# Posing 3D Models from Drawings

ALEXANDROS GOUVATSOS, Centre for Digital Entertainment, National Centre for Computer Animation, Bournemouth University and Hibbert Ralph Animation, UK
ZHIDONG XIAO, Centre for Digital Entertainment, National Centre for Computer Animation, Bournemouth University, UK
NEIL MARSDEN, Hibbert Ralph Animation, UK
JIAN J. ZHANG, Centre for Digital Entertainment, National Centre for Computer Animation, Bournemouth University, UK

Inferring the 3D pose of a character from a drawing is a complex and under-constrained problem. Solving it may help automate various parts of an animation production pipeline such as pre-visualisation. In this paper, a novel way of inferring the 3D pose from a monocular 2D sketch is proposed. The proposed method does not make any external assumptions about the model, allowing it to be used on different types of characters. The inference of the 3D pose is formulated as an optimisation problem and a parallel variation of the Particle Swarm Optimisation algorithm called PARAC-LOAPSO is utilised for searching the minimum. Testing in isolation as well as part of a larger scene, the presented method is evaluated by posing a lamp, a horse and a human character. The results show that this method is robust, highly scalable and is able to be extended to various types of models.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*Shape*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms: Global search, 3D animation

Additional Key Words and Phrases: 3D animation, automatic 3D posing, optimisation, global search

**ACM Reference Format:**
Alexandros Gouvatsos, Zhidong Xiao, Neil Marsden, and Jian J. Zhang. 2015. Posing 3D Models from Drawings. *ACM Comput. Entertain.* 12, 2, Article 0 (January 2015), 15 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

## 1. INTRODUCTION

The field of animation has advanced rapidly, offering a plethora of ways to bring characters to life from sketching and joint manipulation to puppet posing and motion capture.

This variety though, requires users to have different sets of skills and work-flow styles. The disparity between the knowledge an artist is required to have may create fragmentation and increase the learning curve for even the most basic creative tasks.

Since animation has its roots in hand drawings, being able to show one's ideas on paper is a skill most animators possess. As such, many have argued sketching is an intuitive interface for artists from both traditional and modern backgrounds, be it for posing [Mao et al. 2006], animating [Davis et al. 2003], modelling [Yang and Wunsche

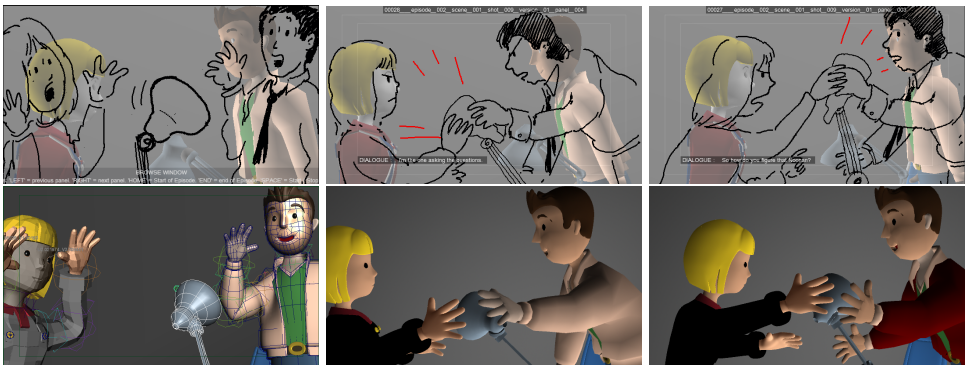

Fig. 1. From initial storyboard to pre-visualisation. The lamp in these three shots is automatically posed using the storyboard drawing. On the top row, drawings are overlaid on top of the 3D scene with the assets unposed. On the bottom row, the lamp has been posed automatically using our method.

2010], stylising motion [Li et al. 2003] or simulating secondary movement [Jain et al. 2012] and crowds [Mao et al. 2007]. By providing an intuitive link between the initial sketching process and the end result, the cognitive load and the necessary training for the artist may be significantly reduced.

Initial sketching is already part of most animation pipelines, usually in the form of storyboarding during pre-production. These drawings are then used by animators or pose artists to set up the initial pose layout before going into the animation stage. Rather than introducing a new work-flow or step, the aim of the proposed method is to automate one of the existing steps. Specifically, the focus is on making the initial posing stage of a production less labour intensive.

Posing 3D models from 2D sketches is a complex, open problem, closely related to the computer vision problem of inferring a 3D pose from a 2D photograph but with several differences. Photographs are usually accurate in terms of scale and include colour information. Sketches may have squashing and stretching, usually lack colour and contain noise in the form of auxiliary strokes e.g. from shading.

This paper focuses on using sketches to pose 3D characters automatically, for layout or pre-visualisation. It is general enough to pose any character model, in any stage of the pre-production phases e.g. storyboard blocks or hand-drawn animation frames.

Either in the field of computer vision [Hua et al. 2005] or in previous attempts at automatic 3D posing for animation [Jain et al. 2009], successful attempts have used data-driven approaches. In this case, a data-driven approach is not suitable because a pose database can be challenging and expensive to build and maintain, given the variety of fantastic characters in animation productions. As such, the method proposed in this paper does not use any database.

A hand drawing is processed and split into two descriptors: an edge map and a silhouette. The user quickly pinpoints the joints over the drawing to form the third descriptor. A variation of the particle swarm optimisation algorithm (PSO) called PARAC-LOAPSO (Section 3.2) is used, which works by manipulating the pose of a pre-existing 3D model. The process iterates, comparing the 3D render with the drawing (Section 3.3) until a suitable 3D pose is found for the character.

## 2. RELATED WORK

Before proposing a new method for automatically posing 3D models from drawings, it is important to summarise previous research. Previous work consists of the main-

stream method, posing interfaces and automatic 3D posing. Automatic 3D posing can be further spit into the *database* approach and the *explicit shape model* approach.

**The mainstream method** of posing 3D characters is by manipulating controls of a skeletal structure, usually joints [Burtnyk and Wein 1976]. Even with the invention of Inverse Kinematics [Zhao and Badler 1994], this remains a time consuming process which requires the user to be trained in specific suites of software.

A drawing represents an idea or internal model the artist has about a scene and it contains no depth information. As such, the problem of inferring the depth from a drawing is under-constrained, creating problems such as the forward-backward ambiguity problem (Figure 2).

Fig. 2. An example of the forward-backward ambiguity issue in inferred 3D pose of a lamp when not using the internal lines for additional information.

**Posing Interfaces:** Interfaces to replace the mainstream skeleton manipulation process have been the focus of previous research e.g. tangible devices [Jacobson et al. 2014]. Examples of sketch-based interfaces make use of differential blending [Öztireli et al. 2013] or the line of action to pose 3D characters [Guay et al. 2013]. These sketch-based interfaces may offer improvements, but are still operated by artists. The proposed method aims to reduce the process to an automatic method a user with no artistic background would be able to perform.

### 2.1. Automatic 3D Posing

In general, inferring a 3D pose from a drawing can be broken down into two subproblems. Firstly, matching the 3D model projection to the drawing. Secondly, inferring the missing depth. For 3D productions, the camera represents the position from which the characters in the drawing are viewed, according to the artist's internal model.

A comprehensive overview of progress in this computer vision problem [Gavrila 1999] highlights two main types of approach, the *database* approach and the *explicit shape model* approach. In this case the focus is on data from drawings as input, rather than photographs or video frames.

*2.1.1. Database Approach.* Previous computer vision research has used database approaches not only for tracking and posing [Jain et al. 2009] but also for positioning [Xu et al. 2013] in a scene or simply classifying [Eitz et al. 2012]. In animation, Jain et al. [Jain et al. 2009] follow a database approach in order to infer the missing depth and pose humanoid characters. However, it is only able to handle orthogonal camera projections, while the method proposed in this paper can handle both orthogonal and perspective projection. Building a database of poses may be expensive, especially since motion capture cannot be easily used for the non-humanoids that are so common in 3D animation productions. As such, an approach that does not require a database was preferred.

*2.1.2. Explicit Shape Model Approach.* This approach is very effective when multiple camera views are available, because the problem is no longer under-constrained (no occlusions). It consists of comparing the 3D render pose to the 2D image in order to estimate how close it is to a correct pose. One the limitations of this computer vision approach is that it requires a 3D model that is a good match to the 2D image. In 3D productions this limitation does not apply, since a 3D model exists anyway.

Favreau et al. successfully pose animals using Radial Basis Functions of 3D pose examples [Favreau et al. 2006]. Ivekovic et al. [Ivekovic et al. 2008] infer 3D poses of humans in video sequences using multi-view images. Ivekovic et al. still make human-specific assumptions, limiting its use to a small subset of characters while both Favreau et al. and Ivekovic et al., use video data and exploit temporal relations between frames.

### 2.2. Proposed Approach

The proposed approach is a method for posing 3D models from 2D drawings, aiming at pose layout and pre-visualisation. It makes no assumptions about the 3D model in order to deal with many types of characters. Unlike Jain et al. [Jain et al. 2009], it does not rely on a database. Instead, a particle swarm optimisation (PSO) variant called Parallel Convergence Local Optima Avoidance PSO (PARAC-LOAPSO) is used. This allows for posing of both cooperative (e.g. humans) and uncooperative (e.g. animals) characters. It does not require the model to have a skeleton; any type of controller is suitable. Unlike Ivekovic et al. [Ivekovic et al. 2008] the input data are monocular, stand-alone drawings with no temporal relations between them. Minimal user input is required, in the form of pinpointing the joints on top of the drawing. To avoid forward-backward ambiguity, the proposed method uses a combination of various descriptors [Fleischmann et al. 2012]. There are currently no other methods that perform automatic posing for the layout phase of animation, that work irrespective of the character model and that can be applied without the need for a pre-existing database.

## 3. METHODOLOGY

### 3.1. Overview

To propose a general posing method which can automate the initial layout pass or pre-visualisation of a production, the solution takes into account several requirements:

— That it makes no assumptions about the type of character, in order to be versatile (e.g. humans, horses, lamps, octopuses).
— That it provides results that are good enough for the initial posing and not the final animation.
— That it ensures the necessary user input is quick and requires little learning from the user side.
— That it does not restrict the traditional pipeline structure.
— That it is feasible for both small and big budget productions.

Since the proposed method is focusing on posing and not generating a 3D model from a 2D sketch, it is assumed that there is already a 3D model that corresponds to the character in the drawing. Therefore, the *explicit shape model* approach is utilised for solving the problem.

The 2D sketch is an artistic drawing e.g. a storyboard panel. It does not need to be of high quality, but it does need to be accurate in terms of the shape and scale of the object that will be posed.

To create a search space containing the desired 3D pose, an interface is used which allows users to overlay drawings on top of a 3D space (Figure 1), moving the camera

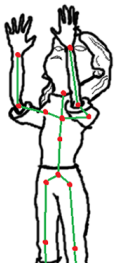

Fig. 3. Joints overlaid by a user on top of a drawing of a human.

Fig. 4. Edge map of a drawing of a human, extracted using the Canny algorithm.

Fig. 5. Silhouette of a drawing of a human.

and placing unposed 3D models. This reduces the search space so that it consists of only local poses of a character, not translation in 3D space (Section 3.2). To compare the 2D to the 3D data in the search space the system extracts information from the sketch. That is when the only user input required - the pinpointing of the joints - takes place (Section 3.3). To navigate the search space and find the desired pose, the system iteratively poses and renders the 3D model of the character to match the drawing (Section 3.2).

The process is not distruptive or encumbering on pipelines, as it is no different from a traditional pipeline with a 3D layout pass. Additionally, drawings exist early on in pre-production in the form of storyboards. The pinpointing of the joints can be done in a few seconds during the clean-up phase and does not require technical training.

### 3.2. PARAC-LOAPSO

As previously mentioned, the problem of inferring the 3D pose from 2D data is treated as a minimisation problem, by navigating a formulated search space using a variation of PSO called PARAC-LOAPSO.

The original PSO algorithm was inspired by the social behaviour of animals who work together in order to explore and find desirable positions in a larger area; the classic example is a flock of birds flying over a landscape looking for food, communicating their findings to each other in order to converge om the position with the most food.

PSO as an optimisation algorithm performs global search and a reason for choosing it as the foundation of PARAC-LOAPSO is that it is parallel by nature, which makes it straight-forward to implement on a GPU [Mussi et al. 2010]. As a meta-heuristic algorithm, it makes no assumptions about the problem, which means it can be general enough to be applied on different models or model controls (e.g. joints). This is important in order to meet the requirement for a general solution, as explained in Section 3.1.

The algorithm consists of a swarm of particles navigating a multi-dimensional search space looking for solutions that get a lower value when evaluated through a fitness function. In the swarm, each particle represents a possible pose. The search space is therefore defined as the set of all possible poses. For more complex models (i.e. models with a greater number of joints), the search space becomes larger.

Even for less complex models the search space of possible poses can be large, even after reducing it by making assumptions. Complicating things further, the problem of pose estimation does not necessarily have just one perfect solution when formulated as a minimisation problem. Local search methods would identify one of the many local minima as the best, but global search methods may be able to perform better given

the multimodal nature of the problem, meaning they are able to find poses which are closer to the optimal solution.

The problem is formulated by defining each particle as a possible pose or *solution*. Specifically, each solution is a vector $X = (j_1, ..., j_n)$, where $n$ is the number of joints to manipulate on the model. Each element $j$ represents a joint and is a three dimensional vector $j = (x, y, z)$ where $x$, $y$ and $z$ are its Euler angle rotations in local space. Therefore, the search space contains all the possible poses for that model.

While issues of uncertain convergence, speed and getting stuck at local minima have been explored [Bratton and Kennedy 2007], it is important to re-examine them for this particular problem.

**Improved Search:** To deal with slower convergence, a constriction factor $K$ (equation 1) [Bratton and Kennedy 2007] is preferred over the inertia weight used by Ivekovic et al. [Ivekovic et al. 2008] or Salehizadeh et al. [Salehizadeh et al. 2009]. To speed up convergence and overall runtime, the algorithm is implemented to run in parallel on the GPU. To deal with getting stuck at local minima, the method proposed by Salehizadeh et al. is used to add variation in the population [Salehizadeh et al. 2009]. Additionally, the optimisation process is faster and more accurate as a result of navigating a smaller search space. That is achieved by using joints' rotation limits from the given 3D model as constraints. Instead of using biological data about human joint limits, only the joint limit data from the 3D model are acquired. Any hierarchical assumptions would decrease the method's versatility.

$$K = \frac{2}{|2 - a - \sqrt{a^2 - 4a}|} \tag{1}$$

$$a = \phi_1 + \phi_2 \tag{2}$$

The cognitive (influence by own findings) and social (influence by others' findings) weights are defined as $\phi_1$ and $\phi_2$.

The balance between exploration and exploitation is based on the idea that for each particle $i$, larger changes in its position $X_i$ are preferred earlier during the optimisation process, in order to examine larger areas of the search space. Towards the end of the optimisation, smaller changes in $X_i$ are preferred in order to refine the solution as much as possible.

When it comes to improving search and ensuring convergence, another important parameter is the communication topology or neighbourhood of the swarm. There are two types of topologies, the global and the local. The global topology means all particles exchange information with all other particles and can update the global best solution found at each iteration step. The local topology means any non-global topology. In this paper, local topology refers to the ring topology, where each particle has two other particles in its neighbourhood.

The global topology converges faster than the local topology and this is often why it is preferred by researchers. It is important to ensure convergence of the swarm, but if the particles are converging towards a local minimum and not the global minimum, then this quick convergence results in a sub-optimal solution. For specific problem sets it has been shown that the additional time it takes for the local topology to converge also results in a solution closer to the global minimum [Bratton and Kennedy 2007].

When the search space is simple, like in unimodal problems, Bratton and Kennedy [Bratton and Kennedy 2007] found that the global topology is better than any local topologies. However, when it comes to multimodal problems, they suggest that a local topology might be better in order to avoid local minima.

Even for problems where the local topology is better, the optimisation needs to run for more epochs in order for the improved results to show. Depending on how many op-

erations occur per epoch, this can take its toll on the runtime of the algorithm, depending on the swarm population. Bratton and Kennedy recommend that both topologies are always tested and this is the case for PARAC-LOAPSO.

**Initialisation:** The overall performance of the algorithm is affected by its initialisation. Since no temporal data is assumed, as would be in the case of a video, a previous pose cannot be used as an initial point. Therefore, each element $j_n$ of position $X_i$ of each particle $i$ is initialised from a uniform random distribution: $j_n \sim U(X_{min}, X_{max})$. The upper and lower boundaries $X_{min}, X_{max} \in [-360, 360]$ differ for each element of $X_i$, since the joint rotation limits are used to set them for each particle element.

Given the link between the drawing and the 3D scene, an alternative initialisation approach for the particles would be to use inverse kinematics (IK) handles to match the approximate position of the joints as pinpointed on the sketch. Then the swarm would be initialised based on a random distribution centred on the IK solution pose. It was decided to not use this approach because it relies on the assumption that the pre-existing 3D model contains IK handles, which in turn would make the proposed method less general in application.

In cases where temporal data exists, like for example in an animation sequence where each frame follows logically from the previous one (Figure 10), the system can easily be configured to initialise each particle randomly in a distribution centred around the previous frame. For example, given a successfully posed model for a frame at the beginning of the sequence, the frame which follows it will have a comparatively similar pose. As such, automatically posing models in an animation sequence with frequent key-framing contains more information about the problem and may benefit from better initialisation. It is also possible to create a second optimisation pass, after the first, where each estimated pose is further corrected based on the previous and the subsequent frames.

**Fitness evaluation:** After the initialisation step a fitness function $f$ allows for the evaluation of each particle. Based on this, the personal best position $B_i^t$ of each particle $i$ and the global best position $B_g^t$ from all particles are stored.

$$f = w_1 D_j + w_2 D_s + w_3 D_e \tag{3}$$

where $D_j$ is a distance metric for the joint positions, $D_s$ one for the shape silhouette and $D_e$ for the edges (both internal and external), while $w_1, w_2$ and $w_3$ are the respective weights for each component (e.g. $\frac{1}{3}$ for equal weighting of all parts).

**Update:** At the beginning of epoch $t$, the population of $N$ particles is split into two subgroups of size $\gamma N$ and $(1-\gamma)N$, where $\gamma = r, r \sim U(0,1)$ if $t < T_c t_{max}$, or 1 otherwise. $T_c$ is denoted as the threshold of convergence, which controls how many epochs are dedicated to exploration and how many to exploitation. For example, for $T_c = \frac{3}{4}$ only the last quarter of the epochs will be dedicated to exploitation. $t_{max}$ is the maximum number of epochs for a run.

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{4}$$

For every particle $i$, its position $X_i$ is updated (equation 4). If the particle belongs to the first population subgroup, its velocity component $V_i^{t+1}$ is updated using equation 5. Otherwise, to attempt and avoid local minima by adding variation, it is updated using equation 6.

$$V_i^{t+1} = K[V_i^t + c_1(B_i^t - X_i^{t+1}) + c_2(B_g^t - X_i^{t+1})] \tag{5}$$

$$V_i^{t+1} = K\{V_i^t - L^t[r_1(B_i^t - X_i^{t+1}) + r_2(B_g^t - X_i^{t+1})]\} \tag{6}$$

$$L^t = 2 - \frac{2t}{t_{max}} \tag{7}$$

$$r_1, r_2 \sim U(0,1) \tag{8}$$
$$c_1 = \phi_1 r_1 \tag{9}$$
$$c_2 = \phi_2 r_2 \tag{10}$$

**Algorithm:** The above components are combined in the following steps to form the overall algorithm.

(1) Initialise the swarm population, with global best $B_g^t$ and/or neighbourhood bests (depending on topology).
(2) For each particle $i$, with personal best $B_i^t$ (in parallel):
  (a) Update particle position (equation 4).
  (b) Update descriptors of 3D pose (Section 3.3).
  (c) Evaluate fitness of particle (equation 3).
  (d) If current fitness ¡ $B_i^t$, set $B_i^t$ to current fitness.
  (e) If current fitness ¡ $B_g^t$, set $B_g^t$ to current fitness.

### 3.3. Comparing Drawings to Renders

The main issue when trying to infer a 3D pose from a 2D drawing or image, is that the problem is under-constrained: there is not enough information in the 2D drawing to get a perfect estimation of the pose. The results show that a combination of various descriptors is able to provide enough information to infer the 3D poses.

Three ways to compare the drawing with the 3D render were chosen given their previous reported advantages and disadvantages. Joints are used by Jain et al. in order to find close poses from a database. In order to make use of the internal edges, edge maps are simple, fast and as effective [Tresadern and Reid 2007] as more expensive methods like Shape Context Histograms [Agarwal and Triggs 2006]. Finally, silhouettes have been used to find poses generatively [Ivekovic et al. 2008] or to train models [Agarwal and Triggs 2004].

**Joints:** The user overlays the positions of the joints on top of the drawing (Figure 3). The position of the overlaid joints on the drawing is compared to that of the 3D positions of the joints, transformed into screen space coordinates. This process differs from the work from Jain et al. in that it is not contextual. This is because the orientation of the pose has to match the drawing in order for the other two comparison methods to work.

The metric for comparing joint positions between a drawing and a projection is the sum of the Euclidean distance between a joint in a drawing and the respective joint in the render (e.g. the wrist in the drawing and the wrist of the rendered 3D model). The sum can then be normalised in order to be a value between 0 and 1, where 0 means the joints are matching perfectly.

**Edge map:** To use internal lines in the drawing, the Canny algorithm [Canny 1986] is used to extract a binary map of the external and internal edges (Figure 4). Then the binary map of the drawing is compared with that of the rendered image.

For a rendered potential pose, a number of points are sampled from its edge map. Then for each sampled point it is checked whether the same point exists in the edge map of the drawing. Therefore for a list of sampled points $S$, the edge map difference or distance $D_e$ is calculated:

$$D_e = 1 - (\frac{1}{|S|} * \sum_{i}^{|S|} p_i) \tag{11}$$

where $p_i$ is the value of the edge map at the $i$th pixel. The value of this calculation is normalised between 0 and 1, where 0 is a perfect match. It is worth noting that this

metric favours cases where the edges of the projection are as small as possible and can therefore give inappropriate results if used by itself.

**Silhouette:** Finally, for the overall shape, the silhouette of the drawing is used as a binary map (Figure 5). The *silhouette distance* between the rendered image and the drawing can be calculated [Fleischmann et al. 2012] (Figure 6).

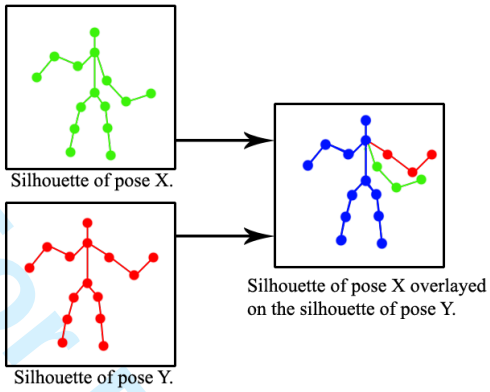


Fig. 6. Example of silhouette comparison to calculate similarity between two poses X and Y. Green colour signifies the silhouette of pose X, red colour the silhouette of pose Y and blue colour the overlap between the two poses.

Similarly to the edge map comparison, it may favour poses where the 3D model makes itself as small as possible (in order to fit into the silhouette).

## 4. RESULTS

The system was tested on three problems of different dimensionality and difficulty: a horse (Figure 7), a human (Figure 8) and a lamp (Figure 9). The subjects were chosen in order to evaluate whether the proposed method is suitable irrespective of the problem. The lamp and the horse models would be difficult to be posed via methods such as motion capture, especially the horse. Most motion capture solutions focus on humanoids, but since they are often not affordable for smaller studios, posing a human is an important test.

All three 3D models are setup as kinematic hierarchical chains of joints manipulated through rotation. The lamp has 6 joints, the human 20 joints and the horse over 25 joints. The horse was however manipulated through 10 controllers set in Autodesk Maya.

Apart from posing models in isolation, the proposed method is evaluated by posing a lamp within a scene using an existing storyboard drawing (Figure 1), in order to test the feasibility of the proposed method for posing models within a complex scene. Moreover, the method is also tested in an animation sequence of a lamp, where temporal coherence between the frames exists. The test was designed to analyse possible discontinuities created in the sequence if the posing is inaccurate.

In an actual drawing, such as a storyboard, multiple characters may appear and may even overlap one another. In a complex drawing where multiple characters may be overlapping, separating their individual lines constitutes an additional problem. The system needs to calculate metrics as described in Section 3.3 for each character separately, thus making character segmentation necessary.

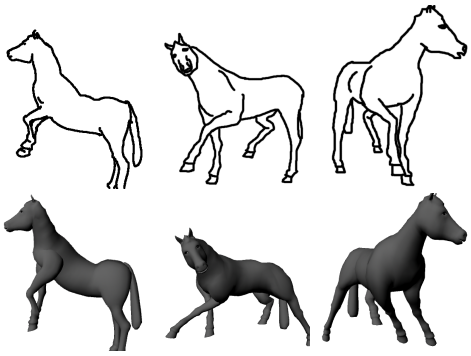

Fig. 7. Side by side comparison of the drawings (top) and the estimated 3D pose (bottom) for the horse model.

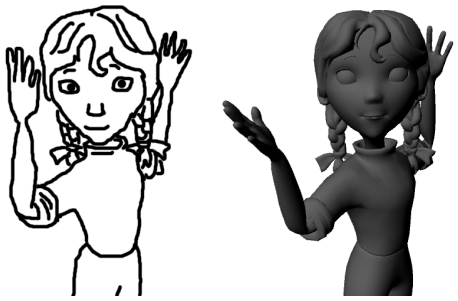

Fig. 8. Side by side comparison of a drawing (left) and the estimated 3D pose (right) for the human model. The estimated 3D pose is close to the drawing.

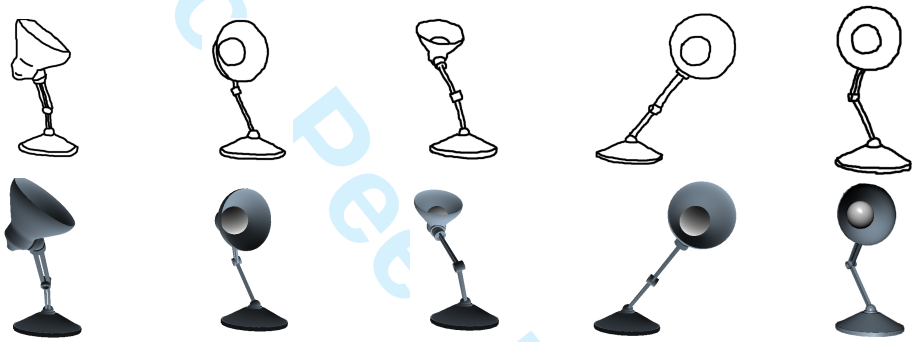

Fig. 9. Side by side comparison of the drawings (top) and the estimated 3D pose (bottom) for the lamp model.

In order to do this, two approaches were considered. The first approach was to perform the drawing process with layers, where the lines of each character are in a separate layer. The second approach was to add an additional 'highlighting' stage which is performed together with the pinpointing of joints. This allows the user to quickly trace the outline of the character to be posed.

With the first approach of drawing in different layers, the artist who performs the creative task of drawing is encumbered by a different work-flow. This is not the case with the second approach, as both the pinpointing of joints and tracing of character outline is performed by a non-skilled user in a separate pass. Since the aim is to reduce the time skilled artists spend on non-creative tasks, the second approach was preferred and thus employed for these tests.

For all the tests, the algorithm was ran for $t_{max} = 1000$ epochs. The values $\phi_1$ and $\phi_2$ as defined in Section 3, were set to a value of 2.05, which is the default value in the canonical PSO [Bratton and Kennedy 2007] and ensures convergence of the swarm. The minimum and maximum velocity of each particle were set to -360.0 and 360.0 respectively, since this is the space of possible rotations. This means that in one epoch, a joint can rotate at most by one complete rotation either clockwise or counter-clockwise. The minimum and maximum position of each particle was set dynamically based on the angle rotation limits of each joint of the model, reducing the search space of possible poses. The exploration phase (Section 3.2), was set to $T_c = \frac{3}{4}$, so the first

$t_{max}T_c = 750$ epochs were dedicated to exploration and the last quarter of epochs was focused on exploitation.

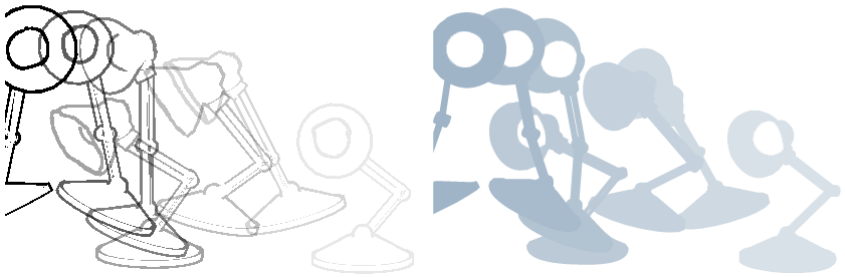

Fig. 10. Side by side comparison of a sequence of drawings (left) with temporal relationships and a sequence from the estimated 3D poses (right) for the lamp model. Lighter opacity and colours signify frames earlier in the sequence while darker opacity and colours signify frames later in the sequence.

A runtime comparison of the algorithm on different hardware can be seen in Table I. All tests except the ones under the "Discrete GPU" column were performed on a system with 8GB of RAM, an Intel HD Graphics 3000 GPU and an Intel Core i5-2520M 2.50GHz CPU, unless stated otherwise. A run of 1000 epochs with a swarm size of 50 particles took approximately 81 seconds on average. The runtime tests used all three methods of comparing the drawing to the 3D render but with varying population sizes to examine the scalability of the algorithm. The discrete GPU tests were performed on a system with 8GB of RAM, an Intel Core i5 2.5GHz CPU and an NVIDIA GeForce GTX 260 GPU.

## 5. DISCUSSION

The proposed method has been evaluated by posing a lamp, a horse and a human from both simple and complex drawings as part of an actual production scene. It has also been tested on a sequence of drawings. The results show that the proposed method is a general approach that is able to perform accurately on different problems and is not constrained to one type of character.

It is important to note that this method does not aim to produce final animated output. It aims at automating the initial pose layout phase or pre-visualisation, which is currently performed manually by skilled operators.

A swarm with global topology was preferred for these tests to ensure timely convergence of the swarm, since all tests were limited to 1000 epochs. However, when the optimisation ran for longer than 1000 epochs, the local topology seemed to produce better results. This difference in performance was more pronounced for complex models like the human.

The simple lamp model was successfully posed in most tests, as was the horse. However, the estimated human model pose was not always as accurate. It is likely that for more complex models, better results may be obtained by running the optimisation for more epochs, with a larger swarm and using a local topology.

The choice of method for comparing drawings to renders may have a notable effect on the end result. For example, by not using the internal lines of a drawing, the forward-

backward ambiguity problem can become an issue as seen in Figure 2. On the other hand, by only using joint locations or the silhouette, run-times are shorter.

Table I. Runtime comparison for various PARAC-LOAPSO swarm sizes, on different hardware as well as a non-parallel implementation on the CPU.

| Particle count | CPU | GPU | Non-parallel CPU | Discrete GPU |
|---|---|---|---|---|
| 1 | 2 min. 13 sec. | 20 sec. | 2 min. 8 sec. | 10 sec. |
| 50 | 2 hour 30 min. | 1 min. 21 sec. | $>$ 4 hours | 1 min. 3 sec. |
| 100 | $>$ 4 hours | 2 min. 15 sec. | $>$ 4 hours | 1 min. 15 sec. |
| 500 | $>$ 4 hours | 6 min. 44 sec. | $>$ 4 hours | 3 min. 31 sec. |
| 1000 | $>$ 4 hours | 12 min 37 sec. | $>$ 4 hours | 6 min. 18 sec. |

The method proposed in this paper easily fits into traditional animation pipelines. Unlike Jain et al. [Jain et al. 2009], it does not require pre-existing data and can use perspective and orthographic camera models. The user input involved in pinpointing the joints can take place as part of the clean up phase. The system is flexible and applicable to a broad range of models, from objects like lamps, to quadrupeds like horses. Additionally, while information such as joint limits help improve the results, it is by no means a necessity for the algorithm to run. This information usually comes together with the actual 3D model. It is not even necessary to work with skeletons and joints or any type of hierarchy; any type of control may be used instead, according to the preferences of the artist who created the model.

However, it is this flexibility which leads to the main disadvantage of this method. Since the approach is generative, it requires a rendering step in every iteration, which may mean the time taken to return a result is longer. It is important to note that the time to return a result is computational only, meaning that this method is still able to contribute to reducing costs. Furthermore, implementing the method on a GPU reduces the effects of this disadvantage significantly.

The runtime tests (Table I) show that PARAC-LOAPSO scales very well on better hardware. It is worth noting that a GPU implementation is much faster compared to a threaded CPU implementation. Moreover, while a swarm with more than 100 particles is very impractical to run on the CPU, a GPU can handle it and return results in under 15 minutes, even on onboard GPU hardware. When using a discrete GPU, the runtimes are approximately halved. These measurements show a trend which suggests that the algorithm can scale very gracefully and return results faster as hardware with parallel capabilities improves.

Since the PARAC-LOAPSO algorithm is stochastic, result accuracy may vary between runs. However, a system with a powerful graphics card can normalise the results through the use of a large population of particles, which helps both in searching as well as in having a more varied initialisation.

The canonical PSO recommends a population of 50 particles [Bratton and Kennedy 2007]. Swarms of 5, 15, 25 and 50 particles were used early on during testing and the best results as presented in this paper were output by the swarm of 50 particles. This hints that larger populations may perform better on this problem.

Having tested the runtimes of PARAC-LOAPSO on different hardware, the results are encouraging in terms of running tests for thousands more epochs with swarm populations in the thousands and local topologies, while modern GPUs handle the computational downsides that would normally arise from these parameters. In effect, the scaling nature of the algorithm means that better hardware will always allow for better results.

### 5.1. Future Work

Future work will primarily focus on formally evaluating the method using more examples as well as comparing it with other state-of-the-art methods in the field.

It is worth examining more general or accurate descriptors and methods of comparing drawings to renders, such as a context-based approach on joints [Jain et al. 2009]. Completely removing the need for user input may be possible by using medial axis methods [Abeysinghe et al. 2008] to extract the curve skeleton automatically from the drawing and then fit the character skeleton to the curve skeleton.

A hybrid between the current optimisation approach and a data-driven approach such as Jain et al. [Jain et al. 2009] is another area where future work may expand to, to get results faster while remaining more general than a pure database method. This would be particularly effective when it comes to humanoids, since extensive databases can be obtained by using the vast array of cheap motion capture devices, like Microsoft Kinect [Microsoft 2014]. Moreover, poses that have already been found in previous attempts can be stored in a database and be used as an initialisation point for the PARAC-LOAPSO run.

Another interesting research direction is to examine how accurate an initial 3D pose needs to be in order to be considered helpful by a professional animator. This may lead to a new benchmark for evaluating automatic 3D posing, not only in terms of accuracy, but also in terms of usefulness.

Dealing with traditional cartoon exaggeration techniques such as squash and stretch is another interesting venture for the future. Assuming that a model is rigged to allow for squash and stretch deformations, then these rig controllers could be directly used to match the drawing.

## 6. CONCLUSION

This paper proposes a method to automatically pose 3D models using information from monocular drawings. This approach is generative and deals with inferring a 3D pose as an optimisation problem. The results show that it is general enough to deal with many types of characters. A lamp, a horse and a human model are posed in different scenarios as evaluation. Moreover, it does not require changes in the pipeline to accommodate it. Minimal user input is still required to pinpoint the joints on the drawing, but this is a task that can be quickly performed by an unskilled operator.

The focus of the proposed method is to pose models for the pose layout phase or pre-visualisation, not final animated output. It may serve as a direct link between storyboarding and pose layout phases of a pipeline.

## ACKNOWLEDGMENTS

Many thanks to Keith Pang for his valuable help and insight throughout this research. Thanks also go to the Technology Strategy Board (TSB), for partially funding this research project.

## REFERENCES

Sasakthi S. Abeysinghe, Matthew Baker, Wah Chiu, and Tao Ju. 2008. Segmentation-free skeletonization of grayscale volumes for shape understanding. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*. 63–71. DOI:http://dx.doi.org/10.1109/SMI.2008.4547951

Ankur Agarwal and Bill Triggs. 2004. Learning to track 3D human motion from silhouettes. In *In International Conference on Machine Learning*. 9–16.

Ankur Agarwal and Bill Triggs. 2006. Recovering 3D human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28, 1 (Jan 2006), 44–58. DOI:http://dx.doi.org/10.1109/TPAMI.2006.21

Dan Bratton and James Kennedy. 2007. Defining a Standard for Particle Swarm Optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. 120–127. DOI:http://dx.doi.org/10.1109/SIS.2007.368035

Nestor Burtnyk and Marceli Wein. 1976. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Commun. ACM* 19, 10 (Oct. 1976), 564–569. DOI:http://dx.doi.org/10.1145/360349.360357

John Canny. 1986. A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8, 6 (Nov 1986), 679–698. DOI:http://dx.doi.org/10.1109/TPAMI.1986.4767851

James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. 2003. A Sketching Interface for Articulated Figure Animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 320–328. http://dl.acm.org/citation.cfm?id=846276.846322

Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

Laurent Favreau, Lionel Reveret, Christine Depraz, and Marie-Paule Cani. 2006. Animal Gaits from Video: Comparative Studies. *Graph. Models* 68, 2 (March 2006), 212–234. DOI:http://dx.doi.org/10.1016/j.gmod.2005.04.002

Patrick Fleischmann, Ivar Austvoll, and Bogdan Kwolek. 2012. Particle Swarm Optimization with Soft Search Space Partitioning for Video-Based Markerless Pose Tracking. In *Advanced Concepts for Intelligent Vision Systems*, Jacques Blanc-Talon, Wilfried Philips, Dan Popescu, Paul Scheunders, and Pavel Zemcik (Eds.). Lecture Notes in Computer Science, Vol. 7517. Springer Berlin Heidelberg, 479–490. DOI:http://dx.doi.org/10.1007/978-3-642-33140-4_42

Dariu Gavrila. 1999. The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding* 73 (1999), 82–98. Issue 1. DOI:http://dx.doi.org/10.1006/cviu.1998.0716

Martin Guay, Marie-Paule Cani, and Rémi Ronfard. 2013. The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph.* 32, 6, Article 205 (Nov. 2013), 8 pages. DOI:http://dx.doi.org/10.1145/2508363.2508397

Gang Hua, Ming-Hsuan Yang, and Ying Wu. 2005. Learning to estimate human pose with data driven belief propagation. In *CVPR*. 747–754.

Spela Ivekovic, Emanuele Trucco, and Yvan R. Petillot. 2008. Human Body Pose Estimation with Particle Swarm Optimisation. *Evolutionary Computation* 16 (2008), 509–528. Issue 4. DOI:http://dx.doi.org/10.1162/evco.2008.16.4.509

Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilleges, and Olga Sorkine-Horning. 2014. Tangible and Modular Input Device for Character Articulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 33, 4 (2014).

Eakta Jain, Yaser Sheikh, and Jessica K. Hodgins. 2009. Leveraging the talent of hand animators to create three-dimensional animation. In *Symposium on Computer Animation*. 93–102. DOI:http://dx.doi.org/10.1145/1599470.1599483

Eakta Jain, Yaser Sheikh, Moshe Mahler, and Jessica Hodgins. 2012. Three-dimensional proxies for hand-drawn characters. *ACM Trans. Graph.* 31, 1, Article 8 (Feb. 2012), 16 pages.

Yin Li, Michael Gleicher, Ying-Qing Xu, and Heung-Yeung Shum. 2003. Stylizing Motion with Drawings. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 309–319. http://dl.acm.org/citation.cfm?id=846276.846320

Chen Mao, Sheng Feng Qin, and David Wright. 2006. Sketching-out virtual humans: From 2d storyboarding to immediate 3d character animation. In *Proceedings of ACM SIGCHI International Conference On Advances In Computer Entertainment Technology*. ACM.

Chen Mao, Sheng Feng Qin, and David Wright. 2007. Sketch-Based Virtual Human Modelling and Animation. In *Proceedings of the 8th International Symposium on Smart Graphics (SG '07)*. Springer-Verlag, Berlin, Heidelberg, 220–223. DOI:http://dx.doi.org/10.1007/978-3-540-73214-3_24

Microsoft. 2014. Kinect. (January 2014). http://www.xbox.com/en-us/kinect

Luca Mussi, Spela Ivekovic, and Stefano Cagnoni. 2010. Markerless Articulated Human Body Tracking from Multi-view Video with GPU-PSO. In *Proceedings of the 9th International Conference on Evolvable Systems: From Biology to Hardware (ICES'10)*. Springer-Verlag, Berlin, Heidelberg, 97–108. http://dl.acm.org/citation.cfm?id=1885332.1885344

Cengiz A. Öztireli, Ilya Baran, Tiberiu Popa, Boris Dalstein, Robert W. Sumner, and Markus Gross. 2013. Differential Blending for Expressive Sketch-Based Posing. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. ACM, New York, NY, USA.

Amin S. M. Salehizadeh, Peyman Yadmellat, and Mohammad-Bagher Menhaj. 2009. Local Optima Avoidable Particle Swarm Optimization. In *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*. 16–21. DOI:http://dx.doi.org/10.1109/SIS.2009.4937839

Phil Tresadern and Ian Reid. 2007. An Evaluation of Shape Descriptors for Image Retrieval in Human Pose Estimation. In *Proc. 18th British Machine Vision Conf., Warwick*, Vol. 2. 800–809.

Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: Sketch-based Co-retrieval and Co-placement of 3D Models. *ACM Transactions on Graphics* 32, 4 (2013), 123:1–123:12.

Rong Yang and Burkhard C. Wunsche. 2010. LifeSketch - A Framework for Sketch-Based Modelling and Animation of 3D Objects. In *in Proceedings of the Australasian User Interface Conference (AUIC)*. 2010.

Jianmin Zhao and Norman Badler. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics* 13 (1994), 313–336.

Received January 2015; revised N/A; accepted N/A