

# Vision-Based Posing of 3D Virtual Actors

Ameya S.Vaidya, Appu Shaji, and Sharat Chandran

Computer Science and Engineering Department, I.I.T Bombay  
Mumbai 400076, India  
{ameyasv, appu, sharat}@cse.iitb.ac.in

**Abstract.** Construction of key poses is one of the most tedious and time consuming steps in synthesizing of 3D virtual actors. Recent alternate schemes expect the user to specify two inputs. Along with a neutral 3D reference model, more intuitive 2D inputs such as sketches, photographs or video frames are provided. Using these, of all the possible configurations, the “best” 3D virtual actor is posed

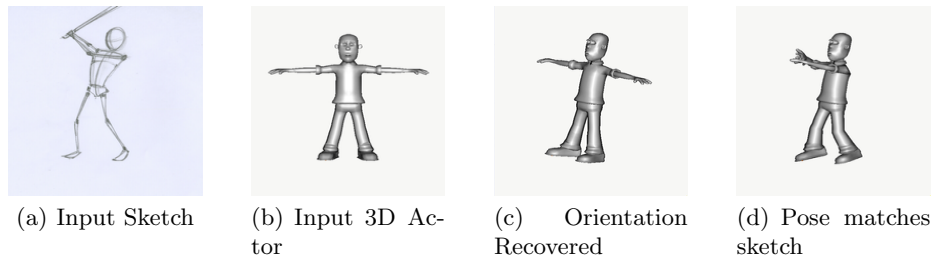
In this paper, we provide a solution to this ill-posed problem. We first give a solution to the problem of finding an approximate view consistent with the 2D sketch. Elements of this rigid-body solution are novel. Next, we provide a new solution to the process of extending or retracting limbs to more accurately suit the sketch. This posing algorithm, is based on a search based scheme inspired by anthropometric evidence. Less *physical work* is required by the actor to reach the desired pose from the base position. We also show that our algorithm converges to an acceptable solution much faster compared to the previous methods. ...

## 1 Introduction

Consider the following:

- A cricket coach uses an animation system in correcting the flaws in the strokes of one of his current players based on proven, vintage stars of the past. He generates “on the fly” a new three-dimensional (3D) sequence rapidly based on the combination of the 3D model of his current player and past videos at his disposal.
- An occupational therapist takes a scanned picture from her textbook. She overlays correct posture styles for the computer hacker hunched up over his laptop.

Animation systems currently used fall well short of providing the necessary amenities to realize the above. Due to the highly articulate and complex structure of human 3D characters and their respective motions, *posing* them in a 3D world and specifying their motion is by no means a trivial task. Alternative schemes [1] [2] have thus evolved that *compute* a desired pose from a 2D sketch, a photograph, or from a video frame. For simplicity, we assume in the rest of the paper that our inputs are artist drawn sketches, and a reference 3D virtual actor modeled as an articulated skeleton.



**Fig. 1.** A sketch and a 3D actor (top row) is presented to our system. It first (bottom left) re-orientes the actor rigidly and then “moves” the limbs to match the sketch. In this example, the positions of all four limbs are computed and the knee adjusted.

### 1.1 Problem Statement and Contributions

There are two issues in computing a solution as in Fig. 1. First, an approximate viewing direction must be found that orients the 3D actor to match the given sketch. At the end of this step, the 3D actor has rigidly oriented himself to be ready to move to the new configuration in the sketch.

Second, the actor changes relative positions of the “bones” so as to match the sketch.

Note that there are potentially infinite configurations that match the sketch. Multiple positions may be used to construct an animation sequence based on key intermediate poses.

We provide a partially automated solution to this problem. In our scheme, the end user specifies a few correspondences between points on the 2D sketch and points in the 3D reference model. Our system automatically *constructs* the gross orientation. Next, based on this orientation, the system automatically moves the limbs in a non-rigid fashion to match the sketch. We list the features of our work.

1. A more robust (albeit domain-specific) rigid body camera recovery algorithm is presented (see Section 3).
2. For the “most-likely” non-rigid motion, a novel search based scheme inspired by anthropometric<sup>1</sup> evidence is introduced.
3. The notion of a physically based metric to quantify the results is introduced. Compared to existing methods, our scheme requires less physical work to reach the specified sketched pose from the previous position. The application of this to energy efficient robotics is immediate.
4. Compared with prior iterative solutions, our method takes less time and can be done at interactive rates.
5. Since the reconstruction from 2D sketches to 3D poses is not unique, we provide the end user the option to select from multiple solutions. The solutions are returned in an order of “less movement” to “more movement” on the part of the 3D actor.

<sup>1</sup> Anthropometry: measurement and study of the human body and its parts and capacities

- The sketches provided by the artist are not expected to be the exact projections of the desired pose. But a loose sense of proportion is expected.

The rest of the paper is organized as follows. After considering related work in Section 2 we give a brief overview of the camera recovery algorithm in Section 3. We present the conceptual and implementation details of our posing algorithm in Section 4. We analyze our results and perform statistical comparisons with earlier methods in Section 5. Final remarks appear at the end.

## 2 Previous Work

The problem of extracting 3D poses from 2D poses has been tackled in various domains like robotics, CAD/CAM, computer vision, animation and graphics before. A popular approach is to use two or more images from different viewpoints to resolve ambiguity between multiple valid poses.

A technique for reconstructing human body poses from single images with the aid of anthropological data is discussed in [3]. In [4], the authors have outlined a technique that relies on known point correspondences between predefined landmarks on the human body. Most of these are learning based schemes. An alternative strategy, useful for some 3D animation applications, is to use information from “previous” frame(s) when available. See for example, [5].

Another school of thought is to re-structure the problem as an optimization problem [6][7]. The hypothesis behind optimization-based posture prediction is that human motion concerning different tasks is governed by different performance measures. These measures can be aggregated using multi-objective optimization techniques.

Pose recovery techniques close to our stated goals are discussed in [1] and [2]. The method proposed in [2] achieves a good amount of automation but works only with “stick-figures” as 2D input. Also they require the 2D skeleton to be an isomorph of the 3D skeleton, which limits the applicability of the method. This assumption, for example, may not hold true with motion capture tracking data [2]. Our work is essentially patterned around [1]. We re-work the posing scheme so as to make it more robust, faster and closer to actual human motion. Further, our algorithm has the option of returning multiple solutions.

Our posing algorithm is loosely based on Cyclic Coordinate Descent (CCD) method [8] [9]. An excellent introduction of all of the problems and general approaches to Inverse Kinematics is provided in [8].

## 3 Recovery Of Gross Orientation

The key to recover the orientation is to find a “camera” such that when it looks at the 3D shape in the *correct* orientation, the projection of the 3D shape matches the input sketch. Mathematically, the camera is a matrix  $\mathbf{P}_{3 \times 4}$

$$\mathbf{x} = \mathbf{P}\mathbf{W} \tag{1}$$

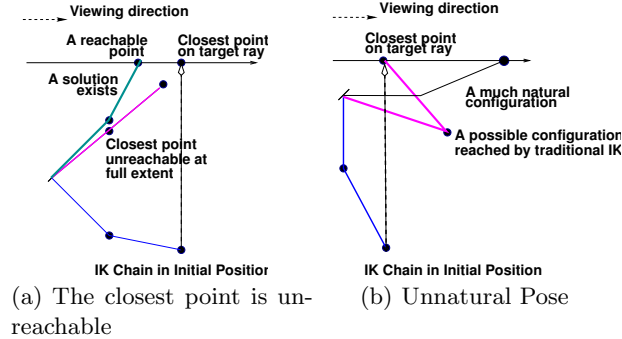


Fig. 2. Problems with closest point assumption

where  $\mathbf{W}_{4 \times 1} = [X \ Y \ Z \ 1]^T$  is an object point and  $\mathbf{x}_{3 \times 1} = [u \ v \ w]^T$  is the corresponding image point.

$\mathbf{P}$  can be computed given a set of *user-clicked* point correspondences  $(\mathbf{x}_i, \mathbf{W}_i)$ , between the image and the reference actor. Normally, at least six point correspondences (in general position) are required for the simplest camera model.

### 3.1 Our Method

Instead of finding the camera matrix, and then recovering the 3D points, our domain-specific method directly computes the required 3D points. The method requires no more than *five* points such that the clicked points belong to the same object and no four of these points are co-planar.

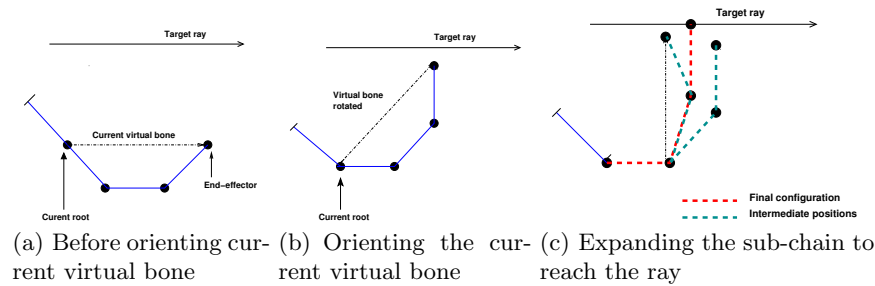
The clicked sketch points,  $\mathbf{x}_i$ , are 2D projections of the corresponding, currently unknown, 3D points  $\mathbf{W}'_i = [X'_i \ Y'_i \ Z'_i \ 1]$ . However, the 3D positions, and hence distances, are known in the reference position. Because the camera recovery phase is a rigid body transformation, the distances between joints in the skeleton is preserved. The unknown points  $\mathbf{W}'_i$  are computed using this invariant. The details of this step (keeping in mind issues such as the scaling, origin alignment, and the like) are skipped for brevity.

The unknown transformation  $\mathbf{T}$  given by  $\mathbf{W}'_{4 \times 1} = \mathbf{T}_{4 \times 4} \mathbf{W}_{4 \times 1}$  can now be obtained. Finally,  $\mathbf{T}$  is related to the camera matrix by the relation  $\mathbf{P}_{3 \times 4} = \mathbf{K}_{3 \times 4} \mathbf{T}_{4 \times 4}$  where  $\mathbf{K}_{3 \times 4}$  is the projection matrix. This enables us to find  $\mathbf{C}$ , the viewing direction, as the right null space of the camera matrix.

## 4 Non-Rigid Posing

The non-rigid transformation is the next step. The problem is set up as an inverse kinematics problem. At this point, the user clicks corresponding positions of the desired limb, termed as the *end effector*. This is the pair  $(\mathbf{e}, \mathbf{W})$  on the sketch and the reference character. The system then back-projects the 2D sketch position  $\mathbf{e}$  to obtain the target ray in 3D space  $\mathbf{R}_e = \lambda \mathbf{C} + \mathbf{P}^{-1} \mathbf{e}$  where  $\lambda$  is a real number.

As a minimum, only the position of the end effector is given, and that too in an approximate sense. The true 3D position, and the intermediate joints are not specified by the user. Of course, to construct a more accurate pose, the user may decide to provide the 2D positions of the intermediate joints as well, and the system will use this information when available. However we have found that this is rarely necessary. A properly constructed model along with our strategy of returning multiple discrete configurations, as discussed in section 4.2, yields satisfactory poses in most cases.



**Fig. 3.** Phases in our algorithm

#### 4.1 Basic Idea

Where on this ray will the actual point lie? This is an important question which drives the quality of the solution. Two choices are considered

- The method proposed in [1] uses the *closest point* on this ray to the *current* end-effector position as the target position and applies traditional inverse kinematics. However the closest point may be unreachable as seen, for example, in Fig. 2(a) or may lead to unnatural poses as in Fig. 2(b).
- Alternatively, blind Jacobian based inverse kinematics may be used where the first satisfying end point is automatically computed. However, the 3-D reference character has to perform more physical work (Section 5). Besides this method needs more iterations to converge to a solution.

The intuition behind our scheme comes from the way human limbs operate. The human limb motion compromises on various factors like the effort required in planning the motion, the energy expended in executing the motion and stability of the resultant posture. As a result, limb motion occurs by an overall gradual rotation towards the goal along with simultaneous extension or retraction to span the required distance [10].

We mimic the above behavior by using a search based scheme. We use a recursive bi-directional search for the best configuration to reach the target ray

starting from the smallest sub-chain. At each step of the recursion, let *current root* be the joint at the base of the current sub-chain. We call the vector from the current root to the end-effector a *virtual bone*. The algorithm orients the current virtual bone by rotating the current root so that the end-effector is closest to the target ray. The algorithm then extends or retracts the chain to try and reach the ray. To do this, it recurses with a smaller sub-chain to search for a suitable configuration that places the end-effector on the ray. If successful, the algorithm returns. If we are unsuccessful, but this step reduces the posing error, the resulting configuration is saved before proceeding further. Finally if instead it increases the posing error, the rotation applied to *current root* is undone and the chain is restored to the last saved state<sup>2</sup>. The process is shown in Fig. 3 and the algorithm is given in Algorithm 1. Finally longer sub-chains are considered in Algorithm 2.

---

**Algorithm 1** PoseChain (IN : *start*, IN : *end*, IN : *ray*, IN : *thresh*, IN/OUT : *error*) : *Success, Partial*

---

```

1: if error < thresh then
2:   return Success
3: else
4:   if start == end then
5:     return Partial
6:   end if
7: end if
8: virtBone ← (curRoot, end)
9: Compute rotation(s) for virtBone
10: Select best rotation bestRot
11: Save the current value of curRoot
12: Apply bestRot to virtBone
13: Compute current posing error locError
14: if PoseChain(start.child, end, ray, locError) == Success then
15:   return Success
16: end if
17: if locErr < error then
18:   error = locErr
19: else
20:   Restore the saved value of curRoot in step 11
21: end if
22: return Partial

```

---

Since PoseChainTop considers increasingly longer chains, the time complexity of the algorithm,  $T(n)$  in terms of chain length  $n$  can be computed by the

<sup>2</sup> The desired orientation of the current virtual bone  $PQ$ , where  $P$  is the current root and  $Q$  is the current end-effector, is computed by drawing a sphere centered at the current root and radius equal to the length of the current virtual bone. The closest point to the ray be  $S$ . The rotation  $R$  is the one which aligns  $PQ$ , along  $PS$ .

---

**Algorithm 2** PoseChainTop (IN : *start*, IN : *end*, IN : *ray*, IN : *thresh*) :  
*Success, Partial*

---

```

1: error ← ∞
2: for curRoot = end.parent to start do
3:   if PoseChain(curRoot, end, ray, thresh, error) == Success then
4:     return Success
5:   end if
6: end for
7: return Partial

```

---

recurrence

$$T(n) = \sum_{i=1}^{i=n} T_1(i) \quad (2)$$

where  $T_1$  is the time complexity of the recursive algorithm `PoseChain`. Consider the call to algorithm `PoseChain` with chain size  $m$ . To compute  $T_1(m)$  we note that it consists of a single recursive call of size  $m - 1$  in step 14. All other steps can be done in constant time. Therefore the algorithm `PoseChainTop` is quadratic.

This is much better than a Jacobian based scheme, where at every step a  $6 \times n$ , matrix must be computed and inverted. Further, in our case the “steps” taken by the IK chain are much larger than the Jacobian scheme. Thus our algorithm executes much faster than the traditional scheme.

`PoseChainTop` is loosely based on CCD [9]. In fact, the **for loop** in the algorithm `PoseChainTop` is a conceptual implementation of CCD. CCD is a linear time algorithm. However the length of an IK chain seldom exceeds 10, hence the execution time of `PoseChainTop` is well within interactive rates. Further, basic CCD suffers from the problem of excessive folding since the search proceeds in only one direction, from the end-effector towards the root of the chain. Once a bone is rotated, the sub-chain rooted at that bone is never considered again. In contrast, our method reconsiders the sub-chain via the recursive call at the end of algorithm 1, thus correcting for the excessive rotation that happens with standard CCD. This also takes care of convergence issues in the presence of joint-constraints<sup>3</sup>. Another advantage we have over CCD is the ability to generate multiple discrete configurations as discussed in section 4.2 below.

## 4.2 Generating multiple configurations

Using a search-based approach allows us to generate multiple discrete configurations using the following strategy. In general, at the point in the search tree where the algorithm returns successfully, the corresponding virtual bone will have two possible orientations—see step 9 of `PoseChain`. In the basic scheme, we select the “best” and discard the other. In the modified version, we cache

<sup>3</sup> In our system, joint constraints are modeled in the form of *constraint cones*, enclosing the joint in its parent coordinate frame (see Fig.6)

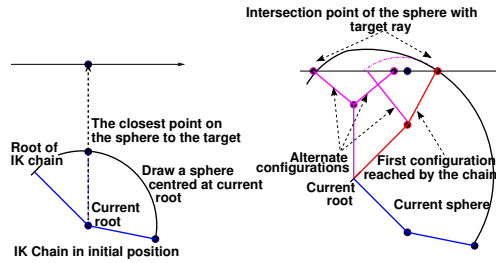


Fig. 4. Generating multiple solutions

the second orientation in algorithm 2, subject to joint constraints, before returning. This represents the node in the search tree from which the search must be restarted for other solutions. Now when the user requests another solution, we do an inorder search starting from this node, returning one “next” solution for every request. The modified scheme is shown in Fig. 4. Example output showing two poses generated by this method is shown in Fig. 5. Note that the input sketch given in the figure can correspond to two possible actions. One is during the *back-lift* of the bat for a righthanded batsman before the stroke is made and other the *follow-through* after the stroke for a left handed batsman. Observe that Fig. 5(b) is a valid representation of the follow-through pose and Fig. 5(c) that of the back-lift pose. Details of the algorithm are skipped in this version.

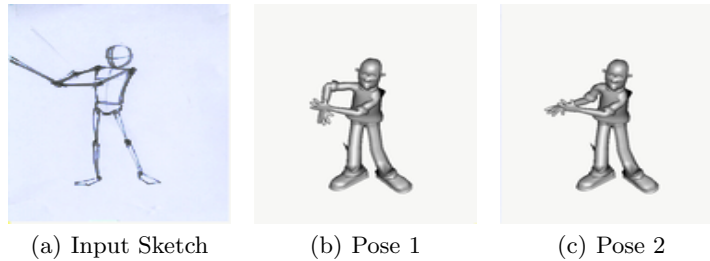


Fig. 5. Returning multiple solutions

## 5 Experiments and Results

A sample posing result for our scheme is shown in Fig. 1. We also implemented the Jacobian based method mentioned in Section 4. By and large, based on our discussions with kinesiological experts, our method looks more natural.

For quantitative comparisons, we compared the *physical work* done against the force of gravity to bring an IK chain from initial position to final position,

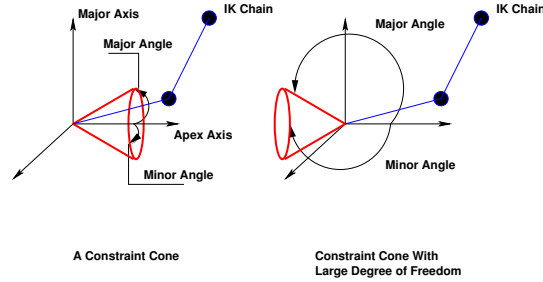
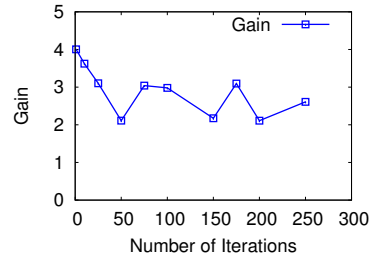
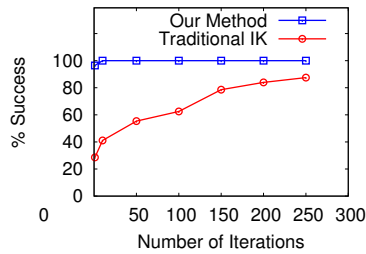


Fig. 6. The Constraint Cone



(a) Error versus computation time (b) Gain in work we achieve with respect to closest point approach

along with the *posing error* using each method. The posing error was computed as the perpendicular distance squared of the final end-effector position from the target ray. The work done on link  $i$  was taken as  $w_i = mgh_i$  where  $h$  is the vertical displacement of the center of mass of the link. The total work done in posing a chain was obtained by summing the contribution from each link.

About 250 experiments in different configurations were performed. A method was deemed successful if the error between the goal ray and the end-effector was less than a threshold (2% of the chain length).

The maximum number of iterations for which the closest point based iterative schemes were allowed to run is indicated. An equivalent amount of maximum compute time was allotted for our recursive method. As seen in Fig. 7(a) we observe that our method records success with far lesser computation time than the previous method. Also, in general our method constructs poses that require less *physical work* on the part of the character as seen in Fig. 7(b). We define gain as the ratio of the work done by our method with respect to the work done by the closest point method. Note that we are getting an improvement of at-least a factor of 2 in all cases. We compared our method with two methods: a Jacobian based blind IK method that attempts to minimize the distance with the target ray, and recent IK method[1] that targets the closest point on the target ray.

Though the posing accuracy of blind IK method is comparable to our method, we got an average improvement of a factor of 1.67 in terms of the work done.

## 6 Conclusion and future work

In this work, we have implemented a scheme for constructing 3D poses from 2D sketches, photographs, and video frames. We have demonstrated that our method robustly constructs poses that look natural, and can be constructed at interactive rates. An energy efficiency paradigm has been introduced and multiple solutions are provided and in these senses too our method performs well. There are a few areas that we would like to explore further.

1. A more complete actor model that has twist and pole vector rotations. While this is a matter of detail in the forward graphics problem, it will handle issues such as head rotation in Fig 1.
2. In many applications, it may be possible to compute a *homomorphism* between the 2D and 3D skeletons, thus eliminating the need for the user to manually click corresponding end-effectors. This has several interesting issues like
  - Efficient computation of the homomorphism
  - Handling the symmetries in the structure of a character

## References

1. Chaudhuri, P., Kalra, P., Banerjee, S.: A system for view-dependent animation. In: Eurographics 2004. (2004)
2. Davis, J., Agrawala, M., Chuang, E., Popovic, Z., Salesin, D.: A sketching interface for articulated figure animation. In: Eurographics/SIGGRAPH Symposium on Computer Animation. (2003)
3. Remondino, F., Roditakis, A.: 3D reconstruction of human skeleton from single images or monocular video sequences. In: DAGM-Symposium. (2003) 100–107
4. Parameswaran, V., Chellappa, R.: View independent human body pose estimation from a single perspective image. In: CVPR (2). (2004) 16–22
5. Bowden, R., Mitchell, T., Sarhadi, M.: Reconstructing 3D pose and motion from a single camera view. In Carter, J.N., Nixon, M.S., eds.: In Proceedings of the British Machine Vision Conference. Volume 2., University of Southampton (1998) 904–913
6. Marler, R.T.: Development of real time multi objective optimization based posture prediction. Technical report, The University of Iowa (2004)
7. Kim, J., Abdel-Malek, K., Yang, J., Nebel, K.: Motion prediction and inverse dynamics for human upper extremities. In: SAE 2005 World Congress. (2005) 11–14
8. Welman, C.: Inverse kinematics and geometric constraints for articulated figure manipulation. Master’s thesis, Simon Frase University (1993)
9. Wang, Chen, C.C.: A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. In: IEEE Transactions on Robotics and Automation. Volume 7:4. (1991) 489–499
10. Kibler, W.B., McMullen, J.: Scapular dyskinesis and its relation to shoulder pain. Journal of American Academy of Orthopaedic Surgeons (2003) 142–151