The God-Finger Method for Improving 3D Interaction with Virtual Objects through Simulation of Contact Area

Anthony Talvas* INSA/Inria Rennes, France
Maud Marchal* INSA/Inria Rennes, France
Anatole Lécuyer* Inria Rennes, France

ABSTRACT

In physically-based virtual environments, interaction with objects generally happens through contact points that barely represent the area of contact between the user's hand and the virtual object. This representation of contacts contrasts with real life situations where our finger pads have the ability to deform slightly to match the shape of a touched object. In this paper, we propose a method called god-finger to simulate a contact area from a single contact point determined by collision detection, and usable in a rigid body physics engine. The method uses the geometry of the object and the force applied to it to determine additional contact points that will emulate the presence of a contact area between the user's proxy and the virtual object. It could improve the manipulation of objects by constraining the rotation of touched objects in a similar manner to actual finger pads. An implementation in a physics engine shows that the method could make for more realistic behaviour when manipulating objects while keeping high simulation rates.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Direct Manipulation

1 INTRODUCTION

In real life, when we touch an object or pick it with our fingers, they undergo slight deformations to adapt to the shape of the object and form a contact area between finger and object. This contact area modifies the frictional properties of the interaction between the two, and overall leads to a different behaviour. An example of this is attempting to push a sheet of paper by the corner with the tip of a sharpened pencil or with a fingertip. With the pencil tip, the paper will most likely rotate, while with a fingertip it will most likely translate. In the case of interaction with virtual environments (VE), these areas of contact are most often not simulated at all, contacts between the user's proxy and virtual objects being represented by points with a normal and penetration depth. The rotations of touched objects are thus barely constrained, making their manipulation more difficult. Virtual hands with deformable finger pads have been proposed to solve this problem, but these methods are more computationally expensive in return [10, 5].

We thus propose a *god-finger* method that uses the usual contact point information to generate contact areas by scanning the local geometry of a touched virtual object (Figure 1). A contact area is generated by using the magnitude of the force applied to the object, the direction of that force and by taking into account the normals of the faces surrounding the contact point, which simulates the interaction through a finger pad. This area of contact improves the manipulation of virtual objects by adding additional constraints to the rotations of touched objects, notably around the contact normal. This technique is usable with 3 degree-of-freedom (DOF) interfaces, and is intended to increase the realism of the touching of virtual objects, as opposed to other works that focus on improving their grasping with more than one finger. The method can be implemented in any physics engine, and is notably usable in the case of haptic interaction with VE.

*e-mail:{anthony.talvas,maud.marchal,anatole.lecuyer}@irisa.fr

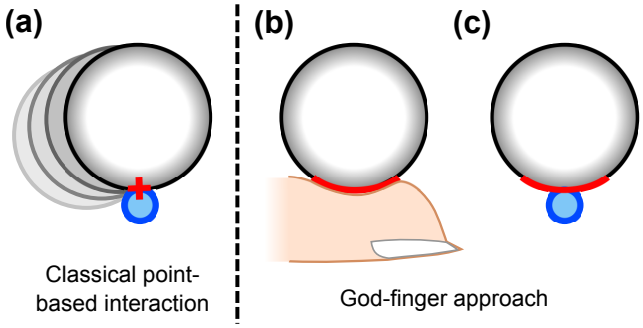

Figure 1: Concept of the *god-finger* method. **(a)** Contacts in virtual environments are represented by points. Lifting a virtual object that way causes it to fall. **(b)** In real life, human fingers generate a contact area with objects through deformation. **(c)** The *god-finger* method simulates that contact area from a single contact point. It is now possible to lift the object without it falling, thanks to the contact area.

2 RELATED WORK

Approaches for interacting realistically with physically-based VE can be classified in two main categories: point-based and rigid proxy-based approaches on the one hand, and deformable hand model-based approaches on the other hand.

Point-based interaction allows to touch virtual objects through a single contact point. The *god-object* method by Zilles and Salisbury [12] pioneered 3DOF constraint-based haptic interaction with virtual objects. It defines a location in virtual space, called the *god-object*, that follows the haptic interface virtual position, but constrained at the surface of virtual objects when a contact occurs. Several works focused on adding static and dynamic friction to the method [11, 3, 8] by only moving the god-object on the surface of the virtual object when it leaves a friction cone defined by the friction coefficient of the object. Ortega *et al.* extended the method to 6DOF interaction, allowing the use of complex rigid bodies as proxies, with any number of contact points with virtual objects [9]. Other works focused on handling the translation and rotation of virtual objects contacted with multiple god-objects or proxies [7, 4], and a generalized *god-object* method was recently proposed to take into account the mutual interdependencies between multiple *god-objects* in the context of a *god-hand* model [6].

In all of these methods, however, contacts are represented by single points, and there is no simulation of any contact area between the virtual representation of the user's hand and virtual objects. In order to account for this contact area, Barbagli *et al.* studied human fingertips to derive a soft-finger model that simulates the resistance

to moments around the axis between two proxies grasping an object [1]. Still, while this method partly simulates the picking of objects between fingertips, it does not account for effects imposed by the local geometry around the contact points. In order to effectively simulate the contact area between the user's proxy and virtual objects, soft hand models were proposed. These models use deformable elements under a rigid skeleton to simulate the softness of the finger pads, either using a Finite Element model with a corotational approach [10] or the Lattice-Shape Matching algorithm with adaptive stiffness [5]. These methods allow realistic and stable grasping of virtual objects, notably, but at the cost of a higher computational load due to deformable body simulation.

In this paper, we propose a *god-finger* method that extends the *god-object* approach to improve 3DOF single-point interaction by providing a contact area with virtual objects, without resorting to deformable body simulation. We first present the method, then showcase a few results obtained with an implementation in a physics engine with haptic interfaces, and finally conclude with perspectives for the method.

3 The God-finger Method

We propose a *god-finger* method to simulate the contact area between a god-object and virtual objects. The *god-finger* method consists of two major parts. First, from the contact point between the *god-object* and the virtual object, several radial vectors are computed, that describe the contact area around the contact point as it would be if the contact spread on a flat surface (Figure 2a). Then, we use these vectors to travel the geometry of the object, taking into account its possible angles and curvatures (Figure 2b). However, as human fingers can not deform infinitely, we consider that the contact area can not spread on faces whose normals are too different from that of the original contact point (Figure 2c). We define the points resulting from the local geometry search as *sub-god-objects*, that are fed back to the simulation engine as additional contact points, which will simulate the presence of a contact area.

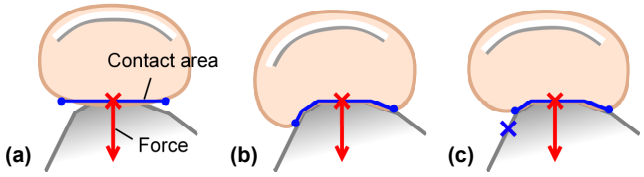

Figure 2: Steps of the *god-finger* method. **(a)** A contact area is generated as if the object was planar. **(b)** The contact area is then fit to the geometry of the object. **(c)** Odd deformations are prevented by adding an angular threshold between the contact normal and the face normals.

Once a contact between the god-object and an object has been detected by the collision detection engine, the contact point c_p, contact normal c_n and contact distance c_d are fetched, in local coordinates of the collided object. It is also determined whether the collision occured on a face, edge or vertex of the object's mesh.

3.1 Radial vectors and distances

The *god-finger* method starts by generating vectors that define the contact area as it would be if it spread on a planar object. The starting point is the determination of the desired contact radius. It depends on the force applied by the *god-object* to the virtual object, which is in turn proportional to the distance between the position of the interface and that of the *god-object* due to the virtual coupling. Barbagli *et al.* performed experimental measures of the contact area of the human finger and derived a model between the normal force and the contact radius [1]. We thus use that model to determine the contact radius r from the distance between the *god-object* position $\mathbf{p_{GO}}$ and the interface position $\mathbf{p_I}$ (expressed in local coordinates), considering an arbitrarily chosen maximum radius r_{max} and distance constant D'_0. Considering $\mathbf{v} = \mathbf{p_I} - \mathbf{p_{GO}}$ as shown in Equation 1.

$$r = r_{max} \times \left(1 - e^{-\mathbf{v}.c_n / D'_0}\right) \tag{1}$$

Then, given a number n_{SGO} of desired *sub-god-objects* (at least 3), we generate n_{SGO} unit vectors orthogonal to the contact normal, defined as the radial vectors $\mathbf{r_i}$ (Figure 3a). If the contact occured on a face, then the radial vectors can be used as is for the following computations. If it occured on an edge or vertex, they need to be projected onto the surrounding faces (Figure 3b). We also define an angular threshold τ_a beyond which we consider that the contact area can not spread anymore over the surface of the object. Thus, if the angle between the contact normal and one of the surrounding faces normals exceeds that threshold, the corresponding radial vectors are dropped.

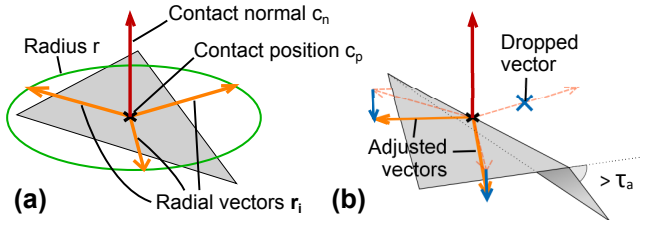

Figure 3: Generation of radial vectors from the contact point (black cross) and normal (red arrow). **(a)** Contact point on a face, the radial vectors (orange arrows) can be used as is. **(b)** Contact point on an edge or a vertex. For faces with normals under a threshold τ_a, the vectors are projected on the faces (blue arrows). For the other faces, the corresponding vectors are dropped (blue cross).

When the human finger applies a force on an object that is tangential to the object, the contact area tends to spread towards the opposite direction of that force (Figure 4a). In order to simulate this behaviour, for each radial vector r_i and with an arbitrary factor α determining how much the contact area will deform, we compute an adjusted distance d_i over which we will travel the geometry of the object, following Equation 2. The result of these adjustments are displayed in Figure 4b, with $\alpha = 2$.

$$d_i = r \times \alpha^{-\frac{\mathbf{v}}{\|\mathbf{v}\|}.\mathbf{r_i}} \tag{2}$$

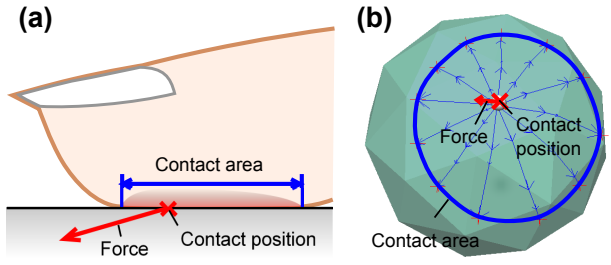

Figure 4: Deformation of the contact area when the force applied to the object has a tangential component. The applied forces are represented by red arrows and the contact areas are represented in blue. **(a)** Example of a human finger. **(b)** Contact area with a virtual object. The contact force is represented by the arrow, and contact area by the ellipse.

3.2 Sub-God-Objects

Once the radial vectors and distances are obtained, the next step is to travel the geometry of the object to find the positions of the *sub-god-objects*. In order to make the search efficient, the neighbor information is precomputed for the meshes of all virtual objects when they are loaded into the simulation. This means for a triangular mesh that, for every triangle, the indices of the three neighboring triangles (*i.e.* those that share two vertices with the considered triangle) are stored. The method thus works for both convex and concave virtual objects.

For each radial vector r_i and distance to travel along that radial vector d_i, the position of the corresponding *sub-god-object* p_{SGO_i} is computed following Algorithm 1. The search starts at a *sub-god-object* position p_{SGO} equal to the contact position c_p, on a face f with normal f_n determined previously. A distance d_i is crossed along r_i while checking if one of the edges of the triangle was reached. If no edge was reached, then the *sub-god-object* is located on the same face, at the end of the vector we followed. If an edge was reached, however, the *sub-god-object* is positioned on the point of the edge that was reached. Then, it is checked if the normal of the neighboring face is within the angular threshold τ_a defined previously. To take into account friction, it is also checked if it lies within the friction cone of friction coefficient θ. If both conditions are verified, then the geometry search continues, by setting the current face as the neighboring face, and projecting the radial vector r_i on that new face. This process loops until the full distance was travelled, or until a face with a normal exceeding the angular threshold is found or when the normal is no more in the friction cone.

Algorithm 1 Computation of a single SGO

$\mathbf{p_{SGO_i}} \leftarrow c_p$
while $d_i > 0$ and $cos^{-1}(\mathbf{f_n}.\mathbf{c_n}) < min(\tau_a, tan^{-1}(\theta))$ **do**
  $\mathbf{r_i} \leftarrow$ projection of $\mathbf{r_i}$ on f
  if $d_i \times r_i$ crosses an edge e of f **then**
    $\mathbf{p_{SGO_i}} \leftarrow$ point reached on e
    $f \leftarrow$ neighboring face
    $f_n \leftarrow$ neighboring face normal
    $d_i \leftarrow d_i -$ distance between new and old $\mathbf{p_{SGO}}$
  else
    $d_i = 0$
    $\mathbf{p_{SGO_i}} = \mathbf{p_{SGO_i}} + d_i \times r_i$
  end if
end while

Once the *sub-god-object* positions are obtained, they are provided to the simulation engine as additional contact points with the same contact normal and distance as the "true" contact point originally detected by the collision detection engine. The simulation will then act as if the virtual object has collided with a finger that deformed to match its shape.

4 Implementation

The method was implemented in Havok Physics, but can be implemented in any other common physics engine. Virtual objects with concave meshes were represented as collections of convex bodies in the simulation, notably using convex decomposition. Interaction scenarios were tested with two haptic devices: a Falcon (Novint Technologies Inc., Albuquerque, New Mexico, USA) and a Phantom Omni (Sensable Technologies, Wilmington, Massachusetts, USA). The Falcon is a 3DOF device, and the Phantom was used as such. The haptic devices and corresponding virtual proxies were linked through a virtual coupling mechanism [2], simulating a spring-damper between them.

4.1 Performances

The implementation of the method was performed on a 2.2 GHz quad-core PC, with both haptic devices directly connected to it. The visual rendering was fixed to a 60 Hz rate, and both the simulation rates and haptic device update rates were fixed to 1000 Hz, which is a suitable rate for haptic interaction.

4.2 Unimanual manipulation

Without the *god-finger* method, simply making a spherical object slide and rotate on a flat surface with a single interface can be a tedious task. Notably, the object has a strong tendency to roll as soon as any force is applied, hence simply translating it while keeping the rolling minimal is near impossible. It also tends to amplify the rotations applied to it, making it also difficult to apply small rotations to it. The *god-finger* method allows to better constrain the object, limiting the amount of erratic rolling while still allowing to perform controlled large rotations, and also makes it easier to let it slide along a surface (Figure 5).

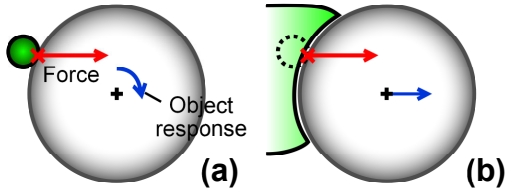

Figure 5: Responses of a virtual object with the regular *god-object* method (a) and with our novel *god-finger* method (b). **(a)** *God-object* case, the object rolls. **(b)** *God-finger* case, the object slides.

Another task made possible only with the *god-finger* method is that of lifting objects with only one interface, when the contact area is sufficiently large compared to the size of the object (Figure 6). It is not a trivial task as moving too fast with the interface will cause the god-finger to drop the object. However, this behaviour could be considered realistic, as for instance a hand can not be moved too fast with an object on the palm without dropping it.

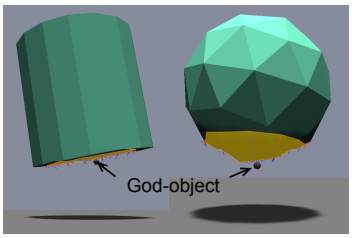

Figure 6: Lifting of virtual objects using only one *god-finger*. The contact areas are represented in yellow.

4.3 Bimanual manipulation

Objects can be lifted with two interfaces as well with the *god-finger* method (Figure 7). They can be taken, just like previously, from under the object, and it is also notably possible in some cases to drop one of the interfaces during the lift to switch to a unimanual lifting scenario.

Objects can also be picked two-handedly from the sides. The use of *god-finger* instead of regular *god-objects* allows to restrain the torques around the contact normals, thus allowing to pick an object up easier without resorting to unnaturally high friction values. However, a side effect is that torques around the other rotational axes are also constrained, making it slightly more difficult to perform small rotations of a picked object with *god-fingers*.

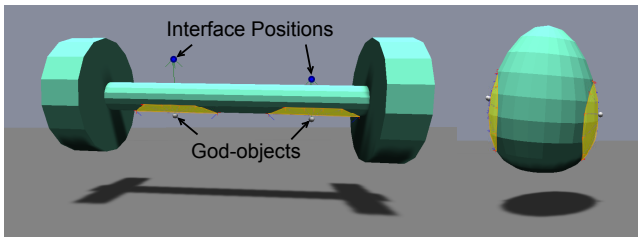

Figure 7: Lifting of virtual objects using two *god-fingers*. The contact areas are represented in yellow.

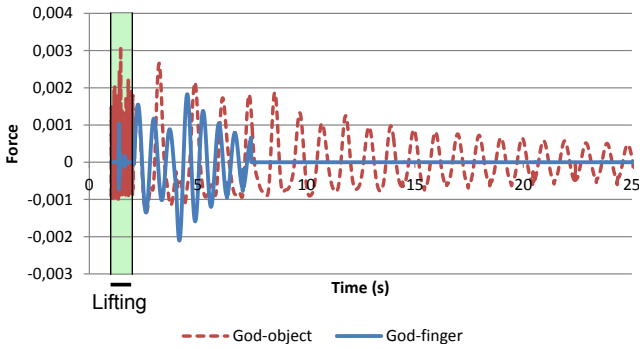

Figure 8: Force (y component) applied on the center of mass of a cylindrical object picked and lifted by two *god-objects* or two *god-fingers* along the x axis, slightly under the center of mass.

The classical *god-object* method and our novel *god-finger* method were directly compared. A cylindrical object was picked by two *god-objects* or *god-fingers* that followed the same motion. First, a fully horizontal motion brings both interface positions inside the object, leading both proxies to pick the object slightly under the center of mass. A vertical motion then causes both proxies to lift the object for 2 seconds, before stopping their motion completely. The force applied on the center of mass of the object was measured for 25 seconds, and the results are shown in Figure 8. With regular *god-objects*, there is no resistance to torques around the contact normals and the object keeps oscillating even longly after the lifting is done. With the god-finger method, the object never rotates around the contact normals, the oscillations measured at the beginning being only translational and due to the spring-damper of the virtual coupling.

4.4 Discussion

The *god-finger* method shows promising results for interaction with rigid bodies. Notably, while a regular *god-object* can make the manipulation of virtual objects with 3DOF interfaces challenging, a *god-finger* is more suitable as it makes objects more compliant while manipulating them. Moreover, a *god-finger* presents the advantage over a soft finger model that it works with 3DOF interfaces, while the latter requires 6DOF to orientate the virtual finger. Future work could focus on adding 6DOF interaction through the technique and compare such interaction to soft fingers.

The technique could also be used for interaction with deformable bodies. Notably, aside the improvement of manipulation capability, the visual feedback of a *god-finger* on a deformable object could be improved by giving the looks of a finger pad touching the object instead of a pointy end. Adding new *sub-god-objects* regularly during the local geometry search would create a more homogeneous repartition of contact points and generate a better deformation for large contact radii. It should not, however, be expected to be as homogeneous as the contact area generated by a deformable finger.

Bimanual manipulation is also improved through the *god-finger* method. However, the information of multiple god-fingers is computed completely independently. Notably, this may lead to overlapping of the contact areas, which is not desirable, but does not seem to hinder the manipulation itself. A possible way to handle overlapping is to apply force feedback on both devices to push them apart and reduce the overlap area. Furthermore, the question of how to integrate the information of multiple *god-fingers* to improve the grasping of virtual objects could be studied.

Finally, the method remains to be evaluated, notably in more complex scenarios, in order to assess its impact on the ease of object manipulation and on the realism of interaction. The *god-finger* method is expected to be less computationally expensive than soft hand models, due to the fact it uses point *vs.* mesh collision detection instead of mesh *vs.* mesh, and the fact it does not use FEM-based deformable simulation. However, the performance cost compared to both the *god-object* and soft hand methods still needs to be evaluated.

5 Conclusion

We have proposed a *god-finger* method for simulating contact area using contact information obtained through classical *god-objects*. The contact area is generated by scanning the local geometry of the object over a radius depending on the force applied on the object, and stopping at positions that would lead to unnatural deformations or loss of friction. The method allows more realistic manipulation of objects while remaining simple to implement in common physics engines. Since it does not rely on deformable elements, it has a low computational cost and allows to keep high physical simulation rates. The technique was illustrated with haptic scenarios but is not limited to them, and can be used to improve 3D interaction with any kind of interface, and any number of fingers involved.

References

[1] F. Barbagli, A. Frisoli, K. Salisbury, and M. Bergamasco. Simulating human fingers: a soft finger proxy model and algorithm. In *Proc. of HAPTICS*, pages 9 – 17, 2004.

[2] J. Colgate, M. Stanley, and J. Brown. Issues in the haptic display of tool use. In *Proc. of IEEE/RSJ IROS*, volume 3, pages 140–145, 1995.

[3] W. S. Harwin and N. Melder. Improved haptic rendering for multi-finger manipulation using friction cone based god-objects. In *Proc. of Eurohaptics*, 2002.

[4] D. Holz, S. Ullrich, M. Wolter, and T. Kuhlen. Multi-contact grasp interaction for virtual environments. *JVRB*, 5(7), 2008.

[5] J. Jacobs and B. Froehlich. A soft hand model for physically-based manipulation of virtual objects. In *Proc. of IEEE VR*, pages 11 –18, 2011.

[6] J. Jacobs, M. Stengel, and B. Froehlich. A generalized god-object method for plausible finger-based interactions in virtual environments. In *Proc. of IEEE 3DUI*, pages 43 –51, 2012.

[7] N. Melder, W. Harwin, and P. Sharkey. Translation and rotation of multi-point contacted virtual objects. In *Proc. of Eurohaptics*, 2003.

[8] N. Melder and W. S. Harwin. Extending the friction cone algorithm for arbitrary polygon based haptic objects. In *Proc. of HAPTICS*, pages 234–241, 2004.

[9] M. Ortega, S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):458–469, 2007.

[10] M. Pouliquen, C. Duriez, C. Andriot, A. Bernard, L. Chodorge, and F. Gosselin. Real-time finite element finger pinch grasp simulation. In *Proc. of WHC*, pages 323–328, 2005.

[11] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Proc. of 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 345–352, 1997.

[12] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *Proc. of IROS*, pages 3146–, 1995.