Analogical Path Planning

Saul Simhon and **Gregory Dudek**
Centre for Intelligent Machines
McGill University
Montreal, Quebec
{saul,dudek}@cim.mcgill.ca

Abstract

We present a probabilistic method for path planning that considers trajectories constrained by both the environment and an ensemble of restrictions or preferences on preferred motions for a moving robot. Our system learns constraints and preference biases on a robot's motion from examples, and then synthesizes behaviors that satisfy these constraints. This behavior can encompass motions that satisfy diverse requirements such as a sweep pattern for floor coverage, or, in particular in our experiments, satisfy restrictions on the robot's physical capabilities such as restrictions on its turning radius. Given an approximate path that may not satisfy the required conditions, our system computes a refined path that satisfies the constraints and also avoids obstacles. Our approach is based on a Bayesian framework for combining a prior probability distribution on the trajectory with environmental constraints. The prior distribution is generated by decoding a Hidden Markov Model, which is itself is trained over a particular set of preferred motions. Environmental constraints are modeled using a potential field over the configuration space.

This paper poses the requisite theoretical framework and demonstrates its effectiveness with several examples.

Introduction

In motion planning (e.g. for a robot), we typically need to consider two types of constraints on the paths that can be taken: constraints imposed by the environment, and constraints imposed by the physical characteristics of the vehicle itself. In particular, *non-holonomic* constraints refer to limitations on the allowed derivatives of the path, and planning in the presence of such constraints in often difficult (an automobile is a commonplace example of a vehicle which such constraints, as it is unable to move perpendicular to the direction in which it is facing). Motion planning is typically complicated not only by the need to generate paths that satisfy various constraints, but also by the need to initially model whatever constraints may be imposed by a particular vehicle or task. Finding valid trajectories that satisfy both the mechanically imposed constraints and environmental constraints can be a difficult task. In this work, we present a method for "analogical path planning" wherein paths are generated by analogy with previous observed acceptable paths, and without an analytic model.

Our approach is based on accepting an input path (which may not satisfy the required constraints) and transforming it into an allowed path. The input path may be produced either by a person (e.g. as a sketch) or by an automated system; this request input path is referred to as a "goal path". The analogies that drive the system are learned from examples, where each example is composed of a pair of paths, an input and an output path fragment. These fragments depict a path fragment that might be requested, and the restricted (allowed) path that should be produced when such a request is made. Further, the constraints on the generated paths can be either hard constraints or probabilistic, and they may derive either from kinematic or dynamic constraints on the vehicle motion, or from other types of bias (for example, we might like to generate circuitous paths for our vehicle to determine if we are being followed by enemy spies).

Because our system learns from examples, it can be applied in a variety of domains. We avoid having to extract and analytically model constraints for each desired motion pattern or mechanical configuration that we may wish to control. Our object is that we should be able to demonstrate how a robot can move and subsequently the system would automatically produce new paths based on these motions. One area of application is tele-operated robots. A human can guide the robot to areas by simply sketching a coarse path. The system would then refine that path based on the learned specification of the robot and generate a new valid path analogous to the goal. Another application would be to complement high level planners to relieve them of the burden of non-holonomic (complex) path planning. Our system can take in as input such path generated by such planners and augment them to avoid objects while maintaining a particular behavior during motion.

Our approach consists of training a Hidden Markov Model for a set of examples that encode the analogies between pairs of *motion patterns* (output curve fragments) and *goal trajectories* (input curve fragments); each of these curve fragments is represented as a succession of tangent angles. The Hidden Markov Model learns local constraints on the sequence of tangents in the motion patterns in conjunction with their correlation to sample points in the goal trajectories. When producing new paths, these learned constraints

are then combined with environmental constraints that bias the distribution of a Markovian Chain. At each step in the trajectory, we compute the posterior probability distribution of states in the

configuration space by combining a prior based on the learned fragments with a likelihood based on the input curve and the environmental constraints. Once the probability distribution over the entire input trajectory, the candidate path with maximum likelihood is chosen.

The paper is organized as follows. First we briefly review previous work. Then, we describe our approach to learning motion constraints. Then we present our method to generate novel motions from the learned constraints. We describe a general framework in combining the learned constraints with arbitrary biases and show cases using environmental and other constraints. Finally we present results and conclude.

Related Work

There is a wide array of works by many authors that examine methods for path planning. One approach, similar in spirit to our work, is to provide stochastic models for maintaining and updating a distribution on predefined states (Thrun 2000; Simmons & Koenig 1995). These distributions generally reflect the likelihoods of a robot's pose or sensory information. Based on these likelihoods, a robot can make decisions on where to go next given a particular goal.

Another key ideas in the area is the notion of path planning under non-holonomic constraints. 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 (Latombe 1991). Dubins (Dubins 1957) and Reeds and Shepp (Reeds & Shepp 1990) 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. Similarly, jerky paths are sometimes smoothed using energy minimization methods (Singh & Leu 1989; Laumond *et al.* 1994).

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 (Mitchell 1997). There has also been some prior work on the relationship between learning and planning, most of this has dealt with more traditional plan formulation problems (Wang 1994) or on learning suitable cues that control or determine plan synthesis or execution (Engelson 1996).

Learning Motion Behavior

Motion behaviors are learned from examples that show the way a path should behave given an goal (input) trajectory. Figure 1 shows a training set used to learn non-holonomic constraints corresponding to motion with a bounded turning angle (or radius of curvature). Each example in this set consists of an acceptable trajectory which is coupled to a goal trajectory (which is unacceptable on its own). For example, when our goal is to make a sharp right turn, the resulting path should be the smooth kinematically correct path. When our goal is to go straight while the robot is facing perpendicular, then the preferred path is a d-turn motion.

Given such a set, we train a two layer model. In one layer, local motion constraints are learned by examining the sequence of sample points in the acceptable paths. Then, in the other layer, the control function (the function describing a subset of valid trajectories given a goal) is learned by examining the correlation of the path sequence over valid trajectories with that of the goal trajectory. In general, we consider the training set as a set of examples that encompass a class of candidate trajectories for specific goals.

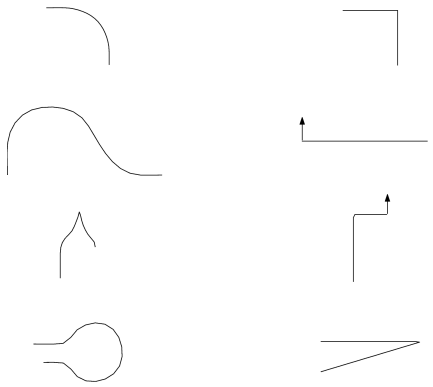

Figure 1: Samples of a training set simulating non-holonomic motions. Paths on the left display the constrained motions while paths on the right display the associated unconstrained goal path. Where specified, arrows indicate additional constraints on the direction of motion to account for the orientation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

Each example trajectory is represented by a parametric function $f(t, a)$ where t is the arc-length of the path and a is a path attribute. We use two attributes in out system; the shape of the path, as a sequence of the curve's tangent angles $\theta(t)$, and the facing direction of the robot axis $\phi(t)$. The shape $\theta(t)$ is represented over multiple scales using a wavelet decomposition $\theta(t, s)$ over the scale s. This allows us to efficiently capture large scale constraints on the paths, where one sample point in the higher scale represents a summary of several points in the low scales. Our examples are suitably normalized and sampled uniformly.

While other work (Simhon & Dudek 2003) has exploited a Hidden Markov Model (HMM) for path synthesis, the present paper is the first to consider environmental landmarks as well as constraints learned from examples. A

HMM is a two layer linear dynamic model. It models two primary components: 1) the transition probabilities of an underlying evolving system that is not directly observable and 2) the probabilistic effect that this underlying system has on observations. A HMM Λ is defined as follows:

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

where M is the transition matrix with transition probabilities on 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 would give rise to an observation o_i, $p\{o_i(t) \mid h_j(t)\}$, and π is the initial distribution of the hidden states.

In our approach, the hidden states represent sample points from the preferred paths while the observations represent sample points from the goal paths. As such, the leaned transitions in the hidden layer impose local constraints on the succession of tangent angles and robot facing direction of a valid trajectory. Further, the learned relationship between the hidden layer and the observation layer identify what constraints should be applied given the current goal path (i.e. the conditional probability of the hidden states given the observation).

A HMM is trained from the statistics of the training set. The transition probabilities in M are computed by counting the number of matches of successive sample points in the preferred paths. Similarly, the confusion probabilities in B are computed by counting the number of matches of associated sample points from the preferred path and the goal path. The initial distribution π is assumed uniform suggesting that at $t = 0$ all motions have equal likelihood. These constitute the learned priors of our system that are later used to produce valid motions over the configuration space and a goal.

The dimensionality of our state space is augmented in order to capture the multi-dimensional function $f(t, a)$. Thus, each state h_i and o_i is considered a multi-dimensional element. This allows us to capture multiple path attributes at various scales using only a single state (simulating a higher order Markov assumption).

Generating Motions from Learned Constraints

Given the unconstrained goal trajectory, we wish to generate a new path that is consistent with the constraints of our trained HMM Λ. That is, the resulting motion should be consistent with the local constraints and input coupling seen in the examples. Decoding a HMM solves for the maximum likelihood hidden state sequence (our constrained motion) which best describes the given observation sequence (our goal trajectory). Let $\alpha = \{h(0), h(1), ..., h(T)\}$ denote the output path and let $\beta = \{o(0), o(1), ..., o(T)\}$ denote the coarse input path, we wish to solve the following:

$$\begin{gathered} \max p\{\alpha \mid \beta, \Lambda\} \\ or \\ \max_{h_i, ...h_n} p\{h(0), h(1), ..., h(T) \mid o(0), o(1), ..., o(T), \Lambda\} \end{gathered} \tag{2}$$

One approach in solving for this (analogous to Variational Calculus techniques) is to evaluate a cost functional by accumulating local costs over the entire sample point sequence of a candidate solution. Then iterate over the solution space to minimize this total cost. This approach can become very inefficient in time and is subject to convergence pitfalls. Rather, our method consists of a dynamic programming technique which is more efficient in time, $O(n^2T)$ for n states and T sample points, and is guaranteed to converge. Although it is more memory intensive.

Our approach is based on the *Viterbi* algorithm for decoding a HMM. For each sample point in the sequence, we compute the distribution over **all** the hidden states (i.e. all paths). This is accomplished by first using the likelihoods of the confusion matrix to condition over the current observation. Then, we propagate that distribution to the next sequence point using the transition matrix likelihoods. For each state at sample point $t + 1$, we compute its posterior likelihood ψ as follows:

$$\begin{gathered} \psi\{h_i(t)\} = p\{o(t) \mid h_i(t)\}\psi\{h_i(t)\} \\ \psi\{h_i(t+1)\} = \max_j [p\{h_i(t+1) \mid h_j(t)\}\psi\{h_j(t)\}] \end{gathered} \tag{3}$$

The propagation phase is similar to the transition method used to generate a Markov Chain. However because our eventual goal is to uniquely specify each successive point, we do not sum over all possible transitions but rather only keep the maximum. Additionally, for each state, we store a back-pointer to the state in the previous iteration that put forward this maximum likelihood transition. Once we have iterated over the entire sequence of observations, we chose the state with maximum likelihood at time T and then backtrack the sequence using the back-pointers. Backtracking is crucial for a consistent sequence. Rather than dealing with each sequence point independently, it considers the maximum likelihood transitions over the entire sequence.

Once we find the most likely sequence of tangent angles, we realize the path by computing the (x, y) points using the uniform sampling rate and corresponding arc-length segments from training. Figure 2 and 3 show examples of goal trajectories and their resulting synthesized paths. Due to memory limitations in practice, we can not explicitly store the entire distribution over all possible sates H_i. Thus, we only keep the top candidate states and threshold candidates with low probability. Similarly, we cannot store all the prior likelihoods in a matrix but rather we must compute them on the fly by searching our training set and counting the number of matches.

Likelihood Blurring

Rather than computing the likelihoods (of Equation 3) by searching for exact matches, we compute them based on the goodness of the matches. This is performed for both the transition and confusion likelihoods and is key in avoiding issues with quantization errors or observation mismatches. (Expecting the observations to match exactly with those in training is overly restrictive, in general, the input curve can be a noisy curve.)

For the transition likelihoods, we assume a Gaussian noise model with the variance empirically set. This variance provides a control on the *mixing* tendency of examples. The

larger the variance, the more likely we are to accept transitions from one tangent angle to another, even if those tangent angles do not necessarily appear in sequence. We typically keep the variance very low (in the order of a half of a degree) to strongly enforce our motion constraints while avoiding issues with quantization errors.

For the confusion likelihoods, we use a Sigmoid distance metric. Using the Sigmoid function, we can suggest that for a given error range, the likelihood of matching is very similar but beyond that range, the likelihood decays exponentially. The Sigmoid shift parameter determines the amount of bias the input curve has over the distribution. A large value suggest that any input will match any state, reducing the impact of the input, while a small value will enforce the strength of the input conditional. The Sigmoid parameters are set empirically.

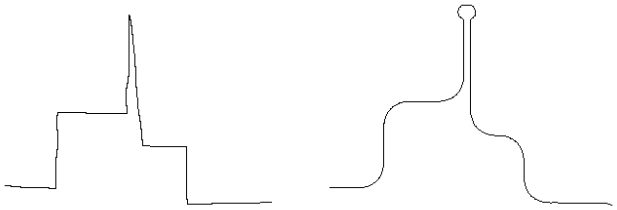

Figure 2: The left shows a coarse input path while the right shows the refined motion consistent with the learned nonholonomic constraints.

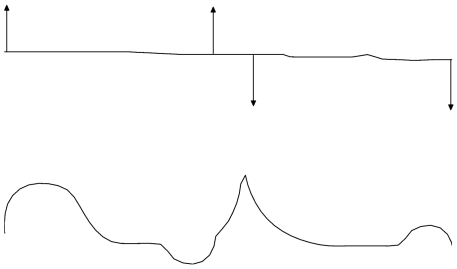

Figure 3: The top shows an example path with the desired robot axis facing directions (shown by arrows) and the bottom shows the resulting output path.

Bayesian Path Reconstruction

Our goal is to reconstruct a *valid* path given a coarse goal trajectory. In our framework, there are several components that contribute to what we constitute a valid path. First, we must take into account the HMM such that the output is consistent with the training examples as described is section . Second, we require that the generated curve should stay near the input trajectory. (The HMM alone only constrains the tangent angles of the curve, it does not guarantee that the resulting path would stay near the input, Figure 4). Finally, the output curve should not go through or approach too close to obstacles in the environment. This combination of constraints can result in complex paths that would otherwise be difficult to determine using traditional analytical models.

The following lists the primary constraints of our system:

- The learned constraints of our HMM
  - local shape consistency with examples
  - control from the goal trajectory
- Distance to the goal trajectory
- Obstacles avoidance

We formalize this problem in a Bayesian framework such that it will become transparent to combine these (or other) constraints within the system. Our method is analogous to regularization techniques which are commonly used to reconstruct objects from partial data. The main idea in regularization methods is to minimize a cost functional that measures the distance between the data and the model in question and includes some additional constraints (such as smoothness) that bias the solution. It can be show that a probabilistic approach to regularization consists of maximizing the posterior likelihoods of a Bayesian model (Keren & Werman 1994; Szeliski 1989):

$$\max_{f \epsilon \mathcal{M}} P\{f \mid D\} \propto \max_{f \epsilon \mathcal{M}} p\{D \mid f\} p\{f \mid \mathcal{M}\} \qquad (4)$$

where f is an object in class $\mathcal{M}$ and D is the coarse data. Typically the *data model term* $P\{D \mid f\}$ assumes a Gaussian noise model and the *regularization term* $P\{f \mid \mathcal{M}\}$ is a prior that adds a bias toward smoother models.

Rather than having a fixed noise model or prior constraint, we learn these from examples and encode them in our HMM. It is easy to see that in a HMM, the confusion matrix represent an arbitrary *data model* ($p\{o_i(t) \mid h_j(t)\}$) and the transition matrix represent *priors* that biases solutions that are locally consistent with our training set. Thus, our HMM is analogous to a cost functional that generalizes to learned constraints.

Using the Bayesian framework, we can embed additional constraints over our HMM. In particular, we wish to include a bias for output curves that are closer to the goal trajectory and avoid obstacles. In general, we can regularize at each sample point using the desired biases as follows:

$$E\{h_i(t)\} = -log\{\psi(h_i(t))\} + \sum_k \lambda_k R_k(t, h_i) \qquad (5)$$

where $R_k(t)$ is the energy of a regularization constraint and λ_k is the associated weight. This embeds the additional constraints within our model such that they are taken into account in the decoding phase. As such, the decoding algorithm can be thought of as an energy minimization algorithm.

Magnetic Attraction

We include a bias that increases the likelihood of solutions that are closer to the goal trajectory. This simulates a magnetic attraction between the input path and the generated

one. While this is an important additions to the system, we need to be careful in the amount of influence it exerts. In order to achieve the desired motion behavior learned in our HMM, it is still necessary for the output path to stray away at some distance. The amount of divergence allowed can be controlled by the weight of this energy term. (One may suggest ways to set λ_{mag} based on the maximum divergence seen in the examples.) We can also relax this bias at the interior region of the path while enforcing it at the endpoints (for closure). We compute the regularization energy for magnetic bias as follows:

$$R_{mag}(t, h_i) = e^{|1-\frac{2t}{T}|}\ d^2 \tag{6}$$

where t is the current sample point position, T is the total sample points and d is the distance between the goal and the candidate state h_i. At the endpoint of the path the relative weight of the energy term is higher than at regions within the path. As such, the bias term will have more influence at the endpoints and less at interior points.

At any given point in the sequence, we need to compute the distance d between each candidate state and the associated point on the input curve. Thus, we augment the of the states to include auxiliary information that identifies the Cartesian co-ordinate. These dimensions are not used during the propagation or conditioning steps but are simply supplementary terms that store the required information. At $t = 0$ the auxiliary values are bootstrapped to the position of the input curve. Then, the co-ordinates of the states at the next iteration are computed by extrapolating their tangent from the co-ordinates of the previous state (identified by the backpointer) using a suitable arc-length (based on the training sampling). Figure 4 shows example paths generated with and without the magnetic term. It can be seen how the output curve without the magnetic term diverges from the input while adding the magnetic term biases the result to stay closer to the input.

Figure 4: Example of magnetic attraction. The left path shows the goal trajectory, the middle shows the resulting output without the magnetic energy term and the right path shows the output with magnetic energy.

Obstacle Avoidance

We compute a regularization constraints over the configuration space of the robot by applying a distance transform on obstacles in the environment. Akin to potential field methods, the transform generates an energy field over the environment. A suitable function would generate high energies at regions near the obstacles and low energies at regions far form obstacles. We compute the energy for a state as follows:

$$R_{obs}(t, h_i) = \max_{obs} \frac{1}{d_i} \tag{7}$$

where d_i is the distance between the current candidate state and the i^{th} obstacle. We choose the obstacle that produces the maximum energy over all the others. At position close to or on the obstacles, the energy goes to infinity while at areas further the energy decays to zero. The weight λ_{obs} controls the amount of influence the obstacles have on the solution. Large value will coerce the robot to stay far from the obstacles while small values will allow the robot to reach closer to the obstacles, traveling through narrower regions.

We prepossess the environment and generate the field over a grid. To evaluate the energy for a state, we simply use its auxiliary parameter that identify the Cartesian co-ordinates and access the girds value. Figure 5 shows and field generated for a sample environment.

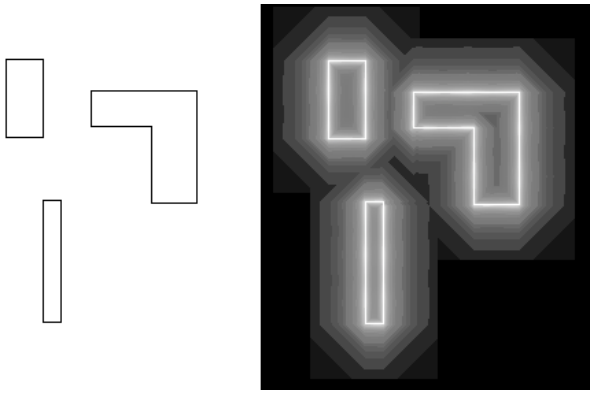

Figure 5: Example energy field. The left image shows the scene and the right image is a logarithmic plot of the energy field.

Results

Figure 6 and 7 shows two example environments with the goal path (left) and the generated path (right). The goal path, sketched by a human operator, directs the robot too close to or through obstacles. It can be seen how the generated path avoids the obstacles while also maintaining the learned constraints.

Figure 8 shows an example of two generated paths from a goal path that goes through objects. The middle figure shows the result using a large value for the environment constraint weight while the left shows a small value. It can be seen that with a large value, the path stays further away from the obstacles while a small value allows the path so go through narrow regions. One noticeable issue here is that when the trajectory of the output path is much longer that the input path, the result does not reach the goal. This is due to the fact that we have not enough sample points in the goal trajectory. One approach to overcome this issue is to dynamically sam-

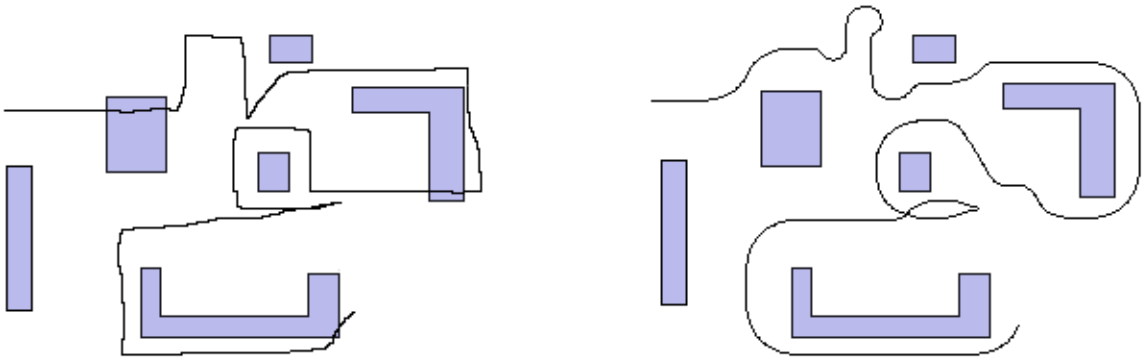

Figure 6: Example path synthesis. Left shows the input and right shows the output.

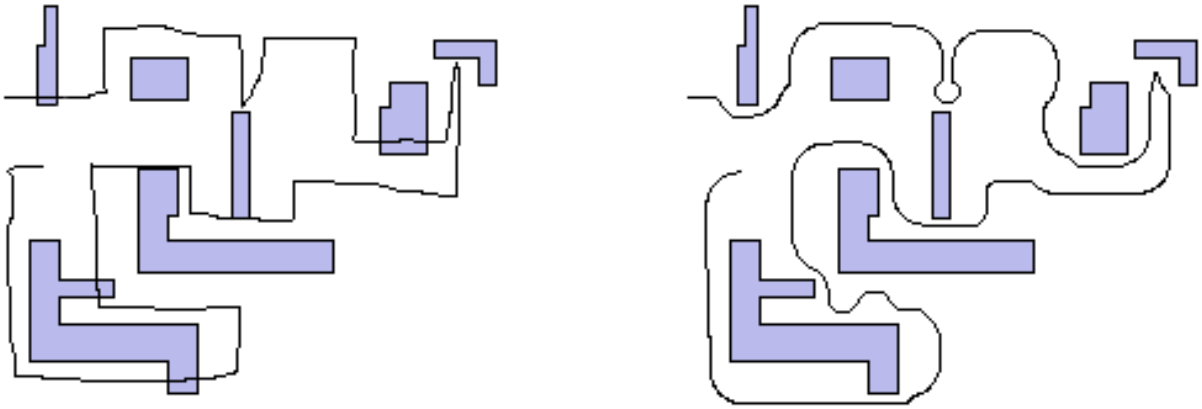

Figure 7: Example path synthesis. Left shows the input and right shows the output.

ple the goal path. This can be based on the projection of the resulting path onto the goal path.

Conclusion and Future Work

We presented a method for analogical path planning. That is a method to generate paths that are analogical to both an input trajectory and a set of examples. The generated (analogical) paths maintain the local constraints expressing a desired motion preference. By formulating our Hidden Markov model in Bayesian terms, we were able to embed supplementary constraints such as proximity to input and obstacle avoidance.

One open problem that remains to be addressed is that of finding good values for the parameters that control the process. While this is performed empirically, we are currently examining how to set these values based on some initial conditions, such as a maximum divergence form the input or a minimum distance to the obstacles. 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.

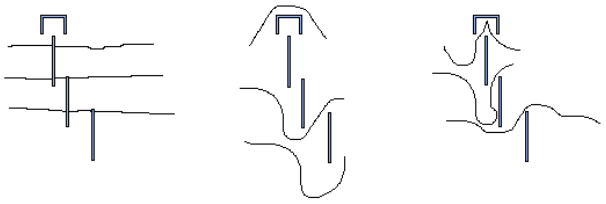

Figure 8: Example path synthesis going through a narrow region. Left shows the input. The middle figure shows the output using a large value for λ_{obs} and right shows the output with a small value for λ_{obs}

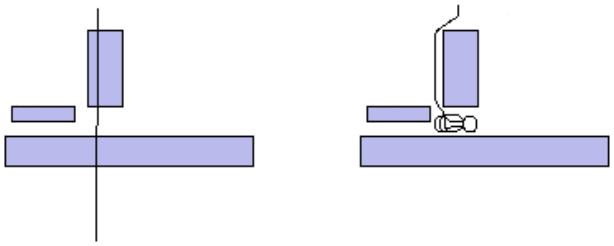

Figure 9: Example where the goal path goes through a large object. This causes the resulting path to be stuck behind the obstacle.

References

Dubins, L. E. 1957. 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*, volume 79, 497–517.

Engelson, S. P. 1996. Learning robust plans for mobile robots from a single trial. In *AAAI/IAAI, Vol. 1*, 869–874.

Keren, D., and Werman, M. 1994. Bayesian interpolation. *ARPA Image Understanding Workshop*.

Latombe, J. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

Laumond, J.-P.; Jacobs, P. E.; Taix, M.; and Murray, R. M. 1994. A motion planner for nonholonomic mobile robots. In *IEEE Transactions on Robotics and Automation*, volume 10, 577–593.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Reeds, and Shepp, L. 1990. Optimal paths for a car that goes both forwards and backwards. In *Pacific Journal of Mathematics*, volume 145(2), 367–393.

Simhon, S., and Dudek, G. 2003. Path planning using learned constraints and preferences. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Simmons, R., and Koenig, S. 1995. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1080–1087.

Singh, S., and Leu, M. C. 1989. Optimal trajectory generation for robotic manipulators using dynamic programming. In *ASME Journal of Dynamic Systems, Measurement and Control*, volume 109.

Szeliski, R. 1989. *Bayesian Modeling of Uncertainty in Low Level Vision*. Kluwer.

Thrun, S. 2000. Probabilistic algorithms in robotics. *AI Magazine* 21(4):93–109.

Wang, X. 1994. Learning planning operators by observation and practice. In *Artificial Intelligence Planning Systems*, 335–340.