Path Planning Using Learned Constraints and Preferences

Gregory Dudek and Saul Simhon

Centre for Intelligent Machines, McGill University
3480 University St, Montréal, Québec, Canada H3A 2A7

***Abstract*— In this paper we present a novel method for robot path planning based on learning motion patterns. A motion pattern is defined as the path that results from applying a set of probabilistic constraints to a "raw" input path. For example, a user can sketch an approximate path for a robot without considered issues such as bounded radius of curvature and our system would then elaborate it to include such a constraint. In our approach, the constraints that generate a path are learned by capturing the statistical properties of a set of training examples using supervised learning. Each training example consists of a pair of paths: an unconstrained (raw) path and an associated preferred path. Using a Hidden Markov Model in combination with multi-scale methods, we compute a probability distribution for successive path segments as a function of their context within the path and the raw path that guides them. This learned distribution is then used to synthesize a preferred path from an arbitrary input path by choosing some mixture of the training set biases that produce the maximum likelihood estimate. We present our method and applications for robot control and non-holonomic path planning.**

I. Introduction

Traditionally, motion constraints have been represented by analytic methods that constrain the differential geometry of the set of admissible paths. Path planning typically entails solving an optimization problem with respect to these constraint equations. In contrast, this paper presents a radically different approach to path planning. Constraints (or preferences) are expressed in terms of a set of examples that illustrate how the robot is permitted to move. Further, these examples indicate how to elaborate an input path from a user or high-level planner (which is typically not acceptable in itself) into a suitable acceptable output path. Informally, the examples say: "if a user asks you to do something like *this* than what you should actually perform is a maneuver like *that*".

Traditional constraint equations for motion control are complex relations that typically model the dynamics of a mobile robot based on its mechanical design. Our method can be used to simulate such constraints without having to explicitly model them. The goal is to learn from examples of constrained motions to properly generate novel paths.

Motion constraints are not only used for modeling a robot's mechanical configuration. In some applications, equations are constructed to model task specific motion requirements, such as a sweeping pattern for full floor coverage or a suitable behavior to scan the environment using a narrow-beam sensor. Specialized paths also occur various specialized contexts; in the classic 1979 film "The In-Laws" Peter Falk instructs Alan Arkin to run along a "serpentine" path while going elsewhere – our system could readily accommodate such a preference bias as well. Additionally, in applications such as obstacle avoidance, motions are not only related to the robot's pose but are also a function of the perceived environment. In all of these examples, the underlying core problem consists of finding a valid transformation between two components: 1) the idealized "raw" path that directs the robot to a goal without taking into account certain preferences or constraints, and 2) the refined path that attempts to reach the goal while also satisfying the system constraints.

Whatever the constraints, expressing them in a suitable formal framework is often challenging. Further, the processes of finding allowable solutions can be costly, particularly since the solution techniques are often engineered for a specific context. In our work, we develop a method to implicitly learn motion constraints. Given a set of sample *motion patterns*, each with an associated *control path*, the algorithm captures statistical properties over the length of the path and configures a Hidden Markov Model. Given a new input control path and the configured HMM, the algorithm generates a new path that is statistically consistent with the learned patterns, enforcing the desired local constraints. As such, the algorithm can be applied over a variety of training examples to generate a rich set of paths without having to explicitly model the constraints.

II. Related Work

Path planning has been extensively examined by many authors. One of the key ideas in the area is the notion of path planning under non-holonomic constraints, specifically path planning using a bound on the radius of curvature on the vehicle [1]. Notable work in the field includes the landmark results of Dubins [2] and Reeds and Shepp [3] on optimal trajectories. Much of this work deals with the quest for an optimal path (or trajectory) under a motion constraint which is expressed analytically (for example a derivative constraint). Prevalent solution techniques include analytic solutions (or expressions regarding their bounds), search methods that seek to optimize a path, and planners that start with a path of one form and seek to refine it.

In particular, a classic approach to the application of non-holonomic constraints is to find an (optimal) unconstrained solution and then apply recursive constrained path refinement to the sub-regions to achieve an admissible plan [1]. This is also typical of probabilistic motion planning methods [4], [5]. Similarly, jerky paths are sometimes smoothed using energy minimization methods [6], [7].

This work shares that common spirit in that it takes an initial path as input and produces a refined path as its result. While traditional methods such as those cited above typically accomplish path refinement based on highly specialized constraints, typically in the domain of differential geometry, our methods learns from examples of acceptable paths. That is, the significant constraints or preferences are indicated by showing the appropriate refinements that should be applied in specific cases.

This idea of learning to generalize specific examples to a broad ensemble of cases is, of course, the crux of classical machine learning [8]. Learning using Markov models is a longstanding classic research area, although to our knowledge it has never been applied to problems like this one. Likewise, although there has been some prior work on the relationship between learning and planning, most of this has dealt with more traditional plan formulation problems [9] or on learning suitable cues that control or determine plan synthesis or execution [10], [11].

III. Learning Motion Patterns

Our objective is to learn attributes from training examples in order to synthesize a constrained path given an arbitrary unconstrained path. Figure 1 shows some training examples that capture a non-holonomic constraint: a bound on the maximum radius of curvature. The examples show smooth right angle turns, wide D-shaped turns, narrow U-shaped turns and an example of a parallel-parking type motion with a direction reversal (note that at every path along these paths the orientation of the vehicle is also represented but not always shown in the figure). In each example, both the desired path and the associated control path are displayed.

We present a method to represents those significant attributes using a Hidden Markov Model. A Hidden Markov Model encodes the dependencies of successive elements of a set of *hidden* states along with their relationship to *observable* states. It is typically used in cases where a set of states, that exhibit the Markov property, are not directly measurable but only their effect is visible through other observable states. Formally, a Hidden Markov Model Λ is defined as follows:

$$\Lambda = \{M, B, \pi\} \tag{1}$$

where M is the transition matrix with transition probabilities of the hidden states, $p\{h_i(t) \mid h_j(t-1)\}$, B is the confusion matrix containing the probability that a hidden state h_j generates an observation o_i, $p\{o_i(t) \mid h_j(t)\}$, and π is the initial distribution of the hidden states.

There is an abundance of literature on Hidden Markov Models and the domain is frequently decomposed into 3 critical sub-problems:

- Evaluation, where the likelihood of an HMM is evaluated for a sequence of observations, $p\{o \mid \Lambda\}$.
- Decoding, where the maximum likelihood sequence of hidden states is predicted for a given HMM and an observation sequence, $\max_h \; p\{h \mid o, \Lambda\}$.
- Learning, where the transition probabilities, the confusion matrix and the initial distribution that best fit an observed set of examples are estimated.

Given only the observations, learning is most commonly performed by algorithms such as the Baum-Welch algorithm or generalized Expectation-Maximization methods. In our application, we have direct access to both the hidden and observable states. They consist of sample points from the desired motion patterns and the associated control paths respectively, which are readily available. Therefore, we can estimate an HMM by the the statistics of the training data, calculating probabilities of successive elements of the desired motions and their relationship to the control paths.

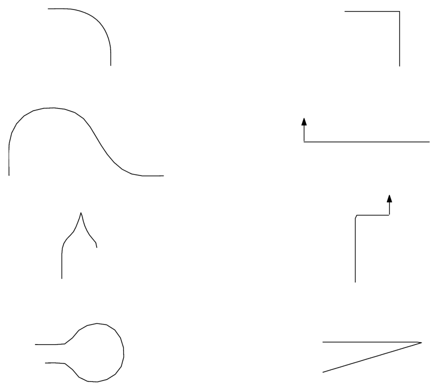

Fig. 1. Samples of a training set simulating non-holonomic motions. Paths on the left display the constrained motions while paths on the right display the associated unconstrained control paths. Where specified, arrows indicate additional constraints on the direction of motion to account for the orientation of the robot. The full set consists of the above at several orientations.

A. Hidden Paths

We represent a path by a curve over 2D space parametrized by the arc-length. Let α represent a parametric curve $(x(t), y(t))$ where t is the arc-length of the curve from $0 <= t <= T$. Since we can, in principle, encode a function using only its derivations, we assume our paths are suitably normalized and encode them as a discreet succession of tangent angles $\theta(t)$.

Consider a stochastic process Δ as the source for a family of paths. As such, each a path is considered to be a random signal with characteristics described by the probability density function of the process. Let α denote the curve representing the constrained motion pattern and $\theta(t)$ as the tangent angles of that path. We assume that the sequence of samples $\theta(t)$ from all constrained motion patterns exhibit an n^{th} order Markov property, i.e. a Markov Process:

$$p\{\theta(t+1) \mid \theta(t), \theta(t-1), \ldots, \theta(t-n+1)\} =$$
$$p\{\theta(t+1) \mid \theta(t), \theta(t-1), \ldots, \theta(0)\}$$

This *locality* condition states that information from recent samples is sufficient to predict the next sample point. Further, the dependency is considered to be invariant, where relationships between successive points are *stationary* with respect to the arc-length.

Sample points of the motion patterns are represented by hidden states in the HMM. Given an ensemble of training examples, we estimate the transition probabilities by the statistics of successive elements in the set and construct the transition matrix M where:

$$P_\theta(t+1) = M\ P_\theta(t) \tag{2}$$

The transition matrix propagates the information embedded in the probability distribution $P_\theta(t)$ to predict the next distribution $P_\theta(t+1)$. We assume a uniform initial probability distribution $\pi = P_\theta(0)$, providing equal likelihoods to all paths at time zero.

B. Observable Paths

Sample points of the control paths are represented by observable states in the HMM, which characterizes the relationship to the constrained motion patterns. Based on this relationship, an input path can condition the distribution in equation 2 and bias the synthesis according to the prescribed characteristics. Because the input path can be any arbitrary shape, we assume that samples of the control paths are independent. For all t and k in the domain:

$$p\{\theta(t) \mid \theta(k)\} = p\{\theta(t)\}$$

That is, previous points generally do not provide information on what the next point may be. This assumption adheres to the HMM condition that the observable state sequence is independent over time.

Let β denote the curve representing the associated control path and $\phi(t)$ as the tangent angles of that path. Then:

$$\alpha = \Psi\ \beta \tag{3}$$

where Ψ is some mapping that transforms the control path to the constrained motion pattern. The mapping in essence encodes the constraint relationship between the coupled pair. Given a normalized ensemble of constrained/unconstrained curves, we estimate the probabilities of the confusion matrix B from the statistics of associated sample points $(\theta(t), \phi(t))$ and form the following relation:

$$P_\phi(t) = B\ P_\theta(t) \tag{4}$$

where the elements of the confusion matrix are the conditional probabilities $p(\phi_i|\theta_j)$ for all states i and j. This is analogous to the inverse relation of the mapping Ψ in equation 3. However, using Bayes law, one can show that solving the decoding problem for a HMM in a maximum likelihood sense is analogous to solving for the desired transformation Ψ. (It is in-essence solving the inverse problem in a maximum likelihood sense.)

IV. Synthesis

Given a set of observations and an HMM trained with a family of path patters, we generate a new path pattern by solving for the maximum likelihood hidden state sequence

$$\max_{\theta_i \cdots \theta_n} p\{\theta(0), \theta(1), \ldots, \theta(T) \mid \phi(0), \phi(1), \ldots, \phi(T), \Lambda\}$$
$$or$$
$$\max_{\alpha} p\{\alpha \mid \beta, \Lambda\} \tag{5}$$

Also known as the decoding problem, the above problem can be solved iteratively. At each time interval, we propagate the underlying probability distribution as in equation 2 and maintain states with maximum consistency across successive elements. The resulting distribution is then conditioned by the current observation as in equation 4. (This method is analogous to the *Viterbi algorithm*.) We iterate up to time T to produce a sequence of probability distributions $\{P_\theta(0), P_\theta(1), \ldots, P_\theta(T)\}$. To instantiate a path, we can select states with maximum probability from each distribution. However, independently selecting states in a greedy fashion can result to an inconsistent sequence, it may break the continuity of valid links between successive elements. Rather, we instantiate the state with maximum probability at time T and then backtrack by choosing the previous most likely state that would generate the current one. Backtracking is essential for generating a consistent path as not only does it consider the links between successive states, but also propagates future information back to earlier points. Figure 7 shows an example path manually drawn and the resulting generated path.

A. Multi-Dimensional State Space

Implementation of a first order Markov Model is generally achievable by storing the transition probabilities in a memory array. However, preliminary empirical results showed that for most training examples, a first order Markov Model does not capture enough information to properly generate the paths. Further, higher order Markov Models increase the state space exponentially and storage of a transition matrix is not practical. To address this issue, we do not explicitly compute and store the transition matrix, rather, we only maintain a list of candidate states with strictly positive probabilities. The algorithm then performs a search comparing the training set and candidate list. When a match occurs, the probability of the proceeding state is calculated and added to the list of candidates for the next sample point.

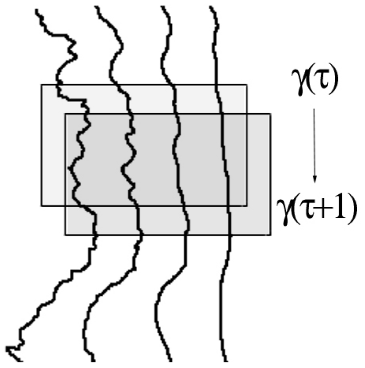

Fig. 2. Each multi-dimensional state consists of points within a fixed size window spanning the multi-scale curve model. When a match for state $\gamma(t)$ is found, the next state $\gamma(t+1)$ is added to the list.

To store high-order information, each state is represented by a multi-dimensional element γ_τ where each dimension corresponds to a sample point further in history. Without exaggerating the dimensionality of the state space, additional history can be attained by sampling over a multi-scale representation of the curve $\gamma_{\tau,s}$. A curve is filtered several times to produce lower scale versions. A single point on a lower scale curve represents a summary for the region of the high scale curve (see figure 2). Figure 3 shows an example where higher scale structures are important to capture. It is easy to see how the first order assumption does not capture enough information to generate the pattern while a synthesis using higher order states produces more consistent results.

Given the flexibility of the multi-dimensional model, additional relevant attributes that are required to further constrain the system can be easily incorporated. For example, we include an additional dimension to provide control over the *direction of motion* of the robot. Normally, the direction is implicitly defined in the input path, however,

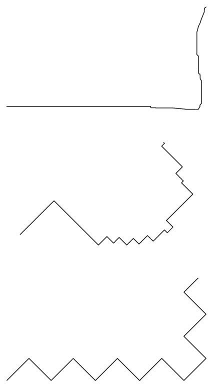

Fig. 3. Generating a sweeping style path pattern. Training data consists of the zig-zag patterns at several directions, each associated to straight line segments. The top curve shows the input path. The middle curve shows the synthesized path using a first order assumption. The bottom curve shows the resulting path using higher order states.

there are cases where we may need to further constrain the system in directions that are not necessarily along the path. For example, this can be used to provide control to align the robot axis orientation at particular points on the path. The training data must also contain this additional parameter in order to define the preferred behavior given both the path and the direction (see figure 1. Figure 4 shows an example where we condition the synthesis over various directions of motion in order to align the orientations of the robot's axis. At first, the initial direction of motion points vertically while the path progresses horizontally. The robot performs a D-shape turn for proper alignment. Later along the curve, the robot performs several other turns in order to align to the other specified directions.

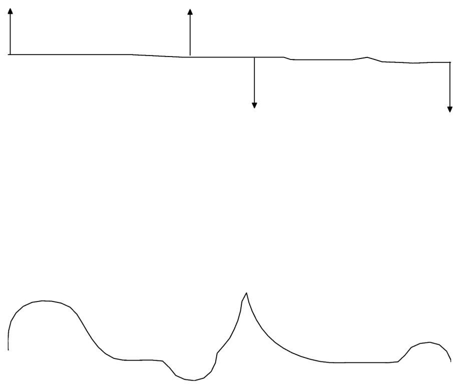

Fig. 4. The above was generated using the training example for non-holonomic constraints. The input path (top) is a hand-drawn path with several desired directions of motion (shown by arrows). The resulting path (bottom) consists of parts of a D-shape turn, a U-shape turn and a parallel parking style motion in order to end up in the right motion directions at the corresponding points.

B. State Blurring

The input path used for conditioning may not provide exact matches to the training data. The path may be generated by some noisy path planner or even manually drawn by a human operator. Further, even with perfect inputs, quantization errors are likely to occur in both the hidden states and the observations. Such errors can abruptly terminate the synthesis by conditioning or propagating all probabilities to zero. Blurring the probability matrices, or synonymously the probability vector, avoids this issue. The probability distribution is modeled as Gaussian mixture over the state space. The probability for state γ_i is given by:

$$p(\gamma_i) = \frac{1}{N} \sum_j p(\gamma_j) e^{\frac{-(\gamma_i - \gamma_j)^2}{v^2}} \tag{6}$$

For the multi-scale representation, the goodness of the match $(\gamma_i - \gamma_j)$ is based on a weighted difference over the scales. Such a blur may result in too many matches where every combination of states will produce strictly positive probabilities. This causes computational complexities and high dimensional models may not be solved in practical time. Therefore, we threshold over the tail of the Gaussian and normalize.

C. Coherency Measures

Since a Stationary Markov Process is assumed, there is no sense of progression or continuity of paths along the arc-length. The search is performed irrespective of the parametric position, choosing matches arbitrarily along the path. Further, the synthesis may get *stuck* at a state or a cycle through small set of states (know as absorbing states or irreducible communication classes). In such a situation, the propagated probabilities will model disjoint and self contained distributions. Conditioning over the observables and using a multi-scale model reduces the chance of this occurrence over long intervals. However, due to the nature of the training set, where there are many line segments that have few distinct features, such situations still often occurs. To address this issue, we define a measure for *coherency over arc-length* as a measure of the number of out-of-sequence states in a synthetic curve. To bias the synthesis for more coherent paths, we enforce a penalty on matches that are out-of-sequence. The probability is penalized by a factor of τ to help promote more coherent path.

While some degree of divergence is necessary to fulfill the desired motion pattern, we wish to avoid situations where the generated curve diverges too much from the input curve. Since the state space only represents the tangent angles as a function of arc-length, there is no indication of how close the generated curve is to the input curve. Therefore, we define a measure for *spatial coherency to input* as a measure of the average distance between the input curve and the generated curve over Cartesian space. To generate more spatially coherent paths, we include a magnetic force that biases the distributions to prefer points that are closer. At each sample point t the probability is updated by:

$$p(\gamma_i) = \frac{p(\gamma_i)}{N(1 + kd^2)} \tag{7}$$

where d is the distance between the input sample point and the resulting sample point generated by the maximum likelihood path up to and including the candidate state, k is the influence factor and N is a normalization constant. Figure 5 shows an example comparing generated paths with and without the coherency conditions. (One can also formulate these measures as regularization terms in a variational calculus problem minimizing entropy.)

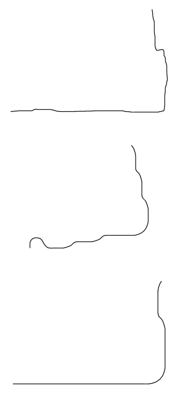

Fig. 5. The above example was generated using the training example for non-holonomic constraints. The input path (top) is a hand-drawn curve. Below it (middle) shows the generated path without any coherence factors and the bottom curve shows the generated path taking into account both spatial and arc-length coherence.

V. Results

Experiments were performed using path styles for sweep patterns, curled patterns and bounded turning radius patterns. The input paths are arbitrary paths hand drawn by a user. When the direction of motion is not specified by the input curve, it is taken along the tangent of that path. All experiments were executed on a Linux PC with a 1GHz Pentium IV processor and 1GB of RAM. The results were generated in real time.

Figure 6 shows examples where a single input path is used to generate several path styles. It can be seen how the resulting paths form analogies to the input path with respect to the learned path styles. Each resulting pattern maintains the local consistency while following the general direction of the input.

Figure 7 shows an example synthesis with non-holonomic constraints. The generated path follows a

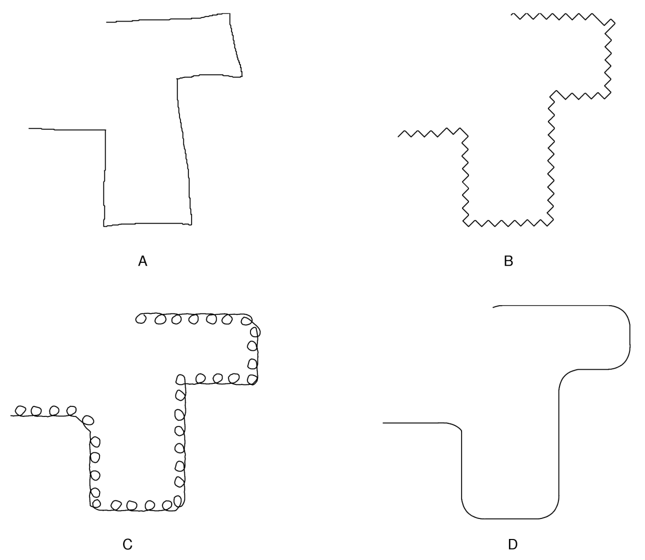

Fig. 6. The above example shows the input path (A) and the synthesized paths (B,C,D) using various training styles. The three training sets consist of a zig-zag patter for a sweep motion, a curl pattern for a narrow-beam sensor scan and bounded radius of curvature pattern.

smooth curve outlining the overall shape of the noisy input path. It can bee seen that the bottom right turn was an extended loop about the corner rather than the typical smoothed out turn. This was due to backtracking effects where the proceeding segment consisted of second turn, immediately after the first. The limited room for turning forced the path to extend around the corner. Figure 8 shows another example with non-holonimic constraints. The input curve restricts the initial direction of motion as displayed by the arrow. This results in the motion pattern with a direction reversal at point (B).

VI. Future Work

We plan on extending this work to take into account obstacles. Conditioning the probabilities over some occupancy grid can be a possible direction of work. Further, we wish to investigate a method that would perform synthesis computations using measurements over local reference frames. This can help avoid the requirement that training data must span all the desired orientations. In addition, another direction for future work is to examine applications of the method using related multiple curve signals, such as control for multi-robot navigation or some other correlated attributes.

VII. Conclusion

We have presented an approach to path refinement: that is, to producing accepting paths for a robot given input curves that indicate roughly what kind of path is suitable. The method consists of using *a-priori* data to automatically learn constraints and preferred patterns to configure

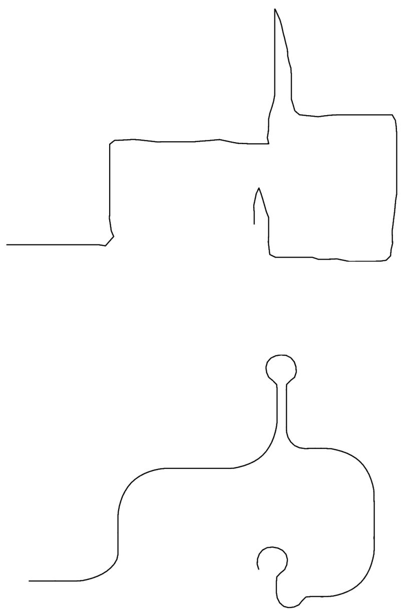

Fig. 7. The training examples are samples of motions with non-holonomic constraints. The input path (top) is a hand drawn (noisy) path. Below shows the generated path maintaining the bounded turning radius constraint for non-holonomic motions.

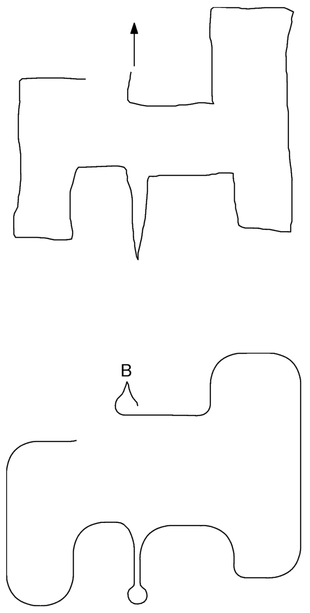

Fig. 8. The figure above was generated using training examples for non-holonomic constraints. The input path (top) is a hand drawn path with a restriction on the initial direction of motion. Below shows the generated path where point (B) marks a direction reversal.

a Hidden Markov Model. The learned probabilities are used to synthesize a new path given an arbitrary normally unconstrained path. Experimental results display how a synthesized path exhibits the constraint properties of the training data while following the overall shape of the input curve.

In this discussion we have assumed that when a path is generated, we know *a priori* which family of statistical biases we should apply. In practice, it may be that in one part of a path we want one style of locomotion and in another part we expect a different style. How to incorporate two different types of bias in the system and, further, how to make a transition between them remains a topic we are still investigating.

Since our approach is based in local refinements to an input curve, it is not readily able to apply large-scale reconfigurations to a path (it a possible, but leads to various technical problems). As such, this approach is best suited to problems where at least the homotopy class of the desired solution is clearly indicated. While this is a restriction, it is not apparent that any other mode of operation would be desirable.

VIII. References

[1] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer Academic Publishers, 1991.

[2] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," in *American Journal of Mathematics*, vol. 79, pp. 497–517, 1957.

[3] Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," in *Pacific Journal of Mathematics*, vol. 145(2), pp. 367–393, 1990.

[4] L. K. J.-C. Latombe, "Randomized preprocessing of configuration space for fast planning," in *IEEE Internation Conference on Robotics and Automation*, 1994.

[5] L. K. J.-C. Latombe, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, August 1996.

[6] S. Singh and M. C. Leu, "Optimal trajectory generation for robotic manipulators using dynamic programming," in *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 109, 1989.

[7] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray, "A motion planner for nonholonomic mobile robots," in *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 577–593, 1994.

[8] M. Learning, "Tom mitchell," in *McGraw Hill*, 1997.

[9] X. Wang, "Learning planning operators by observation and practice," in *Artificial Intelligence Planning Systems*, pp. 335–340, 1994.

[10] J. Miura, "Hierarchical vision-motion planning with uncertainty: Local path planning and global route selection."

[11] S. P. Engelson, "Learning robust plans for mobile robots from a single trial," in *AAAI/IAAI, Vol. 1*, pp. 869–874, 1996.