Fast Motion Synthesis of Quadrupedal Animals Using a Minimum Amount of Motion Capture Data

Mankyu Sung

This paper introduces a novel and fast synthesizing method for 3D motions of quadrupedal animals that uses only a small set of motion capture data. Unlike human motions, animal motions are relatively difficult to capture. Also, it is a challenge to synthesize continuously changing animal motions in real time because animals have various gait types according to their speed. The algorithm proposed herein, however, is able to synthesize continuously varying motions with proper limb configuration by using only one single cyclic animal motion per gait type based on the biologically driven Froude number. During the synthesis process, each gait type is automatically determined by its speed parameter, and the transition motions, which have not been entered as input, are synthesized accordingly by the optimized asynchronous motion blending technique. At the start time, given the user's control input, the motion path and spinal joints for turning are adjusted first and then the motion is stitched at any speed with proper transition motions to synthesize a long stream of motions.

Keywords: Animal motion synthesis, motion capture.

I. Introduction

Many approaches have been proposed for synthesizing realistic motions of 3D characters in real time. In particular, the data-driven motion synthesis method that modifies and stitches multiple pieces of motion capture data to create a long stream of motions provides highly realistic motions at a relatively low computation cost [1]-[3]. However, most of those methods focus on human motions. Since humans and animals differ not only in the number of limbs but also in terms of locomotion gait types [4], synthesizing quadrupedal animal motions needs quite a different approach. Horses, for instance, have at least four different gait types, such as walking, trotting, cantering, and galloping. Each of these gaits has its own beat sequence (the order of feet touching the ground) and limb configuration, as shown in Fig. 1. For example, regarding the walking motion, the front left leg touches the ground first, the hind right leg touches the ground 0.25 seconds later, and then the front left leg and the front right leg simultaneously touch the ground 0.5 seconds later. Therefore, when the animal character goes beyond the speed range that one particular gait has, the proper gait must be determined. Even in the same gait type, the motion must be gradually sped up and slowed down according to the user interaction without causing any artifacts. Moreover, quick response from the user input is required when this technique is applied to interactive applications, such as games.

This paper proposes a novel data-driven motion synthesis algorithm for quadrupedal animals. It utilizes motion capture data to build up a "gait graph," wherein the x axis represents speed and the y axis represents the gait types. Then, according to the speed and direction parameters that a user enters, the algorithm determines a particular gait by checking the gait graph. Next, using a cyclic input motion of that gait type, it

Manuscript received Mar. 31, 2013; revised Aug. 29, 2013; accepted Sept. 6, 2013.

Mankyu Sung (phone: +82 53 580 6684, mksung@kmu.ac.kr) is with the Game Mobile Contents Department, Keimyung University, Daegu, Rep. of Korea.

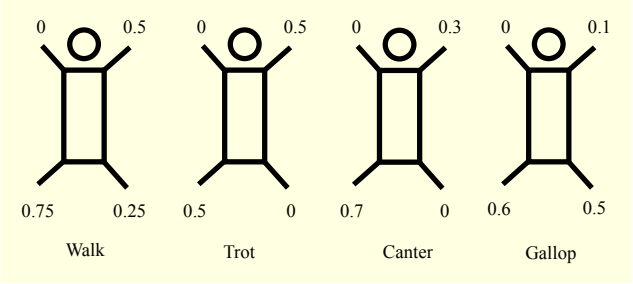

Fig. 1. Four types of gait and their beat sequence. Numbers represent time (s) when each foot touches ground.

computes the Froude number, which is a dimensionless parameter that relates velocity to acceleration according to gravity and the height of the the hip from the ground. Then, it estimates the stride length of the input speed from the gait stride Froude number regression model derived from the motion capture data. With this estimated stride length, it sets the new global root joint position from the input cycle motion and then applies an inverse kinematics (IK) solver to the limb configurations. To synthesize transition motions, which are the intermediate motions transformed from one particular gait (the source motion) to another (the target motion), each leg is treated independently and a search for the best transition frames for each leg on both the source motion and the target motion is conducted, and the asynchronous motion blending is done independently through a predefined number of frames. To change directions upon the user's command, the original motion path and its spinal joint structure are smoothly warped for natural turning. Because building gait graphs, the regression process, and creating transition motions are carried out as preprocessing steps, the proposed algorithm is able to synthesize 3D quadrupedal animal motions in real time. Moreover, since the algorithm is based on short cyclic motions that last less than 1 second, the delay of the response can be minimized when changing the speed and direction through the user interface.

II. Related Work

Computer animation researchers have studied data-driven motion synthesis for the last few decades. However, most of them have focused on humanoid bipedal figures because of data availability [1]-[3]. Due to difficulties in capturing motions of wild animals or dynamic humans, they have used live video sequences such as documentary films to reconstruct 3D animal models or have used existing motion capture databases [5], [6]. Therefore, their methods cannot create completely new motions unless there are live videos with new motions, and only relatively simple motions are possible to capture.

Huang and others proposed a real-time horse gait synthesis algorithm. In that approach, they manually captured horse locomotion data from Eadweard Muybridge's famous photographs and proposed an asynchronous time warping method for synthesizing transition motions [7]. One of the disadvantages of their approach is that because the 3D locomotion is estimated from the 2D pictures, input motions have a limitation in terms of visual quality. Though the proposed method is inspired by theirs, it implements better motion quality because horse motion capture data is used instead. Moreover, the proposed algorithm requires only one cyclic motion per gait type. When the existing algorithm performs the asynchronous time warping process, it does not consider any artifacts, such as foot-dragging on the floor attributed to the severe time increment difference between legs. In the proposed approach, the search for the optimal frame for each leg is based on the objective function that minimizes the standard deviation (STD) of the time difference and checks whether the alternation of legs is correct.

A physical simulation method is another way to generate quadrupedal animal motions. Raivert and Hodgins designed a control system that activates or deactivates actuators to represent some particular behaviors for legged characters [8]. Torkos and de Panne proposed a trajectory-based optimization technique to synthesize motions for quadrupedal animals [9]. In these approaches, users are required to input footprint locations, their timing, and stylistic hints for motions, and then the system applies physics and optimization to estimate the body posture of the quadruped. Wampler and Popović proposed an optimization-based animal locomotion algorithm [10]. In their method, the shape of an animal and its motions are the components of the continuous optimization framework. To animate physically realistic motions, the radius and length of limbs are parameterized so that they are adjusted for the given gait. Although their synthesized motions are physically realistic, they ignore the complexity of the mechanical structure of real animal legs, which sometimes does not convey the details of real animals.

Coros and others proposed a physics-based quadruped gait controller that is able to synthesize a wide variety of gaits, such as parameterized replications of leaping, jumping, sitting, lying down, and standing up [11]. It uses the flexible spine model that connects rear legs and front legs more naturally and allows those legs to make independent decisions for many different gaits. Although this technique allows the quadrupedal animal to show a wide variety of parameterized motions, it requires lots of trajectory input data, including joint height as well as even motion capture data for optimizing the controller.

A lot of experimental biologists have found physical properties of real animal movement over the years [4], [12]. Alexander investigated gaits of bipedal and quadrupedal

ETRI Journal, Volume 35, Number 6, December 2013
http://dx.doi.org/10.4218/etrij.13.2013.0059

animals and found out that mammals of different sizes tend to move in a dynamically similar fashion whenever their Froude numbers are equal [4]. Herein, this fact is exploited and the gait length is modeled against the Froude number through regression using the real horse motion capture data.

III. Algorithm

1. Building Gait Graph

Several cyclic motions of quadrupedal animals are taken as input data. Each motion should have a different gait type. For instance, walk, trot, canter, and gallop cyclic motions are used as input for horse motions. The cyclic motions have exactly the same pose for starting and ending a frame. A gait graph that provides the speed range each gait type covers is built from the input data, which has a constant speed. Suppose that v_1, v_2, v_3, and v_4 are the speeds of input cyclic motions in order (v_1 is the slowest and v_4 is the fastest) and each gait has the speed range $[\alpha v_1, \beta v_1]$, where $\alpha<1$ and $\beta>1$. For a linear connection between ranges of different gaits, βv_1 must be equal to αv_2, βv_2 must be equal to αv_3, and so on. This relationship gives two simple linear systems, $(1+k)v_1=k\beta v_1+\alpha v_1$ and $\beta v_1-\alpha v_2=0$, where k is the constant (k=1.0, in general), which can find α and β by solving this linear system. The gait graph that reflects which speed value falls into which gait type of motion can be plotted from α and β. Figure 2 illustrates the gait graph derived from the input data. Values v_1, v_2, v_3, and v_4 are 7.43, 17.3, 28.7, and 34.2, respectively, for horse motions.

2. Froude Number and Stride Length

The dynamic similarity theory provides a way to predict the gait characteristics when quadrupedal mammals are traveling at a particular speed. It is based on a dimensionless parameter called the Froude number, which relates the velocity of the animal to the acceleration according to gravity and the height of the hip from the ground [4] [13]. The theory says that animals moving at an equal Froude number have similar gaits. The Froude number is computed as follows:

$$Fr = \frac{v^2}{gh}, \tag{1}$$

where v is the velocity, g is the gravity, and h is the height of the hip from the ground when the body is in the standing position. To obtain the relationship between the Froude number and the stride length, Alexander and Jayes proposed the exponential regression model [13].

$$Fr = a(L)^b, \tag{2}$$

where L is the stride length and a, b are variables that must be located.

The proposed algorithm finds a and b through a simple regression by using the input motion capture data. From L and Fr values obtained from cyclic motions, a and b can be estimated by taking the logarithm of both sides, which yields $\log(Fr)=\log(a)+b\log(L)$. This expresses $\log(Fr)$ as a linear function of b in which the slope is $\log(L)$ and the intercept is $\log(Fr)$. From (2), L can be estimated from the new Froude number, Fr.

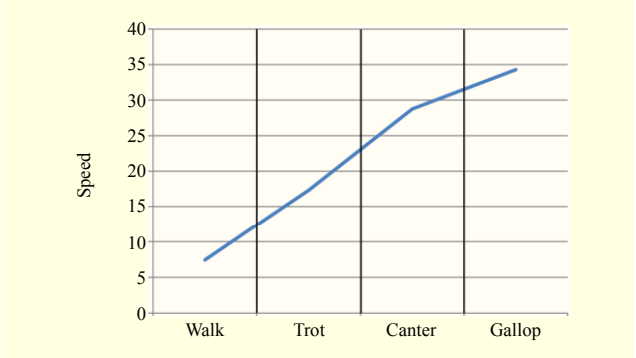

Fig. 2. Gait graph for horse motion.

3. Synthesis of Continuously Varying Speed Motions

Stride length L' can be estimated through (2) according to speed parameter v. Then, from the input cyclic motion M with original stride length L, the new motion M' must be synthesized with length L'. Given n number of joints and m number of frames, the motion M is the set of $\{M^1, M^2, M^3, ..., M^m\}$ and each frame M^i consists of $\{p_r, q_1, q_2, q_3, ..., q_n\}$, where $p_r \in \mathbb{R}^3$ and $q_i \in \mathbb{S}^3$, $0 \le i \le n$. Basically, p_r represents the global position of the root joint, which is responsible for the global position of the whole body, and q_i represents the local joint orientation. Therefore, stride length L can be computed by summing up the root joint position difference between two adjacent frames. To adjust the stride length of original motions, the stride difference is computed, $D=L'-L$, and the offset length, d, is obtained, which is the quotient of D divided by m frames. Suppose that vector p_r^i represents the root joint position at the i-th frame, the new root joint position can be computed as follows:

$$p_r^i + d \cdot (p_r^{i+1} - p_r^i). \tag{3}$$

Adjusting root joint positions is not enough to change the speed of motion because it violates the fidelity of the original motion. Forcing changes of the root joint positions causes a significant foot skating problem. To solve this problem, the entire leg configuration must be adjusted by altering the local joint orientation. After setting new foot positions as end

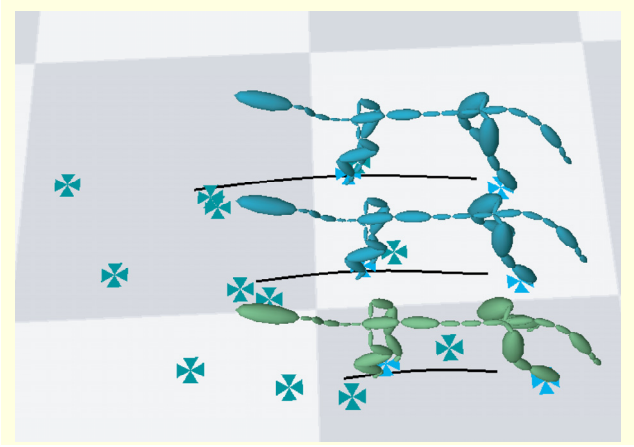

Fig. 3. Three different speed motions.

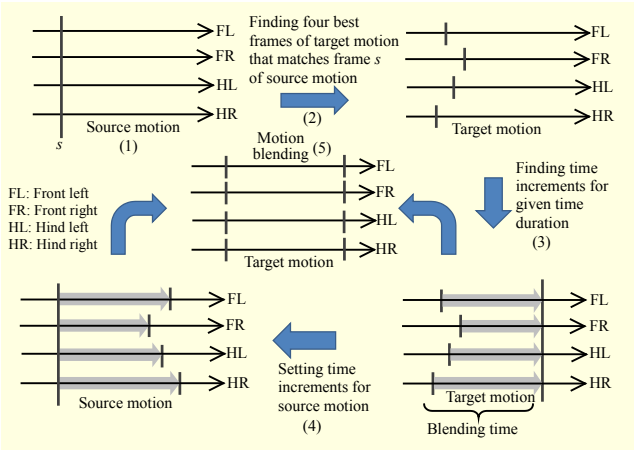

Fig. 4. Asynchronous time warping: From frame *s* of source frames, we first find four best frames of target motion; Then, we compute time increments of four legs for given blending duration; After setting time increments on both source and target motion, we blend source and target motions to create transition motion.

effectors, the IK solver is applied to each frame to obtain the new hip, knee, and ankle orientations. To maintain smoothness, they are filtered out afterward [14]. Figure 3 shows the three different speed motions created from the original input motions.

4. Synthesis of Optimal Transition Motions

The most challenging problem occurs when the animal characters need to change their gaits. When the current speed goes beyond the speed range of a particular gait, some changes in the gait are inevitable. Smooth transit from one gait to another requires transition motions. Because the transition motions do not generally exist as input motions, they must be created artificially from input cyclic motions. Huang and others proposed an asynchronous time warping method for creating transition motions [7], in which the current separate motions

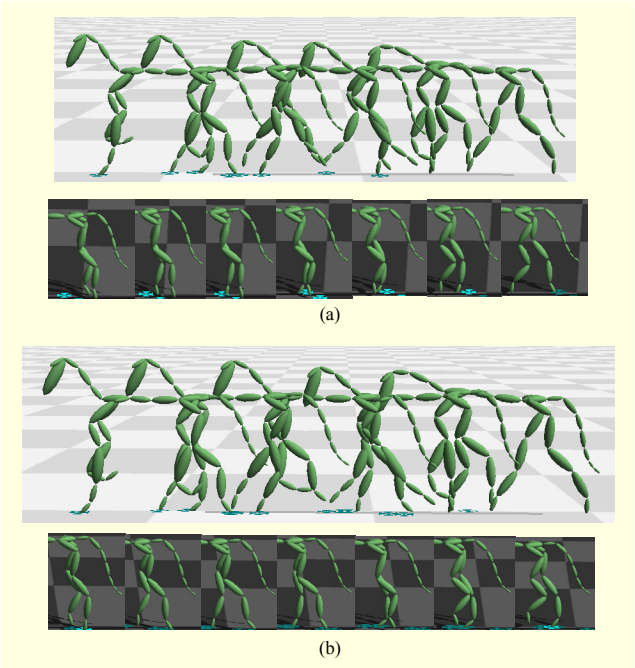

Fig. 5. (a) Left hind leg is dragging on ground (time difference between frames is 0.06 s). (b) Hind legs are alternating properly (time difference between frames is 0.06 s).

correlating to the four legs are blended with those of the new gait while adjusting each leg's speed so that their current motions can gradually converge with the next gait at the end of the transition. Figure 4 illustrates the asynchronous time warp processing step.

One problem of the approach in [7] is that they did not address which starting and ending frames are the optimal frames for blending because the motion quality heavily depends on the starting and the ending frame for blending. Herein, two different starting frames are chosen and two transition motions then compared. If the order of the legs going back and forth during locomotion is not correct, it produces awkward leg-dragging motions. Figure 5 shows a comparison of the two transition motions. The first motion shows that the left hind leg has undesirable "dragging" on the ground whereas the second one has correct alternating legs, which proves the importance of starting frames for blending.

To find optimal frames for blending, each leg is modeled as a point cloud, as Kovar and others did in [3]. To do this, it is assumed that there is a point on each joint position and another point in between the joints. Given two point clouds in the source motion (one gait type) and the target motion (another gait type), the two frames are compared to determine how similar they are in terms of a particular leg configuration. The closed-form solution for calculating distance between two point clouds was proposed in [3]. See the paper for more details. In the proposed algorithm, given frame number *s* of the

ETRI Journal, Volume 35, Number 6, December 2013
http://dx.doi.org/10.4218/etrij.13.2013.0059

source frame, four point clouds for each leg are obtained and each point cloud is compared with all frames of the target motion, then sorted out in the order of numerical difference value. This results in lists of $4 \times m$ frame numbers in that order. For convenience, only the top 10 frames among the frame numbers on the list are kept.

The objective is to find the best four frame numbers among all the combinations of the top 10 frame numbers. The criteria for selection include minimization of the STDs of four frame numbers and whether the blended motion produces any unnatural leg alternation. If the difference of four frame numbers is too big, which also means a high STD, then the time increments for each leg are different accordingly [7]. As a result, it does violate the fidelity of original motions. To prevent this problem, when selecting four frame numbers from the list, the nearest frame should be selected, if possible.

The second criterion to check is whether the blended motion has any artifact, such as "dragging" on the ground, or whether the left and right legs are alternating correctly. This can be confirmed by determining how long the feet touch the ground. Kovar and others proposed an automatic way to detect the foot plant frames in [14]. Herein, their method is adopted and extended to quadrupedal animal motions. In particular, more weight is given to the velocity value than the height value of the end-effector joint to detect whether a particular frame is touching the ground. Once the foot plant frame is determined, verifying whether the left and right legs are alternating correctly is done easily by checking whether the foot plant frames are alternating in the right order.

The overall pseudocode for this process is as follows:

```
For all combination of 4×10 frames {
    /* 10 best frames in the list for FR. FL,HR, HL */
    f1 = FR[i]; f2 = FL[i]; f3 = HR[i]; f4 = HL[i];
    v = STD(f1,f2,f3,f4) //compute the standard deviation
    //asynchronous motion blending
    flag1 = Blend(f1,f2,f3,f4, duration)
    //Does the motion has correct leg alternation?
    flag2 = Check Validity(f1,f2,f3,f4)
    if (v < min && flag == true & flag2 == true) {
        min = v
    }
}
```

Once the four best frames for each leg are found, the time increments of each leg for a given time duration are computed. The time duration is determined by the number of frames of the transition motion. Because the transition motion is not an original motion, it is best that it is as short as possible, but the speed of two input motions (source and target) must be taken into account as well. As a result, if the number of frames of two input motions is m_1 and m_2, then the number of frames of transition motion m_t is set to $\left(\frac{m_1 + m_2}{2}\right) * k$, where k is adjusted by the user. To compute time increments for each leg given the number of frames, a reference frame is set first. In the proposed algorithm, the reference frame is set as the latest frame among the four frames. That is, assume that the four frames found are f_i, where $1 \le i \le 4$, then $\hat{f} = \max(f_i)$. The $\max(f_i)$ returns the latest frame number of f_i, whose maximum number is the ending frame of motion. From the reference frame, $\hat{f}$, and the number of frames, m_t, the time increments of leg i are computed as follows:

$$\delta_i = \frac{(\hat{f} + m_t) - f_i}{m_t}, \quad \text{where} \quad f_i : 1 \le i \le 4. \tag{4}$$

For given time duration δ_i for a leg, the source and target motions are increased by δ_i to get $M^{s+\delta_i}$ and $M^{f_i+\delta_i}$. Because each leg is treated independently, four frames of the source and four frames of the target motion are obtained by the time increments. Then, leg by leg, they are separately blended to obtain the final frame. All other joints that are not part of the leg, such as spine, head, and tail, are blended together. The Spherical Linear Interpolation operation with an increasing weight value is used to blend the joint orientation. Through the increasing weight value, the motions are gradually converted from the source to the target motion. This process continues as δ_i is doubled at each iteration until the number of created frames reaches up to m_t.

One missing point in this process is to determine the global position of the root joint. That is, p_r^i of frame M^i must be located, where $0 \le i \le m_t$. To compute the global position of p_r^i, the offset length of source o_i^s and target frame o_i^t must be computed first, where $o_i^s = \| M^{s+\delta_i} - M^s \|$ and $o_i^t = \| M^{f_i+\delta_i} - M^{f_i} \|$, where o_i^t and o_i^s scalar values represent the distance between the root joints. Then, given direction vector v, indicating the 3D direction vector of motion, $p_r^{i+1} = p_r^i + v * o_i$, where $o_i = (1-w)o_i^s + w(o_i^t)$. In our approach, v is set to point at the z axis.

5. Turning Motion Synthesis

One of the important requirements for applying a motion synthesis technique to real applications, such as games, is the ability to change the direction of a character smoothly. Because there is only a single cyclic motion per gait type as input data, to change the direction, the original direction of the input motions must be changed. Unlike the human character whose spine is upright all the time, the quadruped character has a horizontal spinal structure. Therefore, to change the direction of the input motion, not only must the root joint position trajectory

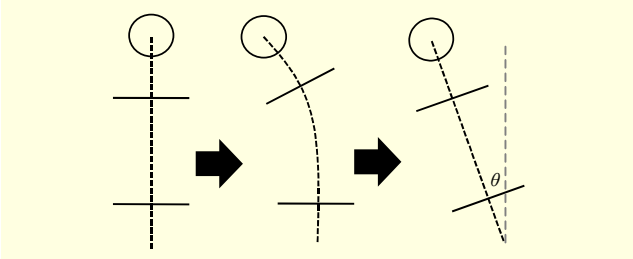

Fig. 6. Spinal joint adjustment for turning.

be curved but the spinal joint configuration (including waist) must be changed. To do that, a simple path editing technique is applied, in which the root joint trajectory is modeled as a curve and the curve is warped to obtain the turning motion [15]. After that, the orientation of the spinal joints is adjusted to ensure the smoothness. Mathematically, to warp the root joint trajectory at frame i, a 2D transformation matrix, T, is constructed with rotation angle $\frac{\theta}{m}i$, where m is the total number of frames and the pivot point is at p_r^0. Then, T is applied to current p_r^i to get the new position, $\hat{p}_r^i$. Because of the gradual change of the rotation angle, the trajectory is also gradually bended accordingly. The next step is to adjust the spinal joint orientation. In general, because the input cyclic motions always go straight, the spinal joints, which are the group of joints constructing the backbone, simply go up and down without bending to left or right while they are moving. When the direction of the motion is changed to the left or right, the spinal joints should bend as well. Figure 6 shows the process of bending the spinal joints. All spinal joints are split into three groups, and each group makes a turn sequentially over time during turning. When the rotation angle of the spinal joints is adjusted, the rotation is only done around the y axis, which is pointing upward to ensure the original motion quality. Changing the root trajectory and spinal joints may produce an unavoidable artifact, such as foot skating. To fix the problem, IK based on the newly established foot positions might be applied as the final step.

6. Control Interface

At the start time, the proposed system keeps receiving control signals from the user. The left and the right arrow buttons are used to turn the character, and the up and down arrow buttons are used to change the speed of the character. An arrow mark is placed in front of the character to indicate the current direction and speed. The direction of the arrow represents the current direction of the character, and the length of the arrow indicates the current speed. The length increases and decreases depending on the current speed. Figure 7 illustrates this arrow user interface. Note that the arrow of the right picture is bigger than that of the left picture because the left reflects walking motion and right reflects trot motion, which is faster than walking. When the left or right arrow button is pressed, a 5-degree turn is made. Similarly, when the up or down arrow button is pressed, the speed increases or decreases 0.1 unit/s.

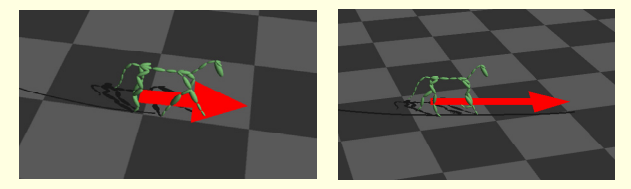

Fig. 7. Arrow user interface: as speed of character is changing, length of arrow is automatically resized.

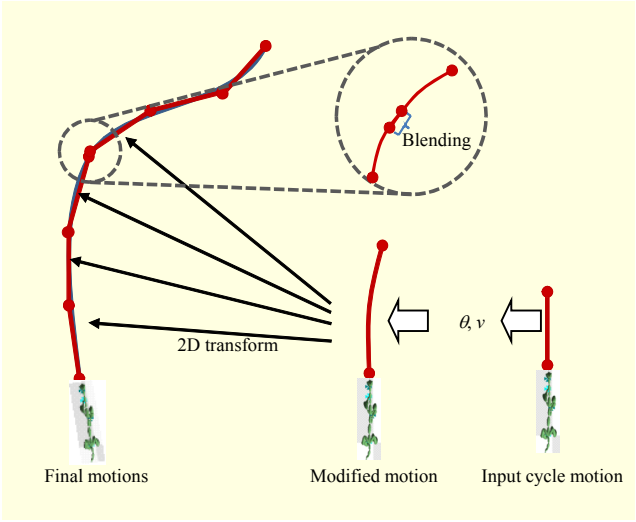

Fig. 8. Motion stitching process for synthesizing long stream of motions.

7. Motion Stitching

Given the speed and direction parameters through the user interface, a long stream of motions is synthesized by stitching modified input cyclic motions and transition motions obtained as described in subsection III.4. When the speed parameter goes beyond the current speed range of the gait, the transition motion created first as a preprocessing step is applied and then the gait motion having exact parameter values is applied. To ensure smoothness of motions when two motions are stitched, the last 10 frames of the first motion are overlapped with the beginning 10 frames of the second motion and then blended with increasing weight values. This procedure is illustrated in Fig. 8. Note that Fig. 8 is a top view, wherein the red lines represent the root joint trajectories.

IV. Experiments

To validate the proposed algorithm, a locomotion system is constructed on the Microsoft Windows platform. OpenGL and the Fast Light Toolkit are used for 3D rendering and putting useful widgets on the control windows. The input cyclic motions are bought through http://www.horselocomotion.com/. The horse motions and the dog motions are tested on this website. As input, 60-Hz captured walking, trotting, cantering, and galloping cyclic horse motions and running and jumping dog motions are used. Besides the motion capture data, to prove the generality of the algorithm, the key-frame animation data of turtles is also tested. For turtle characters, the motions of fast walking and slow walking are used. The lengths of the input motions are shown in Table 1. Note that all transition motions listed in the table are automatically created by the proposed method through the optimal transition search process. Table 2 shows the amount of time needed for transit motion synthesis. This searching time is needed only once for each transition motion. Figures 9, 10, and 11 show screenshots of long streams of horse, dog, and turtle motions, respectively, synthesized through our locomotion system.

Table 1. Input motion clips and their transition motions.

Animal	Motion clip	Length	Type
Horse	Walk	1.20 s	Input
Horse	Trot	0.93 s	Input
Horse	Canter	0.84 s	Input
Horse	Gallop	0.83 s	Input
Horse	Walk2Trot	1.12 s	Transition
Horse	Trot2Walk	1.10 s	Transition
Horse	Trot2Canter	0.88 s	Transition
Horse	Canter2Trot	0.89 s	Transition
Horse	Canter2Gallop	0.84 s	Transition
Horse	Gallop2Canter	0.84 s	Transition
Dog	Run	0.83 s	Input
Dog	Jump	0.59 s	Input
Dog	Run2Jump	0.60 s	Transition
Turtle	SlowWalk	3.10 s	Input
Turtle	FastWalk	6.30 s	Input
Turtle	SlowWalk2FastWalk	4.50 s	Transition
Turtle	FastWalk2SlowWalk	4.60 s	Transition

Table 2. Searching time for transition frames.

Animal	Motion clip	Searching time
Horse	Walk2Trot	56 s
Horse	Trot2Walk	50 s
Horse	Trot2Canter	34 s
Horse	Canter2Trot	32 s
Horse	Canter2Gallop	30 s
Horse	Gallop2Canter	31 s
Dog	Run2Jump	22 s
Turtle	SlowWalk2FastWalk	1 min 20 s
Turtle	FastWalk2SlowWalk	1 min 15 s

1. Horse Motions

For this experiment, the direction of the character is changed to the left every five seconds so that it is moving in spiral manner, as seen in Fig. 9. The total number of frames tested for this experiment is 3,412. The bottom left corner of Fig. 9(a) shows the current frame rate of the window, which is 32 fps. This validates that the algorithm can synthesize motions in real time. While we make the character move in spiral, we gradually increase the speed of the character by clicking the up button. As a result, the character automatically determines its gait type based on the gait graph in Fig. 2 and current speed. Then, the new motions with the exact speed parameter are synthesized. If the current speed parameter goes beyond the range of the current gait type, it puts the transition motion first and goes to the next gait motion. As shown in Fig. 9, the character changes its gait type from walking to trotting, trotting to cantering, and cantering to galloping as the speed increases. As the speed decreases, the character moves in reverse order regarding the gait type.

2. Dog Motions

For this experiment, a single running cyclic motion and jumping dog motion are used as input. Although there are no transition motions between them, the proposed algorithm is able to synthesize the transition motion. By using two input motions and synthesizing transition motions, the algorithm produces a long stream of running and jumping motions. Figure 10(a) shows a screenshot of example motions. Because the algorithm can modify the existing speed of input motions, long and short jumping motions can easily be created from the original jumping motion. Figure 10(b) compares the long original motion with short jumping motions.

3. Turtle Motions

In addition to motion capture data, the proposed algorithm can easily be applied to the key-frame animation data. Two

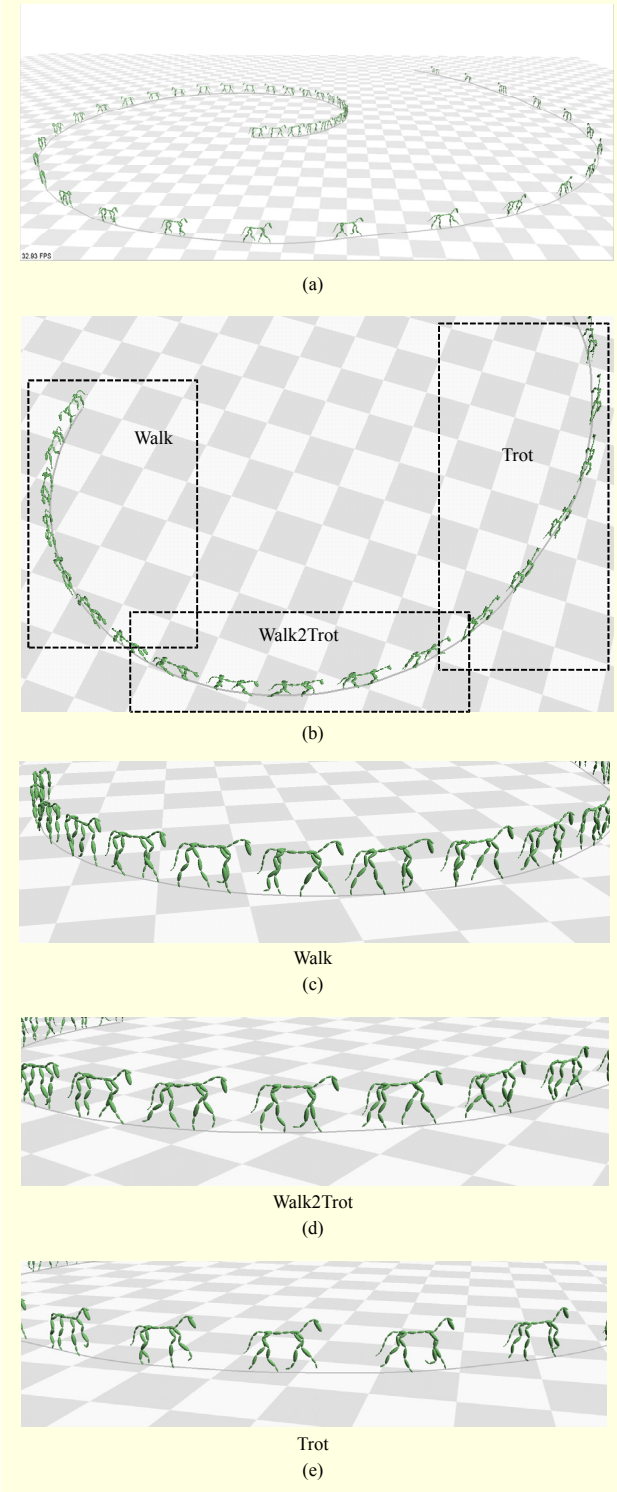

Fig. 9. Screenshots of long stream of horse motions synthesized by locomotion engine: (a) spiral pattern of motion at 32 fps, (b) aerial view of progression of movement from walk to walk2trot to trot, (c) walk, (d) walk2trot, and (e) trot.

cyclic gaits, slow walking and fast walking, are artificially created for testing, and they are converted into motion capture

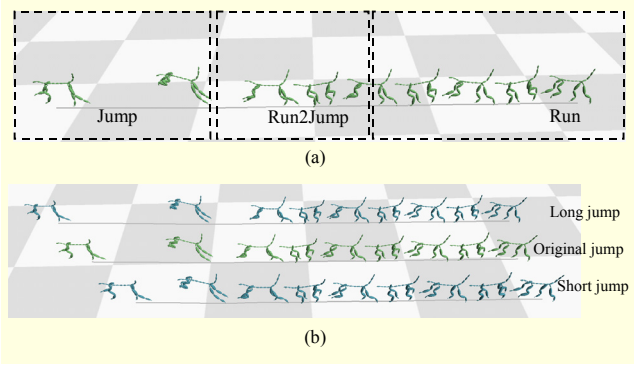

Fig. 10. (a) Synthesized motions of dog changing gait from running to jumping. Jumping length can change, depending on running speed. (b) From original jumping motion, our algorithm can synthesize both long and short jumping motions, depending on user input.

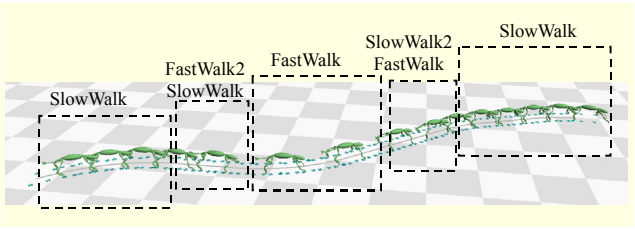

Fig. 11. Synthesized motions of turtle changing its gait from slow walk to fast walk and vice versa. Trajectory is also changing, depending on user input. Note that input motions are manually created by key-frame animation data. Blue stars on ground indicate constrained positions in which foot touches ground.

data format for convenience. From these input motions, the algorithm can synthesize motions at any speed and also create the transition motion between the two gaits. Figure 11 shows a screenshot of synthesized motions of a turtle that changes gait from slow walking to fast walking and then from fast walking to slow walking. When it needs to change its gait type, the transition motions are automatically plugged in for a smooth transition. Note that because only the skeleton of the animal is rendered, the 3D turtle character in the screenshot might not look like a turtle. A results video can be found on YouTube at http://youtu.be/sfKhF9c3r5U.

V. Conclusion

In this paper, a real-time algorithm for synthesizing motions of quadrupedal animals was introduced. Four single cyclic motions of different gaits were used. Then, from the biologically driven Froude number of input motions, the gait length and synthesized motions with different speeds were estimated by applying IK. The speed ranges of each gait type from the regression modeling were also estimated. The core

ETRI Journal, Volume 35, Number 6, December 2013
http://dx.doi.org/10.4218/etrij.13.2013.0059

part of our algorithm is to synthesize the optimal transition motion between the input motions. The four best frames of each leg for the source and the target motion were searched through an optimization process, and an asynchronous motion blending technique was applied to obtain the transition motions. At the start time, from the continuous speed and direction parameter, the long stream of motions was synthesized by stitching the modified motions sequentially. The results of the experiments for the horse, dog, and turtle characters validate that the proposed algorithm is able to synthesize real-time motions.

One of the drawbacks of the proposed approach is that because the motions are synthesized as a cyclic motion unit, quick response from the user interface can be a little bit limited. That is, even if the user presses a button to control speed and direction, the current cyclic motion should be finished before a new motion with different speed or direction begins. However, since the slowest motion cycle is approximately 1 second in duration, excluding turtle motions, the maximum delay of response would not be more than 1 second, which is quite a rare case because the input event should occur exactly when the algorithm synthesizes the first frame of walking motion. After performing several experiments, it was clear that any delay in response is negligible.

In the future, the proposed algorithm should be extended by combining more biologically driven information with the current kinematic-based motion synthesis technique to improve motion quality. It would also be beneficial to solve various retargeting problems that occur when the locomotion algorithm is applied to animals of different sizes, such as cows and mice. In such a case, researchers must extract physical properties of the target animals and figure out how to modify current kinematic data from the physical properties.

References

[1] J. Lee et al., "Interactive Control of Avatars Animated with Human Motion Data," *ACM Trans. Graph.*, vol. 21, no. 3, 2002, pp. 491-500.

[2] O. Arikan and D.A. Forsyth, "Interactive Motion Generation from Examples," *ACM Trans. Graph.*, vol. 21, no. 3, 2002, pp. 483-490.

[3] L. Kovar, M. Gleicher, and F. Pighin, "Motion Graphs," *ACM Trans. Graph.*, vol. 21, no. 3, 2002, pp. 473-482.

[4] R. Alexander, "The Gaits of Bipedal and Quadrupedal Animals," *Int. J. Robotics Research*, vol. 3, no. 2, 1984, pp. 49-59.

[5] L. Favreau et al., "Animal Gaits from Video: Comparative Studies," *Graph. Models*, vol. 68, no. 2, 2006, pp. 212-234.

[6] M. Park et al., "Video-Guided Motion Synthesis Using Example Motions," *ACM Trans. Graph.*, vol. 25, no. 4, 2006, pp. 1327-1359.

[7] T.-C. Huang, Y.-J. Huang, and W.-C. Lin, "Real-Time Horse Gait Synthesis," *Int. Conf. Comput. Animation Soc. Agents 2012 (Comput. Animation Virtual Worlds)*, 2012.

[8] M.H. Raibert and J.K. Hodgins, "Animation of Dynamic Legged Locomotion," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, 1991, pp. 349-358.

[9] N. Torkos and M.V. de Panne, "Footprint-Based Quadruped Motion Synthesis," *Graph. Interface*, 1998, pp. 151-160.

[10] K. Wampler and Z. Popović, "Optimal Gait and Form for Animal Locomotion," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, pp. 1-8.

[11] S. Coros et al., "Locomotion Skills for Simulated Quadrupeds," *SIGGRAPH Comput. Graph.*, vol. 30, no. 4, 2011.

[12] M. Hildebrand, "The Quadrupedal Gaits of Vertebrates," *Biosci.*, Dec. 1989.

[13] R.M. Alexander and A.S. Jayes, "A Dynamic Similarity Hypothesis for Gaits of Quadrupedal Mammals," *Int. J. Zoology* (London), vol. 201, The Zoological Society of London, 1989.

[14] L. Kovar, J. Schreiner, and M. Gleicher, "Footskate Cleanup for Motion Capture Editing," *SCA: Proc. ACM SIGGRAPH/ Eurograph. Symp. Comput. Animation*, New York, NY, USA, 2002, ACM, pp. 97-104.

[15] M. Gleicher, "Motion Path Editing," *Proc. ACM Symp. Interactive 3D Graph.*, ACM, Mar. 2001.

Mankyu Sung received his BS in computer sciences from Chungnam National University, Daejeon, Rep. of Korea, in 1993 and his MS and PhD in computer sciences from the University of Wisconsin-Madison, Madison, WI, USA, in 2005. From January 1995 to July 2000 and then from February 2006 to February 2012, he worked for the Digital Contents Division of ETRI, Daejeon, Rep. of Korea. He has been a professor with the Game Mobile Contents Department, Keimyung University, Daegu, Rep. of Korea, since March 2012. His current research interests include computer graphics, computer animation, computer games, and human-computer interaction (HCI).