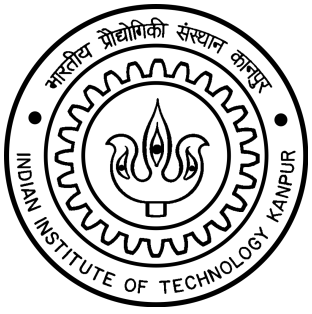
Visual Motion Planning of Multiple Robots by Composing Roadmaps

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Debojyoti Dey



to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

July, 2015

CERTIFICATE

It is certified that the work contained in the thesis titled "**Visual Motion Planning of Multiple Robots by Composing Roadmaps**", by **Debojyoti Dey**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Amitabha Mukerjee

Department of Computer Science & Engineering

Indian Institute of Technology, Kanpur

July, 2015

ABSTRACT

Multi-robot motion planning is a well explored area of research. However, all works till date require us to know the robot type (mobile or articulated), and the parameters that describe the system. Typically sampling approaches (PRM, RRT) are used, where a large number of robot poses are sampled from the C-space. Using these given configuration spaces, traditional methods may obtain a combined C-space as the Cartesian product of each robot C-spaces. In such *centralized* approaches, the time complexity is exponential in the dimensionality of the composite C-space. On the other hand, *decoupled* planning algorithms typically attempt to solve the problem by combining paths planned for the individual robots independently; then combined to give path in the joint space. Despite being highly efficient, these methods suffer from incompleteness due to fixing the path computed beforehand.

In this work, we seek to find efficient solutions that do not require us to know the robot geometry, kinematic structure or even obstacle positions. Image similarity is used to obtain visual manifolds that are homologous to traditional configuration manifolds. Collisions are detected visually and projected onto this visual manifold, and motions can be planned on the underlying graph. We show decoupled planning algorithms that are constructed on a graph obtained by combining the individual neighbourhoods for the individual robots. Our neighbour selection policy guarantees probabilistic completeness in situations where a combined path exists. We also save in terms of space by expanding the parts of the graph by dynamic forward search algorithm and detecting collision on the fly. Thus, this work proposes a radical visual approach to multi-robot motion planning.

Dedicated to my parents

and my teachers

Acknowledgments

I sincerely thank my thesis advisor Prof. Amitabha Mukerjee for his constant support and guidance. I thank him for being a wonderful mentor to always inspire and guide me to the right path.

In addition I would like to thank all the faculty members of CSE department not only for their excellent teaching but also for the constant efforts they put to keep us improving both academically and morally. I thank the institute as a whole for providing us such a beautiful environment. And I thank all of my classmates for their friendship and the happy moments we shared together.

Last, but not the least, I would like to thank my parents, my younger sister for their love, constant support and encouragement. Without their support and encouragement this work would not have been possible.

Debojyoti Dey

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of the thesis is to plan paths for multiple robots. Traditionally, the problem has been handled using a configuration space defined in terms of motion parameters of the robots. An example of such motion parameters are joint angles of an articulated arm. The problem of finding a collision-free path in configuration space is known to be PSPACE-hard [Hopcroft et al. (1984)] in general.

"*A free path in a configuration space of any fixed dimension m, when the free space is a set defined by n polynomial constraints of maximal degree d, can be computed by an algorithm whose time complexity is exponential in m and polynomial in both n(geometric complexity) and d(algebric complexity)*".[Schwartz and Sharir (1983)]

Above statement gives an upper bound. The most efficient algorithm to date is due to Canny [Canny (1988)] who proposed a roadmap method that is singly-exponential in m. In practice, the most effective algorithms for motion planning are probabilistic where the complexity is polynomial in the number of samples N. In this work we propose a novel approach based on visual input alone. Such a model works on a random sample set of images of the robots without knowing the motion parameters concerned. The approach is explained next.

1.1 Random sampling and completeness

We work with a set of random image samples $X = \{x_1, x_2, \ldots, x_n\}$ drawn from the space of robot images. Before moving further, we like to define the notion of *Resolution Completeness* of a motion planner. Completeness requires the motion planner to either produce the solution to a path query or to correctly report that there is none within reasonably finite time limit. Resolution Completeness is defined for a sample based planning algorithm. Such a motion planner is defined to be resolution complete if the planner is complete when the samples are dense enough. However, randomized planning algorithms suffer from a major drawback. If the input path planning problem admits no solution, the planner has no way to recognize it even after a large amount of computation. Hence, a limit on the running time has to be imposed. But, if this limit is attained and no path has been generated, there is no guarantee than no path exist. Experiments show that there is a little chance that one actually exists. It is shown [Latombe (2012)] that randomized planning algorithms are *Probabilistically Resolution-Complete*.

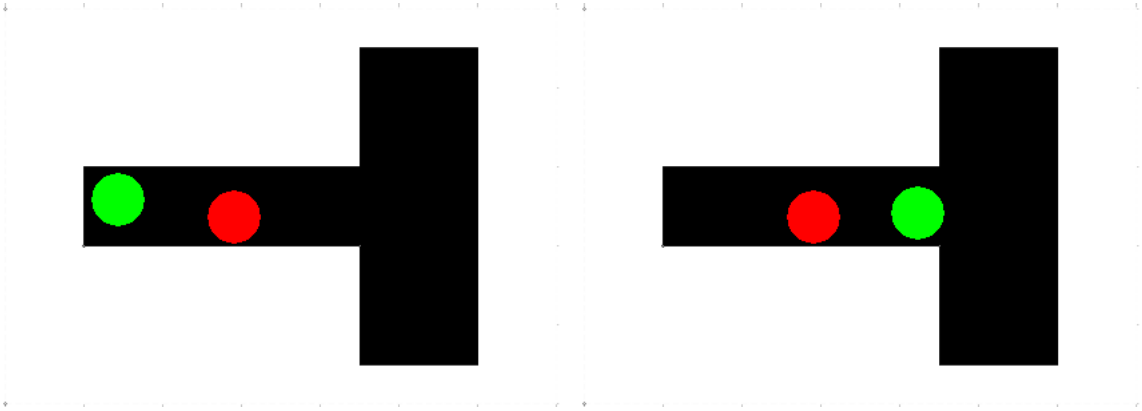
Here we are interested in multi-robot motion planning where there are a number of moving mobile robots but obstacles are static. Multi-robot motion planning algorithms can be broadly classified into 1) Centralized path planning and 2) Decoupled planning. The next sections will give an idea and brief overview of the research works on the two techniques and present a comparative study of resolution completeness.

1.2 Centralized Path Planning

The main idea of centralized planning algorithm is to treat multiple robots $\mathcal{A}_1, \ldots, \mathcal{A}_p$ operating in a same workspace $\mathcal{W}$ as a single multi-bodied robot $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_p\}$. The composite configuration space of $\mathcal{A}$ is $\mathcal{C} = \mathcal{C}_1 \times \ldots \mathcal{C}_p$, i.e.the Cartesian product of the configuration spaces $\mathcal{C}_i$ of individual robots $\mathcal{A}_1, \ldots, \mathcal{A}_p$. Every configuration in $\mathcal{C}$ determines a unique position and orientation for each robot

$\mathcal{A}_i$. $\mathcal{C}^{obs}$ in this space result from the interaction between a robot and the static obstacles or between two robots. We define $\mathcal{C}^{free} = \mathcal{C} - \mathcal{C}^{obs}$. Basic path planning methods can be used to plan a path of $\mathcal{A}$ in $\mathcal{C}^{free}$.

Probabilistic Roadmaps [Kavraki et al. (1996)] is one of the motion planning methods based on random sampling used for holonomic robots moving in a static workspace. The method proceeds in two phases: a learning phase and a query phase. In learning phase the probabilistic roadmap is constructed by iteratively choosing a random point in C-space and connecting it to the nearby points already chosen. So the roadmap is a graph where the nodes correspond to collision free configurations and edges correspond to feasible path between these configurations. It uses a local planner to ensure that a path between nearby configurations is actually collision free i.e. to check edge feasibility. In the subsequent query phase, multiple queries can be answered very fast as in other sampling methods. Being a randomized planning algorithm, PRM is probabilistically resolution-complete. A Rapidly exploring Random Tree [LaValle (1998)] is another method used by motion planner which search non convex high dimensional spaces by randomly building a space-filling tree. An RRT incrementally grows a tree rooted at the starting configuration by using random samples from search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is collision free, the new state is added to the tree, thereby reducing the expected distance of a randomly chosen point to the tree. In this way RRT finds a path to the goal state when the goal node is connected to the tree. A major difficulty in the centralized method is that it can yield high dimensional configuration space. The dimension of the composite configuration space $\mathcal{C}$ is $dim(\mathcal{C}) = \sum_{i=1}^{p} dim(\mathcal{C}_i)$. As discussed earlier, the complexity of the path planning algorithm grows exponentially with the combined dimensionality of the configuration space concerned. Thus it rapidly becomes intractable for planning multi robot tasks. In the following decoupled approaches, one attempts to compute $\mathcal{C}$-spaces separately for each robot and thus keeping the complexity controlled by $\max_i dim(\mathcal{C}_i)$.



(a) Start state (b) Goal state

Figure 1.1: If the red circular mobile robot neglects the query for the green robot the completeness is lost in prioritized planning approach. This example has a solution in general, but prioritized planning fails to find it.Image taken from [Latombe (2012)]

1.3 Decoupled Path Planning

The second approach and our our main emphasis in the current discourse will be Decoupled motion planning for multiple robots sharing a common workspace. Instead of considering all the robots as a single multi-bodied robot, it decouples each individual robot and computes their interactions separately. The robots are considered loosely bound i.e. each of the robots is able to perform action without any dependence on others. Initially a path is planned for each of the component robots individually and independent of others. The interactions among the paths are considered next in the second phase of planning. By doing so, the computation complexity may be significantly reduced. Next, we consider four variants of decoupled planning.

1.3.1 Prioritized planning

In this method the robots are sorted by priority and planning is done for higher priority robots first. Lower priority robots plan by viewing the higher priority robots as moving obstacles. Unless priority is assigned properly, the completeness is lost. Fig. 1.1 gives an example [LaValle (2006)] of a failure case.

Bennewitz,Burgard & Thrun [Bennewitz et al. (2001)] use $A^\star$ algorithm to find

path for individual robots and then perform *randomized search with hill climbing* for assigning priority to the robots. This results in reduction in the number of failures as well as the overall path length.

1.3.2 Fixed-path coordination

Suppose that each robot $\mathcal{A}_i$ is constrained to follow a path $\tau_i : [0, 1] \mapsto \mathcal{C}_i^{free}$, which can be computed using any ordinary motion planning technique. For n robots, an n-dimensional state space called a coordination space is defined that schedules the motions of the robots along their paths so that they will not collide. One important feature is that time will only be implicitly represented in the coordination space. An algorithm must compute a path in the coordination space, from which explicit timings can be easily extracted.

Trajectory coordination model [O'Donnell et al. (1989)] plans deadlock and collision free coordination of the multiple-robots. They incorporate the time history for a path planned individually for a robot by dividing the path into local segments having length proportional to the execution time. Then a *task-completion diagram* is constructed which is basically a grid or matrix formation representing the path segments of two robots on vertical and horizontal axes. If path segments A_i and B_j collides, then the R_{ij} block of the matrix R is blackened to mark dead zone. The optimal solution is given by the *greedy scheduling* to reach from bottom left corner to top right corner point of the matrix while keeping away from the dead zones. The corner points stand for the initial and goal configuration states respectively in the joint configuration space. Siméon et al. (2002) propose a method which searches the *generalized coordination diagram*(GCD) constructed from the precomputed paths for n robots. The collision zones are marked in GCD and the objective becomes to find an optimal path in it without intersecting any such zone. Both of the preceding methods can be grouped under the term *Velocity Tuning*. Though the method is highly efficient it has been proved [Sanchez and Latombe (2002)] to be *incomplete* as the failure rate is very high. As the individual paths are planned beforehand, it

fails to find an alternate path in coordination space in case of collision even if one exists.

1.3.3 Semi centralized model

To address the incompleteness issue of decoupled path planning strategies, a more recent trend has been to find an intermediate solution between centralized path planning performed in the configuration space and decoupled planning performed in coordination space. These methods try to reduce failure by using a hybrid of coordination and configuration space while maintaining the efficiency of decoupled planning.

Incremental Coordination model [Saha and Isto (2006)] for planning motion for n robots iteratively searches path for i^{th} robot in the $(\mathbb{P}^{i-1} \times \mathcal{C}_i)$ space where $\mathbb{P}^{i-1}$ gives the coordination space for the $(i-1)$ robots for which paths have already been planned and $\mathcal{C}_i$ gives d_i dimensional C-space of the i^{th} robot. Thus the dimensionality of the composite space is at most $(n+d_i-1)$ which is a huge savings when compared to $\sum_{i=1}^{n} d_i$ for a full space planning in PRM.

A modified version of centralized planning [van Den Berg et al. (2009)] decouples optimally the path planning problem of multi-robot system into number of sub problems. Each sub-problem is to do *centralized* path planning of a composite subsystem. Then the *sub problems* have to be *executed sequentially* in the order derived by the algorithm proposed. The objective function is to *minimize the maximum degree* of all composite robots. The optimization is performed by creating a collection of *constraint graphs* $G(J)$ from the necessary execution sequence of the robots expressed as logical expression. Each $G(J)$ gives a component directed acyclic graph $G^{SCC}(J)$ where each node is a strongly connected component. An SCC represents one composite subsystem mentioned. One of the component DAG has to be chosen to satisfy our objective function.

In a recent model of subdimensional expansion [Wagner and Choset (2015)] for a multi-robot environment, it starts with planning path for the participating robots

independently by $A^\star$ or RRT algorithm in their respective C-spaces. Then these paths are combined. The combination defines a one-dimensional search space embedded in the joint configuration space. Each robot is expected to move on their own path. When robots are found to collide in the search space, subdimensional expansion locally grows the search space to allow the robots involved to find an alternative path in their joint configuration space by coupled planning. This information is back propagated to all the points leading to the collision. After that the planning restarts keeping the non-colliding robots follow their initial paths. The approach is complete and can handle special cases.

However, in worst case scenario, both the incremental and subdimensional expansion models combines all the individual C-spaces, planning path in which becomes intractable.

1.3.4 Fixed-roadmap coordination

The fixed-path coordination approach may not solve the problem of incompleteness as the paths are computed beforehand. Fortunately, fixed-path coordination can be extended to enable each robot to move over a roadmap or topological graph. This still yields a coordination space that has only one dimension per robot, and the resulting planning methods are much closer to being complete, assuming each robot utilizes a roadmap that has many alternative paths. Gharbi et al. (2009) demonstrates composition of roadmap for multi-arm system in $\mathcal{C}$-space. The thesis work we are going to present is largely inspired by fixed-roadmap coordination model. For a m robot system, each with n poses, it computes the neighbourhood matrix for each robot individually generating its own roadmap. Then it combines the individually created neighbourhood matrices to compose the roadmap of the joint system by constructing a $(n^m \times k)$ neighbourhood matrix. We do roadmap composition in visual Configuration space.

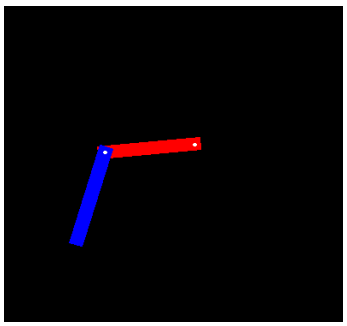
1.4 Cognitive Models of Visuo-motor space

Human beings do not appear to use global coordinates; instead, motion is enccoded (both consciously and implicitly) in terms of relative poses. Mammals navigating in familiar environments acquire "place cells" which indicate their position in space, but these are organized in a system of neighbouring columns in the brain that encode nearby places. Thus, the representation appears to encode space in terms of nearby positions rather than a global coordinate.

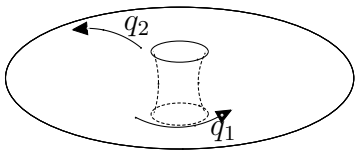
For articulated arm motion, the body appears to maintain a dense model of peripersonal space [Rizzolatti et al. (1996)]. "Objects within peripersonal space can be grasped and manipulated; objects located beyond this space (in what is often termed extrapersonal space) cannot normally be reached"[Holmes and Spence (2004)]. Visual representations in peripersonal space are tightly coupled with motor representations. Thus, certain neurons in the ventral pre-motor cortex respond to visual presentation of objects in the peripersonal space, but stop responding if they are moved out of the space. Again, the same motor neurons that command the right arm in monkeys, also respond to visual stimuli in the graspable positions when the right arm is kept hidden [Graziano (1999),Holmes and Spence (2004)]. All these cues indicate that there is a strong visual aspect to the encoding and modeling of the reachable space.

1.4.1 The visual manifold hypothesis

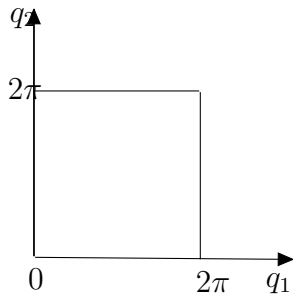
The work here may also have ramifications in the cognitive modeling of visuo-motor space. One possible manner in which the various areas of the brain that encode space may be organized is by maintaining self-organizing structures wherein nearby sensory-motor areas are located nearby and interact with each other. Our model suggests that "nearby" models of a body or an arm in space can be encoded is by creating local tangent spaces i.e. neighbourhoods on the image manifold, by encoding these neighbourhoods as a linear map. We suggest that this is being done



(a) Two link planar manipulator



(b) C-space manifold



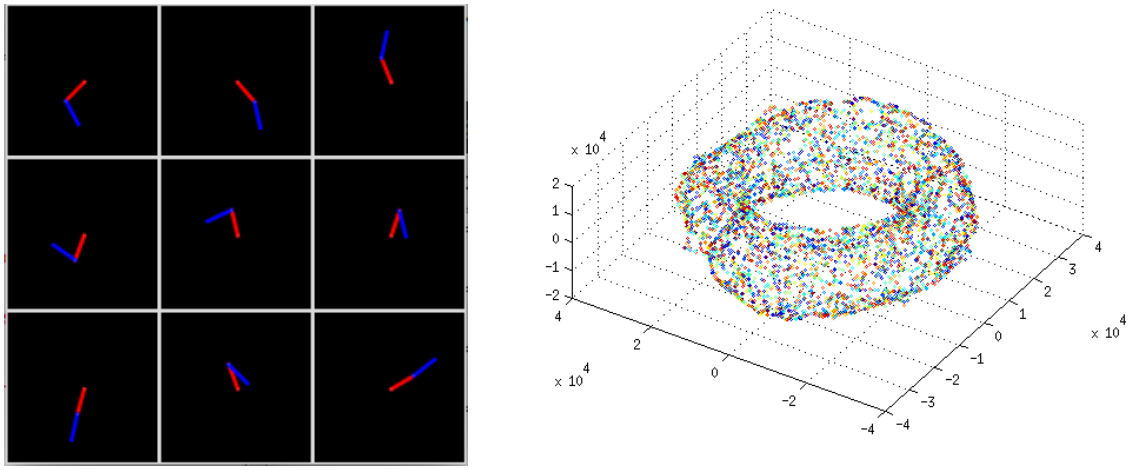
(c) Homeomorphic generator space

Figure 1.2: The configuration space of a planer two-revolute-joint manipulator arm is a torus in $\mathbb{R}^3$. A parametrization of this space is obtained by representing every configuration as $(q_1, q_2) \in [0, 2\pi] \times [0, 2\pi]$ where q_1 and q_2 are angles associated with first and second joint angle.The parametrization corresponds to cutting the torus along two generators. Opposite edges of the square shown in (b) must be procedurally identified through modulo 2π arithmatic in order to capture the multiple connectedness of the torus [Latombe (2012)].

by mapping the local spaces in terms of a Principal Component Analysis, which can be encoded by individual neurons [Oja, 1982]. The set of these local tangent spaces, encoded as a body of neurons, then represent the visual manifolds proposed here. The coordinate system on these visual manifolds are *generalized coordinates* - they uniquely represent robot poses, so they can also be used as configurations.

However, instead of actually solving for these global coordinates, we show that simply encoding the local neighbourhood for each view creates a small space (within which one may retract or add similar views).

One way to solve path planning problem is to discover this intrinsic topological subspace of image space which we call the visual manifold and then planning a path between two points on the manifold. Building a tangent space Atlas [Ramaiah et al. (2013)] head motion can be animated by motion planning on the manifold intrinsic to gaze space. There are several methods of dimensionality reduction that helps in discovering low dimensional manifold embedded in high dimensional space which in our case is necessarily $(w \times h)$ image vectorized. PCA is the most basic dimensionality reduction technique which discovers the low dimensional embedding when intrinsic space is linear. Multi dimensional scaling[Kruskal and Wish (1978)] seeks to obtain a mapping that preserves the relative inter-point proximity(similarity).



(a) Sample images of 2D arm

(b) Torus manifold

Figure 1.3: Visual manifold generated by Isomap

Isomap [De Silva and Tenenbaum (2004)] is a special application of MDS where the similarity is given by geodesic distance between two points.

For multiple robots, we show that the generalized coordinates of the composed space can be obtained as the cartesian product of visual manifolds each defined in terms of the local neighbourhoods. For representing a pose with multiple robots in different poses, we find the cartesian product of their individual neighbourhoods; this model creates a map for the local tangent space in the composite configuration manifold - a much higher dimensional space than each. However, the added computation is limited and the algorithm is dominated by the maximum of the dimensionalities of the input spaces. In addition to proposing this model as a solution to the long standing problem of a visuo-spatial map, we also extend this to make it useful for robots. The visual manifold is homeomorphic to the canonical motor manifold.

Thus, the approach shown here may also be relevant to cognitive tasks such as bimanual coordination, coordination between walking and the upper limbs, or any other coordinated motion between multiple robots.

1.5 Contribution of thesis

To address the incompleteness of fixed-path coordination model, our approach is similar to the fixed-roadmap coordination model. The main difference of our method from that one is that our algorithm is implemented on a visual configuration space as opposed to motion configuration space. We claim our method to be probabilistically resolution-complete and near optimal. The key traits of our method are:

- For greater efficiency, the motion planning we perform is *decoupled*: we plan road maps individually for all the robots. We combine them to create road map for composite system.

- We are oblivious to the configuration parameters of the robots e.g. degrees of freedom, D-H parameters, shape, geometry etc. We use random snapshots of the individual robots while moving in the workspace. Initially the space will have only the permanent(part of environment) obstacles; other static obstacles may be introduced later into the workspace.

- Being oblivious to the configuration parameters(or working mechanism) of the robots, out method is generic and thus usable by various kinds of robots such as planar mobile robots or manipulators.

- Instead of pre-computing and fixing paths for participating robots as in fixed path coordination model, we combine precomputed road maps built for individual spaces. By the virtue of the roadmap composition strategy used by us, we show our method to be probabilistically resolution-complete.

- We define a measure for optimality in terms of the ratio between the maximum time taken by an individual robot $\mathcal{A}_i \in \mathcal{A}$ in decoupled space to reach its own goal and the total time taken by $\mathcal{A}$ to reach collective goal.

- The approach is also less expensive in terms of space complexity. We do not take Cartesian product of individual roadmaps blindly. Instead, we *dynami-*

cally compute the joint tangent space for a point on the composed roadmap by taking product of its component tangent spaces in the form of *neighbourhood product lattice*. Tangent space for a point on visual C-space is given by its local neighbourhood. We selectively pick some entries from the product lattice as representative of local neighbours of the point in composite space. The screening along with dynamic expansion of a point to find its neighbourhood radically reduces the average space complexity. Moreover, the screening plays a pivotal role to ensure probabilistic resolution-completeness.

1.6 Organization of chapters

The next chapter or chapter 2 is the crux of the thesis where we shall develop roadmap composition technique in multi robot environment. We discuss several strategies of neighbourhood construction in composite space by combining the individual neighbourhoods. We describe static and dynamic approaches of roadmap composition. Besides, we shall have a thorough discussion on local planning via collision detection. We present several metrics to analyse the performance. This is followed by some experiments with two kinds of robot: planar mobile robots and articulated robotic arms. In chapter 3 we discuss an alternative method where we find the visual manifold by Isomap and try to plan path on it. To handle quadratic space complexity for two robot system, we implement Landmark-Isomap. The paths computed appear to be unauthentic and highly suboptimal in comparison to our previous method. We tried to improve the performance by modifying the original proximity matrix which is fed to Isomap. We tried to adapt distance metric learning, which is mostly used for supervised learning to the unsupervised system we work with. Even this fails to produce any improvement. The last chapter concludes our work by making an overall observation and comparative study of different approaches discussed.

Chapter 2

Multi-robot Path Planning by Composing Visual Roadmaps

The aim of this chapter is to plan paths for multiple planar robots(mobile or articulated) $\mathcal{A}_1, \ldots, \mathcal{A}_p$ operating in a same workspace $\mathcal{W}$. One may consider the set of robots as a single entity. In a composite system, represented by $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_p\}$, the objective is to move the robots from a composite start configuration to a goal configuration without hitting static obstacles or one another. We call the two types of collision as *static* and *dynamic* collision respectively.

As discussed in the in introduction chapter, motion planning of such systems can be *centralized* or *decoupled*. The key features of our approach are as follows:

- For greater efficiency, the motion planning we perform is *decoupled*: we plan road maps individually for all the robots. We combine them to create road map for composite system.

- We are oblivious to the configuration parameters of the robots e.g. degrees of freedom, D-H parameters, shape, geometry etc. We use random snapshots of the individual robots while moving in the workspace. Initially the space will have only the permanent(part of environment) obstacles; other static obstacles may be introduced later into the workspace.

- Being oblivious to the configuration parameters(or working mechanism) of the

robots, out method is generic and thus usable by various kinds of robots such as planar mobile robots or manipulators.

- Instead of pre-computing and fixing paths for participating robots as in fixed path coordination model, we combine precomputed road maps built for individual spaces. By the virtue of the roadmap composition strategy used by us, we show our method to be probabilistically resolution-complete.

- We define a measure for optimality in terms of the ratio between the maximum time taken by an individual robot $\mathcal{A}_i \in \mathcal{A}$ in decoupled space to reach its own goal and the total time taken by $\mathcal{A}$ to reach collective goal.

- The approach is also less expensive in terms of space complexity. We do not take Cartesian product of individual roadmaps blindly. Instead, we *dynamically* compute the joint tangent space for a point on the composed roadmap by taking product of its component tangent spaces in the form of *neighbourhood product lattice*. Tangent space for a point on visual $\mathcal{C}$-space is given by its local neighbourhood. We selectively pick some entries from the product lattice as representative of local neighbours of the point in composite space. The screening along with dynamic expansion of a point to find its neighbourhood radically reduces the average space complexity. Moreover, the screening plays a pivotal role to ensure probabilistic resolution-completeness.

Our approach bears partial resemblance with fixed roadmap decoupled planning(section 1.3.4) method already discussed. Next, we discuss about the distance metric used to measure inter-pose proximity followed by the details of visual composition of roadmap.

2.1 Tracks and the Notion of Distance

Working in visual space, we are not privy to the motion parameters of the robots concerned. We have sampled set of random pose images for each robot participating

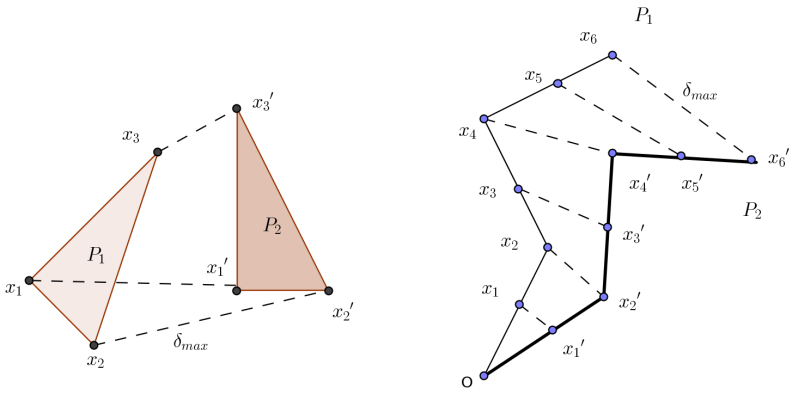


Figure 2.1: Demonstration of track-based distance of mobile and articulated robot

in the composite system. We need to measure proximity between any two poses. The distance metric used aims at measuring the area swept by the robot while going from one pose to another. This can be approximated in terms of the maximum distance covered by any point[Kavraki et al. (1996)] on the robot. To measure this approximately we take a set T of distinguished points on the robot. We call these points *Track by Feature* points or **TF-points**. The connecting straight line segment between the initial and final position of a TF-point is called a track. The distance between two poses P_1 and P_2 is given by:

$$D(P_1, P_2) = \delta_{max} = \max_{x_i \in T} ||x_i(P_1) - x_i(P_2)|| \tag{2.1}$$

This is treated as a distance between two poses in visual $\mathcal{C}$-space. Here we assume that the location of a TF-point in pose P_1 and P_2 can be tracked. In this work, we consider 2D robots only and an overhead camera so that all of the TF-points are always visible. For 3D robots, we can take a subset of tracks based on visibility of the feature points. In general, this is a correspondence problem in vision and local feature descriptors such as SIFT or SURF may be used. Figure 2.1 demonstrates the idea.

2.2 Visual Roadmap composition

A roadmap is a graph which corresponds to the adjacency list where each node stands for a visual configuration given by an image of a robot. The adjacency list is obtained by taking k nearest neighbours of a node in image space. We maintain a weight matrix corresponding to an adjacency list containing the distance between all the nodes and their adjacent nodes. The distance metric coined in the last section enables us to build road-maps without having any knowledge of the motion parameters of the robots. Instead of searching in the coordination space after computing the individual paths, we i) compute the road-maps for the individual robots in their visual configuration space and ii) use them to compose the roadmap in joint space.

The initial images and corresponding visual graph assume that all environmental objects are part of permanent set-up. 'Obstacles' are defined as objects that may be added later to the workspace environment. Subsequently, when obstacles are detected by image mismatch at known poses, in the graph formed of the adjacency list, all nodes colliding with obstacles are marked as invalid and removed. We call this kind of collision a *static collision*. After removal of invalid nodes, the following conditions have to hold for the graphs:

- If the symmetry is not asserted in neighbouring relation, the graph becomes directed. Then we need to ensure that the graph consists of a single strongly connected component.

- If we maintain symmetry in the neighbouring relation, meaning each node is also a neighbour to its neighbour, the graph becomes undirected, then we need to ensure that the graph is connected.

The above conditions are to ascertain that a robot can move from any node to any other node in the graph concerned. Thus the graph serves as the ideal roadmap. For multiple robots, we combine the individual adjacency lists to form adjacency list in the combined space and thus constructing roadmap in the composite space.

This is done by computing the neighbourhood product lattice for all points in the combined system and designating k entries of the lattice as nearest neighbour of the point. We study a number of policies to compute composite neighbourhood matrix for a m-robot system.

2.2.1 Computation of Composite Neighbourhood

We compute product matrix for a two robot system in visual configuration space. Each point in the space is represented by $(x^1, x^2, \ldots, x^m)$ where $x^i \in v\mathcal{C}_i$ is a point in the visual configuration space $v\mathcal{C}_i$ for the robot $\mathcal{A}_i$. If $\forall i \in [m]$, x^i has k_i valid neighbours in its visual $\mathcal{C}$-space. Let $x^i_0, x^i_1, \ldots, x^i_{k_i-1}$ be the neighbours of x^i_0. For m robots the neighbourhood product lattice is represented by the m dimensional tensor M such that an entry at the index $(j_1, j_2, \ldots, j_m)$ is computed as $\sqrt{(d^1_{j_1})^2 + (d^2_{j_2})^2 + \ldots + (d^m_{j_m})^2}$ where d^i is the distance vector of length k_i corresponding to nearest neighbours of x^i_0. This value gives distance between the original state $(x^1_0, x^2_0, \ldots, x^m_0)$ and the state $(x^1_{j_1}, x^2_{j_2}, \ldots, x^m_{j_m})$ in the composite visual $\mathcal{C}$-space.

Our next job is to identify neighbours of the point $(x^1_0, x^2_0, \ldots, x^m_0)$ from the neighbourhood product lattice. We present several possible choices of neighbourhood size k.

1. keeping the value of $k = \min(k_1, k_2, \ldots, k_m)$. This ensures that k nearest neighbours can be found within the $(k_1 \times k_2, \ldots \times k_m)$ composed set.

2. preserving the zero motion nodes i.e. retaining all the elements at index $(j_1, j_2, \ldots, j_m)$ where $m - 1$ elements in the index are zero and only one is non-zero from M along with not greater than $\min(k_1, k_2, \ldots, k_m)$ smallest entries from the remaining elements of M, thus giving $k \leq k_1 + k_2 + \ldots + k_m + \min(k_1, k_2, \ldots, k_m)$. This ensures that $\forall i$ all paths found in $v\mathcal{C}_i$ remain feasible in the new neighbourhood graph.

3. keeping the entire matrix M, i.e. all the $(k_1 \times k_2, \ldots \times k_m)$ entries.

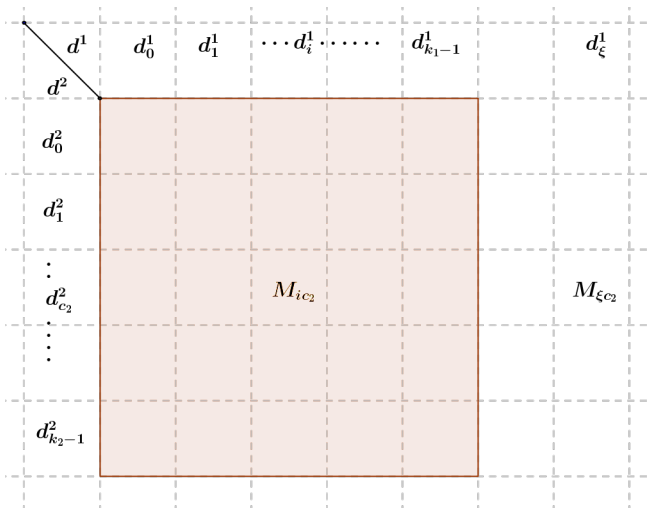


Figure 2.2: Neighbourhood product lattice for two-robot system

In all the policies, after choosing k neighbours from the Cartesian product space, we shall discard the ones that correspond to inter-robot(dynamic) collision. We shall prove that by taking $k = \min(k_1, k_2, \ldots, k_m)$ [approach 1] we have the same neighbourhood that could have been obtained in a centralized $v\mathcal{C}$-space. We present the proof in the following:

Lemma 2.2.1. *k nearest neighbours of the point $(x_0^1, x_0^2, \ldots, x_0^m)$ in the composite space lie within the $k_1 \times k_2 \times \ldots \times k_m$ neighbourhood lattice in the composite space if $k \leq \min(k_1, k_2, \ldots, k_m)$.*

Proof. For the sake of contradiction, lets assume that there is an element within the first k neighbours but lies outside the $k_1 \times k_2 \times \ldots \times k_m$ lattice (Fig.2.2 for two-robots). We assign an index $(j_1, j_2, \ldots, j_m)$ to the element in the extended neighbourhood lattice. It means $\exists i \in [m], j_i = \xi > k_i$. As for any i the distance vector d_i is sorted, each of the rows and columns of M are themselves sorted in ascending order. Without loss of generality we can assume the sorting to be strict. So, $M(j_1 = c_1, j_2 = c_2, \ldots, j_i \leq \xi, \ldots, j_m = c_m) < M(c_1, c_2, \ldots, \xi, \ldots, c_m)$, where every c_i is a constant. That means we already have at least k_i elements in M closer to $(x_0^1, x_0^2, \ldots, x_0^m)$ than the our subject element and thus candidates for k nearest neighbours. But by the LHS of our implication $k < k_i, \forall i$ which contradicts the fact. So our initial assumption about non-containment is wrong. So $k < k_i, \forall i$ implies

that the subject element has to map to an entry in M. Thus $k \leq \min(k_1, k_2, \ldots, k_m)$ is a sufficient condition. Hence *proved*! □

While it is desirable that the composite neighbourhood be the same as that of the v$\mathcal{C}$-space, in practice it means that some neighbours in the original space $v\mathcal{C}_1, v\mathcal{C}_2, \ldots, v\mathcal{C}_m$ are now lost since not all neighbours of x_0^i remains in the reduced space. Even for a greater value of k inclusion of all the original edges is not likely. Since we cannot guarantee connectedness of the resulting composite roadmap, the first policy of choosing k will be incomplete.

One approach of preserving all paths in the original space without inflating the neighbourhood(and hence the graph size) too much, is to retain all the original neighbourhoods, and add the k_{min} new composite neighbours found by closest proximity. This is the second policy noted above which keeps all the edges connecting each robot to all its neighbours. This preserves all the axial elements of the neighbourhood product lattice M. This ensures that any robot can move to all its neighbours while keeping the others static. The resulting path may take a little longer than the optimal. At least it will be found. From the remaining elements of the matrix, we choose $k_{min} = \min(k_1, k_2, \ldots, k_m)$ smallest entries. So, we consider $k = k_1 + k_2 + \ldots + k_m + k_{min}$ composite states as neighbours of $(x_0^1, x_0^2, \ldots, x_0^m)$. This also saves in terms of space as we are not taking all the elements in the product lattice.

The third policy keeps all the $k_1 \times k_2 \times \ldots \times k_m$ elements of M as neighbours. Though it can assure probabilistic completeness by keeping all the edges of the component roadmaps, it incurs high space complexity to store such composite neighbourhood matrix. It may invite redundancy by keeping unnecessary edges.

Supposing the robot $\mathcal{A}_i$ has $\mathcal{N}_i$ samples, one way of maintaining the composite roadmap is to compute offline and store the $\mathcal{N}_1 \times \mathcal{N}_2 \times \ldots \times \mathcal{N}_m \times k$ composite neighbourhood matrix with rows representing the neighbourhood of every point in the composite space. This matrix can be used later to answer multiple queries. Storing this matrix may be impossible when number of robots m is high. So we

compute it *dynamically* by expanding a composite node only when it is explored by *Dijkstra* for shortest path computation in composite space.

Among the three policies discussed to choose k nearest neighbours in the composite space, the second policy is explored more since it provides a good balance between space complexity and probabilistic resolution-completeness. An important assumption in our experiments is that the workspace is showed for all the robots, and that the camera can view the entire workspace. However, during sampling only a single robot is seen at one time. This is impractical for articulated arms. But we can approximate this situation by assuming that the other arm(s) are in poses that are maximally apart. We consider two approaches for combining the individual roadmaps:

a) Complete neighbourhood construction for composite space

b) Dynamically constructing the roadmap by incremental forward search algorithm.

2.2.2 Prior Construction of Complete Neighbourhood

Here all the nodes and their neighbours are computed in priori in the composite visual space $v\mathcal{C}_1 \times v\mathcal{C}_2 \times \ldots \times v\mathcal{C}_m$. Collision with static obstacles for any robot can be identified in the individual roadmaps. Edges between the remaining free space nodes need to be checked for dynamic collision(section 2.2.4). This approach has a space complexity $O(\mathcal{N}_1 \times \mathcal{N}_2 \times \ldots \times \mathcal{N}_m)$ since the entire roadmap is being created. The matrix may be impossible to be stored with limited resource. However, due to the computation of the composite roadmap in priori, the subsequent queries takes constant time.

2.2.3 Dynamic Forward search Algorithm

The algorithm presented in this section[Algorithm 1] is *incremental* search algorithm using *Dijkstra* in the finite state transition graph, or roadmap as we call it, built in composite space. We do not fully specify the graph in advance as discussed in previous section. Every edge $e \in E$ of the graph G has an associated non-negative

cost $c(e)$, which is the cost of going from one state to another represented by the end vertices $s_1, s_2 \in S$ of the edge. From the path planning perspective, we explore the graph outward from the start state incrementally. To make the search time finite, we should keep track of states already visited to avoid revisiting same states indefinitely. At any point during the search, there will be three kinds of states:

1. **Unvisited:** States that has not been visited yet. Initially this is every state except the starting s_o.

2. **Dead:** States that have been visited and which cannot be reached with a lower cost path. These states have been fully expanded to its neighbours and thus it has nothing to contribute more.

3. **Alive:** States that have been visited but the cost to reach it is not known to be optimal.

Algorithm 1: Dynamic Forward Search for path $s_o \rightarrow s_g$

```
1  C(s_o) ← 0
2  Q.insert(s_o) and mark s_o as visited
3  while Q not empty do
4  |   s ← Q.getTop();
5  |   if s == s_g then
6  |   |   Print path
7  |   |   return SUCCESS
8  |   forall the s' ∈ N(s) do
9  |   |   if s' not visited then
10 |   |   |   Mark s' as visited
11 |   |   |   C(s') ← C(s) + c(e)
12 |   |   |   Q.push(s')
13 |   |   else
14 |   |   |   if C(s') > C(S) + c(e) then
15 |   |   |   |   C(s') ← C(s) + c(e)
16 |   |   end
17 |   end
18 end
19 return FAILURE
```

The set of alive states are stored in a priority queue,Q, which is sorted according to a function $C : S \rightarrow [0, \infty]$ called cost-to-come. For each state s, the value $C(s)$ at

any point of time is called the cost-to-come from the original state s_o, as computed till that moment whereas, $C^\star(s)$ is called the optimal cost-to-come. We associate two informations with a state s visited, the cost-to-come $C(s)$ and its immediate predecessor or parent state. This optimal cost is obtained by summing edge costs $c(e)$ over all possible paths from s_o to s and taking the one having least cumulative cost. The total cost of plan can be written as $C^\star(s_g)$, the optimal cost-to-come for the goal state. Initially, Q contains only the start state s_o with $C(s_o) = 0$. A while loop is then executed, which terminates only when the priority queue is empty. In each while iteration, the highest rank element s in Q is accessed and removed(with $Q.getTop()$ function) and added to the set of Dead states. If this element is the goal state s_g, it reports success. Otherwise, it needs to check for all the adjacent states $N(s)$. To find adjacent states, in our case of searching in a composite roadmap graph, we call a *neighbourhood expansion module* **findNeighbours()**, which will be discussed in a while. For each adjacent state s' discovered, we must determine whether the state is alive i.e. already in the priority queue. If it is encountered for the first time, it is pushed into Q with $C(s') = C(s) + c(e)$ where e is the connecting edge between s and s'; it is then added to the set of Alive states. Otherwise, if s' is already in Q, its value of $C(s')$ is compared with the cost given by the current path; if $C(s') > C(s) + c(e)$ its value is updated to $C(s') = C(s) + c(e)$ and its parent is set as s.

findNeighbours() function: As we plan motion for two robot system, any state(node) in the graph is represented as a pair $s = (x_1, y_1)$ by pairing the states of the individual robots. We already possess the neighbours of x_1 and y_1 with us given as input to our algorithm. Now we build the neighbourhood product lattice from individual neighbourhoods, as already discussed. Next we choose k neighbours from it according to our strategy. We choose the second strategy of selecting $\min(k_1, k_2)$ smallest elements along with $k_1 + k_2$ axial elements, here k_1 and k_2 are the count of neighbours for x_1 and y_1. For every selected neighbour which is a pair of states, we check for dynamic collision(section 2.2.4) by using local planner that will be

discussed elaborately in coming section. We check the feasibility of the composite state in terms of overlap between two robots and also the feasibility of the edge for going from original state (x_1, y_1) to it in a single step. If we find the neighbour or the edge infeasible, we drop it. Thus all the remaining states are added as adjacent to s in the composite roadmap.

Input to the algorithm:

- $\mathcal{N}_i \times k_i$ integer neighbourhood matrix nbr_i for each robot $\mathcal{A}_i$, each row containing the neighbours for an instance of the robot. The first element of the row is the instance itself.

- $\mathcal{N}_i \times k_i$ distance matrix for each robot $\mathcal{A}_i$ corresponding to nbr_i. First element of any row is zero as it stores the distance from an instance to itself.

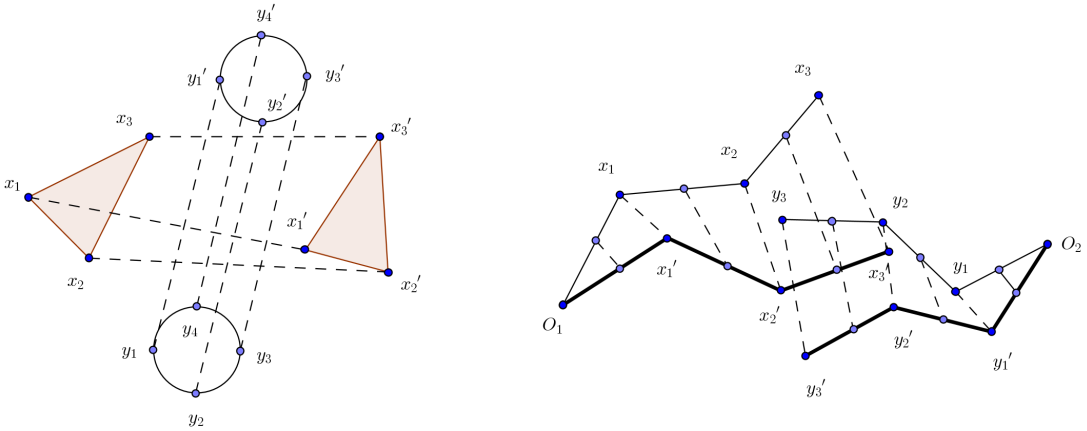
- $\mathcal{N}_i \times k_i$ boolean validity matrix for each robot $\mathcal{A}_i$. In a row of the matrix, zero indicates that the corresponding neighbour is invalid i.e. 1) overlapping with an obstacle, or 2) whether the transition is intersecting with some static obstacle. We shall drop all the invalid neighbours for a node while constructing roadmap graph for individual robots.

- Start state in composite space in the form $s_o = (x_o, y_o)$ and goal state as $s_g = (x_g, y_g)$ for two robots in our experiment.

Output: The shortest path in composite free space $\mathcal{C}_{\mathcal{A}}^{free}$; produced as a sequence of x-y pairs on the path for the two robot system.

Preprocessing stage: We build roadmap graph G_i for each robot $\mathcal{A}_i$. Each node stands for a pose instance. Every instance will be connected to all its valid neighbours by a weighted edge.

Though dynamic computation of composite roadmap saves in terms of space complexity, it incurs longer time for subsequent queries.



(a) inter-track collision (b) track-body collision

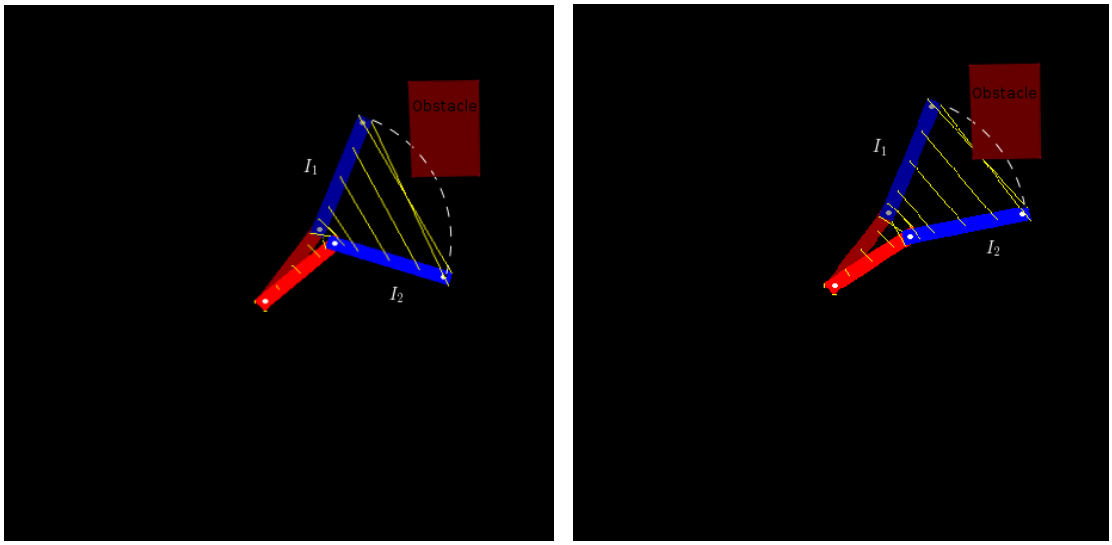
Figure 2.3: Dynamic collision detection for mobile and articulated robot

2.2.4 Collision Detection via Local Planning

In this discourse we categorize collision into static and dynamic collision. *Static collision* means collision with static obstacles when a robot moves from a state to its neighbour. Lets consider two neighbouring instances of a robot. As mentioned earlier, we have distinguished TF-points on the robot generating tracks while moving. If such a track passes through a static obstacle we mark the transition as invalid and drop the edge between them in the roadmap. This check can be performed in two ways:

- Incremental method uses a small step size and iterate over the line segment. At each point it checks for collision in image space.
- Recursive subdivision method uses binary search decomposition to check for collisions.

We perform static collision checking for individual roadmaps constructed for each robot and modify the graph accordingly. *Dynamic collision* is defined as inter robot collision for a multi robot system while the participating robots are in motion. This checking is performed while constructing the composite roadmap for multi-robot system. We perform two checks to detect such collision:



(a) failed to detect collision (b) detects collision

Figure 2.4: Failure in collision detection due to sparse sampling

- Inter track collision[Fig.2.3(a)] checks for intersection between the tracks of participating robots.

- Track-body collision[Fig.2.3(b)] detects collision many times when the first check fails. One such situation is when two mobile robots translate towards each other in parallel and thus their tracks do not intersect. For every track of a robot we check for intersection with the edges of all other robots both in their initial and final position.

In composite roadmap, all the edges causing dynamic collision are discarded.

Although local planning works well for a dense set of sampled images, it may falter when the samples are not dense enough. The following subsection gives an idea of such false positive situation.

2.2.5 Limitations of Local Planner

Tracks are supposed to represent the transit paths of the TF-points on an object while it moves from the pose I_1 to I_2. In case of rotation, actual transit path is an arc with respect to the point of rotation. When the samples are not sufficiently dense, such a track may deviate from the actual arc of transition to a great extent

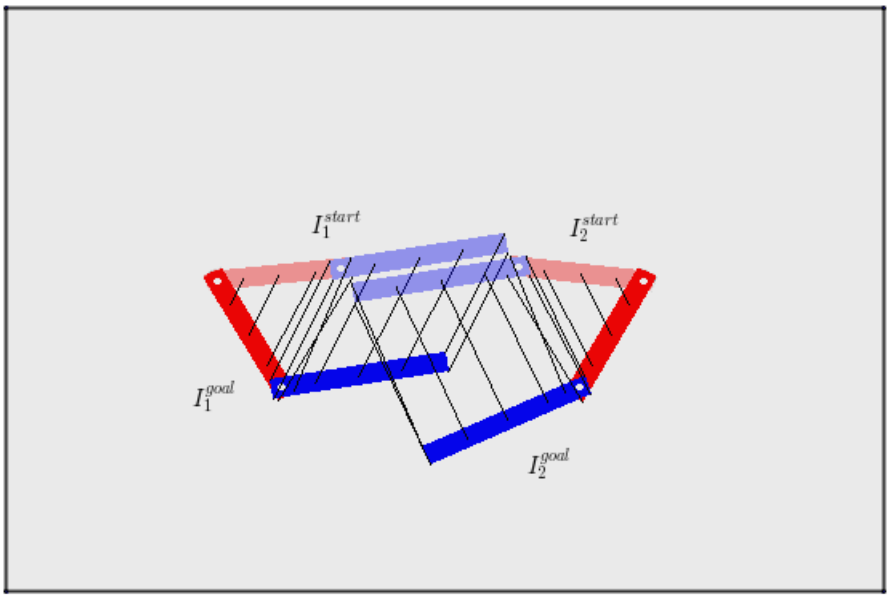


Figure 2.5: Situation of possible failure due to dynamic collision

and may fail to detect collision even if one exists. Fig. 2.4 presents an example where (a) fails to detect collision but (b) correctly detects it because of closeness. Thus

In the left hand side figure, where the sample density is less, I_2 is chosen as a neighbour of I_1 resulting in failure in collision detection; whereas in the other, because of higher density of samples, it finds a closer neighbour and saves from wrongly declaring the transition as safe.

2.2.6 Overcoming possible incompleteness

Lets consider the Fig.2.5 where two articulated arms share a common workspace. The composite space model has a set of nodes, each of which is of the form (I_1, I_2) where I_1 and I_2 are pose instances of robot $\mathcal{A}_1$ and $\mathcal{A}_2$. In this example, the system has to move from the composite state $S = (I_1^{start}, I_2^{start})$ to $G = (I_1^{goal}, I_2^{goal})$. We assume that the goal states are neighbour to the start states individually. As S and G themselves are collision free, in the sense that two arms do not intersect each other, we check for track intersection to check for collision while in motion. Based on the dynamic collision detection method, our model rejects the edge enabling the state transition $S \rightarrow G$ in spite of being a valid transition. By the second policy

of choosing k neighbours in composite space, where $k = k_1 + k_2 + \min(k_1, k_2)$ we preserve the zero motion states. It means that the state $T = (I_1^{start}, I_2^{goal})$ is also retained as a neighbour of $(I_1^{start}, I_2^{start})$. Hence, the system can now move from S to G via the intermediate state T. The first policy choosing only $\min(k_1, k_2)$ composite neighbours does not guarantee the safety. The third policy,being a superset of the second, keeps all $(k_1 \times k_2)$ neighbours and thus also ensures safety in this regard. In other motion planning models doing velocity tuning, the velocity of the two arms can be controlled to avoid collision. But that will add to the complexity of the problem.

2.3 Performance analysis

We use different metric to measure performance of the system, in terms of *completeness* and *duration* to reach goal.

We mark a number of instances of the robots to be significant. Each test case will have a start and goal state of the combined system, each state being a random combination of the marked instances. During the roadmap composition process we ensure that there is always a path between two joint states if there are paths for the individual robots from their corresponding start states to goal states. For all three policies of neighbourhood composition, we report the number of test cases where, even if the individual robots succeed to reach their goal states ignoring the existence of the others, the joint system fails to reach its goal. This gives the percentage of success(S).

We evaluate path duration of the composite system in two ways: firstly, by comparing the number of steps taken by the system($\mathcal{A}$) with respect to the maximum number of the steps taken by any single robot($\mathcal{A}_1$ or $\mathcal{A}_2$). This is to discourage the unnecessary pause of any of the robots when the other is moving resulting in increase of the overall number of steps. This is given by performance,

$$P_{step} = \frac{max(steps(Path_{\mathcal{A}_1}), steps(Path_{\mathcal{A}_2}))}{steps(Path_{\mathcal{A}})} \tag{2.2}$$

Neighbourhood composition Rule	**S**	P_{step}	$P_{distance}$
$k = \min(k_1, k_2)$	0.95	0.65	0.81
$k = k_1 + k_2 + \min(k_1, k_2)$	1	0.79	0.96
$k = k_1 \times k_2$	1	0.89	0.98

Table 2.1: Performance comparison table

secondly, by comparing the distance covered($d^C_{\mathcal{A}}$) by the system with sum of distances covered by individual robots for their path computed singularly ignoring others. This metric is given by,

$$P_{distance} = \frac{1}{2}(\frac{d^D_{\mathcal{A}_1}}{d^C_{\mathcal{A}_1}} + \frac{d^D_{\mathcal{A}_2}}{d^C_{\mathcal{A}_2}}) \tag{2.3}$$

For a joint state transition from initial (x_m, x_n) to goal (x'_m, x'_n), $d^C_{\mathcal{A}_i}$ represents the distance covered by robot $\mathcal{A}_i$ in combined space(C) for going from initial to goal state. $d^D_{\mathcal{A}_i}$ gives the distance covered by the $\mathcal{A}_i$ robot in singular(decoupled) workspace.

In the table above we compare the three performance metric given by the success rate S,P_{step} and $P_{distance}$ for three policies of composing road maps.

The values of the performance metrics are the average of 1000 runs on a set of well distributed instances.

2.4 Experiment 1: Planar mobile robots

In this section we will show the experimental result obtained by motion planning for a two robot system, a triangular and circular robot respectively. In later section we will show example of motion planning for articulated arm.

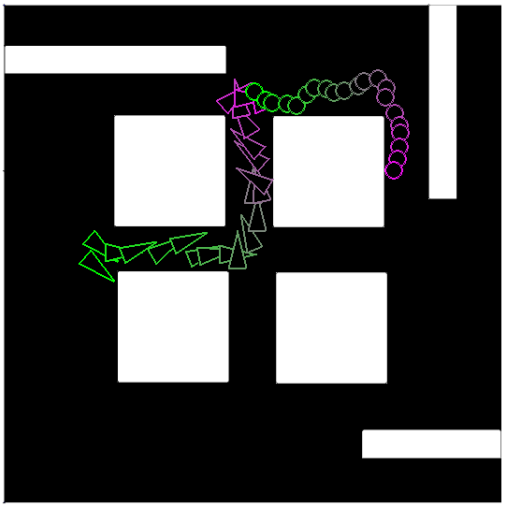


Figure 2.6: Each of the robots maintain its own shortest path computed independently when the paths are non-conflicting. Motion has been shown via color gradient with green shade marking the initial pose.

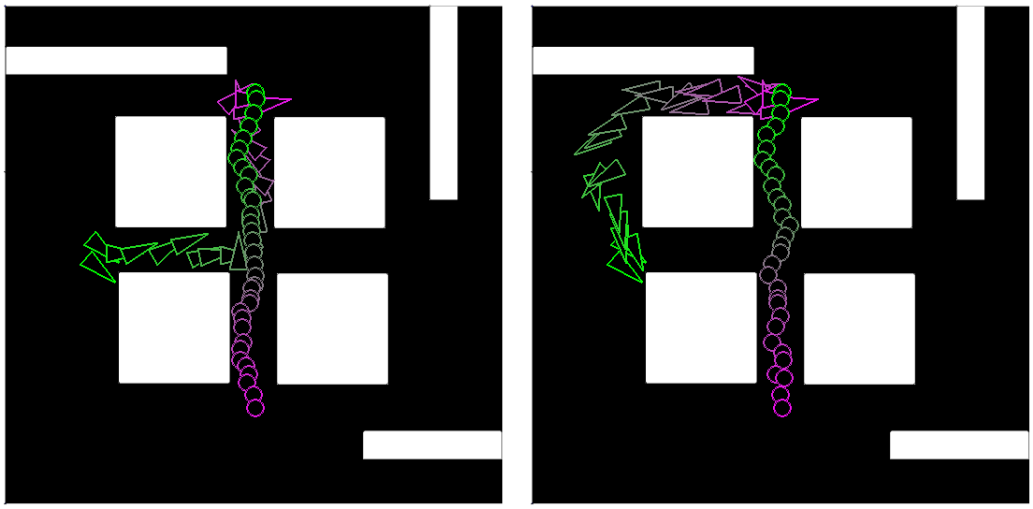
Scenario 1: For some start and goal states, the individual shortest paths for the two robots do not intersect. We expect both robots to follow their individual shortest paths when combined. We present one such scenario in Fig. 2.6. But the number of steps taken by $\mathcal{A}$ may be greater than $\max(steps(Path_{\mathcal{A}_1}), steps(Path_{\mathcal{A}_2}))$ in their individual space when the composite neighbourhood size is not $k_1 \times k_2$.

The image on the left shows the transition in image space for the two robots. Start configuration is shaded in green. In case of composite neighbourhood size $k = \min(k_1, k_2)$ the paths computed for the robots on composite roadmap may deviate from their own shortest path computed in the individual roadmaps. However, for the other two policies of choosing the value of k, such deviation cannot happen for non conflicting paths, as we preserve the zero motion states there.

Scenario 2: Some of the edges of $Path_{\mathcal{A}_1}$ are in conflict with some edges in $Path_{\mathcal{A}_2}$. One solution, often forced by fixed path fixed path coordination methods,

is to let one of the robots to pause before the colliding state and allow the other robot to pass. After the other robot passes it resumes and follow the path to the goal.

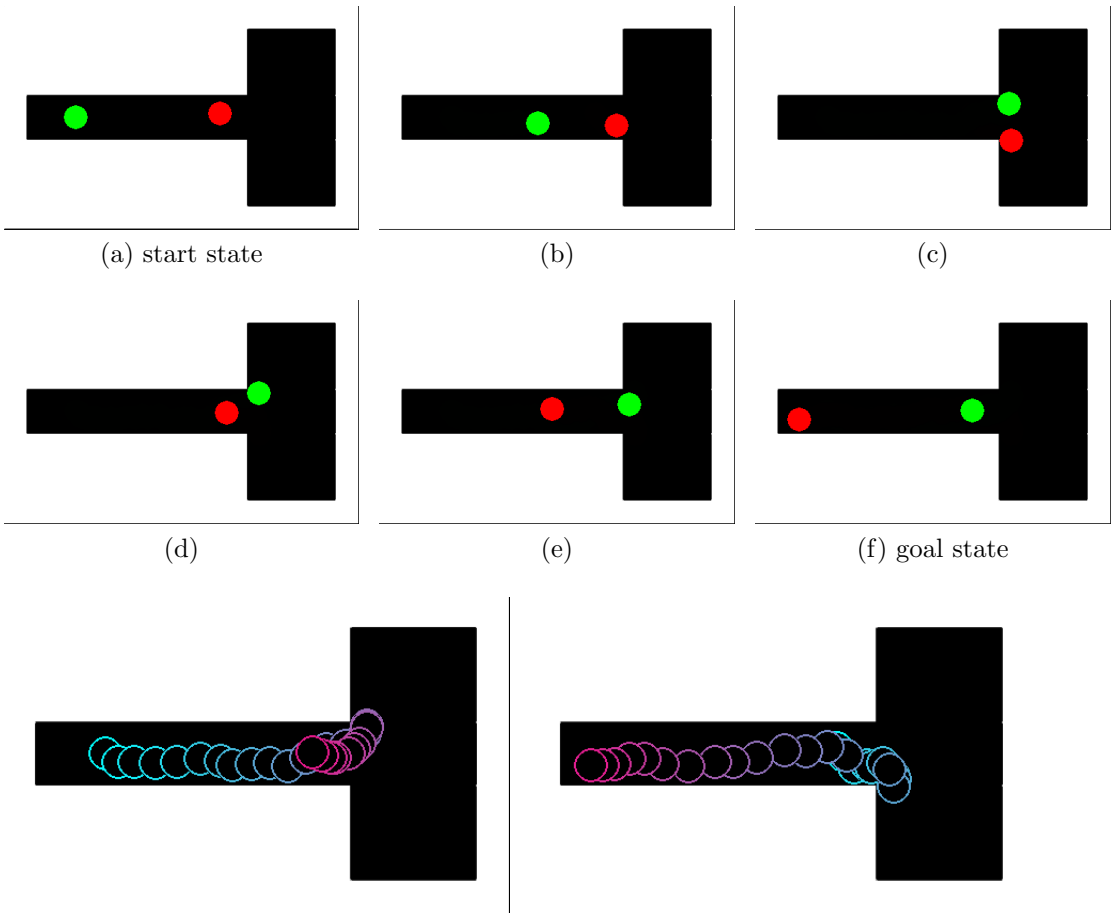
Fig. 2.7 shows such partial intersection of paths. The first figure is a result of motion planning with distance covered as cost function. As we can see, the robots follow their own shortest paths. Distance being the cost function, the wait time of triangle does not contribute much to the total cost; whereas in the second case using total number of steps taken by the whole system as cost function, wait time matters as it results in the inflation of total number of steps. Hence the triangle takes a different non-conflicting path to reach its goal so that it does not have to wait.



(a) motion using distance as cost (b) motion using step-count as cost

Figure 2.7: Effect of using distance and number of steps as cost function for partially overlapping paths. In the first figure using distance as cost function, the triangle has to wait for the circle to cross the collision zone. Motion has been shown via color gradient with green shade marking the initial pose.

Scenario 3: $Path_{\mathcal{A}_1}$ and $Path_{\mathcal{A}_2}$ completely overlaps. In such cases, there are two possibilities. One of the robots can take an alternative way instead of its original shortest path. Other possibility is when they are in a narrow corridor and they want to go in opposite direction. In that case one has to back off to open space to make way for the other. After the other robot crosses the collision zone, the former one moves back in and set off on its original path leading to the goal. Here we can see that one of the robots overshoots its original shortest path.



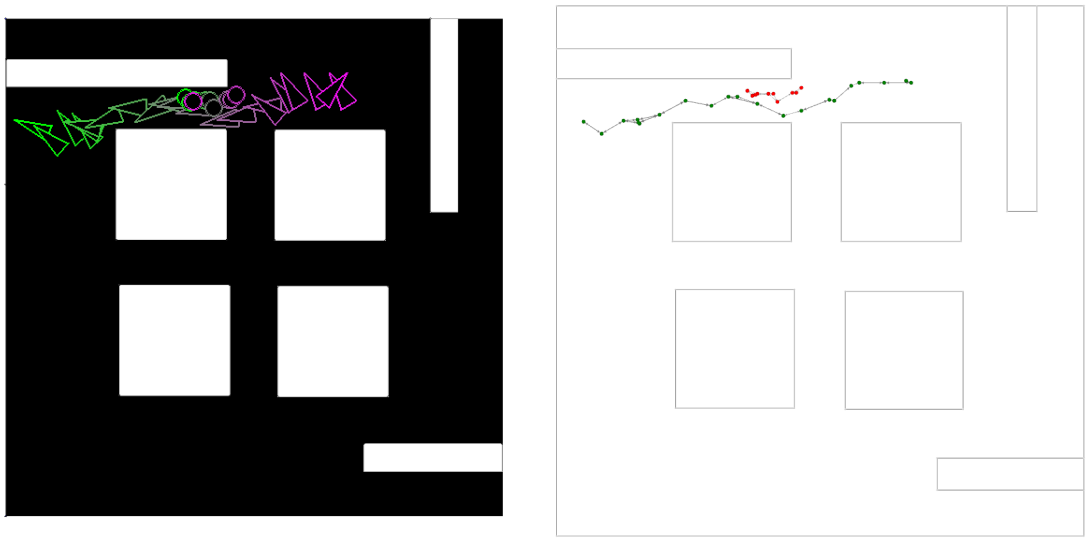
(a) start state (b) (c)

(d) (e) (f) goal state

(g) Motion of green disc (h) Motion of red disc

Figure 2.8: Due to total overlap of paths in the narrow corridor, the red disc has to back off to the open area to give way to the green disc. After the green disc comes out, the red one goes back in followed by the green disc. Motion has been shown via color gradient with cyan marking the initial pose.

Fig.2.8 shows an examples of two circular robots moving in a T-shaped workspace with a wider area at one side. The figure shows the start and goal states. To make the transition possible, the red disc has to come out to the wider area on the right followed by the green disc. Then the red one moves back in to the narrow corridor first to reach its goal. It is followed by the green robot. Fig.2.9 shows this for a triangular and a circular robot case. Start and goal position of the circular robot is same, situated in the narrow corridor. To give way to the triangular robot which has to cross the corridor to reach its goal, it moves to a safe(not intersecting) position, waits until the triangle crosses the corridor and then again moves back its original position, which is its goal too. Fig.2.9 shows a similar situation for triangular and



(a) Motion shown in image space (b) Motion shown on the roadmap graph

Figure 2.9: Example showing total overlap of paths for triangle-circle mobile robots. Similar to the example of the T-shaped workspace, the disc has to back off to the open and goes back in when the triangle has crossed the narrow corridor. Here the start and goal position of the disc are same. Motion has been shown via color gradient with green shade marking the initial pose.

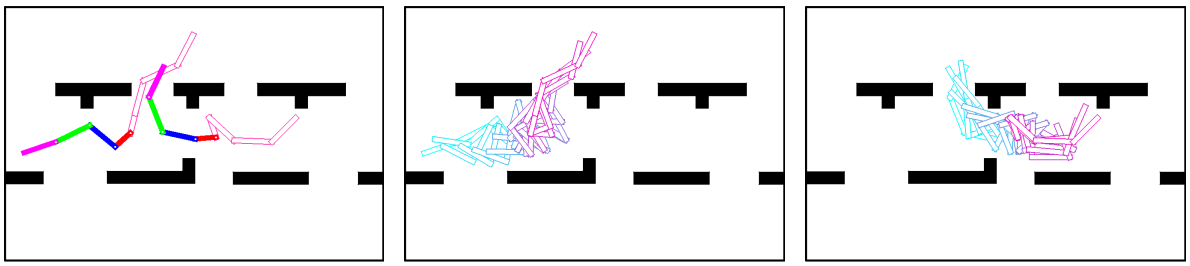
circular mobile robots. Here the circular robot has to move back for the triangular robot to pass.

2.5 Experiment 2: Manipulators

In this section we show the generic nature of our method by extending it to motion planning of two four degree of freedom planer articulated arms or manipulators sharing a common workspace. They are pinned in such a way that they can cross each other. We present some example scenarios in the following:

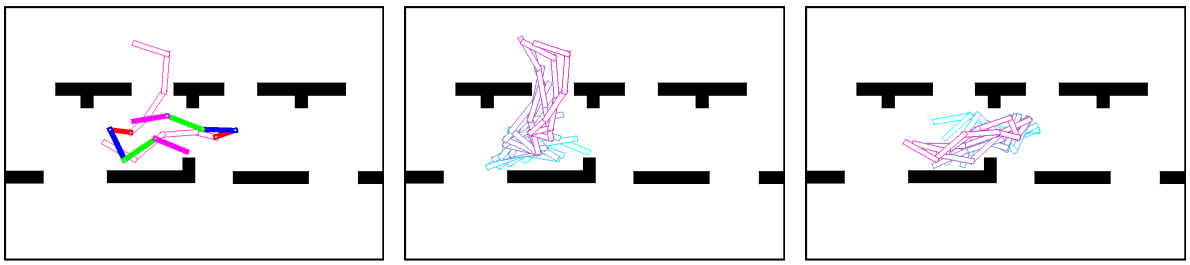
Scenario 1: The shortest path in the joint space do not cause any conflict between two arms. The figure 2.10 shows one such example with the start and goal states as well the transition of the two arms. The transitions have been shown by using color gradient where cyan marks the start and purple marks the end poses.

Scenario 2: The shortest path computed in joint space causes collision which is detected by dynamic collision detection. The edge causing the conflicting step transition in composite roadmap is dropped and alternative path is found in free



(a) Start & goal states. Goal state shown in line diagram (b) Tracing arm1 transition (c) Tracing arm2 transition

Figure 2.10: Non-existence of any possible conflict



(a) Start & goal states. Goal state shown in line diagram (b) Tracing arm1 transition (c) Tracing arm2 transition

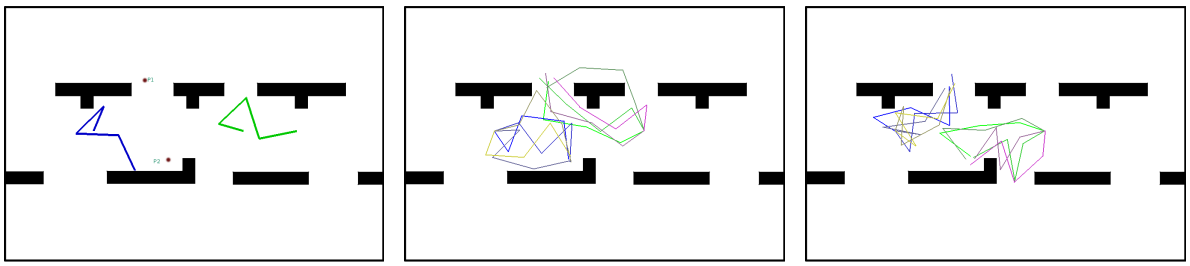
Figure 2.11: Avoidance of possible conflict by dynamic collision detection

space $v\mathcal{C}^{free}$. The start and goal position as well as the transition of two arms are shown in the figure 2.11.

The results presented here are obtained using sample size of 5000 for both arms. We observe that the taken sample density does not fully ensure resolution completeness as there are few cases where no path can be discovered even if one exists. This happens because of excessive deletion of neighbours for some poses by local planner due to collision with static obstacle in individual spaces, and thus disconnecting the pose from its corresponding roadmap graph.

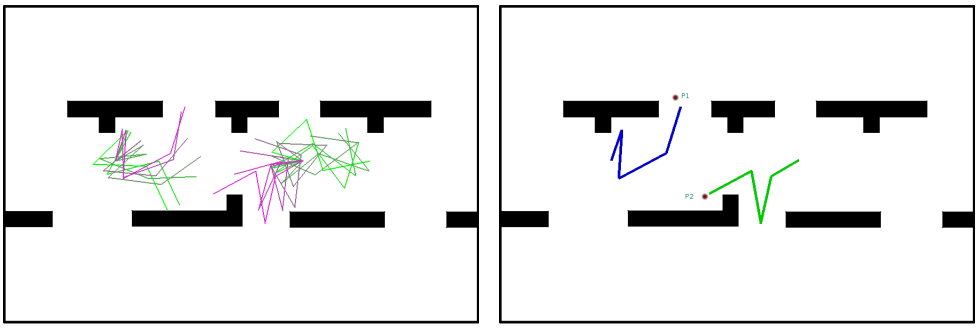
2.6 An application of motion planning with two-Articulated Arms

To exploit the shortest(or fastest) path computation of our method, we use the two articulated robotic arms to perform a task. The task is as follows: given two points in terms of their coordinates in the workspace, the two manipulators need



(a) Starting state with target coordinates (b) Step 1: $\mathcal{A}_1$-P_1,$\mathcal{A}_2$-P_2 (c) Step 2: $\mathcal{A}_1$-P_2,$\mathcal{A}_2$-P_1

Figure 2.12: Start configuration shown in a for case 1.The sets S_g^1 and S_g^2 are demonstrated in b and c respectively



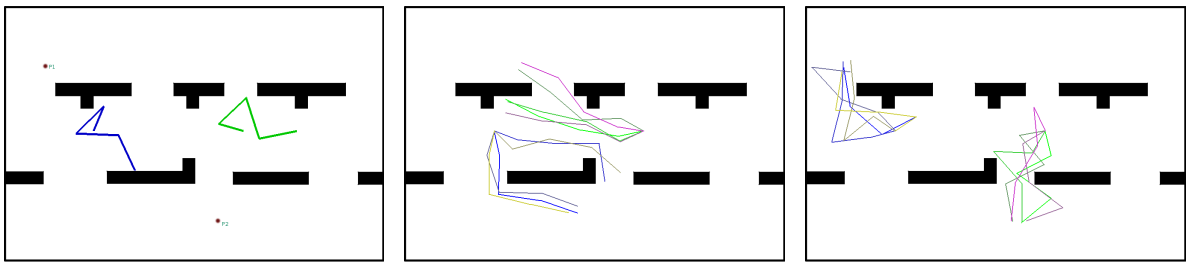
(a) Optimal path (b) Selected goal state

Figure 2.13: optimal path and selected goal in case 1

to reach the points in any order. The objective is to reach in quickest possible way without any collision. As we are working with discrete image samples, we modify the problem statement a little. The task becomes to reach a location inside a circle centred at the points. By reaching a point we mean that the end effector of an arm should touch the point.

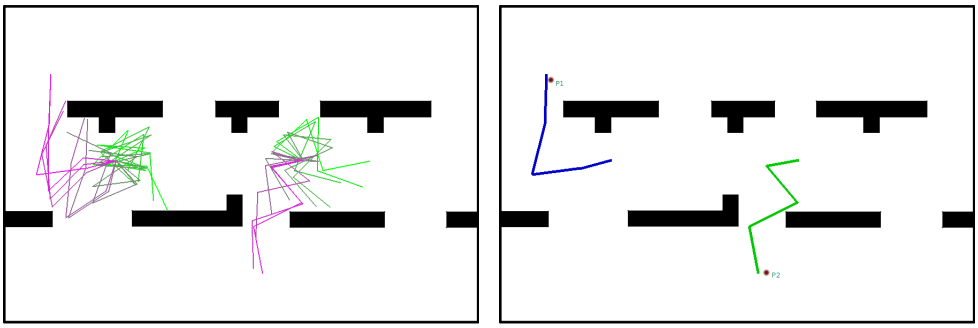
Solution: First for each point we identify k instances of both of the arms which has its end effector closest to the point. Now every location on the workspace is not reachable by an arm. We fix a radius r. If none of the instances of an arm has end effector inside this circle, we declare the point as unreachable for that arm. We use the convention of $\mathcal{A}_1$ for the manipulator arm pivoted right and $\mathcal{A}_2$ for the other. Following are some cases:

Case 1: When both the target points P_1 and P_2 are reachable by both of the arms, we do motion planning in two steps: *Step 1* is to compute cost when $\mathcal{A}_1$ goes to P_1 and $\mathcal{A}_2$ goes to P_2 (Fig.2.12.b). Given a composite start state, we define



(a) Starting state with target coordinates (b) Step 1: $\mathcal{A}_1$-P_1,$\mathcal{A}_2$-P_2 (c) Step 2: $\mathcal{A}_1$-P_2,$\mathcal{A}_2$-P_1

Figure 2.14: Start configuration shown in *a* for case 2.The sets S_g^1 and S_g^2 are demonstrated in *b* and *c* respectively



(a) Optimal path (b) Selected goal state

Figure 2.15: optimal path and selected goal in case 2

composite goal state as any member of the set $S_g^1 = \{(x, y)|x \in v\mathcal{C}_1 \cap N(P_1), y \in v\mathcal{C}_2 \cap N(P_2)\}$ where $N(X)$ gives k instances of a robotic arm closest to point X. We save the joint state $h_1 = \arg\min_{s \in S_g^1} \textit{cost-to-come}(s)$ giving the goal state corresponding to the minimum cost path, computed by dynamic search algorithm as already discussed. In *Step 2* we similarly find path producing minimum cost and corresponding goal state when $\mathcal{A}_1$ goes to P_2 and $\mathcal{A}_2$ goes to P_1 (Fig.2.12.c). Analogous to step 1, S_g^2 is the set of possible goal states (x, y) in composite space where $x \in N(P_2)$ and $y \in N(P_1)$. We save $h_2 \in S_g^2$ giving the minimum cost. Finally, we choose one between h_1 and h_2 giving lower cost as the goal state. Fig.2.13 shows optimal path along with the winning goal state. This is an example of **success.**

Case 2: The arms can reach only two different points. Figure 2.14.c shows the only possible reachability combination when $\mathcal{A}_1$ reaches P_2 and $\mathcal{A}_2$ reaches P_1. In the other way it (Fig.2.14.b) is not possible to reach goal. This is an example of

success. Fig.2.15 shows the optimal path along with selected goal state.

Case 3: One of the points between P_1 and P_2 is unreachable by both arms. This is an example of **failure.**

Case 4: Both P_1 and P_2 are unreachable by each of the arms. Hence another example of **failure**.

2.7 Space Complexity of Dynamic forward search

Following is the summary of 100 runs for two types of robots:

Robot type	**Sample size per robot**	$HeapSize_{avg}$	$HeapSize_{max}$	**Discarded edges by dyn. Collision**
Mobile robots	5000	291927 nodes	416471 nodes	0.0014
Articulated arm	5000	767872 nodes	1077327 nodes	0.0237

Table 2.2: Space Complexity of Dynamic Search

Chapter 3

Multi-robot Motion planning on Visual Manifold

The foundation of the chapter lies in the visual manifold hypothesis discussed in the introduction. It aims at encoding the configuration space by performing a homeomorphic reconstruction of the canonical motor manifold in the visual space, and planning path on it. Here we consider the entire multi-robot system as a single robot having multiple parts. So we do not disintegrate the system. To retrieve the intrinsic low dimensional embedding from the images we implement dimensionality reduction technique.

The classical techniques for dimensionality reduction PCA and MDS are effectively implemented for low-dimensional embedding of linear data points. Classical MDS finds the embedding that preserves the internal geometry i.e. relative inter-point distances are preserved. Often such data points lie on a low dimensional non linear manifold in high dimensional input space. Each point on the manifold represents one data point. The actual distance between two points spaced apart on the manifold can not be measured along a straight line. Thus PCA and traditional MDS fails as they see just the Euclidean structure. So we use a different measure called *Geodesic* distance. Here we keep the edges only between nearby points and discard the rest. To compute the distance between two far away points we take sum of the

edge-lengths along the path from one point to another. MDS using geodesic distance as proximity measure is known as Isomap. We need to compute the distance matrix for N data points which requires $O(N^2)$ space. For large dataset this turns out to be very expensive. Hence we use a variant of Isomap which is known as *Landmark Isomap*.

3.1 Objective Formulation

Given N input points $X = \{x_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^D$ the goal is to find corresponding embedding $Y = \{y_i\}_{i=1}^{N}$ where $y_i \in \mathbb{R}^d, d \ll D$ such that Y 'faithfully' preserves the spacial geometry i.e $(1-\epsilon)||x_i - x_j|| \leq ||y_i - y_j|| \leq (1+\epsilon)||x_i - x_j|| \ \forall i, j \in [N]$ and $\epsilon > 0$ is a very small positive constant.

3.2 Summary of Landmark method

1. Unless classical MDS computing the inner product matrix of the whole dataset taking $O(N^2)$ space, landmark method uses a random subset of $l \geq d+1$ points and compute the inner product matrix on that subset which essentially takes $O(l^2)$ space.

2. Perform classical MDS on the $l \times l$ inner product matrix to get d dimensional embedding of l landmarks. We call the matrix L.

3. The $l \times N$ distance matrix(taken as input) along with L is used to compute the embedding of all N points.

3.3 Approximate Spectral decomposition: The inspiration behind

MDS works on $N \times N$ positive semi-definite inner product matrix G. When N is large G becomes too large to store. Time complexity to solve for Eigen-decomposition

on G becomes $O(N^3)$. Thus it becomes unaffordable both in terms of space and time. So we go for approximating G which can be decomposed as $G = U\Sigma U^T$. Here Σ contains the eigenvalues of G and columns of U are the associated eigenvectors. Suppose we randomly sample $l \leq N$ columns of G uniformly without replacement. Let C be the $N \times l$ matrix of those sampled columns, and W be the $l \times l$ matrix consisting of the intersection of the l columns with the corresponding l rows of G. Since G is positive semi definite, W is also positive semi-definite. Without loss of generality, we can rearrange[Talwalkar et al. (2008)] the columns and rows of G based on the sampling such that,

$$G = \begin{pmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{pmatrix} \tag{3.1}$$

where $W = l \times l$ matrix and is PSD and $G_{21} = (N - l) \times l$ matrix.

$$C = \begin{pmatrix} W \\ G_{21} \end{pmatrix} \tag{3.2}$$

a $N \times l$ matrix.

Next we discuss two approximation techniques that use eigendecomposition of W or singular value decomposition of C to closely estimate U and Σ.

3.3.1 Nyström Method

Nyström method is used to speed up performance of Kernel machine. It has been used for Landmark Isomap, which we are going to discuss in details in a while. Nyström method uses W and C to approximate G. According to the method,

$$G \approx \tilde{G} = CW^+C^T \tag{3.3}$$

where W^+ is the pseudo-inverse of W. $\tilde{G}$ converges to G as l increases. Substituting 3.2 into 3.3, the Nyström method reduces to approximating G_{22} in $\tilde{G}$ as following:

$$\tilde{G} = \begin{pmatrix} W & G_{21}^T \\ G_{21} & G_{21}W^+G_{21}^T \end{pmatrix} \tag{3.4}$$

The approximate eigenvalues and eigenvectors of G generated from the Nyström method are:

$$\tilde{\Sigma} = \frac{N}{l}\Sigma_W \tag{3.5}$$

$$\tilde{U} = \sqrt{\frac{l}{N}}CU_W\Sigma_W^+ \tag{3.6}$$

where $W = U_W\Sigma_W U_W^T$.

To calculate approximations to top k eigenvalues or eigenvectors of G, the runtime of the algorithm is $O(l^3 + klN)$, l^3 for eigendecomposition of W and klN for multiplication with C.

3.3.2 Column-sampling Approximation

It approximates eigendecomposition of G by using SVD of C directly. Suppose $C = U_C\Sigma(V_C)^T$, then the approximate eigenvectors of G are given by the left singular vectors of C:

$$\tilde{U} = U_C = CV_C\Sigma_C^+ \tag{3.7}$$

and the corresponding eigenvalues are scaled version of the singular values of C:

$$\tilde{\Sigma} = \sqrt{\frac{N}{l}}\Sigma_C \tag{3.8}$$

Combining 3.7 and 3.8 and $\tilde{G} = \tilde{U}\tilde{\Sigma}\tilde{U}^T$, we get:

$$G \approx \tilde{G} = C(\sqrt{\frac{l}{N}}(C^TC)^{\frac{1}{2}})^+C^T \tag{3.9}$$

Comparing this with 3.3 we find two forms of approximating G to be quite similar, only difference being W^+ getting replaced by $\sqrt{\frac{l}{N}}(C^TC)^{\frac{1}{2}})^+$.

SVD on C costs $O(Nl^2)$ but since it cannot be easily parallelized, it is expensive for large N. However, since $C^TC = V_C\Sigma_C^2V_C^T$, one can get U_C and Σ_C by SVD on C^TC and using 3.7. This is easy even for large N since matrix multiplication can be parallelized. Thus time needed to calculate approximations on top k eigenvectors and eigenvalues of G is $O(Nl^2 + l^3)$. Nl^2 to generate C^TC and l^3 for SVD of C^TC.

3.4 Landmark MDS

The Landmark MDS algorithm[De Silva and Tenenbaum (2004)] is used to find an efficient approximation to the Isomap algorithm for non-linear dimensionality reduction suitable for processing large data sets. The process uses Nyström method to approximate the inner product matrix. We present a brief outline of the algorithm.

3.4.1 Outline of Landmark Isomap:

- **Choosing Landmark points:** We designate $l \geq d + 1$ landmark points randomly uniformly without replacement for d dimensional embedding. The random selection works as good as greedy Maxmin approach and with less cost.

- **Create squared distance matrix:** Find k nearest neighbours for all the N datapoints and compute $N \times N$ sparse distance matrix by only retaining the distances between neighbours. Using this matrix as input to *Dijkstra* we find the $l \times N$ Geodesic squared distance matrix Δ which stores squared shortest path distance from l landmark points to all N points.

- **Classical MDS on landmarks:** Perform classical MDS on landmarks. We call the $l \times l$ squared distance matrix Δ_l. We derive the inner product matrix

B_l from Δ_l by,

$$B_l(i,j) = -\frac{1}{2}(\Delta_l(i,j) - \frac{1}{l}\sum_{j=1}^{l}\Delta_l(i,j) - \frac{1}{l}\sum_{i=1}^{l}\Delta_l(i,j) + \frac{1}{l^2}\sum_{i,j=1}^{l}\Delta_l(i,j))$$

We compute d largest positive eigenvalues λ_i together with their corresponding orthonormal set of eigenvectors $\vec{v_i}$. Just to mention, the nonpositive eigenvalues includes a zero corresponding to eigenvector $\vec{1} = [1, 1, \ldots, 1]^T$ because B_l being inner product matrix of mean centered data points its rowsum is a zero vector. The required d dimensional embedding vectors $y_1, y_2, \ldots, y_l$ are given by the columns of the following matrix

$$L = \begin{pmatrix} \sqrt{\lambda_1}.\vec{v_1}^T \\ \sqrt{\lambda_2}.\vec{v_2}^T \\ \vdots \\ \sqrt{\lambda_d}.\vec{v_d}^T \end{pmatrix} \tag{3.10}$$

Based on the assumption that x_is are mean centred by translation, i^{th} row of L gives the components of each data point when projected into i^{th} principal axis of the data (*Young & Householder*). This embedding is automatically mean centred. We can validate it from the fact that $\vec{v_i}^T\vec{1} = 0$ for all i.

- **Distance based Triangulation:** Embedding coordinates of remaining data points are calculated based on their distances from the landmark points. We describe the method below.
  - $\vec{\delta_\mu}$= $(\vec{\delta_1}+\vec{\delta_2}+\ldots+\vec{\delta_l})/l$, where $\vec{\delta_i}$ denotes the i^{th} column in the distance matrix Δ_l.
  - Compute pseudoinverse transpose of L.

$$L^{\#} = \begin{pmatrix} \vec{v_1}^T/\sqrt{\lambda_1} \\ \vec{v_2}^T/\sqrt{\lambda_2} \\ \vdots \\ \vec{v_d}^T/\sqrt{\lambda_d} \end{pmatrix}$$

Each data point a is embedded as follows:

$$\vec{y_a} = -\frac{1}{2}L^{\#}(\vec{\delta_a} - \vec{\delta_\mu}) \tag{3.11}$$

$\vec{\delta_a}$ is the vector of squared distances between point a and l landmark points.

3.5 Experimental Result

Our experiment involves two planar mobile robots, one triangular and one circular. The triangle can translate as well as rotate around its circumcenter. So the joint system has five degrees of freedom. The set of random snapshots of the combined system is used as input. Track-distance is used as dissimilarity measure between two images. We get the geodesic distance between two points far apart by summing up the weights of connecting edges. Then we perform landmark Isomap based on a subset of points randomly chosen from the entire set. L-Isomap generates a low dimensional embedding of the images on vision manifold. Now the work remains to find shortest path between any two points on this manifold.

From the experimental result we see that the path computed on the manifold deviates too much from the actual shortest path. As we treat multiple robots as a single multi-part robot, we are unable to detect dynamic collision, that is, collision between two robots while in motion. Thus the valid paths identified by this method include a large number of false positive examples. Performance wise, this method is much inferior to the roadmap composition technique discussed the previous chapter.

3.6 Learning distance metric

In Isomap we used track distance as the measure of dissimilarity between two points. Here we search for a better measure of dissimilarity by learning it. This is largely inspired by the technique called Large Margin Nearest Neighbour [Weinberger and Saul (2009)]. This is used for KNN classification in supervised setting. We tried to adapt LMNN in our unsupervised scenario. First we perform MDS with track distance to get an intermediate lower dimension. We fixed this dimension to be 100. Now we perform LMNN on the set of 100 dimensional vectors. We used Euclidean distance between original vectorized images to mark k nearest points for each point as target neighbours. Among the rest, we marked the ones mimicking as a candidate for k nearest neighbours in 100-dimensional space as imposters. Thus we get a set of target neighbours and imposters for every point in our set. This information is used by LMNN which aims at updating the Mahalanobis distance matrix(in other way, linear transformation matrix) by solving a semi-definite optimization problem. The optimization problem is solved gradient descent and thus learning the linear transformation matrix. The matrix maps the 100-dimensional vectors into our target dimension of embedding.

The above method of LMNN suffers from a few drawbacks. It generates a linear transformation, which may not be suitable for intrinsically non-linear data. It may be trapped by local minima. Large Margin Component Analysis[Torresani and Lee (2006)] addresses the issue by exploiting a kernelized version of LMNN.

We tried to implement both of the above methods. But they failed to produce any mentionable result. Because of the non-linear nature of our data, LMNN failed miserably. Implementing LMCA with radial basis kernel or Gaussian kernel did not improve it much.

Chapter 4

Conclusion and Further scope

In this work, we presented a method for visual motion planning of multiple robots. As against centralized path planning, our planning technique is decoupled in nature. It is a special implementation of *fixed-roadmap coordination* method where we plan path in visual configuration space. We do not use motion parameters of the multi-robot system. Planning path on a configuration space defined in terms of motion parameters becomes intractable when the dimension of the configuration space grows up. Thus centralized learning methods, though probabilistically complete, suffer from the exponential time complexity for high dimensional multi-robot system. On the other hand, decoupled motion planning algorithms typically plan motion for component robots individually in their own configuration spaces and then find some way to combine the paths by regulating their velocity or by assigning priority to them in a clever way. Despite being efficient, decoupled planning suffers from incompleteness because of the independent pre-computation of the individual paths.

In our method discussed in chapter two, we compose the roadmap in composite space from the individual road maps. The probabilistic completeness of our roadmap composition technique stands on the policy of selecting k nearest neighbours of a composite state from the neighbourhood product matrix. The best policy is when we preserve zero motion state transitions in composite space by retaining the axial entries in the neighbourhood product lattice. By doing this we retain the origi-

nal edges in the individual roadmaps. We build roadmap graph for the composite space dynamically, minimizing the space requirement to store it on the disk. Static and dynamic collisions are checked by local planner. Edges corresponding to unsafe transitions are removed from the roadmap. Planning path on the remaining graph is equivalent to motion planning for all the robots jointly in free space and without mutual interception. We show the generic nature of the visual roadmap composition by implementing it on two kinds of robotic system: planar mobile robots and articulated robotic arms sharing a common workspace.

An alternative method, as discussed in chapter three, attempts to plan motion on visual manifold computed by Isomap on a set of random images. To reduce the space complexity, which is quadratic to the number of samples, we used Landmark Isomap method which converts the space requirement into linear to the number of samples. But this method results in the performance which is suboptimal and unauthentic to a great degree as compared to our method in previous chapter.

4.1 Further Scope

Our experiments involved two robots. As we have established theoretically, one should be able to extend it easily to more than two robots. Nevertheless, this may cause inflation in the time needed to build composite roadmap by doing dynamic forward search.

Bibliography

Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 271–276. IEEE, 2001.

John Canny. *The complexity of robot motion planning*. MIT press, 1988.

Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004.

Mokhtar Gharbi, Juan Cortés, and Thierry Siméon. Roadmap composition for multi-arm systems path planning. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2471–2476. IEEE, 2009.

Michael SA Graziano. Where is my arm? the relative role of vision and proprioception in the neuronal representation of limb position. *Proceedings of the National Academy of Sciences*, 96(18):10418–10421, 1999.

Nicholas P Holmes and Charles Spence. The body schema and multisensory representation (s) of peripersonal space. *Cognitive processing*, 5(2):94–105, 2004.

John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". *The International Journal of Robotics Research*, 3(4): 76–88, 1984.

Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

Steven M LaValle. Rapidly-exploring random trees a ew tool for path planning. 1998.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Patrick O'Donnell, T Lozano-Periz, et al. Deadlock-free and collision-free coordination of two robot manipulators. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 484–489. IEEE, 1989.

MS Ramaiah, Ankit Vijay, Gitika Sharma, and Amitabha Mukerjee. Head motion animation using avatar gaze space. In *Virtual Reality (VR), 2013 IEEE*, pages 95–96. IEEE, 2013.

Giacomo Rizzolatti, Luciano Fadiga, Vittorio Gallese, and Leonardo Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive brain research*, 3(2): 131–141, 1996.

Mitul Saha and Pekka Isto. Multi-robot motion planning by incremental coordination. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5960–5963. IEEE, 2006.

Gustavo Sanchez and Jean-Claude Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 2112–2119. IEEE, 2002.

Jacob T Schwartz and Micha Sharir. On the piano movers problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4(3):298–351, 1983.

Thierry Siméon, Sylvain Leroy, and J-P Lauumond. Path coordination for multiple mobile robots: A resolution-complete algorithm. *Robotics and Automation, IEEE Transactions on*, 18(1):42–49, 2002.

Ameet Talwalkar, Sanjiv Kumar, and Henry Rowley. Large-scale manifold learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. In *Advances in neural information processing systems*, pages 1385–1392, 2006.

Jur van Den Berg, Jack Snoeyink, Ming C Lin, and Dinesh Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and systems*, volume 2, pages 2–3. Citeseer, 2009.

Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10: 207–244, 2009.