Real-time Physically-based 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 XXX. 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 used for geometric overlap avoidance between objects. 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 real-time 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 real-time 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 real time. 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 real-time.

The additional challenge in virtual reality (VR) 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. We see ourselves faced with a number of problems that need to be addressed. Any motion must be physical within limits, a human (or other creature-like) avatar is subject to biological constraints, and all computations underly stern real-time requirements. Biological constraints are for instance joint limits that govern possible body posture. A human will not usually go into contortion to avoid an obstacle but maintains a comfortable posture even if walking through a path with obstacles.

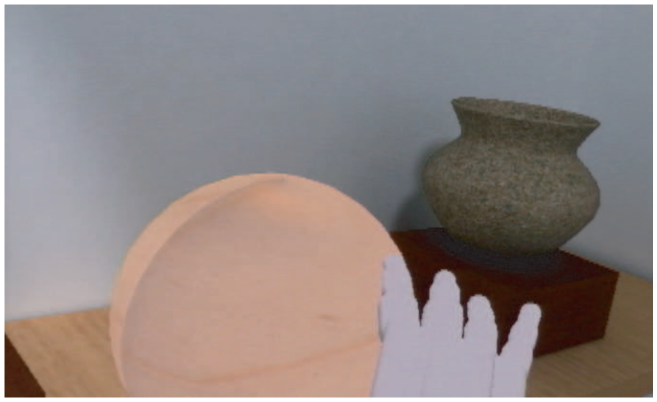
Figure 1: A ball is “snipped” over a table with a finger causing it to roll off the table.

Real-time performance is necessary in order to achieve good immersion into the VE. If the frame rate drops too low, a lag between the user's motion and the visual display will be noticeable. This would considerably lessen the realism with which our VE is perceived.

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 non-trivial and generally expensive for non-convex objects. Only impractical algorithms are known that would not satisfy our quest for real-time [8]. We use the convex PD to quickly estimate the non-convex PD. Combined with an iterative approach this 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 by an inverse kinematics (IK) ap-

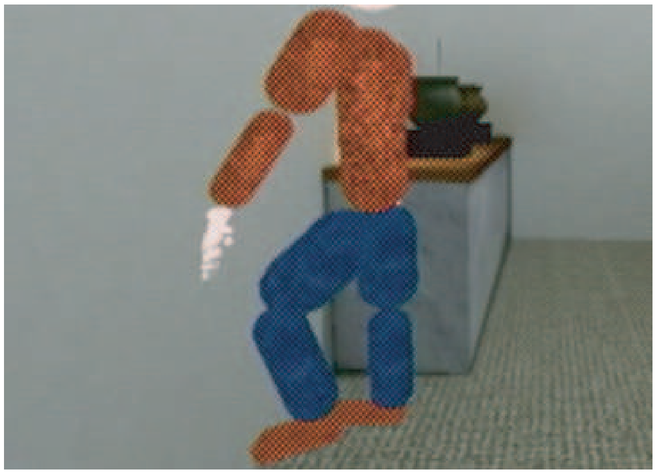
Figure 2: An avatar moves into a wall: the extended arm and a leg are pushed back.

proach utilizing cyclic coordinate descent (CCD) [22]. Collision response is finally handled with impulse-based physics. We chose a hybrid approach of kinematics and dynamics through geometric overlap minimization, CCD, and impulses despite of its inherent simplifications. Exact physics is not our main concern but real-time performance is. We found our system to stand the test of real-time performance and have demonstrated so by incorporating it into a VR application. We use a Cyberglove for interaction of an articulated hand with static and dynamic objects in a VE and a whole body avatar to demonstrate interaction with walls in the VE and handling of avatar self-collisions. Figure 2 shows interaction of a whole body avatar with a wall.

The remainder of this paper is structured as follows. Section briefly summarizes related work. We present the overall system architecture in section . Section introduces our heuristic for estimating non-convex PD quickly. Section presents the hybrid approach of kinematics and impulses for overlap avoidance and collision handling. Section describes and analyzes experiments that we have conducted and provides also some implementation detail. We finally close this paper in section with conclusions and a look at possible future work.

Previous Work

We summarize previous and related work. A paper by Arnaldi *et al.* [1] presents a good introduction into the topic of kinematic and dynamic animation of characters and discusses pros and cons found in each. The authors state that dynamics is especially beneficial if the quality of generated motion is of concern, as is the case for walking or handling collisions. Kinematics is stated to be applicable for grasping tasks or sitting on a chair where joint angles and obstacle avoidance are of concern.

Boulic *et al.* [4] talk about 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.* [3] introduce inverse kinetics, which considers mass distribution besides traditional IK.

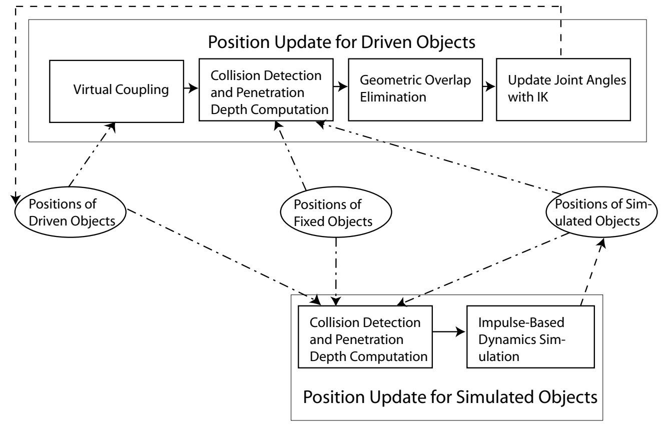
Position Update for Driven Objects
Virtual Coupling
Collision Detection and Penetration Depth Computation
Geometric Overlap Elimination
Update Joint Angles with IK
Positions of Driven Objects
Positions of Fixed Objects
Positions of Simulated Objects
Collision Detection and Penetration Depth Computation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

Figure 3: The architecture of our system.

This approach can be useful for posture control and keeping a synthetic actor's balance. Baerlocher and Boulic [2] 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.* [18] 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.* [21] 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.

Overview of System Architecture

In this section, we give an overview of our proposed system architecture for modeling physically-based 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. All unit joints are planar, *i.e.* they allow a rotation around one axis. This 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 over the architecture of our system. Refer to figure 3 for better understanding. Our algorithm first finds new and non-overlapping positions for all driven objects. This is described by the lower loop in figure 3. Virtual coupling attracts the driven objects to the tracker, collision detection and geometric overlap elimination find plausible end effector positions, and IK finds suitable joint angles. This process loops until all driven objects are updated without overlap.

According to the upper loop, simulated objects move then for the amount of time required for each frame under non-penetration constraints and according to other laws of physics. If collision is detected, the forward motion stops, impulse-based simulation resolves the collision, and object motion continues.

Both processes have as input all object positions. However, only one type of object is updated at a time. First the driven objects, then the simulated objects. 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 present now the mechanism for resolving our non-penetration (or non-overlap) constraint. Good algorithms are known for collision detection [6, 7, 10, 9, 14, 17]. These algorithms have in common the tracking of closest features with Voronoi regions. A dual space extension to these algorithms exists which can efficiently compute the PD for convex objects [11]. In analogy to Voronoi tracking, 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 Minkowski sum's surface. This algorithm has been extended to handle non-convexity [13]. However, this extension has been found to perform poorly from a standpoint of computational efficiency.

General PD computation can be performed based on explicit Minkowski sums. However, explicit computation of the Minkowski sum can be of cost $O(n^6)$ in the worst case [8]. Furthermore, the resulting algorithms are generally known to be subject to robustness problems. More efficient hardware-based approaches for estimating the general PD are known [12]. However, these approaches are best suited for small contact areas with many contacts and high detail. The simulations in which VR applications are interested have generally relatively simple contact geometry but necessitate keeping track of contacts over large areas. Hence, we chose to use PD computation for piecewise convex pairs and estimate the global PD through it.

We assume a convex decomposition of all objects [9, 11] and knowledge of the PD per pair of convex pieces. For two overlapping, non-convex objects we have i convex pairs and associated PDs, pd_i, and normals, $\mathbf{n}_i$. We estimate the general $\mathbf{PD}_g$ with:

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

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.

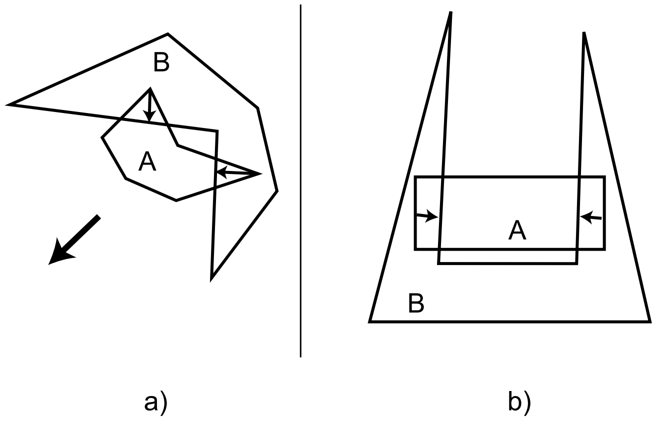

Figure 4: Finding the general PD: a) a “reasonable” case, b) a pathological case.

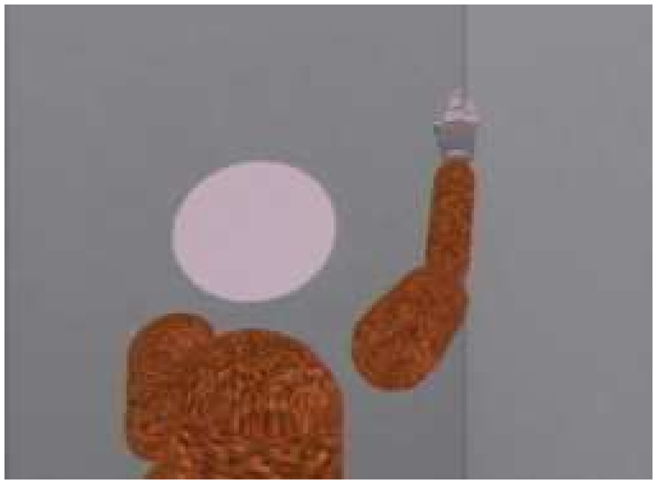

Figure 5: An articulated VR hand slides along a non-convex corner between two walls in a virtual room.

The described approach allows a quick estimate for the global PD but can be made to fail for certain scenarios. For an illustration see figure 4. Case a) shows a “reasonable” scenario where overlap is not too deep and can be resolved by translation following our ideas from the last paragraph. Case b) is a more pathological case: the two normals cancel each other out.

An optimization algorithm that uses also rotation for diminishing overlap is available [15] but for our purposes too expensive. We strictly must achieve real-time performance. 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 moves objects back towards “where they came from” until the overlap is diminished.

Despite of its heuristic character, we found this approach to work well in practice without visual jumps or artifacts, and it performs excellent under the criterion of real-time performance. See figure 5 for an example of an articulated hand and arm sliding along a corner formed by two walls in a virtual room. 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: Estimate the global PD for a general object.

Hybrid Approach of Kinematics and Dynamics

This section explains our hybrid approach to handling overlap and collisions for articulated bodies. This task falls into two categories: geometric and kinematic overlap avoidance, and handling resulting collisions and contacts by application of impulses. We first explain how to find plausible positions by removing all overlap.

Geometric and Kinematic Overlap Avoidance

The simplest form of resolving overlap for just one pair of objects is by pushing the objects apart by their amount of interpenetration. We found that even a complicated object, such as an articulated hand, can be handled in this way. We can find a plausible position for the hand by simply displacing it by $\mathbf{PD}_g$ according to equation 1.

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 is not possible we also move the simulated ones.

For a whole arm we can no longer use simple geometric overlap avoidance but have to revert to a more sophisticated method. We found a combination of a geometric and a kinematic approach to work well. Inverse kinematics can find joint angles that place a kinematic chain's *end effector* at a desired *goal* location.

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 [24]. We chose to employ cyclic coordinate descent (CCD) due to its simplicity, speed, and for our application good results.

CCD was introduced by Wang and Chen [22] and can be viewed as a stepwise optimization algorithm. Joints are optimized one at a time until we arrive at a global minimum, are close enough, or it was decided the minimum cannot be reached. Despite of its heuristic character we found CCD to produce pleasing results.

The principle of CCD is explained in figure 6. 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:

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

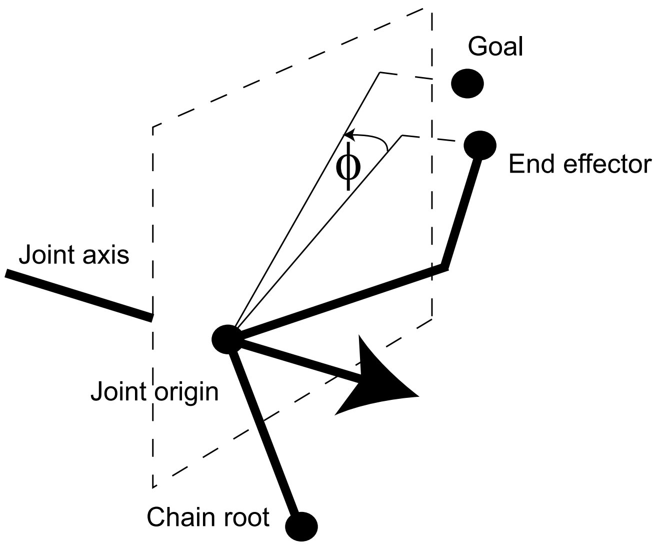

Figure 6: Explanation of the CCD algorithm: the joint would have to be rotated by ϕ in order to minimize the distance to the goal.

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

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

In this equation, $\mathbf{a}_h$ 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 tolerance. 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 to. Using CCD this way allows overlap resolution also for self-colliding chains, *e.g.* right arm 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 [23] 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 has been described as coach-trainee metaphor [4].

Impulses

Having found non-overlapping positions with purely geometric overlap avoidance and IK, we can now apply impulses to make dynamic objects bounce off the avatar and the environment realistically. We chose impulse physics [16] for collision response. Despite of its shortcomings for systems with large numbers of collisions, it is well suited for handling virtual contacts for VEs. We are not interested in large, crowded clusters of objects with many collisions, but rather localized interaction between the human user and the environment. Impulse physics achieves good level of realism and superior running time behavior 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 [15, 20]. 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 $(v^{\perp} < 0)$ [5, 16, 15, 20]. 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 [20].

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.

Adding friction

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 [20].

Implementation and Results

We will now discuss and analyze experiments conducted. Implementation data is also provided. Our model simulates an art gallery with two rooms and several objects in them. This model was provided by UNC's EVE Group. Our algorithms were implemented on a 1.7GHz Intel Xeo machine. The articulated hand uses a Cyberglove. Tracking is done with a UNC HiBall and ceiling tracker. Distance computations are performed with SWIFT++ [9] and DEEP [11]. 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/.

We created a number of scenarios that show the performance of our system. The environment used for all scenes has about 20 objects, some of which are composed of triangles ranging in the hundreds. The articulated hand consists of 15 elliptical parts with triangles numbers around 100. The hand interacts with the environment by touching objects and sliding along walls and the table, but also by playing with a ball. The hand is assumed rigid for purposes 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.

It became clear that fast collision handling is, as expected, essential for real-time performance. Pairwise collision queries are handled after a sweep and prune routine [7] is used to filter out the object pairs for which exact collision query is not necessary. Figure 7 shows the frame time. The first half of the graph shows 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

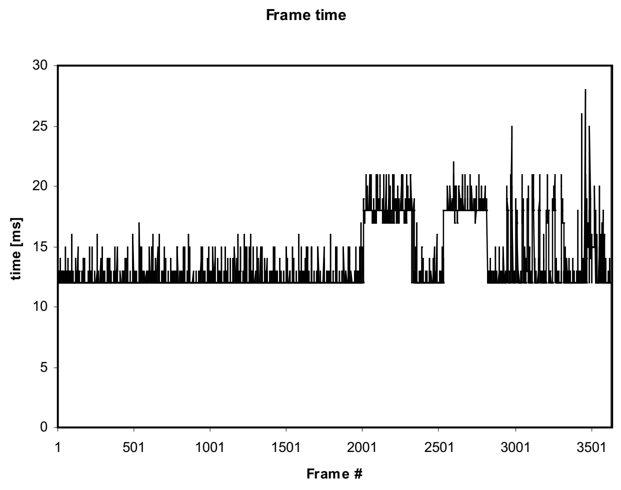

Figure 7: Frame time during a simulation of 3653 frames.

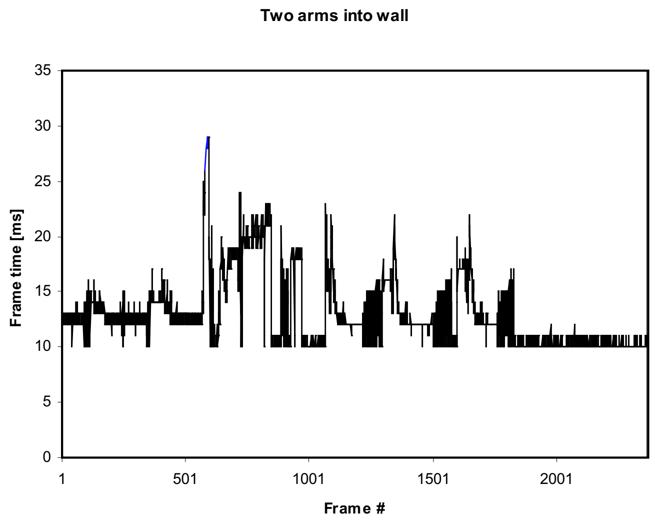

Figure 8: Frame time during the *two arms* scene.

sliding over a table, first in one 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 as collisions occur.

We achieve overall real-time performance with varying frame rate between 30Hz and 60Hz. In our particular scenario, overlap elimination 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 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 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 done, consecutive frames have a lower number of iterations and the total cost per frame is reduced. Figure 8 shows three spikes in the second half of the plot where the frame time rises rapidly but falls off rather quickly. This was generated by the avatar making a sequence of pushes into the wall. The spikes correspond to solving the initial interpenetration with the following falloff where the number of necessary iterations is gradually becoming less.

There is a similar correlation for the number of contacting pairs and the part that collision detection plays in the total frame 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

We close this treatise with conclusions and a look at possible future work. Our approach merges physically-based and kinematic methods. Input is given by a human user and visible motion is as close as possible to it. IK is only used for overlap avoidance, not for generation of the whole motion. We can handle self-collisions and achieve real-time performance. The latter is important for VR applications but also games. The general PD is estimated efficiently and sufficiently accurate through pairwise PD for convex pieces. Despite of these simplifications we find collision queries to be a bottleneck in our system. It also cannot be excluded that collisions are missed due to discreet collision testing. We plan to include continuous collision detection in the future.

Acknowledgement

The authors wish to thank Eric Burns for help with implementing the framework for tracker reading and using the Cyberglove.

References

[1] 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.

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

[3] 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.

[4] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figures motion

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.

editing. *Computer Graphics Forum*, 11(4):189–202, 1992.

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

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

[7] 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.

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

[9] 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.

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

[11] 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.

[12] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation using rasterization hardware and hierarchical refinement. Technical Report TR02-014, Department of Computer Science, University of North Carolina, 2002.

[13] 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.

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

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

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

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

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

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

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

[22] 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.

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

[24] 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.