Geometry-Driven Physical Interaction between Avatars and Virtual Environments

Harald Schmidl and **Ming C. Lin**
University of North Carolina at Chapel Hill
Email: {schmidl,lin}@cs.unc.edu

Abstract

We present an interactive technique on virtual contact handling for avatars in virtual environments using geometry-driven physics. If a contact has occurred between an articulated avatar and a virtual environment, the global penetration depth and contact points are estimated based on a fast local penetration depth computation for decomposed convex pieces. The penetration depth and contact information are then used to resolve overlap between the avatar and the virtual environment. If applicable, joint angles for an articulated body are computed using an inverse kinematics approach based on cyclic coordinate descent. Resulting dynamic response with friction is modeled with impulse-based dynamics under the Coulomb friction law. We demonstrate the algorithm on a modestly complex virtual environment. The resulting system is able to maintain an interactive frame rate of 30-60 Hz.

Keywords: Virtual reality, inverse kinematics, dynamics, simulation, animation

Introduction

Motion is ubiquitous within both the physical world and any virtual environment (VE). The problems of collision detection and contact response are central to many tasks involving realtime interaction and physically-based manipulation in VEs, computer animation, simulation-based design, electronic prototyping, acquisition, evaluation, computer games, etc. In many of these applications, motion among different entities is often simulated by modeling the contact constraints and impact dynamics. This is especially important in creating an immersive VE for training and mission rehearsal.

The non-penetration constraints between a moving avatar (driven by a user) and the VE need to be enforced in realtime for interactive applications. This poses some challenging computational issues. First, a collision must be detected; next, any overlap between the avatar and the environment must be resolved, and appropriate physical response for the entire articulated body must be computed – all in realtime. Contact resolution must be handled in a physically plausible way. Figure 1 shows an example of a user's articulated hand interacting with a ball in a VE in realtime.

The additional challenge in interacting with a VE is to avoid overlap and resolve contact in a way that not only produces physically plausible motion but also preserves human characteristics of the avatar. Several issues need to be addressed. Any motion must be physical while satisfying various constraints. For example, an avatar is subject to biomechanical joint limits and all computations must meet realtime requirements. Realtime performance is necessary in order to achieve immersion in a VE. If the frame rate drops too low, a lag between the user's motion and the visual display will be noticeable. This artifact would considerably lessen the perceived realism.

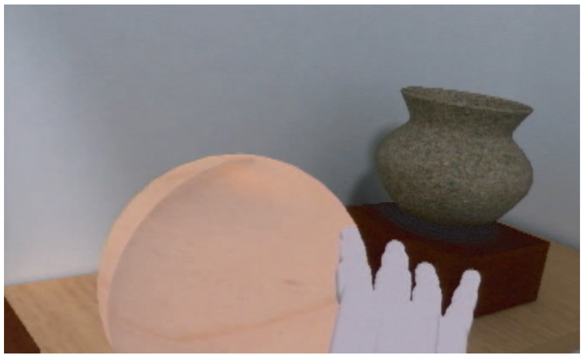


Figure 1: An articulated hand interacts with a ball in a VE.

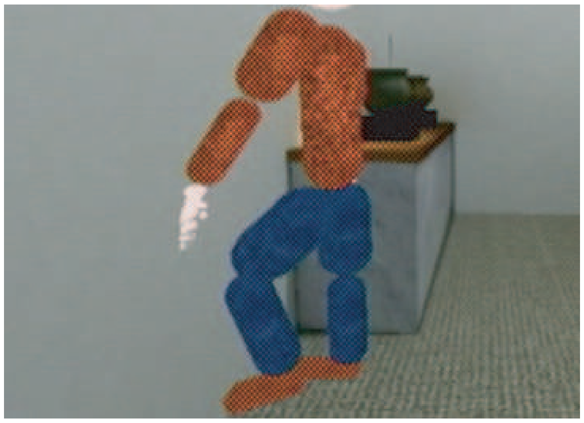


Figure 2: An avatar touches an object: the arm's joint angles are adjusted with IK to avoid overlap.

Our system addresses these problems with ”geometry-driven physics” using a quick estimation of global penetration depth (PD), combined with a hybrid approach that utilizes dynamics and kinematics. Computation of the PD is essential for our approach. Having the PD and associated normal direction allows us to separate overlapping objects. Unfortunately, computation of the PD is generally expensive and can take up to $O(n^6)$ for non-convex objects where n is the number of polygons representing the objects [1]. We introduce a quick approximation to computing global PD by decomposing the non-convex polyhedra into a collection of convex pieces by using surface decomposition. Combined with an iterative approach, this technique produces good results in practice. Furthermore, avoiding collisions for *e.g.* the avatar's articulated arm will necessitate adjustment of joint angles. We address this issue by using an inverse kinematics (IK) approach based on cyclic coordinate descent (CCD) [2]. Collision response is finally handled with impulse-based physics [3].

We have chosen a hybrid approach, *geometry-driven physics*, using a combination of geometric overlap minimization, IK, and impulse-based dynamics. This hybrid approach sacrifices some physical accuracy for speed. All motion stays visually plausible. Realtime performance is met, which is paramount.

We have demonstrated our system by incorporating it into a VR environment. We use a Cyberglove to interact with both static and dynamic objects in a VE and a whole-body avatar to demonstrate its interaction with objects in the VE and handling of avatar self-collisions. Figure 2 shows the whole-body avatar touching a box.

The remainder of this paper is structured as follows. We briefly summarize related work and present the overall architecture of our system. Then, we introduce our heuristic for estimating the global PD for non-convex objects quickly. We move on to present our hybrid approach of kinematics and impulse-based dynamics for overlap resolution and collision handling. Following, we describe and analyze experiments that we have conducted and provide some implementation detail. We finally conclude our paper and suggest possible future research directions.

Previous Work

We briefly summarize related work in this section. Arnaldi *et al.* [4] presented a good introduction on the topic of kinematic and dynamic animation of characters and discusses pros and cons in each approach. The authors stated that dynamics is especially beneficial if the quality of generated motion is important, as is the case for walking or handling collisions. Kinematics is more appropriate for grasping tasks or sitting on a chair, where joint angles and obstacle avoidance are of concern.

Boulic *et al.* [5] discussed the drawbacks of pure IK for animation. Direct kinematics is added to achieve better realism of motion and preserve dynamics. IK is exclusively used for overlap avoidance. The coach-trainee metaphor is coined and expresses that avatar motion is as close as possible to user motion. Boulic *et al.* [6] introduced inverse kinetics, which considers mass distribution besides traditional IK. This approach can be useful for posture control and keeping a synthetic actor's balance. Baerlocher and Boulic [7] present task-priority formulation for IK. Their approach allows specification of an order in which tasks, such as obstacle avoidance and aiming, are to be weighted. Monzani *et al.* [8] discuss motion retargeting with IK and an intermediate skeleton. The advantage is that user motion can be mapped easily onto a performer skeleton, even if the latter has rather different geometry. Their system will not handle self-collisions. Tolani *et al.* [9] discuss strategies for purely kinematic handling of human arms and legs. The goal is to develop a set of kinematically feasible solutions from which the user can choose. The incentive is to meet goals of aiming, position, orientation, and constraining an articulated chain's end effector to lie on a plane.

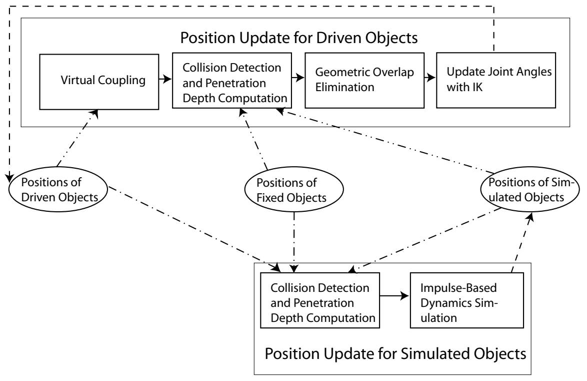

Position Update for Driven Objects
Virtual Coupling
Collision Detection and Penetration Depth Computation
Geometric Overlap Elimination
Update Joint Angles with IK
Positions of Driven Objects
Positions of Fixed Objects
Positions of Simulated Objects
Collision Detection and Penetration Depth Computation
Impulse-Based Dynamics Simulation
Position Update for Simulated Objects

Figure 3: The architecture of our system.

Overview of System Architecture

In this section, we give an overview of our proposed system architecture for modeling geometry-driven physical interaction between an avatar and a VE.

Preliminaries

We assume that all objects are rigid. Articulated bodies can be represented as a collection of rigid objects, held together by joints with joint constraints. (Local deformation of muscles and skins can be easily incorporated into our system as well.) All unit joints are planar, *i.e.* they allow a rotation around one axis, which is the principle of a knee joint. More complicated joints, such as the ball and socket shoulder joint, can be modeled as a collection of planar joints. For the shoulder we are using a collection of three planar joints whose axes share a common center to allow three degrees of freedom.

We classify the objects in the VE as *fixed*, *simulated*, and *driven* objects. A fixed object is locked at a given position and orientation in space. Simulated objects can move around freely subject to the laws of physics. A driven object follows the movements of a human user. All objects in the VE are modeled and simulated to interact with each other as we expect in the physical world. In particular, collisions have to be modeled realistically and overlap has to be prevented.

System Architecture

We will now give an overview for the architecture of our system, as shown in figure 3. Our algorithm first finds new and non-overlapping positions for all driven objects. This step is described by the upper loop in figure 3. Input happens along the marked paths $(- \cdot \cdot)$ to the appropriate modules. Virtual coupling attracts the driven objects to the tracker, the collision detection and geometric overlap elimination modules find plausible end-effector positions, and then IK computes suitable joint angles for articulated chains. Positions for driven objects are then updated along the dashed path.

This process loops until all driven objects are updated without overlap.

According to the lower loop, simulated objects then move with a time-step equal to the frame time under non-penetration constraints and according to the laws of physics. All object positions are input to the collision detection module along the indicated path ($-\cdot$). If collision is detected, the forward motion stops, impulse-based dynamic simulation resolves the collision, and object motion continues until the end of the frame is reached. Positions of simulated objects are updated along the dashed path.

Both processes have as input all object positions. However, only one type of objects is updated at a time. First the driven objects, then the simulated objects. In this way, we prioritize the position update, giving driven objects a stronger incentive to meet their desired goals. Fixed objects always stay at their assigned positions for the whole simulation. After all bodies had their final positions computed they are rendered.

Estimating the Global Penetration Depth

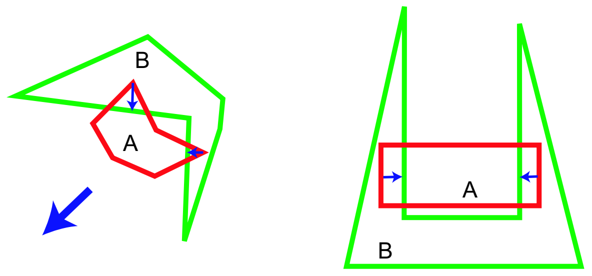
We now present the mechanism for resolving our non-penetration (or non-overlap) constraint. Good algorithms are known for collision detection [10, 11, 12, 13, 14, 15]. These algorithms typically work by tracking the closest features using local information. However, very few practical approaches are known for computing the PD, even for simple convex polyhedra. General PD computation can be performed based on explicit Minkowski sums. However, explicit computation of the Minkowski sum can have $O(n^6)$ computational complexity in the worst case [1]. Furthermore, the resulting algorithms are generally known to be subject to robustness problems.

A recent algorithm based on dual-space (i.e. Gauss map and Minkowski Sum) extension has been introduced by Young et al. for efficient computation of the PD between convex objects [16]. Analogous to Voronoi tracking [12, 14, 15], the algorithm finds a locally optimal solution by walking on the surface of the Minkowski sums. A local Gauss map allows implicit computation of the surface of the resulting Minkowski sum. Another PD algorithm using graphics processors and hierarchical refinement has been proposed to handle non-convex polyhedra [17]. However, this approach may take up to seconds to compute the global PD between two non-convex objects and cannot be used in realtime applications. Therefore, we developed a new global PD algorithm based on the computation of multiple convex pairs in the contact region to quickly estimate the global PD required to separate the two objects.

We assume a convex decomposition of all objects is known [12, 16]. We can compute the PD per pair of decomposed convex pieces using the dual-space expansion technique [16]. For two overlapping, non-convex objects we have i convex pairs and their associated PDs, pd_i, and penetration normals, $\mathbf{n}_i$. We estimate the general $\mathbf{PD}_g$ by:

$$\mathbf{PD}_g = \frac{\sum {pd_i}^2 \mathbf{n}_i}{|\sum pd_i|}. \quad (1)$$

Note that $\mathbf{PD}_g$ defines a direction. Intuitively, the length of $\mathbf{PD}_g$, $|\mathbf{PD}_g|$, defines a measure for how much two overlapping objects must be translated to diminish the overlap, and the normal $\mathbf{PD}_g/|\mathbf{PD}_g|$ defines the direction of translation. For overlapping objects **A** and **B** with contact normals per convex pair pointing out of **B** and into **A**, we would translate **A** by $0.5\mathbf{PD}_g$ and **B** by $-0.5\mathbf{PD}_g$ in order to diminish the overlap purely by translation.



(a) A "reasonable" case. (b) A pathological case.

Figure 4: Finding the general PD.

This approach enables a quick estimation for the global PD in most cases, but can fail for certain degenerate scenarios. For an illustration, see figure 4. Case 4(a) shows a "reasonable" scenario where overlap is not too deep and can be resolved by translation following our new approach described above. The large arrow represents the global PD. Case 4(b) is a pathological case: the two normals cancel each other out.

An optimization-based algorithm that also allows for rotation to eliminate overlap is available [18] but is too costly for interactive VEs. Since realtime performance is crucial in an immersive VE, we can mimic this optimization approach by interpolating between the last cached, non-overlapping state and the currently overlapping state to find positions close to the initial positions without overlap. This technique moves objects back towards "where they came from" to resolve overlap.

We have found this approach to work effectively in practice without visual artifacts, and it satisfies our realtime performance constraints. We summarize our algorithm for estimating the global PD:

```
Estimate global PD
Input  Convex decomposition of all objects.
Output  Global PD per pair of general objects.
1. Cull non-overlapping pairs.
2. Determine the PD per convex pair.
3. Compute $\mathbf{PD}_g$ with equation 1.
4. If needed, interpolate between current and last state.
```

ALGORITHM 1: Global PD Estimation for nonconvex objects.

Hybrid Approach of Kinematics and Dynamics

This section describes our hybrid approach to resolve overlap and handle collisions for articulated bodies. This step consists of two components: geometric and kinematic overlap avoidance, and handling resulting collisions and contacts by applying impulses. We first explain how to find plausible positions by resolving all overlap.

Geometric and Kinematic Overlap Avoidance

The quickest approach to resolve overlap for a pair of objects is by pushing the objects apart by their amount of interpenetration, i.e. global PD. We found that even for a relatively

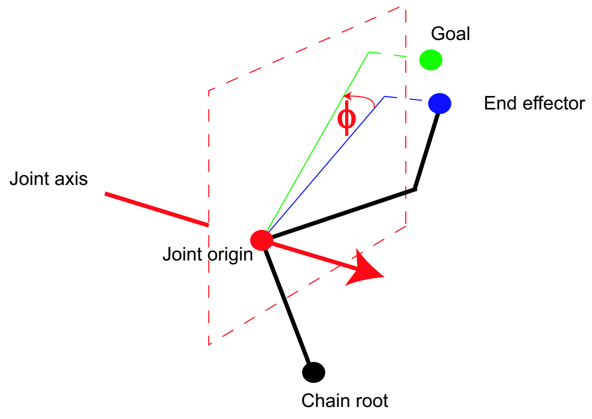


Figure 5: Explanation of the CCD algorithm: the joint would have to be rotated by ϕ in order to minimize the distance to the goal.

complex object, such as an articulated hand, this approach can work effectively. We have been able to find a plausible position for the hand by simply displacing it by $\mathbf{PD}_g$ according to equation 1.

However, we need to distinguish several cases. If a fixed object is penetrated by a dynamic or driven object, only the latter will be displaced to remove the overlap. If a driven and simulated object overlap we assign a priority and try to remove overlap only by displacement of driven objects. If this approach does not resolve overlap, we also need to move the simulated ones.

For a whole arm we can no longer use simple geometric overlap avoidance but have to combine it with inverse kinematics (IK). IK can compute joint angles that place a kinematic chain's *end effector* at the desired *goal* location, the configuration where overlap is eliminated for the end-effector.

The IK problem can be solved by different numerical techniques. Inverse Jacobian methods are known to not always have stable solutions [19] due to singularities. Nonlinear optimization methods are generally providing good results but the algorithms tend to be expensive [20]. We chose to employ cyclic coordinate descent (CCD) due to its simplicity and speed.

CCD was introduced by Wang and Chen [2] and can be viewed as a stepwise optimization algorithm. Joints are optimized one at a time until we arrive close enough at a global minimum or it was decided the minimum cannot be reached. Despite of its heuristic nature we found CCD to produce visually pleasing results.

The principle of CCD is explained in figure 5. Identify link i as the closest link to the chain's root which interpenetrates with an obstacle. Find all contacts $\mathbf{c}_{il}$ on it ($l = 1..k$). Calculate the end effector on link i:

$$\mathbf{ee}_i = \frac{\sum \mathbf{c}_{il}}{k}. \tag{2}$$

To determine the goal, we project $\mathbf{ee}_i$ onto the exterior of the obstacle:

$$\mathbf{goal}_i = \mathbf{ee}_i \pm \mathbf{PD}_{gi}, \tag{3}$$

with $\mathbf{PD}_{gi}$ the global PD of link i with the obstacle according to equation 1. If link i is object $\mathbf{A}$ in the pair made up by link i and the obstacle we have to add $\mathbf{PD}_g$, and if it is object $\mathbf{B}$ we subtract according to the direction of the contact normal as pointing into object $\mathbf{A}$.

To find the adjustment for joint h that will minimize the end effector's distance to the goal, CCD projects the vectors that connect the joint origin with the goal and end effector onto the joint plane. The projection of a point $\mathbf{p}$ onto the joint plane is calculated with

$$\mathbf{p}' = \mathbf{p} - (\mathbf{n} \cdot (\mathbf{p} - \mathbf{o}_h)) \cdot \mathbf{a}_h. \tag{4}$$

where $\mathbf{a}_h$ is the joint axis, $\mathbf{o}_h$ is the joint origin, and $\mathbf{p}'$ is the projected point.

After finding the projections of the end effector $\hat{\mathbf{ee}}_i{}'$ and the goal $\hat{\mathbf{goal}}_i{}'$ with equation 4, the angle adjustment for joint h is then given by:

$$\phi_h = acos(\hat{\mathbf{ee}}_i{}' \cdot \hat{\mathbf{goal}}_i{}'). \tag{5}$$

We run this step from the end effector inwards to the root over all joints until the distance between end effector and goal was minimized to within a small tolerance within a few millimeters. We repeat the process per chain until there are no more links with interpenetration. Our examples have five chains for arms, legs, and neck. The order in which we treat the chains is randomized. Using CCD, this approach allows collision resolution for also self-colliding chains, *e.g.* right arm colliding with left arm.

Joints in a robot or human are subject to mechanical and biological limits respectively. For planar joints, we enforce limits by simply defining an allowable maximum and minimum angle. For joints that are a collection of planar joint components we use reach cones [21] and perform a projection of a limb that has left a reach cone back to the nearest location inside of the reach cone.

We observe a special case where the interpenetrating link i is almost parallel to the obstacle's surface. In order to avoid unnatural poses we move limb i out of the obstacle with pure translation by $\mathbf{PD}_{gi}$, and reconnect limb $i-1$ by adjusting all joints from limb $i-1$ inwards with CCD.

We allow pseudo-magnetic attraction between the actual joint angles as calculated with CCD and the desired angles as given by the tracker. Even if there is no user motion the joints try to move to the desired angles in order to have the visually displayed posture as close as possible to the user posture. This technique has been described as coach-trainee metaphor [5].

Impulses

Having found non-overlapping positions using purely geometric overlap avoidance and IK, we can now apply impulses to move the other dynamic objects in the VE realistically. We have chosen impulse-based physics [3] for collision response. Despite of its limitations for systems with large numbers of collisions, it is well suited for handling virtual contacts for VEs. Since our training application does not require modeling large, crowded clusters of objects with many collisions, but rather localized interaction between the avatar and the environment. Impulse-based physics achieves good physical realism and offers superior running time performance in this case.

Dynamic objects move forward in time until the next collision occurs, the collision is resolved by updating velocities, and motion continues. Problems can arise when the number of collisions rises or objects are in close proximity [18, 22]. We alleviate the problem by virtual coupling between the tracker and driven object. A stiff spring is inserted between the tracker and the driven object:

$$m\ddot{\mathbf{x}} = -K\mathbf{x} - R\dot{\mathbf{x}}, \tag{6}$$

with m the tracked object's mass, $\mathbf{x}$ its position, and R and K the spring constant and damping factor respectively.

We define a collision as a contact between objects with negative relative normal velocity $(v^{\perp} < 0)$ [23, 18, 3, 22]. Assume for now we have two colliding objects **A** and **B** with the collision normal $\mathbf{n}$ pointing from **B** to **A** and the relative contact velocity $\mathbf{v}_{ab}$. The relative contact normal velocity can be computed according to:

$$v^{\perp} = \mathbf{n} \cdot \mathbf{v}_{ab}. \tag{7}$$

An impulse is applied at each collision $(v^{\perp} < 0)$ such that the objects become separating $(v^{\perp} \geq 0)$. Newton's empirical model relates relative contact normal velocities before, v^{-}, and after impact, v^{+}, by a coefficient of restitution ϵ:

$$v^{+} = -\epsilon v^{-}. \tag{8}$$

To make colliding objects instantaneously receding, we calculate equal but opposite, frictionless impulses $\mathbf{j}$ along the direction of the contact normal: $j\mathbf{n}$ for object **A** and $-j\mathbf{n}$ for object **B**. The following formula computes the impulse magnitude j:

$$\begin{aligned} A &= \mathbf{n} \cdot \left(\mathbf{I}_a^{-1}(\mathbf{r}_a \times \mathbf{n})\right) \times \mathbf{r}_a, \\ B &= \mathbf{n} \cdot \left(\mathbf{I}_b^{-1}(\mathbf{r}_b \times \mathbf{n})\right) \times \mathbf{r}_b, \\ j &= \frac{-(1+\epsilon)v^{-}}{m_b^{-1} + m_a^{-1} + A + B}. \end{aligned} \tag{9}$$

Subscripts refer to the appropriate properties on objects **A** and **B** with m for object mass, $\mathbf{I}$ for object inertia, and $\mathbf{r}$ for the moment arms connecting a object's center of mass and the contact point. Collisions are treated by application of impulses sequentially in a priority queue [22].

Adding friction

The previous section does not take into account the effects of friction. To model friction, typically a tangential, frictional impulse is applied that opposes the direction of sliding. According to Coulomb's friction law, the magnitude is set to μ times the magnitude of the normal impulse j as computed according to equation 9:

$$\mathbf{j}_t = -\mu j |\mathbf{v}_t|^{-1}\mathbf{v}_t, \tag{10}$$

where $\mathbf{v}_t = \mathbf{v}_{ab} - (\mathbf{n} \cdot \mathbf{v}_{ab}) \cdot \mathbf{n}$ is the tangential part of the collision velocity $\mathbf{v}_{ab}$ between objects **A** and **B** and j was computed with equation 9 [22].

Implementation and Results

We will now discuss and analyze experiments conducted. Implementation data is also provided. Our benchmark simulates an art gallery with two rooms and several objects in them. UNC's EVE Group provided this model. We implemented our algorithms on a 1.7GHz Intel Xeo machine. The articulated hand is driven by a user wearing a Cyberglove. Tracking is performed with a UNC HiBall and

ϵ/μ	wood	human	metal
wood	0.3/0.3	0.2/0.2	0.2/0.1
human	0.2/0.2	0.1/0.1	0.2/0.1
metal	0.2/0.1	0.2/0.1	0.3/0.1

Table 1: Coefficients of restitution and friction for our possible material combinations.

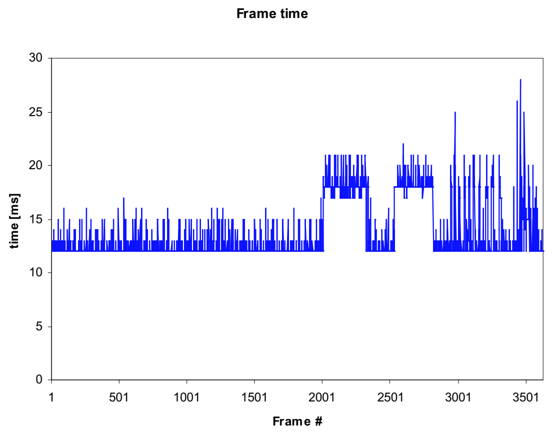


Figure 6: Frame time during a simulation of 3653 frames.

ceiling tracker. Distance computations are performed with SWIFT++ [12] and DEEP [16]. All objects in the environment have material properties for friction and restitution assigned. Refer to table 1 for the values used for ϵ and μ. The reader may access videos and other related material on http://www.cs.unc.edu/GeomPhysics.

We created a number of scenarios that demonstrate the performance of our system. The room environment used for all scenes has about 20 objects with a total of 476 triangles. The articulated hand consists of 18 parts with triangles numbering 4964. The hand interacts with the environment by touching objects and sliding along walls and the table, but also by playing with a ball (960 triangles). The hand is assumed to be rigid for the purpose of resolving overlap. Once $\mathbf{PD}_g$ is found, we push the hand back by pure translation. This example does not use any IK.

We experimented with different values for K and R for the virtual coupling in impulse handling and found $K = 1000.0$ and $R = 40.0$ to work well. A stiff spring is necessary to give the hand just enough freedom to prevent infinite loops when it collides with simulated objects, yet it should not move too far from its current location.

Pairwise collision queries are handled using *sweep and prune* [11] to cull away object pairs not in close proximity of each other. Figure 6 shows the frame rate. The first half of the graph indicates the application in idle state. The hand moves around in the room and overlap must be checked. The following two plateaus correspond to the hand sliding over a table, first in one direction and then in the opposite direction. The frame time rises accordingly, falls off briefly as contact breaks, but rises again as the hand slides in the opposite direction. The spikes towards the end correspond to a ball being pushed over the table with the frame time rising instantaneously, indicating collisions.

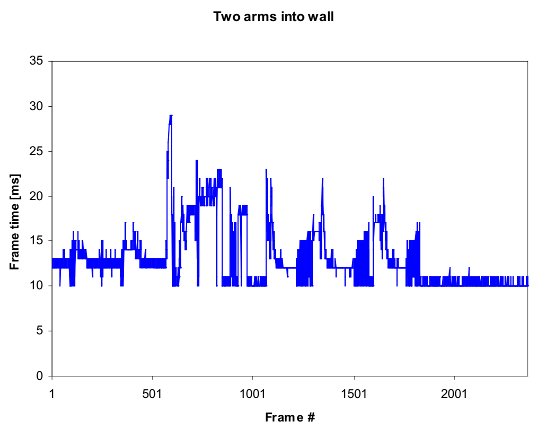


Figure 7: Frame time during the *two arms* scene.

We achieve overall realtime performance with varying frame rates between 30Hz and 60Hz. In our particular scenario, overlap resolution takes up about 50% of the running time and collision handling with impulses 25%. The remainder of the time goes into rendering, maintaining general data structures, and tracker reading.

Our next set of experiments uses an avatar (3600 triangles) in the same art gallery environment to test our IK approach. The avatar's motion is controlled by user input via mouse and keyboard due to lack of full body tracking hardware. Table 2 summarizes timing data for some scenarios we tested. Note that the numbers provided are averages. Some frames do not have any interaction at all, while some others have many collisions.

The scenes are ordered according to computational cost. It is clear that the cost for the overlap avoidance with IK rises with the number of CCD iterations and joints handled. The part that IK takes in the total frame time grows moderately as the number of adjusted joints and iterations grows.

We also observed a correlation between the number of iterations and motion coherence. If a limb makes contact for the first time more iterations will be necessary to remove the initial overlap. Once this task is performed, consecutive frames have a lower number of iterations and the total cost per frame is reduced. Figure 7 shows three spikes in the second half of the plot where the frame time rises rapidly but falls off rather quickly. This sequence was generated by the avatar making several pushes into the wall. The spikes correspond to solving the initial interpenetration, with the following falloff where the number of necessary iterations is gradually reduced.

There is a similar correlation for the number of contacting pairs and the part where collision detection dominates in the total running time. As expected, collision queries play a bigger role for the non-convex *arm in corner* scene. The arm is more tightly contacting from two sides and the cost of collision detection rises accordingly. For all scenes, the total frame rate stays clearly above 30fps, *i.e.* realtime.

Conclusion and Future Work

Our hybrid approach combines physically-based and kinematic methods. IK is only used to correct the avatar postures, not for generation of the whole motion. We can handle self-collisions and achieve realtime performance. The latter is crucial for immersive VR applications but also interactive games. The global PD is estimated efficiently and sufficiently accurate using our new approach. Occasionally we have found that collisions are missed due to discrete collision checks. We plan to extend our approach using continuous collision detection in the near future.

References

[1] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[2] Li-Chun Tommy Wang and Chih Cheng Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, August 1991.

[3] Brian Mirtich. Impulse-based simulation of rigid bodies. *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 181–189, 1995.

[4] B. Arnaldi, G. Dumont, G. Hgron, N. Magnenat-Thalmann, and D. Thalmann. Animation control with dynamics. *State-of-the-Art in Computer Animation*, pages 113–124, 1989.

[5] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figures motion editing. *Computer Graphics Forum*, 11(4):189–202, 1992.

[6] R. Boulic, R. Mas, and D. Thalmann. Inverse kinetics for center of mass position control and posture optimization. *Proc. European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, November 1994.

[7] P. Baerlocher and R. Boulic. Task priority formulations for the kinematic control of highly redundant articulated structures. *IEEE IROS '98*, pages 323–329, 1998.

[8] J.-S. Monzani, P. Baerlocher, R. Boulic, and D. Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Proc. Eurographics 2000*, 2000.

[9] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[10] Stephen Cameron. Enhancing GJK: computing the minimum and penetration distances between convex polyhedra. *IEEE Int. Conf. Robotics & Automation*, pages 3112–3117, April 1997.

[11] Jonathan C. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scaled environments. *Symposium on Interactive 3D Graphics*, pages 189–196, 1995.

[12] S. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of Eurographics'2001)*, 20(3), 2001.

scene	#frames	t_frame[ms]	#joints/frame	iter/frame	t_IK[%]	pairs/frame	t_cq[%]
two arms	1496	14.63	16	3	12.7	3	17.4
self-collision	2327	15.99	28	5	14.6	5	21.1
two arms and leg	2440	17.47	32	6	16.7	6	24.8
arm in corner	2146	20.49	40	7	18.5	13	31.3

Table 2: Timings for different scenes with IK. We record the number of frames, frame time, joints per frame solved with IK, iterations of CCD with readjusted goal per frame, part of IK per total frame time, overlapping pairs per frame, and part of collision detection per total frame time.

[13] S. A. Ehmann and M. C. Lin. SWIFT: Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. Technical report, Computer Science Department, University of North Carolina at Chapel Hill, 2000.

[14] Ming Lin and John Canny. A fast algorithm for incremental distance calculation. *International Conference on Robotics and Automation*, pages 1008–1014, 1991.

[15] Brian Mirtich. V-Clip: Fast and robust polyhedral collision detection. *Transactions on Graphics*, 17(3):177–208, 1998.

[16] Y. Kim, M. Lin, and D. Manocha. Deep: An incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics and Automation*, 2002.

[17] Young J. Kim, Miguel A. Otaduy, Ming C. Lin, , and Dinesh Manocha. Fast penetration depth computation for physically-based animation. *ACM Symposium on Computer Animation*, July 2002.

[18] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. *SIGGRAPH 01 Conference Proceedings*, pages 37–46, 2001.

[19] Z.R. Novakovic and B. Nemec. A solution of the inverse kinematics problem using the sliding mode. *IEEE journal of Robotics and Automation*, 6(2):247–251, 1990.

[20] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

[21] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *Journal of graphics tools*, 6(2):27–41, 2001.

[22] Harald Schmidl and Victor J. Milenkovic. A fast impulsive contact suite for rigid body simulation. *IEEE TVCG*, 10(2):189–197, March/April 2004.

[23] Raymond M. Brach. Rigid body collisions. *Journal of Applied Mechanics*, 56:133–138, March 1989.