# A Grasp-based Motion Planning Algorithm for Character Animation

Maciej Kalisiak[1]
Michiel van de Panne[1]

Department of Computer Science
University of Toronto

## Abstract

The design of autonomous characters capable of planning their own motions continues to be a challenge for computer animation. We present a novel kinematic motion planning algorithm for character animation which addresses some of the outstanding problems. The problem domain for our algorithm is as follows: given a constrained environment with designated handholds and footholds, plan a motion through this space towards some desired goal. Our algorithm is based on a stochastic search procedure which is guided by a combination of geometric constraints, posture heuristics, and distance-to-goal metrics. The method provides a single framework for the use of multiple modes of locomotion in planning motions through these constrained, unstructured environments. We illustrate our results with demonstrations of a human character using walking, swinging, climbing, and crawling in order to navigate through various obstacle courses.

## 1 Introduction

The animation of human figures has been a challenge that has seen the evolution of many tools, operating at a variety of levels of abstraction. Many of the available methods target the creation of specific motions in structured environments, such as walking on flat terrain. However, there are remarkably few methods which tackle the problems involved in making human figures navigate in complex, unstructured environments. Examples of this type of problem include a climber on a mountain face, a child playing on a jungle-gym, or a game character crawling through a tunnel.

[1]{mac|van}@dgp.utoronto.ca

The automated synthesis of motion for characters in unstructured environments is difficult because it requires solving a planning problem subject to multiple constraints. Obstacles in the environment constrain the motion in an obvious fashion, as typified by a narrow passageway in a cave. Other types of constraints include a character's joint limits, the requirements for balance and support throughout the motion, as well as the character's natural disposition for particular postures and motions. This set of complex, heterogeneous constraints motivates our use of stochastic optimization techniques in addressing this problem.

Navigation in unstructured environments entails some particular challenges. Global and local solutions can be strongly linked; the choice of a particular route towards a goal is predicated on the route being viable every step along the way. Planning algorithms for such problems thus require the ability to plan motions across both small and large time scales. A second challenge is that creating motions involves both discrete and continuous decisions. An example of a discrete decision is that of deciding whether to step on or over an obstacle, or simply deciding which of a finite set of possible hand-holds to use. Once the contact points of a character with the environment have been chosen, the remaining decisions shaping the motion can be regarded as being continuous in nature.

An example of the type of problem that can be solved by our motion planner is presented in Figure 1. The diagram illustrates one particular solution obtained for a simple 10-link, 9-joint character, which is further depicted in Figure 2. The small boxes on the obstacle surfaces represent *grasp points* which are points at which the character is allowed to grasp, pull, or step on. These represent part of the problem specification in our algorithm, as will be discussed later. This particular environment requires the alternating use of four modes of locomotion in order to navigate towards the goal: walking, crawling, climbing,

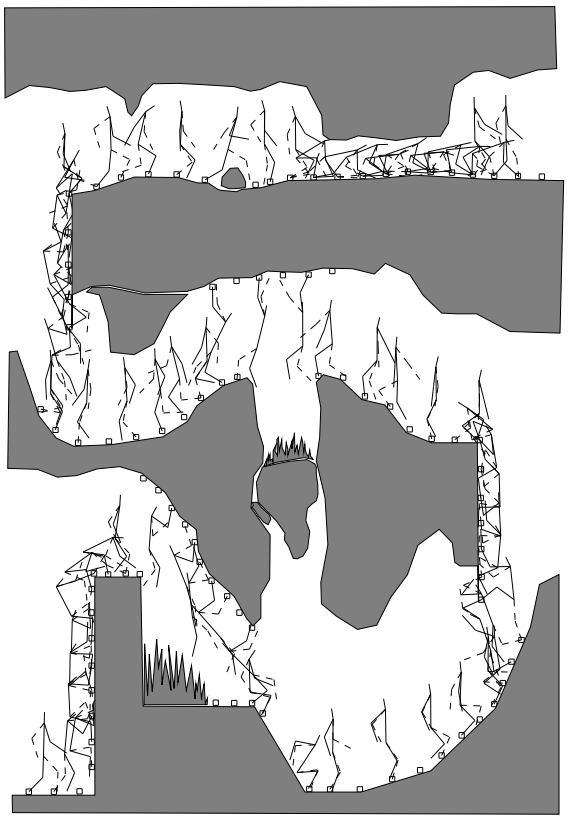

**Figure 1:** An exemplary solution to an unstructured environment traversal.

and swinging. The solution also necessitates variations of these basic modes, such as walking up hills, stepping over obstacles, and ducking the head when necessary.

Our planner uses the randomized path planning (RPP) methods of Latombe et al.[3, 18] as a point of departure. This previous work deals with a class of robot motion planning problems, typified by the example shown in Figure 3. The problem statement for this example is to move the three-link articulated figure from the initial configuration, A, to the goal configuration, B, without colliding with the constraining environment. The piano mover's

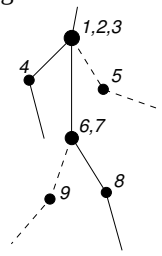


**Figure 2:** The 10-link, 9-joint character model used by our planner. The numbers in the diagram enumerate the joints.

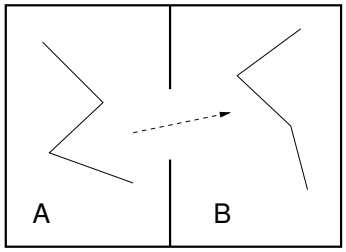


**Figure 3:** Moving a simple articulated robot in a constrained environment.

problem is a strongly related problem: determine how to move and orient a piano through a set of rooms and hallways to a given goal location without getting stuck. As shown in [3, 18], these types of problems can be effectively solved using RPP techniques.

When applied to character animation, the basic RPP algorithm is capable of generating free motions through an environment between given start and end configurations, as shown schematically in Figure 4. In order to produce more realistic motions, we shall augment the basic RPP algorithm in several ways. Grasp points are introduced as a means of representing possible points of contact with the environment, such as footholds and handholds. A finite state machine structure is used to represent particular modes of locomotion, possible transitions among them, as well as their relative preference. A posture correction step is introduced at key points in the solution as a means of modeling preferences for particular posture characteristics. Lastly, trajectory filters are added to ensure the fluidity of the final synthesized motion. In section 3, we shall expand upon each of these additions in turn.

The remainder of this paper is structured as follows. In section 2, we describe related previous work. Section 3 describes the various elements of the motion planning algorithm. Our results are presented in section 4, followed by a brief discussion in section 5. The paper concludes with some proposed future work in section 6.

# 2 Previous work

Many methods have been brought to bear on the problem of character animation. This variety stems in part from the unique requirements of various applications such as games, film production, and ergonomic analysis. The following review of previous work briefly touches on general character animation methods and then focusses more closely on character animation methods which emphasize motion planning.

While keyframing continues to be the mainstay of character animation, a variety of alternative kinematic and

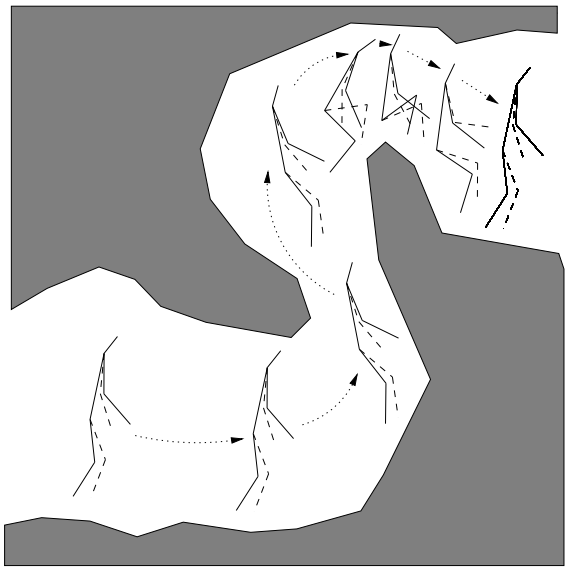

**Figure 4:** Contact-free motion planning

dynamic methods exist. Several kinematic methods are dedicated specifically to producing human walking[4] or running[6] gaits. Other walking and running methods employ a hybrid mixture of kinematics and dynamics[5, 9]. Lastly, dynamic simulations have had some success in reproducing human walking[17] and running[22, 11] gaits. In general, all of these methods are thus far restricted to well-structured environments.

Spacetime constraint methods[28] and their subsequent variations offer promise in that they can readily incorporate a mix of hard and soft constraints on a motion. By using appropriate simplifications for the physics, it has been shown that the principles of trajectory optimization can be applied to animating bipedal[27] and quadrupedal characters[24].

A different set of techniques offer the capability to make flexible use of motion capture data by allowing various transformations to be applied. In relatively simple, unconstrained situations, smooth deformations of trajectories can be used to meet particular keyframe constraints[29]. More sophisticated methods can further take contact constraints and character proportions into account[10, 13, 23], and more recently, also the physical correctness of the motion transformation[21]. Yet other methods apply signal processing methods to motion data in order to capture and modify particular motion characteristics[1, 7, 26].

The *Jack* system[2, 20, 19] is a system which aims to solve motion planning problems closer in nature to the ones we address. The Jack system is a complex, multifaceted system designed in part to perform ergonomic studies. It allows the user to perform field-of-vision analysis, comfort assessment, as well as testing reachability. It has been further outfitted with strength modeling and collision avoidance, and is capable of grasping objects. Developed at University of Pennsylvania, it is now a commercial product[25].

The Jack system is particularly adept at solving the local motion planning problems found in ergonomic studies. However, to the best of our knowledge, it does not solve the particular problem being addressed in this work, namely the automatic planning of global motions through complex unstructured environments, exploiting multiple modes of locomotion as necessary.

Similar work has been done in [16], where a framework is outlined for animating autonomous agents. It allows virtual humans to navigate and carry out high-level tasks in a realtime environment. Motion is generated by using a simple 2D planner to construct a collision-free trajectory through the overhead-view of the environment, which is subsequently fed to a path-following controller. This controller assembles the final motion by modifying and combining motion capture data. The main difference between the Jack system and this framework is the latter's use of visual feedback, the character's field of view, in the planning process.

The robot motion planning work of Latombe et al.[3, 18] proposes the use of the randomized path planning (RPP) method and is the starting point for our character motion planning algorithm. The RPP method has many benefits: it is among the fastest known methods for solving constrained motion planning problems, and it scales well with the complexity of both the object and its environment.

The RPP algorithm has been extended to deal with 3D manipulation tasks in [14], which focusses on the cooperative multiarm manipulation of objects and is suited especially well to tasks which require regrasping of the object being manipulated. However, problems of locomotion are not addressed in this or previous RPP work.

The Motivate 3D game system[8, 15] is a commercial 3D game development system which aims to address some of the same motion planning issues as we do. However, as a result of the stringent requirements of games, both the goals and the methods employed differ from the work we shall present. The Motivate system, much like many game engines, places the emphasis on real-time character animation at the expense of motion continuity and planning sophistication, as the real-time requirement is a must for game playing environments. It also addresses object manipulation, which we do not address. Motions are synthesized in the Motivate system by making liberal

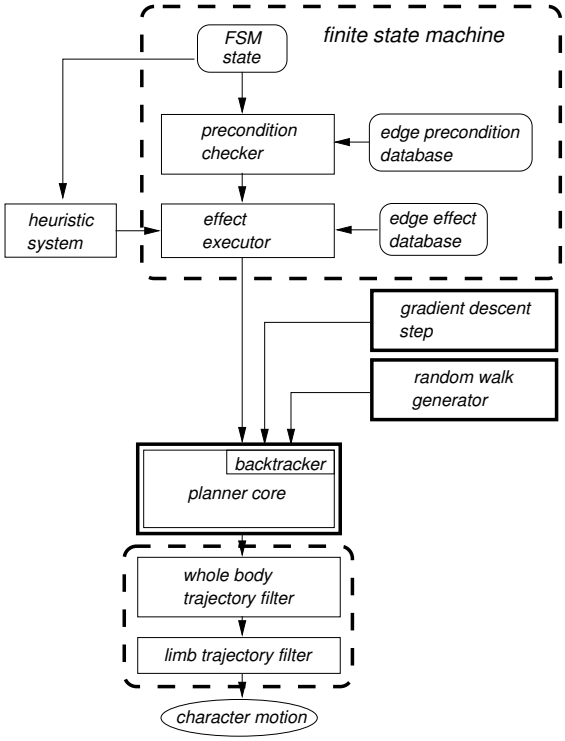


**Figure 5:** Overview of our motion planner.

use of a form of motion warping to adapt motion instances retrieved from a 'skills' database to the specifics of the current situation.

# 3 The motion planner

Our motion planner can be described in terms of five interacting components, as shown in the block diagram of Figure 5. In this section, each of the components of the planner is described in turn, although we shall on occasion refer the reader to [12] for particular details and parameter values that will be of use in precisely reproducing our results.

Grasp points are a fundamental concept throughout our motion planner. These are an enumerated set of points of the environment which may be used as footholds or handholds by the character. Given an environment, grasp points can be designated manually, or through an automatic process. Three types of grasp points exist: load-bearing, pendent, and hybrid. Generally, the first represents a potential foothold, the second a potential handhold, and the last can be used as either. The job of the motion planning algorithm is to find a natural sequence of grasp-point-to-limb pairings which the character can use to move towards the goal configuration.

## 3.1 The planner core

As its name would imply, the planner core is central to the motion planning process. It acts as an arbitrator and scribe for three possible sources of motion sequences: (1) the locomotion mode finite state machine, (2) a gradient descent single step, and (3) the random walk generator. The finite state machine is always consulted first. If it cannot provide a motion segment, then the gradient descent module is queried. If this also fails, as in the case of a local minimum, then the planner core falls back on the random walk generator. The core iterates through this process until the goal is reached.

In addition to invoking a motion source and concatenating the results to the developing solution trajectory, the planner core can also decide to backtrack. Backtracking is employed in situations where the current motion plan is perceived to have reached a dead end. In this case the planner rolls back the current motion plan to a stochastically chosen backtracking point and then restarts the planning process from there. The conditions under which the backtracking procedure is invoked will be described shortly.

## 3.2 Gradient descent

The gradient descent process provides the means to drive the character towards the goal configuration. Our implementation of this particular process generally follows that presented in the original work on randomized path planning (RPP)[3, 18]. A single gradient descent step makes a small change to the *configuration* of the character such that the character moves closer to its goal configuration. The configuration, $q$, of a character is a complete specification of all the degrees of freedom, typically consisting of the 2D or 3D location of the root of the character in space, the Euler angles specifying the subject's general orientation, as well as all the internal joint angles of the character.

Computing a motion towards the goal first requires defining a distance-to-goal metric, which we shall refer to more formally as $P(q)$, the configuration-space potential function[18]. $P(q)$ thus computes a scalar value representing the remaining distance to the target or goal configuration, $q_{target}$. There are many possible ways of defining a distance-to-goal function. One simple possibility is to track the positions of a collection of *control points* placed on the character. The sum of the geometric distances between each control point in the current configuration, $q$, and the target configuration, $q_{target}$ then defines our distance metric. This metric is more meaningful than simply computing a norm on $|q - q_{target}|$, as such a difference of

configurations contains both linear and angular measures which cannot readily be combined in an even-handed way. However, this metric does not take the environment into account in any way. A better solution then is to use the shortest *free-space* path between each control point in its current and final configurations as a substitute for the geometric distance. In our implementation, we use only one control point that is located at the character's center of mass.

Computing the shortest free-space path between two points in a complex environment remains a non-trivial subproblem. For this, our algorithm relies on a discrete approximation, similar to that in [18], which can be efficiently computed as follows. First, a binary-valued *occupancy map* is created by using an axis-aligned grid to uniformly divide the environment into a set of rectangular cells. A cell in the occupancy map is marked as *unoccupied* if more than half of the cell is free space. Otherwise, it is marked as *occupied*. The occupancy map is then used to compute a corresponding *distance map*, which for each cell stores the Manhattan distance through freespace to the cell containing the target control point (i.e., the cell containing the character's center of mass when the character is in configuration $q_{target}$). The distance is measured as the number of free-space cells that need to be traversed, using 4-connectivity, in order to reach the target cell. The distances can be efficiently computed using a simple form of dynamic programming, which manifests itself as a wavefront expansion algorithm in this case.

Given the potential field $P(q)$ as computed above, we need a means to take a step in the direction of the gradient of this field, $\nabla P$, in order to move our character towards its goal. Because of the high dimensionality of the configuration space and the numerous possible ways in which collisions can occur with the environment, using an analytic computation for $\nabla P$ is infeasible. Instead, the RPP method evaluates $P(q + \Delta q)$ for a number of stochastic choices for $\Delta q$. The choice associated with the largest collision-free decrease in value of the potential field, $P$, is accepted and the next gradient descent step can proceed. As will be described shortly, additional mechanisms provide means to escape local minima.

The gradient descent step as described thus far cannot be directly applied to character animation, given that any kind of locomotion requires maintaining contact foothold and handhold constraints with the environment. To address this for single contact configurations, we reroot the skeletal description of the character at the grasping point, allowing the contact constraint to be trivially enforced. Additional contact constraints can be maintained by invoking inverse kinematics to reinstate the given constraints after each stochastic choice of $\Delta q$.

### 3.3 Random walk generator

The gradient descent process is prone to becoming trapped in local minima, given the potential complexities of a human figure moving in its environment. As in [3, 18], we employ random walks to escape these local minima by applying Brownian motion to the character's configuration for a prespecified duration. Given that the first such attempt may not lead to success, the random walk may be performed a number of times. For a thorough discussion of Brownian motion in the context of RPP we refer the reader to [18, 3]. Our implementation of the random walk is as follows: at each step of the walk the current character configuration $q = (q_1, ..., q_j, ..., q_n)$ is modified such that each coordinate $j$ has a uniform chance ($\frac{1}{3}$) of being either increased, decreased, or left unaltered. If the resulting configuration results in a collision with the environment then we discard this choice of $q$ and try again. The amount of increase or decrease in each coordinate $j$ is uniformly distributed over $[0, \Delta q_j)$, where $\Delta q_j$ are precomputed maximal values that ensure that the the character does not penetrate obstacles in the transition between the two configurations.

In the case of deep local minima this tactic can sometimes still prove ineffective. We therefore resort to backtracking, as outlined in the RPP algorithm [3, 18] to deal with this situation. Backtracking consists of restarting the planner at an earlier point along the solution trajectory computed so far. The restart point is chosen randomly with a uniform distribution over the domain of all randomly generated configurations in the current solution, i.e., ones derived from a previous random walk. One reason for choosing from these is that the complement of this set consists of configurations generated by a gradient descent; these configurations are more likely to lie near local minima since each gradient descent unfailingly ends in one. By choosing from the randomly generated set we therefore increase the probability of a successful escape. If no random walks have yet been undertaken, we use the whole solution as the domain for randomly choosing the restart point. Once the character is placed in the restart configuration, a new random walk is performed so that hopefully the character is placed on an alternative slope of $P$, one which will ultimately lead to a different path taken towards the goal. In general, the probability of difficult-to-escape local minima is a function of the frequency of sub-character-sized inter-obstacle gaps, as well as the degree of environment confinement.

Figure 6 illustrates backtracking, using a free-space

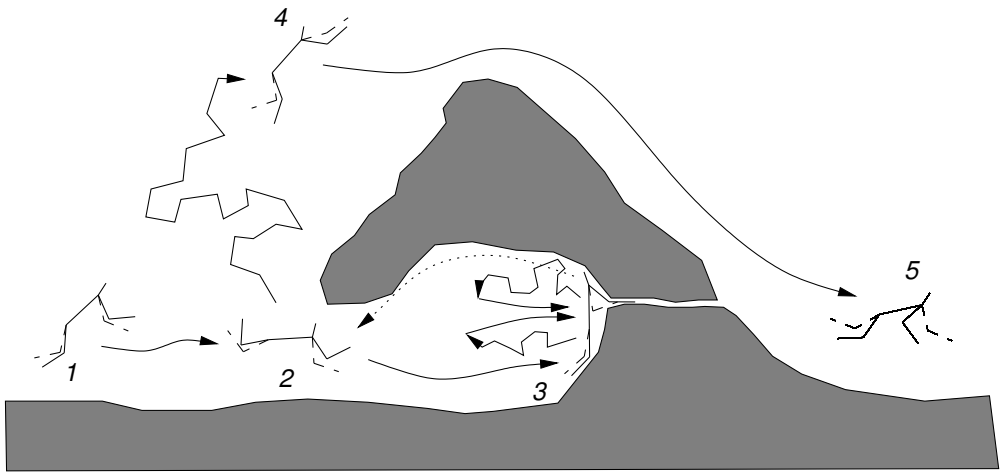


**Figure 6:** Backtracking example (dotted line represents the backtracking operation)

motion for illustrative purposes. The character starts at configuration #1. It floats towards the cave, passing through some configuration (#2), and ends up stuck in a deep local minimum at configuration #3. A number of random walks followed by gradient descents do not yield any progress. The solver then backtracks, randomly choosing configuration #2. A random walk is performed which happens to succeed in escaping the local minimum of the cave (resulting in configuration #4). The character continues using gradient descent until it arrives at the goal, in configuration #5.

## 3.4 The locomotion mode FSM and heuristics

All modes of locomotion, including walking, must continually acquire and release grasp points. Coming up with an appropriate model for this process is critical to the success of the motion planning algorithm. A simple model for acquiring new grasp points would be to do so whenever the opportunity arises, i.e., when a hand or foot is sufficiently close to a new grasp point. In order to release grasp points, an appropriate rule could likewise be defined, such as "release when a grasp point is no longer needed to support the character's motion". Figure 7 illustrates how this process works for a representative walking step.

The simple regrasping procedure described above is problematic in several respects, however. First, the motion produced is largely unnatural, resembling that of a shaky-yet-nimble contortionist leaning forward against the wind. The forward lean is a result of the configuration potential field $P$, which rewards any motion of the center of mass towards its goal position. Thus, the motion displays little regard for gravity and balance. Second, the character will typically move towards its goal in a haphazard fashion as a result of the randomized nature of the path planner. For example, the character may readily use an alternating mix of hands and feet to 'walk' across flat terrain. As unnatural as this is for locomotion across flat terrain, it is worthwhile noting that this kind of unstructured motion may be precisely what is needed in the case of some complex, unstructured environments.

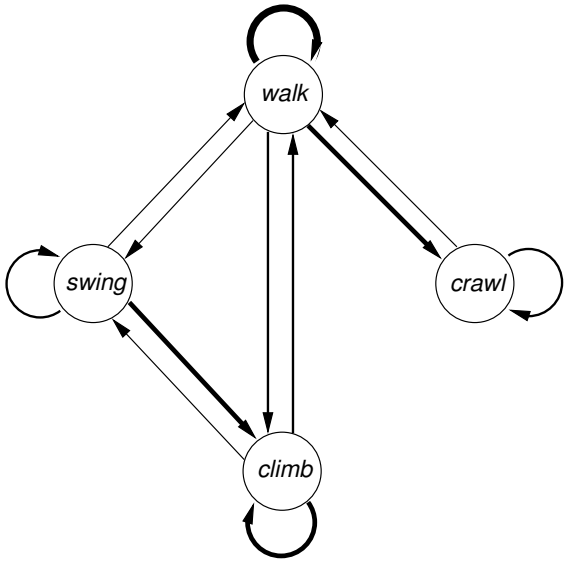


**Figure 8:** The locomotion mode finite state machine; thicker edges indicated higher preference for transition

The problems of unnatural and unorthodox motions are addressed through the use of heuristics and a locomotion mode finite-state machine (FSM), respectively. We first discuss the locomotion FSM.

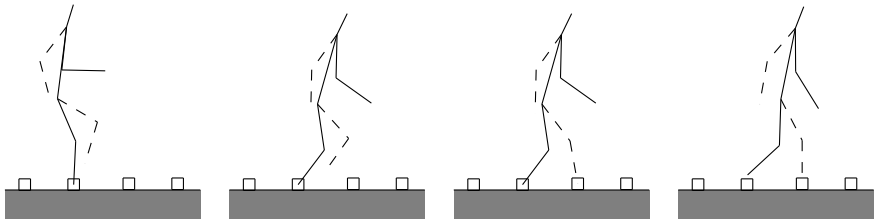

**Figure 7:** The walking cycle; a) starting posture; b) after a few gradient descent steps; c) IK used to reach the next grasp point; d) grasp switched to other leg and gradient descent continued

*Locomotion mode FSM*: Figure 8 shows the FSM, which enumerates the currently available modes of locomotion and defines transitions and preferences among the various modes of locomotion. The edges of the FSM, which represent transitions between modes of locomotion, have associated with them a number of preconditions which must be met in order for the traversal to take place. The preconditions typically consist of a number of geometric constraints that must be satisfied. The edges further specify a set of actions that are to be performed in the event of a transition. These can be as simple as a single change of grasp (acquisition or release), or in more complex cases can consist of a sequence of regrasps and posture corrections. In a limited number of situations, a form of backtracking may also be invoked. In all cases, the actions and their resulting motion consist of the required changes to the character's posture needed to bring it into compliance with the dominant characteristics of the new mode of locomotion.

Of particular note are the self-loops in the graph. Even though these transitions return to the same locomotion mode, they provide the necessary regrasping operations which allow the character to keep advancing using that particular mode. The full details of the locomotion FSM are available in [12].

*Heuristics*: In order to achieve more natural motions, we employ a system of heuristics to guide the character towards desired postures at key points in the solution. We define these key points to be the time instances at which any change of grasp occurs, this being mandated by the finite state machine. Each heuristic analyzes the character's posture and provides feedback on one particular property or characteristic, returning a value ranging from 1 to $+\infty$, 1 being optimal and $+\infty$ being unacceptable. Multiple heuristics are combined into a single *discomfort function* in a multiplicative fashion. To correct a character's posture we employ a stochastic gradient descent procedure, much like that employed for the configuration potential. Table 1 describes which heuristics are used for which modes of locomotion. The details of these heuristics can be found in [12].

| | walk | climb | swing | crawl |
|---|---|---|---|---|
| balance | • | | • | |
| upright_spine | • | | | |
| limb_counter | • | | | |
| comfy_limbs | • | | | • |
| head_up | • | • | • | • |
| hand_down | | • | • | |
| knees_down | | | | • |

**Table 1:** Heuristic usage by locomotion modes.

## 3.5 Motion filters

The system described thus far produces results which still have a serious flaw. The character's motion remains irregular as a result of the stochastic processes used to optimize the character's configuration with respect to both the distance to the goal and the set of posture heuristics. In short, the motion embodies the history of the search process used to produce it, and as a result, does not exhibit the degree of anticipation and fluidity required to achieve natural motions. A separate process is therefore introduced in order to cull any unwanted motion segments as well as optimize the subsequent trajectory, thereby making it more fluid. We refer to this process as "smoothing" or as "the motion filters", and it is carried out on the intermediate solution produced by the planner. The smoothing algorithm we present is borrowed from the work on RPP[3], with modifications necessitated by the addition of grasps, as we shall now explain.

The smoothing process works by attempting to replace portions of the trajectory with a linear interpolation between the starting and ending configurations of that trajectory segment. This strategy works well in smoothing the motion of a free object through a constrained environment, but linear interpolation of joint angles leads to direct violation of grasp constraints in the case of character an-

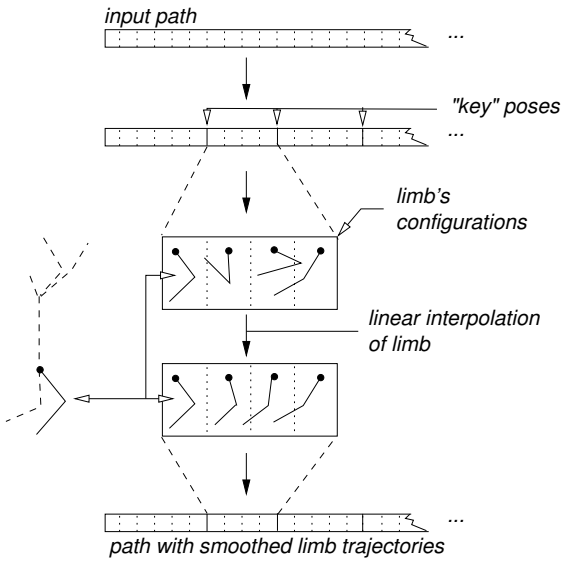


**Figure 9:** The limb trajectory smoothing process; the limb "key" points are ones where the limb's grasping state is changed.

**Figure 10:** A solution prior to smoothing (every 5th frame shown).

imation. Our smoothing process copes with this in three ways. First, smoothing is only applied to portions of the motion trajectory which have no change in grasp configuration. Second, inverse kinematics are used in order to maintain the grasp constraints throughout the interpolated motion. Third, a second smoothing pass is applied independently to each limb, one that only modifies the configuration coordinates which relate to the joint angles of that particular limb. This ensures that the motion of a limb exhibits the desired anticipation in leaving one grasp point and approaching another. Because the second pass treats limb motions independently, changes in grasp configurations for the other limbs are irrelevant, which is not the case for the first pass. The second smoothing pass is illustrated in Figure 9. Figures 10 and 11 show an example solution before and after smoothing.

**Figure 11:** A solution after smoothing (every 5th frame shown).

# 4 Results

Our implemented system is capable of planning motions in complex constrained environments such as that shown in Figure 1. The problem specification for that particular example consists of the starting configuration, located in the bottom left; the target configuration, located in the top right; the character model, as shown in Figure 2; and the polygon-based description of the environment, populated with a large number of grasp points. The planned motion requires 10–15 minutes[2] to compute on a 266MHz Pentium II machine, resulting in about 1400 frames.

Figure 12 shows snapshots from additional motion plans computed by our algorithm and then rendered with a more complex 3D character model. These were rendered with the Poser 4 package, after importing the motion from our planner in BVH format, and applying it to the default character. It should be noted that due to some obvious fundamental differences between the geometries of the two models involved, as well as some difficulties presented by importing environment geometry into Poser, the resulting animations exhibit some obstacle penetration and minute skating problems which are not present in the original motion exported from our planner.

Figure 13 is an illustrative example for the synthesis of a motion transition. The transition from climbing to walking is an interesting problem, as the motion is highly constrained throughout the transition. As the solution shows, the planner can successfully plan a plausible motion which satisfies the required constraints.

MPEGs depicting a sample of obtained solutions for various problems may be viewed online at `http://www.dgp.toronto.edu/~mac/thesis`.

As Figure 1 shows, our results to date have been obtained for scenarios which pose 2D motion planning problems. This is not a general restriction of the planning algorithm, but rather a restriction of our current implementation. The randomized path planning algorithm upon

[2]Note that the compute time can vary significantly due to the non-deterministic nature of the motion planner.

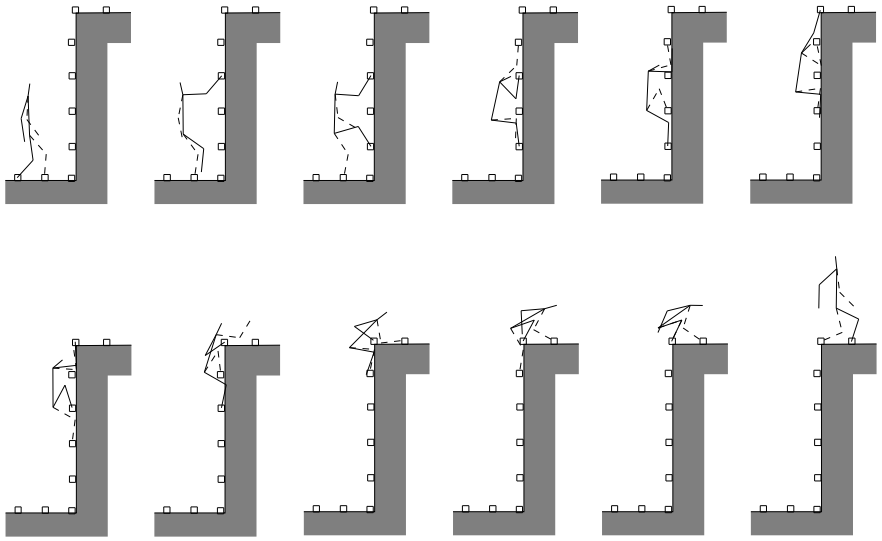

**Figure 13:** Climbing example

which our planner is based has been shown to generalize well to planning motions in 3D environments[3, 18]. We expect that our character motion planning algorithm will scale in a similar fashion. We plan to explore this issue further in future work. The current 2D implementation is still applicable to many interesting scenarios, given the 2D nature of climbing a planar mountain face with grasp points, or moving through an environment such as that illustrated in Figure 1.

# 5 Discussion

In qualitative terms, the motion planner must solve several types of problems. All locomotion modes must make the necessary accommodations to cope with the available grasp locations and variations in the environment. The planner must determine when a change of locomotion mode is justified. The planner must then also synthesize the necessary transitions from one mode of locomotion to another. The planning algorithm described in section 3 serves as a single framework for all of these problems.

What makes the algorithm interesting is that it must tread the line between discrete and continuous optimization problems, given that the choice of grasps is discrete while the remainder of the motion is continuous. Yet, because choices in the continuous domain affect the discrete domain and vice versa, the algorithm must optimize a combined set of discrete and continuous choices. The algorithm also exploits both deterministic and stochastic methods; the FSM and heuristics belonging to the former, and the core parts of the planner, such as gradient descent and random walk, displaying significant amounts of the latter.

*Limitations*: The algorithm as outlined so far has a significant limitation: if presented with an environment where two or more paths lead to the goal, the planner will almost invariably take the shortest path. This is because the downward gradient of the potential field at the "fork in the road" will be always pointing in that direction. Other paths might be taken only if they are roughly of the same length as the shortest one, as this will cause the ridge-like discontinuity in the potential field to be near the bifurcation point, and thus possibly within reach of some random walks.

This limitation can cause the planner to fail in finding a solution, even though one may exist. This can happen if a number of paths to goal are present, but the shortest one is not traversable, such as in the case when there are insufficient grasp points, they are of incorrect types, or the size of the passage is too small or inconveniently shaped. Nevertheless, this problem could be resolved with modest effort. One simple solution would be to mark these "discovered dead-ends" as impassable when they are found, followed by a consequent adjustment to the potential field.

Another minor limitation of our implementation is the situation illustrated in Figure 14. We currently use a very simple 2-link inverse kinematic algorithm for attaching limbs to grasp points. In the course of finding a solution situations may arise where the gradient descent module pulls the character away from potential grasp points, and the simple IK is unable to compensate. Employing a full

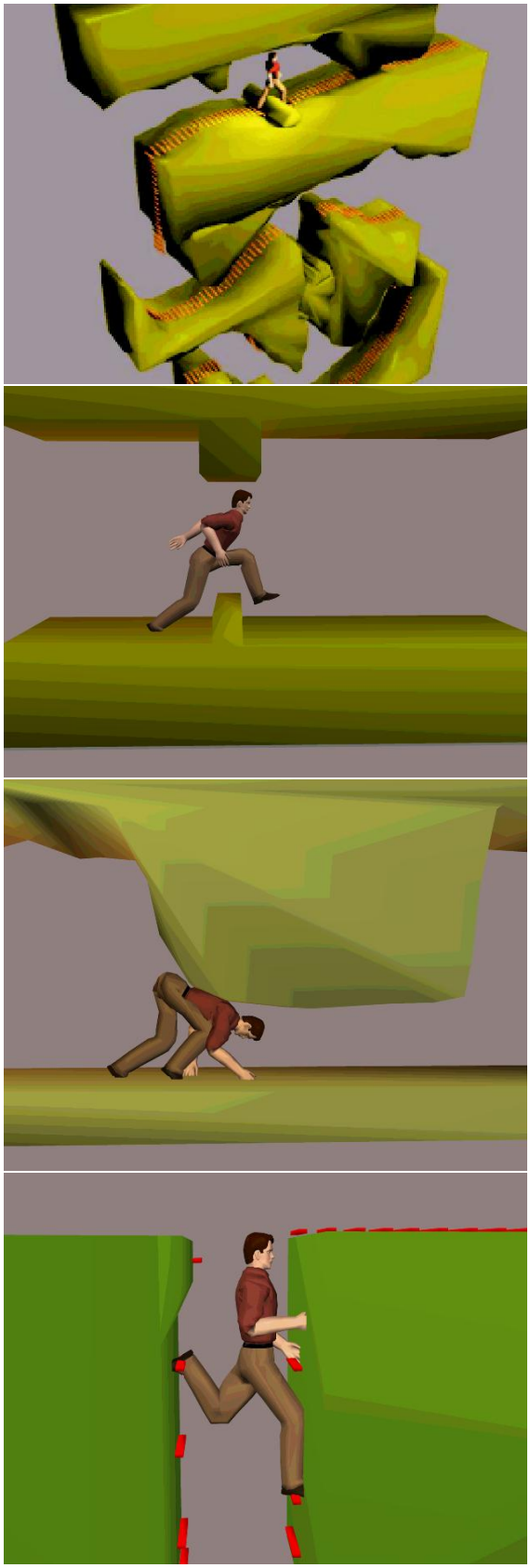

**Figure 12:** Snapshots from several animations.

**Figure 14:** If the overhang is missing (dashed line), the planner tends to get stuck since the character is pulled towards the upper left, away from the grasp points it should be using.

IK engine would solve this problem.

*Complexity*: The above observation results in the categorization of all possible problems into two types: *normal* and *unworkable*. All problems belong to the former except those that cannot be solved due to the untraversability of the shortest paths, which then belong to the latter category.

When operating on a normal problem, the planner is probabilistically complete, much like RPP. That is, if the problem has a solution, the planner will find it given sufficient amount of time, which could be arbitrarily large. The ubiquitous randomness in the planner and the ever-present option of backtracking ensure the possibility of hitting upon just the right trajectory at some point in time. In the unworkable case, our current implementation will never find the solution, and is thus incomplete. If the limitation is remedied as outlined above, the planner would then most likely be probabilistically complete in this case also.

*Operational intuition*: The planner's progress varies with the type of motion it is currently working on. We have found that transitions between locomotion modes, such as from climbing a wall to walking, tend to be the bottlenecks. This is understandable as they are the more constrained and, in general, difficult part of the problem. In many cases though, there is *much* potential for improvement through better formulation of the edge preconditions and actions for a given transition.

The random walk operation plays a much smaller role in our planner than it did in RPP. This is because it is not allowed to acquire or release grasps. With the character anchored to the environment in this manner, the random walk's ability to wander the configuration space is severely limited. It now mainly serves in escaping local minima in the discomfort potential field, when the character's posture is being corrected.

An abundance of grasp points does not adversely affect the planner. Mincing steps are avoided by incorporating a "preferred stride/handspan length" in the appropriate edge preconditions, or by implementing the edge actions in a greedy manner. Inconveniently placed grasp points do not affect the planner very much either, other than lengthening solving time; if a set of grasp points is traversable, backtracking ensures that it will be traversed at some point in time. By the same token, the planner can never commit itself to a dead-end path — the above-mentioned path fork limitation aside — since everything may be rolled back.

# 6 Conclusions

The motion planning algorithm described in this paper provides a novel method for automated character animation. It is particularly well suited for planning motions in unstructured, constrained environments and for generating plausible transitions between various modes of locomotion.

Our work integrates configuration-space planning methods[3, 18] with the requirements of character animation. At the heart of this problem is the question of how to efficiently exploit knowledge of a character's motion preferences while solving potentially complex global motion planning problems. The use of grasp points serves to explicitly model key aspects of the motion, while a collection of heuristics implicitly model motion preferences. A finite state machine is used to imitate the polarization of human motion into distinct locomotion modes.

There are numerous possible directions for improving and extending this work. For one, the animations generated occasionally exhibit unstable or gravity-defying postures. We plan to construct some better heuristics for the imitation of gravitational pull on suspended characters, as well as a method for prioritizing the various heuristics to give them varying importance.

Given that our planner has no explicit notion of time nor speed, we currently perform a one-to-one mapping between the configurations of the solution path and the keyframes used in playback. This results in undesirable discontinuities in the speed of the motion. The results could be made more fluid by varying the mapping such that the playback speeds change in a manner appropriate to the situation.

A minor limitation in our planner is that only the hands and feet are allowed to grasp. Although this is typically sufficient, there are motions which require more complex grasps. Two examples of this are the using the posterior as a support when sliding on the floor, and the leaning of the back and shoulders against a wall when wriggling up a narrow crevice. These types of motions cannot be employed by the planner at this point in time.

Further improvements in the planner could be obtained by the judicious use of machine learning algorithms in various parts of our method. Some obvious applications would be the optimization of calibration values in the various heuristics, as well as the memorization and prediction of postures in commonly occurring situations.

# References


[1] Kenji Amaya, Armin Bruderlin, and Tom Calvert. Emotion from motion. In *Graphics Interface '96*, pages 222–229, May 1996.

[2] Norman I. Badler, Cary B. Phillips, and Bonnie L. Webber. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, 1993.

[3] Jérôme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, December 1991.

[4] R. Boulic, N. M. Thalmann, and D. Thalmann. A global human walking model with real-time kinematic personification. *The Visual Computer*, 6:344–358, 1990.

[5] A. Bruderlin and T. W. Calvert. Goal-directed animation of human walking. *Proceedings of ACM SIGGRAPH*, 23(4):233–242, 1989.

[6] Armin Bruderlin and Tom Calvert. Knowledge-driven, interactive animation of human running. In *Graphics Interface '96*, pages 213–221, May 1996.

[7] Armin Bruderlin and Lance Williams. Motion signal processing. In *Computer Graphics Proceedings*, Annual Conference Series, pages 97–104. SIGGRAPH, 1995.

[8] Motion Factory. Motivate 3D Game Development System. `http://www.motionfactory.com/`.

[9] M. Girard. Interactive design of computer-animated legged animal motion. *IEEE Comptuer Graphics and Applications*, 7(6):39–51, June 1987.

[10] Michael Gleicher. Retargetting motion to new characters. In *Computer Graphics Proceedings*, Annual Conference Series, pages 33–42. SIGGRAPH, 1998.

[11] J. K. Hodgins. Simulation of human running. *Proceedings, IEEE International Conference on Robotics and Automation*, pages 1320–1325, 1994.

[12] Maciej Kalisiak. A grasp-based motion planning algorithm for intelligent character animation. Master's thesis, Uni-

versity of Toronto, 1999. Available online at http://www.dgp.utoronto.ca/~mac/thesis.

[13] Hyeongseok Ko and Norman I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Proceedings of Graphics Interface '92*, pages 273–281, 1992.

[14] Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. Planning motions with intentions. In *Computer Graphics Proceedings*, Annual Conference Series, pages 395–408. SIGGRAPH, 1994.

[15] Yotto Koga, Geoff Annesley, Craig Becker, Mike Svihura, and David Zhu. On intelligent digital actors. http://www.motionfactory.com/products/whppr_imagina.htm.

[16] James Kuffner, Jr. *Autonomous Agents for Real-Time Animation*. PhD thesis, Stanford University, 1999.

[17] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. In *Computer Graphics Proceedings*, Annual Conference Series, pages 155–162. SIGGRAPH, 1996.

[18] Jean-Claude Latombe, Cary B. Phillips, and Bonnie L. Webber. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[19] Philip Lee, Susanna Wei, Jianmin Zhao, and Norman I. Badler. Strength guided motion. In *Computer Graphics*, volume 24, pages 253–262. SIGGRAPH, 1990.

[20] Cary B. Phillips and Norman I. Badler. Interactive behaviors for bipedal articulated figures. In *Computer Graphics*, volume 25, pages 359–362. SIGGRAPH, July 1991.

[21] Zoran Popovic and Andrew Witkin. Physically based motion transformation. *Proceedings of SIGGRAPH 99*, pages 11–20, August 1999.

[22] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In *Computer Graphics*, volume 25, pages 349–358. SIGGRAPH, July 1991.

[23] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Proceedings of SIGGRAPH 96*, pages 147–154, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[24] Nick Torkos and Michiel van de Panne. Footprint-based quadruped motion synthesis. In *Proceedings of Graphics Interface '98*, pages 151–160, 1998.

[25] Transom Technologies. Ann Arbor, Michigan. http://www.transom.com.

[26] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Computer Graphics Proceedings*, Annual Conference Series, pages 91–95. SIGGRAPH, 1995.

[27] Michiel van de Panne. From footprints to animation. In *COMPUTER GRAPHICS forum*, volume 16, pages 211–223, 1997.

[28] A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics*, volume 22, pages 159–168. SIGGRAPH, August 1988.

[29] Andrew Witkin and Zoran Popović. Motion warping. In *Computer Graphics Proceedings*, Annual Conference Series, pages 105–108. SIGGRAPH, 1995.