o_1^{\Delta}, v)$, $\kappa(v) \leftarrow \kappa(v) \cup o_1^{\Delta}$

update(κ, o_1^{Δ})

Pseudo Code 9: Update(κ,c) (updates the variables related to κ after a new vertex c has been added)

for $v_1 \in V$

if $\|c - v_1\| > |\kappa(c), \kappa(v_1)| \wedge \kappa(c) \neq \kappa(v_1)$

$|\kappa(v_1), \kappa(c)| = \|c - v_1\|$, $x|\kappa(v_1), \kappa(c)| = v_1$, $y|\kappa(v_1), \kappa(c)| = c$

$|\kappa(c), \kappa(v_1)| = |\kappa(v_1), \kappa(c)|$, $x|\kappa(c), \kappa(v_1)| = c$, $y|\kappa(c), \kappa(v_1)| = v_1$

Pseudo Code 10: Update(κ,v,v_1) (updates the variables related to κ after a new edge (v,v_1) has been added)

for $k \in \kappa$

if $|\kappa(v_1),k| < |\kappa(v),k|$

$|\kappa(v),k| = |\kappa(v_1),k|$, $x|\kappa(v),k| = x|\kappa(v_1),k|$, $y|\kappa(v),k| = y|\kappa(v_1),k|$

$|k,\kappa(v)| = |\kappa(v),k|$, $x|k,\kappa(v)| = x|k,\kappa(v_1)|$, $y|k,\kappa(v)| = y|k,\kappa(v_1)|$

if $\kappa(v) \neq \kappa(v_1)$, $\kappa \leftarrow \kappa - \kappa(v_1)$

2.4 Robot Motion Planning

The aim of the current work was to generate a roadmap for a given configuration space, such that all homotopic groups are discovered. Much research on using the roadmap for the task of motion planning of a single and multiple robots has already been performed by the author (e.g. [27]). A brief summary to motion planning of a single robot is taken in this section for the sake of completeness.

The roadmap produced is used for the motion planning of a robot. It is assumed that the source (*S*) and goal (*G*) are already known. These are attempted to be connected to the nearest *k* edges of the roadmap to carry out the search. The extra vertex corresponding to the source and goal, and the corresponding edges are temporarily added in the roadmap. The robot is planned using an A* algorithm [28]. The algorithm finds a path from a given source node to a given goal node by maintaining a fringe of nodes to be explored and a set of processed nodes. The node with the least expected cost to goal is picked from the fringe and added to the set of processed nodes, while all non-explored nodes connected to the node being processed are added to the fringe. Initially the fringe is initialized by addition of the source alone. The algorithm relies upon cost functions. The historic cost indicates the cost of getting to the node from the source, which for this problem is the addition of step costs of all elementary steps. An edge corresponding to every set of nodes is given a step cost equal to the Euclidian distance between the nodes. The heuristic cost estimates the shortest distance possible from the node to the goal, which for this problem is taken as the Euclidian distance to goal. The total cost is the sum of historic and heuristic costs.

Post-processing [29] is commonly applied to the outputs of the A* algorithm. Pos-processing mechanisms use a local search to reduce the path length, increase clearance and increase smoothness of the path. Since a local search is utilized, post-processing is not very time consuming and a few iterations can drastically remove the pitfalls of the A* algorithm path. Since the aim of the current work is primarily to produce a good roadmap, post-processing is not applied to the final path to show the raw results using the roadmap alone. A small post-processing can many times result in good paths even for poor roadmaps in scenarios where the final path is rather simplistic.

The chief aim behind the work was to use the homotopic conscious roadmap so generated for the motion planning of multiple mobile robots. The naive methods have already been tried by the author (e.g. [27]). The centralized approach is not realizable for a large number of robots, while a decentralized approach may result in too much congestion around a few areas with the robots not considering the plans of other robots for initial coarser/roadmap level planning. The overall aim is to coordinate the robots in a coarser level planning as was done in [30], the method was specifically designed for a road-like scenario and is not generalizable for open mobile robotic environments. The first step in the direction has been made by capturing all homotopic groups with some redundant connections which do not largely affect the computation time. The next task of using this roadmap for coordination at the coarser/roadmap level planning is a challenge in itself and will be taken in the future research of the author.

3. Results

The algorithm is tested using simulations. The algorithm was tested against a variety of scenarios, only two of those are discussed in detail here to save space. The first scenario is shown in figure 3 and comprises of multiple narrow corridors which are hard to discover and connect, along with some wide open spaces.

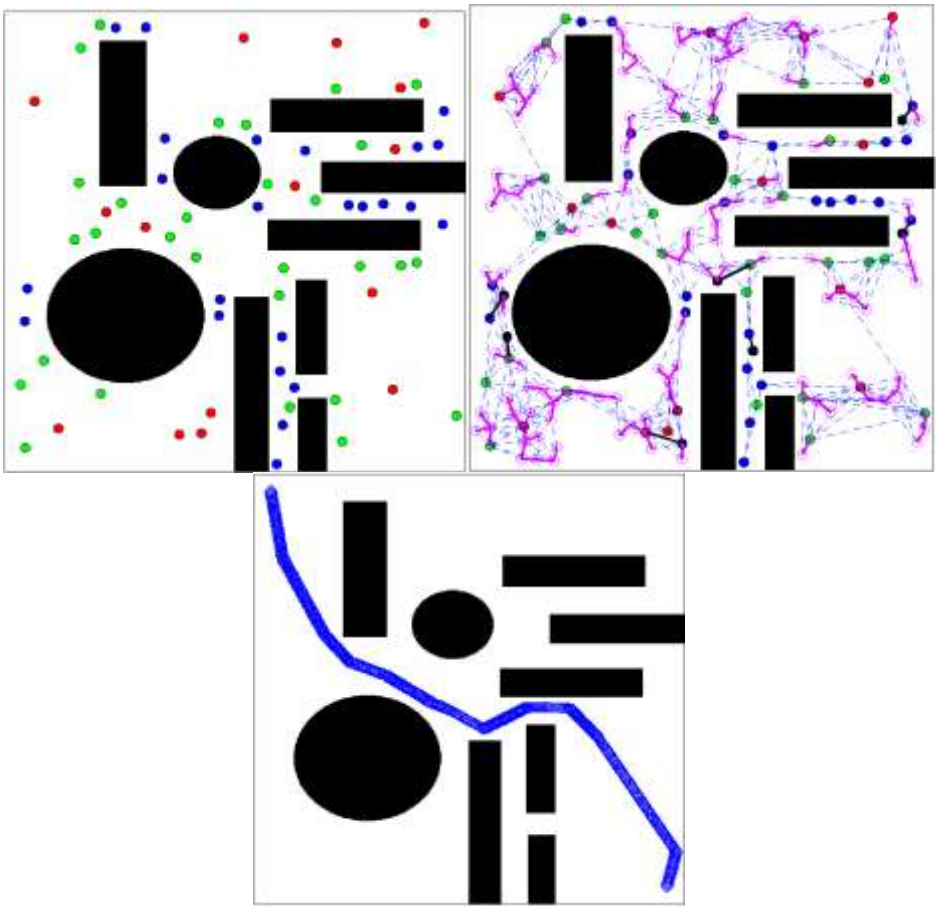

Figure 3. Results for a synthetic scenario (a) Initial Vertices. Blue vertices are from narrow corridor sampling, Green from obstacle sampling and Red from Uniform Sampling. (b) Final Roadmap. Connections to neighbouing k vertices are shown by a dotted line while edges by other strategies are shown by bold solid lines. Black nodes and edges denote additional nodes placed in initial edge connectivity strategy. (c) Path Traced by a robot.

The algorithm was executed assuming a corridor width D of 50 units. The strategies of narrow corridor sampling, obstacle based sampling and uniform sampling were executing for a maximum of 100, 40 and 15 failed attempts or iterations. The distance threshold Δ was fixed as 15 units while the step size δ was fixed to be 15 units. Maximum failed attempts for connecting disjoint vertices

(η) was kept as 50. The main edge connectivity loop was applied for a maximum of 3000 iterations, with the strategies being called with probabilities p_κ=0.4, p_L=0.2, p_ψ=0.2, p_R=0.2.

Figure 3(a) shows the initial vertices generated, which are connected using the edge connection strategy shown in figure 3(b). In both the figures a rectangular robot of a reasonable size is taken and hence the corridors are narrower than they appear. A 3-D configuration space is superimposed on the workspace for viewing. The path traced by the robot is shown in figure 3(c).

Two metrics are taken to assess the performance of the algorithm. These are: success ratio, which is the ratio of the total number of test cases for which the algorithm can find a path, to the total number of test cases given; and path length ratio, which is the ratio of the path length to the Euclidian distance between the source and the goal averaged for all the test cases. The metrics averaged for 100 test cases are presented in figure 4. The figures have been smoothened to a small extent by the use of moving average method. It can be seen that the algorithm quickly converges to a 100% success rate and near-optimal path length along with increase in the number of iterations. However the computational cost also shows an increase. Even for a challenging scenario, it is possible to get good results in small computational time.

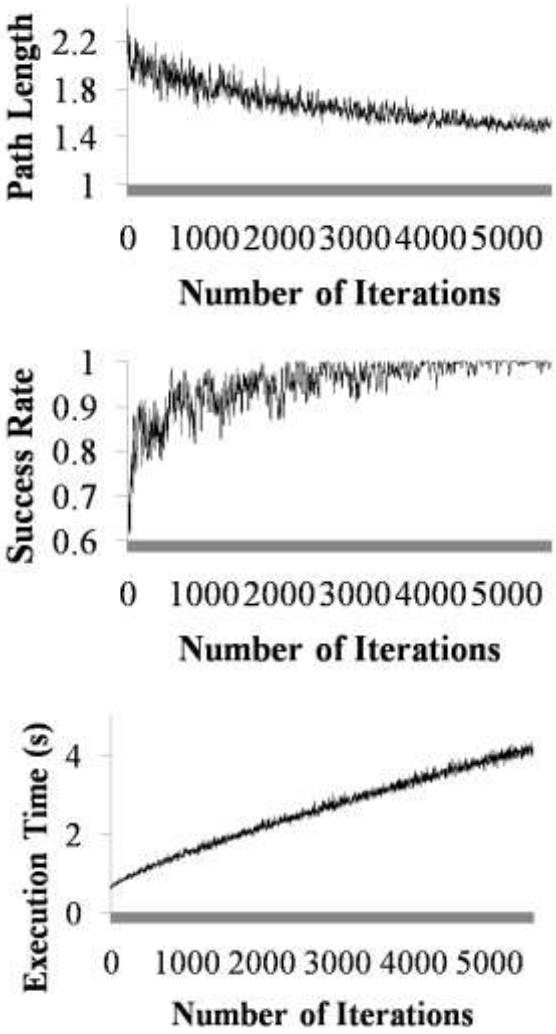

Figure 4. Performance on the scenario (a) Path Length, (b) Success Rate and (c) Execution Time v/s Number of Iterations

The performance of the algorithm is compared with 2 techniques: obstacle based sampling and uniform sampling technique. The only performance criterion common to the 3 algorithms is the computational cost. The 3 algorithms were executed a number of times by increasing the number of iterations (proposed algorithm) or the roadmap size (others). The success rate and path length ratio

obtained for the approaches is plotted in figure 5. It can be seen that the proposed algorithm is significantly better than the other two approaches. Although it initially resulted in poor path lengths, as the focus was primarily narrow corridor discovery, it quickly surpassed the other two algorithms; while also consistently resulting in a higher success ratio.

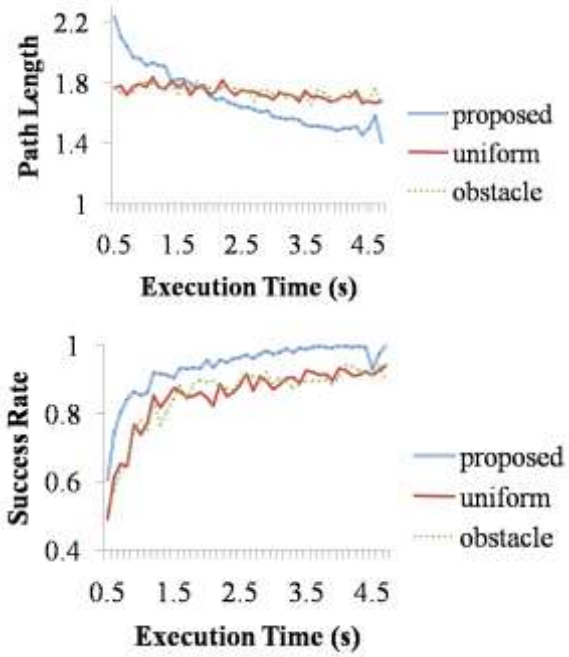

Figure 5. Comparison of the proposed algorithm with uniform and obstacle based sampling methods

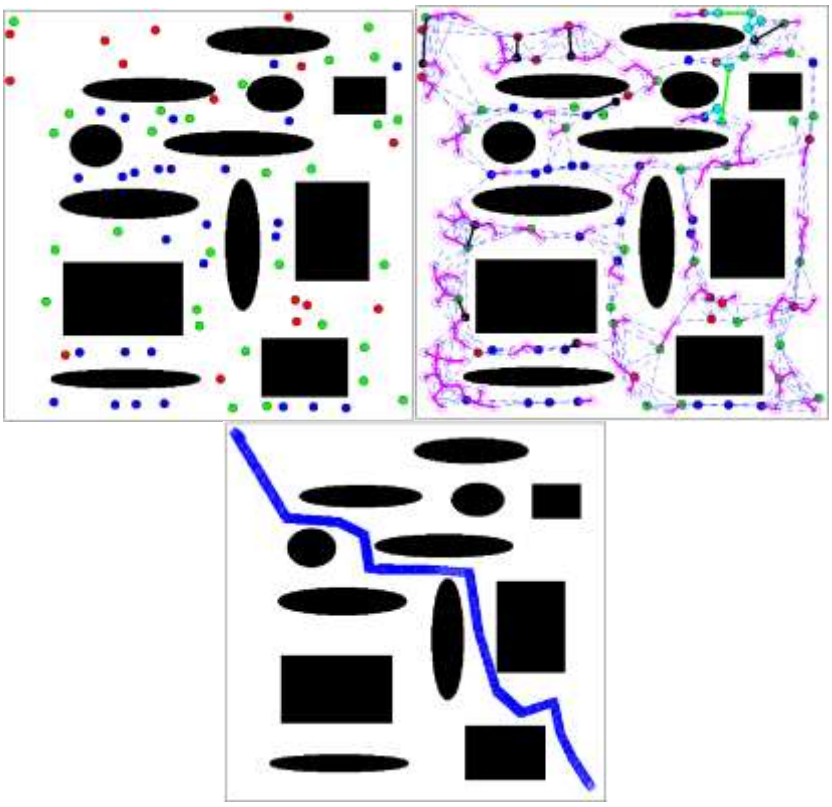

Figure 6. Results for a scenario 2 (a) Initial Vertices. (b) Final Roadmap. (c) Path Traced by a robot.

An additional test case is presented for the problem and its comparisons are made with the same other approaches to make the claims more convincing. The scenario had a few narrow corridors of varying widths and some open spaces in different parts of the roadmap. The second scenario is shown in figure 6. Figure 6(a) shows the initial vertices, figure 6(b) shows the roadmap hence generated and figure 6(c) shows the motion of a robot for some source and goal. It can be seen that the algorithm was again able to produce a roadmap, discovering all narrow corridors and connecting them with the rest of the roadmap.

The performance metrics of success rate, path length and computational time are studied to assess the working of the algorithm. The performance metrics for different iterations are shown in figure 7. It can be seen that the time to construct a roadmap with respectable path lengths and success ratio is still very small. The success rate converges to the maximum value in a few iterations only, while the path length metric shows a reasonable performance by quickly getting closer to the optimal values.

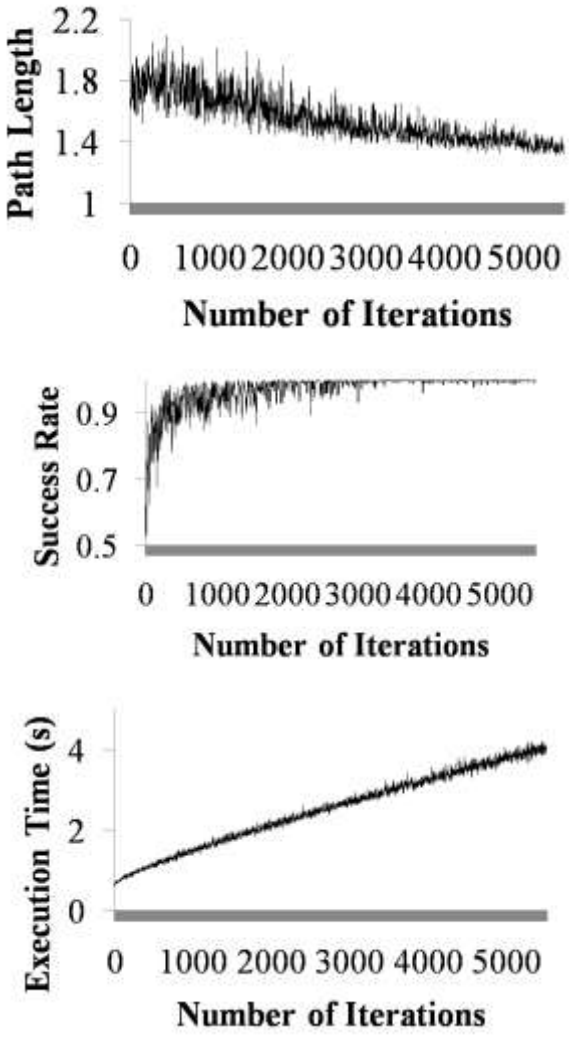

Figure 7. Performance on scenario 2 (a) Susccess Rate, (b) Path Length and (c) Execution Time v/s Number of Iterations

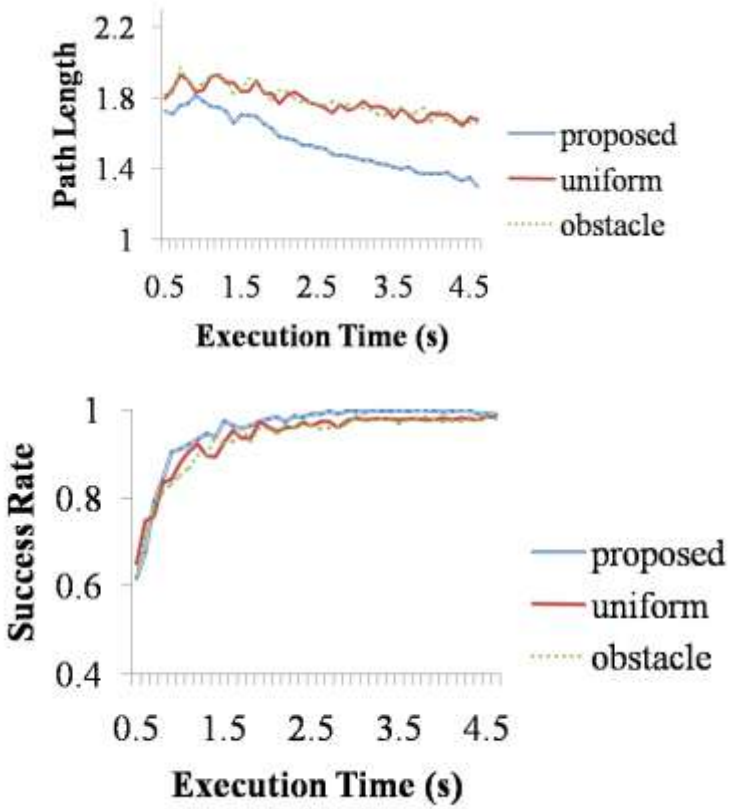

Figure 8. Comparison of the proposed algorithm with uniform and obsatcle based sampling methods for scenario 2.

Comparisons with the other methods are shown in figure 8. It can be seen that even for this test case, the proposed algorithm is significantly better than the other approaches. Since the scenario had enough open spaces and most of the corridors of respectable widths, some path using a longer homotopic path group was always possible. Hence the success rate does not show much trend among the methods. However the fact that all homotopic groups were not discovered by the other methods is clear from the metric of path length. Not discovering a narrow corridor led to the robot taking a longer path length of a different homotopic group.

4. Conclusions

In this paper a new method was proposed to quickly generate good roadmaps which result in discovering all narrow corridors and conveniently connecting them with the rest of the roadmap. A multi-strategized initial vertex generation and edge connection strategy was proposed, which very quickly resulted in a redundantly connected roadmap discovering all narrow corridors. The later iterations, if available, were used for increasing the optimality and adding good redundant edges. The approach was adjudged better than the uniform and obstacle based sampling using the metrics of success rate and path length over computation time. The proposed method hence acts as a mechanism to quickly compute a homotopic conscious roadmap which has vital use in the applications of motion planning for single and multiple mobile robots, and complex mission planning.

The roadmap construction was the first and most important step towards the overall problem. The aim is to use the roadmap to schedule different robots at different parts of the map at different times, much like the routing algorithms in traffic scenarios. The homotopic conscious roadmap gives a basis to formulate coarser level planners and some coordination mechanism for the task. The actual task of planning and coordination will be taken in the future. Further, the task to adapt the roadmap in a dynamic environment, incrementally build the roadmap as the robots move and use the roadmap for multi-robot planning and mission planning tasks needs to be undertaken in the future.

References

[1] Kavraki LE, Latombe JC (1998) Probabilistic Roadmaps for Robot Path Planning. In: Practical Motion Planning in Robotics: Current Approaches and Future Directions, John Wiley, pp. 33-53.
[2] LaValle SM (2006) Planning Algorithms. Cambridge University Press, Cambridge.
[3] Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Rob Res 30(7): 846–894.
[4] Kavraki LE, Svestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Rob Autom 12(4): 566–580.
[5] Bohlin R, Kavraki LE (2000) Path planning using lazy prm. In: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, pp. 521–528.
[6] Amato NM, Bayazit OB, Dale LK, Jones C, Vallejo D (1998) Obprm: An obstacle-based prm for 3d workspaces. In: Robotics: The Algorithmic Perspective, A.K. Peters, pp. 155–168.
[7] Boor V, Overmars MH, van der Stappen AF (1999) The Gaussian sampling strategy for probabilistic roadmap planners. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation vol. 2, IEEE, pp. 1018-1023.
[8] Hsu D, Jiang T, Reif J, Sun Z (2003) The bridge test for sampling narrow passages with probabilistic roadmap planners. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation vol. 3, IEEE, pp. 4420-4426.
[9] Sun Z, Hsu D, Jiang T, Kurniawati H, Reif JH (2005) Narrow passage sampling for probabilistic roadmap planning. IEEE Trans Rob 21(6): 1105-1115.
[10] Aubry M (1995) Homotopy Theory and Models. Birkhäuser, Boston, MA.
[11] Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, Thrun S (2005) Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge, MA.
[12] Nissoux C, Simeon T, Laumond JP (1999) Visibility based probabilistic roadmaps. In: Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems vol. 3, IEEE Press, pp. 1316-1321.
[13] Siméon T, Laumond JP, Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. Adv Rob 14(6): 477-493.
[14] Nieuwenhuisen D, Overmars MH (2004) Useful cycles in probabilistic roadmap graphs. In Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, pp. 446–452.
[15] Marble JD, Bekris KE (2011) Computing spanners of asymptotically optimal probabilistic roadmaps. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 4292-4298.
[16] Kuffner JJ, LaValle SM (2000) RRT-connect: An efficient approach to single-query path planning. In: Proceedings of the IEEE International Conference on Robotics and Automation vol.2, IEEE, pp.995-1001.
[17] LaValle SM, Kuffner JJ (1999) Randomized kinodynamic planning. In: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, pp. 473–479,

[18] Plaku E, Bekris KE, Chen BY, Ladd AM, Kavraki LE (2005) Sampling-Based Roadmap of Trees for Parallel Motion Planning. IEEE Trans Rob 21(4): 597 - 608.
[19] Karaman S, Frazzoli E (2009) Sampling-based motion planning with deterministic μ-calculus specifications. In: IEEE International Conference on Decision and Control, IEEE, pp. 2222-2229.
[20] Kala R (2013) Rapidly-exploring Random Graphs: Motion Planning of Multiple Mobile Robots. Adv Rob 27(14): 1113-1122.
[21] Dobson A, Bekris KE (2014) Sparse roadmap spanners for asymptotically near-optimal motion planning, Int J Rob Res 33: 18-47.
[22] Littlefield Z, Li Y, Bekris KE (2013) Efficient sampling-based motion planning with asymptotic near-optimality guarantees for systems with dynamics. In: Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 1779-1785.
[23] Gayle R, Sud A, Lin MC, Manocha D (2007) Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments. In: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, San Diego, CA, pp. 3777 - 3783.
[24] Gayle R, Sud A, Andersen E, Guy SJ, Lin MC, Manocha D (2009) Interactive Navigation of Heterogeneous Agents Using Adaptive Roadmaps. IEEE Trans Visual Comput Graph 15(1): 34-48.
[25] Quinlan S, Khatib O (1993) Elastic bands: connecting path planning and control. In: Proceedings of the 1993 IEEE International Conference on Robotics and Automation, IEEE, pp. 802 -807.
[26] Hilgert J, Hirsch K, Bertram T, Hiller M (2003) Emergency path planning for autonomous vehicles using elastic band theory. In Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics vol. 2, IEEE, pp. 1390- 1395.
[27] Kala R, Warwick K (2013) Planning Autonomous Vehicles in the Absence of Speed Lanes using an Elastic Strip. IEEE Trans Intell Transp Syst 14(4): 1743-1752.
[28] Nilsson NJ (1971) Problem Solving Methods in Artificial Intelligence. McGraw-Hill, New York.
[29] Kala R (2014) Coordination in Navigation of Multiple Mobile Robots. Cybern Syst 45(1): 1-24.
[30] Kala R, Warwick K (2013) Multi-Level Planning for Semi-Autonomous Vehicles in Traffic Scenarios based on Separation Maximization. J Intell Rob Syst 72(3-4): 559-590.