EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2010)
M. Alexa and E. Do (Editors)

A Sketching Interface for Sitting-Pose Design

Juncong Lin[1] Takeo Igarashi[1,2] Jun Mitani[1,3] Greg Saul [1]

[1] JST, ERATO, IGARASHI Design Interface Project, 7th floor, 1-28-1, Koishikawa, Bunkyo-ku, Tokyo 112-002, Japan
[2] The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0003, Japan
[3] University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-0005, Japan

Abstract

We present a sketch interface for interactively placing a 3D human character in a sitting position on a chair. The user first sketches the target pose as a 2D stick figure. The user can specify whether a joint will be attached to the environment (for example, the feet may be put on the ground) with a pin tool. Our system then reconstructs the 3D pose from the sketch figure considering the constraints specified by the user and the interaction between the character and the environment. This paper presents a user interface and a reconstruction algorithm that combines a genetic algorithm and a quasi-Newton solver to efficiently find a collision-free pose. An informal user study showed the effectiveness of our approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.4 [Computer Graphics]: Graphics Utilities—Graphics editors

1. Introduction

Manual design of a character pose is one of the most basic processes in computer graphics authoring. Although many advanced methods are available today, manual character-pose design is mostly done using direct control of joint angles and inverse kinematics. Sketching interfaces have also been applied to character-pose design to make the design process easier and faster. However, most of these design tools only consider the character body structure and do not consider interaction with the environment (leaning on a wall, sitting on a chair, stepping on a stair, and so on). When posing the character, great care must be taken to avoid collisions with the environment. Our goal is to provide a user interface to quickly and easily set a character's pose while considering interaction with the environment.

In this paper, we focus on the design of a sitting pose as an example of interaction with the environment. Our primary target application is furniture design, in which we usually need to examine ergonomic and structural concerns. For example, when designing a chair, it must be the appropriate height to allow one's feet to rest on the ground. The chair must remain stable even if the person sits on it in a strange way. We believe that our interface is also useful for general computer graphics production in which character poses are manually specified.

In our system, the design process starts with the sketching operation. A user draws a sketch figure on the screen plane to specify the desired sitting pose. The sketch figure consists of circular dots, representing joints, connected by lines. The user can specify whether a joint is attached to the environment using the pin tool (for example, the feet of a sitting character are usually pinned to the floor). The system then reconstructs a reasonable pose from the input sketches and environmental constraints. Figure 1 shows examples of sitting poses generated by the system based on various sketches and environments.

There are two challenges to providing such a system: First, there are usually multiple 3D poses consistent with a single 2D sketch figure; and second, there may be many interactions between a character and a chair. These two problems are not unrelated. In this paper, we demonstrate how exploiting interactions between the character and the chair can eliminate some unexpected poses.

The contribution of this paper is a holistic system for interactively placing a character in a sitting position on a chair. The system has an intuitive interface to allow non-professional users to easily solve this type of pose-design problem. The core of the system is a technique that reconstructs a collision-free pose from a 2D input sketch figure. We formulate it as an optimization problem and design a

DOI: 10.2312/SBM/SBM10/111-118

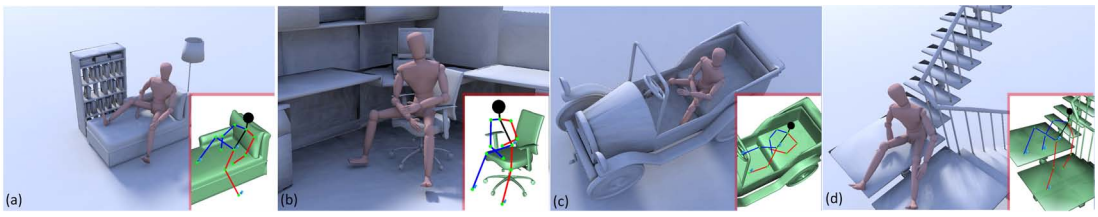

Figure 1: *Various sitting poses designed with our system. The user specifies the desired pose with sketches on 2D canvas (inset of each subfigure), and our system generates the 3D pose.*

hybrid solver for the problem. Our solver consists of two components: a genetic algorithm-based solver (G-A solver) is used to compute a collision-free pose, and then, a quasi-Newton procedure (Q-N solver) is applied to the pose to refine the result.

2. Related Work

Inverse kinematics (IK) is one of the most important techniques for generating character poses satisfying given kinematic constraints [ZB94]. The well-known difficulty with the IK technique is that the problem is almost always underdetermined. Attempts have been made to restrict the solution space [YN03, BR04, GMHP04].

The interface typically adopted by existing character-posing systems allows the user to interactively position the end effectors of a character, and the interior joints are updated by solving the non-linear IK problem. Such an interface require the user to manipulate 3D widgets, and this may make it difficult to obtain the desired pose. Several attempts have been made to reduce the difficulty with a sketch interface [RK92, DAC*03]. These methods try to reconstruct a 3D pose from an artist-drawn 2D sketch figure. We adopt a similar sketch interface as [DAC*03]. However, we emphasis more on the interaction between the character and the environment which is discussed less in [DAC*03]. Besides, we formulate the reconstruction of 3D pose from 2D sketch as an optimization problem combining sketch constraints and physical constraints while [DAC*03] adopts the reconstruction method of [Tay00] which considers the foreshortening of each body segments in a orthographic camera configuration.

Our work is also closely related to the reconstruction of a 3D pose from a monocular image. Due to insufficient spatial information, the problem is inherently ill-posed. Most existing works focus on using various domain constraints to solve the underconstrained depth ambiguity problem. Model-based techniques are a common strategy [Tay00]. Another approach to the pose-reconstruction problem is to use probabilistic techniques [AT04, AC04] to automatically learn the mapping between 2D image features and 3D poses. The most prominent feature differentiating our system from previous methods is the consideration of interaction between character and environment. Also, our input is a hand-drawn user sketch, which is usually imprecise and needs special treatment.

3. User Interface

In a typical design session, a user first sketches the elements of the desired pose on the screen space (Figure 2(a)) in arbitrary order. The system then analyzes the sketch and organizes it into a tree structure similar to the character's skeleton. The user then checks the sketch structure and switches the left and right limbs if necessary to resolve ambiguities due to the symmetry of the skeleton (Figure 2(b)). Red links represent the left side of the body (left upper limb and left lower limb), and blue links represent the right side. The user can also use the pin tool to specify where joints are attached to the environment (Figure 2(c)). For example, the user can tell the system to constrain the feet to the ground, to put the hand on the armrest, and so on. The 3D pose is generated by clicking the 3D button (Figure 2(d)).

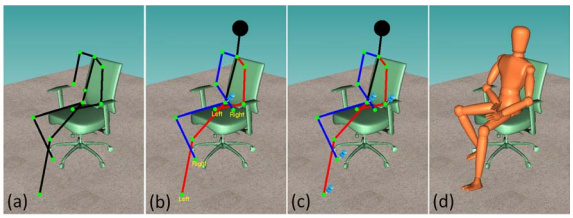

Figure 2: *Sitting pose design paradigm: (a) user sketch of the expected pose on 2D canvas; (b) user can switch the left and right limbs to resolve symmetric ambiguities; (c) user can specify where joints are attached to the environment with the pin tool; (d) the 3D pose is generated by clicking the attach button.*

3.1. Sketching Tool

We use a sketching interface similar to [DAC*03], in which the user draws straight limb segments by dragging, rather than drawing freeform. The user clicks to specify the position of the first joint and then drags and releases the cursor

at the desired position of the second joint. The process is repeated until the whole skeleton has been specified. A newly sketched limb will automatically snap to an existing one if its joint is close enough to the joints of the previous limb. The snap function makes it easy for the user to ensure that segments connect to one another properly. The user sketches elements of the pose in an arbitrary order. Camera rotation and translation are disabled during sketching.

3.2. Pin Tool

The pin tool is used to specify a target position in the environment to which a joint is to attach. The pin tool can also be used to attach a joint to a limb of the character. To pin a joint, the user simply clicks on the sketch node. When selected, the pin tool attaches the joint to another limb when they appear close together in the sketch (e.g., forearm on knee). Otherwise, the joint is attached to the environment at the joint's location (feet on the floor). The pelvis joint is an exception. It can only be pinned to the environment (sitting plane).

4. Technical details

4.1. Human model

We currently use a human model with 29 degrees of freedom (DOF) as shown in Figure 3(a). The human model is represented by a tree of joints ($H = \{\mathbf{j}_k\}(1 \leq k \leq m, m = 20)$) rooted at the pelvis.

The sketch figure (Figure 3(b)) is used to specify the pose in the screen space. It is also organized as a similar tree structure with 13 nodes ($S = \{\mathbf{s}_i\}(1 \leq i \leq n, n = 13)$).

The sketch nodes map to the skeleton joints, $\Phi : \mathbf{s}_i \longrightarrow \mathbf{j}_k$.

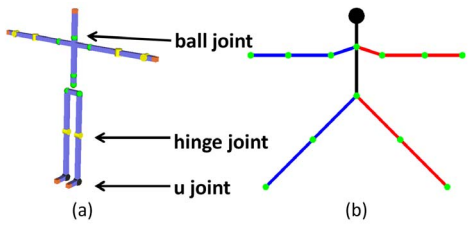

Figure 3: *The human model used in our system: (a) template skeleton of the 3D character; (b) template sketch structure used to specify the pose.*

4.2. Sketch Analysis

The user-drawn sketch is initially defined as an undirected graph. To construct a tree structure from the graph, we first determine the root of the tree, which has three linked graph edges. We also identify the neck node, which has four linked graph edges. We then extract the chains from the root to the leaf nodes (those with only one linked graph edge). We determine the body part that each chain represents according to the number of nodes in the chain. In the template structure, chains representing upper limbs (from pelvis to left/right hands) contain five graph nodes. For both the upper body (from pelvis to the head) and the lower limbs (from pelvis to the left/right feet), the number of nodes is three. We distinguish between the upper body chain and the lower limb chains by checking whether the chain contains the neck node.

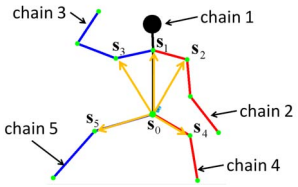

Figure 4: *Symmetric ambiguities in the sketch figure are resolved.*

To define the mapping Φ, we need to further resolve the ambiguities due to the symmetries in the tree structure. We assume that the user draws a front view pose of the character, and we then identify the left and right limbs by checking the following equation (Figure 4):

$$sign = \mathbf{s}_0\mathbf{s}_i \cdot \mathbf{s}_0\mathbf{s}_1 (i \neq 0, 1).$$

Chain i represents a left limb when $sign > 0$. Otherwise, it represents a right limb. We provide a switch tool for users to manually switch limb left/right orientation (Figure 2(b)) when the above method does not work.

4.3. Seat positioning

To reconstruct a pose from a sketch, we first need to locate where the person is seated. As Figure 5(a) shows, it is not obvious how a user will draw a sketch. Some may draw the central line of each limb (red), whereas others may place the sketch joint at the contact between the character and the floor or chair (blue). The algorithm for locating the sitting position highly depends on which strategy the user chooses. Therefore, we conducted an observational study to better understand the way people draw. We showed users examples of sitting poses (Figure 5(b)) and asked them to sketch the pose with our system. Figure 5(c) shows the various sketches collected from the study. We found that most users draw the center line of a character. Therefore, our default system locates the sitting position based on the center-line assumption.

To locate the sitting position, we first calculate the intersection between the extended bounding box and the eye ray $l_e(\mathbf{p}_e, \mathbf{v}_e)$ passing the root joint of the sketch character (Figure 6(a)). We extend the bounding box in an upward direction $\mathbf{v}_u = (0.0, 1.0, 0.0)$ to ensure the intersection with some object, such as the foot stool in Figure 7. We then sample the intersection line segment in the bounding box and emit a

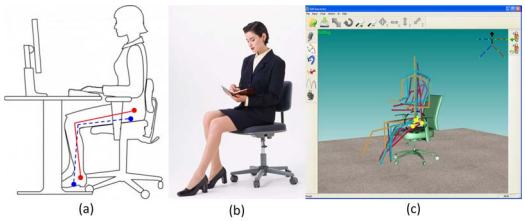

Figure 5: *Observation Study. (a) different sketching styles adopted by users; (b) seated figure shown to users; (c) sketch characters drawn by users.*

ray from each sample in the direction $\mathbf{v}_s = -\mathbf{v}_u$(Figure 6(b)). We collect the intersected faces whose normal satisfies

$$\mathbf{n}_f \cdot \mathbf{v}_u > \sqrt{3}/2 \tag{1}$$

and where the distance to the sample point satisfies

$$s_1 \cdot h \leq d \leq s_2 \cdot h, \tag{2}$$

where h is the hip height, $s_1 = 0.5$ and $s_2 = 1.5$. Using these intersected faces as seed, we can grow several regions by collecting all the faces for which the normal satisfies Equation 1. For each region R_i, we assign a normal $\mathbf{n}_r$ by averaging the face normals over the whole region. We can then generate a cut plane P_i passing through the eye position $\mathbf{p}_e$ and the normal $\mathbf{n}_p = \mathbf{n}_r \times \mathbf{v}_e$. We calculate the uniformly resampled intersection curve $I_i = (\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_m)$ between P_i and R_i, and find the best sitting point of R_i by minimizing the following equation:

$$M(R_i) = \arg\min_k(|d(\mathbf{p}_k, l_e, \mathbf{n}_r)) - h| \cdot C(k,m)). \tag{3}$$

The metric consists of a distance term and a centerness term: The distance term measures the distance between $d(\mathbf{p}_k, l_e, \mathbf{n}_r)$ and the hip height h. Figure 6(d) shows how to calculate $d(\mathbf{p}_k, l_e, \mathbf{n}_r)$; The centerness term $C(k,m) = e^{-\frac{(k-t)}{2(t)^2}}$ (t=m/2) is used to reduce the ambiguity of the user input, and we assume that the user places the character at the center of the seat area. The sitting region R_s is the region with the minimal metric across all regions (Figure 6(c)), and the root position

$$\mathbf{p}_r = \mathbf{p}_{sk} + h \cdot \mathbf{n}_r. \tag{4}$$

In some cases, the hip may collide with the chair under the optimal sitting position. We then search those intersection points $(\mathbf{p}_{k-r} \ldots \mathbf{p}_k \ldots \mathbf{p}_{k+r})$ around the optimal sitting position for a certain range r ($r = 5$ in our current implementation) until we find a collision-free arrangement.

We also provide a scheme for users who prefer to draw the contact points instead of the center lines. The user can choose between the two different sketching styles. For the contact-point method, we simply set $h = 0.0$ in Equation 3 and rewrite Equation 2 as $s_1 \leq d \leq s_2$ with $s_1 = -0.05$ and $s_2 = 0.05$.

4.4. Pose reconstruction

4.4.1. Overview

The core of our character-posing system is the reconstruction of a 3D pose from a 2D sketch figure. We formulate it as an optimization problem and find the 3D pose with minimum energy. The input of the reconstruction algorithm is the 2D positions of the joints in the sketch ($\mathbf{s}_i$), and the output is the joint angles of the reconstructed 3D character pose ($\mathbf{q}_i$). We represent each joint angle with angle vectors (Euler angles) with the dimension of the DOF of the joint. The objective function evaluates the match between the projected 3D pose and the 2D input sketch, the plausibility of the pose, and the distance between the pinned joints and their bases. Collision handling is done separately. We first describe the details of the objective function and then describe how we solve this optimization problem. Finally, we describe how we handle collisions in this framework.

4.4.2. Objective Function

The objective function we use to determine the optimal 3D pose from a 2D sketch is described by the following equation:

$$E = w_p E_p(\mathbf{Q}) + w_b E_b(\mathbf{Q}) + w_a E_a(\mathbf{Q}) \tag{5}$$

$\mathbf{Q}$ is the total joint angle vector:

$$\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_n] \tag{6}$$

which is subject to the range of motion of each joint. $\mathbf{W} = [w_p, w_b, w_a]$ are the weights given to the sub-energies. We currently use $w_p = 1.0$, $w_b = 0.5$, and $w_a = 5.0$.

The primary term of our objective function is the **projection energy** E_p, which measures the consistency between the projection of the reconstructed pose and the 2D sketch figure. We consider both the orientation and the position, with an emphasis on the orientation. Denoting the projection of the corresponding joint as $\mathbf{p}_i$, the orientation energy for sketch node $\mathbf{s}_i$ is defined as:

$$E_{i-ori} = \sum_j^{c_i}(1 - \frac{(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{s}_j - \mathbf{s}_i)}{\| \mathbf{p}_j - \mathbf{p}_i \| \cdot \| (\mathbf{s}_j - \mathbf{s}_i) \|}) \tag{7}$$

where c_i is the set of children nodes of the joint corresponding to sketch node $\mathbf{s}_i$. The position energy is described by the following equation:

$$E_{i-pos} = \| \mathbf{p}_i - \mathbf{s}_i \|^2 \tag{8}$$

The projection energy is then defined as the sum of the energy of all sketch nodes:

$$E_p = \sum_i^n (w_{ori} E_{i-ori} + w_{pos} E_{i-pos}) \tag{9}$$

w_{ori} and w_{pos} are the weights for orientation and position respectively; we currently use $w_{ori} = 1.0$ and $w_{pos} = 0.0003$

We introduce the second term, **balance energy** E_b, to

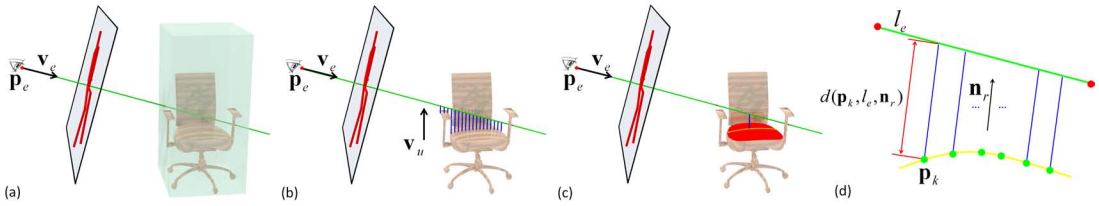

Figure 6: *Locating the sitting position: (a) Intersect the eye ray with the bounding box of the chair; (b) Find all possible sitting areas on the chair; (c) Determine the final sitting position.*

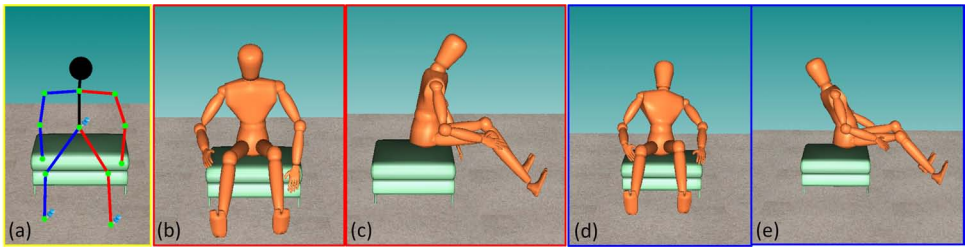

Figure 7: *Balance constraint: (a) sketch figure; (b,c) with balance constraint; (d,e) without balance constraint.*

keep the character balanced, leading to a visually plausible pose (Figure 7). We achieve the balance constraint by forcing the center of mass of the whole body to stay over the supporting polygon [PB91]. The supporting polygon is defined by the set of joints pinned to the environment . We minimize the distance between the ground projection of the character barycenter and the center of the supporting polygon:

$$E_b = \| \mathbf{MC} - \mathbf{c} \|^2 \tag{10}$$

$\mathbf{c}$ is the supporting center, matrix $\mathbf{M} = \mathbf{P} \cdot \mathbf{T}$ calculates the new barycenter of the character and projects it onto the ground, $\mathbf{T} = (m_1\mathbf{T}_1, m_2\mathbf{T}_2, \ldots, m_n\mathbf{T}_n)$($\mathbf{T}_k$s and m_ks are the rigid transform and the mass of each skeleton bone respectively, $m_k s$ are normalized by the overall mass) , $\mathbf{P}$ is the projection matrix, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_h]^T$ are the barycenters of the skeleton bones in standard pose. Both m_ks and $\mathbf{C}$ can be precomputed. We currently only consider explicitly pinned joints in computing the supporting polygon. In reality, limbs can rest on a chair (thigh on seat and back on backrest) and the supporting polygon must take these limbs into account; resolving this aspect remains for our future work. Little weight is assigned to this balance energy, so it is only used when the input sketch is very ambiguous, when it is helpful in eliminating some invalid poses (Figure 7).

We name the third term **attach energy** E_a, which is used to constrain a joint attached to certain place (Figure 8). There are two different attach styles: (1) the user specifies a joint attached to a limb (e.g., hand on knee); and (2) the user specifies a joint attached to the environment (e.g., feet on ground); If the selected sketch node $\mathbf{s}_k$ is very close to a non-neighbor sketch limb $\mathbf{s}_{k0}\mathbf{s}_{k1}$, we treat it as the first case (Figure 8(b)). Otherwise, we generate a eye ray passing through the selected sketch node $\mathbf{s}_k$ in the screen space. Then, we

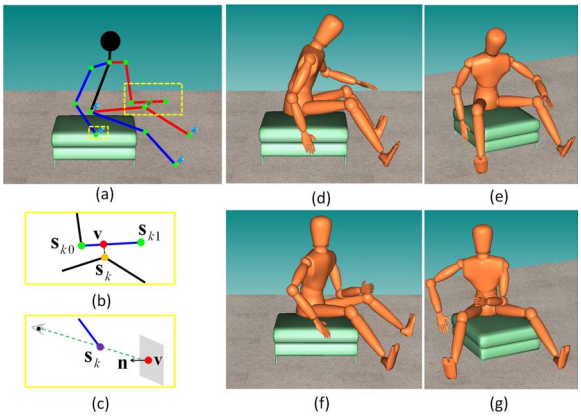

Figure 8: *Attach constraint: (a) sketch figure; (b) attach style-1; (c) attach style-2; (d,e) with attach constraints; (f,g) without attach constraints.*

find the intersection point $\mathbf{v}$ between the ray and the chair or the ground (Figure 8(c)). For the first case, we minimize the distance between the joint $\mathbf{j}_k$ and the closest point on the bone segment $\mathbf{j}_{k0}\mathbf{j}_{k1}$:

$$E_a = \| \mathbf{j}_k - (\mathbf{j}_{k0} \cdot (1-t) + \mathbf{j}_{k1} \cdot t) \|^2 \qquad t = \frac{|\mathbf{s}_{k0}\mathbf{v}|}{|\mathbf{s}_{k0}\mathbf{s}_{k1}|} \tag{11}$$

We define the energy for the second case with the following equation:

$$E_a = ((\mathbf{j}_k - \mathbf{v}) \cdot \mathbf{n})^2 \tag{12}$$

Here, we minimize the distance between the corresponding skeleton joint $\mathbf{j}_k$ and the plane defined by the attach point $\mathbf{v}$ and its normal $\mathbf{n}$ rather than the distance between the two

points directly, as it is difficult to exactly specify the 3D position to which we want the joint to attach. Figure 8 shows the difference between 3D figures generated with and without attach constraints.

4.4.3. Solver pipeline

To minimize the objective function in Equation 5 with some gradient methods such as the quasi-Newton solver, we need to provide a good initial pose. The initial pose should be collision-free, as the gradient solver cannot guarantee that the optimized pose is collision-free, and it should be close enough to the optimal pose to avoid a complex collision problem. Therefore, a hybrid framework is adopted in our system. Figure 9 shows the optimization process pipeline. First, a genetic algorithm-based solver (G-A solver) is used to generate an initial collision-free pose, and then refinement is conducted with a quasi-Newton solver (Q-N solver) to generate the final optimal pose.

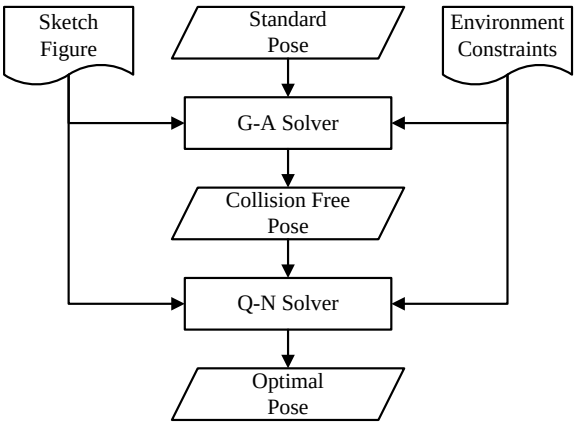

Figure 9: *Hybrid solver for pose reconstruction: the G-A solver is used to generate an appropriate initial pose, and then the pose is optimized by the quasi-Newton solver.*

There has been previous work using genetic algorithms to solve IK problems [TMC09] or pose reconstruction problems [ZLK05]. However, there are few discussions of interaction with the environment. We use the steady-state genetic algorithm provided in GALIB [GAL] with some modifications for collision handling. Figure 10 shows the flow chart of the modified genetic algorithm:

- **Initialization:** A population of size $S_1 = 90$ (P_1) is randomly created. Each individual consists of n joint angles (Equation 6).
- **Fitness value calculation:** The fitness values of each individual is calculated using Equation 5.
- **Evolution:** We use tournament selection to create the parent pool. The better one of two individuals picked from population P_1 using the RouletteWheel selection will be returned. The likeliehood of selection in RouletteWheel is proportionate to the fitness value. A temporary population with size $S_2 = 90$ (P_2) is then created by crossover and mutation of the parent pool. Simulated binary crossover (SBX) [DA95] is adopted for the crossover operation. P_2 is then merged to P_1, and invalid individuals are removed from P_1 according to certain criteria until the size of P_1 is reduced to S_1.
- **Termination criterion:** After each evolution, we check the current generation against the termination criterion. The evolution stops when either the maximum generation number (15 in our current implementation) is exceeded or the deviation of the population is under a certain limit (0.05).

The second component of our framework, a quasi-Newton solver, is then applied to the solution generated by the G-A solver to improve the reconstruction result. We combine Rosen's projection method [Ros60] with our Q-N solver to deal with the inequality constraints from joint range of motion as in [ZB94]. The graident of Equation 5 is computed by numerical difference method.

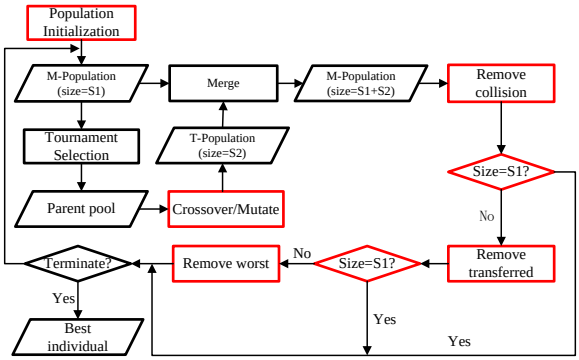

Figure 10: *Genetic algorithm for collision-free pose reconstruction.*

4.4.4. Collision Handling

Collision handling is an important issue in our application. The collision handling scheme in our system consists of two components: (1) the genetic solver provides a collision-free initial configuration for the downstream non-linear optimization solver; and (2) collisions are avoided during the refinement step.

To generate a collision-free pose with the G-A solver, we first run a collision test on each individual during the population initialization. Joint parameters are randomly regenerated for individuals failing the test until they become collision-free or the maximum iteration number ($4 \times n_{cd}$, $n_{cd} = 5$) is reached. A similar strategy is applied on the cross operation and mutation operation for those children that failed the test. During each evolution step, we first remove the individuals who collide with the environment from the merged popularization and then remove the transferred, collision-free ones. Finally the individuals with the worst

score are removed. The red elements in Figure 10 are the modifications we have made.

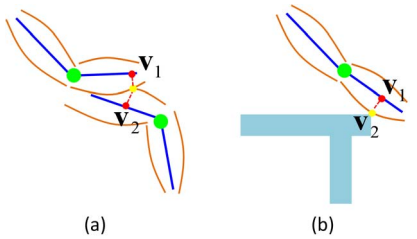

Figure 11: *Definition of collision energy: (a) limb-limb collision energy; (b) limb-environment collision energy.*

When a collision is detected in the refinement step, we introduce a collision energy that tries to keep the two colliding objects at the previous collision-free iteration. For collisions between limbs, we find the two closest points $\mathbf{v}_1$ and $\mathbf{v}_2$ (Figure 11(a)) and try to keep them as in collision-free iteration $k-1$. For limbÚenvironment collisions, $\mathbf{v}_2$ is replaced by the collision point in the environment (Figure 11(b)).

$$E_{collision} = \| (\mathbf{v}_1^{[k]} - \mathbf{v}_2^{[k]}) - (\mathbf{v}_1^{[k-1]} - \mathbf{v}_2^{[k-1]}) \|^2 \qquad (13)$$

The COLDET package [COL] is used for collision detection. The collision handling is very helpful in eliminating unnatural poses. Figure 12 gives an example.

4.4.5. Handling the pose of feet and hands

We omit feet and hands in the sketch structure to reduce the user's burden. We keep the character feet and hands at standard 3D poses by default. We also run collision detection for feet and hands during the optimization process. Once a collision is detected, we search for collision-free orientations in the range of motion and assign them to the foot or hand in question. If no collision-free poses are available, we deal with the collided character as described in Section 4.4.4.

5. Results

We developed a prototype system with C++ on a laptop with an Intel Pentium 2.26-GHz processor. The current algorithm successfully reconstructs plausible 3D poses from various 2D poses. Examples are shown in Figures 1, 2, 7, 8, and 12. It usually takes $1-2$ seconds to produce a 3D reconstruction. Table 1 shows time statistics for these figures. In our own experience, the system sometimes returns undesirable results, but we can quickly obtain a good result by running the 3D reconstruction a few times (the system returns different results each time).

Table 1: *Time Statistics*

Figure	1a	1b	1c	1d	7b	8f	12b
Time(sec)	1.9	1.8	2.1	2.1	2.2	1.7	2.0

An informal user study was conducted to compare our proposed method with the traditional inverse kinematics method. An inverse kinematic method [ZB94] is implemented in our system in which the user designs the pose by manipulating the end effectors. Eight users participated in the study, and 10 minutes were given for each interface to let the users learn and practice pose design. We then asked the users to design a 3D pose in both interfaces according to given pose figures. Then, they were asked to compare the two interfaces using a questionnaire. Figure 14 shows the analysis of data from the post-study questionnaire. Some users pointed out that it was difficult to estimate the length of sketches in current system. They also expected to further edit the stick figure so as to better design the pose. It took about 50 and 90 seconds to design a pose with the sketch interface and inverse kinematics, respectively, for those with some 3D software experiences. For novice users, it took about 70 and 150 seconds, respectively.

We intentionally separated sketching and IK to clarify the strengths and limitations of each. However, in practice, they are complementary. It would be useful to start a design with sketching and then use IK to refine the reconstructed pose.

6. Limitations

Our current implementation has several limitations. We currently consider only collisions and balance as the interaction between environment and the character. In fact, such interaction is usually very complex in reality. For example, we do not consider a limb's capacity to rest on a chair, which may affect the balance constraint. The hip is deformable in reality, and the interaction between hip and chair is very complex. Besides, the determination of supporting polygon totally rely on the user specification with pin tool. It would be expected to automatically detected some extra contact point thus to further reduce user's burdern. To maintain an interactive speed, we currently use a quite small population size for the G-A solver, which is sufficient for a regular sitting pose. However, for some complex cases such as the one in Figure 13 , these parameters may not be able to generate a valid pose.

Figure 13: *Pose cannot be generated with our system.*

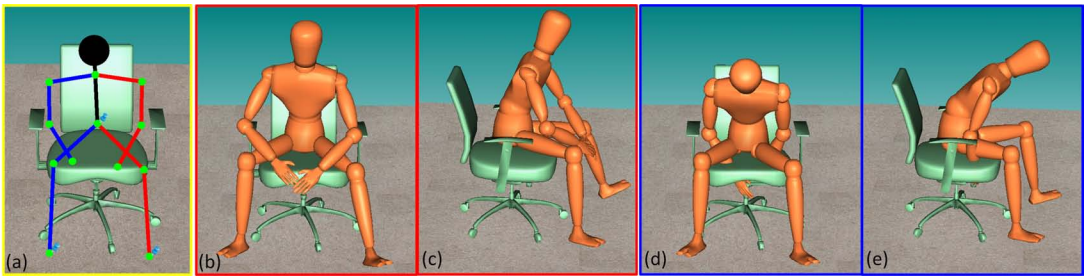

Figure 12: *Collision handling results: (a) sketch figure; (b,c) with collision handling; (d,e) without collision handling.*

7. Conclusion

We present a sketch-based system for the design of sitting poses. With the sketch interface, the user only needs to draw in a 2D canvas to design a pose. Our method is easier than a traditional IK interface because the user does not need to specify the depth of each joint, freeing the user from frequent camera rotation. Our system is novel in its consideration of the interaction between the character and the environment. Our prototype system successfully reconstructs plausible 3D models in under a second, allowing for interactive exploration.

In future work, we will continue to improve our system based on the feedback from the user study and the limitations mentioned above. A first step is to use biomechanics and ergonomics knowledge to constrain the search space and thus improve the pose generated by the solver. We also plan to introduce dynamic simulation to help detect extract contact points for equilibrium computation. We may also explore the possibilities of introducing more modeling operators to edit the stick figures. It would be interesting to combine with multi-touch during the edit of stick figure. We may also provide a suggestive interface to resolve the ambiguity in the pose reconstruction.

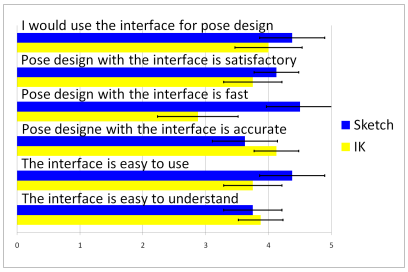

Figure 14: *Results of analysis of post-study questionnaire. The rating scales at the bottom indicate how strongly participants agreed with the statement (1 = strongly disagree, 5 = strongly agree).*

References

[AC04] A.ELGAMMAL, C.LEE: Inferring 3d body pose from silhouettes using activity manifold learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2004), vol. 2, pp. 681–688.

[AT04] AGARWAL A., TRIGGS B.: 3d human pose from silhouettes by relevance vector regression. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2004), vol. 2, pp. 882–888.

[BR04] BAERLOCHER P., RONAN B.: An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer 20*, 6 (2004), 402–417.

[COL] COLDET: Free 3d collision detection library. http://sourceforge.net/projects/coldet.

[DA95] DEB K., AGRAWAL R.: Simulated binary crossover for continuous search space. *Complex System 9*, 2 (1995), 115–148.

[DAC*03] DAVIS J., AGRAWALA M., CHUANG E., POPOVIC Z., SALESIN D.: A sketching interface for articulated figure animation. In *Proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation 2003* (2003).

[GAL] GALIB: A c++ library of genetic algorithm components. http://lancet.mit.edu/ga/.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIC Z.: Style-based inverse kinematics. *ACM Transacitons on Graphics 23*, 3 (2004), 522–531.

[PB91] PHILLIPS C. B., BADLER N. I.: Interactive behaviors for bipedal articulated figures. *Computer Graphics 25*, 4 (1991), 359–362.

[RK92] RONIE H., KEN P.: Controlling 3d objects by sketching 2d views. In *Proc. SPIE* (1992).

[Ros60] ROSEN J.: The gradient projection method for nonlinear programming. part i. liner constraints. *Journal of the Society for Industrial and Applied Mathematics 8*, 1 (1960), 181–217.

[Tay00] TAYLOR C.: Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding 80*, 3 (2000), 349–363.

[TMC09] TABANDEH S., MELEK W. W., CLARK C. M.: An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots. *Journal Robotica* (2009).

[YN03] YAMANE K., NAKAMURA Y.: Natural motion animation through constraining and deconstraining at will. *IEEE Transactions On Visualization and Computer Graphics 9*, 3 (2003), 352–360.

[ZB94] ZHAO J., BADLER N.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics 13*, 4 (1994), 313–336.

[ZLK05] ZHAO J., LI L., KEONG K. C.: 3d posture reconstruction and human animation from 2d feature points. *Computer Graphics Forum 24*, 4 (2005), 759–771.