Vis Comput (2014) 30:213–227
DOI 10.1007/s00371-013-0797-1

ORIGINAL ARTICLE

Sketching human character animations by composing sequences from large motion database

Innfarn Yoo · Juraj Vanek · Maria Nizovtseva · Nicoletta Adamo-Villani · Bedrich Benes

Published online: 7 May 2013

Abstract Quick creation of 3D character animations is an important task in game design, simulations, forensic animation, education, training, and more. We present a framework for creating 3D animated characters using a simple sketching interface coupled with a large, unannotated motion database that is used to find the appropriate motion sequences corresponding to the input sketches. Contrary to the previous work that deals with static sketches, our input sketches can be enhanced by motion and rotation curves that improve matching in the context of the existing animation sequences. Our framework uses animated sequences as the basic building blocks of the final animated scenes, and allows for various operations with them such as trimming, resampling, or connecting by use of blending and interpolation. A database of significant and unique poses, together with a two-pass search running on the GPU, allows for interactive matching even for large amounts of poses in a template database. The system provides intuitive interfaces, an immediate feedback, and poses very small requirements on the user. A user study showed that the system can be used by novice users with no animation experience or artistic talent, as well as by users with an animation background. Both groups were able to create animated scenes consisting of complex and varied actions in less than 20 minutes.

Keywords Sketching · Character animation · Interactive systems

Electronic supplementary material The online version of this article (doi:10.1007/s00371-013-0797-1) contains supplementary material, which is available to authorized users.

I. Yoo (✉) · J. Vanek · M. Nizovtseva · N. Adamo-Villani · B. Benes
Purdue University, West Lafayette, IN, USA
e-mail: yooi@purdue.edu

1 Introduction

Animating a 3D character is a challenging task that has been approached from three main directions. In *artistic 3D animation*, the animator uses a variety of techniques, such as keyframe animation, parameter curve editing, inverse and forward kinematics (IK/FK), and multiple targets morphing to craft the character poses and motions. In *data-driven animation* (e.g., motion capture), live motion is recorded directly from an actor, digitized, and then mapped onto a 3D character. In *procedural animation*, a computational model is used to create and control the motion, e.g., the animator sets conditions for some type of physical or behavioral simulation.

Although existing 3D animation systems provide powerful tools that are appropriate for precise character posing and realistic motion, they are expensive, and have a steep learning curve. The goal of rapidly posing 3D articulated figures, although addressed by previous work, is not fully solved. A good inspiration is traditional 2D animation where the experienced animator sketches the main character poses (keyframes) and the rough character movements using pencil and paper and the assistant animator draws the missing in-betweens.

Using 2D sketching to create 3D animation allows the user to take advantage of the benefits of 2D modeling: intuitiveness, fast pose definition, and quick production of simple, first-pass animated scenes. This paper is not the first to recognize the potential of sketches for 3D animation. The problem has already been addressed in computer graphics by various researchers [5, 19, 21, 22, 34]. However, most of the previous work uses 2D annotated sketches and matches each sketch with a single pose, either by extracting the 3D pose directly from the sketch [19] or by matching and obtaining a single pose from an existing database [34].

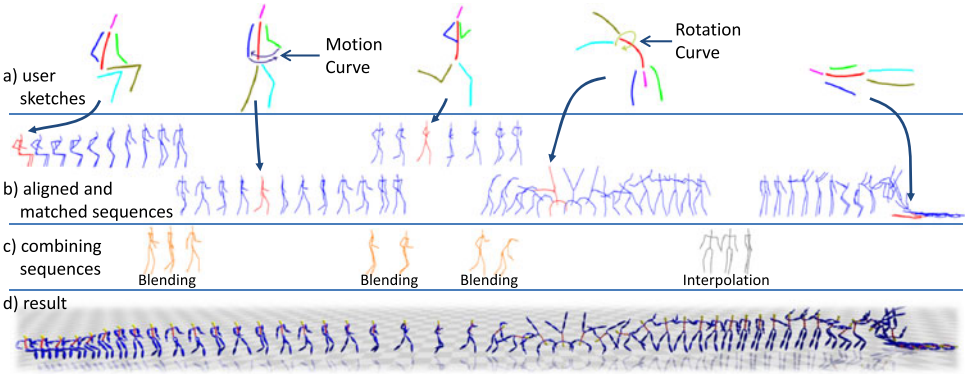


Fig. 1 The user sketches poses with additional information about the desired motion of some joints (**a**). The system finds the best matching sequences in a large motion database (**b**) and the user defines if sequence blocks should be interpolated or blended partially together (**c**), which results in the final animation (**d**)

Extracting motion sequences, rather than individual poses, from hand-drawn sketches could be an effective method to create simple, first-pass 3D character animations because it provides frame-to-frame coherence and motion continuity. The key observation of our approach is that an animated scene can be effectively built by combining the sequences from the database containing poses corresponding to the input sketches. In other words, the user input 2D pose is used within the *context* of an existing animation. For instance, when an animator sketches the walk "*contact*" pose, it is very likely that the intention is to animate a walking motion. With our approach, the contact pose sketched by the animator is considered to be part of a walking sequence. Therefore, the animator has the ability to select the identified walking sequence, or part of it, and is relieved of the task of drawing additional walking positions. However, in order to contextualize the pose within the motion, additional information should be provided. We use simple motion and rotation curve sketches to define the motion. In this way, our approach is an extension of the previous work, because it also allows for animation using single poses.

We introduce a framework for quick creation of first-pass 3D character animations from 2D sketches, as shown in Fig. 1. We use a simple and intuitive 2D sketching system. Once the sketch is drawn, it can be enhanced by 2D strokes that define motion and rotation of joints. The reconstructed 2D pose with specified motion curves is matched to a large database of motion data, and the corresponding poses are found within the animated sequences that are used as the basic building blocks of the final animated scene. The user can select from a set of identified sequences, define the beginning and the end of each sequence, and very quickly compose the resulting animated scene by trimming, blending, or interpolating the sequences. The main contributions of our work include:

1. using sequences from the motion database as basic animation building blocks with the detected pose considered in the context of the animation;
2. a novel intuitive sketching interface that allows for a quick definition of the character and its movements through strokes defining joint motion and rotation; and
3. efficient matching of character poses with animation curves from a large, unannotated database of motion sequences running in parallel on the GPU.

2 Previous work

Using sketches as 2D input to generate full 3D models has been successfully applied in various areas of computer graphics. One of the first attempts was a sketch-based modeling system SKETCH [36] and Teddy [10] that used input strokes for mesh creation. These works have been followed by many, such as Gingold et al. [7], who created 3D freeform surfaces from 2D sketches, or Rivers et al. [27], who presented an approach for the creation of 3D models from silhouettes. Recently, Lee et al. [17] presented a system where the user draws a single sketch, and an improved shadow image is derived automatically. They can generate objects with correct proportions and spacing, and it could be applied for 2D character drawing. Another area where 2D sketching has been successfully used is for facial expressions by [16] and [29]. Ulicny et al. [32] used sketching for crowds. Thorne et al. [31] used a predefined vocabulary of 18 motions to create high level control for sketching a 3D animation.

Sketching as an input for articulated characters has also been suggested. An approach that is closely related to ours is the work of Davis et al. [5], who created a sketch-based system where the user draws a 2D stick model with annotated joints and bones, and the system reconstructs possible 3D poses to create the final animation. This system, however, uses stick figures as the input, whereas ours uses simple freehand strokes that are more intuitive to many users. Mao et al. [21] expanded this approach by online joint and bone recognition during the sketch procedure. They applied a projection method to the numerical solution by considering the joint range of motion and joint grouping. Annotated

poses were used to create 3D animated character meshes in [22]. Recently, physics has been used to enhance control of animated characters in [24]. Hecker et al. [8] presented a key framing and posing system for animating characters with unknown morphologies that uses an inverse kinematics solver to animate the character. Recently, Lo and Zwicker [20] presented a system that allows animation of characters by sketching motion trajectory combined with a search in motion database. However, their approach does not allow creation of the pose that must be already created.

The aforementioned methods provided easy ways for 3D stick-figure reconstruction and creation of animation. The problem with these forward approaches is that they are prone to generating false positives; thus the user is typically asked to correct the output. One preferred solution is by using databases of motion-capture data. Kovar et al. [14] introduced motion graphs that were able to create motions following desired paths from motion-capture data. Arikan et al. [1] used cut and paste of motion-capture data to generate natural looking animations. Another application called MotionMaster [18] requires two user inputs: labeled sketch with joint locations and a trajectory (motion curve). This system can find and refine a 3D Kung-Fu motion-capture sequence. Since MotionMaster required detailed initial sketches, Jianyuan et al. [23] introduced an easier approach that quickly creates natural motion by gestures. However, their method also required a predefined skeleton. Lin et al. [19] used a similar approach where the input is an annotated stick figure drawn in a predefined camera position. The extracted pose is then used for matching against a database of motion-captured animations. However, their approach requires a user interaction during the semiautomated matching. Recently, Wei and Chai [34] used an approach similar to ours that exploits a motion-capture database with predefined poses. This method has unique features such as natural-pose creation using probabilities, the way to impose constraints of the inverse kinematics for overcoming the unnatural-posing problem, and a probabilistic learning system that was trained to generate millions of poses. However, this approach still requires the user to identify certain key features in the input sketch, rendering the rapid creation of animation problematic. Chao et al. [2] proposed a motion retrieval and search method that analyzes joint trajectories by using spherical harmonic functions. Jain et al. [13] used a 2D- to 3D-sketching system with a motion capture database. Similar to our approach, their 3D poses are projected and compared to 2D poses. Contrary to their method, where the user is required to create virtual markers that aid the matching, ours is fully automated. An important difference is that our framework matches the pose within the context of predefined motion sequences.

Several approaches related to our work allow for efficient searching in motion-capture data. Krueger et al. [15] used multidimensional kd-trees to improve searching and Forbes and Fiume [6] have introduced data search using weighted principal component analysis. Sakamoto et al. [28] presented a mapping method of motions into an image plane. The method improved searching motion differences from motion captured data, but did not take into account 3D motion retrieval. Real-time motion-capture data has also been used for a mapping to an animation in [12]. Pullen and Bregler [26] introduced a support tool for incorporating motion-capture data into animation by matching 2D motion to motion-capture data. Instead of detecting a static pose, our approach finds a pose within a motion context. Choi et al. [4] suggested motion retrieval and visualization using stick figures including motion curves. They compared sketches with projected stick figures using predefined camera location per character. Although the proposed method succeeded to find suitable motion, reconstructing 3D poses from sketches and rotational curves was not considered.

A substantial body of previous work is related to mapping transformations between the 2D and the 3D pose. One class of previous work attempts to determine the 3D position by estimating the position of the 3D camera. For example, Parameswaran et al. [25] calculate the 3D position of important points in a body-centric system, assuming that the hips and shoulder joints are coplanar. Chaudhuri et al. [3] presented an animation system that contains a camera reconstruction algorithm and a mesh retargeting method to find correspondence between a drawn 2D sketch and a given 3D model. The model is deformed using view-dependent IK to obtain the best match with the given sketch. However, the user still needs to manually specify correspondences between the sketched character and the reference pose joints. Our approach uses an automatic matching and camera position estimation by adaptive projecting the 3D character.

Our solution is complementary to the previous work and extends it in various directions. It is not intended to creating exact and precise animation; main usage is quick animation creation. It does not require extra information while drawing the input model; only strokes to define main body parts are needed. Additionally, a user can also specify joint motion and rotation curves, using strokes to better express desired movement of the pose. Because it is a database-based approach, it always provides a valid pose, as long as all poses in the database are valid. Moreover, our method also provides an estimation of camera position from the drawn sketch. Instead of selecting and creating animation with just single poses as a key frame, the matched pose is considered to be a part of motion sequence from the database, and those sequence blocks are used to build the resulting animation.

After this review of previous and related work, a brief overview of the entire method is provided. The individual parts of the system's pipeline are discussed afterward, namely, sketching the 2D pose and its reconstruction in

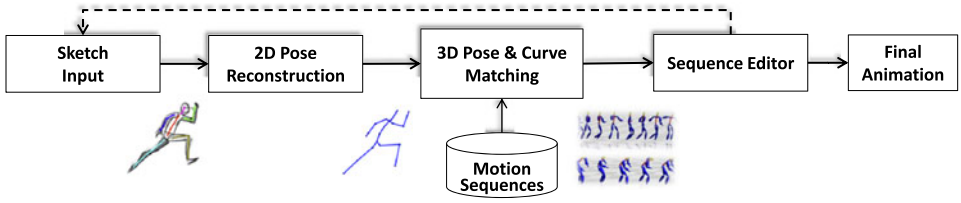


Fig. 2 The overall schema of our framework. A user-defined 2D sketch is converted into a 2D annotated pose that is matched to a 3D pose from a large database. A segment of the sequence containing the detected pose is then used in the sequence editor to compose the final animated scene

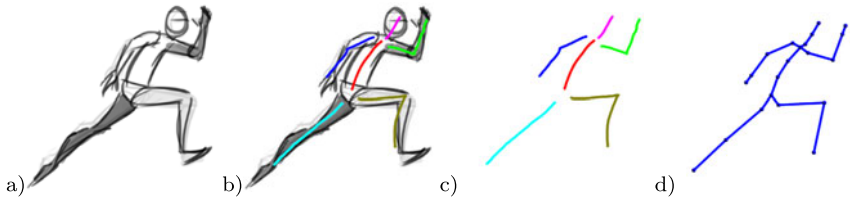


Fig. 3 The user-defined sketch (**a**) is extended by strokes that define the 2D pose (**b**), the pose is separated (**c**), and the connected 2D pose is extracted (**d**)

Sect. 4, the 3D motion-sequence database and 2D pose-matching outlined in Sect. 5, and the final assembly of the animation is in Sect. 6. After that, we present implementation and results in Sect. 7, and the paper is concluded with a section discussing limitations and possible avenues for future work in Sect. 8.

3 Method overview

The overall schema and the main blocks of our framework are in Fig. 2. The animator draws a simple 2D sketch corresponding to a key character pose (e.g., keyframe). The sketch can be in low detail and does not need to include anatomically correct body proportions; even if the sketch is very rough, the system is able to get a sufficient amount of information to extract the 2D skeleton defining the pose and comparing it against the motion database. This pose contains semantic character information about arms, legs, head, elbows, shoulders, knees, and spine.

The 2D pose itself can be a part of various 3D sequences in widely different contexts. To supply additional information that would help to better select the motion of the character, the user can optionally add motion or rotation paths near joints (see Fig. 1b). For example, drawing a motion curve near a hand joint indicates moving the hand, and an arc or an ellipse around a joint represents its rotation. These motion and rotation curves provide additional clues that are used to further improve the pose-matching process.

In the next step, the 2D pose is compared to a large database of 3D motion sequences. Because it would be difficult to directly compare 2D and 3D poses, projected 3D poses from the database are compared with the input reference 2D pose; i.e., each 3D pose in the database is projected from the estimated camera viewpoint. A similarity measure is evaluated for each projection, the sequences are sorted, the matched 3D pose is highlighted, and the sequences are shown to the user.

The selected sequences are composed in the sequence editor into the final animation. An individual sequence can be adjusted in time, trimmed, and two sequences can be blended or interpolated. The process of sketching and selecting a sequence is repeated until the desired animation is created. The animation can be either viewed directly or exported into the common file formats used in professional animation software. In the following sections, details of each part of our framework are described and discussed.

4 2D pose sketching and reconstruction

Humans have the ability to perceive the 3D structure of the character pose from a 2D drawing, and they can even guess the implied motion from a static image or sketch. However, this task is very complicated in automated recognition, and there is no robust way of recognizing 2D sketches that could be applied to direct 3D pose reconstruction. In our approach, the 3D motion is recreated from 2D sketches by imposing minimal requirements on the sketcher, who needs to draw simple strokes identifying important parts of the character's body, such as spine, head, arms, and legs. Moreover, the user can add strokes that identify the motion and rotation paths of certain parts of the body. This information is later used to match the pose to an animated sequence from the database more efficiently.

4.1 Sketch-based 2D pose definition

The user can optionally draw a base 2D sketch as shown in Fig. 3a, or upload into the system an existing sketch and

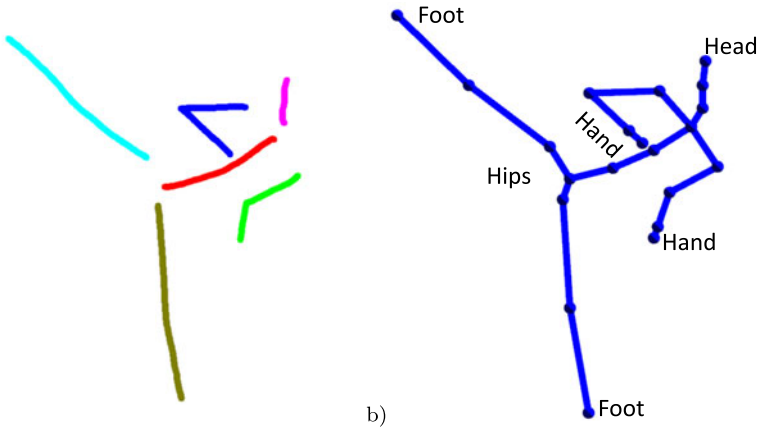


Fig. 4 User sketched pose (**a**) is reconstructed into an annotated 2D pose (**b**)

use it as a reference image, as shown in Fig. 3b. The designer draws curves identifying different parts of the character; the drawing interface was designed with classical animators in mind as they usually draw stick figures in the following order: spine, head, left/right arms, left/right legs [35]. The fixed order of the strokes of the sketch is the only requirement on the sketcher; they can be drawn quickly and without consideration of the correct proportion of body parts as shown in Fig. 3b and c. By using the fixed order, the character orientation is implicitly stated. Without this, for some unusual positions such as character hanging upside down, legs and arms could be erroneously switched. However, the order of left/right arms and legs is not fixed and all four possible combinations are searched (Sect. 5). Once the sketch is drawn, it is converted into a set of annotated line segments and joints—a 2D pose as shown in Fig. 3d and described below.

4.2 2D pose reconstruction

Because the semantic information of each stroke is explicitly known, the important parts, such as collar, pelvis, head, hands, and feet, are identified and connected. We first reconstruct joints belonging to the spine. From the strokes of arms and spine, the joint points defining the position of head, neck, pelvis, and collar is extracted. Because the lower and upper spinal joints do not significantly affect the 2D to 3D matching, the lower and upper spinal joints are located as evenly distributed from pelvis to collar.

In general, animators tend to draw the strokes defining arms in the direction from shoulder to hand. By considering the distance between the two end points of the strokes defining arms and their distance to the collar joint, we can detect if the sketch was drawn from hand to shoulder, in the opposite direction, or in any combination. From the arm orientation, the position of hands is detected. The shoulder is found when the arm is drawn significantly far from the collar joint. The elbow joints are determined by considering the angles of arm stroke. If there is a sharp angle in the stroke defining arm, its location is assigned to the elbow joint. For the opposite, the elbow is put in the center of the stroke. The same mechanism is applied to the leg strokes to detect knees and Fig. 4 shows an example of a 2D sketch and the reconstructed and annotated 2D pose.

5 3D pose matching

Once the 2D pose is reconstructed from the sketch, its occurrences in the database of prerecorded motion samples are searched. A shared advantage of all methods that use the database is that the matching always finds a valid and well-defined pose (assuming that the database includes only correct poses). A disadvantage is that new poses cannot be created because the content of the database is limited. Moreover, the time required to search in the database can become a bottleneck.

5.1 Motion snapshot database

The motion sequence database from the CMU Graphics Lab Motion Capture Database includes 4 million unannotated poses in nearly 2,400 different animation sequences occupying 3.1 GB of space and totaling 6.5 hours. The database includes sequences in more than 40 different skeleton formats, so they were converted into a single hierarchical skeleton structure. Our framework is intended for a quick animation creation; so the poses were simplified by removing toes, fingers, and wrists. However, the pose structure contains essential data for character animation such as joint hierarchy, bone direction vectors, length of bones, and rotation angles. The default Eulerian rotations were converted into quaternion representation [30] to prevent the gimbal-lock problem.

A direct look-up of the 2D pose for each 3D pose in the sequence database would be difficult; it would find many similar poses for closely located frames, and it would impose large search time if the database were to be extended.

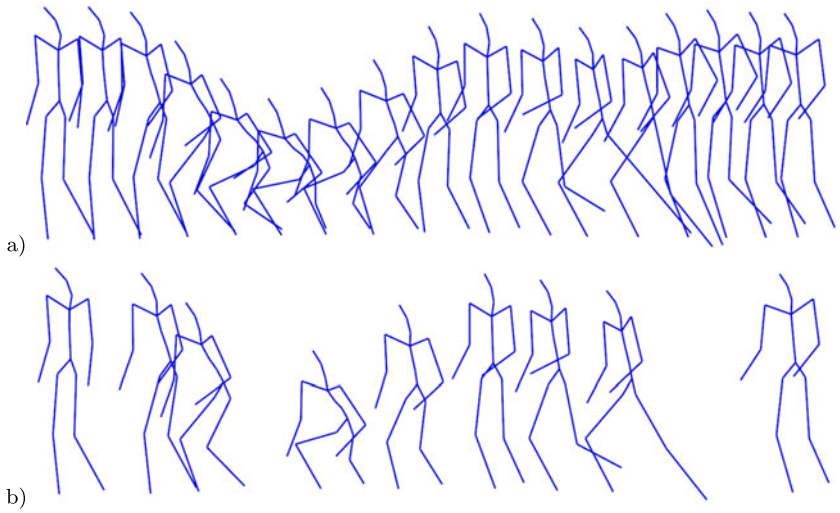


Fig. 5 The initial sequence (**a**) and the frames that have significantly different poses (**b**)

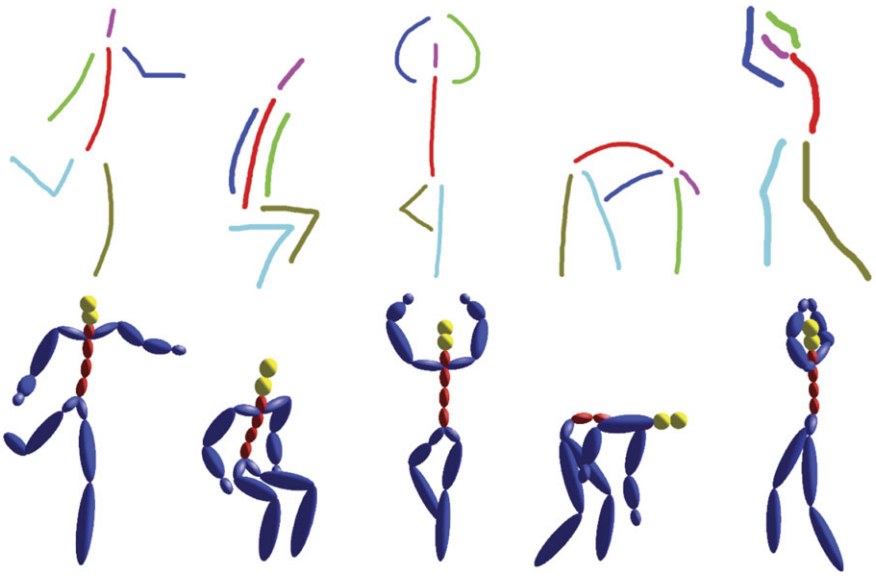


Fig. 6 Sketched poses (*up*) and poses found in the database

Also, the matching algorithm (Sect. 5.2) requires absolute positions and rotations of joints, but the motion database contains a relative animation position that is recalculated for every animation frame. To overcome these issues, an additional database called *the motion snapshot database* is created, storing each joint of the character in its absolute positions that simplifies the matching. Moreover, only poses that have a significant difference are extracted and stored. As shown in Fig. 5 the motion snapshot database also presents a significant savings of storage space that, in effect, increases the search speed.

Most sequential poses are similar to each other, as can be seen in Fig. 5. Thus, we reduce the number of snapshots in the database by saving significantly different poses. Two frames are considered significantly different if the average angular difference of all joints is higher than a predefined threshold (15° in our application), or if a single joint moves more than a predefined threshold (45° in our application) between the frames. The motion snapshot database stores only significantly different frames and it has about 200,000 different poses out of the 4 million from the input. This reduces application memory requirements in the matching phase significantly.

5.2 2D to 3D pose matching

In the next step, we detect 3D poses that match the 2D input pose. Because the sequences are independent, the 3D pose matching is done on the GPU in parallel. Sample results of the lookup can be seen in Fig. 6.

A correct projection of the 3D pose to match the 2D pose requires estimating the camera position in which the sketch has been drawn, and it is done by an iterative approach. First, each 3D pose from the snapshot database is projected from

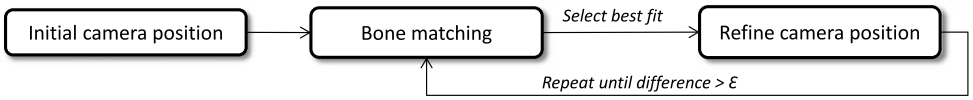


Fig. 7 The estimation of the camera position. From the initial camera position, reference and matched skeleton are compared and best fit is selected. Angle associated with the best fit is used to refine the camera position until differences are insignificant

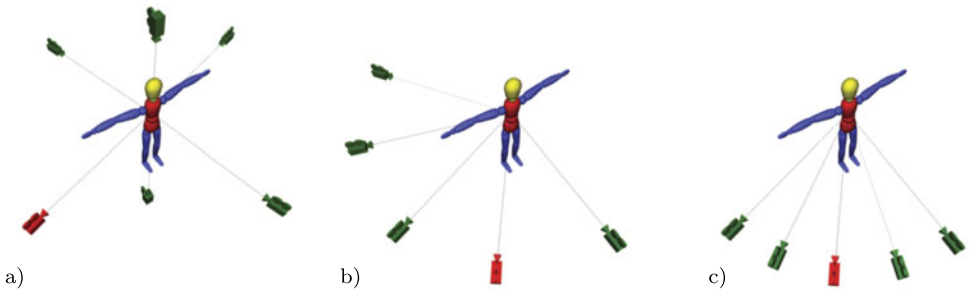


Fig. 8 2D pose matching. The pose is sampled from the six principal directions (**a**), the best match is found, and refined (**b**, **c**)

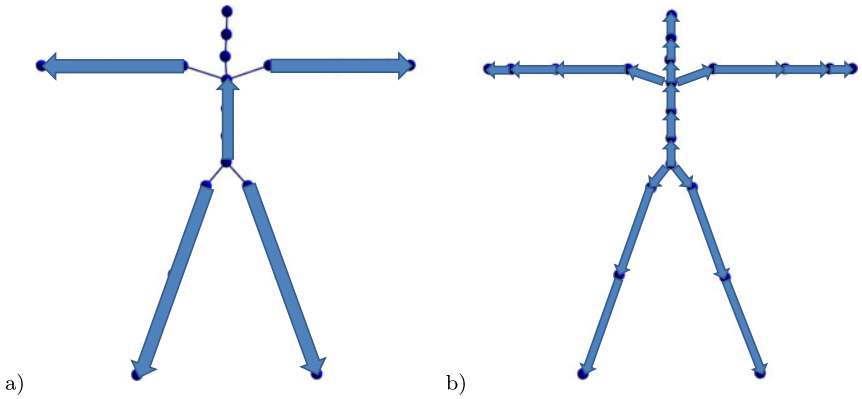


Fig. 9 In the first pass of the matching algorithm, only the main parts of the skeleton expressing the overall pose are compared (**a**); the best candidates from the first pass are compared in the second pass (**b**). Bone weights were established after discussion with animators and they define significance of each bone during the matching process (most significant are the limbs)

six basic views: front, back, left, right, top, and bottom (see Fig. 8). Top and bottom views have low priority because animators rarely draw sketches from these views, and they can be difficult to detect because of the overlapping curves in the pose. The projected 3D poses are then compared with the reference 2D pose, and their angular similarity is evaluated.

The view with the best fit is selected, and the match is further improved by binary searching of the area around this viewing direction and its neighbor views. This process continues iteratively until the differences between the detected viewpoint and the two successive steps are insignificant (Fig. 7).

To evaluate the robustness of our algorithm, we loaded 600 random poses from the database and put the camera at random locations obtained from regularly sampled points on a sphere. The 3D pose was projected from that direction, joint positions were jittered, and we then attempted to find the original pose in the database. The results show that even with high jitter values above 5 % which changed the pose significantly, the success rate was above 75 %. This has been later verified by the user study, where the intended poses were found for the vast majority of the sketches.

Bone matching priority We have observed that the angles between different parts of the body express the overall pose better than the actual joint positions, and there is no need to normalize poses. This is the reason why angles rather than joint positions were used for the comparison. When comparing 2D poses with the projected 3D pose, some bone relationships are more relevant to overall pose configuration: spine line (pelvis to collar joints), hand lines (shoulder to hand joints), and leg lines (hip to foot joints), as well as limb joints such as elbow and knee, are also important, since they provide the configuration of arms and legs.

Based on this observation, a two-pass method for pose detection has been developed (Fig. 9). The first pass quickly discards invalid poses by detecting differences between the most important parts of the character. The best 10 % poses of the candidates from the first pass are then compared in-depth in the second pass, the poses are sorted, and the best options are displayed to the user. The threshold of 10 % has been found by measuring the error of the search, which was 2.5 %, as compared to the full search using only the more-precise second pass. However, the two-pass approach reduces the search time to 30 %. Further increasing this threshold brought no significant improvement to the search. Tech-

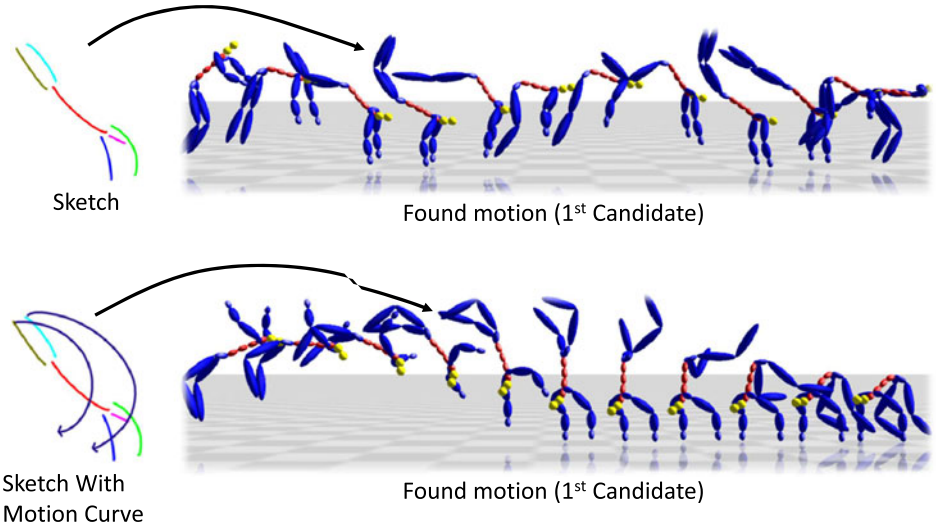


Fig. 10 Pose definition with motion curve (*down*) helps to find the intended flip sequence

niques based on kd-trees [15] or weighted PCA [6] could be used if a faster search is necessary.

Let us denote the normalized 3D bone vector $\mathbf{v}_i$ and the corresponding normalized bone vector from the 2D pose $\mathbf{u}_i$. Because the exact camera location is known from the previous step, $\mathbf{v}_i$ is orthogonally projected from the camera resulting in $\mathbf{v}_{\perp i}$. Let us denote α_i the angle between the projected vector from the 3D pose and the corresponding vector from the 2D pose

$$\alpha_i = \arccos(\mathbf{v}_{\perp i} \cdot \mathbf{u}_i).$$

The normalized error ε between the 2D pose r and the 3D projected pose $p_\perp$ is then

$$\varepsilon(p_\perp, r) = \frac{1}{n \cdot \pi} \sum_{i=0}^{n-1} w_i \alpha_i, \tag{1}$$

where w_i is the importance weight of each bone as shown in Fig. 9 and n is the number of joints in the pose. Although the comparison is the same for both passes, the bones and the importance weights (see Fig. 9) are different for each pass. Importance weights were established according to the priority of the bones determined after a discussion with animators. Lastly, the *pose matching score* between the 3D pose p and the 2D pose r is

$$S_p(p, r) = 1 - \varepsilon(p, r). \tag{2}$$

Although the matching process converges rapidly, it becomes computationally intensive with a high number of poses. However, the sequences in the database are independent, and the matching process can be easily parallelized. We have implemented this algorithm on the GPU by using CUDA and the matching process takes less than one second for direct pose comparison and less than five seconds for comparisons with curves. This performance enables us to run multiple matching steps for all possible configurations of the pose with swapped left/right hands and legs in order to explore more configurations and resolve ambiguities. These ambiguities result from the fact that the order of left/right hands and legs is not fixed and also comparing hands and legs of hand-drawn 2D character with predefined 3D characters yields to multiple possible configurations.

5.3 Motion and rotation curves matching

The static pose matching gives reliable results for static poses, but it can be further improved by considering the pose in the context of the motion sequence. The user can define two kinds of additional motions—motion or rotation of a joint by sketching a curve in the proximity of a joint (see Figs. 10 and 11). This is inspired by the sketch design in 2D animation, and it is intuitive and easy-to-understand not only for professional, but also for novice users.

Motion curve The node affected by the user-sketched motion curve is detected by finding the closest joint to the center of the bounding box of the curve. If this is not the intended node, the user can override this automatic selection. The sketched curve is first resampled and smoothed into approximately 100 points to simplify the further steps. The two principal perpendicular directions of the sketched motion curve are detected using principal component analysis algorithm (PCA). The curve is then transformed so that the first sample point is in the origin, and the main axis is aligned with the x-axis of the coordinate system. The curve is then divided into monotonous blocks, and each block's direction is coded as one of the four: up-left, up-right, down-left, and down-right. Each block is then sampled in a predefined number of points, for which the first derivative c' and the curvature κ are calculated.

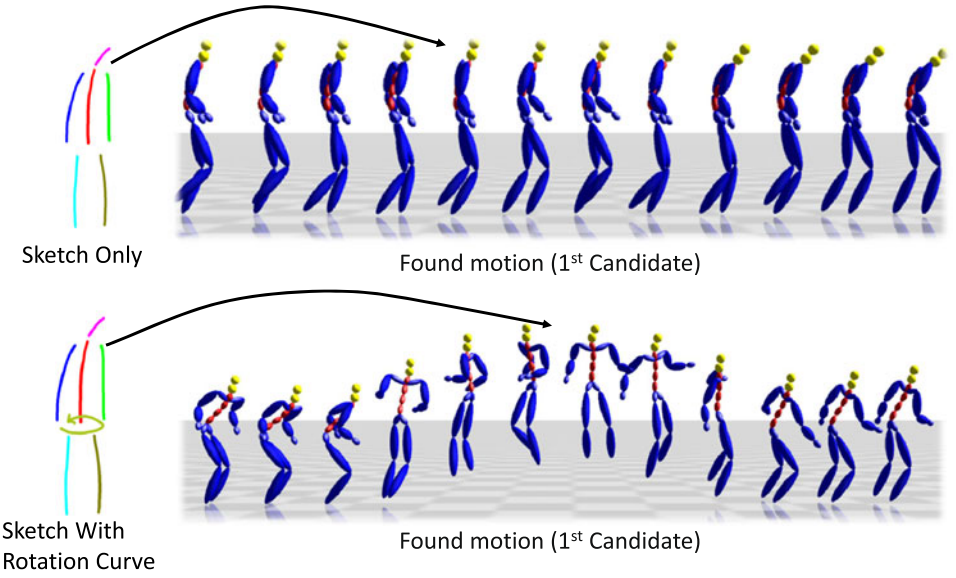


Fig. 11 The rotating character (*down*) was found with the help of the sketched rotation curve around the hip

In the next step, the *motion curve matching score* $S_m(p, r)$ between 3D pose p with the motion curve c_p and the 2D pose r with the motion curve c_r is calculated. The 3D pose p is considered within a one-second time interval ($\pm 1/2$ sec). The motion curve of the compared joint is then projected from the camera view and compared with the user-sketched curve. The projected curve is divided into monotonous blocks, and the prevailing directions are compared with the input. If the directions do not match, the matching score is set to zero. If all parts of the curve have the same direction, the $S_m(p, r)$ is calculated as

$$S_m(p,r) = 1 - \frac{1}{2n}\sum_{i=0}^{n-1}\left(\left|c'_p(i) - c'_{r\perp}(i)\right| + \left|\kappa(i) - \kappa_\perp(i)\right|\right), \tag{3}$$

where $c'_\perp(i)$ and $\kappa_\perp(i)$ are the first derivative and curvature of the ith block of the projected curve c_p.

Rotation curve We can assume that the rotation curve (Fig. 11) is essentially a circular arc that can be unprojected by using determined camera position (Sect. 5.2), and the angle of the rotation can be measured in 3D. To apply the rotation, we need to find the rotation axis and the rotation angle.

The affected node of a rotation curve is detected by finding the closest joint to the center of the curve (similar as with motion curve). The curve is then resampled, and the PCA finds the principal directions. The rotation curve is an arc distorted by projection so that the two radii of the ellipse can be detected. By comparing the radii, the ellipse is unprojected, the rotation axis is tilted, and the best candidate bone that is parallel to the 3D axis is found. The angle of rotation α_r is then calculated from the two end points of the joint rotation curves.

The joint rotation axis and the rotation angle are compared during the database curve matching. Similar to the motion-curve matching, the rotation curve is considered within the context of one second of the animation. The rotation angle of the joint is calculated by numerical integration over the time interval resulting in the rotation angle of the joint α_p for the interval of one second. The rotation angles are then compared, resulting in the *rotation curve matching score*

$$S_r(p,r) = 1 - \frac{|\alpha_p - \alpha_r|}{2\pi}. \tag{4}$$

5.4 Sequence selection

Depending on the user-provided input, there are various options for pose matching. The final matching score of the 3D pose p and the 2D pose r is denoted by $S(p, r)$ and it is used to sort the matched sequences that are offered to the user for selection.

Pose matching only If the user provides a single 2D pose with no additional information, the pose matching score from Eq. (2) becomes the absolute score so that $S(p, r) = S_p(p, r)$.

Pose matching with rotation or motion curve If the user inputs a pose with motion and rotation curves $C = \{c_1, c_2, \ldots, c_{|C|}\}$, the final score is evaluated as the weighted sum of the pose- and curve-matching results

$$S(p,r) = (1-w)\cdot S_p(p,r) + \frac{w}{n}\sum_{i=1}^{|C|} S_i(p, r_i), \tag{5}$$

where $S_p(p, r)$ is the pose-matching score from Eq. (2), $S_i(p, r_i)$ is the motion- or rotation-curve-matching scores

from Eqs. (3) and (4) depending on whether c_i is a motion or a rotation curve, and if w is the user-specified weight. We use $w = 0.25$ in our application that makes the pose matching more important than the curve fitting.

Multiple pose matching If the user specifies multiple poses with or without curves, we use an approach inspired by [33] who use weighted linear blending to create the output sequence. Let us denote the input 2D poses $R = \{r_1(t_0), r_1(t_1), \ldots, r_{|R|}(t_{|R|})\}$ and the matching attempts to find multiple poses in the same sequences. We assumed that each pose is allotted to a different time step $t_1 < t_1 < \cdots < t_{|R|}$. The score for multiple matches is then calculated as a weighted sum of the contributing poses.

$$S(p, R) = \sum_{i=1}^{|R|} \big(S_p(p, r_i) \cdot G(t_i, t)\big), \tag{6}$$

where $S_p(p, r_i)$ is the score of the ith pose from Eq. (2) or from the pose with curve Eqs. (3) and (4); t is the relative time of the pose p; t_i is the relative time of the ith pose; and $G(t_i, t)$ is the weight of the contribution of the Gaussian.

When the scores of all poses are evaluated, the sequences are sorted according to their score, and the user selects the best-fitting one.

6 Sequence editor

The sequence editor allows the user to align the sequences' view directions by rotating the selected sequence and also to efficiently combine the selected sequences into the final animation. The sequences can be trimmed, resampled, blended, and interpolated. The operations that use multiple sequences also use the motion and rotation curves, if supported, to assist the transition.

An existing sequence can be trimmed from the beginning or from the end. In the extreme, the entire sequence can be trimmed into an individual pose. However, if the motion curves were used for certain joints, the pose will still contain the information about their motion that will be used in the interpolation and blending. If the timing of a sequence needs to be modified, it is resampled by changing its length in the sequence editor.

Two sequences can be blended into one by connecting the trajectories of motions and rotations from different sequence blocks. This operation is applied when the user locates the sequences in the sequence editor in such a way that they (partially) overlap. The quaternion interpolation [30] is used to avoid the gimbal-lock problem. Similarly, if two sequences are separated in time, the quaternion interpolation is used to connect them.

The final animation can be previewed directly in our framework, or it can be exported and further processed in an external application. Although blending and interpolating of the sequences provide an easy and quick way of prototyping, the sequence editor has some limitations, e.g., it cannot change the path of the animation or natural motion transitions are not guaranteed.

7 Implementation and results

Our framework is implemented in C++; it uses OpenGL and GLSL for rendering of the character poses and CUDA for fast, parallel sequence matching. All tests were done on an i7 Intel CPU equipped desktop with 16 GB of memory, and NVIDIA GTX 480 graphics accelerator.

The user interface was designed to be as simple as possible. It has five panels to control the animation process: sketching, transition, matching, animation, and pose selection as can be seen on the accompanying video.

The sketching panel contains tools to sketch the pose using strokes. The user can load a background image as a reference for drawing strokes. The strokes are indicated by different colors for a better control, and each stroke can be redrawn if necessary. Motion and rotation curves are also defined by strokes.

When the sketch is finished, it is matched with the database, and the matching panel shows the most similar sequences containing the drawn pose in descending order. The feedback is immediate, and the sequences are animated. They can be stopped, slowed down, or zoomed, and the user can select any of them by scrolling through results. The GPU-oriented implementation provides immediate results and stores the order of the entire database of 6.5 hours on an off-shelf computer. In addition, our approach does not require additional database pre-processing, so that any motion capture data can be used immediately.

Performance comparisons between the GPU and the CPU matching with various iterations during the camera searching process is shown in Table 1. GPU comparison is 500–800× faster than matching using only the CPU and it allows for real-time search of the best fitting poses.

The selected sequence is brought to the transition panel where it can be further edited by trimming, blending, and resampling. All changes can be immediately seen in an animation panel where they are composed into the final animation, which can be exported into the common animation formats and further edited in professional animation programs.

A user study was performed to evaluate our framework. Four professional animators with experience in using animation packages and five novice users were asked to create three sequences demonstrating animations with varying complexity. The novice users needed additional time to get familiar with the interface (this time is also included in the measurement). There was no limit on the number of

Table 1 Performance comparison between CPU and GPU implementation

Iterations	32	64	128	256
CPU (Core i7 920, 2.66 GHz)	9,351 ms	18,647 ms	37,711 ms	75,193 ms
GPU (NVidia Geforce GTX480)	17 ms	28 ms	48 ms	88 ms
GPU Speedup	550.0588	665.9643	785.6458	854.4659

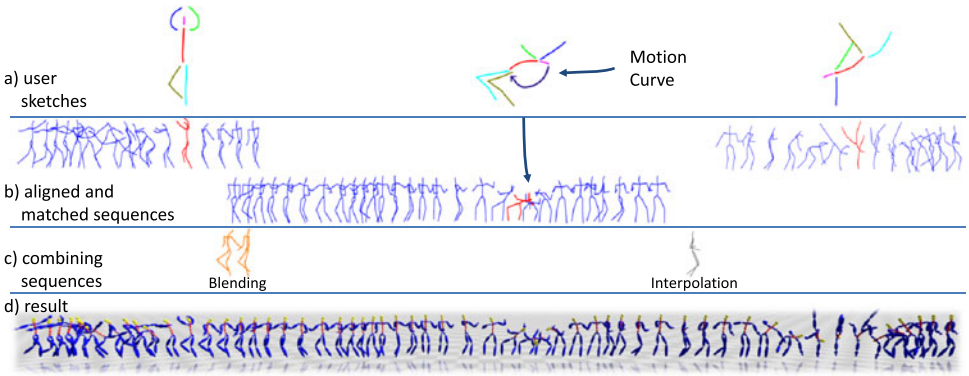


Fig. 12 The resulting animation showing the composition of three different dances. The first sequence showing a classical dance blends with a modern dance that is interpolated into another sequence of modern dance. Line (**a**) shows the input sketches, (**b**) shows the used sequences with the detected poses, and (**c**) shows the sequences generated as a result of processing the detected sequences. The last line (**d**) shows the final animation

sketches, nor on the maximum and minimum durations of the sequences, and the participants were free to use any sequence blocks with the only requirement being that the resulting animation should follow the provided scenario.

Sit-stand-run-flip-fall (SSRFF) The first animation is shown in Fig. 1 and consists of a sitting character that stands up, walks, starts running, flips over an obstacle, and falls on the ground. It is a fairly complex animation requiring variety of sequences that demonstrate all aspects of our system. On average, this animation took about 20 minutes to complete. At least 6 sketches (average 10) and 5 sequences were used to complete the task. The length of the resulting animation varied from 8 to 19 seconds with an average of 14.6 seconds.

Dance The second sequence required the test subjects to create an animation that would mix various dance styles. As a result, they combined several classical dances with modern dances, as shown in Fig. 12. This animation shows usage of the interpolation and blending between various sequences. It required at least 4 sketches (average 6.75) and 4 sequence blocks. Length of the final animation was around 20.4 seconds (longer parts from motion sequences were used), and it took about 12 minutes to complete. This sequence has required a variety of motion and rotation curves for the defined poses. The motion curves helped to distinguish a simple walk from dancing motion, where the body moves in a specific way.

Picking a coin In this acting example, the character finds a coin, grabs it, and then flees the scene (Fig. 13). The purpose was to create a scene that could be used as part of a real scenario. A variety of animations was used, as well as motion sequences. The average animation was 11 seconds long, it was created with 4 sketches and 3 sequences (walk, pick, and run), and it took under 10 minutes to complete.

Table 2 shows the averaged values of time that animators spent creating the animation, how many sketches were drawn, how many sequences were used to build it and what its resulting length was. As additional information, animators used 35 % of the time for the operation of interpolation and about 65 % for blending. Most of the blending operations were also trimmed at least from one side.

The SSRFF animation was the most complicated requirement; it took the most time to create, about 20 minutes and used 10 sketches on average. In traditional software packages like Maya, even a highly experienced user would take several hours to preform a similar task. In all animations, the subjects spent most of the time was selecting the right sequences in the list and editing them. Also, sequence blocks used were tied with the number of sketches drawn (some sub-sequences required more than one sketch with motion curves). We have observed that users tend to draw the poses first and if a good match is not found, they extend them with the motion or rotation curves to refine the search. An example is common motions, such as walk and run, which are very general, and they are present in different variations in many motion sequences. These sequences were refined and

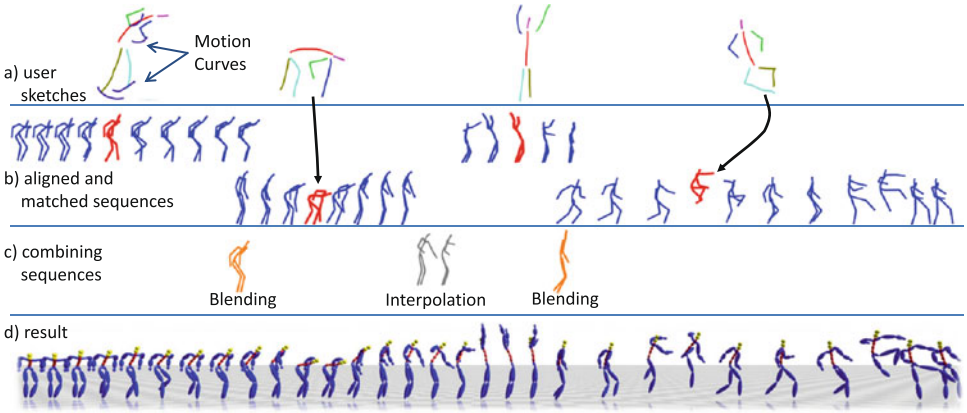


Fig. 13 An action sequence of picking a coin was created from three sketches (**a**). The detected sequences (**b**) were blended (**c**) resulting in the final animation (**d**)

Table 2 Results of the user study

Animation	Creation time (min.)	# of Sketches	# of Blocks	Animation length (s)
SSRFF	20.9	10	5	14.7
Dance	12.3	7	4	20.3
Pick a coin	9.8	4	3	10.9

matched much more easily with the help of the motion and rotation curves. In general, sequences from the large motion database significantly speed up the animation process, and it is possible to create a rough first-pass animation within minutes.

Participants were also asked to comment on the intuitiveness of the interface. They found drawing simple skeleton and using motion curves to refine the motion intuitive and easy to use. Also the fixed order of drawing the figure was not perceived as a limiting factor, as most of the animators tend to draw in the same order.

Time required to create the animation was approximately the same for the novice and also professional users. That seems to indicate our system can be used without any prior experience with the animation software and techniques because the way our application is used differs from the traditional animation packages. However, more in-depth user study would be required to justify this assumption. Also, the learning process seems to be fast, after a 15 minutes of demonstration, participants were able to create their own animations.

Figure 14 shows different motions that are created by our system using simple sketches. Samples of the animations created by the participants are included in the accompanying video.

Additional evaluation was performed by comparing our algorithm with the Dynamic Time Warping (DTW) approach introduced in [18]. We have tested 2,000 motion curves. Each curve was randomly jittered 500× by displacing its vertices perpendicularly by distance given as a percentage of the curve length. So jittering by 0.1 displaced the vertices randomly by 10 % of the length of the curve. After that, the curves were resampled so that they have exactly 12 vertices that was the maximum reasonable time for the DTW algorithm. The randomization step was increased by 1 %. Then the input and the randomized curve were compared using our algorithm and DTW.

Precision of the similarity detection is shown in Fig. 15. The x-axis shows the randomization of the input curves and the y-axis shows the percentage match. As can be seen, our method provides better result.

The speed of the comparison depends on the number of compared vertices. The DTW is a recursive algorithm and its processing time increases exponentially; comparing 15 sampling points for two curves took more than 45 seconds. However, in order to achieve a good accuracy we require to match at least 50 points from the input curve, which was impossible using the DTW (the comparison time was over several hours). Our method required only 5 ms to compare curves with 50 sampling points, thus we conclude that the DTW algorithm is not suitable for processing in real-time whereas our methods performs well.

8 Conclusions and future work

We have presented a framework for a quick creation of first-pass 3D character animations from 2D sketches using a large motion database. The basic building block of our framework is a sequence from the database with a registered pose or multiple poses. The sketches can be enhanced by motion or rotation curves that help the searching process and the combination of the detected sequences into the final animation.

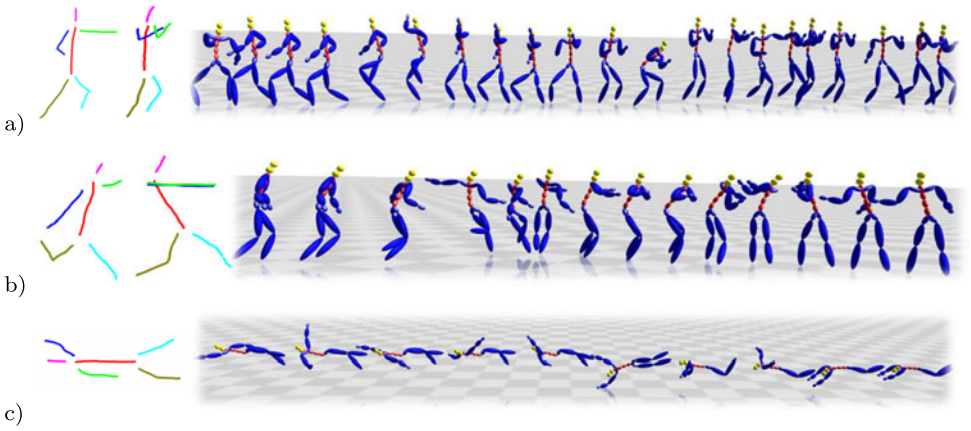


Fig. 14 Kung-fu to boxing motions (**a**), soccer kicking to baseball swing motion (**b**), and backstroke to the butterfly swimming motion (**c**)

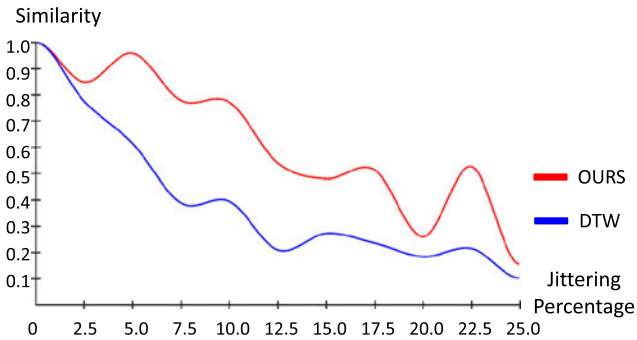


Fig. 15 Comparison of precision of our method and DTW

The additional motion curves allow defining the pose in the context of the animation that is useful for common animations such as walk. These sequences can then be refined, and the motion curves provide an important detail that helps the animation process. The user study indicates that there is no significant difference for advanced animators and novice users as all participants were able to create the animated sequences quickly and they used all the functions that the framework provided.

Our system has various limitations. The first is that it allows for the creation of an individual character that cannot be used in the context of a scene, it cannot interact with objects, or different characters in the scene. The second limitation stems from the fact that our system is used for quick creation, so it does not support the animation of hands, toes, or facial expressions. Another limitation is that our method is not suited for finding subtle motion differences. This limitation is common to database-oriented approaches. It is partially alleviated by the motion curves that allow for better tuning of the detected results, but further editing of the selected models is usually necessary to create detailed animations. Moreover, the expressive power of the system is given by the content of the sequences stored in the motion database, and the user cannot create new sequences or sequences that go beyond the simple blending or interpolation of existing sequences.

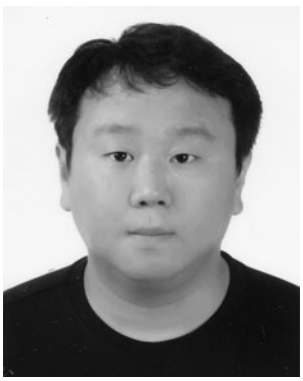
There are several possible extensions of our work. One would be in allowing the character to follow a defined path [18]. It would be a simple extension of the sketching interface, but extra work would be necessary to assure that the motion is realistic even for sharp edges or abrupt motions. Additional options to provide better navigation of the characters would be by spatial keyframing of [11] or by motion curves of [20]. Another avenue for future work would use multiple characters. An interesting approach was recently presented by Ho et al. [9], who used extended motion and retargeting for multiple animated characters by exploiting scene and character semantics. Also, in order to evaluate results more precisely, an in-depth user study with a large sample of professional and novice users would be required.

Acknowledgements The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

1. Arikan, O., Forsyth, D.A.: Interactive motion generation from examples. ACM Trans. Graph. **21**(3), 483–490 (2002)
2. Chao, M.-W., Lin, C.-H., Assa, J., Lee, T.-Y.: Human motion retrieval from hand-drawn sketch. IEEE Trans. Vis. Comput. Graph. **18**(5), 729–740 (2012)
3. Chaudhuri, P., Kalra, P., Banerjee, S.: A system for view-dependent animation. Comput. Graph. Forum **23**(3), 411–420 (2004)

4. Choi, M.G., Yang, K., Igarashi, T., Mitani, J., Lee, J.: Retrieval and visualization of human motion data via stick figures. Comput. Graph. Forum **21**(7), 2057–2065 (2012). Pacific Graphics, 2012
5. Davis, J., Agrawala, M., Chuang, E., Popović, Z., Salesin, D.: A sketching interface for articulated figure animation. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03, pp. 320–328. Eurographics Association, Aire-la-Ville (2003)
6. Forbes, K., Fiume, E.: An efficient search algorithm for motion data using weighted pca. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '05, pp. 67–76. ACM, New York (2005)
7. Gingold, Y., Igarashi, T., Zorin, D.: Structured annotations for 2d-to-3d modeling. ACM Trans. Graph. **28**(5), 148 (2009)
8. Hecker, C., Raabe, B., Enslow, R.W., DeWeese, J., Maynard, J., van Prooijen, K.: Real-time motion retargeting to highly varied user-created morphologies. In: ACM SIGGRAPH 2008 Papers, SIGGRAPH '08, pp. 27:1–27:11. ACM, New York (2008)
9. Ho, E.S.L., Komura, T., Tai, C.-L.: Spatial relationship preserving character motion adaptation. ACM Trans. Graph. **29**(4), 33 (2010)
10. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 409–416. ACM/Addison-Wesley, New York (1999)
11. Igarashi, T., Moscovich, T., Hughes, J.F.: Spatial keyframing for performance-driven animation. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '05, pp. 107–115. ACM, New York (2005)
12. Ishigaki, S., White, T., Zordan, V.B., Karen Liu, C.: Performance-based control interface for character animation. In: ACM SIGGRAPH 2009 Papers, SIGGRAPH '09, pp. 61:1–61:8. ACM, New York (2009)
13. Jain, E., Sheikh, Y., Hodgins, J.: Leveraging the talent of hand animators to create three-dimensional animation. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '09, pp. 93–102. ACM, New York (2009)
14. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, pp. 473–482. ACM, New York (2002)
15. Krüger, B., Tautges, J., Weber, A., Zinke, A.: Fast local and global similarity searches in large motion capture databases. In: 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10, pp. 1–10. Eurographics Association, Aire-la-Ville (2010)
16. Lau, M., Chai, J., Xu, Y.-Q., Shum, H.-Y.: Face poser: interactive modeling of 3d facial expressions using facial priors. ACM Trans. Graph. **29**(1), 3 (2009)
17. Lee, Y.J., Zitnick, C.L., Cohen, M.F.: Shadowdraw: real-time user guidance for freehand drawing. ACM Trans. Graph. **30**(4), 27 (2011)
18. Li, Q.L., Geng, W.D., Yu, T., Shen, X.J., Lau, N., Yu, G.: Motionmaster: authoring and choreographing kung-fu motions by sketch drawings. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '06, pp. 233–241. Eurographics Association, Aire-la-Ville (2006)
19. Lin, Y.: 3d character animation synthesis from 2d sketches. In: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, GRAPHITE '06, pp. 93–96. ACM, New York (2006)
20. Lo, W.-Y., Zwicker, M.: Bidirectional search for interactive motion synthesis. Comput. Graph. Forum **29**(2), 563–573 (2010)
21. Mao, C., Qin, S.F., Wright, D.K.: A sketch-based gesture interface for rough 3D stick figure animation. In: Proceedings of Eurographics Workshop on Sketch Based Interfaces and Modeling, Dublin, pp. 175–183 (2005)
22. Mao, C., Qin, S.F., Wright, D.K.: Sketching-out virtual humans: from 2d storyboarding to immediate 3d character animation. In: Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06. ACM, New York (2006)
23. Min, J., Chen, Y.-L., Chai, J.: Interactive generation of human animation with deformable motion models. ACM Trans. Graph. **29**(1), 9 (2009)
24. Muico, U., Popović, J., Popović, Z.: Composite control of physically simulated characters. ACM Trans. Graph. **30**(3), 16 (2011)
25. Parameswaran, V., Chellappa, R.: View independent human body pose estimation from a single perspective image. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, 27 June–2 July, vol. 2, pp. II-16–II-22 (2004)
26. Pullen, K., Bregler, C.: Motion capture assisted animation: texturing and synthesis. ACM Trans. Graph. **21**(3), 501–508 (2002)
27. Rivers, A., Durand, F., Igarashi, T.: 3d modeling with silhouettes. In: ACM SIGGRAPH 2010 Papers, SIGGRAPH '10, pp. 109:1–109:8. ACM, New York (2010)
28. Sakamoto, Y., Kuriyama, S., Kaneko, T.: Motion map: image-based retrieval and segmentation of motion data. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, pp. 259–266. Eurographics Association, Aire-la-Ville (2004)
29. Seol, Y., Seo, J., Kim, P.H., Lewis, J.P., Noh, J.: Artist friendly facial animation retargeting. ACM Trans. Graph. **30**(6), 162 (2011)
30. Shoemake, K.: Animating rotation with quaternion curves. SIGGRAPH Comput. Graph. **19**(3), 245–254 (1985)
31. Thorne, M., Burke, D., van de Panne, M.: Motion doodles: an interface for sketching character motion. In: ACM SIGGRAPH 2004 Papers, SIGGRAPH '04, pp. 424–431. ACM, New York (2004)
32. Ulicny, B., de Heras Ciechomski, P., Thalmann, D.: Crowdbrush: interactive authoring of real-time crowd scenes. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, pp. 243–252. Eurographics Association, Aire-la-Ville (2004)
33. Wang, J., Bodenheimer, B.: Synthesis and evaluation of linear motion transitions. ACM Trans. Graph. **27**(1), 1 (2008)
34. Wei, X., Chai, J.: Intuitive interactive human-character posing with millions of example poses. IEEE Comput. Graph. Appl. **31**, 78–88 (2011)
35. Williams, R.: The Animator's Survival Kit, 2nd edn. Faber & Faber, London (2009)
36. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: Sketch: an interface for sketching 3d scenes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, pp. 163–170. ACM, New York (1996)



Innfarn Yoo is a Ph.D. student at Purdue University in the Department of Computer Graphics Technology. He received his Master's degree from Purdue, and got bachelor degree at Konkuk University in South Korea, majoring Mathematics. He also has 5 years of working experiences in game industry. He is recently concentrating on animation, photo-realistic rendering, and parallel programming.

Juraj Vanek is a Ph.D. student pursuing a doctoral degree at Purdue University in United States. He got his Master's degree from Brno University of Technology in Czech Republic, majoring in Computer Graphics and Multimedia. His work/research focuses primary on real-time computer graphics, 3D modeling and high-performance GPU computing.

Maria Nizovtseva received her Master's degree in computer graphics technology at Purdue University. She holds an undergraduate degree in computer graphics design from the State University of Information Technology, Mechanics and Optics in St. Petersburg, Russia. She always enjoyed generating ideas and creating projects that are original and visually interesting. Currently her research area of interest is the role of sketching in CAD modeling processes.

Nicoletta Adamo-Villani is Associate Professor of Computer Graphics Technology and Purdue University Faculty Scholar. She is an award-winning animator and graphic designer and creator of several 2D and 3D animations that aired on national television. Her area of expertise is in character animation and character design and her research interests focus on the application of 3D animation technology to education, HCC (Human Computer Communication), and visualization. Nicoletta is co-founder and co-director of the IDeaLaboratory.

Bedrich Benes is an Associate Professor and Purdue Faculty Scholar in the Computer Graphics Technology department at Purdue University. He obtained his Ph.D. and M.S. degree from the Czech Technical University. His research is primarily in the area of procedural modeling, real-time rendering, and 3D computer graphics in general. To date he has published over 50 peer reviewed publications.