Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Signal Processing: *Image Communication* 20 (2005) 869–890

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.com/locate/image

3D visual reconstruction of large scale natural sites and their fauna

Nikos Komodakis*, Costas Panagiotakis, George Tziritas

Computer Science Department, University of Crete, Greece

Received 30 September 2004; accepted 26 April 2005

Abstract

The European DHX project is introduced in this paper and the vision-based 3D content authoring tools that have been implemented as a core part of its framework are presented. These tools consist mainly of two interrelated components. The first one is a hybrid (geometry and image based) modeling and rendering system capable of providing photorealistic walkthroughs of real-world large landscapes based solely on a sparse set of stereoscopic views. The second one deals with obtaining lifelike animal motion using as input captured video sequences of animal movement and prior anatomy knowledge. Both of these components have been successfully applied to a DHX research prototype aiming to create an integrated virtualized representation of the Samaria Gorge in Crete and the principal animals living in the ecosystem.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Image based rendering; Morphable model; Wide-baseline optical flow; Animal animation; Motion synthesis; 3D mosaic; Interactive walkthrough

1. Introduction

A main objective of the IST programme is enhancing the user-friendliness of information technology especially with regard to accessibility and quality of public services. Museums and public sites represent important and well-established services that attract a large and general public. Studying and experimenting new ways to transfer knowledge to the museums' visitors by exploiting the most advanced information technologies in a networked context is a promising field with a number of unexplored applications. A new use of information technology is the integration of VR and communication technologies. VR allows the creation of new forms of digital contents and user interaction. Having reached a well mature stage and affordable costs, VR lends itself to set up reliable interactive systems for public presentations. However, developing rich digital contents for museums still clashes over a

*Corresponding author.
E-mail addresses: komod@csd.uoc.gr (N. Komodakis), cpanag@csd.uoc.gr (C. Panagiotakis), tziritas@csd.uoc.gr (G. Tziritas).

number of problems that reside on the lack of appropriate authoring tools and standardization. The main tasks that are needed to carry out this process are: generating 3D models of whichever heritage objects, gathering and storing knowledge descriptions of objects, designing presentation scenarios, and conceiving new interactive paradigms for groups of users. In addition, another advanced aspect is to be able to share and communicate heritage contents between museums worldwide. DHX (Digital Artistic and Ecological Heritage Exchange) is a project funded by the European Union that aims to establish a common environment for museums for content development and a networked virtual reality infrastructure to share immersive experiences. The main goals of the project are the following:

- providing a distributed IT infrastructure for globally shared immersive experience;
- improving the authoring tools for digital storytelling by computer vision methods;
- developing distributed heritage experiences as next generation of digital collections;
- accessing existing multi media knowledge bases and digital libraries for detailed information and education;
- presentation of human heritage to large scale networked audience for interactive exploration, edutainment and education.

An overview of the DHX infrastructure is presented in Fig. 1. DHX deals with the entire production pipeline from the content production side up to the actual presentation system. DHX intends to ease the production of content by providing new content creation tools as well as an authoring interface to make easier generating the "story" behind the content. Special issues concern artificial characters ("synthetic characters", and

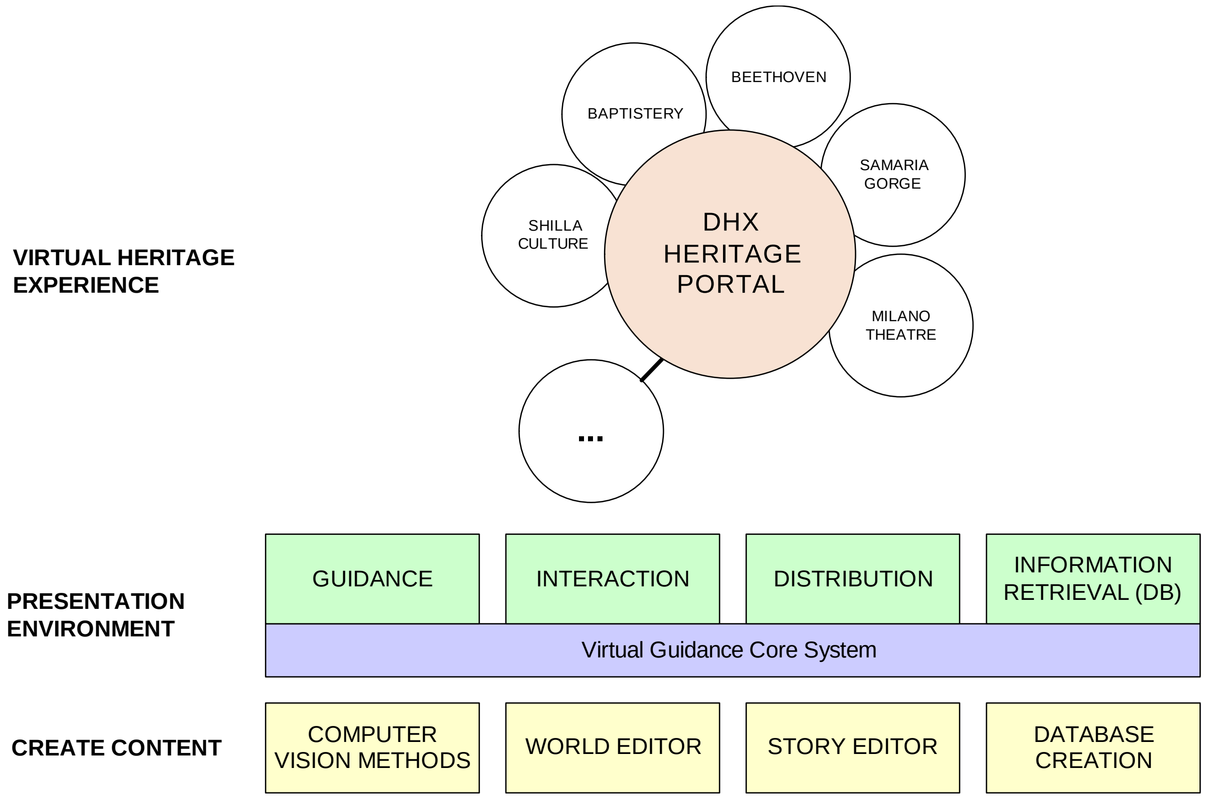


Fig. 1. An overview of the DHX architecture.

“avatars”) which are used for guiding visitors through the content environment, data base enquiring from VR systems, remote interaction, and how people can communicate with each other via audio and video channels (like videoconferencing). DHX presents different visualizations systems, from low budget till high-cost installations, to consider different budgets of end-users. To demonstrate the DHX infrastructure, the project consortium has developed a set of research prototypes (see top of Fig. 1) that reflect the cultural heritage of the participating countries: the discovery of the ancient Shilla culture in the South Korea, a virtual tour along the Samaria gorge in Crete, the immersion inside a typical 19th century Milan Drama Theatre, the interactive exploration of the life and works of Beethoven and interactive narrative paths inside the Baptistery of Pisa.

In this paper the focus is on the vision based authoring tools that have been implemented as a core part of the DHX framework. These mainly consist of two components. The first one is a hybrid (geometry and image based) 3D modeling and rendering system, called “morphable 3D-mosaics” [18], that is capable to reconstruct and visualize (in a photorealistic way) real-world large scale scenes. The input to the system is a sparse set of stereoscopic views captured along a predefined path. An intermediate representation of the environment is first constructed consisting of a set of local 3D-models. Then at rendering time, a continuous morphing (both photometric and geometric) takes place between successive local 3D models, using what we call a “morphable 3D-model”. Care is taken so that the morphing proceeds in a physically valid way.

The second component of the vision based tools deals with capturing, describing and visualizing the movement of animals. As a first step, the position of the main joints are obtained using image sequences and prior knowledge about the anatomy of the considered animal. Knowing also the approximate size of the members of the considered animal, and some predefined poses of it, the 3D movement of the animal is finally acquired. Since the main animal movements are characterized by periodicity, only one cycle of this movement is considered for the model description. This 3D model is then provided to the visualization system for the immersion in the virtualized scene.

The rest of the paper is organized as follows: we begin with a brief overview of the related work in Section 2. Our 3D modeling and rendering pipeline is described in Section 3 while the animal motion analysis and synthesis system is studied in Section 4.

2. Related work

One example of geometry-based modeling of real world scenes is the work of Pollefey et al. [24] on 3D reconstruction from hand-held cameras. Debevec et al. [7] propose a hybrid (geometry- and image-based) approach which makes use of view dependent texture mapping. However, their work is mostly suitable for architectural type scenes. Furthermore, they also assume that a basic geometric model of the whole scene can be recovered interactively. In [11], an image-based technique is proposed by which an end-user can create walkthroughs from a sequence of photographs while in “plenoptic modeling” [22] a warp operation is introduced that maps panoramic images (along with disparity) to any desired view. However, this operation is not very suitable for use in modern 3D graphics hardware. Lightfield [19] and Lumigraph [12] are two popular image-based rendering methods but they require a large number of input images and so they are mainly used for small scale scenes.

To address this issue work on unstructured/sparse lumigraphs has been proposed by various authors. One such example is the work of Buehler et al. [6]. However, in that work, a fixed geometric proxy (which is supposed to describe the global geometry of the scene at any time instance) is being assumed, an assumption that is not adequate for the case of 3D data coming from a sequence of sparse stereoscopic views. This is in contrast to our work where view-dependent geometry is being used due to the continuous geometric morphing that is taking place. Another example of a sparse lumigraph is the work of Schirmacher et al. [28]. Although they allow the use of multiple depth maps, any possible inconsistencies between them are not taken into account during rendering. This is again in contrast to our work where an optical flow between wide-baseline images is estimated to deal with this issue. Furthermore, this estimation of optical flow between wide baseline images reduces the required number of views. For these reasons if any of the above two approaches were to be applied to large-scale scenes like those handled in our case, many more images (than ours) would then be needed. Also, due to our rendering path which can be highly optimized in modern graphics hardware, we can achieve high frame rates (>25 fps) during rendering while the corresponding frame rates listed in [28] are much lower due to an expensive barycentric coordinate computation which increases the rendering time.

In [31] Vedula et al. make use of a geometric morphing procedure as well, but it is used for a different purpose which is the recovery of the continuous 3D motion of a non-rigid dynamic event (e.g. human motion). Their method (like some other methods [21,34]) uses multiple synchronized video streams combined with IBR techniques to render a dynamic scene, but all of these approaches are mostly suitable for scenes of smaller scale (than the ones we are interested in) since they assume that all of the cameras are static. Also, in the "interactive visual tours" approach [30], video (from multiple cameras) is being recorded as one moves along predefined paths inside a real world environment and then image based rendering techniques are used for replaying the tour and allowing the user to move along those paths. This way virtual walkthroughs of large scenes can be generated. Finally, in the "sea of images" approach [2], a set of omnidirectional images are captured for creating interactive walkthroughs of large, indoor environments. However, this set of images is very dense with the image spacing being ≈ 1.5 in.

The analysis of image sequences containing moving animals in order to create a 3D model of the animal and its physical motion is a difficult problem because of unpredicted and complicated (most of the time) animal motion. 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 (IK) [16]. Canned animations are more expressive, but less interactive, than numerical procedural techniques which often appear "robotic" and require parameter tweaking by hand. Motion capturing could be performed by means of image analysis. Ramanan and Forsyth [25] present a system that builds appearance models of animals based on image sequences. Many systems use direct motion capture technology [8].

Until now, a lot of work has been done in 3D animal modeling. A.J. Ijspeert [15] designed the 3D model of a salamander using neural controllers. Wu and Popovic [32] describe a physics-based method for the synthesis of bird flight animations. Their method computes a set of wing-beats that enables a bird to follow the specified trajectory. Favreau et al. [10] use real video sequences to extract an animal's 3D motion, but they require that the user provides the animal's 3D pose for a set of key frames, while for the rest of the frames interpolation is being used. In [13] Grochow et al. show that given a 2D image of a character they can reconstruct the most likely pose by making use of a probability distribution over all possible 3D poses. However, they require a database of training data to learn the parameters of this distribution. In [26] the estimation of a character's motion is cast as an optimization problem in a low-dimensional subspace, but one limitation of the method is that for defining this subspace a suitable set of motions must be selected by the user. Fang and Pollard [9] also use optimization as a way to generate new animations and, for this purpose, they consider certain physics constraints which are shown to lead to fast optimization algorithms. In [33], Yamane et al. make use of a motion capture database to synthesize animations of a human manipulating an object and they also provide a planning algorithm for generating the object's collision-free path. A different approach for generating animations is by using large motion capture databases along with direct copying and blending of poses [3], while some recent works try to model the muscle system of the character getting even more realistic results at the end [1].

One difference between our work and the approaches mentioned above is the fact that we are using video sequences in order to reconstruct both the animal's 3D model and its motion at the same time. Furthermore, our framework can handle both articulated and non-articulated motion equally well, while it is also capable of generating new unseen motions. In addition, the animal's motion is adapted so as to match the geometry of the surrounding 3D environment. The robustness of our system is ensured through the use of a small number of motion parameters, while due to its modularity it can be easily extended to new animals.

3. The "morphable 3D-mosaics" framework

One research problem of computer graphics that has attracted a lot of attention during the last years is the creation of modeling and rendering systems capable to provide photorealistic and interactive walkthroughs of complex, real-world environments. Two are the main approaches proposed so far in the computer vision literature for this purpose. On one hand (according to the purely geometric approach), a full 3D geometric model of the scene needs to be constructed. On the other hand, image-based rendering (IBR) methods skip the geometric modeling part and attempt to create novel views by appropriate resampling of a set of images.

While a lot of research has been done regarding small-scale scenes, there are only few examples of work dealing with large scale environments. The presented framework is such an example of a hybrid (geometry- and image-based) approach, capable of providing interactive walkthroughs of large-scale, complex outdoor environments. For this purpose, a new data representation of a 3D scene is proposed consisting of a set of morphable (both geometrically and photometrically) 3D models.

The main assumption is that during the walkthrough, the user motion takes place along a (relatively) smooth, predefined path of the environment. The input to our system is then a sparse set of stereoscopic views captured at certain points ("*key-positions*") along that path (one view per key-position). A series of *local 3D models* are then constructed, one for each stereoscopic view, capturing the photometric and geometric properties of the scene locally. These local models need to contain only an approximate representation of the scene geometry. During the transition between any two successive key-positions pos_1, pos_2 along the path (with corresponding local models L_1 and L_2), a "*morphable 3D-model*" L_{morph} is displayed by the rendering process. At point pos_1 this model coincides with L_1 and while we are approaching pos_2, it is gradually transformed (both photometrically and geometrically) into L_2 coinciding with the latter upon reaching key-position pos_2. The morphing proceeds in a physically-valid way and for this reason, a wide-baseline image matching technique is proposed handling instances, where the dominant differences in the images appearance are due to a looming of the camera. Therefore, during the rendering process a continuous morphing (each time between successive local 3D models) takes place.

Our system can be extended to handle the case of having multiple stereoscopic views per key-position, which are related by a pure rotation of the stereoscopic camera at that position of the path. In that case, a *3D-mosaic* per key-position needs to be constructed in addition. This 3D-mosaic comprises the multiple local models coming from the corresponding stereoscopic views at that position.

The main advantages of our approach are:

- No global 3D model of the environment needs to be assembled (a cumbersome process for large scale scenes).
- Scalability to large environments since at any time only one "morphable 3D-model" is displayed. In addition, we make use of a rendering path that is highly optimized in modern 3D graphics hardware.
- Being an image-based method, it can reproduce the photorealistic richness of a scene.
- Ease of data acquisition (e.g. collecting data for a path over 100 m long took us only about 20 min) as well as an almost automated processing of these data.

3.1. Overview of the modeling and rendering pipeline

We will first consider the simpler case of having one stereoscopic view per key-position of the path. Prior to capture, a calibration of the stereoscopic camera takes place [4]. The left and right camera calibration matrices (denoted hereafter by K_{left} & K_{right}) as well as the external geometry of the stereo-rig are estimated. Also lens distortion is removed. The main stages of the modeling are:

(1) Local 3D models construction (Section 3.2). A photometric and geometric representation of the scene near each key-position of the path is constructed. The geometric part of a local model must be just an approximation of the true scene geometry.
(2) Relative pose estimation between successive local 3D models (Section 3.3). Only a coarse estimate of the relative pose is needed, since this will not be used for an exact registration of the local models, but merely for the morphing procedure that takes place later.
(3) Estimation of morphing between successive local 3D models along the path (Section 3.4).

The rendering pipeline is examined in Section 3.5, while the extension of our system to the case of multiple views per key position is described in Section 3.6

3.2. Local 3D models construction

For each stereoscopic image pair, a 3D model describing the scene locally (i.e. as seen from the camera viewpoint) must be produced during this stage. To this end, a stereo matching procedure is applied to the left and right images (I_{left} and I_{right}), so that disparity is estimated for all points inside a selected image region dom_0 of I_{left} (we refer the reader to [27] for a review on stereo matching). A Gaussian as well as a median filter is further applied to the disparity map for smoothing and removing spurious disparity values, respectively. Using this disparity map (and the calibration matrices of the cameras) a 3D reconstruction takes place and thus the maps X_0, Y_0 and Z_0 are produced (see Fig. 2(a)), containing, respectively, x, y and z coordinates of the reconstructed points with respect to the 3D coordinate system of the left camera.

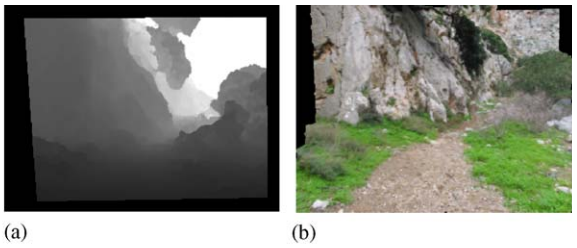


Fig. 2. (a) Depth map Z_0 of a local model (black pixels do not belong to its valid region dom_0); (b) a rendered view of the local model using an underlying triangle mesh.

The set $L_0 = (X_0, Y_0, Z_0, I_{\text{left}}, \text{dom}_0)$ consisting of the images X_0, Y_0, Z_0 (the geometric-maps), the image region dom_0 (valid domain of geometric-maps) and the image I_{left} (the photometric map) makes up what we call a "*local model*" L_0. Hereafter that term will implicitly refer to such a set of elements. By applying a 2D triangulation on the image grid of a local model, a textured 3D triangle mesh can be produced. The 3D coordinates of triangle vertices are obtained from the underlying geometric maps while texture is obtained from I_{left} and mapped onto the mesh (see Fig. 2(b)). It should be noted that the geometric maps of a local model are expected to contain only an approximation of the scene's corresponding geometric model.

3.3. Relative pose estimation between successive local models

Let $L_k = (X_k, Y_k, Z_k, I_k, \text{dom}_k)$ and $L_{k+1} = (X_{k+1}, Y_{k+1}, Z_{k+1}, I_{k+1}, \text{dom}_{k+1})$ be two successive local models along the path. For their relative pose estimation, we need to extract a set of point matches (p_i, q_i) between the left images I_k, I_{k+1} of models L_k, L_{k+1}, respectively. Assuming that such a set of matches already exists, then the pose estimation can proceed as follows: the 3D points of L_k corresponding to p_i are $P_i = (X_k(p_i), Y_k(p_i), Z_k(p_i))$ and so the reprojections of p_i on image I_{k+1} are: $p'_i = K_{\text{left}} \cdot (R \cdot P_i + T) \in \mathbb{P}^2$, where R (a 3×3 orthonormal matrix) and T (a 3D vector) represent the unknown rotation and translation, respectively.

So the pose estimation can be achieved by minimizing the following reprojection error: $\sum_i \mathrm{dist}(q_i, p'_i)^2$, where dist denotes euclidean image distance. For this purpose, an iterative constrained minimization algorithm may be applied with rotation represented internally by a quaternion q ($\|q\| = 1$). The *essential matrix* (also computable by the help of the matches (p_i, q_i) and $K_{\mathrm{left}}, K_{\mathrm{right}}$) can be used to provide an initial estimate [14] for the iterative algorithm.

Therefore the pose estimation problem is reduced to that of extracting a sparse set of correspondences between I_k, I_{k+1}. A usual method for tackling the latter problem is the following: first a set of interest-points in I_k are extracted (using an interest-point detector). Then for each interest-point, say p, a set of candidate points CAND_p inside a large rectangular region SEARCH_p of I_{k+1} are examined and the best one is selected according to a similarity measure. Usually the candidate points are extracted by applying an interest-point detector to region SEARCH_p as well.

However unlike left/right images of a stereoscopic view, I_k and I_{k+1} are separated by a wide baseline. Simple measures like correlation have been proved extremely inefficient in such cases. Assuming a smooth predefined path (and therefore a smooth change in orientation between I_k, I_{k+1}), it is safe to assume that the main difference at an object's appearance in images I_k and I_{k+1}, comes from the forward camera motion along the Z axis (looming). The idea for extracting valid correspondences is then based on the following observation: the dominant effect of an object being closer to the camera in image I_{k+1} is that its image region in I_{k+1} appears scaled by a certain scale factor $s > 1$. That is, if $p \in I_k$, $q \in I_{k+1}$ are corresponding pixels: $I_{k+1}(sq) \approx I_k(p)$. So an image patch of I_k at p should look similar to an image patch of an appropriately rescaled (by s^{-1}) version of I_{k+1}.

Of course, the scale factor s varies across the image. Therefore the following strategy, for extracting reliable matches, can be applied:

(1) Quantize the scale space of s to a discrete set of values $S = \{s_j\}_{j=0}^{n}$, where $1 = s_0 < s_1 < \cdots < s_n$.

(2) Rescale I_{k+1} by the inverse scale s_j^{-1} for all $s_j \in S$ to get rescaled images I_{k+1,s_j}.
For any $q \in I_{k+1}$, $p \in I_k$, let us denote by $\overline{I_{k+1,s_j}(q)}$ a (small) fixed-size patch around the projection of q on I_{k+1,s_j} and by $\overline{I_k(p)}$ an equal-size patch of I_k at p.

(3) Given any point $p \in I_k$ and its set of candidate points $\mathrm{CAND}_p = \{q_i\}$ in I_{k+1}, use correlation to find among the patches at any q_i and across any scale s_j, the one most similar to the patch of I_k at p:

$$(q', s') = \arg\max_{q_i, s_j} \mathrm{corr}(\overline{I_{k+1,s_j}(q_i)},\ \overline{I_k(p)}).$$

This way, apart from a matching point $q' \in I_{k+1}$, a scale estimate s' is provided for point p as well.

The above strategy has been proved very effective, giving a high percentage of exact matches even in cases with very large looming. Such an example can be seen in Fig. 3 wherein the images baseline is ≈ 15 m, resulting in scale factors of size ≈ 2.5 for certain image regions. Even if we set as candidate points CAND_p of a point p, all points inside SEARCH_p in the other image (and not only detected interest-points therein), the above procedure still picks the right matches in most cases. The results in Fig. 3 have been produced this way.

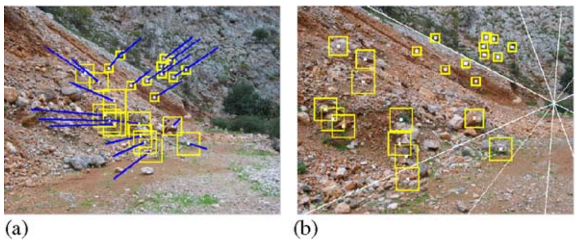


Fig. 3. (a) Image I_k along with computed optical flow vectors (blue segments) for all points marked white; (b) image I_{k+1} along with matching points (also marked white) for all marked points of (a). A few epipolar lines are also shown. In both images, the yellow square around a point is analogous to the point's estimated scale factor (10 scales $S = \{1, 0.9^{-1}, \ldots, 0.1^{-1}\}$ have been used).

3.4. Morphing estimation between successive local models along the path

At the current stage of the modeling pipeline, a series of approximate local 3D models (along with approximate estimates of the relative pose between every successive two) are available to us. Let $L_k = (X_k, Y_k, Z_k, I_k, \mathrm{dom}_k)$, $L_{k+1} = (X_{k+1}, Y_{k+1}, Z_{k+1}, I_{k+1}, \mathrm{dom}_{k+1})$ be such a pair of successive local models and $\mathrm{pos}_k, \mathrm{pos}_{k+1}$ their corresponding key-positions on the path. By making use of the approximate pose estimate between L_k and L_{k+1}, we will assume hereafter that the 3D vertices of both models are expressed in a common 3D coordinate system.

Rather than trying to create a consistent global model by combining all local ones (a rather tedious task requiring among others high quality geometry and pose estimation) we will instead follow a different approach which is based on the following observation: near path point pos_k, model L_k is ideal for representing the surrounding scene. On the other hand, as we move forward along the path approaching key-position of the next model L_{k+1}, the photometric and geometric properties of the environment are much better captured by the latter model. (For example compare the fine details of the rocks that are revealed in Fig. 3(b) and are not visible in Fig. 3(a)). So during transition from pos_k to pos_{k+1}, we will try to gradually morph model L_k into a new destination model, which should coincide with L_{k+1} upon reaching point pos_{k+1}. (In fact, only part of this destination model can coincide with L_{k+1} since in general L_k, L_{k+1} will not represent exactly the same part of the scene). This morphing should be geometric as well as photometric (the latter wherever possible) and should proceed in a physically valid way. For this reason, we will use what we call a "*morphable* 3D-*model*":

$$L_{\mathrm{morph}} = L_k \cup (X_{\mathrm{dst}}, Y_{\mathrm{dst}}, Z_{\mathrm{dst}}, I_{\mathrm{dst}}).$$

In addition to including the elements of L_k, L_{morph} also consists of maps $X_{\mathrm{dst}}, Y_{\mathrm{dst}}, Z_{\mathrm{dst}}$ and map I_{dst} containing, respectively, the destination 3D vertices and destination color values for all points of L_k. At any time during the rendering process, the 3D coordinates vert_{ij} and color col_{ij} of the vertex of L_{morph} at point (i,j) will then be:

$$\mathrm{vert}_{ij} = \begin{bmatrix} (1-m)X_k(i,j) + mX_{\mathrm{dst}}(i,j) \\ (1-m)Y_k(i,j) + mY_{\mathrm{dst}}(i,j) \\ (1-m)Z_k(i,j) + mZ_{\mathrm{dst}}(i,j) \end{bmatrix}, \qquad (1)$$

$$\mathrm{col}_{ij} = (1-m)I_k(i,j) + mI_{\mathrm{dst}}(i,j), \qquad (2)$$

where m is a parameter determining the amount of morphing ($m = 0$ at pos_k, $m = 1$ at pos_{k+1} and $0 < m < 1$ in between). Specifying therefore L_{morph} amounts to filling-in the values of the destination maps $\{X, Y, Z, I\}_{\mathrm{dst}}$ for each point $p \in \mathrm{dom}_k$.

For this purpose, a two-step procedure will be followed that depends on whether point p has a physically corresponding point in L_{k+1} or not:

(1) Let Ψ be that subset of region $\mathrm{dom}_k \subseteq I_k$, consisting only of those L_k points that have physically corresponding points in model L_{k+1} and let $u_{k\to k+1}$ be a function which maps these points to their counterparts in the I_{k+1} image. (Region Ψ represents that part of the scene which is common to both models L_k, L_{k+1}.) Since model L_k (after morphing) should coincide with L_{k+1}, it must then hold:

$$\begin{bmatrix} X_{\mathrm{dst}}(p) \\ Y_{\mathrm{dst}}(p) \\ Z_{\mathrm{dst}}(p) \\ I_{\mathrm{dst}}(p) \end{bmatrix} = \begin{bmatrix} X_{k+1}(u_{k\to k+1}(p)) \\ Y_{k+1}(u_{k\to k+1}(p)) \\ Z_{k+1}(u_{k\to k+1}(p)) \\ I_{k+1}(u_{k\to k+1}(p)) \end{bmatrix} \quad \forall p \in \Psi. \qquad (3)$$

Points of region Ψ are therefore transformed both photometrically and geometrically.

(2) The rest of the points (that is points in $\bar{\Psi} = \mathrm{dom}_k \backslash \Psi$) do not have counterparts in model L_{k+1}. So these points will retain their color value (from model L_k) at the destination maps and no photometric morphing will take place:

$$I_{\mathrm{dst}}(p) = I_k(p) \quad \forall p \in \bar{\Psi}. \qquad (4)$$

But we still need to apply geometric morphing to those points so that no distortion/discontinuity in the 3D structure is observed during transition from pos_k to pos_{k+1}. Therefore, we still need to fill-in the destination 3D coordinates for all points in $\bar{\Psi}$.

The two important remaining issues (which also constitute the core of the morphing procedure) are:

- How to compute the mapping $u_{k \to k+1}$. This is equivalent to estimating a 2D optical flow field between the left images I_k and I_{k+1}.
- And how to obtain the values of the destination geometric-maps at the points inside region $\bar{\Psi}$, needed for the geometric morphing therein.

Both of these issues will be the subject of the two subsections that follow.

3.4.1. Estimating optical flow between wide-baseline images I_k and I_{k+1}

In general, obtaining a reliable, relatively-dense optical flow field between wide-baseline images like I_k and I_{k+1} is a particularly difficult problem. Without additional input, usually only a sparse set of optical flow vectors can be obtained in the best case. The basic problems being that:

- For every point in I_k, a large region of I_{k+1} image has to be searched for obtaining a corresponding point. This way the chance of an erroneous optical flow vector increases significantly (as well as the computational cost).
- Simple measures (like correlation) are very inefficient for comparing pixel blocks between wide-baseline images.
- Even if both of the above problems are solved, optical flow estimation is inherently an ill-posed problem and additional assumptions are needed.

For dealing with the first problem, we will make use of the underlying geometric maps X_k, Y_k, Z_k of model L_k as well as the relative pose between I_k and I_{k+1}. By using these quantities, we can theoretically reproject any point, say p, of I_k onto image I_{k+1}. In practice since all of the above quantities are estimated only approximately, this permits us just to restrict the searching over a smaller region R_p around the reprojection point. The search region can be restricted further by taking the intersection of R_p with a small zone around the epipolar line corresponding to p. In addition, since we are interested in searching only for points of I_{k+1} that belong to dom_{k+1} (this is where L_{k+1} is defined), the final search region SEARCH_p of p will be $R_p \cap \mathrm{dom}_{k+1}$. If SEARCH_p is empty, then no optical flow vector will be estimated and point p will be considered as not belonging to region Ψ.

For dealing with the second problem, we will use a technique similar to the one described in Section 3.3 for getting a sparse set of correspondences. As already stated therein, the dominant effect due to a looming of the camera is that pixel neighborhoods in image I_{k+1} are scaled by a factor varying across the image. The solution proposed therein (and which we will also follow here) was to compare I_k image patches with patches not only from I_{k+1} but also from rescaled versions of the latter image. We will again use a discrete set of scale factors $S = \{1 = s_0 < s_1 < \cdots < s_n\}$ and $I_{k+1,s}$ will denote I_{k+1} rescaled by s^{-1} with $s \in S$. Also, for any $q \in I_{k+1}$, $p \in I_k$ we will denote by $\overline{I_{k+1,s}(q)}$ a (small) fixed-size patch around the projection of q in $I_{k+1,s}$ and by $\overline{I_k(p)}$ an equal size patch of I_k at p.

Finally, to deal with the ill-posed character of the problem, we will first reduce the optical flow estimation to a discrete labeling problem and then formulate it in terms of energy minimization of a first-order Markov random field [20]. The labels will consist of vectors $l = (d_x, d_y, s) \in \mathbb{R}^2 \times S$, where the first two coordinates denote the components of the optical flow vector while the third one denotes the scale factor. This means that after labeling, not only an optical flow but also a scale estimation will be provided for each point (see Fig. 4(a)). Given a label l, we will denote its optical flow vector by $\mathrm{flow}_l = (d_x, d_y)$ and its scale by $\mathrm{scale}_l = s$.

For each point p in I_k, its possible labels will be: $\mathrm{LABELS}_p = \{q - p : q \in \mathrm{SEARCH}_p\} \times S$. That is we are searching for points only inside region SEARCH_p but across all scales in S. Getting an optical flow field is then equivalent to picking one element from the cartesian product $\mathrm{LABELS} = \prod_{p \in \Psi} \mathrm{LABELS}_p$. In our case, that element f of LABELS which minimizes the following energy will be chosen:

$$E(f) = \sum_{(p,p') \in \aleph} V_{p,p'}(f_p, f_{p'}) + \sum_{p \in \Psi} D(f_p).$$

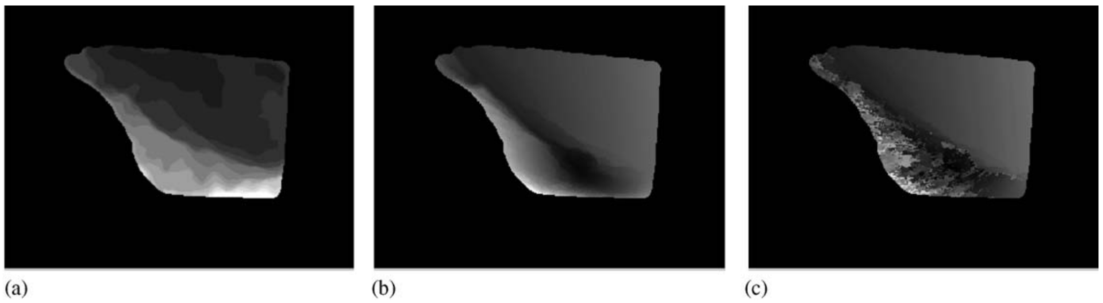


Fig. 4. Maps[1] of: (a) *scale factors*; (b) *optical flow magnitudes* for all points in Ψ, as estimated after applying the optical flow algorithm to the images of Fig. 3 and while using 10 possible scales $S = \{1, 0.9^{-1}, \ldots, 0.1^{-1}\}$; (c) corresponding optical flow magnitudes when only one scale $S = \{1\}$ has been used. As expected, in this case the algorithm fails to produce exact optical flow for points that actually have larger scale factors.

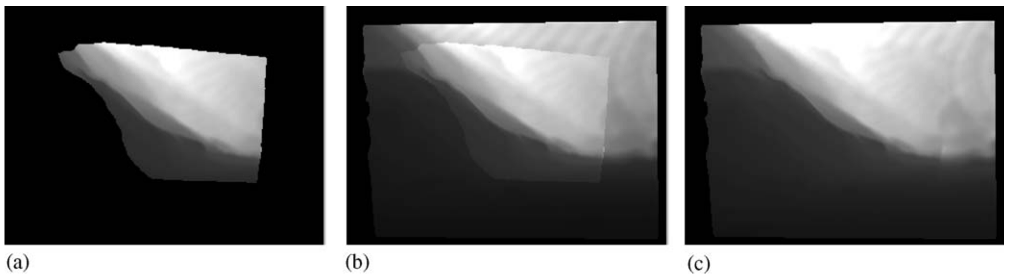


Fig. 5. (a) Destination depth map Z_{dst} for points in Ψ after using optical flow of Fig. 4(b) and applying Eq. (3). (b) Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ without applying geometric morphing. Observe the discontinuities along $\partial\Psi$. (c) Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ after applying geometric morphing.

The symbol $\aleph$ denotes a set of interacting pairs of pixels inside Ψ. The first term of $E(f)$ represents the sum of potentials of interacting pixels where each potential is given by the so-called Potts model (in which case the function $V(\mu, v)$ is equal to a non-zero constant if its arguments are not equal and 0 otherwise).

Regarding the terms $D(f_p)$, these measure the correlation between corresponding image patches as determined by labeling f. According to labeling f, for a point p in I_k, its corresponding point is the projection of $p + \text{flow}_{f_p}$ in image $I_{k+1,\ \text{scale}_{f_p}}$. So:

$$D(f_p) = \text{corr}\,(\overline{I_k(p)},\ \overline{I_{k+1,\ \text{scale}_{f_p}}(p + \text{flow}_{f_p})}).$$

Energy $E(f)$ can be minimized using either the iterated conditional modes (ICM) algorithm [20] or recently introduced algorithms based on graph cuts [5]. The results after applying the ICM method to the images of Fig. 3 appear in Fig. 4.

3.4.2. *Geometric morphing in region $\bar{\Psi}$*

After estimation of optical flow $u_{k\to k+1}$, we may apply Eq. (3) to all points in Ψ and thus fill-in the X_{dst}, Y_{dst}, Z_{dst} arrays therein (see Fig. 5(a)). To completely specify morphing, we still need to fill-in the values at the rest of the points, that is at points in $\bar{\Psi} = \text{dom}_k \backslash \Psi$. In other words, we need to specify the destination vertices for all points of L_k in $\bar{\Psi}$. Since these points do not have a physically corresponding point in L_{k+1}, we cannot apply (3) to get a destination 3D coordinate from model L_{k+1}. The simplest solution would be that no geometric morphing is applied to these points and that their destination vertices just coincide with

[1]Darker pixels (of a grayscale image) correspond to smaller values.

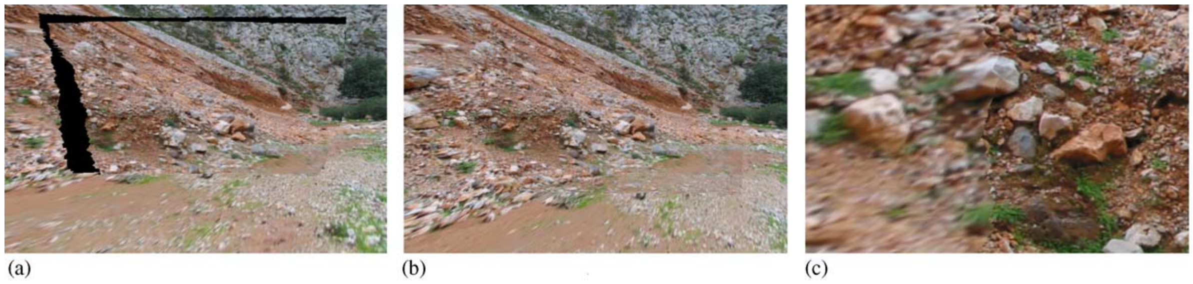


Fig. 6. Rendered views of the morphable 3D-model during transition from the key-position corresponding to image 3(a) to the key-position of image 3(b); (a) when no geometric morphing is applied to points in $\bar{\Psi}$; (b) when geometric morphing is applied to points in $\bar{\Psi}$; (c) a close-up view of the rendered image in (b). Although there is no geometric discontinuity, there is a difference in texture resolution between the left part of the image (points in $\bar{\Psi}$) and the right part (points in Ψ) because only points of the latter part are morphed photometrically.

their L_k vertices. However, in that case:

- points in Ψ will have destination vertices from L_{k+1};
- while points in $\bar{\Psi}$ will have destination vertices from L_k.

The problem resulting out of this situation is that the produced destination maps $X_{\text{dst}}, Y_{\text{dst}}, Z_{\text{dst}}$ (see Figs. 5(b) and 6(a)) will contain discontinuities along the boundary (say $\partial\Psi$) between regions Ψ and $\bar{\Psi}$, causing this way annoying discontinuity artifacts (holes) in the geometry of the "morphable 3D-model" during the morphing procedure. This will happen because the geometries of L_k and L_{k+1} (as well as their relative pose) have been estimated only approximately and will not therefore match perfectly.

The right way to fill-in the destination vertices at the points in $\bar{\Psi}$ is based on the observation that a physically valid destination 3D model should satisfy the following two conditions:

(1) On the boundary of $\bar{\Psi}$, no discontinuity in 3D structure should exist.
(2) In the interior of $\bar{\Psi}$, the relative 3D structure of the initial L_k model should be preserved.

Intuitively this means that as a result of morphing, vertices of L_k inside $\bar{\Psi}$ must be deformed (without distorting their relative 3D structure) so as to seamlessly match the 3D vertices of L_{k+1} along the boundary of $\bar{\Psi}$. In mathematical terms, preserving the relative 3D structure of L_k implies:

$$\begin{bmatrix} X_{\text{dst}}(p) - X_{\text{dst}}(p') \\ Y_{\text{dst}}(p) - Y_{\text{dst}}(p') \\ Z_{\text{dst}}(p) - Z_{\text{dst}}(p') \end{bmatrix} = \begin{bmatrix} X_k(p) - X_k(p') \\ Y_k(p) - Y_k(p') \\ Z_k(p) - Z_k(p') \end{bmatrix} \quad \forall p, p' \in \bar{\Psi},$$

which is easily seen to be equivalent to

$$[\nabla X_{\text{dst}}(p) \quad \nabla Y_{\text{dst}}(p) \quad \nabla Z_{\text{dst}}(p)] = [\nabla X_k(p) \quad \nabla Y_k(p) \quad \nabla Z_k(p)] \quad \forall p \in \bar{\Psi}.$$

We may then extract the destination vertices by solving three independent minimization problems (one for each of X_{dst}, Y_{dst}, Z_{dst}), all of the same type. It is therefore enough to consider only the Z_{dst} case. Z_{dst} is then the solution to the following optimization problem:

$$\min_{Z_{\text{dst}}} \int\int_{\bar{\Psi}} \|\nabla Z_{\text{dst}} - \nabla Z_k\|^2, \quad \text{given } Z_{\text{dst}}|_{\partial\bar{\Psi}}. \tag{5}$$

The finite-difference discretization of (5) (using the underlying pixel grid) yields a quadratic optimization problem which can be reduced to a sparse (banded) system of linear equations, solved easily.

An alternative solution to the problem can be given by observing that a function minimizing (5) is also a solution to the following Poisson equation with Dirichlet boundary conditions [35]:

$$\triangle Z_{\text{dst}} = \text{div}(\nabla Z_k), \quad \text{given } Z_{\text{dst}}|_{\partial\bar{\Psi}}. \tag{6}$$

Therefore in this case the solution can be given by solving three independent Poisson equations of the above type. See Figs. 5(c) and 6(b) for a result produced with this method.

3.5. Rendering pipeline

At any time, only one "morphable 3D-model" L_{morph} needs to be displayed. Therefore, during the rendering process just the geometric and photometric morphing of L_k (as described in Section 3.4) needs to take place .

For this purpose, two 3D triangle meshes tri_{Ψ}, $\mathrm{tri}_{\bar{\Psi}}$ are first generated by computing 2D triangulations of regions Ψ, $\bar{\Psi}$ and by making use of the underlying geometric maps of L_k. Then, application of the geometric morphing (as defined by (1)) to the vertices of these meshes, amounts to exploiting simple "vertex-shader" capabilities [29] of modern graphics cards (along with the $\{X, Y, Z\}_{\mathrm{dst}}$ maps of course).

Regarding the photometric morphing of mesh tri_{Ψ}, "multitexturing" [29] can be utilized as a first step. In this case, both images I_k, I_{k+1} will be applied as textures to mesh tri_{Ψ} and each vertex of tri_{Ψ} will be assigned texture coordinates from a point $p \in I_k$ as well as its corresponding point $u_{k\to k+1}(p) \in I_{k+1}$ (see (3)). Then, a straightforward "pixel-shader" [29] can be used to implement photometric morphing (as defined by (2)) for mesh tri_{Ψ}. Regarding $\mathrm{tri}_{\bar{\Psi}}$, no actual photometric morphing takes place in there (see (4)) and so only image I_k needs to be mapped as a texture onto this surface.

Therefore, by use of pixel- and vertex-shaders, just two textured triangle meshes are given as input to the graphics pipeline at any time (a rendering path which is highly optimized in modern 3D graphics hardware).

3.6. Extending the modeling pipeline

Up to this point we have been assuming that during the image acquisition process, we have been capturing one stereoscopic image-pair per key-position along the path. We will now consider the case in which multiple stereoscopic views per key-position are captured and these stereoscopic views are related to each other by a pure rotation of the stereoscopic camera. This scenario is very useful in cases where we need to have an extended field of view (like in large VR screens). In this new case, multiple local 3D models per key-position will exist and they will be related to each other by a pure rotation in 3D space.

In order to reduce this case to the one already examined, it suffices that a single local model per key-position (called *3D-mosaic* hereafter) is constructed. This new local model, say $L_{\mathrm{mosaic}} = (X_{\mathrm{mosaic}}, Y_{\mathrm{mosaic}}, Z_{\mathrm{mosaic}}, I_{\mathrm{mosaic}}, \mathrm{dom}_{\mathrm{mosaic}})$, should replace all local models, say $L_i = (X_i,\ Y_i,\ Z_i,\ I_i, \mathrm{dom}_i)\ i \in \{1, \ldots, n\}$, at that position. Then at any time during the rendering process, a morphing between a successive pair of these new local models (3D-mosaics) needs to take place as before. For this reason, the term "*morphable 3D-mosaics*" is being used in this case.

Constructing L_{mosaic} amounts to filling its geometric and photometric maps. Intuitively, L_{mosaic} should correspond to a local model produced from a stereoscopic camera with a wider field of view placed at the same path position. An overview of the steps that needs to be taken for the construction of L_{mosaic} now follows (for more details see [18]):

- As a first step the rotation between local models needs to be estimated. This will help us in registering the models in 3D space and aligning their maps in a common image plane.
- Then a geometric rectification of each L_i must take place so that the resulting local models are geometrically consistent with each other. This is a necessary step since the geometry of each L_i has been estimated only approximately and thus contains errors.
- Eventually, the maps of the refined and consistent local models will be merged so that the final map of the 3D-mosaic is produced.

The most interesting problem that needs to be handled during the 3D-mosaic construction is that of making all models geometrically consistent so that seamless (without discontinuities) geometric-maps of L_{mosaic} are produced. For this reason an operator $\mathrm{RECTIFY}_{L_i}(L_j)$ needs to be defined which takes as input two local models, L_i, L_j, and modifies the geometric-maps only of the latter so that they are consistent with the geometric-maps of the former (the geometric maps of L_i do not change during RECTIFY). Then ensuring consistency between all models can be achieved by merely applying $\text{RECTIFY}_{L_i}(L_j)$ for all pairs L_i, L_j with $i<j$.

It can be proved [18] that the new rectified Z map of L_j, say Z_{rect}, as produced by $\text{RECTIFY}_{L_i}(L_j)$ (X_{rect}, Y_{rect} maps are treated analogously) is found by solving:

$$\min_Z \int\int_{\bar{\Psi}} \|\nabla Z - \nabla Z_j\|^2, Z|_{\partial\bar{\Psi}} = Z_i|_{\partial\bar{\Psi}}, \tag{7}$$

where $\Psi = \text{dom}_i \cap \text{dom}_j$ is the overlap region of the two models (we assume that the maps of the local models are aligned on a common image plane since the rotation between the models is already known) and $\bar{\Psi} = \text{dom}_j \backslash \Psi$. The above problem, like the one defined by (5), can be reduced to a Poisson differential equation, as explained in Section 3.4.2. See Fig. 7(d) for a result produced with the latter method.

3.7. Results

The "morphable 3D-mosaic" framework has been successfully applied to a DHX research prototype aiming to provide a virtual tour of the Samaria Gorge in Crete. A predefined path inside the gorge has been chosen that was over 100 meters long and 45 stereoscopic views have been acquired along that path. Specifically, 15 key-positions of the path have been selected and three stereoscopic views per key-position have been captured (covering this way approximately a 120° field of view). Using the reconstructed "morphable 3D-mosaics", a photorealistic walk-through of the Samaria Gorge has been obtained at interactive frame rates using a GeForce4 3D graphics card and a Pentium 4 CPU at 2.4 GHz. A sample from these results is available at the following URL: http://www.csd.uoc.gr/~tziritas/DEMOS/samaria.htm.

4. Animal motion analysis and synthesis

A method computing 3D animations of animals will be described in this section. Its main stages are the following: creation of a 3D animal model, motion synthesis, path planning in the outer environment and motion transformation. Initially, a 3D animal model is produced using segmented images created by background subtraction. As a part of the 3D model construction process a set of points at the joints and the skeleton of the animal, called joint and sketelon points, respectively, are defined. During the motion synthesis stage, these joint and sketelon points are tracked through time in the video sequence and a prototype animal animation is produced. For the articulated motion case, this prototype animation always takes place along a straight line. During the next stage, 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 resulting in a zigzagging type of motion. In order that all of the above issues are avoided, an obstacle avoidance algorithm is proposed in this paper yielding a

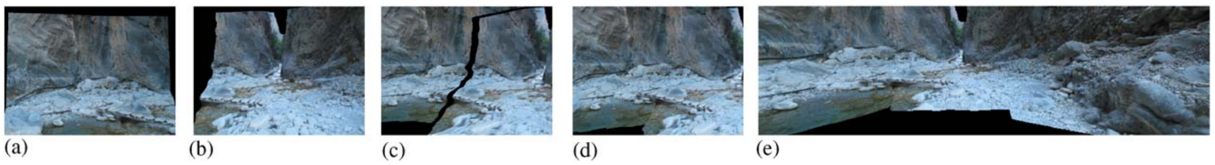


Fig. 7. Rendered views of: (a) a local model L_1; (b) a local model L_2; (c) 3D-mosaic of L_1, L_2 without geometric rectification (holes are due to discontinuities in geometry and not due to misregistration in 3D space); (d) 3D-mosaic of L_1, L_2 after $\text{RECTIFY}_{L_2}(L_1)$ has been applied; (e) a bigger 3D-mosaic created from local models L_1, L_2 (shown in (a), (b), respectively) plus another local model which is not shown due to space limitations. Geometric rectification has been used in this case too.

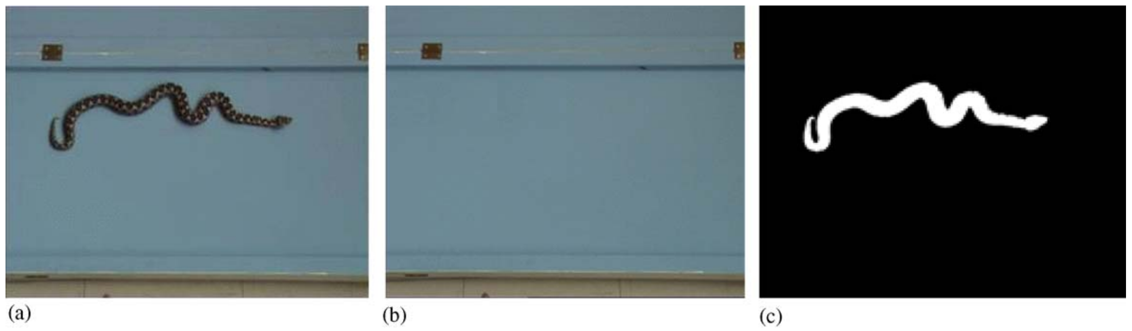


Fig. 8. (a) A snake image; (b) its background image; and (c) the extracted snake silhouette.

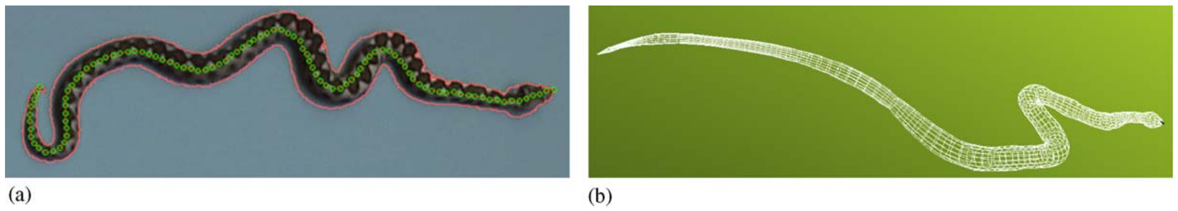


Fig. 9. (a) The skeleton points (green points) and the snake boundary (red curve); (b) a 3D wireframe model of the snake.

"safe" and smooth animal trajectory. Finally, since in the articulated motion case the prototype animal motion is always taking place along a straight path, this motion is transformed so that the animal is moving along the trajectory estimated during the path planning.

4.1. Creation of a 3D animal model

In this section the first step of our method, i.e. the construction of the animal 3D model, is described. Perfect 3D reconstruction, using only images as input, is a difficult task due to the complexity of the animal's shape. So instead, making use of a priori knowledge about the animal anatomy, its body is divided into parts whose vertical slices are ellipses. In our case, the snake model (Fig. 8) uses only one such part (Fig. 9(b)), the goat model consists of 11 such parts (main body, two horns, two ears, four legs, beard, tail) (Fig. 10) and the lizard is composed of five parts. This way the number of modeling parameters is minimized, thus increasing the robustness of the reconstruction.

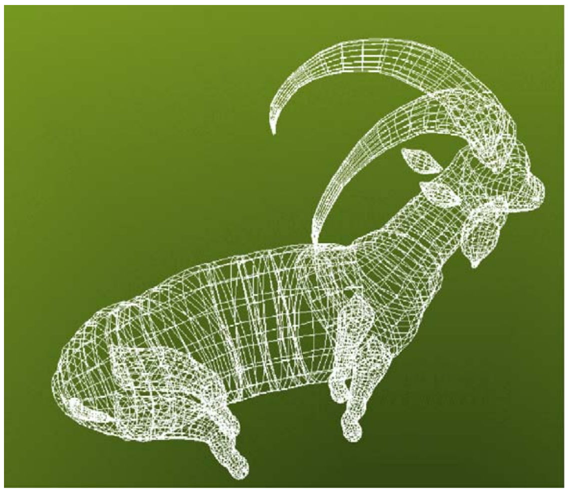


Fig. 10. A 3D wireframe model of the goat.

Each part is reconstructed separately using the following procedure, which is described only for the case of a snake. Two images (a background image and a top view image of the snake) are given as input to a background subtraction method for extracting the boundary of the snake body (Fig. 8). Equally spaced points from the medial axis of the boundary (skeleton points) are then chosen as the centers of these ellipses (Fig. 9(a)). The size of each ellipse is determined based on the distance between its center and the boundary. All of the produced ellipses make up the final 3D model of the considered animal part.

4.2. Motion synthesis

The motion synthesis algorithm is using as input the position of the skeleton points (non-articulated case) or the joints points (articulated case) during all the frames of the video sequence. The skeleton points are detected by repeatedly applying the method described in Section 4.1 to each frame of the video sequence. On the other hand, the joints points are tracked throughout the video sequence by placing markers on the body of the animal. As an output, the algorithm synthesizes new motions by computing the positions of either the skeleton or the joints' points through time.

4.2.1. Non articulated motion

First, we analyze the case of non-articulated motion (i.e. snake motion). Since the input video sequence contains a limited number of motions, it is important that the motion synthesis algorithm can generate new unseen motions, thus producing realistic and non-periodic snake animations. For this reason, no simple mathematical models have been used for the motion description but instead a novel technique has been implemented. According to this method, a graph is constructed whose nodes correspond to states of the snake skeleton and then motion synthesis merely amounts to traversing suitable paths through this graph.

First, it will be described how a skeleton state is represented. Since each frame contains a fixed and large number (≈ 100) of skeleton points, a more compact representation of the skeleton state will be used (utilizing fast fourier transform (FFT) coefficients). This representation will be related to the 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. 11(a)). Let $\hat{F}(u)$[2] be the FFT of the sequence $A(s_k)$ (Fig. 11(b)). Ignoring the first one of these FFT coefficients (which measures the global rotation of the snake's shape), the next seven FFT coefficients (say $C(i),\ i = 1, \dots, 7$) of $\hat{F}(u)$ are chosen to represent a state of the snake skeleton. Due to the smooth snake shape, these coefficients are enough to recover the angles $A(s_k)$ (and therefore the snake shape) suppressing noise at the same time. In addition, these are invariant to translation, rotation and scaling of the snake shape.

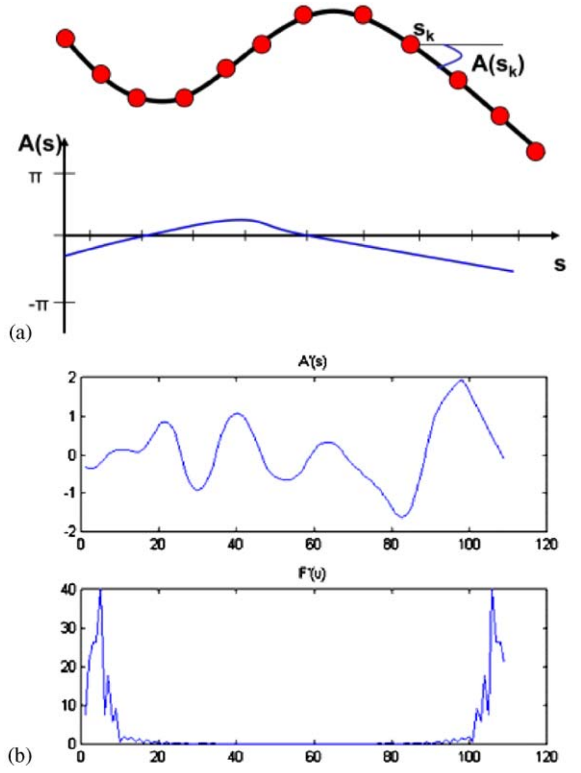


Fig. 11. (a) Equally spaced skeleton points and the resulting $A(s_k)$ function computed at these points. $A(s_k)$ is the angle between the horizontal axis and the skeleton at point s_k; (b) the functions $A(s)$ and its FFT $\hat{F}(u)$.

[2]Actually, $\hat{F}(u)$ is the FFT 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 term of the initial $A(s_k)$.

Having defined a representation of the skeleton state, the construction of the graph (that will be used for the motion synthesis) is now described. Each node of the graph corresponds to a skeleton state (i.e. a seven-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 following sampling algorithm.

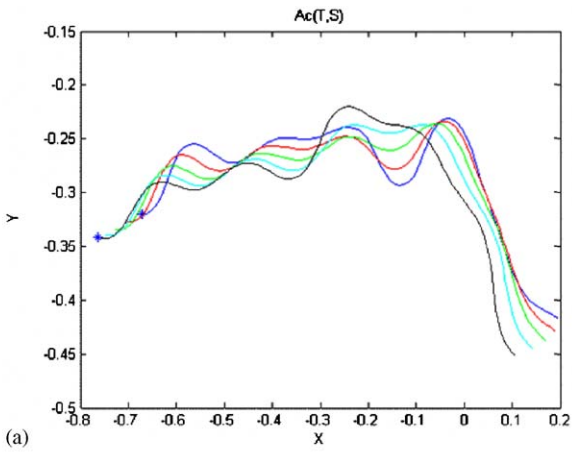
Let $C_t(i)$, $i = 1, \dots, 7$ be the existing skeleton states for all video frames $t = 1, \dots, n$. Let $R_k(i)$ be a new random skeleton state to be generated. To completely define R_k the modulus and phase for all seven components of R_k need to be set. Each modulus $|R_k(i)|$ of the new state is strongly related to the curvature of the shape, and so the valid values should lie between the corresponding minimum $C_{\min}(i) = \min_t(|C_t(i)|)$ and maximum $C_{\max}(i) = \max_t(|C_t(i)|)$ modulus of all the existing states. The angle $\angle R_k(i)$ is related to the position where the snake skeleton is curved. Since the snake skeleton can have a curve at any of its skeleton point, no constraints are imposed on $\angle R_k(i)$. So $R_k(i)$ is defined by Eq. (8) (r, g are random numbers in $[0, 1]$):

$$\begin{aligned} |R_k(i)| &= C_{\min}(i) + r \cdot (C_{\max}(i) - C_{\min}(i)), \\ \angle R_k(i) &= 2\pi \cdot g. \end{aligned} \tag{8}$$

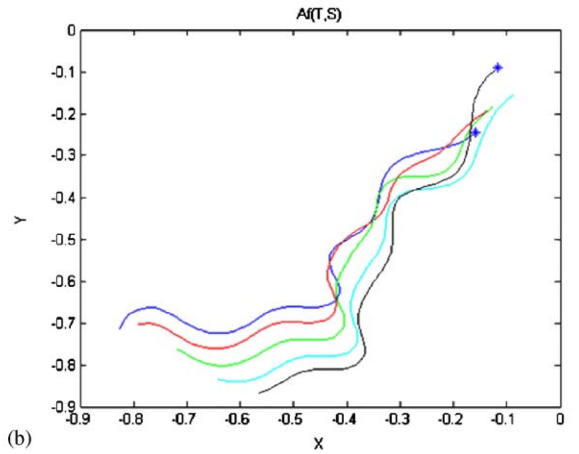
The edges of the motion graph are defined by connecting skeleton states whose euclidean distance is below a specified threshold. Therefore, the construction of the motion graph resembles the generation of a probabilistic roadmap [17]. A motion sequence can be produced by starting from an arbitrary state (node) of the graph and then selecting the next states by traversing edges of the graph. During the state selection process we use the following criterion so that non-periodic motions are obtained. For each state q we store the number of times $v(q)$ that this state has been selected. Among all neighboring states of the current state, the one that maximizes the following function is chosen: $W(q) = (1/2^{v(q)}) \cdot \deg(q)$, where $\deg(q)$ denotes the degree of node q. This way, those states that have not been visited many times and are also connected to many other states are preferred.

After the path traversal, a set of states (consisting of seven-tuples) will have been selected. For the complete motion synthesis, the intermediate states between two consecutive states are produced by using linear interpolation of the corresponding FFT coefficients. Using these states and an inverse FFT, the angles $A_c(t, s_k)$ of the skeleton joints at any time instance t can be recovered.

The produced snake motion looks realistic for the case of water snakes (Fig. 12(a)). However, this is not true for the case of snakes that live in land.



(a)



(b)

Fig. 12. (a) Skeleton points as recovered by using angles $A_c(t, s)$; (b) skeleton points as recovered by using angles $A_f(t, s)$. The blue and black skeletons correspond to the first and last states, respectively. The * point corresponds to the position of the head.

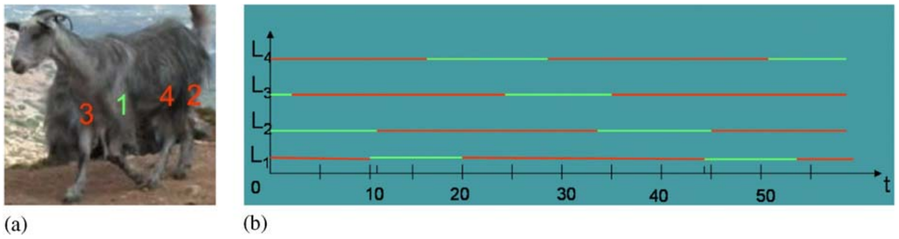


Fig. 13. (a) A frame from the goat video sequence in which legs 2, 3, 4 are touching the ground while leg 1 is on the air; (b) time chart showing which of the four legs L_1, L_2, L_3, L_4 are touching the ground (red line segments) or not (green line segments).

Their motion is characterized by the fact that the tail motion always follows the head motion with a small phase delay. For this purpose the angles $A_c(t, s_k)$ are modified and a new set of angles $A_f(t, s_k)$ is produced so that:

$$A_f(t, s_k) = m(s_k) \cdot A_c(t, s_k) + (1 - m(s_k)) \cdot A_f(t-1, s_{k-1}). \tag{9}$$

The function $m(s_k)$ is restricted to lie in the $[0, 1]$ interval. $m(s_k)$ is close to 0 for points at the tail while it is close to 1 for points at the head of the snake. This way the angles defined at the points of the head are gradually transferred to the points of the tail (Fig. 12(b)).

Since each skeleton state is invariant to snake rotation and translation, we also need to provide the global translational component of the snake motion as well as the direction it is looking at. These can be extracted at any time based on a user specified path which contains the animal trajectory. The snake translation is defined by imposing the restriction that the center of mass of the snake always lies on that path while the direction of the snake is provided by the tangent of the path.

4.2.2. *Articulated motion*

As already mentioned in Section 4.1, the 3D model of an animal is composed of certain parts. During the articulated motion synthesis, a parametric motion model (based on physical constraints about the actual motion) is applied to each of these parts. The parameters of the motion model are estimated using a statistical analysis of the joints tracking data. Since the animal motion is characterized by periodicity, only one cycle of this movement is considered for the model description.

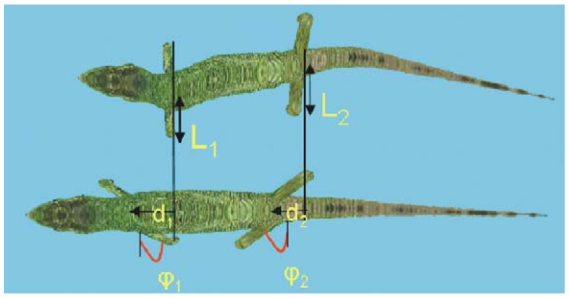


Fig. 14. The lizard motion model. ϕ_1, ϕ_2 represent the angles between the legs and the vertical axis.

This method has been tested on two animals: a lizard and a goat (Figs. 13 and 14).

The lizard consists of a main body, two front legs (assumed collinear) and two back legs (assumed collinear as well) (Fig. 14). The skeleton of the main body is allowed to bend at its intersection with these two pairs of legs. At any time during motion, the angles between the main skeleton and the legs are estimated based on the tracking data. These angles can then be used to fully specify the bending of the main skeleton during the animation.

Regarding the goat motion, the most difficult part is the extraction of the legs motion. For this reason, a time chart is computed (based on the tracking data) showing which of the four legs touches the ground at any time (Fig. 13(b)). In addition, the angles of all leg joints are computed at each moment in time. Based on this information, the parametric motion model that will be applied to the legs can be fully estimated. Using a priori physical constraints about the goat's motion, the motion for the rest of the body can be

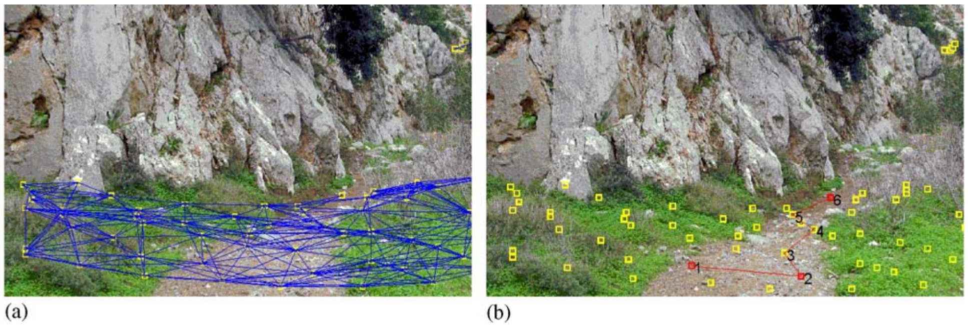


Fig. 15. (a) Weighted graph G used during motion planning; (b) the red path is the minimum score path selected by the path planning algorithm. Points (1,2,6) were specified by the user.

specified as long as the legs motion has already been computed (for more details see [23]). For producing a more lifelike animal animation, an oscillation of the goat's tail and beard are also generated during motion synthesis.

It must be noted that during the articulated motion synthesis, the produced prototype motion is always taking place along a straight line segment on the horizontal XZ plane.

4.3. *Path planning in virtual environment*

During this stage, the path (of the outer environment) that will be traced by the animal is computed. We assume that a 3D model of the outer environment is provided as input. We also assume that the outer environment consists of obstacles (like rocks, trees, etc.) which are overlaid on a mostly horizontal surface. Let $W = \{K_1, \ldots, K_n\}$ $(K_j \in \Re^3)$ be the set of vertices of that 3D model. We will denote by $X(K)$, $Y(K)$, $Z(K)$ the 3D 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 them.

The user then picks interactively a sequence of points $(\{U_1, \ldots, U_l\})$ out of the set S and the path planning algorithm outputs a path as smooth as possible which passes through these points but does not 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. 15(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 3D model. The weight of an edge is set equal to the variance of the elevation (Y coordinate) of all world 3D 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 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. 15(b). The red points (1, 2, 6) were specified by the user. The final continuous path is computed by using cubic spline interpolation.

4.4. *Motion transformation*

The articulated motion synthesis algorithm, as described in Section 4.2.2, computes a prototype motion which is always taking place along a straight line segment, say A_0B_0, in the horizontal XZ plane (Fig. 16(a)). This prototype motion should be now transformed so that the animal moves along the actual path as computed by the path planning at the previous stage.

Since we assume that the animal moves on a mostly horizontal surface (with slope less than 30°), the actual path is first projected on the horizontal XZ plane producing the projected curve A_1B_1. For the motion transformation, correspondences need to be established between the points of A_0B_0 and the curve A_1B_1. For each point P_1 of

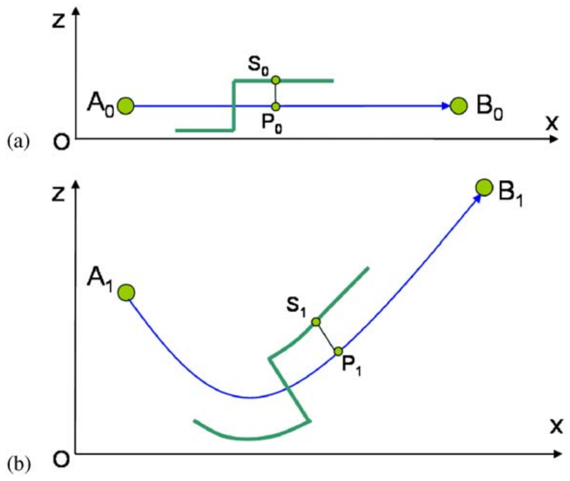


Fig. 16. (a) One skeleton state of the prototype motion along the straight line A_0B_0; (b) the corresponding transformed skeleton state for the actual curved path A_1B_1.

A_1B_1, its corresponding point P_0 in A_0B_0 is computed using the following criterion:[3] $\|A_0P_0\| = \text{arclength}(A_1P_1)$.

The animal skeleton (as estimated at point P_0) must "bend", so that it fits the curve at point P_1. This way the animal will appear turning its body upon reaching positions where the trajectory is curved. For this reason any point S_0 of the skeleton at point P_0 of the prototype motion must be transformed to a new points S_1 as follows: the point S_1 of the transformed skeleton will be lying on the normal of the curve A_1B_1 at P_1 and it should hold that $\|P_0S_0\| = \|P_1S_1\|$ (see Fig. 16).

After transforming all skeleton points, a plane (local ground plane) approximating the scene locally is computed and the animal skeleton is rotated so that it stands vertically on that plane. This procedure is repeated for every frame of the motion sequence. In Fig. 17(f) a wild goat is moving on a horizontal ground plane while in Fig. 17(g) the goat is moving on a ground plane with about 20° slope. In both cases, the animal motion has been adapted to the local ground plane.

[3]We assume that the line segment A_0B_0 has already been scaled so that its length is equal to the arc length of curve A_1B_1.

4.5. Results

Our motion synthesis algorithm has been applied to a snake, a lizard and a goat. All of these animals were then placed inside a 3D virtual environment representing Samaria Gorge. Some frames from the produced animations are displayed in Fig. 17. Two frames of a snake animation are shown in Figs. 17(a) and (b). The lizard in Fig. 17(c) has just finished taking a turn and is now moving straight while the lizard in Figs. 17(d) and (e) is just starting to turn. 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 3D animal model was about 26 s, using as input images of 1.5 megapixels. This step needs to be executed only once per animal. Also, the mean computation time for producing a 300 frames animation was about 22 s. This time includes the time occupied by the motion synthesis stage, the path planning stage (inside a 3D virtual world of 10.000 vertices) as well as the motion transformation stage. For our experiments, we have been using a Pentium 4 CPU at 2.4 GHz. Sample animations are available at the following Internet address: http://www.csd.uoc.gr/~tziritas/DEMOS/samaria.htm.

5. Conclusions

Two computer vision tools have been introduced in this paper. The first one deals with a new approach for interactive walkthroughs of large, outdoor scenes. No global model of the scene needs to be constructed and at any time during the rendering process, only one "morphable 3D-mosaic" is displayed. This enhances the scalability of the method to large environments. In addition, the proposed method uses a rendering path which is highly optimized in modern 3D graphics hardware and thus can produce photorealistic renderings at interactive frame rates. In the future we intend to extend our rendering pipeline so that it can also take into account data from sparse

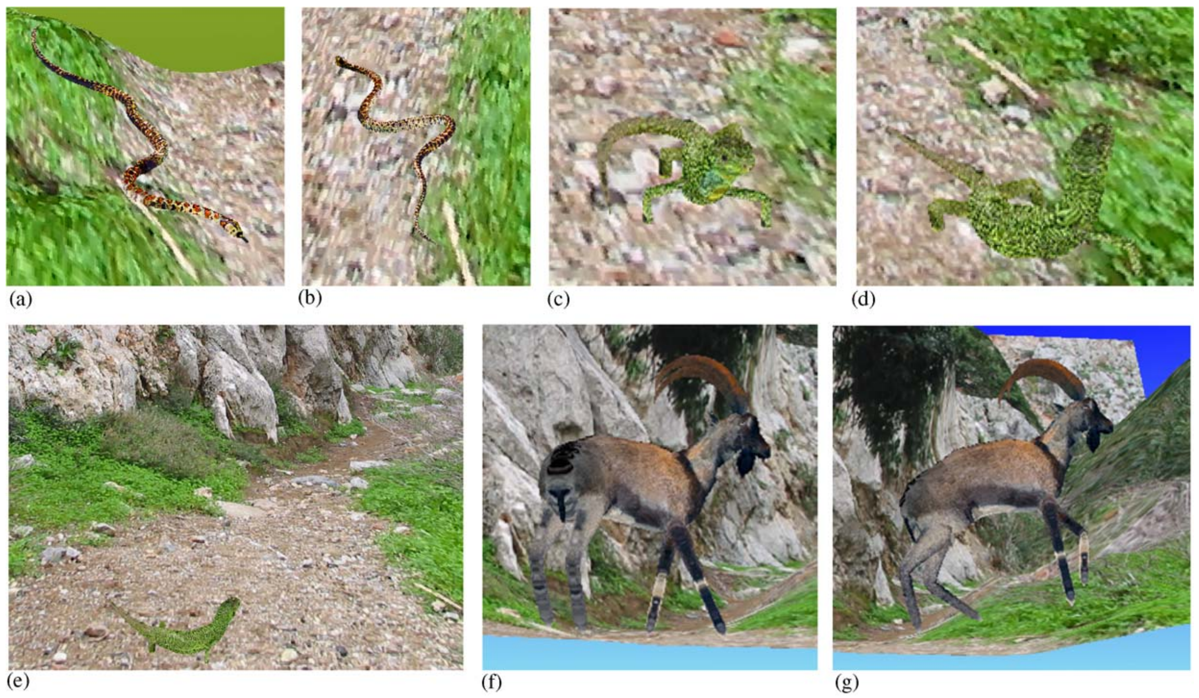


Fig. 17. (a)–(g) Some frames from the produced animal animations inside a virtual 3D environment.

stereoscopic views that have been captured at locations throughout the scene and not just along a predefined path. This could further enhance the quality of the rendered scene and would also permit a more extensive exploration of the virtual environment. Furthermore, we intend to eliminate the need for a calibration of the stereoscopic camera as well as to allow the stereo baseline to vary during the acquisition of the various stereoscopic views. Another issue that we want to investigate is the ability of capturing the dynamic appearance of any moving objects such as moving water or grass that are frequently encountered in outdoor scenes (instead of just rendering these objects as static). To this end we plan to enhance our "morphable 3D-mosaic" framework so that it can also make use of real video textures that have been previously captured inside the scene. One limitation of our method is that it currently assumes that the lighting conditions across the scene are not drastically different (something which is not always true in outdoor environments). For dealing with this issue, images obtained at multiple exposures along with HDR techniques need to be employed.

The second tool that has been introduced deals with generating realistic animal animations using as input real video sequences of the animal as well as with integrating them into a 3D virtual environment. A robust algorithm for the reconstruction of the animal's 3D shape has been proposed while both articulated and not articulated motions have been considered. Furthermore, a path planning algorithm has been implemented and a novel non-linear motion transformation method has been developed in order to transform a motion along a straight line into a motion along any curved path. One important extension of our method would be the use of multiple cameras along with multiview stereo techniques for aiding in the reconstruction of both the animal's 3D model and its motion. Furthermore, we plan to incorporate into our framework the frictional forces between the snake body and the ground, so that a more accurate representation of the snake's locomotion is obtained, while for the case of articulated motion we plan to explore more complex non-periodic animal motions in the future. Another useful extension of our method would be to allow the user of a virtual reality system to have real-time control of the virtual animal and its behavior as well as to be able to interact with it (for example by feeding the animal etc.). This would greatly enhance the user's virtual experience. To this end we also plan to investigate producing more complex articulated motions by concatenating simpler periodic motions that have been previously captured. One problem, appearing at integration stage, is that if a certain part of the virtual scene contains too many noisy 3D surface points (due to errors in the 3D reconstruction) then an animal's motion inside that part will not be smooth and will also look unnatural. In that case, the noisy surface needs to be approximated with a smoother one during the adaptation of the animal's 3D motion to the surrounding environment.

It should be mentioned that both of the presented tools have been implemented as a core part of the DHX framework. Due to the image-based representations that these tools provide as well as due to the minimal human intervention that they require they have thus contributed to one of the main goals of the DHX architecture: the development of a virtual platform offering new ways to present or create content for natural heritage. This is achieved through the augmentation of the virtual world with realistic imagery and the improvement of the authoring tools needed for the virtual reconstruction of this imagery. Finally, it should be noted that a permanent installation of our system at the National History Museum of Crete has already been planned aiming at providing the visitors of the museum with a virtual tour of the Samaria Gorge and its ecosystem.

References

[1] I. Albrecht, J. Haber, H. Seidel, Construction and animation of anatomically based human hands models, in: Proceedings of SIGGRAPH 2003, 2003.

[2] D.G. Aliaga, T. Funkhouser, D. Yanovsky, I. Carlbom, Sea of images, in: VIS 2002, 2002, pp. 331–338.

[3] O. Arikan, D.A. Forsyth, J.F. O'Brien, Motion synthesis from annotations, ACM Trans. Graph. 22 (3) (2003) 402–408.

[4] J.-Y. Bouguet, Camera Calibration Toolbox for MATLAB, http://www.vision.caltech.edu/bouguetj/calib_doc.

[5] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (11) (November 2001) 1222–1239.

[6] C. Buehler, M. Bosse, L. McMillan, S.J. Gortler, M.F. Cohen, Unstructured lumigraph rendering, in: Proceedings of SIGGRAPH 2001, 2001, pp. 425–432.

[7] P.E. Debevec, C.J. Taylor, J. Malik, Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach, in: SIGGRAPH 96, 1996, pp. 11–20.

[8] M. Dontchena, G. Yngve, Z. Popovic, Layered acting for character animation, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), 2003, pp. 409–416.

[9] A. Fang, N. Pollard, Efficient synthesis of physically valid human motion, in: Proceedings of SIGGRAPH 2003, 2003.

[10] L. Favreau, L. Reveret, C. Depraz, M.-P. Cani, Animal gaits from video, in: Symposium on Computer Animation. ACM SIGGRAPH/Eurographics, 2004.

[11] S. Fleishman, B. Chen, A. Kaufman, D. Cohen-Or, Navigating through sparse views, in: VRST99, 1999, pp. 82–87.

[12] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen, The lumigraph, in: SIGGRAPH 96, 1996, pp. 43–54.

[13] K. Grochow, S. Martin, A. Hertzmann, Z. Popovi, Style-based inverse kinematics, in: Proceedings of SIGGRAPH 2004, 2004.

[14] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge, 2000.

[15] A. Ijspeert, Design of artificial neural oscillatory circuits for the control of lamprey- and salamander-like locomotion using evolutionary algorithms, Ph.D. Dissertation, University of Edinburgh, 1998.

[16] M. Johnson, Exploiting quaternions to support expressive interactive character motion, Ph.D. Dissertation, MIT, 2003.

[17] L. Kavraki, P. Svestka, J. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robotics Automat. 12 (4) (1996) 566–580.

[18] N. Komodakis, G. Pagonis, G. Tziritas, Interactive walkthroughs using morphable 3d-mosaics, in: Proceedings of the Second International Symposium on 3DPVT (3DPVT 2004), 2004.

[19] M. Levoy, P. Hanrahan, Light field rendering, in: SIGGRAPH 96, 1996, pp. 31–42.

[20] S. Li, Markov Random Field Modeling in Computer Vision, Springer, Berlin, 1995.

[21] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, L. McMillan, Image-based visual hulls, in: Proceedings of SIGGRAPH 2000, 2000, pp. 369–374.

[22] L. McMillan, G. Bishop, Plenoptic modeling: an image-based rendering approach, in: SIGGRAPH95, 1995, pp. 39–46.

[23] C. Panagiotakis, G. Tziritas, Construction of animal models and motion synthesis in 3D virtual environments using image sequences, in: Proceedings of the Second International Symposium on 3DPVT (3DPVT 2004), 2004.

[24] M. Pollefeys, R. Koch, M. Vergauwen, L. Van Gool, Hand-held acquisition of 3d models with a video camera, in: Proceedings of 3DIM'99, 1999, pp. 14–23.

[25] D. Ramanan, D. Forsyth, Using temporal coherence to build models of animals, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), 2003.

[26] A. Safonova, J.K. Hodgins, N.S. Pollard, Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces, ACM Trans. Graph. 23 (3) (2004) 514–521.

[27] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Internat. J. Comput. Vision 47 (1–3) (April 2002) 7–42.

[28] H. Schirmacher, M. Li, H.-P. Seidel, On-the-fly processing of generalized lumigraphs, in: Proceedings of Eurographics 2001, vol. 20(3), 2001, pp. C165–C173; C543.

[29] M. Segal, K. Akeley, The OpenGL Graphics System: A Specification (Version 1.5), http://www.opengl.org.

[30] M. Uyttendaele, A. Criminisi, S.B. Kang, S. Winder, R. Hartley, R. Szeliski, High-quality image-based interactive exploration of real-world environments, IEEE Comput. Graph. Appl. (2004).

[31] S. Vedula, S. Baker, T. Kanade, Spatio-temporal view interpolation, in: Proceedings of the 13th ACM Eurographics Workshop on Rendering, June 2002.

[32] J. Wu, Z. Popovic, Realistic modeling of bird flight animations, in: Proceedings of SIGGRAPH 2003, 2003.

[33] K. Yamane, J. Kuffner, J.K. Hodgins, Synthesizing animations of human manipulation tasks, in: Proceedings of SIGGRAPH 2004, 2004.

[34] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Trans. Graph. 23 (3) (2004) 600–608.

[35] D. Zwilliger, Handbook of Differential Equations, Academic Press, Boston, MA, 1997.