In Proceedings of Motion in Games (MIG), 2015. This is the manuscript of the authors.

Multi-Modal Data-Driven Motion Planning and Synthesis

Mentar Mahmudi* Marcelo Kallmann†

*École Polytechnique Fédérale de Lausanne †University of California Merced

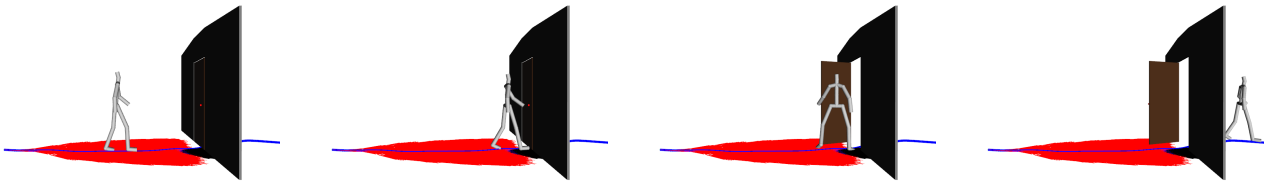

Figure 1: *Our planner produces collision-free walking motions allowing reaching the handle, opening the door, and passing through it.*

Abstract

We present a new approach for whole-body motion synthesis that is able to generate high-quality motions for challenging mobile-manipulation scenarios. Our approach decouples the problem in specialized locomotion and manipulation skills, and proposes a multi-modal planning scheme that explores the search space of each skill together with the possible transition points between skills. In order to achieve high-quality results the locomotion skill is designed to be fully data-driven, while manipulation skills can be algorithmic or data-driven according to data availability and the complexity of the environment. Our method is able to automatically generate complex motions with precise manipulation targets among obstacles and in coordination with locomotion.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation.

Keywords: character animation, motion capture, motion planning.

Links: DL PDF

1 Introduction

High quality human-like character animation has attracted a great body of research in the past decades. Although many methods are able to successfully generate motions for a variety of problems, more work is needed in order to improve the capability of generating complex motions without impacting the realism of the obtained solutions.

Improving the capability of handling complex tasks around obstacles and at the same time obtaining realistic results is particularly challenging because improving one aspect often comes at the expense of deteriorating the other. Our present work is sensitive to this trade-off and seeks to achieve a balance between the involved constraints.

Previous methods in computer animation have rightfully put a strong emphasis on realism. High fidelity, realistic and human-like motions can be produced with the employment of data-driven methods which reuse and combine motion capture data in order to achieve high quality results. When properly designed, these methods can automatically provide a wide range of motions from a finite set of previously captured motion capture data. Their main advantage is that they are generic and work with different styles and types of motions.

Although pure data-driven methods can easily synthesize highly realistic motions, several complex tasks among obstacles will either require a high amount of specific input motion to be collected and included, or the integration of dedicated algorithmic solutions in order to augment the capabilities of the original data-driven technique.

A typical example addressed in this paper is the problem of a character having to walk to a door, open it, and then walk through it without collisions. Given the narrow passage to be addressed it is difficult to utilize a generic data-driven method to solve such types of problems. Moreover, finding valid solutions as the user changes the position of the door handle relative to the door, or even how wide the door is, can make the problem increasingly more difficult to be solved. Even when additional example motions can be considered, data-driven methods typically do not scale well with the size of their motion capture databases.

Our proposed planner addresses these issues by decomposing the overall problem into sub-tasks, or modes of execution, which can be solved by specialized motion skills. Our method is a hybrid approach that still produces high quality results because it is highly based on a data-driven locomotion skill; but it also combines motion primitive skills based on specialized algorithmic planners. Our planner can be used in such a way that if a motion capture database is enough to represent the solution space of the task at hand, then a pure data-driven solution might be found. At the same time, if the problem is too complex or the data-driven skills are not sufficient, algorithmic skills will be automatically used and multiple combinations of skills will compete with each other until a solution is found.

Our multi-modal planner is therefore a global planner able to coordinate different motion skills in order to solve generic mobile-manipulation problems. Our method is able to solve complex tasks

*e-mail:mentar.mahmudi@epfl.ch (work partially done while at UCM)
†e-mail:mkallmann@ucmerced.edu

among obstacles and is particularly able to automatically control locomotion towards places that allow successful manipulations in coordination with locomotion.

2 Related Work

There have been two major groups of approaches to address the problem of realistic human-like motion generation for virtual characters: data-driven and physics-based methods. Physics-based methods, although more involved to be implemented and controlled, provide the ability to react to unforeseen events in the environment by taking into account external forces and automatic reactions to it.

One relevant work is the work of Bai et al. [2012], which introduced a method for planning actions involving concurrent object manipulations. Physics-based controllers are used to handle the coordination between the concurrent motions; however, without focus on solving complex manipulations such as the ones we address. Our proposed planner is mostly related to data-driven methods and we therefore focus on analyzing related data-driven methods.

The first methods designed to re-use data in a motion capture database were introduced a decade ago [Kovar et al. 2002; Lee et al. 2002; Arikan and Forsyth 2002; Arikan et al. 2003]. These works introduced the motion graph data structure which is able to automatically create transitions between similar frames in the database. Using a search or optimization process, they were able to generate motions which were numerically similar but semantically different from the ones in the original database.

Many variants of motion graphs were then developed to improve different aspects of the method. Parametrized motions clips and techniques to transition between blended clips [Shin and Oh 2006; Heck and Gleicher 2007] made possible for fast motion synthesis; however, did not easily allow for planning motions in environments with many obstacles. Time interpolation of motion paths in order to improve the solution space of motion graphs [Safonova and Hodgins 2007] improved the range of possible solutions but at the expense of increasing the size of the motion graph. Furthermore, complex tasks would require interpolation of more than two motion paths, which would make the approach even more impractical. Further improvements were proposed [Zhao and Safonova 2008; Zhao et al. 2009] but mostly to construct motion graphs with improved connectivity. None of these methods have been used for solving mobile-manipulation tasks.

Other approaches, such as the work of Lau and Kuffner [2006], have relied on precomputation and on a finite state machine of behaviors in order to obtain a search tree for solving locomotion tasks. Similarly, motion maps [Mahmudi and Kallmann 2012] were precomputed for all the nodes of a motion graph in order to achieve interactive motion synthesis from an unstructured motion capture database. While these works can efficiently solve locomotion problems, none of them have been extended to solve mobile-manipulation tasks.

Blending motion examples in order to generate in-between examples is an effective way to solve object manipulation problems. Kovar et al. [2004] automatically extracted similar motions from large databases to then build parameterized motion skills. Rose et al. [1998] constructed blend spaces with radial basis functions in order to generate motions of different styles. Basten et al. [2011] built a coordination model which realistically combined spliced motions. While blending can generate a variety of motions, integration with a locomotion planner has been only addressed once [Huang et al. 2011]. Our present work improves the approach and effectively integrates heterogeneous motion skills in a single global planning framework.

Another important approach is to employ reinforcement learning methods. Treuille et al. [2007] and Levine et al. [2011] have employed learning in order to achieve complex real-time characters controllers. Although reinforcement learning is a promising direction to generate controllable motion, it is time-consuming to learn control policies and controllers have to remain in a low dimensional space. While reinforcement learning is suitable for real-time control of characters, it does not directly address planning long sequences of motions for solving complex tasks. In such cases, planning approaches are more suitable.

Relevant planning approaches have combined PRMs and motion capture in order to address biped locomotion among obstacles [Choi et al. 2002]. Yamame et al. [2004] employed planning for solving various object manipulations. Pan et al. [2010] combined a hierarchical model decomposition with sampling-based planning. While these methods have generated impressive results, they have not incorporated the ability to search for competing solutions involving different combinations of locomotion and manipulation in order to solve complex mobile-manipulation scenarios.

Several planning methods have been developed in the robotics domain. In particular, a multi-modal planning approach was used by Hauser et al. [2007] to generate motions for humanoid robots. Kallmann et al. [2010] introduced the coordination of parameterized motion skills to generate mobile manipulations. These works however did not include data-driven techniques.

In summary our approach reaches a unique balance between high capability of planning complex solutions among obstacles while maintaining the quality of data-driven methods.

3 Multi-Modal Planner

Our planner Π operates over a set Σ of parametrized motion skills. We represent a motion skill as $\sigma_i \in \Sigma$. Each invocation of a motion skill σ_i generates a motion m_i. The global solution is produced by appending and/or superimposing the partial motions m_i. Motion skills produce partial motions according to parameters π_i:

$$\sigma_i(\pi_i) = m_i. \tag{1}$$

Motion m_i is encoded as a l by c keyframe matrix where l is the number of frames and c is the dimensionality of the controlled configuration space of a standard hierarchical character. Π also maintains a list of modes $\theta \in \Theta$, a function τ specifying mode transitions, and an activation function λ which determines which motion skills can be activated from any given mode:

$$\tau : \Theta \rightarrow \Theta, \tag{2}$$

$$\lambda : \Theta \rightarrow \Sigma. \tag{3}$$

The global planner $\Pi(\Sigma, \Theta, \lambda, \tau)$ operates by expanding motion skills and combining their partial motions until a solution is found. The planner performs a multi-modal search which starts from the initial location of the character and proceeds until the ground-projected root position of the character is close enough to a given target goal point. The search procedure maintains a search tree, and the search frontier is prioritized using a A*-like heuristic [Hart et al. 1968] based on the given goal location.

Each skill is responsible for generating their partial motions m_i, which are supposed to be short segments of valid motions. On each

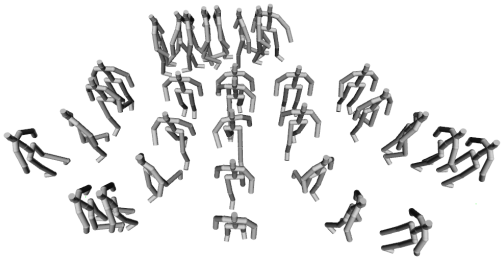

Figure 2: *Motion capture database used by the locomotion skill.*

iteration during the search the planner retrieves the lowest cost configuration in the current search frontier, invokes each skill possible to be activated (using λ), and concatenates and/or superimposes the returned partial motions, updating the search tree accordingly. A motion skill is allowed to return a null partial motion in order to indicate that it was unable to generate a partial motion, what can happen due collisions or any other constraints which were not possible to be met.

This overall approach allows motion skills to execute their own local planners for generating partial motions while the search tree in Π will make skills compete with each other during the search for a global solution. This approach makes our planner able to solve complex problems that cannot easily be handled with traditional methods.

Since in this work our multi-modal planner is applied to mobile manipulation tasks, it is important to rely on a well-designed locomotion skill able to move the character around and to prepare the character for mode transition into manipulation skills.

3.1 Locomotion Skill

Our locomotion skill σ_{loc} is based on a feature-based motion graph [Mahmudi and Kallmann 2013; Mahmudi and Kallmann 2011] and operates on the entire configuration space of the character. The input motion capture database is shown in Figure 2. It contains only locomotion and does not include motions such as opening doors or picking up objects. After the motion graph is built we use the connectivity of the graph to determine the parameter list of the skill and also which modes may be invoked in from the locomotion mode.

A locomotion skill can transition to another skill if transitional constraints can be satisfied. The transfer skill tests at each step whether an object of interest such as a door or a shelf is within the close vicinity of the character. If that is the case, the activation function λ skill will invoke all available manipulation skills that can be activate and that will lead to allowed modes according to τ.

The test performed by λ is mostly a fast check if the character is close enough to an object accepting manipulation. If a manipulation skill is successful in generating a partial motion, the partial motion is integrated in the corresponding search branch of Π, and the mode of the branch leaf is switched for the new one.

The mode transitions encoded in τ will ensure that consistent manipulations are achieved. For example, by specifying that after a reaching mode we must have an opening mode we avoid generating motions where the character would reach the door handle and then release it without opening the door. Function τ therefore ensures correct sequences of motion primitives to be activated.

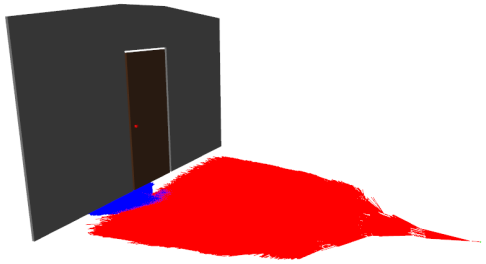

Figure 3: *Example of a typical multi-modal search tree. Red segments represent unmodified segments generated by the locomotion skill and blue segments represent motions modified by reaching skills. The segments are ground-projections of the wrist (blue) or root (red) joints, taken at the extremes of each corresponding partial motion in the global search tree.*

Overall, the locomotion skill will be expanded towards multiple locations around the object to be manipulated, generating several possible locomotion branches to be considered for starting manipulations. Multiple manipulation skills will be invoked from the different branches, until eventually a solution is found. See Figure 3. If additional time is available, multiple solutions can be also obtained.

Every time Π expands a branch of the search tree in locomotion mode, the locomotion skill is called with parameter $\pi_{loc} = (n, j, \alpha)$, where n is the current node of the motion graph to be expanded n, j is the maximum number of children nodes of n to be expanded, and α is a deformation vector indicating rotation values to deform each branch of the motion graph expansion in order to augment the number of generated partial motions.

Deformations are performed in order to improve the number and resolution of the generated locomotion segments, greatly improving the planner capability to find solutions in complex scenarios.

Given an original locomotion segment m associated with a node n of the motion graph, m is deformed into m_i motions according to a list of α_i deformation angles. The deformed motions will be then appended to n in the corresponding search tree branch.

For a motion m and a deformation angle β, each frame m_j of m is deformed taking into account the root position p_j and orientation q_j of the character at frame j, and the position differential vector v_j such that $v_j = p_{j+1} - p_j$. We then create a quaternion rotation r_j representing the rotation of β about the positive Y axis (which is perpendicular to the floor plane). Then, for each frame j, the orientation q_j is incremented by r_j, and the position p_{j+1} is updated to $p_j + v_j$. In order to achieve improved variations, the deformation is applied iteratively. In the first iteration all the frames of m are deformed. In the next iteration, only frames between the second and the penultimate frame are deformed. See Figure 5 for an example.

In order to limit deformation amounts we provide a method to quantify the effect of a deformation. The most undesirable artifact of deformation is feet sliding. Therefore, we observe the toe and heel joints of the skeleton and measure the averaged squared distances of the corresponding toe and heel joint positions in the original and the deformed motions. Figure 4 shows an example. We can see that for the original motion (red points) the heel and toe joints appear close to a point. As the deformation is increased (green and blue) the heel and toe joints start drifting further away from their initial fixed positions. The deformation quality is defined as the weighted sum of squared distance averages. A deformed segment is then only kept if the quality metric stays under a pre-specified thresh-

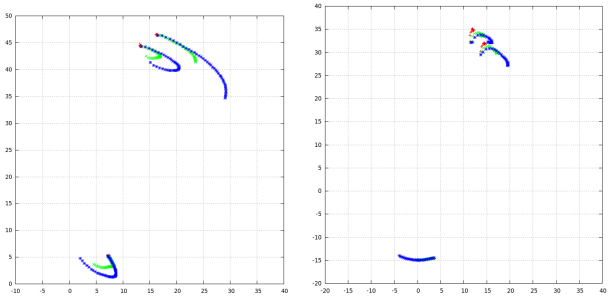

Figure 4: *Left: Heel joint position projection for three stepping motions. Red points represent the original motion, the green points represent a deformed motion with $\alpha = 0.5^{\circ}$ and blue represents a deformed motion with $\alpha = 1.0^{\circ}$. Right: Same comparison as in the left image, however, in this instance for the toe joint.*

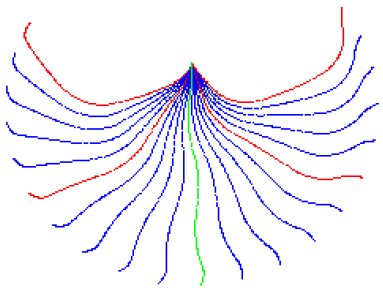

Figure 5: *Motion deformation at various angles. The green projection is the original motion, and the blue and red deformations show the range of the motion when deformed between -1 and 1 degrees.*

old. In our experiments we have deformed each node of the motion graph by $\alpha = \{-0.2, -0.1, 0.1, 0.2\}$ and used a very conservative quality threshold which would not allow for visually disturbing feet sliding.

3.2 Reach and Release Skills

The reach σ_{reach} and release $\sigma_{release}$ skills are based on a time-parameterized RRT planner. These skills control only the arm of the character. The rest of the character's posture is defined by the locomotion, therefore reaching and releasing motions are superimposed to the last locomotion segment in the search branch being expanded.

The skills take two parameters $\pi_{reach} = \pi_{release} = (s, s')$. The first parameter s comes directly from the first posture of the locomotion segment to be superimposed. The second parameter s' is the posture we wish the character to assume at the end of the motion generated by σ_{reach} or $\sigma_{release}$. This is usually a posture where the character is considered touching or holding the target manipulation location, as for example a door handle, a button, or a book. Posture s' is determined with the use of an analytical IK solver for the arm of the character. Given a target position p and orientation q at the manipulation point, the last frame of the locomotion is updated with IK to have the hand of the character to reach the desired target location $t = (p, q)$. The value differences of the modified arm joints are decreasingly distributed to previous frames in order to achieve a smooth arm movement towards s'. Skills σ_{reach} and $\sigma_{release}$ therefore generate arm motions on top of locomotion sequences, achieving simultaneous walking and target reaching.

3.3 Displacement Skill

The displacement skill σ_{disp} is implemented similarly to σ_{reach} and $\sigma_{release}$, however it allows for a number of frames in the resulting motion to have the hand of the character attached to a moving target location. This allows for the arm of the character to follow the movements of an object. This is exactly the case needed for achieving a door opening type of movement. The same IK solver is used to compute the several arm postures needed to follow the desired movement with the hand.

Displacement skills can be applied on top of the locomotion or while the character stands still; however, a special collision avoidance procedure has been integrated in order to allow our examples, in particular the door opening cases, to be successfully computed. Our collision avoidance procedure is applied to the torso joints of the character by small amounts at each frame, such that the spine moves in the opposite direction to the object being manipulated. The result not only looks more natural but at the same time makes it possible for the character to fully open the door in many cases that wold otherwise create collisions.

3.4 Generic Action Skills

Generic action skills can be easily added to our framework. For instance, a number of actions can be implemented based on example motions collected from motion capture, achieving realistic motions for pointing, pouring, kicking, etc., which can be implemented by a number of data-driven techniques [Rose et al. 1998; Kovar and Gleicher 2004; Huang et al. 2011]. Our overall planner is generic enough to incorporate generic actions in order to achieve complex multi-modal motions.

4 Results

For our experimental setup we have built a motion graph from a simple motion capture database which contained 6 motions: 1 straight walking motion, 1 gentle left turn, 1 gentle right turn, 1 sharp left turn, 1 sharp left turn and 1 U-turn motion. The motions were captured using 18 Vicon MX cameras in an environment of 5.5m by 4m. All the motion were sampled at 60Hz. The total number of frames in the database was 1543 which corresponded to 25.7s of total motions. No door opening, book picking or lamp motions where recorded. An snapshot depicting our database is shown on Figure 2. The initial motion graph contained 125 nodes. The maximum average transition threshold was set to 6cm. When deformations were applied to all nodes the graph increased to 625 nodes. This motion graph was only used by skill σ_{loc}.

4.1 Door Opening

Opening doors represent a particularly challenging problem for data-driven methods because they cannot easily accommodate slight variations in the character's position during motion synthesis, and these variations are important to solve problems involving narrow passages. In our approach we add deformations to the motion graph and decouple manipulation motions from the locomotion in order to successfully solve door opening problems.

The problem is encoded in four different modes: locomotion mode, handle reaching mode, door opening mode, door release mode and back to locomotion mode. The user specifies the start and goal positions for the character to be on the opposite sides of the door. The skill set Σ for the door opening problem contains the following

parametrized motion skills: $\Sigma = \{\sigma_{loc}, \sigma_{reach}, \sigma_{disp}, \sigma_{release}\}$.

At the beginning, for each locomotion segment expanded, planner Π repeatedly tests if a transition to the next mode is possible. This involves IK reaching tests in order to reach for the handle of the door. After the handle is reached, σ_{disp} will have the arm of the character to follow the door opening motion, until the door cannot be further opened without breaking contact with the handle, or until a (self-)collision is detected. The goal is to open the door as much as possible without introducing collisions. Whether the character can successfully walk through the door in the next mode transition will be determined by the search expansion of Π in the following iterations. Many different alternatives involving varied body positions are evaluated by the planner, until the right combination of skills is found. See Figure 6 for an example.

Solutions also depend on the collision avoidance mechanism which rotates the torso of the character while the displacement skill is trying to open the door. The introduced rotation improves the chances of opening the door enough for the character to pass through it.

Skill $\sigma_{release}$ works similarly to σ_{reach}; however, its function is to move the arm from a posture where the arm is holding the handle to a posture back to the last frame of the current locomotion segment generated by σ_{loc}. This brings back the character to locomotion mode where it can then attempt to pass through the door.

As a result the overall planner invokes several skills maintaining a multi-modal search tree where the sorted frontier is associated with an A*-like cost in order to order the next nodes to be expanded. One technique that can be used to improve the performance of the overall procedure is to implement a grid on the floor plan and keep track of the number of nodes that reach each cell of the grid. By limiting a maximum number of nodes per cell it is possible to cull branches of the search concentrating in only one region, thus forcing the search to also visit nearby positions. The culling parameter (maximum number of nodes per cell) provides a mechanism to explore the trade-off between exploration and exploitation that arises during the search and should be tuned according to the difficult of the problem, which is mainly related to how narrow is the door passage.

4.2 Book Reaching and Relocation

In this scenario the character walks towards a bookshelf and tries to pick up a book and then replace it to another location on the shelf. The motion skills used in this example are the same as the ones explained for the door opening case. This shows that our primitive motion skills are generic enough to be used for a variety of mobile-manipulation tasks.

The main difference with respect to the door opening example is that σ_{disp} is used to relocate books instead of opening doors. The key difference is that, instead of following with the hand a fixed door motion, σ_{disp} can generate any collision-free trajectory that is able to relocate the book from its initial position to the new position specified on the shelf. In the presented book manipulation examples we have integrated our bi-directional RRT planner in σ_{disp}, therefore being able to compute generic collision-free trajectories for relocating the book. Whenever a relocation trajectory cannot be found, the skill returns null, which will force the planner to expand other branches of the search tree and eventually find other body placements that will allow the displacement to be successfully computed.

The overall task of the character is to first approach the bookshelf to a distance where it can reach the book, then it will pick the book from the shelf and move it to a new place specified on the shelf, while avoiding collisions with other books and items on the shelf. The character will then release the book it may then go back to locomotion and perform any additional tasks.

Several book manipulation scenarios could be successfully solved with our planning framework. See Figure 7 for an example.

5 Conclusion and Future Work

The presented results demonstrate that our multi-modal planner is able to successfully coordinate skills in order to solve complex mobile-manipulation problems. This work also includes an accompanying video which presents several additional results obtained with our planner.

A number of areas deserve consideration for future work, as for example, integration of stepping motions for supporting manipulation, and coordination rules between primitives, in order to improve the task-level quality of solutions.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Synthesizing constrained motions from examples. *Proceedings of SIGGRAPH 21*, 3, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *Proceedings of SIGGRAPH 22*, 3, 402–408.

BAI, Y., SIU, K., AND LIU, C. K. 2012. Synthesis of concurrent object manipulation tasks. *ACM Trans. Graph. 31*, 6 (Nov.), 156:1–156:9.

CHOI, M. G., LEE, J., AND SHIN, S. Y. 2002. Planning biped locomotion using motion capture data and probabilistic roadmaps. *Proceedings of SIGGRAPH 22*, 2, 182–203.

HART, P., NILSSON, N., AND RAPHAEL, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2, 100–107.

HAUSER, K. K., NG-THOWHING, V., GONZALEZ-BAOS, MUKAI, H., AND KURIYAMA, S. 2007. Multi-modal motion planning for a humanoid robot manipulation task. In *International Symposium on Robotics Research (ISRR)*.

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*, ACM Press, New York, NY, USA, 129–136.

HUANG, Y., MAHMUDI, M., AND KALLMANN, M. 2011. Planning humanlike actions in blending spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

KALLMANN, M., HUANG, Y., AND BACKMAN, R. 2010. A skill-based motion planning framework for humanoids. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transaction on Graphics (Proceedings of SIGGRAPH) 23*, 3, 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. H. 2002. Motion graphs. *Proceedings of SIGGRAPH 21*, 3, 473–482.

LAU, M., AND KUFFNER, J. J. 2006. Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, 299–308.

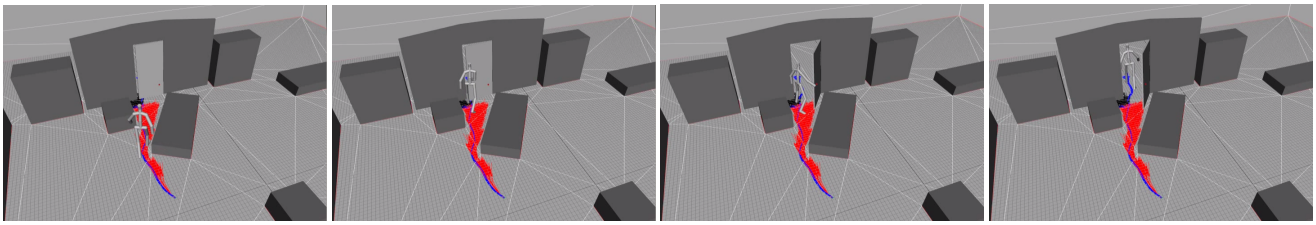

Figure 6: *Similar door opening example as in Figure 1 but with obstacles in front of the door.*

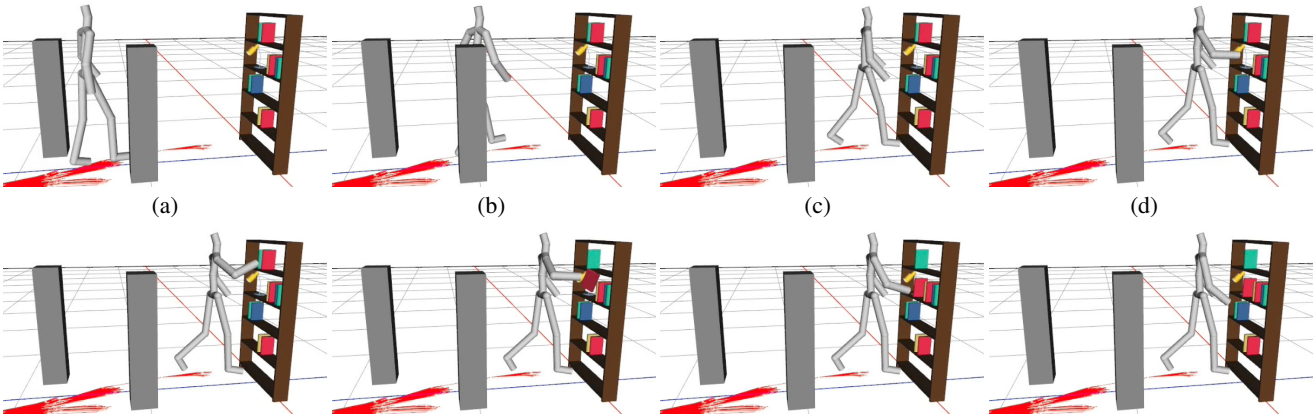

Figure 7: *Book reaching and relocation. The character begins by walking towards the shelf (a), planning its motion around obstacles with the locomotion skill (b), until a suitable position for starting a book reaching motion (c). The reaching skill then moves the arm (d), in parallel with the locomotion, until the book is reached (e). Then, the reaching skill is again employed to relocate the book (f) to a new location on the shelf (g), where the book can be finally released (h). All motions are collision-free.*

LEE, J., CHAI, J., REITSMA, P., HODGINS, J. K., AND POLLARD, N. 2002. Interactive control of avatars animated with human motion data. *Proceedings of SIGGRAPH 21*, 3 (July), 491–500.

LEVINE, S., LEE, Y., KOLTUN, V., AND POPOVIĆ, Z. 2011. Space-time planning with parameterized locomotion controllers. *ACM Trans. Graph. 30*, 3 (May).

MAHMUDI, M., AND KALLMANN, M. 2011. Feature-based locomotion with inverse branch kinematics. In *Proceedings of the 4th International Conference on Motion In Games (MIG)*.

MAHMUDI, M., AND KALLMANN, M. 2012. Precomputed motion maps for unstructured motion capture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, 127–136.

MAHMUDI, M., AND KALLMANN, M. 2013. Analyzing locomotion synthesis with feature-based motion graphs. *Visualization and Computer Graphics, IEEE Transactions on 19*, 5, 774–786.

PAN, J., ZHANG, L., LIN, M., AND MANOCHA, D. 2010. A hybrid approach for synthesizing human motion in constrained environments. In *Conference on Computer Animation and Social Agents (CASA)*.

ROSE, C., BODENHEIMER, B., AND COHEN, M. F. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 32–40.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (Proceedings. of SIGGRAPH) 26*, 3.

SHIN, H. J., AND OH, H. S. 2006. Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation (SCA)*, 291–298.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. In *Proceedings of ACM SIGGRAPH*, ACM Press.

VAN BASTEN, B. J. H., AND EGGES, A. 2011. Flexible splicing of upper-body motion spaces on locomotion. *Comput. Graph. Forum 30*, 7, 1963–1971.

YAMANE, K., KUFFNER, J. J., AND HODGINS, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 23*, 3, 532–539.

ZHAO, L., AND SAFONOVA, A. 2008. Achieving good connectivity in motion graphs. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation (SCA)*, 127–136.

ZHAO, L., NORMOYLE, A., KHANNA, S., AND SAFONOVA, A. 2009. Automatic construction of a minimum size motion graph. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.