The Visual Computer manuscript No.
(will be inserted by the editor)

Costas Panagiotakis · Georgios Tziritas

Snake Terrestrial Locomotion Synthesis in 3–D Virtual Environments

Abstract We present a method for a 3–D snake model construction and terrestrial snake locomotion synthesis in 3–D virtual environments using image sequences. The snake skeleton is extracted and partitioned into equal segments using a new iterative algorithm for solving the equipartition problem. This method is applied on 3–D model construction and on motion analysis stage. Concerning the snake motion, the snake orientation is controlled by a path planning method. An animation synthesis algorithm, based on a physical motion model and tracking data from image sequences, describes the snake's velocity and skeleton shapes transitions. Moreover, the proposed motion planning algorithm allows a large number of skeleton shapes, providing a general method for aperiodic motion sequences synthesis in any motion graph. Finally, the snake locomotion is adapted to the 3–D local ground, while its behavior can be easily controlled by the model parameters yielding the appropriate realistic animations.

Keywords Snake motion modeling · graph exploration · snake animation

1 Introduction

The analysis of image sequences containing moving animals in order to create a 3–D model of the animal and its physical motion is a difficult problem because of the unpredictable and complicated (most of the time) animal motion. On the other hand, there are many techniques and methods in character modeling. We can distinguish them in canned animation from motion capture data and numerical procedural techniques such as Inverse Kinematics [11]. Canned animations are more expressive, but less interactive than numerical procedural techniques, which often appear as "robotic" creatures and require parameter tuning by hand.

Motion capturing could be performed by means of image analysis. Ramanan and Forsyth [22] present a system that builds appearance models of animals from image sequences. Many systems use direct motion capture technology [5] having high accuracy on tracking targets. Motion graphs [14] constructed from motion capture data generate different styles of locomotion by building walks on the graph. The motion graph consists both of pieces of original motion and automatically generated transitions. Our motion planning algorithm is executed in a similar graph. The computer vision based techniques in specific environments can also give high accuracy tracking data using predefined 2–D [21] or 3–D models [25] or without using explicit shape models [24]. One of the first attempts to reconstruct animal motion from videos is Wilhelms's work [28]. Deformable contours (snakes) are used to extract the 2-D motion of each limb's contour from a video sequence of a running horse. However the active contours methods are very sensitive to noise and have parameters which are difficult to tune. In [6] is is presented a 3–D animal motion reconstruction method from segmented video objects. The user provides the object pose in some key frames and the other poses are computed using interpolation.

In robotics, a lot of work has been done on the construction of snake-like robots with elegant and flexible motion, which can move in two or three dimensions [23], [26], [4]. Usually, these robots have many degrees of freedom in order to achieve the flexibility of the real snake and their motion is periodic. Snake-like robots can move on rugged, sandy, terrestrial environments, such as rough or muddy terrains, where the wheeled mechanisms are not effective.

Until now, a lot of work has been done in 3–D animal modeling. A 3–D animal model and its texture mapping can be computed using images captured from specific views and a predefined animal model. This methodology has been applied successfully in snake, lizard and goat 3–D model construction [20]. A.J. Ijspeert [9] designed

Computer Science Department, University of Crete
Greece, P.O. Box 2208
Tel.: +30-2810-393585
Fax: +30-2810-393501
E-mails:{cpanag, tziritas}@csd.uoc.gr

the 3–D model of a salamander using neural controllers. In [17], is described an approach for the design of simple proportional derivative controllers for dynamic locomotion applying it in fish locomotion. Miller [15] simulates muscle contractions of snakes and worms by animating spring tensions. In [29] a physics-based method for synthesis of bird flight animations is described. A set of wing-beats is computed, that enables a bird to follow a specified trajectory. The most recent works try to model the muscle system of a character getting realistic results. Morphing techniques can be applied for creating and controlling the metamorphosis of two animated and textured models [2]. A lot of work has been done on human animation. In [1] is presented a deformation algorithm to create hand animations from images using muscle models. In [30] is simulated a human swimmer, who attempts to follow a dynamic user-defined target by augmenting cyclic stroke control with a set of prespecified variations, based on the current state of the character and its environment.

The motion planning capability under the obstacle avoidance problem support the autonomy of a virtual character. It has been studied using the Probabilistic Roadmap Methods (PRM) [13]. Under PRM, a random sample of configuration space, constructs an accessible point graph (roadmap), which is used to search for a path during the planning stage, yielding good performance. In [7], a motion planning method, based on finite state machines, is used to solve the path planning problem under dynamic obstacle avoidance for virtual humans without knowledge about the environment topology.

The realistic snake movement depends on the mode of locomotion used by the snake. When snakes encounter different environments, they are remarkably adept at changing their pattern of movement so that they can propel themselves effectively. Gray [8] provides a good review of his earlier papers, some of which emphasized modeling, while others had direct observations of snake movement. In [10] is presented the first quantitative kinematic analysis of the major modes of terrestrial snake locomotion that use lateral bending of the vertebral column to generate propulsive forces. In [16] muscular basis and propulsive mechanism of terrestrial lateral undulation in gopher snakes are examined using patch electrodes. The snake mass center trajectory depends mainly on the snake orientation. Thus, depending on exactly how periodic motion is defined, most of the modes of snake locomotion involve some sort of periodic motion or repeating pattern. In both aquatic and terrestrial locomotion a given point on the body periodically moves to the left (L) and then to the right (R). During swimming the frequency of L, R movements is constant along the entire length of the snake. During terrestrial lateral undulation, which is the kind of undulation that we are going to animate, the amplitude and wavelength of the wave of bending can be variable along the length of the snake. Thus, the duration of a cycle of L, R movement can be variable producing aperiodic motion. During concertina locomotion, snakes moving with a steady speed periodically (at regular time intervals) stop although the pattern of left and right movement is highly variable. As our experiments show, the most impressive results are provided when the motion sequence is aperiodic, since if the snake motion is periodic, the motion will be predictable. For this reason, the proposed motion planning algorithm obtains aperiodic motion sequences.

A preliminary version of our work was presented in [20]. In this paper motion synthesis has been significantly improved by adding a model of acting forces, modifying the motion planning algorithm, using mathematical models on skeleton shape transitions derived by tracking data from image sequences, adding the snake behavior control module and by better fitting the snake 3–D shape into the 3–D virtual world. The basic purpose of our work is to create simple and accurate 3–D models and realistic animations of terrestrial snakes in any 3–D virtual environments. We study the curve equipartition problem, whose solution is used in the creation of the 3–D model and on motion analysis stage. The proposed snake model coefficients can be distinguished to those that describe the rotation, translation, scaling of the snake and to those that describe the snake's shape. Moreover, the minimal number of the proposed snake model coefficients, without accuracy loss, provide an easily tuned method that controls the snake behavior. Thus, the proposed snake model coefficients and the motion planning algorithm are the main contributions of this work.

The rest of the paper is organized as follows. Section 2 presents the proposed method. The experimental results are presented in section 3. Finally, in section 4 we summarize and discuss the results.

2 Methodology

2.1 System Overview

The main stages of the proposed method for computing 3–D snake animations are the following:

- Creation of a 3–D Snake Model
- State Graph Construction
- Motion Synthesis

Initially, a 3–D animal model is produced using segmented images obtained by background subtraction. For the 3–D model construction process a set of points located on the skeleton of the snake is extracted. These points are called hereafter *skeleton points*. During the motion analysis stage, the skeleton points are tracked through time in the video sequence. Then a graph of snake skeleton states is created. Each state describes only the snake shape independent from position and global orientation. During the motion synthesis stage, a motion planning algorithm based on a set of possible states is executed on the graph allowing a large number of skeleton

shapes and ensuring aperiodic motion sequences. Mathematical models, derived by tracking data, describe the skeleton shape transitions. The user selects some points of the outer environment in order to specify the trajectory that the animal will trace. However, it is possible that several obstacles appear in the outer world. In addition, the trajectory may not be smooth enough resulting in a zigzagging type of motion. In order that all of the above issues should be tackled, an obstacle avoidance algorithm is proposed yielding a "safe" animal trajectory. Finally, since initially the snake locomotion is computed to be a plane displacement, this locomotion is transformed to fit the 3–D virtual environment.

2.2 Creation of a 3–D Snake Model

Generally, perfect 3–D body reconstruction using only 2-D images as input, is a difficult task difficult task due to the complexity of the animal's 3–D shape. So instead, making use of *a priori* knowledge about the snake anatomy, we assume that its body cross sections are ellipses (see Fig. 2(b)). This way the number of modeling parameters is minimized, increasing the robustness of the reconstruction. Therefore, the proposed method is simple and it yields accurate results using priori knowledge about the snake anatomy. Moreover, we can use the estimated snake skeleton points on motion synthesis stage, applying the method on the whole image sequence.

The snake body is reconstructed using the following "Skeleton Points Extraction" procedure. Two images are given as input to a background subtraction method for extracting the boundary of the snake body (Fig. 1): a background image and a top view image of the snake. Equally spaced points from the medial axis of the boundary, *skeleton points*, are then chosen as the centers of these ellipses (Fig. 2(a)). We propose an iterative curve equipartition algorithm that segments any 2–D curve to equal segments given the first and the last point of the curve (see section 2.3). The texture mapping is done in the same time using the mirroring technique, each couple of antisymmetrical from ellipse center points is corresponded to the appearance of a snake image pixel. The ellipse size at a point p_n is determined based on the distance d_n between its center and the boundary. Its eccentricity can be given as a known constant parameter, as it is almost the same for all ellipses. Otherwise, it can be computed using as input image instead of a top view image and a beside view image of the snake. All of the produced ellipses make up the final 3–D model of the snake.

2.3 Curve Equipartition

The "Skeleton Points Extraction" procedure is based on curve equipartition (EP), and it is used on 3–D snake model creation (section 2.2) and on motion analysis stage (section 2.4). The EP problem can be defined for any 2–D continuous curve. Let A, B be the start and the end point of the curve, respectively. The EP problem is to locate $N-1$ consecutive curve points, so that the curve can be divided into N equal length chords, under the constraint the first starts from A and the last ends on B (see Fig. 3). We have proved [18, 19] that the above problem has always at least one solution providing algorithms that solve the problem.

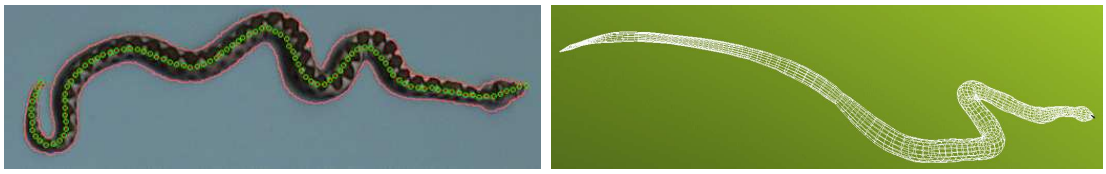
(a) (b) (c)

Fig. 1 (a) A snake image, **(b)** the background image and **(c)** the extracted snake silhouette.

The above problem can have more than one solutions depending on curve shape and number N. As N tends to infinity the problem solution (equal chords) will be unique and it will provide an approximation of the curve (see Fig. 3). Next, we present a steepest descent based EP algorithm (see Algorithm 1) that converges to a solution, and it is efficient for large N. Let $P_i = C(t_i)$, $i = 0, \cdots, N$, $P_0 = A, t_i < t_{i+1}$ be the curve points computed at an iteration. In each iteration, point P_i is computed based on the constraint that the line segments length ($|P_{i-1}P_i|$) is constant. Finally, P_i will converge to a solution, this means that $P_N \simeq B$. The first $N-1$ segments will have absolutely the same length. Let T and r be the convergence error and the algorithm learning rate ($r < 1$), respectively. The final solution, that the above algorithm computes, will be very close to the real one as the length of each chord will be exactly the same.

$P_0 = A$
$s = \dfrac{curvelength}{N}$
repeat
  for *i=1* **to** N **do**
    $P_i = C(t_i) : t_i > t_{i-1} \wedge |P_{i-1}P_i| = s$
    /* If there is not solution, we compute point P_i using mirroring technique, $\Rightarrow$ the P_i will be out of curve.*/
  end
  $s = \begin{cases} s + r\frac{|B-P_N|}{N}, & P_N \text{ is curve point} \\ s - r\frac{|B-P_N|}{N}, & P_N \text{ is being out of curve} \end{cases}$
until $|P_N - B| < T$

Algorithm 1: Steepest Descent Based Curve EP.



(a) (b)

Fig. 2 **(a)** The skeleton points (green points) and the snake boundary (red curve). **(b)** A 3–D wire-frame model of the snake.

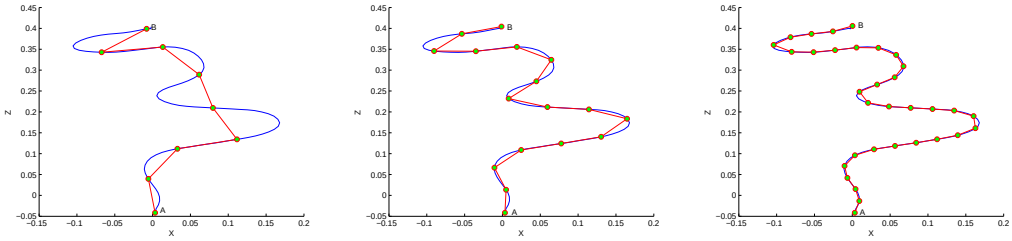
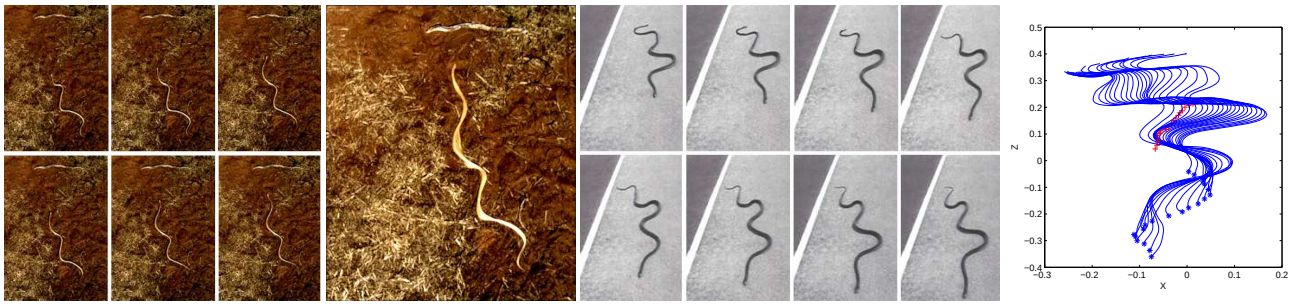
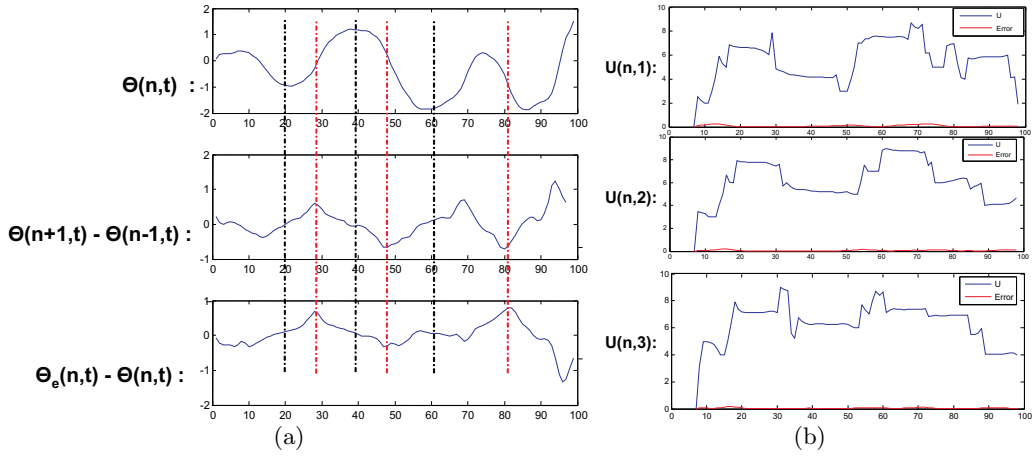


Fig. 3 EP examples for different N. The higher the N, the better the curve approximation.



(a) (b) (c) (d)

Fig. 4 **(a)** Frames of a snake motion in a high friction surface and **(b)** the tracking results using a maximum intensity filter over the time. **(c)** Frames of a snake motion in a low friction surface and **(d)** the snake skeleton points tracking results.



(a) (b)

Fig. 5 Motion analysis results (in rads).

2.4 Motion Analysis Stage

In this section, the motion analysis stage is described. The skeleton points are tracked with pixel accuracy in the whole image sequence using the "Skeleton Points Extraction" procedure. The tracking accuracy has been improved using stable camera and high contrast background. We have captured terrestrial snake locomotion into various types of surfaces in order to measure how the friction affects the snake locomotion.

Let $\Theta(n,t)$ be the angle between the horizontal axis and the skeleton on point n at frame t (see Fig. 6(a)). When the snake is moving in high friction surfaces (Figs. 4(a), 4(b)) it holds that each skeleton point traces the head point trajectory: $\Theta(n,t) = \Theta(n-U(t), t-1)$, where $U(t)$ denotes the snake pulse speed at frame t. Otherwise, the snake slides on low friction surfaces (Figs. 4(c), 4(d)). Let $\Theta_e(n,t)$ denotes the estimation of $\Theta(n,t)$ using the model $\Theta_e(n,t) = \Theta(n-U(t), t-1)$. Then, the absolute estimation error $|\Theta_e(n,t) - \Theta(n,t)|$ is proportional to the first derivative of $\Theta(n,t)$ with respect to n ($\Theta_n(n,t)$) (see Fig. 5(a)). Let $U(n,t)$ denotes the n snake skeleton point pulse speed at frame t. $U(n,t)$ can be estimated by applying the linear model : $\Theta_e(n,t) = \Theta(n-u(n,t), t-1) + w(n,t)(\Theta(n+1-u(n,t), t-1) - \Theta(n-u(n,t), t-1)$, where $u(n,t) \in \{1,2,\cdots\}, w(n,t) \in [0,1)$. Then, it holds that: $U(n,t) = u(n,t) + w(n,t)$. For more robust estimation, $U(n,t)$ was estimated in nine points neighborhoods using least mean square error minimization. Fig. 5(b) illustrates results of the estimation.

2.5 State Graph Construction

Since the input video sequence contains a limited number of motions and skeleton shapes, it is important that the motion synthesis algorithm can generate new unseen motions, thus producing realistic and non-periodic snake animations. For this reason, we have not used simple mathematical models for motion description, but instead a novel technique has been proposed and implemented. According to this method, a graph is constructed, whose nodes correspond to states of the snake skeleton. A graph node is independent of translation, rotation and scaling of the snake, describing just the snake shape. Then the stage of shape motion synthesis merely amounts to traversing suitable paths through this graph.

At first, a skeleton state representation will be described. Since each frame contains a fixed and relatively large number (100) of skeleton points, a more compact representation of the skeleton state is obtained using Fourier Transform coefficients. This representation will be related to the local curvature of the snake shape and will thus facilitate our motion synthesis. More specifically, let $A(s_k)$ be the angle computed at the equally spaced skeleton points s_k (Fig. 6(a)). Let $\hat{F}(u)$[1] be the Fourier transform of the sequence $A(s_k)$ (Fig. 6(b)).

Ignoring the first Fourier transform coefficient, which measures the global rotation of the snake's shape, the next seven Fourier transform coefficients, say $\{C(i); i = 1, \cdots, 7\}$ of $\hat{F}(u)$ are chosen to represent the state of the snake skeleton. Due to the smoothness of snake's shape, these coefficients suffice to recover the angles $A(s_k)$ because they contain about 99% of the signal energy (see Fig. 6(b)), on average in the tracking data. Therefore, a good approximation of the shape is reconstructed, reducing noise at the same time. In addition, these coefficients are invariant to translation, rotation and scaling of the snake's shape.

Having defined a representation of the skeleton state, the construction of the graph, that will be used for the motion synthesis, will be now described. Each node of the graph corresponds to a skeleton state, *i.e.* a 7-tuple of complex numbers. Since a limited number of skeleton states can be extracted directly from the video frames, more "random" skeleton states need to be generated based on the existing ones using the below described sampling algorithm.

Let $C_t(i)$, $i = 1, \cdots, 7$ be the existing skeleton states for all video frames[2] $t = 1, \cdots, n$, where n denotes the number of video frames. Let $R_k(i)$ be a new random skeleton state to be generated. To completely define R_k the modulus and phase for all 7 components of R_k need to be set. Each modulus $|R_k(i)|$ of the new state is strongly related to the curvature of the shape, and so the valid values should lie between the corresponding minimum $C_{min}(i)$ and maximum $C_{max}(i)$ modulus of all the existing states,

$$C_{min}(i) = \min_t(|C_t(i)|), \quad C_{max}(i) = \max_t(|C_t(i)|).$$

The angle $\angle R_k(i)$ is related to the position where the snake skeleton is curved. Since the snake skeleton can be curved at any of its skeleton point, no constraints are imposed on $\angle R_k(i)$. So $R_k(i)$ is defined by the following equation, the $r_k(i), g_k(i)$ being random numbers uniformly distributed in the interval $[0,1]$:

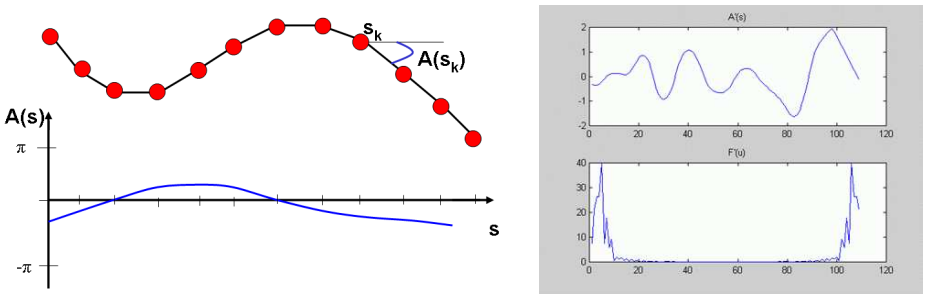
$$|R_k(i)| = C_{min}(i) + r_k(i) \cdot (C_{max}(i) - C_{min}(i)) \quad (1)$$

$$\angle R_k(i) = 2\pi \cdot g_k(i) \quad (2)$$

The edges of the motion graph are defined and updated during the motion synthesis algorithm on Trajectory Computation procedure. Therefore, the construction of the motion graph resembles the generation of a probabilistic roadmap [13], [12]. Let S_A be the current snake skeleton

[1] Actually, $\hat{F}(u)$ is the Fourier transform of a signal $\hat{A}(s_k)$ produced by appending samples to the end of $A(s_k)$ so that no discontinuities appear after repetition of $A(s_k)$. These extra samples may be computed by interpolation of the first and last terms of the initial $A(s_k)$.

[2] The skeleton states are produced by the skeleton points that are tracked with pixel accuracy in the whole image sequence.



(a) (b)

Fig. 6 **(a)** Equally spaced skeleton points and the resulting $A(s_k)$ function computed at these points. $A(s_k)$ is the angle between the horizontal axis and the skeleton at point s_k. **(b)** The functions $A(s)$ (in rads) and its Fourier transform $\hat{F}(u)$.

(skeleton state, snake orientation and snake position). The meaning of connection between a skeleton S_B and S_A is that the S_B is placed (rotated and translated) consecutively on S_A so that the head of S_A will be the tail of S_B (see Fig. 8(b)).

2.6 Motion Synthesis

As input for the motion synthesis algorithm are provided points of the trajectory in the 3–D virtual environment. The path planner creates a safe and short path in the 3–D virtual environment that passes from the user defined points. The path planner output is a set of L points (T_n, $n = 1, \cdots, L$) that the snake has to follow. Each point is admitted as reached, when the distance D_n of the snake from the considered point is lower than a threshold (D_{max}). Then, the trajectory computation procedure is executed, computing an initial trajectory of snake motion and initializing the skeleton animation angles sequence, $\Theta(n,t)$. The pseudo-code of the Motion Synthesis algorithm is given hereafter.

```
n = 1
pathPlanner()
trajectoryComputation()
repeat
    if D_n < D_max then
        n = n + 1
    end
    getVelocity()
    getSkeletonPoints()
until n < N
```

Algorithm 2: Motion Synthesis

The motion synthesis loop is split into steps: snake's velocity determination, skeleton angles computation and skeleton points computation. The snake velocity is defined applying a physical model. Next, the snake's shape is computed. Finally, the snake motion is being adapted to the local ground model.

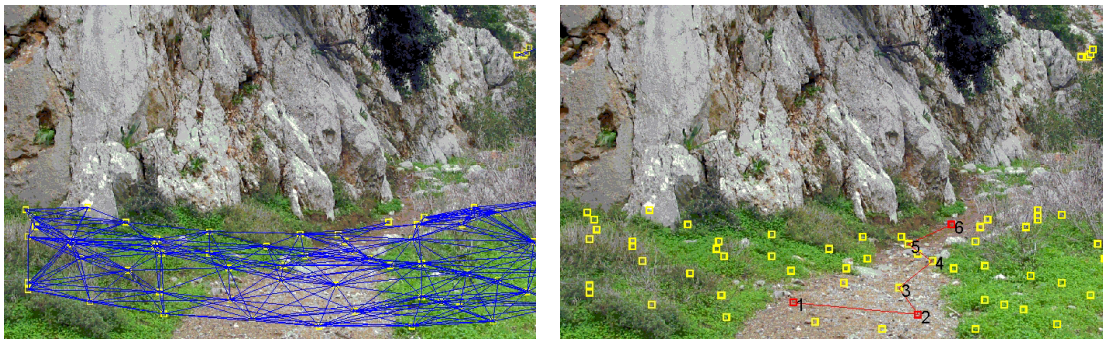
2.6.1 Path Planning in 3–D Virtual Environment

During this stage, the path of the outer environment, that will be traced by the animal, is computed. We assume that a assume that a 3–D model of the outer environment is provided as input. We also assume that the outer environment may contain obstacles, like rocks, trees, etc., which are overlaid on a mostly horizontal surface. Let $W = \{K_1, \cdots, K_n\}$ ($K_j \in \Re^3$) be the set of vertices of that 3–D model. We will denote by $X(K)$, $Y(K)$, $Z(K)$ the 3–D coordinates of any world point K. Based on the geometry of the world model, initially a subset S of W is automatically selected containing all those points having no obstacles around obstacles around them. A point will be a node of S, if the variance of the elevation (Y coordinate) around this point is lower than a predefined threshold.

The user then picks interactively a sequence of points $\{U_1, \cdots, U_l\}$ out of the set S and the path planning algorithm outputs a path, which passes through these points, but doesn't pass through any obstacles in between. For this purpose a weighted graph $G = (S, E)$ is computed having the points of the set S as nodes (Fig. 7(a)). The edges E are set so that there is a path between two nodes of G, only if there is a path between the corresponding points in the original world 3–D model. The weight of an edge is set equal to the variance of the elevation (Y coordinate) of all world 3–D points that appear along that edge. The sum of the edge weights along a path is called the path score and indicates the amount of obstacles that the path contains. The path ($\{T_1, T_2, \cdots, T_L\}$) with the minimum score is computed by applying Dijkstra's algorithm to the weighted graph G. A sample output of the algorithm is shown in Fig. 7(b). The red points numbered (1, 2, 6) were specified by the user, the final path points sequence is (1, 2, 3, 4, 5, 6).

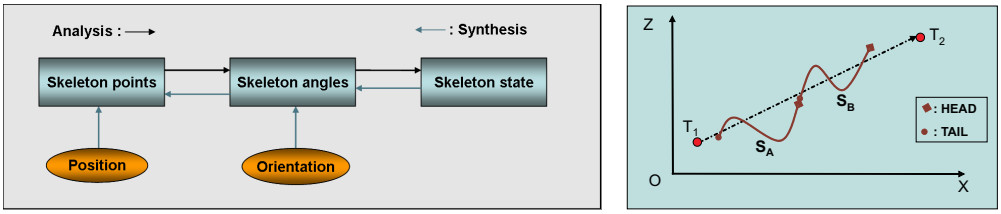
2.6.2 Trajectory Computation

In this procedure, the trajectory and the skeleton angle sequence of the animation are initialized using a mo-



(a) (b)

Fig. 7 The 3–D points graph and the final path in a virtual environment.



(a) (b)

Fig. 8 **(a)** The motion analysis and synthesis scheme. **(b)** The snake shape and orientation are defined by the snake target (T_2).

tion planning (exploration) algorithm on the state graph. This is a general algorithm, which can be applied to any motion graph obtaining aperiodic motion sequences. First, an arbitrary state (node) of the graph is selected and then the next states are chosen by traversing edges of the graph.

Using an inverse kinematic transform, the skeleton angles can be computed by the snake state and snake orientation. In each iteration (k), we compute the state skeleton angles using the state vector $S(k)$ of size $N = 7$ and the snake orientation $D(k)$ (see Fig. 8(a)). Let G be a vector with a size equal to the size of the initial transformed vector.

$$G = [c \cdot D(k)\ R^T(k)\ 0\ \cdots\ 0\ R^H(k)]^T$$
$$R^H = [\bar{R}(N)\ \bar{R}(N-1)\ \cdots\ \bar{R}(1)]$$

Then, the skeleton angles $\Theta_k(n)$ sequence is computed by the inverse Fourier transform of the vector G.

The goal of the algorithm is that the snake should trace the target points sequence T_i selecting unvisited states. The optimal snake orientation $D_o(k)$ can be computed by the direction of the vector $T_i - S_k$, where S_k denotes the snake mass center at iteration k and T_i denotes the current target point. So, the edges in graph are defined using the current snake target point. Let S_A, S_B be the current skeleton state and an arbitrary state, respectively. The S_B is placed (rotated and translated) consecutively on S_A (see Fig. 8(b)). The edge between S_A and S_B is created, if the orientation (after rotation) of S_B is close to the optimal orientation $D_o(k)$. Therefore, the graph edges are updated on each iteration of the procedure yielding a star graph, with the center in current state.

During the state selection process, which is executed until the snake passes all the target points, we use the following criteria so that non-periodic motions are obtained and the snake will visit most of the graph states, as soon as possible:

1. The graph will be covered as soon as possible without many cycles.
2. The steps between two visits ($Ts(q)$) of the same state q will at least $0.05 \cdot |V|$, where $|V|$ is the number of states.
3. Low computation cost per execution step ($O(|V|)$).

The first constraint will solve the Hamilton Path problem which is an NP-complete problem. So, we have to develop an approximate algorithm as the NP-complete problems have not been solved in $O(|V|)$.

In bibliography, there are two low computation cost algorithms that can be used: the random walk and the depth first search (DFS) [3]. We are going to compare them with the proposed method. Random walk selects the next state randomly from the unvisited states, otherwise if all neighboring states have been visited, it selects the next state randomly. The major problem of this algorithm is that it needs $O(|V|^2)$ in the mean case to cover the graph. An appropriate version of depth first search

(stack walk) selects from all unvisited states, that with the highest degree as the next state. If all neighboring states have been visited, it will select the previous state as the next state (it keeps in stack the visited states). The advantage of this algorithm is that it needs $O(|E|)$ steps in the worst case to visit the total states of a graph while the optimal is $|V|$, where $|E|$ is the number of graph edges. However, when the backtracking is performed the second constraint is not satisfied as the time between two visits of the same state is being low.

We propose the snake walk algorithm that satisfies the above constraints. For each state q we store the number of times $M(q)$ that this state has been selected and the number of times $F(q)$ that this state has been selected and the next state was an unvisited state. If $deg(q)$ denotes the degree of node q then the following statement is true:

$$deg(q) + F(q) \geq M(q) \geq F(q) \tag{3}$$

Among all neighboring states of the current state, the one that maximizes the function $W(q)$, defined hereafter, is chosen:

$$D_u(q) = 1 - sign(M(q)) \tag{4}$$
$$D_v(q) = deg(q) \cdot (sign(M(q)) + sign(F(q))) \tag{5}$$
$$D_f(q) = M(q) \cdot 2^{F(q)} \tag{6}$$
$$W(q) = \frac{deg(q)}{D_u(q) + D_v(q) + D_f(q)} \tag{7}$$

The properties of $W(q)$ describe the selection strategy of the algorithm.

- $W(q) \geq 1 \leftrightarrow$ the state q is unvisited[3].
- $1 > W(q) \geq \frac{1}{2} \leftrightarrow$ the state q has been visited and $F(q) = 0$[4].
- $W(q) < \frac{1}{2} \leftrightarrow F(q) > 0$[5].

Therefore, those states that have not been visited are preferred. If there does not exist unvisited states to visit in one step, those states that it is possible to connect to unvisited states are selected. The states q where $F(q) > 0$ are not preferred ($W(q) < \frac{1}{2}$) as they will certainly lead us to select another already visited state. If there are many unvisited states, the states with high degree are preferred, because these states could drive the algorithm to unvisited parts of the graph. If there are only visited states, the states q with $F(q) = 0$ and with high degree and low number of visits are preferred, since these states are more probable to drive the algorithm in the next step to an unvisited state and the number of steps that passed from the last selection of q is the maximum (second constraint) with the highest probability.

For many known graphs, like connected cliques with bridges, high density graphs (mean node degree ≥ 6), which are possible state graphs, it can be proved that the above algorithm needs $O(N)$ steps to cover the graph. In many cases of high density graphs approximates the optimal ($|V|$). The snake walk covers quickly any type of graph and as our experiments[6] show, in the worst case, it needs less than $NlogN$ steps.

As our experiments show, in high density graphs the snake walk always covers the graph faster than the stack walk, while, in most of the cases in the low density graphs, the stack walk is proved faster (Fig. 9). Concerning the second constraint, the percentage of the cases, in low density graphs, where $Ts(q) < 0.05 \cdot |V|$ is less than 10% and it is decreased to 0% in high density graphs. Using the stack walk, this percentage varies between 20% and 50% in the same graphs because of the backtracking phenomenon. So, under the problem constraints the snake walk yields better results than the stack walk and it is almost optimal in high density graphs.

2.6.3 Snake Velocity Computation

In this procedure, the snake velocity is computed applying a physical model. Let $P(n_0, t)$ be the position of the n_0 snake skeleton point at frame t in the horizontal XZ plane. The frame rate is chosen high enough, so that $P(n_0, t)$ can be estimated with high accuracy using just the 2–D snake velocity $\mathbf{v}(t)$. $P(n_0, t) = P(n_0, t-1) + \mathbf{v}(t)\Delta t$, where the Δt denotes the short time period between the two successive frames t, $t-1$. The point n_0 changes over the time. We have developed a simple snake model for the dynamics of snake locomotion. We assume that there are two types of forces interacting with the snake, a friction force $|T(t)|$ and a force produced by the snake motion $F(t)$. The friction model includes viscous friction and Coulomb friction, $|T(t)| = c_1|\mathbf{v}(t)| + c_2$.

The norm of the snake motion force $|F(t)|$ is given in Equation (8). The $K(t), \dot{O}(t)$ denote the standard deviation of the skeleton angles and the first time derivative of snake orientation, respectively. The snake force is related with the snake curves. Moreover, it has been observed that a real world snake usually decelerates, when it is turning. Equation (8) describes this behavior.

$$|F(t)| = c_3 K(t) - c_4|\dot{O}(t)|. \tag{8}$$

We now integrate the forces applying the second Newtonian law,

$$|\dot{\mathbf{v}}(t)| = |F(t)| - |T(t)| \tag{9}$$

Using the last equation, we can compute the norm of the acceleration vector. The speed norm can be computed using the assumption that the acceleration direction is parallel to the speed direction. This is a very good approximation, as the friction direction is exactly parallel

[3] $W(a) > W(b) \leftrightarrow deg(a) > deg(b)$.
[4] $W(a) > W(b) \leftrightarrow deg(a)M(b) > deg(b)M(a)$.
[5] $W(a) > W(b) \leftrightarrow deg(a)M(b)2^{F(b)} > deg(b)M(a)2^{F(a)}$

[6] The worst case is appeared on low density graphs where the graph diameter is high. We have tested the snake walk in more than 10.000 low density small worlds and random graphs [27] with $|V| \in \{100, 200, 300, \cdots, 1000\}$.

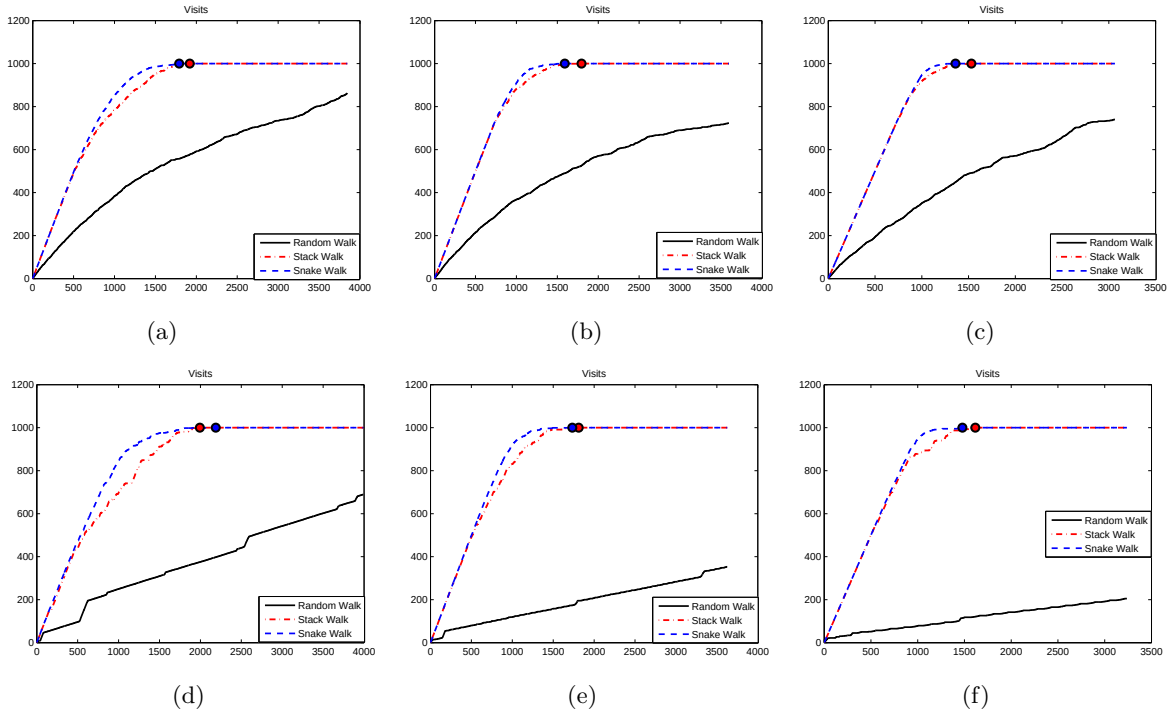


Fig. 9 The total number of distinct visited states per algorithm execution step at graphs of 1000 nodes. The red, blue cycles denote the time step in which the stack walk, snake walk cover the graph, respectively. **(a)** Random graph with mean degree of 3, **(b)** random graph with mean degree of 6, **(c)** random graph with mean node degree of 9, **(d)** small world graph with mean node degree of 3, **(e)** small world graph with mean node degree of 6, **(f)** small world graph with mean node degree of 9.

to the speed direction and the snake exerts the force F in the direction of the snake body at the n_0 skeleton point. This direction is the speed direction of the n_0 skeleton point, so we are able to compute the speed vector $\mathbf{v}(t)$.

The constants c_1, c_2, c_3, c_4 are the model parameters. The snake behavior is affected by these parameters. The parameters c_1, c_2 depend on the friction model. In our implementation we used $c_1 = 1$, $c_2 = 0.5$. Concerning the c_3, c_4, they are related to the snake mass and snake force. In our experiments, we used $c_3 = 6, c_4 < 2$ getting realistic results.

2.6.4 Skeleton Points Computation

The skeleton points at each frame t are computed using the snake velocity and the initial snake angle sequence estimated by the Trajectory Computation procedure. The snake motion is characterized by the fact that the tail motion always follows the head motion with a small phase delay. However, in a low friction surface, this rule does not hold. Using the results of the Motion Analysis Stage, we propose the following general mathematical model for the skeleton angles $\Theta(n,t)$. Let $\Theta_1(n)$, $\Theta_2(n)$, $\cdots$, $\Theta_M(n)$ be the skeleton angles estimated by the Trajectory Computation procedure. We initialize $\Theta(n-(k-1)\cdot N, 1) = \Theta_k(n), k \in \{1, 2, \cdots, M\}, n \in \{1, 2, \cdots, 100\}$. $\Theta(n,t), t > 1$ is given by the equation:

$$\hat{\Theta}(n, t-1) = \Theta(n - \frac{u(n,t)\cdot \Delta t}{r}, t-1) \quad (10)$$

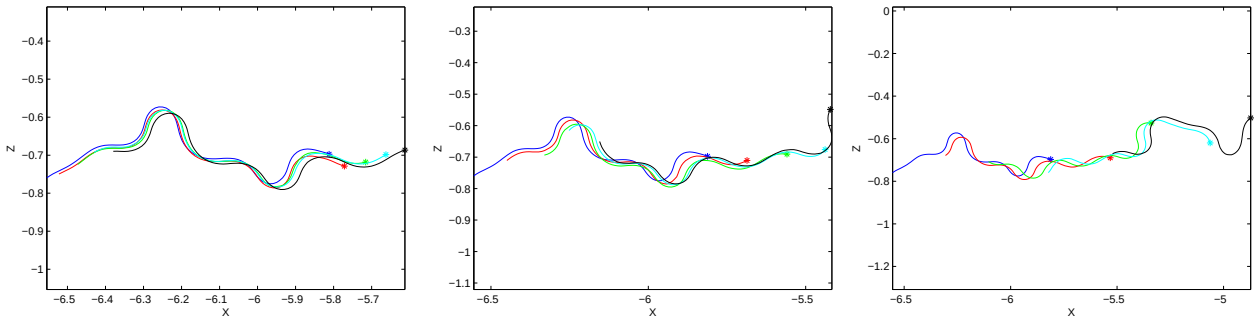
$$\bar{\Theta}(n, t-1) = \Theta_n(n - \frac{u(n,t)\cdot \Delta t}{r}, t-1) \quad (11)$$

$$\Theta(n,t) = \hat{\Theta}(n, t-1) + w_d(n,t)\cdot \bar{\Theta}(n, t-1) \quad (12)$$

$$u(n,t) = |\mathbf{v}(t)| + g(n,t) \quad (13)$$

Parameter r denotes the constant distance between two consecutive skeleton points. $\Theta_n(n,t)$ is the first numerical derivative of $\Theta(n,t)$ with respect to n. The weighting function $w_d(n,t)$ is defined for each skeleton point n. This function describes how the skeleton point n slides at frame t. If $w_d(n,t) = 0$ then the skeleton points will trace the snake head trajectory with phase delay (see Fig. 12(a)). This way the angles defined at the points of the head are gradually transferred to the points of the tail. Using a non constant function as $g(n,t)$, the skeleton points pulse speed can vary over the skeleton. A useful parameter that controls this behavior is the variance of $g(:,t)$, $\sigma_g^2(t)$. As our motion analysis stage shows, it holds that $g(n,t) = c_i, n_i < n < n_{i+1}$ and $|g(n,t) - g(n-1,t)| < T$ and $|g(n,t) - g(n,t-1)| < T$.

As the snake speed vector $\mathbf{v}(t)$ is known, the skeleton points $P(n,t), t > 1$ can be computed recursively from the head to the tail using the following equation under



(a) (b) (c)

Fig. 10 Skeleton points as recovered by using angles $\Theta(n,t)$ with different sampling rate over time. The blue and black skeletons correspond to the first and last states respectively.

the hypothesis $P(-1,t)=0$,

$$y_1 = \sum_{i=1}^{t} \mathbf{v}(i)\cdot \Delta t$$
$$y_2 = [sin(\Theta(n,t)) \;\; cos(\Theta(n,t))]^T$$
$$P(n,t) = P(1,1) + y_1 + P(n-1,t) + r\cdot y_2.$$

In Fig. 10 are depicted skeleton samples from three snake motion sequences, that were extracted by our method. An example of a snake animation is illustrated in Fig. 11.

2.6.5 Snake Locomotion Adaptation to the 3–D Local Ground

The synthesis algorithm computes a prototype locomotion which is taking always place in the horizontal XZ plane. Finally, the Y coordinates of the snake skeleton are computed, so that the snake moves on any surface of the virtual 3–D environment. We assume however that the animal moves on a mostly horizontal surface[7] (with slope less than 25 degrees). An efficient way to solve the problem without many computations is to ignore the changing of X, Z and to compute just the Y coordinate of any skeleton point using just the 3–D world model. Otherwise, we have to recompute the X, Z coordinates of each skeleton point. This can be done iteratively starting from the end of tail skeleton point, whose the position is known. The skeleton point $P(n,t)$ can be computed from the skeleton point $P(n+1,t)$ using the following constraints:

- Its distance from the skeleton point $P(n+1,t)$ is r.
- The angle on XZ plane between skeleton point n and $n+1$ is $\Theta(n,t)$.
- The distance between the point $P(n,t)$ and the 3–D world model is $\frac{d_n^t}{2}$ (see section 2.2).

[7] In the proposed force model, we have ignored the gravity force, which is important for the motions at inclined planes with slope more than 25 degrees.

Let define the vertical hemicycle on XZ plane C_{n+1}, whose center is the skeleton point $P(n+1,t)$, its radius is r, and its direction is $\Theta(n,t)$. The skeleton point n can be computed by the section of C_{n+1}^t with the 3–D world model. Finally, we add the constants $\frac{d_n^t}{2}$ to the estimated Y coordinates, so that the skeleton point $P(n,t)$ distance from the ground will be $\frac{d_n^t}{2}$ and the 3–D snake model will be adopted to the local ground. Results of this adaptation are depicted in Fig. 15.

2.6.6 Snake Behavior Control

In many cases, a real world snake can slide on low friction surfaces. We can enable this behavior by using a non zero function $w_d(n,t)$. Moreover, the skeleton points pulse speed can vary, this behavior is controlled by the function $g(n,t)$. In Fig. 12 skeleton points sequences over the time are depicted, these sequences have been computed using different $w_d(n,t)$, $g(n,t)$ functions.

The snake's shape coefficients can efficiently affect the snake curvature (the number - type of waves appearing on snake skeleton) and the snake behavior. Each coefficient $C(i), i \in \{1,\cdots,7\}$ of a snake state corresponds to a frequency of the snake skeleton signal. The first from them correspond to low frequencies (low number of waves) and the last from them to high frequencies (high number of waves). We can affect the snake curvature on State Graph Construction module, as the State Graph consists of different snake states that the snake is going to cover. Therefore, we can set some thresholds on the coefficients' energy or we can allow many types of waves under a predefined proportion getting the appropriate snake curvature on the synthetic snake motion sequences. Fig. 13 illustrates skeletons derived by different rules on snake state coefficients of State Graph Construction module.

Concerning the snake tongue model, it is described below. The snake tongue is modeled by a cylinder whose height varies over time. The direction of the snake tongue

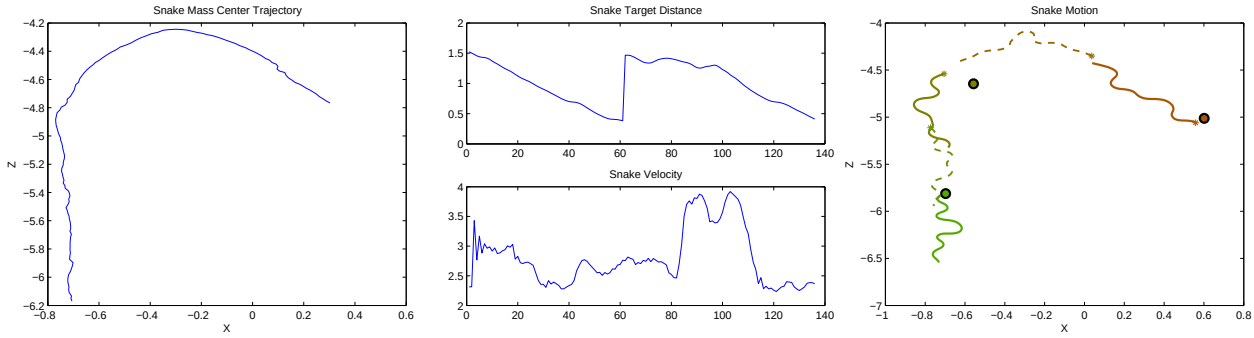


Fig. 11 **(a)** The snake mass center trajectory. **(b)** Statistics about snake motion over time: the distance of snake middle point from the current target and the snake velocity. **(c)** Sampling skeletons from a snake motion sequence and the three target points (green , brown and red cycle).

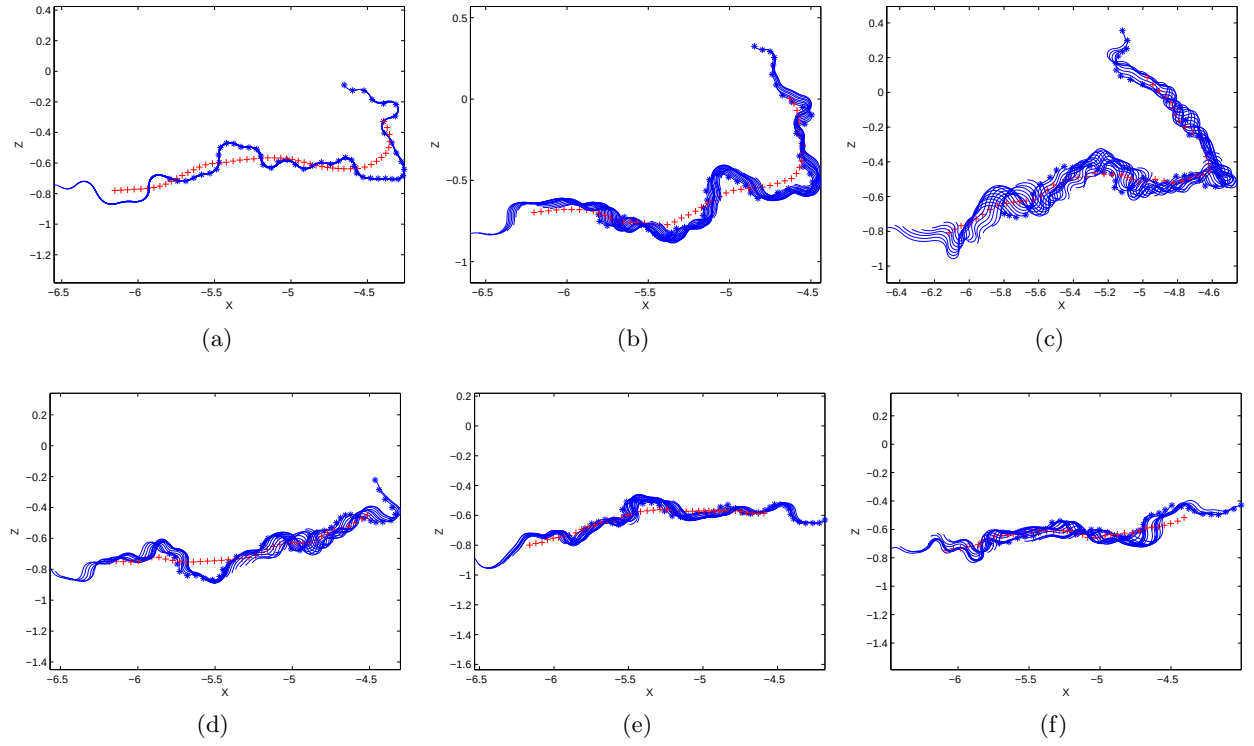
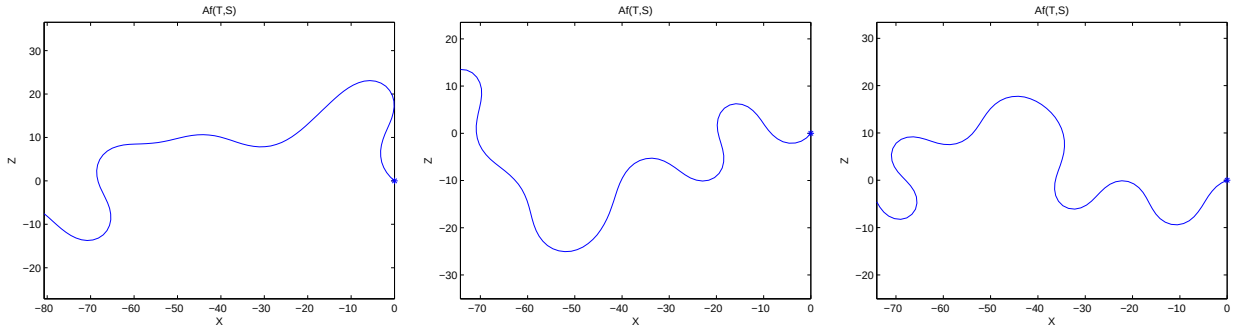


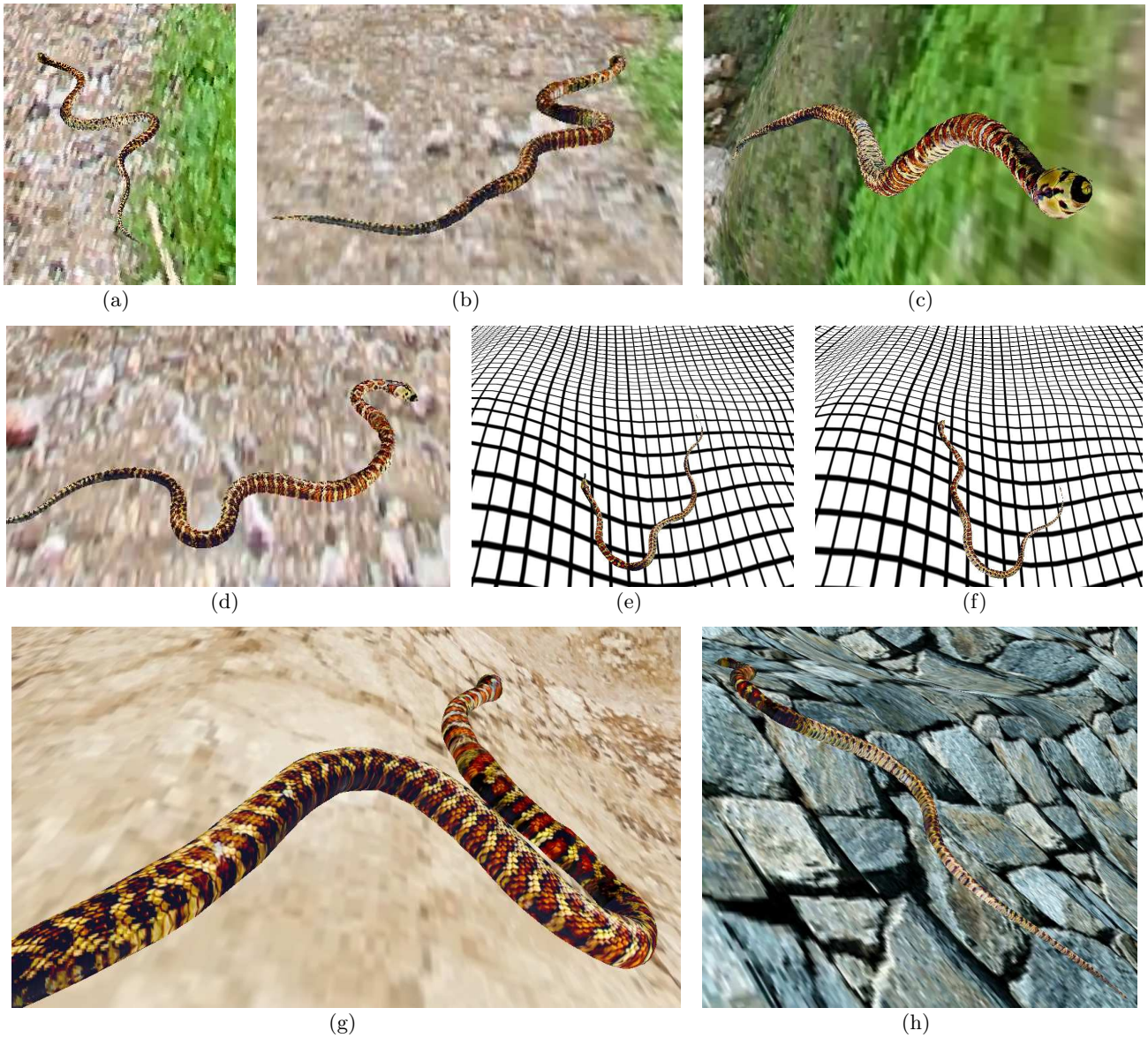
Fig. 12 Skeleton points sequences as recovered by using angles $\Theta(n,t)$. The *, + points correspond to the position of the head and mass center, respectively. **(a)** $w_d(n,t)=0$, $g(n,t)=0$ **(b)** $w_d(n,t)=0.5$, $g(n,t)=0$ **(c)** $w_d(n,t)=1$, $g(n,t)=0$ **(d)** $w_d(n,t)=cos^2(0.05\cdot t)$, $g(n,t)=0$ **(e)** $w_d(n,t)=0.3$, $V_g(t)=0.5$ **(f)** $w_d(n,t)=0.3$, $V_g(t)=2$.

is almost the same as the snake head. Let $H_{st}(t)$ be the height of the cylinder at time t. We used image tracking data in order to estimate the probability $P(H_{st}(t) = x/H_{st}(t-1)=y,\ H_{st}(t-2)=z)$. If this probability is known, it is trivial to compute a sequence of $H_{st}(t)$. The snake tongue behavior (how fast the the snake tongue is moved) can be adjusted by using the $H_{st}(b_{st}\cdot t)$ as the $H_{st}(t)$, where $b_{st}\in[0.4,2.5]$ is the adjustment parameter of the snake tongue behavior.



(a) (b) (c)

Fig. 13 **(a)** Low frequency skeleton, **(b)** medium frequency skeleton and **(c)** high frequency skeleton.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Fig. 15 Some frames from the produced snake animations inside 3–D virtual environments.

Fig. 14 The synthetic 3–D snake.

3 Results

The first step of the algorithm is the 3–D snake model construction. In Figs. 14 and 2(b) the 3–D snake model and the corresponding wire-frame are depicted.

Hundreds of synthetic animations have been produced by the proposed motion synthesis algorithm. The snake was placed inside 3–D virtual environments like a representation of Samaria Gorge and synthetic virtual environments. Some frames from the produced animations are displayed in Fig. 15. In Figs. 15(c), 15(f), 15(e) and 15(h) the 3–D shape of the snake body is adapted to the inclined local ground. The accurate texture mapping and 3–D model and 3–D model allow us to observe the snake skin details in high resolution (see Fig. 15(g)). Consequently, the snake motion algorithm passes from many states producing smooth and aperiodic motion, which are the major features of the real snake motion. Some videos of the snake animations produced by the proposed method are available at the Web address: *www.csd.uoc.gr/˜cpanag/DEMOS/snakeAnimation.htm.*

The method has been implemented using C and Matlab and the animations are generated in VRML 2.0 format. The mean computation time for the construction of a 3–D snake model was about 45 seconds, using as input images of 1.6 megapixels. This step needs to be executed only once and it has been implemented using C. Regarding the rest steps of the method, which have been implemented using Matlab, the mean computation time for producing a 250 frames animation was about 65 seconds. This time includes the time occupied by the path planning stage (inside a 3–D virtual world of 10.000 vertices), as well as the motion synthesis stage. The code, as it is written in Matlab, was not speed optimized. So, our method can be implemented in real time. For our experiments, we have been using a Pentium 4 CPU at 2.8 GHz.

4 Conclusion

In this paper, we have presented a method for 3–D snake model building and animation synthesizing using image sequences as input. The 3–D model is created by ellipses, whose centers are defined by the skeleton points. In the same step, the proposed method provides the high detailed texture mapping, using high resolution real snake images and prior knowledge about the snake anatomy, making the synthetic snake resemble more genuine. Thus, the precision of image analysis data suffices to give realistic modeling.

The snake motion data consist of tracking skeleton points in the whole image sequences. The tracking accuracy has been improved by using stable camera and high contrast background. This method is based on a solution of the curve equipartition problem, which has been also presented in this article. The proposed snake model coefficients can be distinguished to those that describe the rotation, translation, scaling of the snake and to those that describe the snake's shape, leading on the development of an efficient motion synthesis method. The minimal number of the proposed snake model coefficients decreases the memory cost of the state graph construction, providing the ability of a large graph construction which means a huge number of different skeletons that the snake is going to cover. Moreover, the noise of image data and the number of 3–D model parameters are reduced by using the Fourier transform coefficients of the angles sequence between consecutive skeleton points. To move snake from one user predefined point to another a graph of surface vertices is used. Edges of this graph have different weights concerning the safety of the edge path. A safe and smooth trajectory of the snake is calculated as a trajectory along this graph edges and it should be of minimal weight.

Concerning the snake locomotion, an animation synthesis algorithm, based on a physical motion model, describes the snake's velocity. To overcome the problem that the input video sequence contains a limited number of motions and skeleton shapes, the proposed motion synthesis algorithm, based on a set of possible states, can generate new unseen motions, allows a large number of skeleton shapes, and ensures aperiodic motion sequences. The combination of them yields special characteristics on snake locomotion behavior, like the natural locomotion, that is locomotion derived following physical laws, and the unpredictable - aperiodic motion, yielding impressive animations. The motion planning algorithm yields better results than the existed methods, random walk and stack walk and it can be applied to any motion graph. Finally, the snake locomotion is adapted to the 3–D local ground yielding realistic results on fitting the snake skin into the 3–D local ground surface. Concerning the snake behavior, the snake curvature over a synthetic motion sequence can be efficiently affected by the snake's shape coefficients on the State Graph Construction module. Moreover, the "sliding" snake behavior can be easily controlled by a model parameters.

As future work, we plan to extend our forces model between the snake body and the ground including the gravity force, so that a more accurate representation of the snakes locomotion is obtained. Moreover, we can ap-

ply the 3–D model construction method and the path planning algorithm to other creatures, characterized by skeleton based 3–D model and no periodic motion. The motion planning algorithm can be also applied to robotic systems that should cover as soon as possible a state graph (e.g. a region) without many cycles.

Acknowledgments

This research was partially supported by the European IST Digital Artistic and Ecological Heritage Exchange (DHX) project and by the Greek PENED 2003 program. The authors would like to thank the researchers of Natural History Museum of Crete (NHMC), Dr. Apostolis Trichas and Dr. Petros Lymberakis for the fruitful discussions on animation and help on video data collection.

References

1. I. Albrecht, J. Haber, and H. Seidel. Construction and animation of anatomically based human hands models. In *SIGGRAPH*, 2003.
2. A. Barbier, E. Galin, and S. Akkouche. A framework for modeling, animating, and morphing textured implicit models. *Graphical Models*, 67(6):166–188, 2005.
3. M. A. Batalin and G. S. Sukhatme. Efficient exploration without localization. In *Int. Conference on Robotics and Automation*, 2003.
4. F.L. Chernousko. Modelling of snake-like locomotion. *Applied Mathematics and Computation*, pages 415–434, 2005.
5. M. Dontchena, G. Yngve, and Z. Popovic. Layered acting for character animation. In *Proc. of the IEEE Int. Conf. on Computer Vision*, pages 409–416, 2003.
6. L. Favreau, L. Reveret, C. Depraz, and M. Cani. Animal gaits from video. In *SIGGRAPH*, 2004.
7. Pascal Glardon, Ronan Boulic, and Daniel Thalmann. Dynamic obstacle avoidance for real-time character animation. *The Visual Computer*, to appear, 2006.
8. J. Gray. Animal locomotion. *Norton, London, 1968.*
9. A.J. Ijspeert. *Design of artificial neural oscillatory circuits for the control of lamprey- and salamander-like locomotion using evolutionary algorithms.* PhD thesis, Univ. of Edinburgh, 1998.
10. Bruce C. Jayne. Kinematics of terrestrial snake locomotion. *Copeia*, pages 915–927, 1986.
11. M.P. Johnson. *Exploiting Quaternions to Support Expressive Interactive Character Motion.* PhD thesis, Massachusetts Institute of Technology, 2003.
12. L. Kavraki, S. LaValle, and J. Yakey. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1999.
13. L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
14. Lucas Kovar, Michael Gleicher, and Fred Pighin. Motion graphs. In *SIGGRAPH*, 2002.
15. G. Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–178, 1988.
16. Brad R. Moon and Carl Gans. Kinematics, muscular activity and propulsion in gopher snakes. *The Journal of Experimental Biology*, 201:2669–2684, 1998.
17. J. Nougaret, B. Arnaldi, and F. Multon. Coarse-to-fine design of feedback controllers for dynamic locomotion. *The Visual Computer*, 13:435–455, 1997.
18. C. Panagiotakis, G. Georgakopoulos, and G. Tziritas. The curve equipartition problem. *submitted to Computational Geometry*, 2005.
19. C. Panagiotakis, G. Georgakopoulos, and G. Tziritas. On the curve equipartition problem: a brief exposition of basic issues. In *European Workshop on Computational Geometry*, 2006.
20. C. Panagiotakis and G. Tziritas. Construction of animal models and motion synthesis in 3d virtual environments using image sequences. In *Proc.of the 2nd Intern. Symp. on 3-D Data Processing Visualization and Transmission*, 2004.
21. C. Panagiotakis and G. Tziritas. Recognition and tracking of the members of a moving human body. In *Proc. of the 3rd Int. Workshop on Articulated Motion and Deformable Objects*, pages 86–98. Springer, 2004.
22. D. Ramanan and D.A. Forsyth. Using temporal coherence to build models of animals. In *Proc. of the IEEE Int. Conf. on Computer Vision*, 2003.
23. M. Saito, M. Fukaya, and T. Iwasaki. Modeling, analysis, and synthesis of serpentine locomotion with a multilink robotic snake. *IEEE Control Systems Magazine*, 22(1):64–81, 2002.
24. E. Sifakis, I. Grinias, and G. Tziritas. Video segmentation using fast marching and region growing algorithms. *EURASIP Journal on Applied Signal Processing*, pages 379–388, 2002.
25. C. Sminchisescu and B. Triggs. Building roadmaps of minima and transitions in visual models. *Int. Journal of Computer Vision, 2004.*
26. D.P. Tsakiris, M. Sfakiotakis, A. Menciassi, G. LaSpina, and P. Dario. Polychaete-like undulatory robotic locomotion. In *Proc. of the IEEE Intl. Conference on Robotics and Automation*, 2005.
27. D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, pages 340–442, 1998.
28. Jane Wilhelms and Allen Van Gelder. Combining vision and computer graphics for video motion capture. *The Visual Computer*, 19(6):360–376, 2003.
29. J. Wu and Z. Popovic. Realistic modeling of bird flight animations. In *SIGGRAPH*, 2003.
30. Yang, J. Laszlo, and K. Singh. Layered dynamic control for interactive character swimming. In *SIGGRAPH*, 2004.