2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)
Banff, Alberta, Canada, July 12–15, 2016

Recent Progress on Sampling Based Dynamic Motion Planning Algorithms

Andrew Short, Zengxi Pan, Nathan Larkin and Stephen van Duin

Abstract— **This paper reviews recent developments extending sampling based motion planning algorithms to operate in dynamic environments. Sampling based planners provide an effective approach for solving high degree of freedom robot motion planning problems. The two most common algorithms are the Probabilistic Roadmap Method and Rapidly Exploring Random Trees. These standard techniques are well established, however they assume a fully known environment and generate paths ahead of time. For realistic applications a robot may be required to update its path in real-time as information is gained or obstacles change position. Variants of these standard algorithms designed for dynamic environments are categorically presented and common implementation strategies are explored.**

I. Introduction

Robots commonly operate in changing and unknown environments and need to be able to modify motion plans as information changes. This is particularly challenging for high Degree of Freedom (DOF) systems where motion planning is a computationally expensive operation. Sampling Based Planning (SBP) algorithms have proven to be an effective way to solve high-DOF planning problems [1], however conventional algorithms assume a static, known environment. Recent work on SBPs has expanded these algorithms to perform real-time planning in dynamic or unknown environments allowing new behaviours and interactions for high DOF robotic systems.

Elbanhawi [1] defines the motion planning problem as generating a path P from a start configuration q_{start} to a goal configuration q_{goal} such that P lies entirely within the free space C_{free}. Planning is performed within a robot's configuration space (C-space). C-space is divided into free space C_{free} and obstacle space C_{obs}. A configuration q is a single point in C-space, alternatively called a node, sample, or milestone. A path P is defined by a continuous sequence of configurations.

In a dynamic motion planning problem C_{obs} and C_{free} may be known or unknown and may change with sensor data, or as a function of time. This reduces the suitability of conventional sampling based motion planning algorithms. In this paper we review recent work to expand the capability of SBP for solving the motion planning problem in dynamic environments and explore implementation strategies that make use of parallel computing, pre-processing and distributed computing systems.

Andrew Short, Zengxi Pan, Nathan Larkin and Stephen van Duin are with the School of Mechanical, Materials and Mechatronic Engineering at the University of Wollongong, Wollongong 2500, AU.

This paper begins with a background on sampling-based planning and dynamic planning. Section II then details a number of sampling-based algorithms designed to solve the motion planning problem in dynamic environments. Section III discusses some common implementation techniques and Section IV concludes this paper with key themes.

A. Sampling Based Planning

Sampling-Based Planners (SBPs) discretely sample the configuration space to find robot configurations within C_{free} to approximate the free space. A local planner then creates continuous connections between samples. Since only samples and connections are used, an explicit representation of the workspace is not required. The two dominant SBP paradigms are the Probabilistic Roadmap Method (PRM) [2] and Rapidly exploring Random Trees (RRT) [3] frameworks. Both algorithms are highly generic and can be customised in many ways; for example, different sampling techniques can be employed to improve performance and path quality [4].

The PRM approach is most often employed in multi-query planning, where multiple motion queries are performed in the same environment. PRM is composed of two stages: 1) the learning phase and 2) the query phase. In the learning phase a roadmap is created by generating samples and attempting to connect them to one another using a local planner. During the query phase, the start and goal configurations are added to the roadmap and a graph search algorithm is utilised to traverse the roadmap between the start and the goal [2]. Common variants include the Lazy PRM, where collision checking is delayed [5], or PRM*, where a distance metric is adjusted to converge to the optimal path [6].

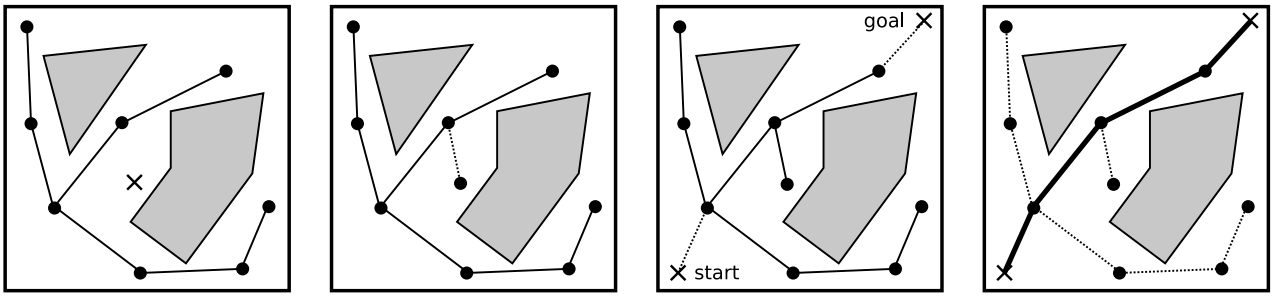
RRTs in contrast are generally used as a single query planner. RRTs grow a search tree from a start configuration by repeatedly sampling a configuration and attempting to grow a connection to it from the closest tree node. A key advantage of RRTs over PRMs is the relative ease of integrating motion parameters such as system dynamics [3]. Common variants include RRT-Connect which also grows a tree from the goal configuration to the start for efficiency [7], or an optimisation to find the shortest path by discarding cycles with a longer path [6]. Fig 1 shows a graphical illustration of the PRM and RRT algorithms.

B. Dynamic Motion Planning

The basic approach to the motion planning problem is to generate a path ahead of time. This assumes complete knowledge of the environment, such as that derived from a CAD model. Recent motion planning developments have

978-1-5090-2065-2/16/$31.00 ©2016 IEEE

(1) PRM Algorithm



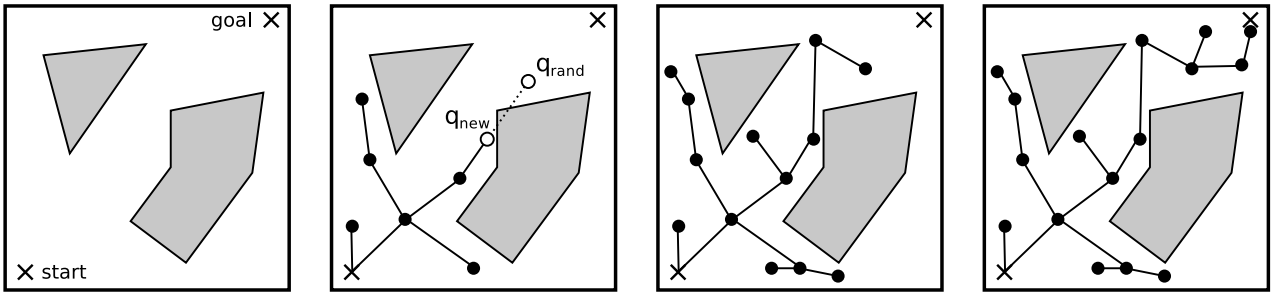
(a) The *learning* phase: a random sample, denoted by $\times$, is generated

(b) A local planner is used to connect the new sample to nearby roadmap vertices.

(c) The *query* phase: the start and goal configurations are added to the roadmap.

(d) A graph search algorithm is used to connect the start and goal through the roadmap.

(2) RRT Algorithm



(a) A tree is grown from the start configuration towards the goal.

(b) The planner generates a configuration q_{rand}, and grows from the nearest node towards it to create q_{new}.

(c) The tree rapidly explores the free space.

(d) The planner terminates when a node is close to the goal node. Common implementations will connect directly to the goal.

Fig. 1. Illustration of the (1) PRM and (2) RRT algorithms.

focused on cases where the workspace is not fully known or those that change over time. In such cases motion paths need to be solved (or adjusted) in real-time such that the solution is available while the current environment knowledge is valid. This is considered a more realistic application for motion planning [8]. [9] characterises the motion planning problem into four cases based on the obstacles being static or dynamic and the state of knowledge of the workspace (Table I). The cases that are considered to be a dynamic motion planning problem are shown in bold.

There are two primary approaches to adapting sampling-based planners for dynamic environments. The first is to modify the algorithms to re-use information to efficiently update motion paths when the environment changes. This can involve re-planning portions of the path, re-using computed information across steps, or adjusting generated paths rather than re-planning them completely. The second approach is to parameterise time, explicitly planning motions in configuration-time space. This introduces an additional dimension to the problem and makes the problem non-holonomic, as the robot cannot move back in time. There may also be additional constraints imposed on the path due to velocity and acceleration limits meaning some connection are infeasible.

A concept related to dynamic planning is real-time planning. This requires a robot to be capable of planning motions in time to react to dynamic environments whilst on the move. For real-time planning to be accomplished, the motion plan must be determined during an execution cycle [10]. To be considered real-time, Kunz et al. specifies re-planning operations to be completed within 200ms to mimic the reaction time of a human [11].

TABLE I
CASES REQUIRING DYNAMIC MOTION PLANNING FROM [9]

	Static Obstacles	Moving Obstacles
Known Environment	Case I	**Case II**
Partially Unknown	**Case III**	**Case IV**

II. PLANNING ALGORITHMS

This section presents SBP algorithms tailored for dynamic environments. These are grouped into three categories. Section II-A presents PRM based planners which use a roadmap internally. Section II-B shows RRT based planners which grow a tree either from q_{start} or q_{goal}. Section II-C details planners which combine aspects of multiple planning algorithms. Table II lists the approaches included and provides a brief description of their key features.

TABLE II

SUMMARY OF DYNAMIC PLANNING ALGORITHMS

Algorithm	Based On	Summary
Cell-based PRM (CPRM) [12]	PRM	Discretises configuration space into cells, and attempts to sample in cells which are likely to contain a solution. Cells close to a direct path and with disconnected components are prioritised. This heuristic allows for faster re-planning as it biases searching around the solution.
Dynamic Roadmap (DRM) [13]	PRM	Pre-computes a roadmap assuming a free environment, then discretises the workspace and maps workspace cells to roadmap edges. During online planning, edges which pass through obstructed workspace cells are invalidated.
Elastic Roadmap [14]	PRM	Combines aspects of planning and control - feedback controllers are used to connect roadmap vertices, and these controllers respond to environment changes. Vertices are maintained by associating them with workspace features which are then tracked.
Flexible Anytime Dynamic PRM (FADPRM) [15]	PRM	The workspace is segmented into zones of relative desirability. An A* search is done propagating from the goal state, and samples are generated near frontier nodes based on a priority heuristic. The path can then be continually improved and updated during execution.
g-Planner [16]	PRM	Highly parallel PRM variants which runs on the GPU. Uses parallelised Bounding Volume Hierarchy and sample generation. Local planning is done lazily, and the roadmap is searched in parallel to satisfy queries.
Hsu et al. [17]	PRM	Generates a tree-like PRM online in state $\times$ time space by sampling control inputs. The planner is given a time window to plan within. If the observed obstacles motion changes during execution, the planner then generates a new path.
Jaillet et al. [18]	PRM	Uses a pre-computed roadmap to attempt to answer question. If planning fails, a single-query planner (such as RRT) is invoked to update the roadmap.
Pomarlan et al. [19]	PRM	Pre-computes a roadmap assuming free space, and then uses Lazy PRM [5] to generate a path. When a vertex is invalidated, nearby vertices have their cost increased to push planning away from them. This process can also be run in parallel to execution for increased performance.
Reactive Deformation Roadmap (RDR) [20]	PRM	Builds a roadmap based on dynamic milestones and reactive links modeled as particles. Uses a physically inspired model in which milestones attract and obstacles repulse. Also adds and removes milestones and links to maintain connectivity.
Anytime Dynamic RRT (AD-RRT) [21]	RRT	Builds on DRRT, combining it with the behaviour of Anytime RRT which generates a possible suboptimal solution and then improves it. The algorithm alternates between the behaviour of the two component algorithms in a loop.
Closed Loop RRT (CL-RRT) [22]	RRT	Grows a tree an input space from the robot's current state, and then chooses the best trajectory at each time step. Executes this trajectory, and then re-roots the tree at the end of the trajectory for further exploration. Edges are re-validated when they are chosen as part of the trajectory.
Dynamic RRT (DRRT) [23]	RRT	Edges which intersect with an observed obstacle are invalidated. The tree is then trimmed by iteratively discarding children of invalid vertices. The plan is then repaired by growing the tree from the goal towards the current robot configuration.
Execution Extended RRT (ERRT) [24]	RRT	When a path is planned, states are added to a fixed size waypoint cache. If the path is invalidated, the path is re-planned and samples are probabilistically taken from the waypoint cache in addition to being randomly sampled.
Frazzoli et al. [25]	RRT	Similar to RRT, but attempts to connect new states to every existing node rather than just the closest. Additionally, a cost function assuming an obstacle-free environment is used as a distance metric. Nodes are checked for τ safety, ensuring at least τ time is available to re-plan during path execution.
Greedy, Incremental, Path-directed (GRIP) [26]	RRT	Grows a tree ahead of execution, and trims it as the robot moves and edges are invalidated. Deterministically propagates samples from existing edges. Also discards states which inevitably lead to collision.
Lazy Reconfiguration Forest (LRF) [27]	RRT	Combines the approaches of LRF and DRRT. A forest of trees is maintained during task execution, similar to LRF. However, only edges along the currently executing path are validated, similar to DRRT.
Multipartite RRT (MP-RRT) [28]	RRT	Combines concepts from other approaches: sampling is biased towards previously valid states as in ERRT and discards invalidated tree sections as in DRRT. However, discarded branches are maintained in a cache which is then sampled from.
Partial Motion Planner (PMP) [10]	RRT	Assigns fixed time windows for planning and execution. Grows a tree from the current robot state for a time δ, and uses the best candidate path when planning is done, invalidating nodes that will inevitably lead to collision.
Reconfigurable Random Forests (RRF) [29]	RRT	Invalidated tree edges are removed, splitting trees and forming a set of disconnected trees (a forest). Trees are then connected together, and pruned if they grow too dense.
RRT$^{\mathrm{X}}$ [30]	RRT	An asymptotically optimal re-planning algorithm, generates paths which are continually valid and optimal in the current static environment. A tree rooted at the goal is rewired when obstacles are detected.

A. PRM Based Algorithms

The PRM approach has been extended in several ways to efficiently solve dynamic environment problems. If the motion of obstacles is known or predicted ahead of time, planning can be done in configuration-time space [17]. For some applications re-planning can be done quickly enough if obstacles change unexpectedly. [31] creates robust roadmaps which, given a possible set of obstacle locations, are probabilistically complete. The strength of this approach is that processing is performed in the learning phase, rather than the query phase. The rest of the approaches in this section do not assume knowledge of obstacle locations.

Dynamic Roadmaps (DRM), introduced by Leven and Hutchinson [13], [32], is a method that has been commonly used. The pre-computed roadmap is mapped to a discretisation of the robot's workspace. Kallman and Mataric [33] showed that a simplified DRM which used reference counting rather than full collision checking can outperform single-query planners.

Heuristic optimisations to DRM were developed by Kunz et al. [11] which allowed them to demonstrate planning in under 100ms. Computationally efficient data structures and distance metrics were used when adding starts and goals to the roadmap and lazy collision checking was integrated with the graph search to avoid backtracking. They also developed a heuristic which allowed them to use A* graph search with DRM in contrast to [13] where depth and breadth-first search are used.

An approach combining DRM with Lazy PRM [5] style collision checking was developed by Liu et al. [34]. The process of mapping the workspace from configuration space was shortened by only mapping roadmap vertices to the workspace, rather than also including edges. In this case, the mapping process took only seconds or minutes, rather than the hours or days in the original DRM approach. Edges are only fully validated when they are part of a candidate path. Results showed that online planning times were comparable to the original DRM, and that the algorithm could also handle deforming robots.

DRM has also been combined with the Dynamic Bridge Builder algorithm by Liu et. al [35] to enhance narrow passage capability. Computational speed is also improved by only considering moving obstacles within the reachable area of the robot. Parallel DRM (PDRM) was shown in [36], where the roadmap was built using motion primitives and planning queries were performed in approximately 50ms.

Another approach based on pre-computing an obstacle free roadmap is proposed by Pomarlan and Sucan [19]. No workspace mapping is created, instead the standard Lazy PRM [5] planner is used. When a vertex is invalidated during planning, nearby vertices have their cost increased. This *pushes* the planner away from failed vertices, sacrificing path length for a higher probability of success. This is a simple and efficient technique and can be applied to many planners as well as being coupled with a roadmap repair process. One issue is that edges are not considered, so edges that pass near failed vertices will not have their costs increased.

Other PRM-based approaches bias sampling rather than pre-computing a roadmap. Flexible Anytime Dynamic PRM (FADPRM) [15] is analogous to the anytime AD* extension to the A* graph search algorithm [37] in that motion plans can be continually improved during execution. A search is done from the goal state. A priority queue of frontier nodes to sample near is maintained based on the relative density of nearby nodes, the distance to goal and workspace desirability. Experiments showed that initial plans were slow to generate, but subsequent changes effectively exploited existing roadmap information.

Similarly, Cell-based PRM [12] uses heuristics to guide sampling in order to speed up re-planning queries. The configuration space is decomposed into cells and sampling is biased towards cells along the ideal path and those with many disconnected components. This sampling heuristic was initially slower than a standard PRM, but subsequent queries were more rapidly computed.

A third approach is to use control strategies to deform the roadmap in response to workspace changes. Elastic Roadmaps [14] use workspace information and control techniques to allow planning at interactive rates. Roadmaps are generated in the workspace with feedback controllers used as local planners. Reactive Deformation Roadmaps (RDR) [20] model vertices as particles and edges as a series of particles. These particles are then controlled using a Newtonian physics model where vertices attract and obstacles repel. These approaches exploit control techniques so the roadmap is adjusted to changes, rather than requiring re-planning.

In summary, the three main PRM-based techniques are pre-computation, sampling biasing and roadmap deformation. Pre-computed roadmaps can either include cycles to be robust to obstacles, or exploit features such as a workspace mapping to allow for quick re-planning. Alternative, sampling can be biased to repair the roadmap when the workspace changes, or the roadmap itself can be deformed.

B. RRT Based Algorithms

RRTs are traditionally used as single-query planners – they rapidly grow a tree from the start configuration, and as such are problem-specific. This makes them inherently useful for dynamic planning where the start configuration and workspace can rapidly change.

Execution Extended RRT (ERRT) [24] inserts all states along a planned path into a *waypoint cache*, which is a fixed size array. When generating samples, states are taken from the waypoint cache with some probability rather than using random generation. The algorithm exploits temporal coherence, and works best in situations where workspace changes are small and states can be re-used. An adaptive distance cost metric is also introduced that favours generation of smaller paths for repeated successful planning queries, balancing path quality with exploration. The waypoint cache concept is re-used in a number of other algorithms presented.

Dynamic RRT (DRRT) [23] also takes advantage of temporal coherence. Rather than reconstructing the tree as in

[24], edges that are dynamically obstructed are invalidated. A trimming process then iteratively discards any states which have invalid parents, thus discarding invalid branches. If the tree no longer reaches the goal, the tree is regrown. An additional enhancement is to bias sampling towards configuration space areas affected by changes. [23] found DRRT outperformed ERRT by a factor of 5 in terms of time.

Multipartite RRT (MP-RRT) [28] combines aspects of both ERRT and DRRT. When edges are invalidated, disconnected subtrees are inserted into a cache. When sampling, the root node of disconnected subtrees are selected probabilistically. MP-RRT has been shown to outperform both ERRT and DRRT [28].

Anytime Dynamic RRT (AD-RRT) [21] also builds upon DRRT, combining techniques from Anytime RRTs [38]. Anytime RRTs can incorporate cost into their planning process, and generate a series of RRTs where each is guaranteed to improve on the cost of the previous one. DRRT is used when workspace changes are observed, and Anytime RRTs optimise the path at other times. This combination allows trimming and optimisation costs to be minimised.

Similarly to DRRT, Reconfigurable Random Forests (RRF) [29] identifies invalidated edges and discards them. However, instead of trimming tree branches a forest is created consisting of any disconnected trees, similar to MP-RRT. The forest is then maintained by reconnecting branches and pruning when it grows large. Lazy Reconfiguration Forest [27] extends RRF by only validating edges along the currently executing path, rather than maintaining the entire forest.

Another approach is to more explicitly take into account the plan-execute cycle. This works well with RRTs, as the tree can be re-rooted as the robot moves. The Partial Motion Planning formulation [10], nominally based on RRTs, allocates fixed time windows for planning to be completed before a candidate path is executed. Frazolli. et al [25] ensure that a generated path allocates at least a certain time for an alternative plan to be generated if required.

Closed Looped RRT (CL-RRT) [22] uses the PMP approach, re-rooting a tree at each time step and generating the best path possible in the allocated time. Greedy, Incremental, Path-directed (GRIP) [26] also grows a tree ahead of the robot's current state. The tree is grown by splitting and propagating existing edges, and states which will inevitably lead to collision are discarded.

As can be seen, the RRT method adapts well to repeated queries by retaining information across planning requests. Different algorithms work more efficiently depending on the amount of workspace change expected. RRTs also allow for tracking the robot execution, as the tree can be re-rooted to constantly move with the robot.

C. Hybrid Methods

Hybrid methods combining multiple planning techniques have also been applied to dynamic motion planning. One technique is to combine a multiple-query planner for initial motion plans along with a single-query planner for rapid roadmap repairs. This approach was demonstrated by Jaillet et al. [18], which initially used a static roadmap to plan motions. If a valid motion cannot be generated, blocked sections are identified and an RRT-based planner is invoked to repair the roadmap. If this too fails, the roadmap is reinforced with the addition of new vertices. [39] combined a PRM with the Anytime D* [37] algorithm to achieve a similar result.

Other hybrid planners use global sampling-based planning for high-level motion plans combined with local planners that react to environmental changes. This allows the global property of the sampling-based planner to be maintained, while gaining the rapid reactionary planning of the local planner. Sanchez et al. [40] implemented a reactive lazy PRM planner that models a Deformable Virtual Zone (DVZ) around the robot. When dynamic obstacles impeach on the virtual zone, control commands are generated to move away from them. The planner then uses re-connection or re-planning to move back to the roadmap.

Hybrid methods allow aspects of appropriate algorithms to be combined to enable dynamic planning. Similarly to as demonstrated in Section II-B, it is often combining existing techniques that leads to dramatic improvements rather than developing new algorithms.

III. Implementation Strategies

This sections briefly presents common implementation techniques that can be applied to several of the presented algorithms. The techniques are roadmap pre-computation and workspace mapping, parallelisation, and cloud computing.

A. Roadmap Pre-Computation and Workspace Mapping

Increasing processing capability and memory space means that extensive pre-computation is an increasingly attractive option for motion planning. Several of the PRM-based methods presented in Section II-A use a dense pre-computed roadmap, optionally mapped to a discretised workspace.

Choosing the density of a pre-computed roadmap is critical. A sparse roadmap will allow for quick planning operations, but may lead to low quality paths or planning failures due to inadequate coverage [19]. Planning roadmaps generally do not include cycles, but in dynamic environments they may be useful to allow alternative plans if edges are invalidated. [18] uses reinforcement which can introduce cycles and [41] found that introducing cycles did not significantly affect processing time.

Although motion planning operations are generally performed in a robots configuration space, obstacles are observed in the robot's workspace. The pre-computed roadmap can be mapped to the robot's workspace for rapid change identification [32], [33]. The workspace is most commonly decomposed through a straightforward uniform decomposition. Leven et al. [13] proposed an approach similar to run-length encoding to compress the mapping by exploiting spatial coherency. Computation time is minimised by using reference points on workspace bodies rather than complex meshes, and by using a binary search-like method for performing collision and mapping them to workspace cells [11].

B. Parallelisation and Cloud Computing

The move towards multi-core, distributed, and Graphics Processing Unit (GPU) based computing means that parallel algorithm variants are capable of running significantly more quickly than serial variants. One of the earliest parallelisation techniques proposed was [42], in which the C-space of the first three joints of a six-axis serial link robot are discretised and the remaining three joints are planned in parallel. More recently, GPU-optimised variants of existing planners have been developed.

For example, the g-Planner family of planners showed that a GPU-optimised version of the PRM and Lazy-PRM algorithms can run orders of magnitude faster than CPU-based planners [16]. The GPU has also been used to perform collision checking operations, which is generally the most time consuming part of the SBP workflow, allowing for dramatic performance improvement [43].

Distributed and cloud computing also provides another avenue to parallelise computation and allow for expensive pre-computed knowledge to be shared. A synergistic roadmap pre-computation scheme proposed by Bekris et al. [44] uses cloud hardware for pre-computation of a high quality roadmap, which is then shared with a local computer for time-sensitive planning operations.

IV. Conclusion

Sampling-based planning is an effective method to solve high-dimensional motion planning problems in known environments. The basic PRM and RRT algorithms are simple in concept yet can be extended to solve complex scenarios, such as a dynamic environment. This paper has reviewed and contrasted a number of extensions which have been used to make these approaches usable in dynamic and changing environments, which is a key requirement for real-world robotics.

Both PRM and RRT-based algorithms have been demonstrated. PRM-based dynamic algorithms tend to focus around pre-computation of a robust roadmap, and then either deforming this roadmap or rapidly invalidating it through a workspace mapping. RRT-based algorithms maintain state between planning queries for efficiency, and also maintain and attempt to repair tree branches when they are partially invalidated. Combining PRMs and RRTs, as well as sampling based planners and local planners, has also been shown to be a useful technique.

A key implementation technique that is being exploited is the continual growth in computing power, both in single-processor terms and the availability of distributed computing. Dynamic planning algorithms specifically designed to exploit parallel architectures or GPUs are more performant than traditional serial algorithms. The ability to perform expensive pre-computation such as explicit discretisation and mapping is now straightforward due to cloud computing resources.

V. Acknowledgements

Andrew Short is supported by the Australian Postgraduate Award (APA), the University of Wollongong Global Challenges program and the ARC Research Training Centre for Naval Design and Manufacture.

References

[1] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Access, IEEE*, vol. 2, pp. 56–77, 2014.

[2] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[3] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.

[4] J. Polden, Z. Pan, and N. Larkin, "Path planning for industrial robots; lazy significant edge algorithm (lsea)," in *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*. IEEE, 2013, pp. 979–984.

[5] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 521–528.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[8] K. I. Tsianos, I. A. Sucan, and L. E. Kavraki, "Sampling-based robot motion planning: Towards realistic applications," *Computer Science Review*, vol. 1, no. 1, pp. 2–11, 2007.

[9] K. Fujimura, *Motion planning in dynamic environments*. Springer Science & Business Media, 2012.

[10] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Integrating perception and planning for autonomous navigation of urban vehicles," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 98–104.

[11] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," in *Intelligent Robots and Systems, 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5906–5911.

[12] K. Klasing, D. Wollherr, and M. Buss, "Cell-based probabilistic roadmaps (cprm) for efficient path planning in large environments," in *Proc. of the 2007 Int. Conf. on Advanced Robotics*, 2007.

[13] P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," *Algorithmic and Computational Robotics: New Directions: The Fourth International Workshop on the Algorithmic Foundations of Robotics*, pp. 363–376, 2000.

[14] Y. Yang and O. Brock, "Elastic roadmapsmotion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[15] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime dynamic path-planning with flexible probabilistic roadmaps," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2372–2377.

[16] J. Pan, C. Lauterbach, and D. Manocha, "g-planner: Real-time motion planning and global navigation using gpus." in *AAAI*, 2010.

[17] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[18] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *Intelligent Robots and Systems, 2004 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2004, pp. 1606–1611.

[19] M. Pomarlan and I. A. Sucan, "Motion planning for manipulators in dynamically changing environments using real-time mapping of free workspace," in *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on*. IEEE, 2013, pp. 483–487.

[20] R. Gayle, A. Sud, M. C. Lin, and D. Manocha, "Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3777–3783.

[21] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1310–1315.

[22] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[23] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1243–1248.

[24] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2383–2388.

[25] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[26] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 704–710.

[27] R. Gayle, K. R. Klingler, and P. G. Xavier, "Lazy reconfiguration forest (lrf)-an approach for motion planning with multiple tasks in dynamic environments," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1316–1323.

[28] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1603–1609.

[29] T.-Y. Li and Y.-C. Shie, "An incremental learning approach to motion planning with roadmap management," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 3411–3416.

[30] M. Otte and E. Frazzoli, "RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 461–478.

[31] J. P. Van den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars, "Creating robust roadmaps for motion planning in changing environments," in *Intelligent Robots and Systems, 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 1053–1059.

[32] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[33] M. Kallman and M. Matarić, "Motion planning using dynamic roadmaps," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4399–4404.

[34] H. Liu, X. Deng, H. Zha, and D. Ding, "A path planner in changing environments by using wc nodes mapping coupled with lazy edges evaluation," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 4078–4083.

[35] H. Liu, Y. Li, H. Wen, J. Xia, and T. Chu, "Hierarchical roadmap based rapid path planning for high-dof mobile manipulators in complex environments," in *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*. IEEE, 2009, pp. 189–195.

[36] H. Schumann-Olsen, M. Bakken, Ø. H. Holhjem, and P. Risholm, "Parallel dynamic roadmaps for real-time motion planning in complex dynamic scenes," in *3rd Workshop on Robots in Clutter*, 2014.

[37] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm." in *ICAPS*, 2005, pp. 262–271.

[38] D. Ferguson and A. Stentz, "Anytime RRTs," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5369–5375.

[39] J. Van Den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2366–2371.

[40] A. Sánchez, R. Cuautle, R. Zapata, and M. Osorio, "A reactive lazy prm approach for nonholonomic motion planning," in *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*. Springer, 2006, pp. 542–551.

[41] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 446–452.

[42] T. Lozano-Pérez and P. A. Donnell, "Parallel robot motion planning," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1000–1007.

[43] J. Pan and D. Manocha, "Gpu-based parallel collision detection for fast motion planning," *The International Journal of Robotics Research*, p. 0278364911429335, 2011.

[44] K. Bekris, R. Shome, A. Krontiris, and A. Dobson, "Cloud automation: Precomputing roadmaps for flexible manipulation," *Robotics & Automation Magazine, IEEE*, vol. 22, no. 2, pp. 41–50, 2015.