International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:5, 2012

Adopting Procedural Animation Technology to Generate Locomotion of Quadruped Characters in Dynamic Environments

Zongyou He, Bashu Tsai, Chinhung Ko, and Tainchi Lu*

***Abstract*—**A procedural-animation-based approach which rapidly synthesize the adaptive locomotion for quadruped characters that they can walk or run in any directions on an uneven terrain within a dynamic environment was proposed. We devise practical motion models of the quadruped animals for adapting to a varied terrain in a real-time manner. While synthesizing locomotion, we choose the corresponding motion models by means of the footstep prediction of the current state in the dynamic environment, adjust the key-frames of the motion models relying on the terrain's attributes, calculate the collision-free legs' trajectories, and interpolate the key-frames according to the legs' trajectories. Finally, we apply dynamic time warping to each part of motion for seamlessly concatenating all desired transition motions to complete the whole locomotion. We reduce the time cost of producing the locomotion and takes virtual characters to fit in with dynamic environments no matter when the environments are changed by users.

***Keywords*—**Dynamic environment, motion synthesis, procedural animation, quadruped locomotion

I. Introduction

IN the past few years, people have been familiar with high diversities of 3D animated characters because of the enormous strides of computer games and entertaining animations. In particular, walking or running on an uneven 3D terrain plays an important part in dynamic environments. In this paper, we propose a procedural animation technology [1] that it can generate locomotion of the virtual quadruped characters for the purpose of adopting the locomotion of the characters to adapt to the varied dynamic environments. Concerning a basic walking motion, the proposed technology analyzes motions to obtain the important parameters, such as the step length, motion cycle, and stance time, for accomplishing the procedural animation in a dynamic environment. When a user changes the environment dynamically, the current state of the environment will be sensed immediately, and the next footsteps of a character will be predicted in advance. Once the footsteps have been decided, the procedural animation generator selects the most suitable motion model for the character to meet the confronted terrain. After the motion model has been determined, the modified motion parameters which based on the terrain are being substituted for the original one, and the generator calculates the legs' trajectories by means of the Bezier curve. Eventually, the generator makes use of the inverse kinematics to adjust the undesirable motion to satisfy the physical constraints.

Z. Y. He, B. S. Tsai, C. H. Ko, and T. C. Lu are with the Department of Computer Science and Information Engineering, National Chiayi University, Chiayi, Taiwan, R.O.C. (phone: 886-5-271-7730; fax: 886-5-271-7741; e-mail: tclu@mail.ncyu.edu.tw).

The proposed system will generate the smooth motion adapting to the dynamic environments automatically even though users alter the environments.

The development of the animation technology can be roughly classified into two parts, real-time interactive animation and realistic simulated animation. We focus on the interaction between the virtual character and the environment. In this paper, we construct the dynamic environment with the unit cubes. The unit cubes can be placed arbitrarily in the dynamic environment. Therefore, the most effective factor is the way of constructing the dynamic environment. The unit cubes which we proposed can be placed, aligned, and stacked in the dynamic environment for constructing the desired terrain. However, we cannot expect how the user places these unit cubes. We consider the possible problem of suddenly events. The user may construct complex terrain in front of the virtual character. We observe the reaction of the natural animals when they are facing the suddenly events. The natural animals will stop moving and try to understand what they are facing. After figuring out the situation, the animals start to handle the events. Hence, we let the virtual character stop acting when the user placed unit cube closing to the virtual character suddenly for reasonable reaction and calculating time of motion synthesis.

II. Related Work

In this section, we survey two major parts of related methods, one is data-driven approaches, and the other is physics-based simulating approaches. Most of data-driven methods utilize a large amount of captured motion data to generate locomotion or synthesize response motions in dynamic environment. Previous data-driven approaches analyze lots of captured data with objective functions for obtaining identity of motions [2]–[4], such as posture similarity between motions; they classify those motions through the similar identity into categories. Due to above analysis, all captured motion data can be segmented into several categories with specific tags, and these data become a classified motion dataset to serve as reference motions. Subsequently, the authors apply some appropriate physical constraints or rules of dynamics to modify corresponding reference motion as the modified motions, which can fit in with the requirements of the specific situations. Ultimately, these modified motions with other needed motions to accomplish the whole desired simulation [5], [6]. Unlike the data-driven approaches, the physics-based simulation approaches set up a mass of parameters to fit in with the physical rules which stand for a motion, and then the simulation can be accomplished by adjusting the parameters well. Although physics-based simulation approaches focus on physical rules and adjusting the parameters, they also exploit captured motion data to analyze

motion cycles for obtaining motion attributes and keytimes [7]–[10]. These extracted attributes and keytimes can be regarded as a kind of motion, and they improve the motion simulation because the physics-based simulation approaches need to adjust parameters for the main characteristic of a motion.

III. Character Locomotion in Dynamic Environments

We combine the procedural animation with dynamic environments to increase the interactivity of animation editing.

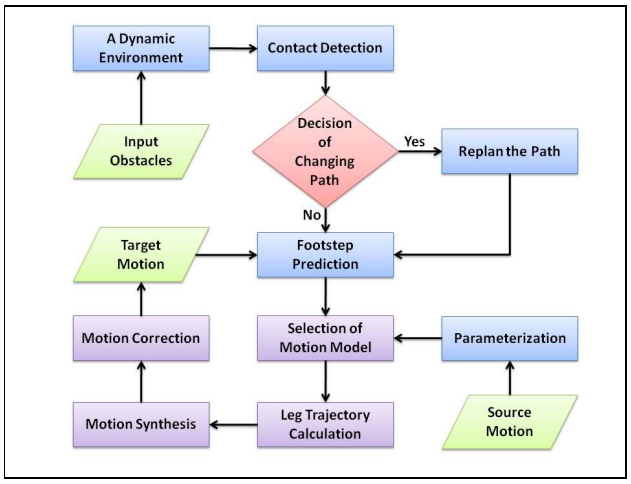

Fig. 1 The system flowchart of the proposed approach

As shown in Fig. 1, first, a user designates two terminal points in a dynamic environment; subsequently, a virtual character will travel from the starting point to the end point. Once the terminal points have been set up, the user cannot modify them anymore. When the virtual character travels in the dynamic environment, the user can place any obstacles anytime and anywhere in the dynamic environment. After the user placed an obstacle into the dynamic environment, the route of the dynamic environment and contact detection will be calculated as soon as possible. If the obstacles that the user has been placed will not influence on the progressing path of the virtual character, the current motion model does not change. Suppose that the placed obstacles block the walking path and form a contact area, we analyze the contact area and make a decision whether the character changes the path according to the new route. Therefore, the virtual character can obtain the right situation of the next motion which fits in with constrains of the dynamic environment. Moreover, the footstep prediction will be carried out when the path is decided. The footstep prediction decides to pick up either a new motion model or the origin one. Finally, we exploit the pre-defined motion model and leg trajectory to interpolate the required motion that adapts to the dynamic environment.

A. Environment Construction

We exploit the unit cube to construct the dynamic environment. The user can add the unit cube any time after setting up the terminate points for the virtual character. Each unit cube can be placed, aligned, and stacked in the dynamic environment for constructing the desired terrain. Note that there are three constrains when placing the unit cubes. The unit cube is not allowed to overlap the virtual character. That is, the unit cube cannot be placed on the current location of the virtual character. We define that aligning the unit cube is placing in a line with the unit cube on the x-z plane. Similarly, stacking the unit cube is placing a line with the unit cube along the y-axis. As shown in Fig. 2, no matter how the user places the unit cubes, a pair of adjacent cubes is full connected with one surface.

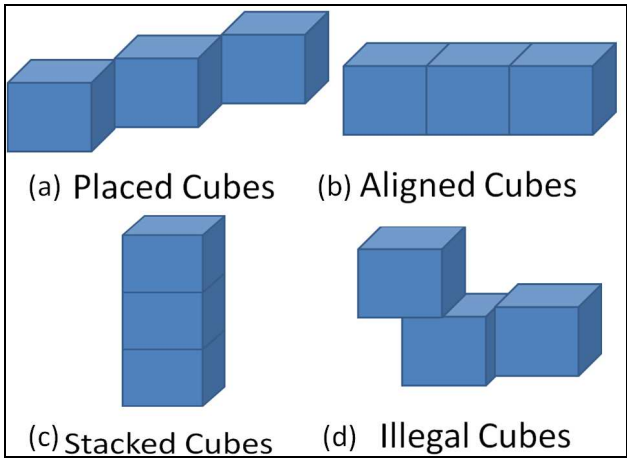

Fig. 2 Unit cubes can be placed, aligned, stacked arbitrarily, but full connected for a pair of adjacent cubes

B. Path Planning

At first, there is no unit cube placed in the dynamic environment. The user can set up the starting point and ending point as the two terminate points for the virtual character. Once the terminate points are decided, we exploit the A* search algorithm, as (1), to calculate the initial path *p(n)* from the starting point to the ending point. The path *p(n)* is the sum of the movement cost *g(n)* from the starting point to the current point through the path, and the heuristic cost *h(n)*, which is an estimated cost from the current point to the end point.

$$p(n) = g(n) + h(n) \quad (1)$$

As soon as the user adds the unit cube to the dynamic environment, we vary the weight w of the position of the unit cube, so that the cost *c(n)* changed with the weight, as (2).

$$c(n)' = c(n) + w \quad (2)$$

Because of the dynamic environment, the user can add the unit cube anywhere. It is necessary to detect whether the unit cube is on the path. This is because the path is a series of nodes, the user may add the unit cube to one or more nodes on the path. If the unit cube is located at the n node of the path, we calculate the cost distance between *n* and *n-1*. The cost distance means the height difference of two nodes of the path. For reasonable motion synthesis, the virtual character cannot get over dynamic terrain with huge height difference. Therefore, we set up a threshold to decide whether the modified path is passable or not. If the cost distance is smaller than the pre-defined threshold, the virtual character gets the corresponding motion model for

motion synthesis. Otherwise, the virtual character re-plans the path from current position to the ending points for the blocked path.

C. Footstep Prediction

Footstep prediction is a necessary section for synthesizing smooth walking motion faster in the dynamic environment. The terrain of the dynamic environment is possibly changing by the user. Therefore, the virtual character cannot know in advance about what they are facing in the dynamic environment. We exploit the footstep prediction to predict the probable footstep position. The stance time is an important time reference for making an effective footstep prediction. Before predicting the footstep, the analysis of the reference motion is needed. We can obtain the angle of rotation, step length and the duration time through the analysis. Note that the duration time is a period time from the foot lifting off the ground to touching on the ground. Given an initial velocity, we can calculate the next footstep by (3) for leg *l*, where the $p_n(l)$is the next position and $p_c(l)$ is the current position of leg *l*.

$$p_n(l) = p_c(l) + v_{ref} t_{dur} \tag{3}$$

We do not consider the terrain when we are predicting the foot step. That is, we predict the footstep without considering the height of the environment. The footstep prediction only concerns about where is the next footstep. When the quadruped animal needs for turning, we do not apply the footstep prediction. In general, the quadruped animals would not move forward when they are making a turn. Hence, we select the turning motion model to achieve the goal of turning, and then we apply the footstep prediction after finishing the turning phase.

D. Procedural Animation

A procedural animation is used to generate animations or motions automatically in real-time. In general, the procedural animation is created by the predefined motions and tuned to fit the restrictions. Therefore, the predefined motions are the kernel of the procedural animation. To get the suitable predefined motions, we define serial motion models for different motions. The motion model consists of the key-frames, parameters and trajectories. When creating the procedural animation, we choose the motion model corresponding to the terrain. Afterwards we modify the parameters of the key-frame for fitting the restrictions of the terrain. Finally, we synthesize the motion by interpolation with trajectories which we calculated for adapting the terrain. The footstep will be decided according to the prediction if the path is not changed. Once the footstep and motion model are decided, the leg trajectory should be calculated for avoiding collision with the terrain and interpolating. We exploit the Bezier curve, as (4), to calculate the leg trajectory *l(t)*.

$$l(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, t \in [0,1] \tag{4}$$

When calculating the leg trajectories, we let the current foot position *p0* and the next predicted foot position *p4* be the ending point of the Bezier curve, as shown in Fig. 3. Besides, we choose the first quartile point, midpoint, and third quartile point to be the control points. If the *l(t)* collides with the terrain, we revise the value *t* for avoiding the collision.

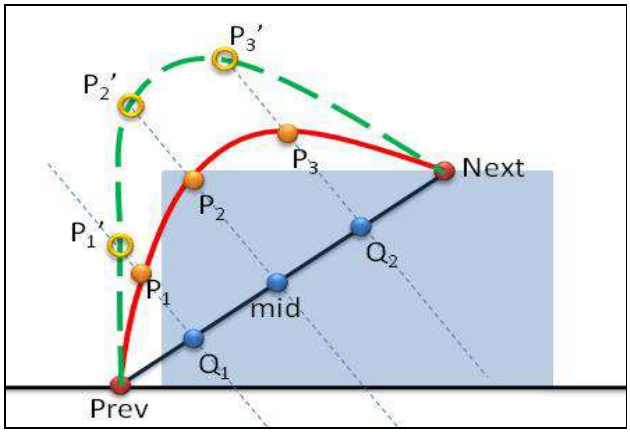

Fig. 3 We choose 5 control points to calculate the Bezier curve for the leg trajectory

We adopt the spinal and quadruped skeleton in this research, as shown in Fig. 4. Note that the spine consists of the bones between the root and the reference point, which can form a straight line or a curve. If the skeleton the user loaded contains the other parts, like the tails, horns, wings, etc., it will not be considered for the procedural animations. Motions of these unique parts will be reserved and added to the procedural animation after synthesized. For a quadruped animal, like the bear, dog, or cat, there are four motions using in their daily life: walking, climbing, turning right or left, and turning back. According to the research of the Bionics, most of the quadruped animals follow the two gaits, the symmetrical gaits and diagonal gaits. Suppose that the quadruped animals start from the right foreleg when waking, the sequence of the symmetrical gaits is right foreleg, right rear leg, left foreleg, and left rear leg. As the same situation, the sequence of the diagonal gaits is right foreleg, left rear leg, left foreleg, right rear leg. In this work, we adopt the symmetrical gaits as the basic gaits. To build up a walking motion model, first we have to construct the motion cycle. As shown in Fig. 5, the motion cycle consists of four concentric cycles, each of cycle stands for one leg.

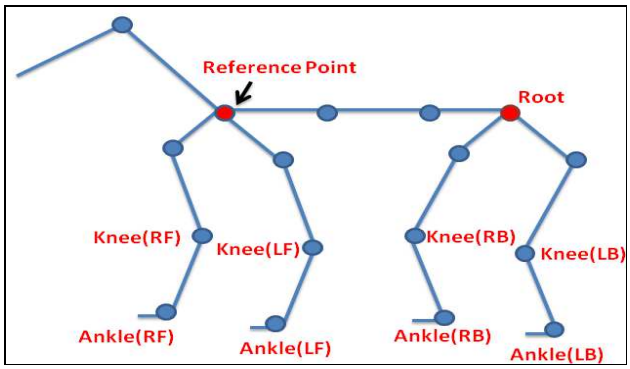

Fig. 4 A spinal and quadruped skeleton must contain the root, reference point, spine, four knees, and four ankles

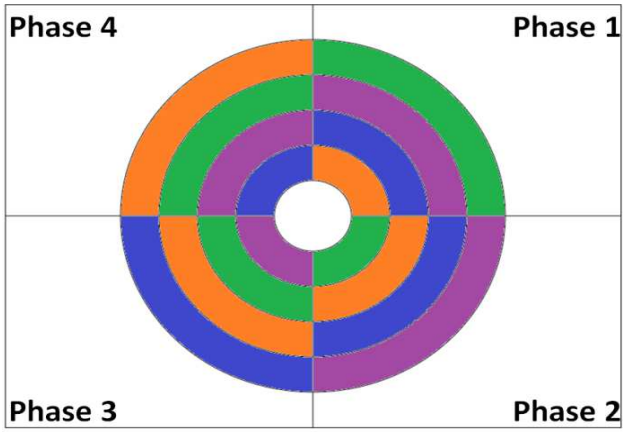

Fig. 5 The motion cycle consist four leg cycles. There are four phases of one single leg cycle: stance (blue), lift (orange), swing (green), and strike (purple)

The quadruped animals would not move one leg at once when they are walking. Actually, they start to move the next leg when the current leg is going from swing phase to strike phase. Fig. 6 shows the key-frames of the walking. The walking motion model starts from the left rear leg. We parameterize the step length, step height, and the angle of swing for flexible motions. Note that the quadruped animals walk differently with the humans. Human's spine will not bend for obtaining moving space. The quadruped animals need to bend their spine for enough moving space when they walking, turning, climbing, or downing stairs. Hence, the spine should bend when the rear leg is going to the strike phase, then lease the spine and move forward when the rear leg is going to the stance phase. For walking motion model, we fix the reference point first. We move the root forward and bend the spine followed the y-axis for creating the moving space. After that, we fix the root, move the reference point forward, and gradual change the spine from curve to straight line.

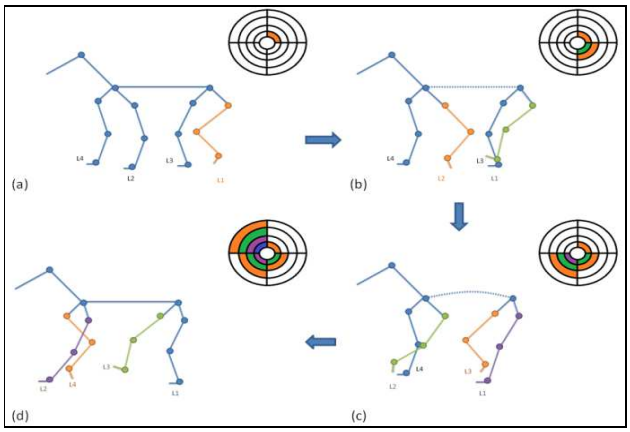

Fig. 6 Four key-frames of the walk motion model (a) The left rear leg enter the lift phase, other legs stay in the stance phase (b) The left rear leg enter the swing phase, and the left foreleg enter the lift phase (c) The left rear leg enter the strike phase, the left foreleg enter the swing phase, and the right rear leg enter the lift phase (d) The left rear leg back to the stance phase, the left foreleg enter the strike phase, the right rear leg enter the swing phase, and the right foreleg enter the lift phase

The quadruped animal is different from the human being when they are passing through a unit cube. Fig. 7 shows the key-frames of the climbing.

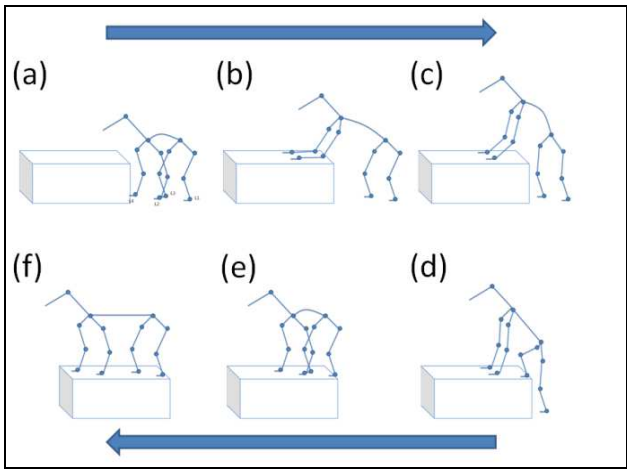

Fig. 7 The key-frames of the climbing motion model

For the beginning, the quadruped animals bend their spine for creating moving space with the same method of walking motion model (Fig. 7(a)). After obtaining enough moving space, we set the root as a fixed point and rotate the spine with an angle Θ along the y-axis. Since we rotate the spine, the two forelegs will be lifted up deservedly. Then we gradual change the spine from high- curvature curve to low-curvature curve and place the two forelegs (including the knees of forelegs) onto the unit cube (Fig. 7(b)). After that, we move the rear legs forward for closing the unit cube. At the same time, we fix the foot of forelegs, lift up the knee of forelegs, and change the spine into a straight line (Fig. 7(c)). So far, the quadruped animal is almost upping the unit cube. At last, we modify the step height of the rear legs to fit the height of unit cube (Fig. 7(d) and (e)), and let the quadruped animal walk on the unit cube with the walking motion (Fig. 7(f)).When the quadruped animal turns right or left, they will move the foot of the turning side first. That is, the quadruped animal will move right foreleg first when they turning right. Fig. 8 shows the key-frames of turning right. First, the foreleg L_1 of the desired direction moves to the desired direction side (Fig. 8(a)). Second, we bend the spine along the opposite direction of the desired direction, move the other foreleg L_2 across the foreleg L_1 from the outer side, and rotate the ankle toward the desired direction (Fig. 8(b)). After that, we move the foreleg L_1 to the appropriate position and adjust to the nature pose of two forelegs (Fig. 8(c)). Note that we change the spine from the curve to the straight line when moving the foreleg L_1 (Fig. 8(d)).At last, we rotate the root to the desired direction, move the outer rear leg for reverting the original stance pose (Fig. 8(e)), and rotate ankle of the inner rear leg toward the desired direction (Fig. 8(f)).

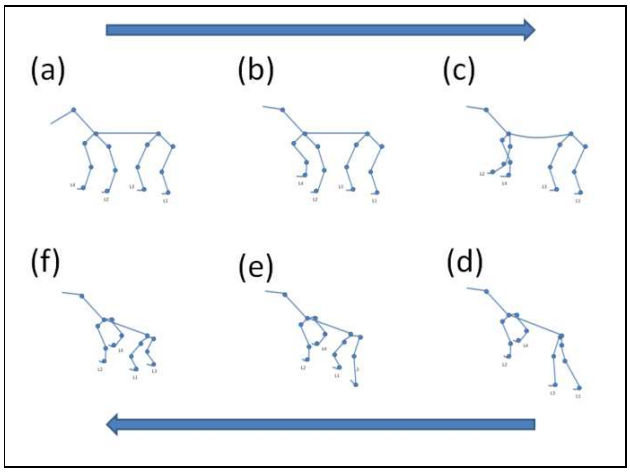

Fig. 8 The key-frames of the turning motion model

IV. Experiment Results

In this section, we experiment on the proposed method for showing the implementation results of interacting with a meshed leopard in a dynamic environment. We develop the proposed method with C# language, and employ the *Unity 3D Game Development Tool* for constructing the whole system. In our system, we demonstrate the 3D model of a leopard in Fig. 9. Users can change the terrain with the defined obstacles. Furthermore, users can also verify the parameters for generating different motions with the defined motion model. As shown in fig. 10, once the parameters and two terminal points have defined for the quadruped character, the proposed system will calculate an appropriate path from one terminal point to another one. No matter whether the quadruped character is moving around in the dynamic environment or not, users can place on the obstacles anywhere except on the quadruped character.


Fig. 9 We take a leopard with mesh as an example for quadrupeds in our system

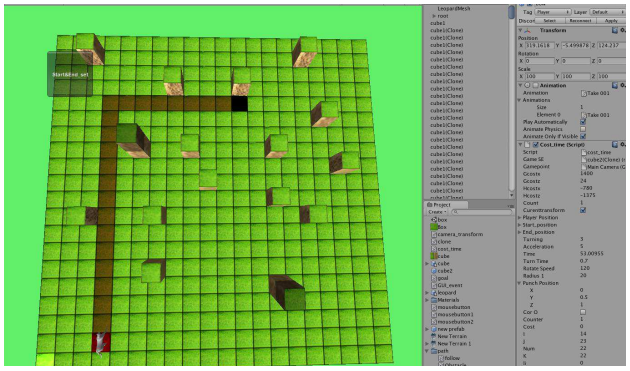
Fig. 10 The right part of this figure is the parameter setting interface, and left part shows the whole view

We demonstrate the four types of locomotion proposed in this paper. Fig. 11 shows the walking motion on an obstacle-free terrain. Users can adjust step length and step height before simulation starts.

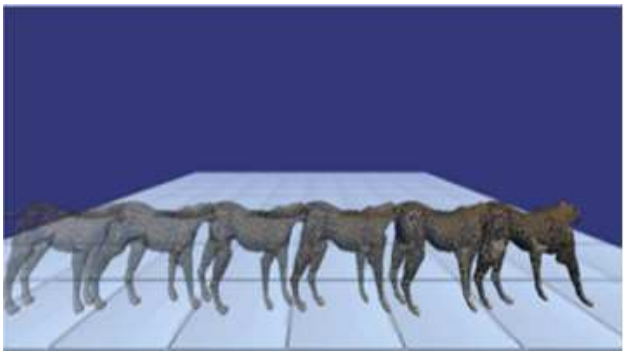
Fig. 11 Walking motion displays in an obstacle-free envirionment.

In Fig. 12 and Fig. 13, we demonstrate the climbing up and down motion with a single obstacle. In our system, the climbing motion will be performed when the quadruped character meet obstacles and movement cost is under the threshold.


Fig. 12 Climbing motion displays with an obstacle. In this figure, the leopard climbs an obstacle

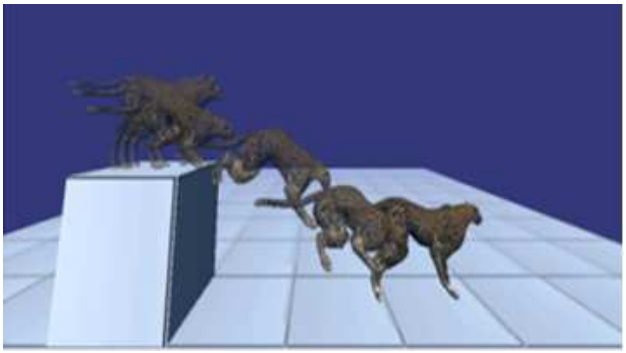

Fig. 13 Jumping down motion only occurs when the height of the next step is lower the current one

In Fig. 14, we show the turning motion when the leopard cannot cross over an obstacle. The turning motion will be taken unless the movement cost greater than current threshold.

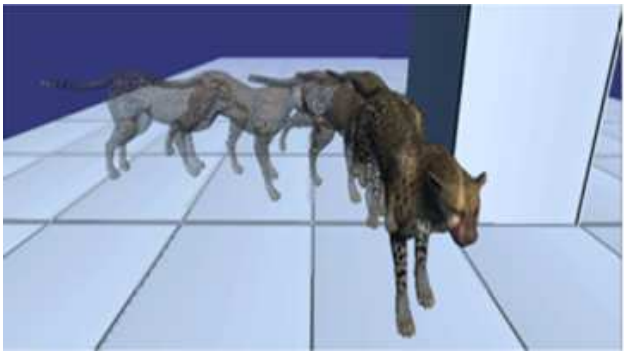

Fig. 14 Turning motion displays with a sets of stacking obstacles

V. Conclusion and Future Works

In this paper, we exploit the procedural animation technology to generate the locomotion of the character in the dynamic environment. We define the motion models on the basis of the observations. By adjusting the parameter of the key-frames or the unit cube, it can lead to another different result of the synthesized motion. The terrain of the environment can be constructed dynamically and arbitrarily with the unit cubes.

We plan to define more motion models to deal with complex terrains. In this paper, we construct the dynamic environment with the unit cubes. The variations of the dynamic environment are limited with the unit cube. Besides, we choose the terrestrial animals as the model for whole motion models. We will choose the aquatic animals or the flying animals as the model for enrich the motion models. In addition, we calculate the walking path the Manhattan distance. That is, the virtual character walks in the dynamic environment with only six directions: front, back, left, right, up, and down. If we adopt the Euclid distance for calculating the path, the virtual character can walk with ten directions. However, the balance of the virtual character and the footholds should be concern when it goes through the diagonal path.

Acknowledgment

This work is supported in part by the National Science Council at Taiwan, R.O.C., under the project grant number NSC100-2221-E-415-019.

References

[1] Z.H. Liang and T. Y. Li "Simulating Human Low-Posture Motions with Procedural Animation," *Proceedings of Computer Graphics Workshop*, Taiwan, 2007.

[2] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica Kate Hodgins, and Nancy S. Pollard, "Interactive control of avatars animated with human motion data," *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 491-500, 2002.

[3] Lucas Kovar and Michael Gleicher, "Automated extraction and parameterization of motions in large data sets," *ACM Transactions on Graphics (TOG)*, Vol. 23, Issue3, 2004.

[4] Rune Skovbo Johansen, "Automated Semi- Procedural Animation for Character Locomotion," *Proceedings of Game Developers Conference*, 2009.

[5] Eamonn Keogh and Chotirat Ann Ratanamahatana, "Exact indexing of dynamic time warping," *Proceedings of the 28th International Conference on Very Large Data Bases*, pp. 406-417, 2002.

[6] Rachel Heck, Lucas Kovar, and Michael Gleicher, "Splicing Upper-Body Actions with Locomotion," *ACM SIGGRAPH posters*, 2007.

[7] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin, "On-line locomotion generation based on motion blending," *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 105-111, 2002.

[8] Min Gyu Choi, Jehee Lee and Sung Yong Shin, "Planning biped locomotion using motion capture data and probabilistic roadmaps," *ACM Transactions on Graphics (TOG)*, Vol. 22, Issue 2, pp. 182-203, 2003.

[9] Jaroslav Semancik, Josef Pelikan, and Jiff Zara, "Interactive synthesis of constrained motion from example movements." *Proceedings of the 4th IASTED International Conference on Visualization, Imaging, and Image*, pp. 878-883, 2004.

[10] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michael van de Panne, "Locomotion skills for simulated Quadrupeds," *ACM Transactions on Graphics (TOG)*, Vol. 30, Issue 4, 2011.