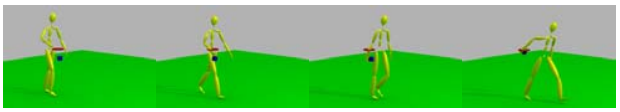
Autonomous Object Manipulation for Virtual Humans



Marcelo Kallmann
mkallmann@ucmerced.edu
http://graphics.ucmerced.edu

UCMERCED

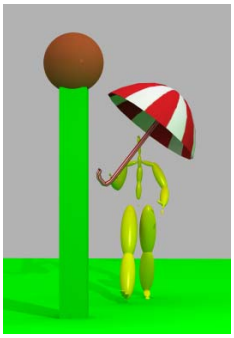
Motion Planning and Autonomy for Virtual Humans
SIGGRAPH 2008 Class Notes

Welcome to the “Autonomous Object Manipulation” section of the class “Motion Planning and Autonomy for Virtual Humans”.

Motivation

- How to algorithmically plan and synthesize complex humanoid motions for object manipulation?





M. Kallmann 2008 - UCM 2

Object manipulation for humanlike characters is a full-body motion planning problem. In order to perform humanlike tasks, body positioning and balance have to be planned in coordination with arm motions. The images in this slide illustrate three example situations which are covered in this course.

The left image illustrates the most common case of object manipulation, which is to reach, grasp and relocate objects. Traditionally this problem is solved with the use of Inverse Kinematics (IK) and collision avoidance, however when several obstacles are on the way sampling-based planners become powerful needed tools.

The center image shows the importance of planning body placement prior to reaching for a door handle: the chosen location of the body has to ensure that the handle is reachable and at the same time that there is enough space left to open the door without colliding it with the body of the character. We refer to this problem as planning the sequencing of different primitive motions, in this case: locomotion and reaching.

The image on the right illustrates the problem of planning two primitive motions concurrently. In the example, an arm motion is planned to manipulate the umbrella in synchronization with a dominant walking motion in order to prevent the umbrella from colliding with the post.

Applications

- Autonomous Virtual Humans
  - For VR applications and Games
- Animation Tools
  - To assist designers to rapidly animate characters
- Ergonomics
  - For studying environments designed for humans
- Robotics
  - For achieving autonomous humanoid robots

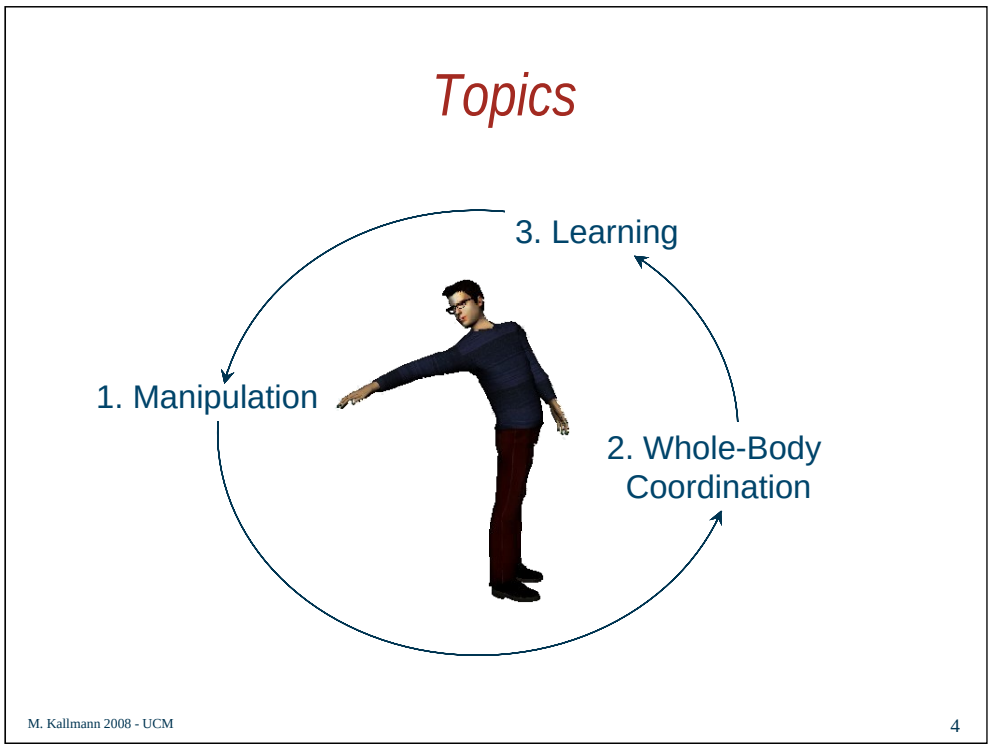
M. Kallmann 2008 - UCM 3

The main goal of motion planning is to empower autonomous and interactive characters with the ability to algorithmically compute their own motions. Therefore the main applications are for VR and Games. However it is important to note that most of current planning methods cannot easily plan motions in real-time and it is also difficult to predict the computation time required for each given planning problem.

This situation leads to a new aspect to be considered when developing new motion planners, which is to target human-like performances where longer computation times are acceptable for problems that are also difficult for humans to solve. Such approach should also integrate reactive behaviors in order to achieve human-like performances.

Although real-time performance is hard to achieve, interactive applications can well make use of motion planners as animation tools for off-line motion editing and design.

Finally, outside of the animation domains, motion planning finds obvious applications in Ergonomics and Robotics.



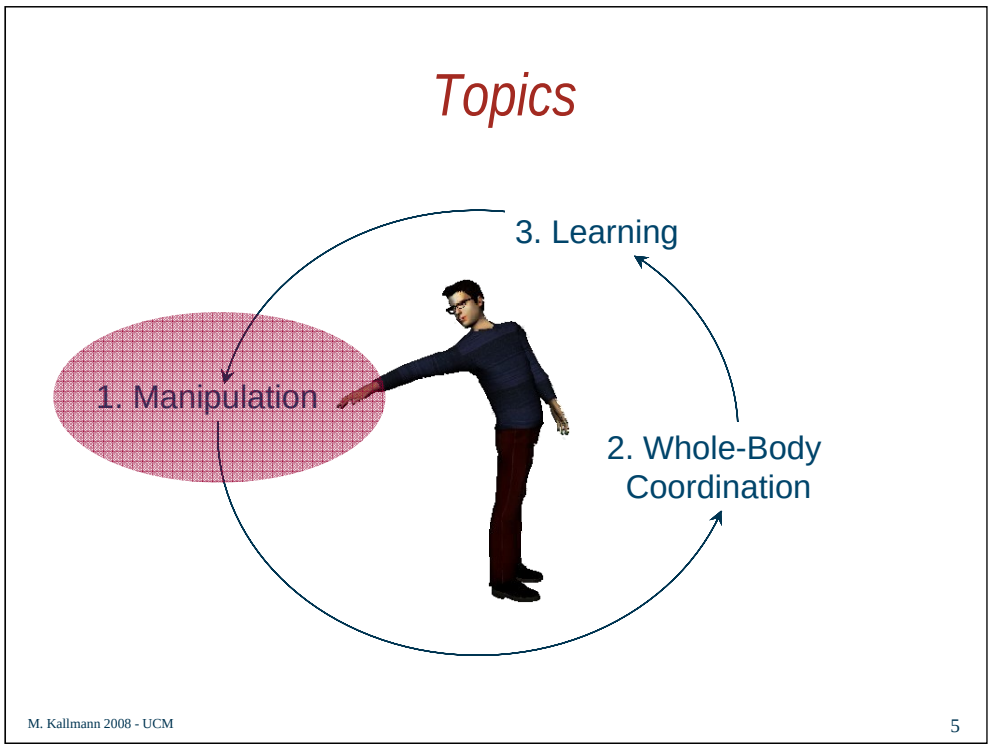
This presentation is divided in three main parts:

(1) Manipulation: the main techniques and approaches for planning object manipulations around obstacles are discussed. This is the most extensive part and it covers all the basics and as well extensions for planning manipulation motions for humanlike characters.

(2) Whole-Body coordination: here two approaches for addressing sequencing and concurrent coordination of primitive motion skills (such as locomotion and reaching) are presented.

(3) Learning is key for achieving motion planners with human-like performances and yet only few learning mechanisms for humanoid manipulation have been proposed. Two approaches specifically applied to humanoid structures are presented.

Note: a set of few selected representative papers is given for each part. These representative papers were selected in consideration to importance and relation to the presented topics. However the reader should be warned that several other important publications are available on each of the discussed topics and the proposed papers are in no way the only relevant ones.



Let’s start with the basic motion planning techniques for manipulation.

Character Representation

- Joint angles control a skeleton with attached geometries
- Geometries are used for collision detection

M. Kallmann 2008 - UCM 6

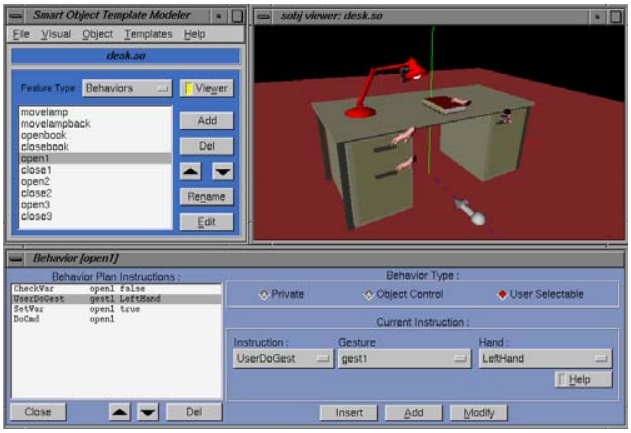
First some review of used representations: a skeleton is a hierarchical kinematic chain used to represent characters. Given a set of joint angles, a pose of the skeleton is defined.

Rigid object parts or deformable skin meshes are then connected to the skeleton for producing realistic visual output, and as well for other types of computation, as for example collision detection.

Collision detection is easier to compute by approximating the character with rigid body parts, and it is very common to have specific “collision geometries” attached to the skeleton (with fewer triangles and with “security margins”) for the purpose of collision detection while the visualization of the character can be actually done with deformable skin meshes.

The right-most image shows that the posture is actually invalid because the two hands are colliding. Note that several body parts may be colliding and yet the character may be in a perfectly valid pose. In particular, the used representation based on rigid body parts will usually mean that all adjacent parts will intersect. A practical way to handle this is to, at initialization, place the character in a known valid position (like a “T-pose”) and then detect and disable all pairs of colliding body parts. These pairs are then kept deactivated for all future collision queries in arbitrary poses.

Representation of Constraints

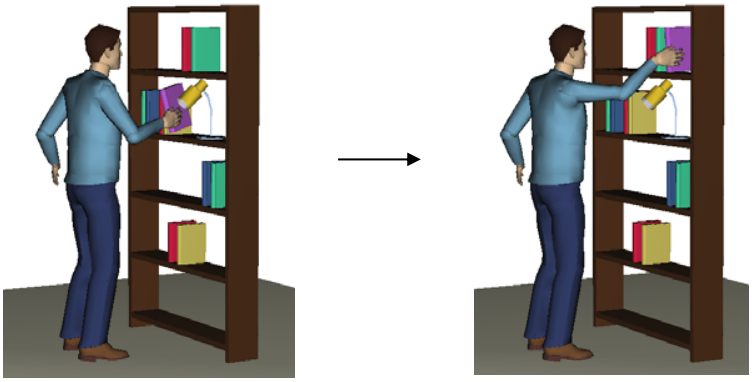


In the planning problems described here it is assumed that target hands and finger placements are given as input to the algorithms and therefore we do not enter into the topic of computing stable and believable grasping hand postures automatically.

As an example, the image illustrates the definition of several of these parameters using the “smart object” interface [Kallmann 2004]. By providing objects with all their manipulation parameters, motion planners can then access the description of hand targets and several other parameters in order to produce the needed manipulation motions.

The Basic Problem

- How to compute a collision-free motion from the initial arm posture to the goal posture ?



M. Kallmann 2008 - UCM 8

The basic manipulation planning problem can be reduced to the problem of moving from an initial configuration to a given goal configuration. A configuration here can be seen as a pose of the character. When only arm motions are required a configuration will refer only to an arm pose.

Note that object reaching and object relocation can be equally treated. In the object relocation case the object being manipulated can be simply attached to the kinematic structure of the character. In that way a new motion computed for the character will therefore be relocating the object from the current location to a new location.

Approaches

- Methods based on motion interpolation
  - High quality but limited solution space
- Inverse Kinematics
  - Most popular, integrates collision avoidance
- Global motion planning
  - Able to address the full solution space
- Hybrid methods
  - Combine different approaches

There are mainly three different approaches for motion planning:

(1) Motion interpolation is based on reusing pieces of motion which are previously modeled or acquired, for example with motion capture devices.

(2) Methods based on Inverse Kinematics (IK) mainly control the trajectories of the end-effectors (for example the hands of a character), and can integrate some reactive collision avoidance behavior in order to cope with obstacles.

(3) Methods based on global motion planning have the potential to find complex solutions by searching the full configuration space of the problem. While in low dimensional spaces cell-decomposition and graph search can be used for globally searching for solutions (as in grid-based 2D path planning), sampling-based methods have to be employed in high dimensional spaces, as is the case for planning arm motions with 7 degrees of freedom (DOFs).

Hybrid methods trying to combine the best aspect of each approach are possible and may represent a promising area for future research.

Motion Interpolation

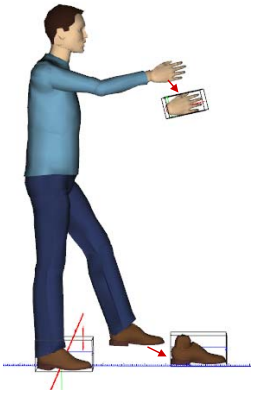
- Capable of High Quality Results
  - However addressing a small portion of the solution space
- Representative Papers
  - Radial Basis Function Interpolation
    - [Rose et al 2001]
  - Parameterization of Motion Graphs
    - [Kovar and Gleicher 2004]
    - [Safonova and Hodgins 2007]
    - [Heck and Gleicher 2007]

Motion interpolation methods recombine example motions in order to produce new motions achieving desired characteristics. This is usually achieved with weighted interpolation of the example motions. Realistic results can be achieved in particular when the example motions are captured from real human performances. The main drawbacks of this approach are: the difficulty to handle planning around obstacles, and the difficulty to generate examples able to cover a sufficiently large solution space.

The work of [Rose et al 2001] proposes an interpolation method based on Radial Basis Functions and additional techniques for ensuring that the hand of the character will end very close to a specified point in space. Works based on motion graphs [Kovar and Gleicher 2004] [Safonova and Hodgins 2007] [Heck and Gleicher 2007] are designed to automatically process motion capture data in order to enable motion interpolation and parameterization. Motion planners can then be plugged to operate in the created parametric spaces. Although this line of work has received a lot of attention for planning locomotion, not much attention has been given to manipulation of objects around obstacles.

Inverse Kinematics

- IK solves joint rotations such that the end-effector reaches a given target



M. Kallmann 2008 - UCM 11

Inverse Kinematics is a method used to solve the problem of computing joint rotations in order to reach given targets with the end-effectors. The figure shows the typical application of moving hands and feet to desired placements.

Inverse Kinematics

- IK is the main available technique for precisely controlling kinematic chains
- Representative Papers
  - Numerical, Jacobian-based methods
    - [Maciejewski and Klein 1985]
    - [Baerlocher 2001]
  - Analytical solutions for 7-DOF arms and legs
    - [Tolani et al 2000]
    - [Kallmann 2008]

$$J(\mathbf{f},\mathbf{x}) = \frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix}$$

M. Kallmann 2008 - UCM 12

There are two main classes of Inverse Kinematics (IK) algorithms: Jacobian-based methods and analytical methods.

The Jacobian matrix J relates small changes in workspace to small changes in joint sapce. Therefore Jacobian-based methods will, at every frame, typically compute the Jacobian matrix and then pseudoinvert it in order to compute small joint increments to be added to the current pose. These increments will make the end-effector get closer to the goal location. This is an iterative method and so a number of iterations will be required in order to reach the goal. Note that the Jacobian matrix can be computed geometrically, i.e., deriving its elements from local joint influences geometrically (typically with a series of cross products for joints parameterized with Euler Angles). Although it is an iterative method, Jacobian-based IK can handle arbitrary open kinematic chains, and extensions are available for integrating collision avoidance [Maciejewski and Klein 1985] and for handling the execution of several tasks in an unified framework [Baerlocher 2001].

Analytical methods in contrast can produce poses reaching the goal with a fixed number of mathematical operations. Analytical methods are extremely fast, however are usually limited to control only 7-DOF arms and legs and the incorporation of secondary tasks becomes less straightforward. Analytical solutions are discussed in [Tolani et al 2000] and extensions for collision avoidance an joint limits avoidance are available [Kallmann 2008].

Examples: Jacobian-Based IK

The images illustrate two examples of postures computed with the multi-task Jacobian-based IK formulation [Baerlocher 2001]. In the left image, in order to reach a low hand target, the IK framework is able to keep the additional constraint of maintaining the feet fixed at the floor while the root of the skeleton is lowered, resulting in knee flexion. In the right image, knee flexion is not necessary but a twisting of the torso is required.

Videos of these two examples are available on-line:

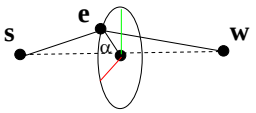
http://graphics.ucmerced.edu/videos/2001_drawer_lower.wmv
http://graphics.ucmerced.edu/videos/2001_drawer_upper.wmv

Examples: Jacobian-Based IK

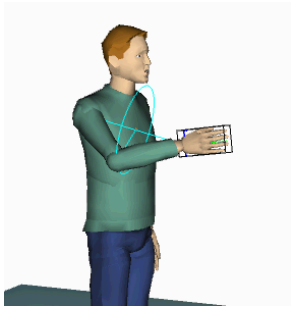
Opening doors and pushing objects can also be addressed with IK by controlling arm movements in different phases: first to reach a given hand placement, then to make the hand follow the motion of the object being manipulated (as to open a door).

Examples: Analytical IK

- 7-DOF linkages have analytical solution



α = swivel or orbit angle



M. Kallmann 2008 - UCM 15

Humanlike arms are usually parameterized with 7 DOFs: 3 rotations at the shoulder joint, flexion and twisting at the elbow joint, and two rotations at the wrist joint. Some representations prefer to define only flexion at the elbow joint and a 3 DOF rotation at the wrist, and others prefer to define an additional joint between the elbow and wrist specifically to model the lower arm twist rotation.

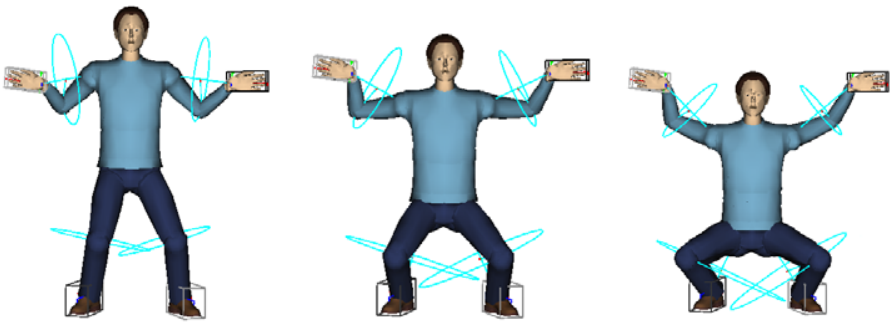
For any of these 7 DOF parameterizations it is possible to derive IK equations analytically. Note that a target hand position and orientation represents a 6 DOF constraint and the number of joint angles is 7. The extra 1 DOF (7-6=1) can be parameterized by giving as input to the IK the desired elbow position along its orbit circle (or swivel circle). The position is defined with an angle of rotation around the shoulder-wrist axis, called the orbit or swivel angle. See the left image for an illustration.

The right image illustrates a character and the orbit circle of its right elbow. The orbit circle represents all possible positions for the elbow (not considering joint limits and possible collisions). A video demonstrating the analytical IK is use is available on-line:

http://graphics.ucmerced.edu/videos/2007_stikarm.avi

Examples: Analytical IK

- Analytical IK can be applied to arms and legs



M. Kallmann 2008 - UCM 16

As both arms and legs can be represented with 7-DOF linkages, the analytical IK formulation can be similarly applied to them. In [Kallmann 2008] a formulation is presented including extensions for the automatic determination of the orbit angle in order to avoid collisions and as well joint limits.

The sequence of images in this slide illustrates how important is the realistic modeling of joint limits. In this example four IK solvers enforce the hands and feet of the character to remain fixed at their locations while the body of the character is lowered. The orbit angles of the legs are automatically determined in order to maintain the joint rotations within defined joint limits. As a result, when the character's root joint is significantly lowered, the IK has to rotate the knee joints outwards in order to maintain the joint rotations of the feet within limits. The same behavior can be observed in humans.

A video showing this animation is available on-line:

http://graphics.ucmerced.edu/videos/2008_stikrootmotion.avi

Motion Planning

- Global motion planning
  - Search trajectories in configuration space
  - Sampling-based methods used for high dimensions
- Most popular sampling-based planners
  - Probabilistic Roadmaps (PRMs)
    - [Kavraki et al 1996]
  - Rapidly-Exploring Random Trees (RRTs)
    - [Kuffner and LaValle 2000]

Inverse Kinematics is a powerful and needed tool, however it is a local method, i. e., it does not perform alone a global search for solutions around obstacles.

Motion planners have the potential to find complex solutions by searching the full configuration space of the problem. While in low dimensional spaces cell-decomposition and graph search can be used to globally search for solutions (as in grid-based 2D path planning), sampling-based methods have to be employed in high dimensional spaces.

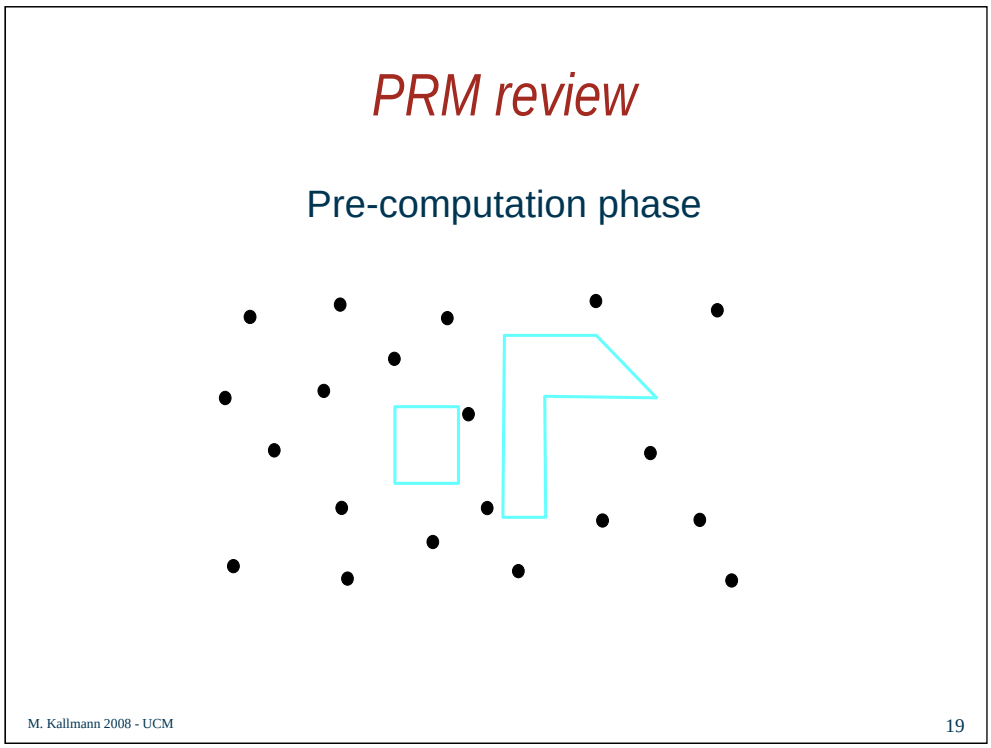
For humanlike manipulation planning the search space has at least 7 dimensions, which is the number of DOFs of the arm. Therefore sampling-based motion planners are the method of choice for most of the problems considered here.

Motion Planning for Virtual Humans

- Representative Papers
  - First application
    - [Koga et al 1994]
  - First RRT Application
    - [Kuffner and Latombe 2000]
  - Several extensions exist
    - [Kallmann et al 2003]
    - [Yamane et al 2004]
    - [Kallmann 2005]

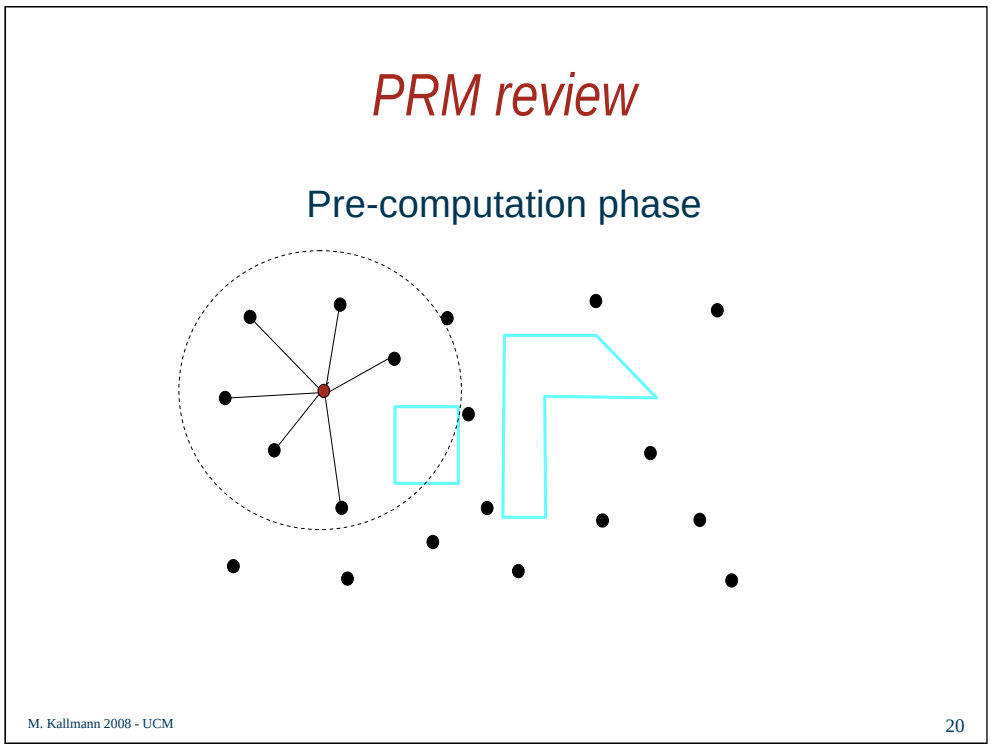
A list of representative papers specifically applying motion planners to animated characters manipulating objects is offered here.

The work of [Koga et al 1994] is probably the first application of a motion planner to an animated character and [Kuffner and Latombe 2000] first apply RRTs for animated characters. In [Kallmann et al 2003] several techniques are presented for achieving full-body manipulations. [Yamane et al 2004] propose to plan the motion of the object being manipulated while the hands of the character remain attached to the object with IK. The authors also propose an IK method which produces humanlike postures by relying on a posture database built with motion capture. Finally in [Kallmann 2005] approaches integrating the analytical IK and RRTs for interactive applications are discussed.

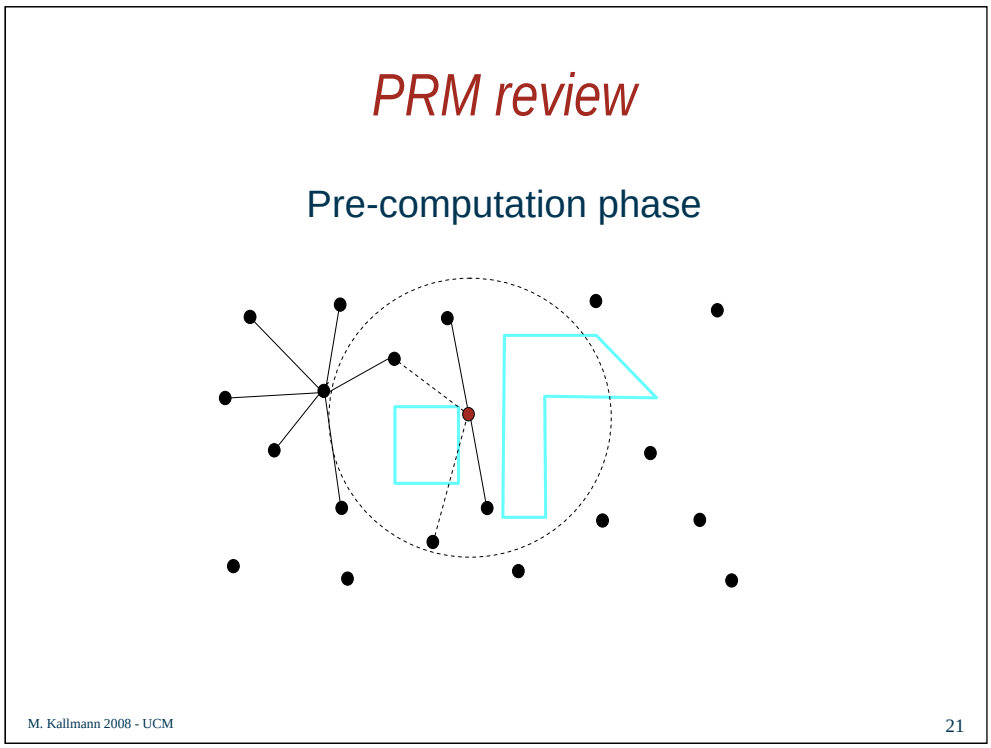


We now review Probabilistic Roadmaps (PRMs).

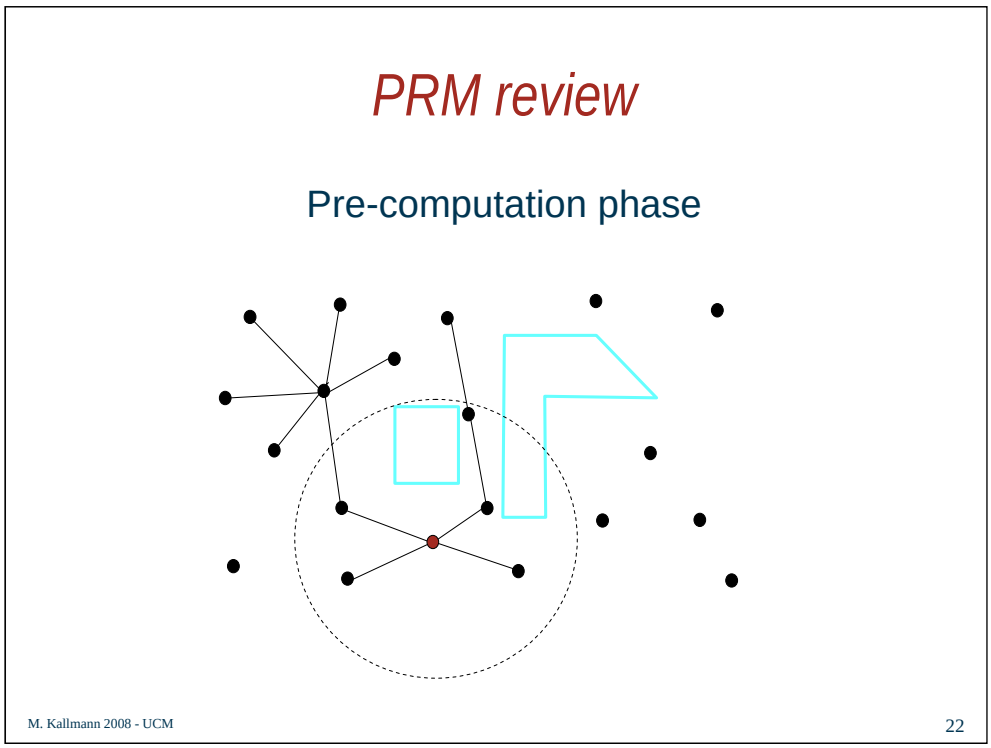
Probabilistic roadmaps are built by sampling the configuration space as a pre-computation phase. Consider the two polygons in the image to be obstacles in the 2D Euclidian plane. Random points are then sampled in the environment and tested for validity (not touching or inside obstacles). The sampling step finishes when a given number of valid samples is reached.



Then, for each sample, its k nearest neighbors are determined and edges are added for each valid connection with a neighbor. In this example a straight line connection is valid when it does not cross any obstacles. In the example all 6 neighbors are connected.



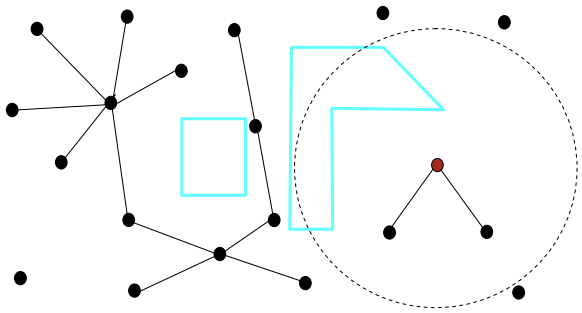
For this new point, only 2 connections out of 4 are valid. A graph starts to appear.



A new set of edges is added, and the graph has now a single connected component.

PRM review

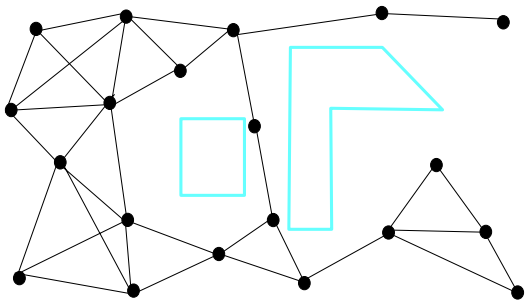
Pre-computation phase



Here a new set of edges is created, now adding a disconnected component.

PRM review

The obtained roadmap

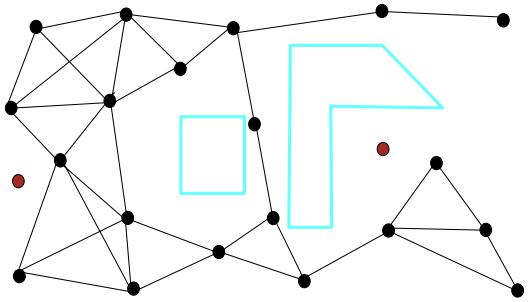


M. Kallmann 2008 - UCM 24

After repeating the process for every point, the final graph is obtained. The graph may have multiple components. The graph represents the free space and is called the roadmap.

PRM review

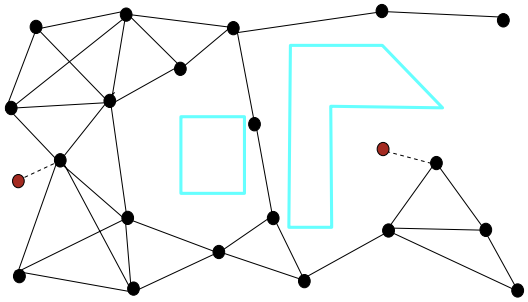
On-line query phase



At run-time motion planning problems for connecting two given points (the red points in the image) can then solved.

PRM review

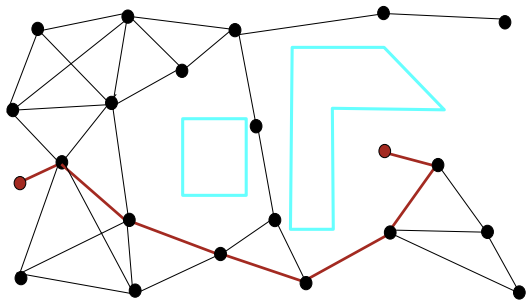
On-line query phase



First the closest nodes in the graph to each of the points are determined.

PRM review

On-line query phase

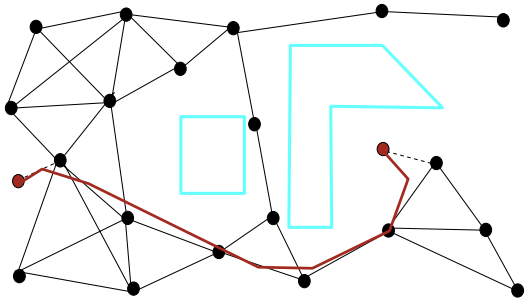


Now that the initial and goal points are connected to the graph, a graph search algorithms (like A*) can be used to retrieve the shortest path in the graph. If the points cannot be connected to the graph, or if the graph search fails due to the initial and final points being in disconnected components, then the planner returns false. A possible alternative is to add more nodes to the graph and try again.

In the example, a path is found.

PRM review

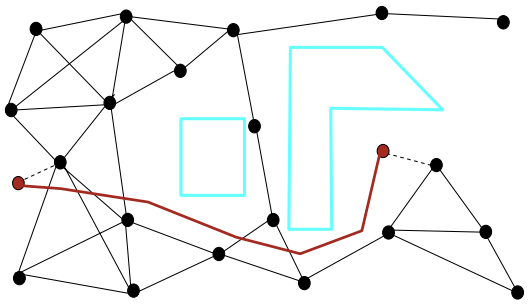
On-line query phase



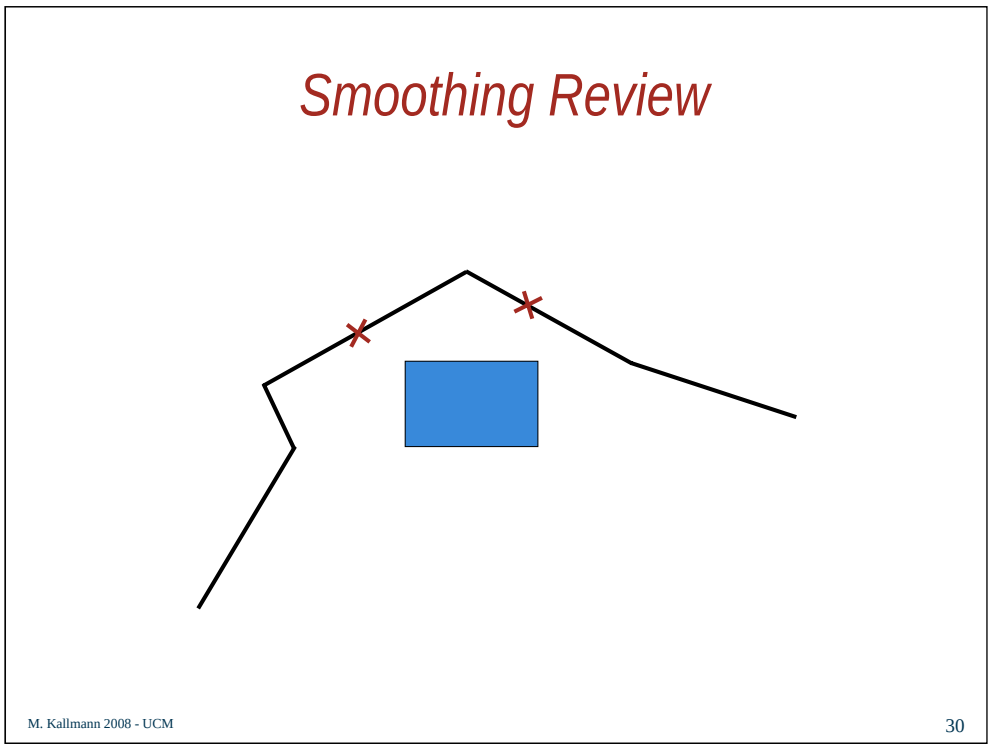
The shortest path represents a solution passing by the original random points, and therefore a smoothing process is often needed to improve the result. Usually, smoothing processes require a few iterations.

PRM review

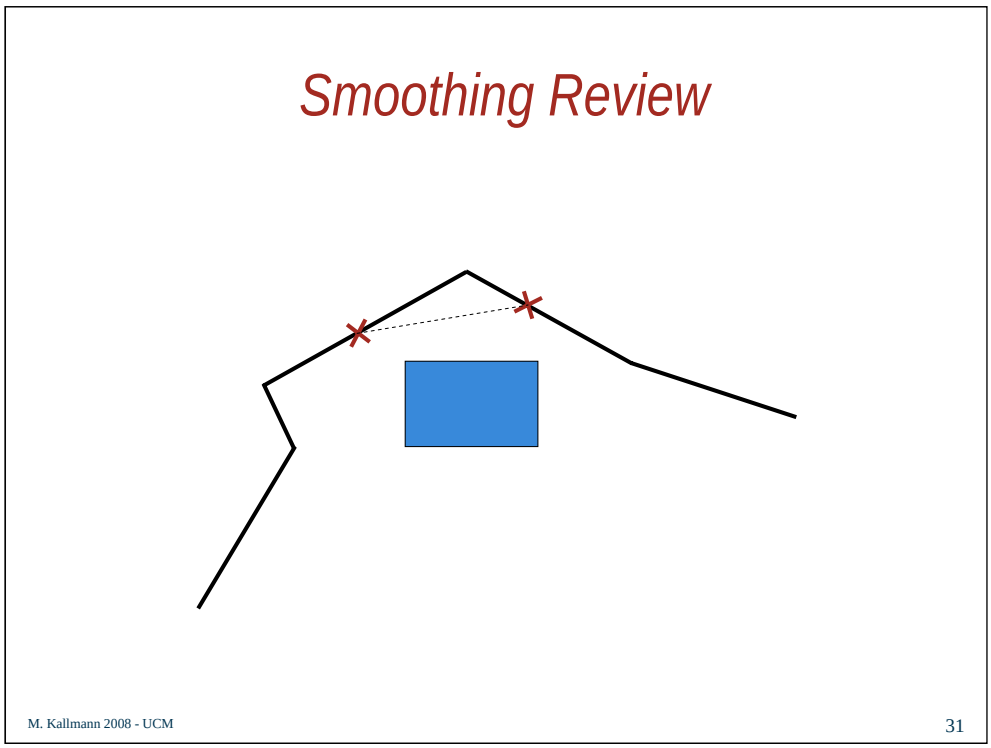
On-line query phase



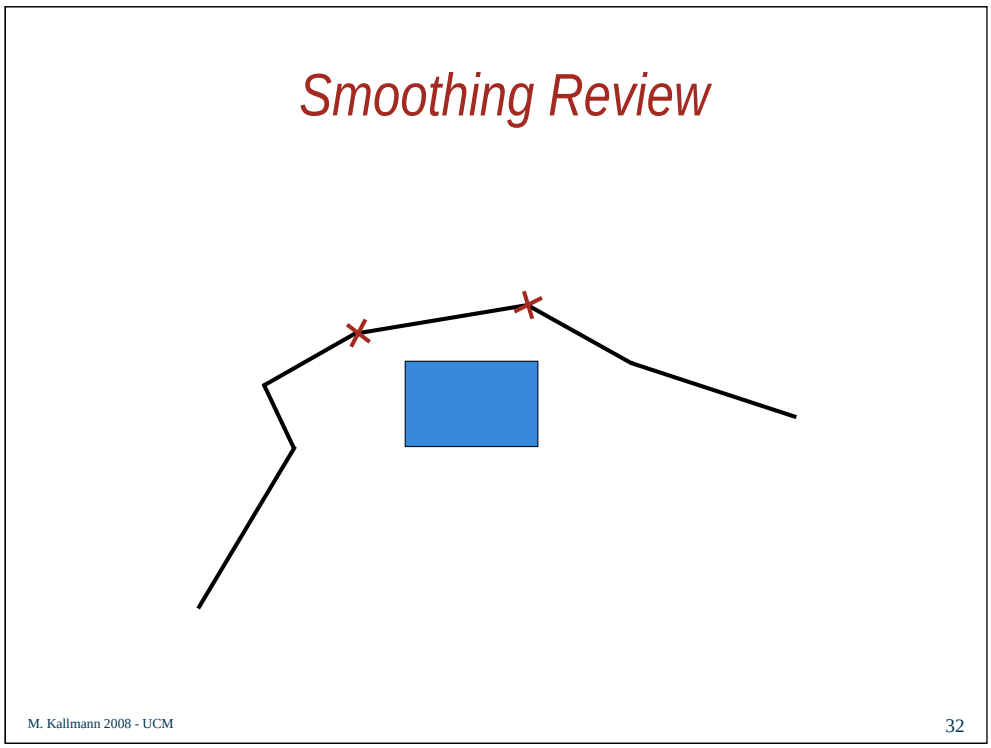
As more iterations are run the path becomes smoother and shorter.



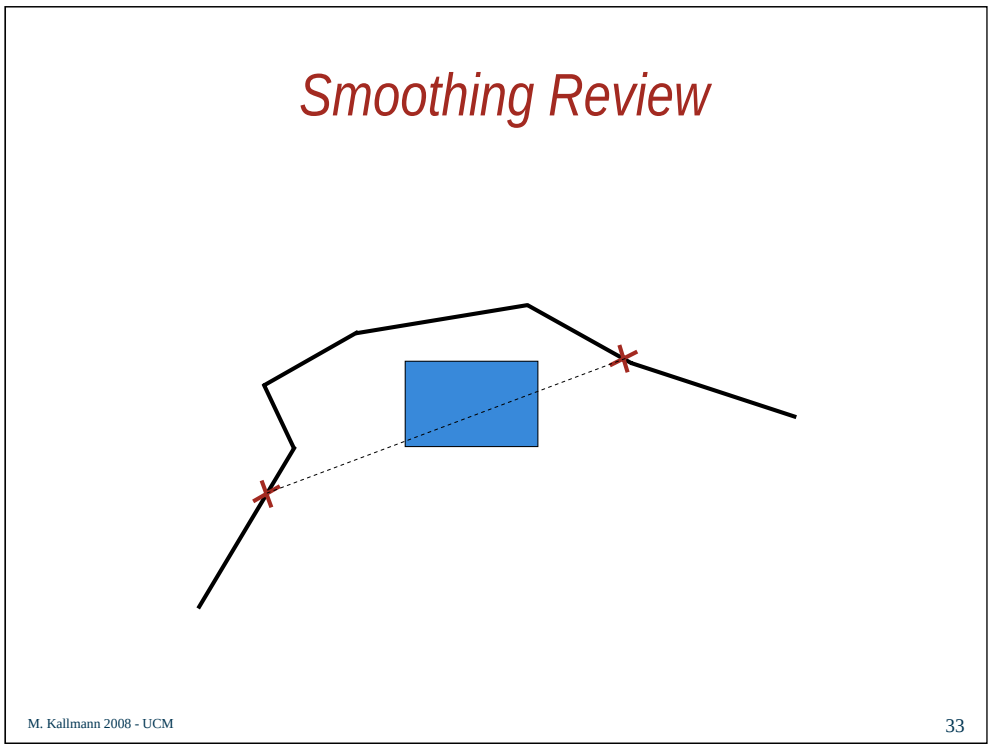
The most popular smoothing procedure is quite simple. First select two random points along the current path.



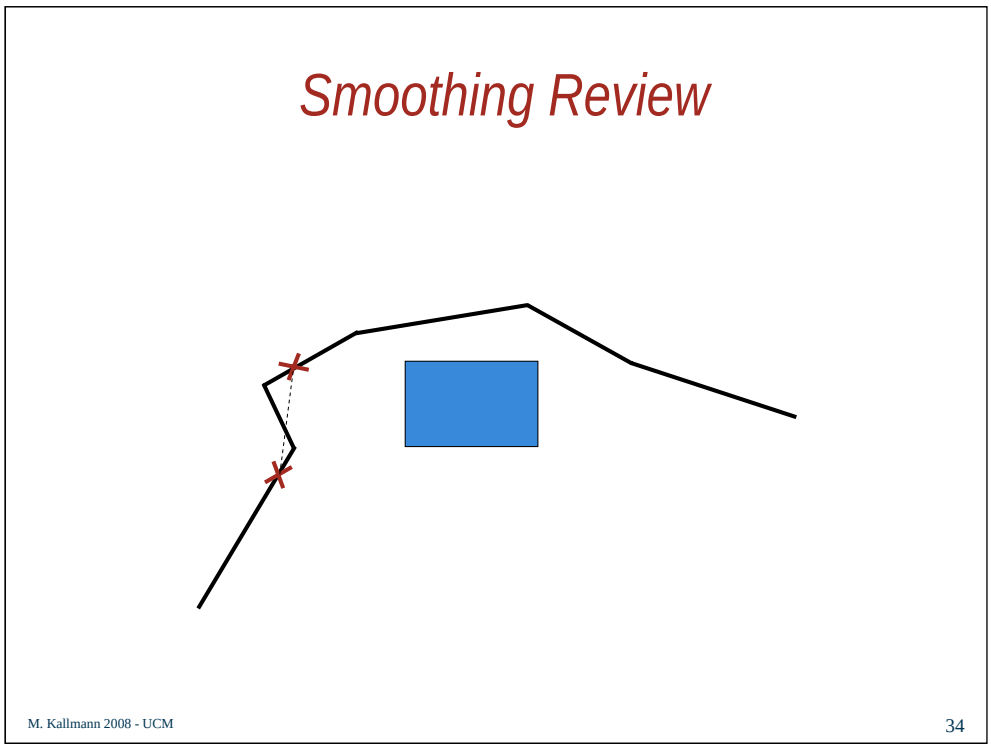
Now a validity check is performed to verify if the straight line connecting the two points is valid. If it is, the line segment will replace the original portion of the path connecting the two points.



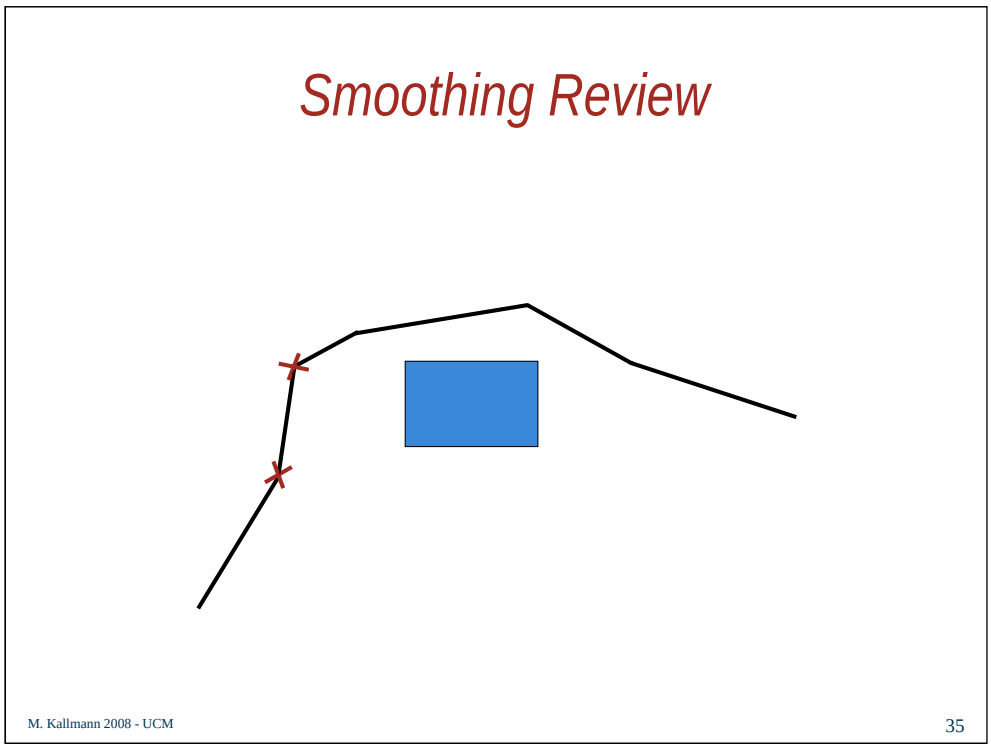
Here is the result after the first iteration.



Here is a replacement which is not valid and therefore rejected.

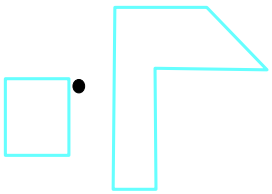


This new replacement is valid.



Iterations can then be repeated until replacements start to not occur very often, or until a given elapsed smoothing time has passed.

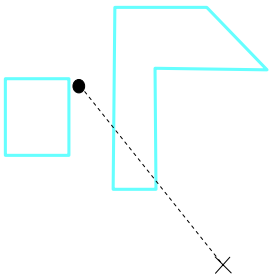
RRT Review



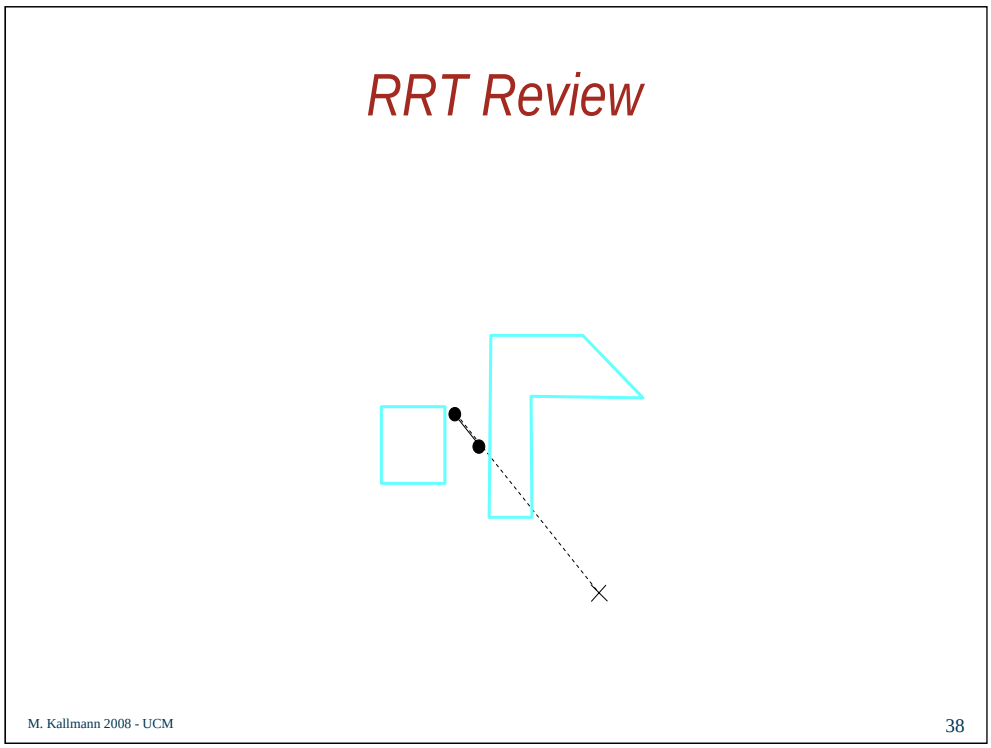
M. Kallmann 2008 - UCM 36

RRTs are trees and are designed to be built on-line, until a given query is solved. Here we explain the construction proces of the RRT. Given an initial position in a valid location, a tree will be built rooted at this position.

RRT Review

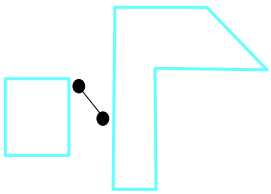


For every random sample in the environment, the closest node in the tree is identified. In the illustrated case the current tree has only one node.

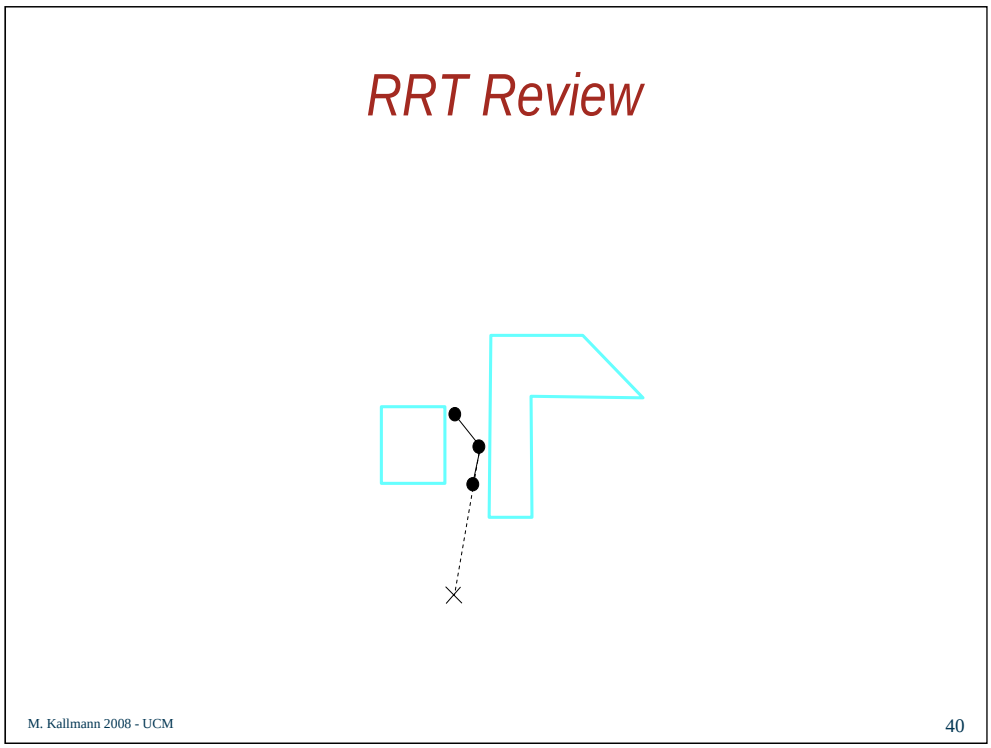


The closest node in the tree is then evaluated for expansion towards the random sample by a given step. If the extension step (towards the sample) is valid (no collisions in the example), then the extension node and edge are added to the tree.

RRT Review

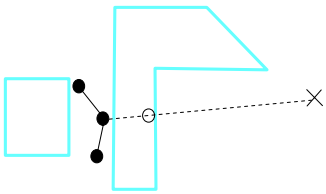


The current tree has now 2 nodes and one edge.



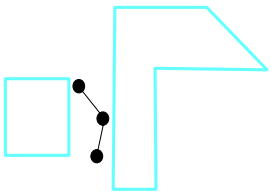
The process repeats. Here a new extension towards a new random sample is tested.

RRT Review



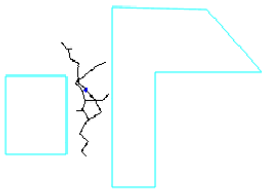
Here the extension towards a new random sample is not valid.

RRT Review



The previous extension was rejected and therefore the tree did not grow during the last iteration.

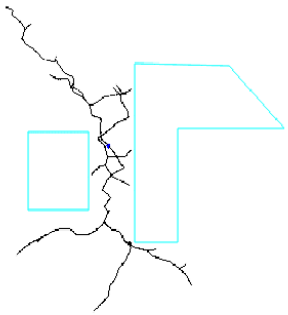
RRT Review



M. Kallmann 2008 - UCM 43

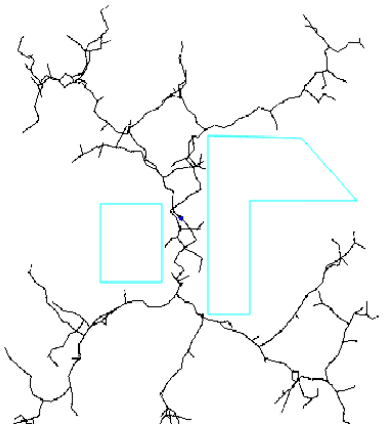
By repeating this iterative process the tree will be expanded only in valid locations.

RRT Review



The expansion is guided towards the empty spaces, where more valid expansions can occur. This aspect is called a Voronoi bias.

RRT Review



The tree can grow as much as needed, until it is possible to connect one of its branches to a given goal location.

Bidirectional RRT

Trees rooted at c_i and c_g grow simultaneously

M. Kallmann 2008 - UCM 46

The RRT method is most efficient in its bidirectional version: one tree grows rooted at the initial location and another tree grows rooted at the goal location. The two trees grow until they can connect to each other. Note that when a connection between the two trees is found there is no need to perform graph search as there will be only one path joining the two tree roots, which is the one composed by the connecting branches.

The example in the image shows two trees being expanded in a 3D configuration space representing the position (2 DOFs) and the orientation (1 DOF) of the polygon. Each node in the tree is therefore associated with the 3 parameters (2D position and orientation) specifying the placement of the polygon and the solution path also encodes the needed changes in orientation in order to move the polygon from the initial position to the goal position. A video showing the resulted motion is available at:

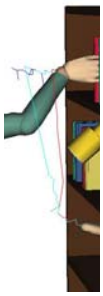
http://graphics.ucmerced.edu/videos/2004_rrt_polygon.avi

Example

RRTs used to animate characters

tree expansion

connection found

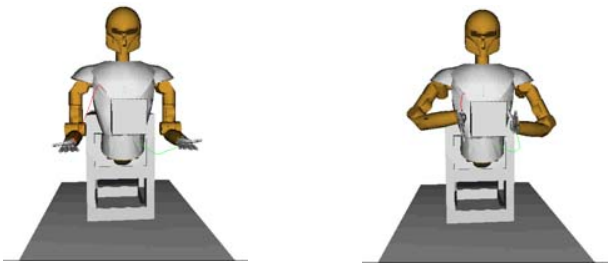


solution optimized

By sampling and connecting 7-DOF arm postures it is possible to apply the bidirectional RRT method to plan a collision-free trajectory from the current arm location to a target arm pose. The image illustrates the trajectories of the wrist joints during the planning process.

Example

RRTs used for both arms



M. Kallmann 2008 - UCM 48

This example shows the application of the planner in the composite configuration space of two arms. Here each tree node contains a 14-DOF configuration specifying the placement of both arms. The planning process is the same and will result in a coordinated motion of the two arms from the initial dual arm placement to a target placement.

The example illustrates moving the hands of NASA's Robonaut to desired pre-grasp positions. Videos showing the resulted motions are available on-line:

http://graphics.ucmerced.edu/videos/2003_rrt_robonaut1.avi
http://graphics.ucmerced.edu/videos/2003_rrt_robonaut2.avi

Example

RRTs for Object Relocation

M. Kallmann 2008 - UCM 49

In this example Inverse Kinematics is used to compute arm postures reaching hand placements attached to books. The bidirectional RRT is then used to find collision-free motions between the arm postures, and as result books can be relocate in the shelf. A video of the resulted animation (which was computed on-line) is available at:

http://graphics.ucmerced.edu/videos/2005_aiide_relocation.avi

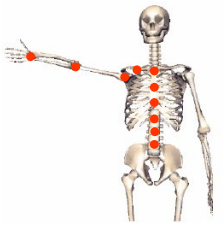
Motion Planners for Characters

- What is needed?
  - Configuration sampling routine
  - Distance function between postures
  - Posture interpolation method
  - Validity check routine
- Special solutions needed for producing whole-body motions

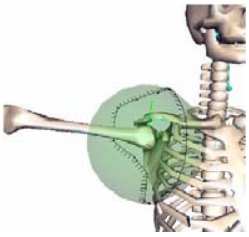
Applying sampling-based planners to humanlike characters requires a number of routines to be developed. Furthermore, in order to improve the results, these routines have to be designed to also control the body of the character, otherwise the result is a rigid-looking character only moving its arms. Small variations on the spine joint for instance can already improve the result. Such variations are simple to include by sampling over a small interval centered around the joint angles of the rest posture. In all cases, a good definition of joint range limits is needed for delimiting the regions for sampling joint values.

Sampling Whole Body Reaching Postures

- 22 DOFs control layer
  - 7 in each arm, 2 in each clavicle
  - 3 in the torso (distributed in spine)
  - 1 knee flexion (translational DOF)



- Specific per-articulation parameterization and range limits
  - Ex: swing-and-twist decomposition for the shoulder:
    $R = R^{twist}\ R^{swing}$
    $R^{twist}=R_z(\theta)$, $R^{swing} = [S_x\ S_y\ 0]$



M. Kallmann 2008 - UCM 51

In order to achieve more complex postures with automatic knee flexion a control layer is designed where one single translational DOF represents the amount of vertical translation of the root joint. This single DOF is then a parameter being sampled during planning. Note that in this case the planner operates on the parametric space of the control layer and not directly over joint angles. The control layer is then responsible for computing the final posture. For instance IK is used to maintain the feet fixed on the floor whenever the vertical translational DOF will lower the root joint in order to achieve knee flexion.

The control layer can also simplify the distribution of rotations along spine joints. In the example, a single 3-DOF rotation in the control layer controls the distribution of rotations over all the spine joints existing in the character model.

Finally, plausible articulation joint limits are very important both to ensure the generation of feasible postures and to reduce the search space of the planner. The lower image illustrates the use of spherical polygons for precisely delineating the range of motion of the shoulder joint, probably the most important articulation of the arm. Details can be found in [Kallmann et al 2003].

Sampling Whole Body Reaching Postures

- 60% Regular
  - little spine and leg flexion
  - no clavicle motion
  - random arm poses
- 40% Distant
  - large spine and leg flexion
  - little elbow flexion
  - shoulder-clavicle coupling
  - arm-legs coupling
  - arm-torso coupling
- Balance, limits, collisions

M. Kallmann 2008 - UCM 52

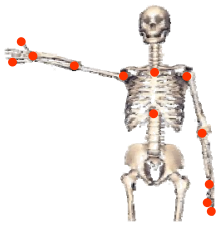
After a suitable control layer is defined the sampling routine will still be responsible for generating meaningful humanlike postures for manipulations.

For instance, it is not reasonable to compute a sample posture with the knees bent and at the same time with one arm outstretched upwards, as if the character is trying to reach for a high location and at the same time flexing the knees. Although such uncommon postures are feasible they should not make part of normal humanlike manipulation activities.

The strategies adopted in [Kallmann et al 2003] are summarized here. 60% of the sampled postures are generated with regular reaching characteristics, and 40% with distant reaching characteristics. While regular postures will have the body nearly straight, distant-reaching postures will have large spine and leg flexion, and will also observe couplings. Arm-leg and arm-torso couplings will ensure that the arms will stretch in the same direction of the leg flexion or torso rotation, avoiding unreasonable situations such as the one described in the previous paragraph.

Distance Function

- Maximum of Euclidean Distances between corresponding meaningful articulations, for ex:
  - bottom most thoracic vertebra
  - top most thoracic vertebra
  - both arms articulations
  - thumb and pinky



- Other distance functions can also be used

M. Kallmann 2008 - UCM 53

In order to grow the RRT during the planning process, the node closest to a new sampled posture has to be determined. A distance function is therefore needed.

Although several different distance functions can be defined, the usual maximum norm between joint positions works well, given that enough meaningful joints are chosen.

Validity Check

- Collision Detection
  - Self collisions and obstacle collisions treated together
- Continuous Collision Tests
  - Needed during tree expansion and smoothing
  - Easier to approximate with discrete tests
- Other validity tests
  - Articulation Limits
  - Static balance
  - Dynamic balance

discrete tests along
a motion segment

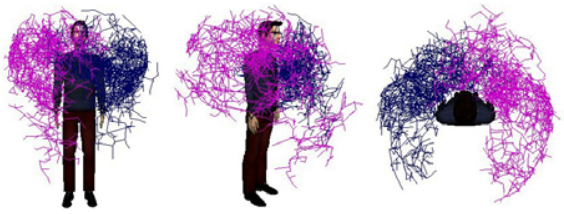
Each edge added to the RRT during the planning process represents a small motion segment interpolating the end-nodes. Before adding the edge to the tree, each edge has to be tested for validity. Also during the smoothing process, at each iteration, each motion segment candidate for replacement has to be tested for validity.

The validity test has to ensure that a motion segment is collision-free, respects joint limits, and also keeps the character in balance. Among these tests, the collision test is usually the most time-consuming test to compute. The main reason for that is because the collision test has to be performed for the entire motion segment, what is called a continuous collision test.

The continuous collision test can be simplified by performing a number of discrete tests along the motion segment. This is the most popular approach and several discrete collision detection packages are available for that [Gottschalk et al 1996]. If needed, continuous collision tests are also available [Redon et al 2004] [Schwarzer et al 2002].

Example

PRM-like roadmap obtained



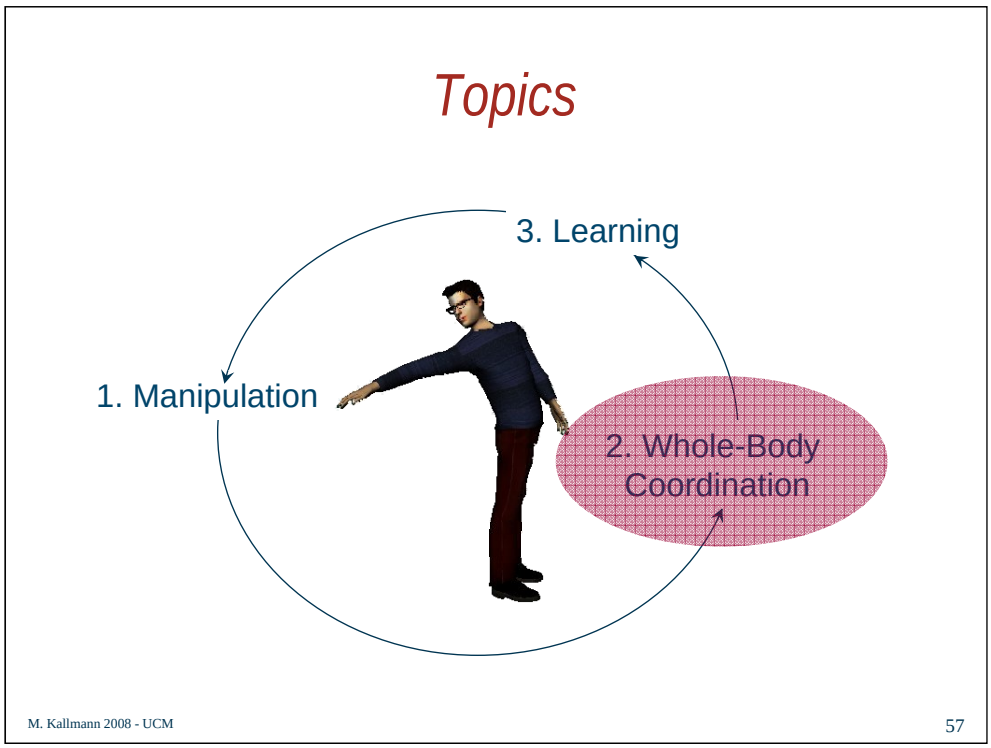
In this example a PRM-like roadmap is built using the 22-DOF control layer previously described. The graph shows, for each node, its encoded position of the right and left wrist joints. Therefore the chosen visual representation for this example shows an approximation of the reachable space of the character.

Note: in this example the graph part relative to the right wrist reaches lower positions because the sampling routine was designed to generate more variations for the right arm, in order to simulated a right-handed character.

Example

These sequences illustrate two examples. In the top sequence postures with knee flexion are used by the planner in order to reach for the given goal location inside the fridge. The second example shows an object relocation where torso rotation can be observed but knee flexion is not needed.

The complete video illustrating these and several other examples is available at:
http://graphics.ucmerced.edu/videos/2003_eg_reaching720x576.mpg



So far our characters had their feet fixed to the floor. We now move to extensions for planning body coordination in more complex tasks.

Whole-Body Coordination

- When choosing between different types of motions is part of the problem
- Coordination as a Planning Problem
  - For sequencing motions in different modes
  - For concurrent synchronization of different skills
- Notion of independent primitive skills

Body coordination is needed for example for deciding where the character has to stand in order to then reach for a goal hand location. Complex coordination is also needed for tasks such as grasping while walking, or even planning biped locomotion among obstacles.

Planning body coordination follows the notion of coordinating primitive motion skills in order to achieve a complex behavior.

Whole-Body Coordination

- Representative Papers
  - Coordinating steps for locomotion
    - [Kuffner et al 2001]
    - [Kallmann et al 2004]
  - Planning motions for climbing
    - [Kalisiak and van de Panne 2001]
    - [Bretl 2006]
  - Coordinating walking and pushing
    - [Hauser et al 2007]
  - Coordinating walking and grasping
    - [Shapiro et al 2007]

The coordination problem has been addressed in a variety of different scenarios. In [Kuffner et al 2001] a discrete search is performed to determine the best sequence of pre-computed leg motions in order to plan locomotion sequences among obstacles for humanoids. In [Kallmann et al 2004] this approach is re-formulated as a multi-mode RRT problem without the need of pre-computed motions.

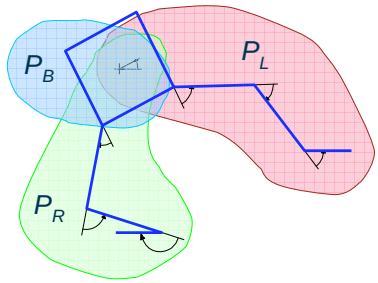
Planning motions for climbing can also be seen as planning coordinated motions manipulating the environment. A planner for a 2D character has been proposed by [Kalisiak and van de Panne 2001] and algorithms for real robots that can climb are available [Bretl 2006].

When it comes to body coordination for manipulation tasks, good examples are the multi-mode planner developed for Honda's ASIMO [Hauser et al 2007], and the approach of using a manipulation planner which synchronizes with motion capture sequences [Shapiro et al 2007].

The approaches in [Kallmann et al 2004] and [Shapiro et al 2007] propose extensions to the RRT planner for handling two main aspects of coordination: (1) sequencing and (2) concurrent synchronization. These two approaches will be now reviewed.

Coordination: Sequencing

- Consider the 9 DOF structure below
- Movement control only allowed through 3 simple primitive controllers

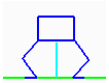


M. Kallmann 2008 - UCM 60

We now review the sequencing approach proposed in [Kallmann et al 2004]. The example shows a biped structure which can be controlled only by 3 simple controllers, each moving a subpart of the structure. The problem then becomes how to plan the operation of each controller and as well the sequence of controllers to use in order to have the structure to perform a valid walk motion.

Although this method has not been integrated in an object manipulation application, it can clearly be used for planning stepping motions in a constrained environment in order to achieve suitable body positioning for object manipulation.

Coordination: Sequencing

Movement Primitive	Instantiation Condition	Primitive Motion	Parametric Space Dim.
 P_L	support in left foot	moves right leg articulations and body rotation	4
 P_B	support in both feet	moves body, feet fixed with IK	3
 P_R	support in right foot	moves left leg articulations and body rotation	4

M. Kallmann 2008 - UCM 61

The first primitive controller assumes the structure has support on the left foot only, and therefore is free to move the right leg. This controller moves the joint angles of the right leg and as well the body orientation. For each movement, Inverse Kinematics is used to maintain the foot of support in contact with the floor.

The second primitive controller assumes the structure has support on both feet, therefore this controller only moves the body position and orientation and relies on Inverse Kinematics to maintain both feet in contact with the floor.

The third primitive controller is equivalent to the first one, but for the case the structure has support on the right foot only.

In all cases, each primitive controller is only allowed to move the structure if the structure remains collision-free, respecting joint limits and in balance.

Animations showing the operation of each primitive controller are available on-line:

http://graphics.ucmerced.edu/videos/2004_sab-bothsup.avi
http://graphics.ucmerced.edu/videos/2004_sab_rightsup.avi
http://graphics.ucmerced.edu/videos/2004_sab_leftsup.avi

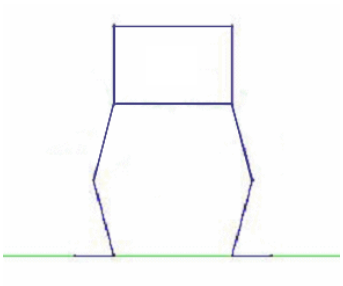
Sequencing: Problem Definition

- Given:
  - Primitives P_i
  - Task completion test function $t(q) : C \rightarrow \{0,1\}$
- Determine a sequence of concatenated valid paths in configuration space, such that:
  - Each path is generated by a single primitive
  - The first point is the current position
  - The last point satisfies completion test

The sequencing problem is then to determine a sequence of primitive controllers able to make the biped structure to walk towards a target location.

Sequencing: Search Tree

- Define a search tree (single component) having the root as the initial configuration c_i

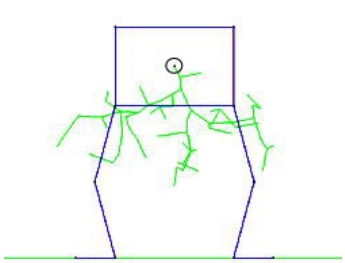


search tree

To solve the problem we first determine the primitive controller which can operate the current configuration of the structure. In this case the structure is in both-feet support mode.

Sequencing: Search Tree

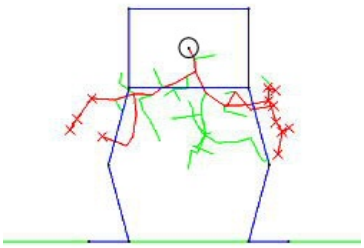
- Expand a roadmap in the parametric space of the motion primitive associated with *c*



A RRT tree is then expanded in the parametric space of the current primitive controller. As the range of motion of the primitive controller is limited, after a certain number of iterations the tree will have difficulties in expanding new edges. At this point the tree expansion stops. The expansion step will determine the density of the nodes. The tree shown in the figure represents the evolution of the body center position.

Sequencing: Search Tree

- Determine paths leading to configurations in a different support mode

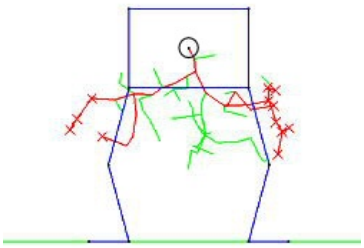


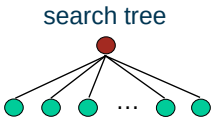
search tree

The obtained tree is then analyzed and each leaf node allowing a valid transition of support mode (and therefore change in primitive controller) is marked. In the figure, these leafs are marked with a cross. Note that the tree is able to provide a valid motion from the initial posture to each leaf posture.

Sequencing: Search Tree

- Each path represents a new child of *c*

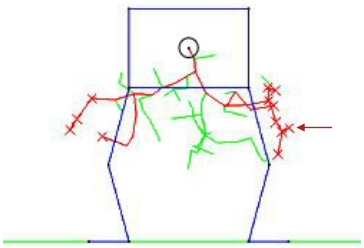


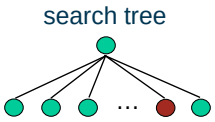


All marked leafs are candidate choices for the final motion and therefore are stored in a discrete search tree, as illustrated in the diagram.

Sequencing: Search Tree

- Select lowest (A*) cost leaf *c*
  cost(*c*) = length(root,*c*) + dist(*c*,goal)

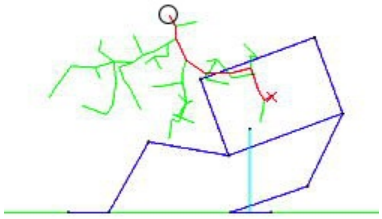


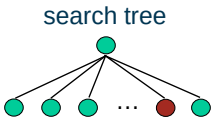


Each of the next possible primitive controller in the search tree represents a valid candidate to be selected. The search will determine the optimal solution based on the given cost which encodes a length measurement of the traversed space. The A* heuristic is then used to select the leaf node of the search tree that should be expanded first. This leaf is marked with an arrow.

Sequencing: Search Tree

- Select lowest cost leaf *c*
  cost(*c*) = length(root,*c*) + dist(*c*,goal)

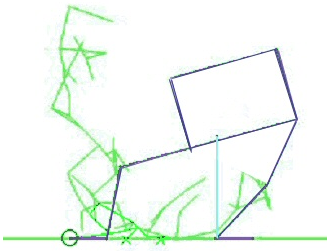


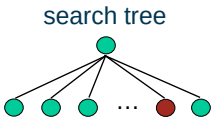


The posture associated with the selected leaf node contains a posture suitable for a change to single support mode.

Sequencing: Search Tree

- Expand a roadmap in the parametric space of the new motion primitive associated with *c*

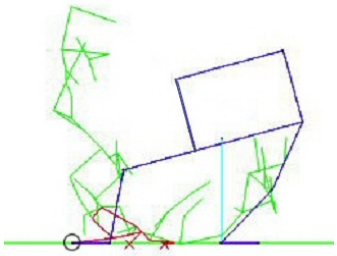


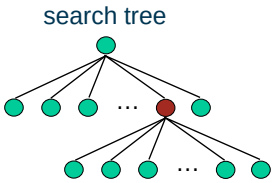


The appropriate single-support mode controller is then invoked and a new RRT is expanded in its parametric space. Now we see the evolution of the free foot position in space.

Sequencing: Search Tree

- Determine paths leading to configurations in a different support mode, and add new leafs



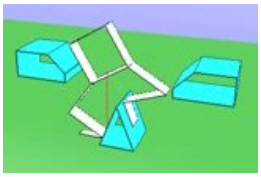


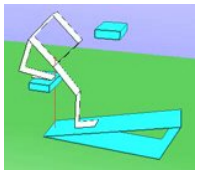
M. Kallmann 2008 - UCM 70

Once again the roadmap leaf nodes leading to a possible change of primitive controller are marked and added to the search tree. The process is then repeated until the task function is satisfied, i. e., until the body center is close enough to the target location.

Sequencing Examples

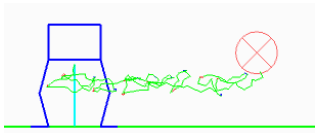
Using obstacles as support

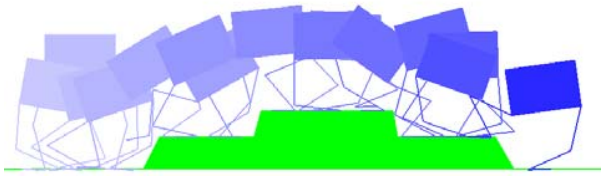




The image illustrates the fact that when a single-support controller is employed, the free foot is analyzed and whenever it is very close to an obstacle, the foot is corrected to be perfectly placed at the obstacle in case the obstacle can serve as support. By doing that we enable the structure to climb on and around obstacles.

Sequencing Examples





M. Kallmann 2008 - UCM 72

The top image shows the evolution of the center of mass in a planned walking motion reaching the target location. The bottom image illustrates the motion obtained for walking over a stairs-like obstacle. Note that smoothing operations have not been applied to these examples.

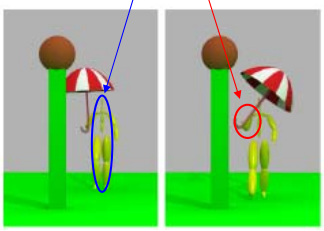
Videos showing these results are available at:

http://graphics.ucmerced.edu/videos/2004_sab_steps.avi
http://graphics.ucmerced.edu/videos/2004_sab_walk.avi

Coordination: Concurrency

- DOFs divided in two parts, for ex:
  - Rear part used by dominant motion skill: walking
  - The rest used by a controlled skill: arm motion

$(p, q^1,\ldots, q^{r-1}, q^r,\ldots, q^n) = (c^w, c^a)$ in $C^W \times C^A$



M. Kallmann 2008 - UCM 73

We now review the approach proposed in [Shapiro et al 2007]. The problem considered now is to plan an arm motion to be executed while the character is walking.

Walking is considered here to be a dominant motion skill and therefore will not be perturbed during the execution of the arm motion. The arm motion is then planned in synchronization with walking. The figure illustrates an arm motion planned to prevent collision between the umbrella and a post during walking.

The used walking motions come from motion capture and therefore ensure the realism of the final results.

Problem Formulation

- Given
  - Initial and goal arm configurations to be reached at times t_a and t_b
  - Parameterized motion m(t) and world state w(t)
- Compute
  - New valid arm motion connecting initial arm configuration at time t_a to goal configuration at time t_b
  - Such that combined body motion ($m^w(t)$, $m^a(t)$) is valid in respect to the dynamic world w(t)

The actual problem being solved can be summarized as follows:

Given:

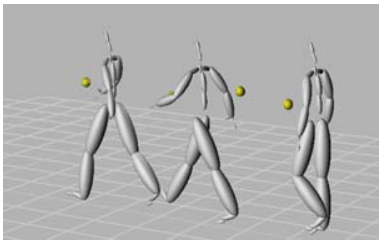
1. initial and goal (arm) configurations c^a_i and c^a_g in C^A
2. initial and goal times t_a and t_b
3. dominant motion (walking) $m^w(t)$ in C^W
4. dynamic world state function w(t)

Compute:

1. valid (arm) motion $m^a(t)$ connecting initial and goal configurations
2. with the new motion starting at initial time t_a and ending at time t_b
3. and such that combined body motion ($m^w(t)$, $m^a(t)$) is valid in respect to the dynamic world w(t)

Search Space

- Sample in configuration-time space:
  - Each configuration (c^a,t) has 7 DOFs for arm or leg, plus time component in [t_a,t_b]
- For every sample (c^a,t)
  - Compose full body posture (m^w(t), c^a) to test validity when dynamic world is w(t)



M. Kallmann 2008 - UCM 75

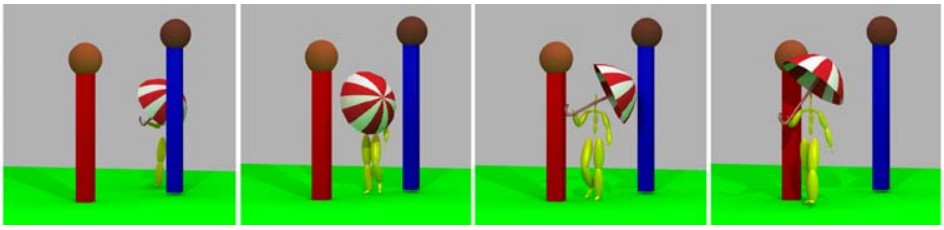
In order to solve the posed problem, the bidirectional RRT planner is modified for sampling in the configuration-time space of the arm. Therefore the usual 7-DOF arm configuration is sampled together with a random scalar value t in [t_a,t_b] representing the time to be associated with the sampled arm configuration.

Then, for every sample, the state of the character and the environment are set in three steps: (1) the full body posture given by the walking motion at time t is applied to the character, (2) the sampled arm posture c^a is then applied to the character overriding the previous arm pose, and (3) the environment state is set to w(t). A distance metric taking into account the time component is used to select the closest node in the RRT for the tree expansion process. Validity tests are then performed to ensure that each tree expansion is valid in the same way as before, with the only difference that each tree expansion has to be monotone in time: the tree rooted at the initial configuration can only be expanded with time components monotonically increasing, while the goal tree can only have time components monotonically decreasing. A solution path is found when the two trees can be connected.

The image in the slide illustrates the sampling process: the spheres represent wrist positions of sampled arm postures at three different random times. For each of these time values the corresponding character posture from the walking motion is shown.

Example: Motion Correction

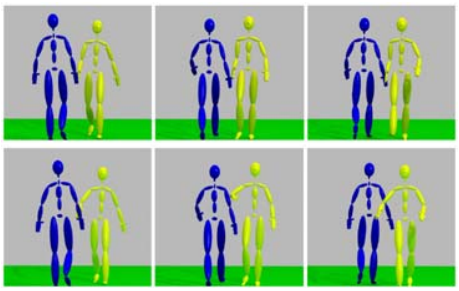
- Arm motion planned to manipulate the umbrella between the two posts, correcting previous collisions



This slide shows how the method can be used for correcting or editing motions. This image illustrates the case of a perfectly valid walking motion which became invalid when two posts were added to the environment (the two posts introduced collisions with the umbrella). The planner could then find a new collision-free arm motion manipulating the umbrella between the posts while the character walks between them.

Example: Controlling Variations

- Sampling heuristics can be biased to preferred ranges in each joint
  - Ex: two different arm motions obtained



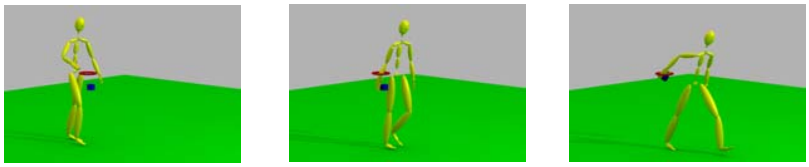
M. Kallmann 2008 - UCM 77

These images illustrate that different results can be obtained by choosing different sampling ranges to be used by individual joint angles.

The result in the top sequence is obtained when high shoulder rotations are not allowed to be sampled. The bottom sequence allows a result with higher arm trajectory.

Example: Dynamic Targets

- Dynamic environments and dynamic targets are addressed together

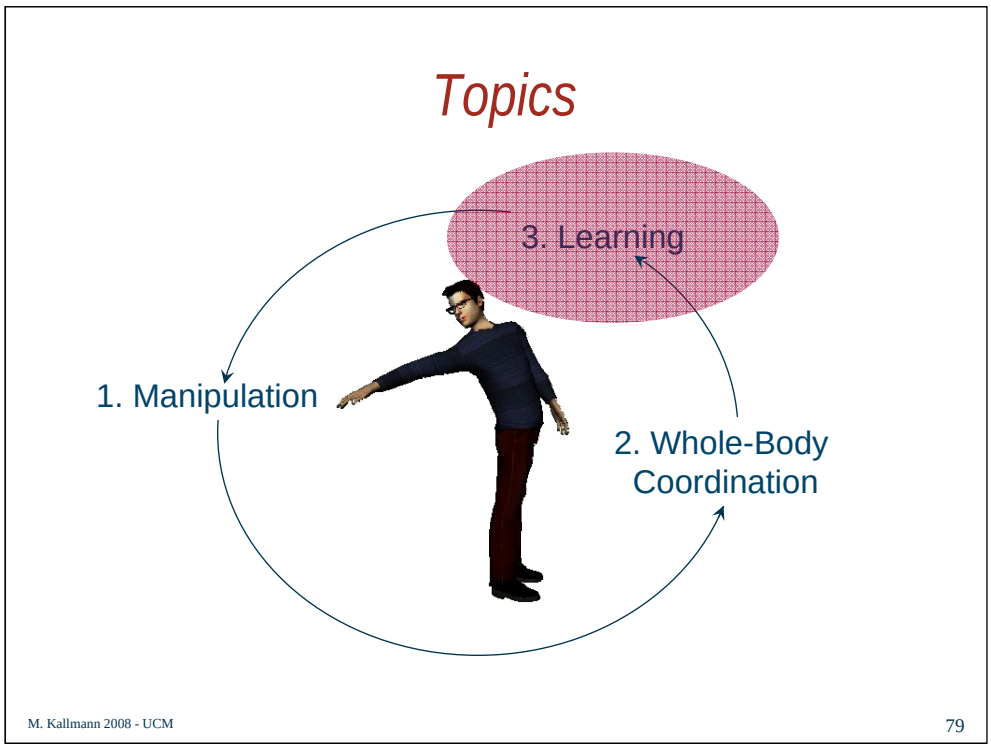


Example of a moving obstacle, moving target, and a moving character

As the sampling is done in the configuration-time space, any time-parameterized motion can be addressed. The example shows a solved problem where the character, during walking, is able to reach for a moving target and without colliding with a moving obstacle.

The full video with this animation and several other results is available at:

http://graphics.ucmerced.edu/videos/2007_i3d_motioncorrection.mp4



The motion planners discussed so far search for solutions from scratch each time a problem is given. Learning mechanism are therefore needed for achieving human-like performances.

Learning to Plan Motions

- Learning
  - Difficult to address, different aspects can be learned
- Representative Papers
  - Dynamic roadmaps
    - [Leven and Hutchinson 2000]
    - [Kallmann and Mataric' 2004]
  - Learning environment models
    - [Burns and Brock 2005]
  - Learning from features of previous motions
    - [Jiang and Kallmann 2007]

It is very difficult to develop learning mechanisms which work well for arbitrary structures and environments. Some representative papers are discussed here.

Probably the first attempt to achieve real-time motion planning was addressed in [Leven and Hutchinson 2000] by maintaining pre-computed roadmaps in a dynamic environment. In [Kallmann and Mataric' 2004] the approach was extended and tested for humanoid manipulation problems.

The approach proposed in [Burns and Brock 2005] concentrates sampling in difficult regions by incrementally learning a model of the environment, which is assumed to be static. In [Jiang and Kallmann 2007] features are learned from previously planned motions and reused to help solving new tasks in environments with few variations.

Due to the specific treatment of humanoid manipulation cases, an overview of [Kallmann and Mataric' 2004] and [Jiang and Kallmann 2007] is presented now.

Dynamic Roadmaps

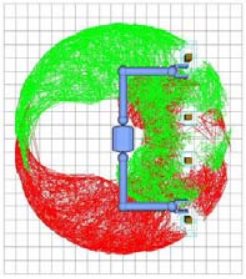
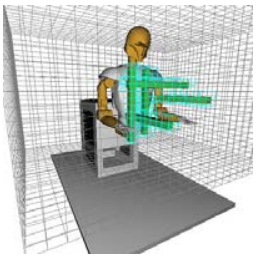
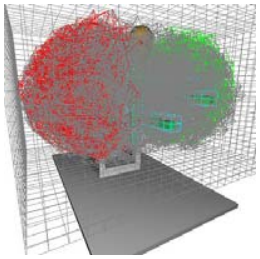
- Pre-Computation
  - PRM-like roadmap pre-computed
  - Spatial subdivision stores, for each cell, all roadmap edges invalidated by the cell
- On-Line
  - Cells intersecting dynamic objects are identified
  - Roadmap edges invalidated by the cells are disabled
  - Remaining roadmap used to solve the problem

PRM-like roadmaps can be built in advance to speed up motion planning queries. However the limitation of pre-computed roadmaps is that they cannot represent dynamic environments. Note that here a dynamic environment means that the position of obstacles in each query may be different, however each motion planning problem is solved considering the environment static during the planning period. The goal is to build a structure which can cope with changes in the environment for improving the performance of several manipulation motions being planned in a changing environment.

Dynamic roadmaps are based on a spatial decomposition of the workspace which gives a fast mapping to decide which portions of the roadmap becomes invalidated when a given cell is occupied by an obstacle. Therefore the roadmap is first computed only considering the static obstacles of the environment, and then continuously updated before each motion planning problem is solved.

Dynamic Roadmaps

- Grid resolution and roadmap sizes tradeoff
  - Finer representations can solve more problems but lead to more maintenance time

M. Kallmann 2008 - UCM 82

The performance of the method will greatly depend on the specific problem being modeled. The main issue to observe is the trade-off between the required maintenance time and the selected grid resolution and roadmap density.

The left image shows an average resolution and a dense roadmap which produce a significant improvement in the computation time obtained for solving planar manipulations of small objects.

The center and right images illustrate the example of a spatial humanoid manipulation environment with larger objects. In this example, improvements are only obtained if the objects do not move significantly from query to query, so that the roadmap maintenance is minimized. These tests were performed in comparison with applying the on-line bidirectional RRT, which does not require any pre-computation. Details about these experiments are available in [Kallmann and Mataric' 2004].

Dynamic Roadmaps

- Drawback: Too much maintenance required in realistic situations

The tests performed with the humanoid simulation example (left image) were modeled in order to simulate the real-case scenario of Robonaut assisting astronauts in a truss assembling operation (right image). In this case several complex arm motions have to be performed around bars which may be constantly moving. This scenario has shown to not be appropriate for dynamic roadmaps.

Learning Attractor Points

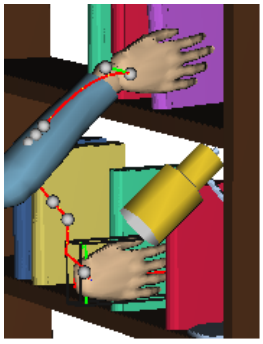
- Three Parts
  1. Extract meaningful attractors from successful paths
     - Line fitting algorithm used to detect important “corners”
  2. Classify attractors in respect to the environment
     - Metric based on local obstacles in respect to init/goal points
  3. Re-use attractors to guide the planning in new similar situations
     - Use an attractor-guided tree expansion strategy

Suitable approaches for realistic situations should require very few maintenance in dynamic environments. The method in [Jiang and Kallmann 2007] addresses this principle by only learning relevant features from successfully planned motions. The learned information is related to the structure of successful paths, capturing not only the structure of the environment but as well specific properties related to the tasks being solved.

The method is based on three main parts: (1) a line fitting-algorithm is used to select meaningful points along each planned motion, (2) these points are then classified in respect with positions of nearby obstacles and positions of the initial and goal locations of the solved task and stored in a database; (3) finally when a new task similar to a previous one is planned, the attractors points of the similar task are used to guide the planning of the new task, and the database entry is updated.

Learning Attractor Points

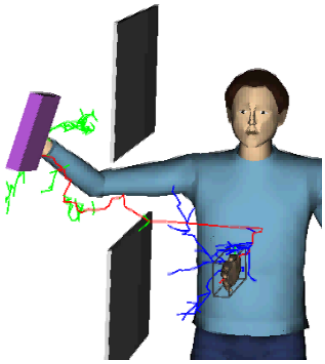
Example of selected attractor points



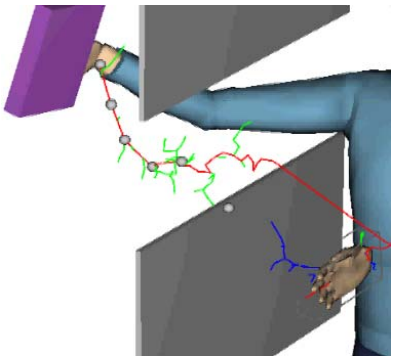
This images shows examples of attractor points selected from a planned path.

Attractor-Guided Planning

Tree expansion comparison



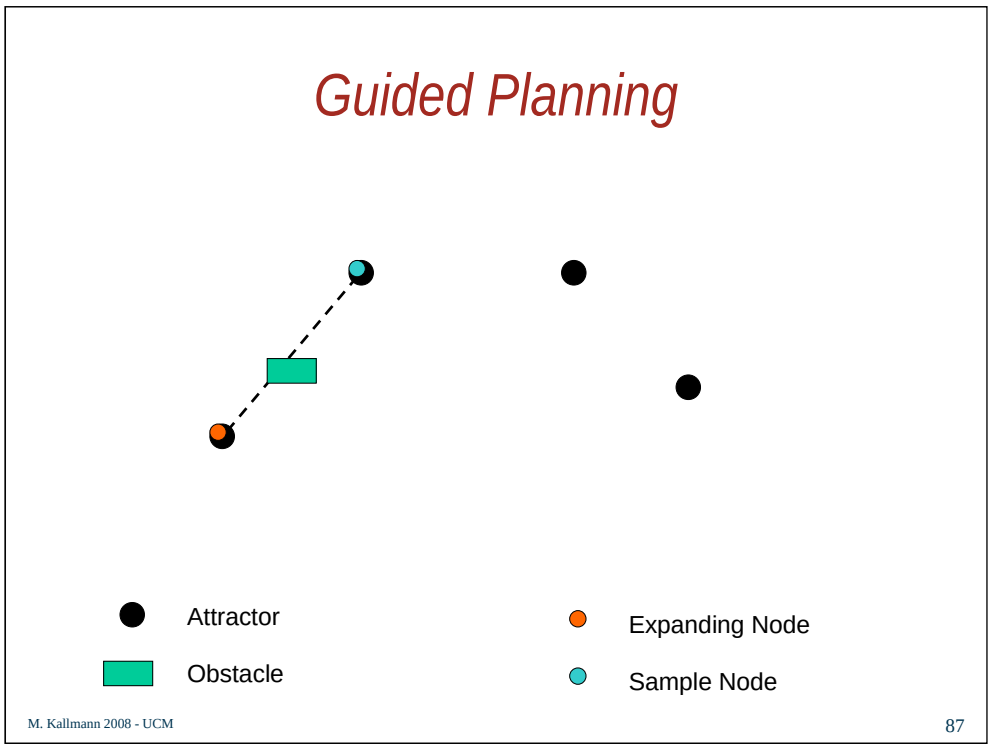
A standard RRT search tree



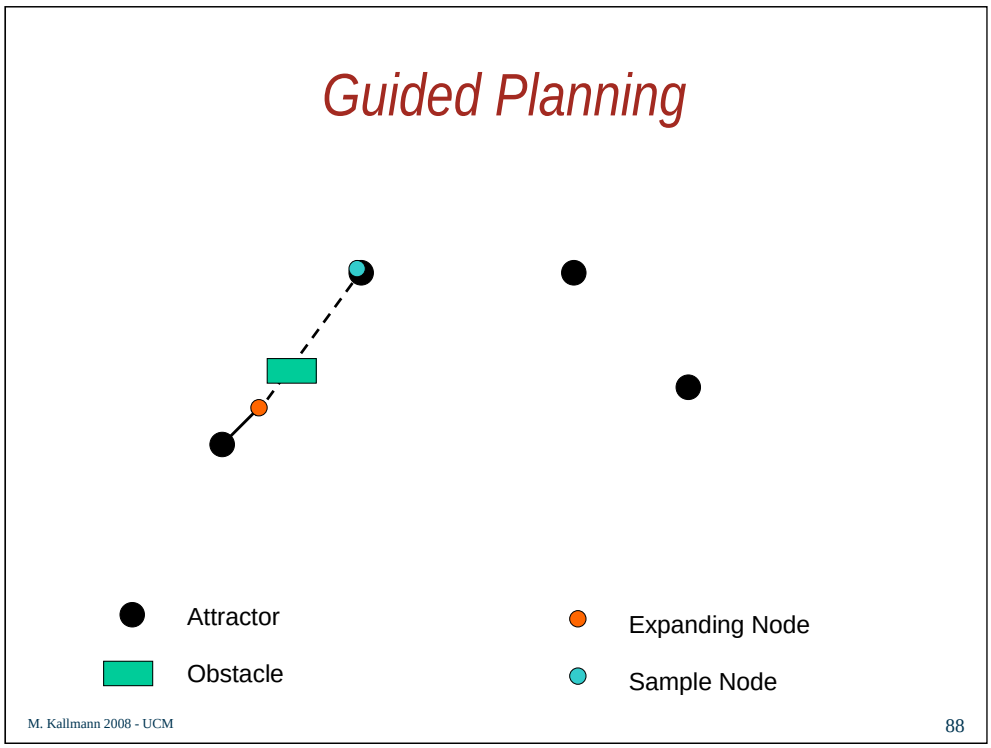
The AGP search tree

Here the left image shows the exploration performed by the standard bidirectional RRT for solving a given motion planning problem.

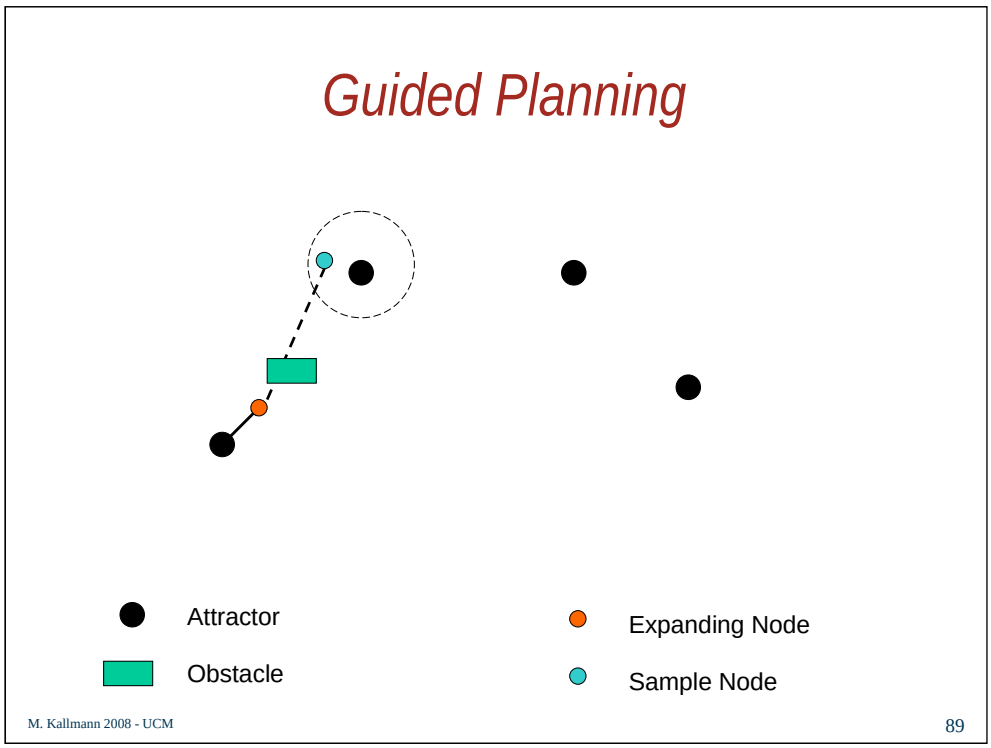
The image on the right shows trees which were guided by attractors computed from a similar previous task. It is possible to note that the trees closely expand towards the locations of the attractors. Branches performing more exploration are only seen in locations where the attractors are no longer useful due to variations on the environment and/or task.



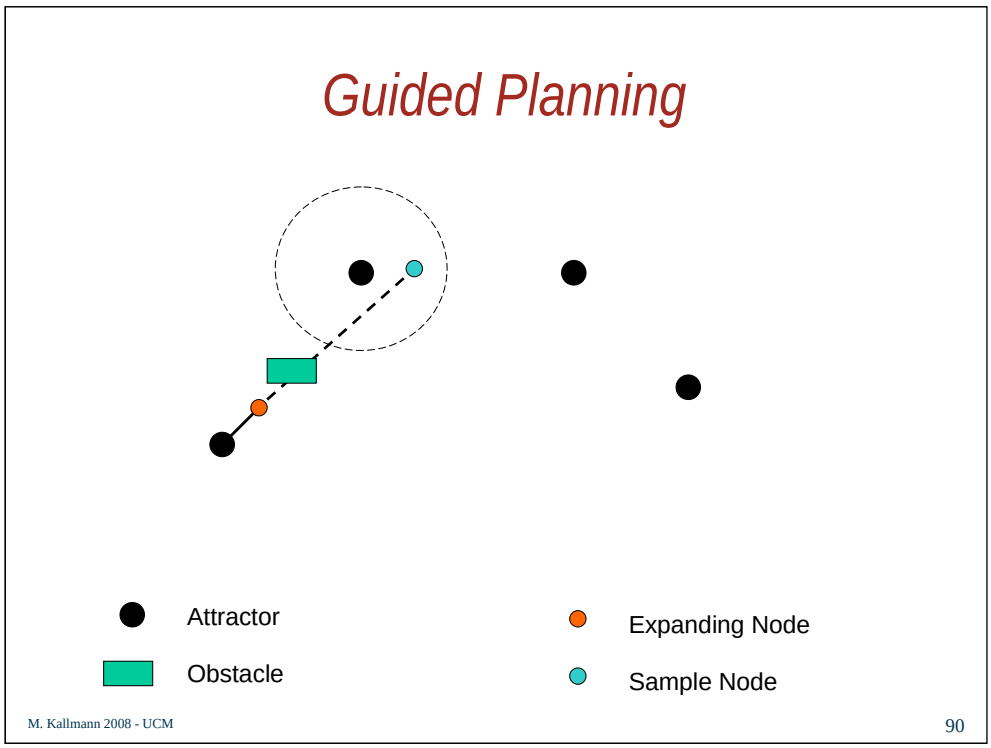
The guided planning works as follows. Suppose a RRT tree is growing directly towards the next attractor in the list of attractors.



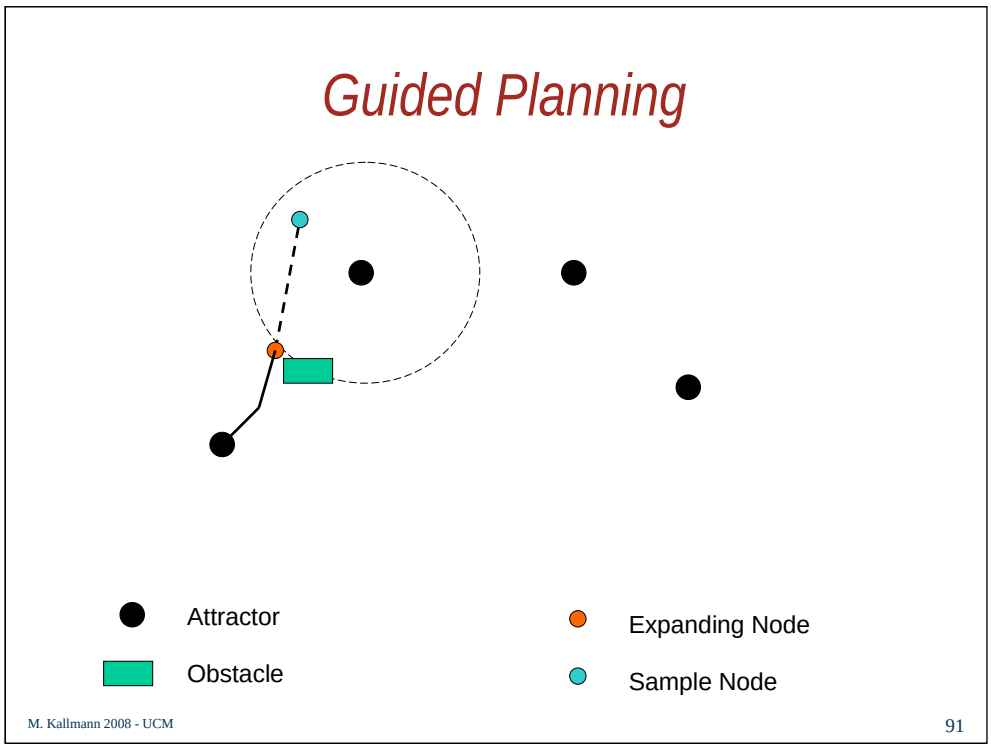
As the tree branch approaches an obstacle which now appears to be in the way, the expansion cannot proceed directly to the next attractor.



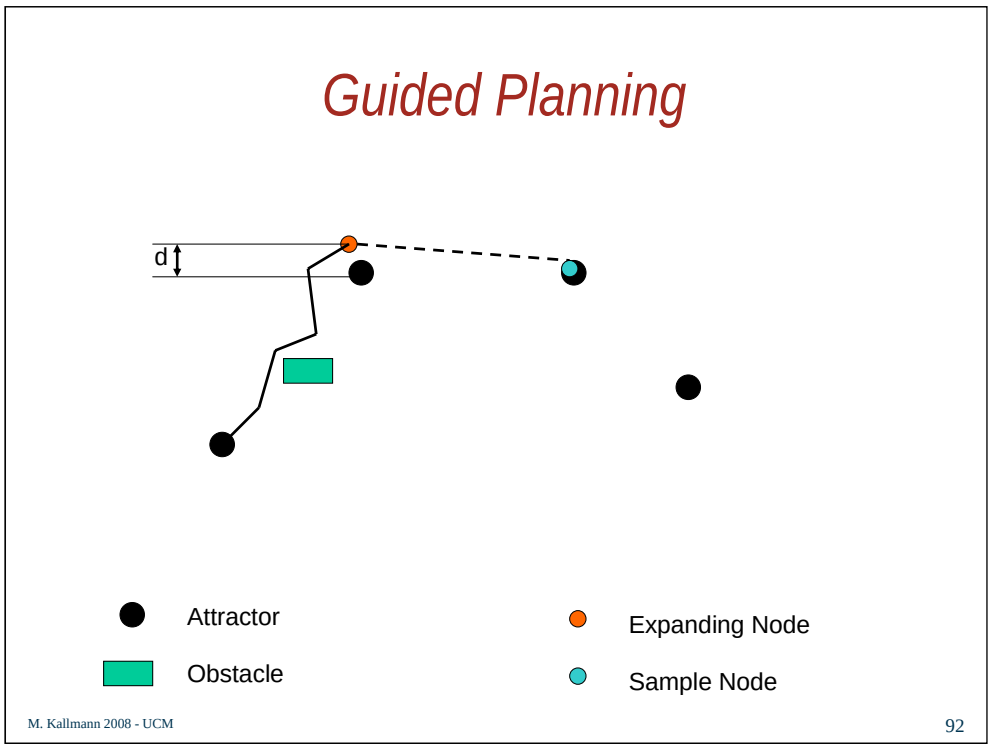
The sampling then occurs in increasingly larger neighborhoods of the next attractor.



Here an even larger neighborhood is needed.



Eventually the branch will be able to escape the obstacle and continue to grow. In the limit case the neighborhood will be as large as the whole configuration space and then the planner will proceed with a normal non-biased RRT expansion.

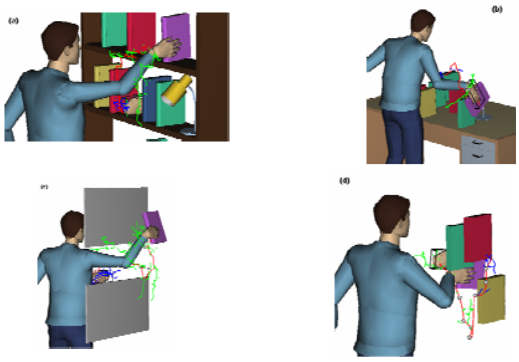


When the expansion gets close enough to the next attractor in a given amount of iterations, the expansion then proceeds toward the following attractor. The trajectory represented by the attractors can therefore be reconstructed and at the same time adapted to few variations in the environment.

If an attractor cannot be reached in a given number of iterations, the expansion tries to proceed towards the next attractor. If no attractors can be re-used, the search abandons the attractors and follows a normal non-biased RRT expansion strategy.

Performance

- Several tests were performed
  - In changing environments with some structure
  - The method was two times faster in average



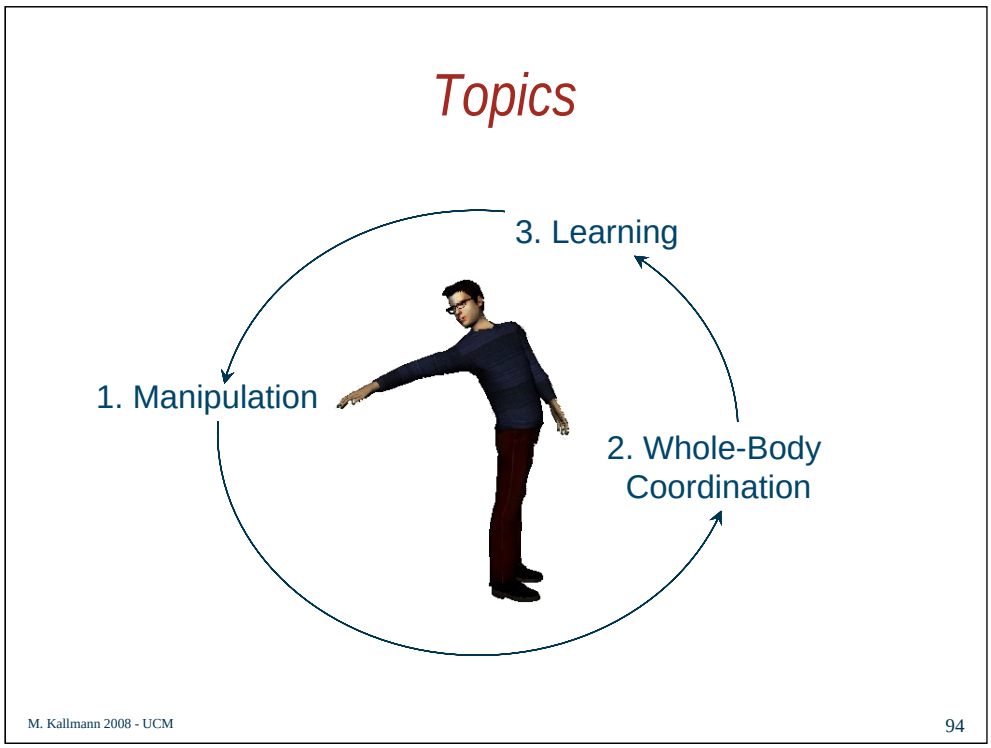
M. Kallmann 2008 - UCM 93

The method will perform well when the obstacles in the environment change in a structured way so that attractors can be often successfully re-used. Fortunately, several realistic manipulation situations seem to follow these characteristics.

Several tests were performed in scenarios involving the relocation of books in different situations and the method showed to be two times faster in average, in comparison with the standard bidirectional RRT.

A video illustrating the approach is available at:

http://graphics.ucmerced.edu/videos/2007_iros_agp.mov



We have now covered our three topics. Hopefully in a near future new methods on these topics will be developed in an integrated fashion, and will lead to efficient manipulation planners for the next generation of autonomous and interactive animated characters.

Final Notes

- Complementary Material
  - The full references of the cited papers are appended to the course notes
  - All videos are available on line and their addresses are given in the course notes

As a reminder, these notes include links to all videos discussed, and the full citations of the references are attached. Most of the cited papers can be found on-line with web search engines.

Further information and resources (including some software) related to this course can be found at our website and as well at the website of the IEEE RAS Technical Committee on Algorithms for Planning and Control of Robot Motion:

http://graphics.ucmerced.edu
http://robotics.cs.umass.edu/tc-apc/Main/HomePage

Finally, our special thanks for our collaborators and students: Amaury Aubel, Ari Shapiro, Jiang Xiaoxi and Petros Faloutsos.

References

[Baerlocher 2001] P. Baerlocher, "Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures", PhD Thesis 2383, Swiss Federal Institute of Technology (EPFL), 2001.

[Bretl 2006] T. Bretl, "Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints: The Free-Climbing Robot Problem", Int'l Jounal of Robotics Research 25(4), 2006, 317-342.

[Burns and Brock 2005] B. Burns and O. Brock, "Sampling-Based Motion Planning Using Predictive Models", In Proc. of ICRA, 2005.

[Gottschalk et al 1996] S. Gottschalk, M. Lin, and D. Manocha, "Obbtree: A Hierarchical Structure for Rapid Interference Detection", In Proc. Of SIGGRAPH 1996, 171–180.

[Hauser et al 2007] K. Hauser, V. Ng-Thowhing, and H. Gonzalez-Baños, "Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task", In Proc. of the Int'l Symposium on Robotics Research (ISRR) 2007.

[Heck and Gleicher 2007] R. Heck and M. Gleicher, "Parametric Motion Graphs", Proc. of Interactive 3D Graphics and Games (I3D 2007).

[Jiang and Kallmann 2007] X. Jiang and M. Kallmann, "Learning Humanoid Reaching Tasks in Dynamic Environments", In Proc. of IROS, 2007.

[Kalisiak and van de Panne 2001] M. Kalisiak and M. van de Panne, "A Grasp-Based Motion Planning Algorithm for Character Animation", The Journal of Visualization and Computer Animation 12(3), 2001, 117-129.

[Kallmann 2004] M. Kallmann, "Interaction with 3-D Objects", In Handbook of Virtual Humans, John Wiley & Sons, UK, 2004, 303-322.

[Kallmann 2005] M. Kallmann, "Scalable Solutions for Interactive Virtual Humans that can Manipulate Objects", AIIDE 2005.

[Kallmann 2008] M. Kallmann, "Analytical Inverse Kinematics with Body Posture Control", Computer Animation and Virtual Worlds, 19(2), May 2008, 79-91.

[Kallmann and Mataric' 2004] M. Kallmann and M. Mataric´, "Motion Planning Using Dynamic Roadmaps", In Proc. of ICRA 2004, 4399-4404.

[Kallmann et al 2003] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping", Proceedings of Eurographics 2003, 313-322.

[Kallmann et al 2004] M. Kallmann, R. Bargmann and M. Mataric´, "Planning the Sequencing of Movement Primitives", in Proc. of the Int'l Conference on Simulation of Adaptive Behavior (SAB), 2004, 193-200.

[Kavraki et al 1996] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Fast Path Planning in High-Dimensional Configuration Spaces", IEEE Transactions on Robotics and Automation 12, 1996, 566–580.

[Koga et al 1994] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, "Planning motions with intentions", In Proc. of SIGGRAPH 1994, 395-408.

[Kovar and Gleicher 2004] L. Kovar and M. Gleicher, "Automated Extraction and Parameterization of Motions in Large Data Sets", ACM Trans. on Graphics (Proc. of SIGGRAPH 2004).

[Kuffner and Latombe 2000] J. Kuffner and J.-C. Latombe, "Interactive Manipulation Planning for Animated Characters", In Pacific Graphics 2000 (short paper).

[Kuffner and LaValle 2000] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning", In Proceedings of IEEE Int’l Conference on Robotics and Automation (ICRA), 2000.

[Kuffner et al 2001] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots", In Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS 2001).

[Leven and Hutchinson 2000] P. Leven and S. Hutchinson, “Toward Real-Time Path Planning in Changing Environments”, Proc. of the fourth International Workshop on the Algorithmic Foundations of Robotics (WAFR), March 2000, pp. 363-376.

[Maciejewski and Klein 1985] A. Maciejewski and C. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," Intl. Journal of Robotics Research, 4(3), 1985, 109–117.

[Redon et al 2004] S. Redon, Y. Kim, M. Lin, D. Manocha1 and J. Templeman, "Interactive and Continuous Collision Detection for Avatars in Virtual Environments". In Proc. of the IEEE Virtual Reality 2004

[Rose et al 2001] C. Rose III, P-P. Sloan, and M. Cohen, "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation", Computer graphics Forum, 20(3), 2001, 239-250.

[Safonova and Hodgins 2007] A. Safonova and J. Hodgins, "Construction and Optimal Search of Interpolated Motion Graphs", ACM Trans. on Graphics (Proc. of SIGGRAPH 2007).

[Schwarzer et al 2002] F. Schwarzer, M. Saha, and J. Latombe, "Exact Collision Checking of Robot Paths", In Proc. of the Workshop on Algorithmic Foundations of Robotics, 2002.

[Shapiro et al 2007] A. Shapiro, M. Kallmann, and P. Faloutsos, "Interactive Motion Correction and Object Manipulation", ACM SIGGRAPH Symposium on Interactive 3D graphics and Games (I3D), 2007.

[Tolani et al 2000] D. Tolani, A. Goswami, and N. Badler, "Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs", Graphical Models and Image Processing, 62(5), 2000, 353-388.

[Yamane et al 2004] K. Yamane, J. Kuffner, and J. Hodgins, "Synthesizing Animations of Human Manipulation Tasks." ACM Trans. on Graphics (Proc. of SIGGRAPH 2004), 2004.

Cited Videos

Reaching
`http://graphics.ucmerced.edu/videos/2001_drawer_lower.wmv`
`http://graphics.ucmerced.edu/videos/2001_drawer_upper.wmv`
`http://graphics.ucmerced.edu/videos/2007_stikarm.avi`
`http://graphics.ucmerced.edu/videos/2008_stikrootmotion.avi`
`http://graphics.ucmerced.edu/videos/2004_rrt_polygon.avi`
`http://graphics.ucmerced.edu/videos/2003_rrt_robonaut1.avi`
`http://graphics.ucmerced.edu/videos/2003_rrt_robonaut2.avi`
`http://graphics.ucmerced.edu/videos/2005_aiide_relocation.avi`
`http://graphics.ucmerced.edu/videos/2003_eg_reaching720x576.mpg`

Whole-Body Coordination
`http://graphics.ucmerced.edu/videos/2004_sab-bothsup.avi`
`http://graphics.ucmerced.edu/videos/2004_sab_rightsup.avi`
`http://graphics.ucmerced.edu/videos/2004_sab_leftsup.avi`
`http://graphics.ucmerced.edu/videos/2004_sab_steps.avi`
`http://graphics.ucmerced.edu/videos/2004_sab_walk.avi`
`http://graphics.ucmerced.edu/videos/2007_i3d_motioncorrection.mp4`

Learning
`http://graphics.ucmerced.edu/videos/2007_iros_agp.mov`