# Simulating Grasping Behavior on an Imaging Interactive Surface

**Andrew D. Wilson**
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
awilson@microsoft.com

**ABSTRACT**
We present techniques and algorithms to simulate grasping behavior on an imaging interactive surface (e.g., Microsoft Surface). In particular, we describe a contour model of touch contact shape, and show how these contours may be represented in a real-time physics simulation in a way that allows more realistic grasping behavior. For example, a virtual object may be moved by "squeezing" it with multiple contacts undergoing motion. The virtual object is caused to move by simulated contact and friction forces. Previous work [14] uses many small rigid bodies ("particle proxies") to approximate touch contact shape. This paper presents a variation of the particle proxy approach which allows grasping behavior. The advantages and disadvantages of this new approach are discussed.

**Author Keywords**
Interactive surfaces, game physics engines

**ACM Classification Keywords**
H5.2 [Information interfaces and presentation]: User Interfaces. – Graphical user interfaces.

**INTRODUCTION**
Much of the allure of interactive surfaces may be attributed to the style of direct manipulation made possible by large multi-touch displays and powerful graphics capabilities. Often these manipulations have a tangible quality, as if the virtual object being manipulated behaves as it would in the real world. The now ubiquitous two finger rotation, translation and scaling of photos, for example, loosely emulates the manipulation of a real photo by friction forces imparted by two fingers in contact.

Imaging interactive surfaces such as Microsoft Surface [9], FTIR-based systems [4] and others [10,11] use video cameras and computer vision techniques to obtain detailed information about the shape of the fingers, hands and other objects in contact with the screen. To date there have been few systems able to take advantage of the richness of this information however. Most multi-touch systems reduce each contact shape to a discrete point which is then hit-tested against onscreen objects in a way that is familiar to most software developers.

The ability to sense more information about hand pose, matched with the ability to animate onscreen objects in a realistic fashion, invites the simulation of more realistic behaviors on imaging interactive surfaces. Wilson et al [14] describe an interactive surface which incorporates precise contact shape information into a real-time gaming physics engine. Onscreen objects move as the result of friction forces and collisions with the contour of the hand or other physical objects. The proposed technique allows users to adopt a wide variety of manipulation strategies drawn from experience in the real world. For example, a rolling ball may be brushed lightly to cause a small amount of motion, while the sudden contact of multiple fingers will quickly stop a ball in motion. In their study, users exercised a variety of manipulation strategies not found in other systems, such as bumping a ball from the side and catching it with a cupped hand on the other side of the table. Importantly, none of these strategies require specific support in code; rather, they are the consequence of the physics simulation.

Wilson et al [14] cite the lack of support for grasping behavior as a disadvantage of particle proxy technique. That is, it impossible to move or "hold onto" an object by "squeezing" a virtual object with multiple fingers placed around it. This is a consequence of the "particle proxy" technique used in their work. This deficiency seems particularly important given the 3D nature of the virtual objects and physics simulation: users are primarily limited to manipulating an object by applying friction forces on its top faces and colliding with its sides.

In this paper we develop a variation of the particle proxy technique that supports the grasping of objects. It accomplishes this primarily by allowing the contour of


ITS '09, November 23-25 2009, Banff, Alberta, Canada

contact objects such as the hand or fingers to conform to the shape of the virtual object in grasp, much as your hands conform to an object that is being picked up.

In the following sections we discuss the nature of grasping, review the particle proxy approach, present our new variation of that approach, show interactions that are enabled by it and discuss some limitations of the approach.

## GRASPING

In the real world the ability to grasp an object is vital. Imagine interacting with the real world only applying friction forces or colliding with only one side of each object! Our grasping ability separates us from much of the animal kingdom, allows us to use tools and in a very concrete sense gives us the ability to rearrange the world around us. Grasping strategies vary widely and depend greatly on the task at hand [8]. The grasp used to pick up a jug of milk will be very different than that used to thread a needle.

An important feature of grasping with hands is the way that multiple fingers and the soft flesh of the hand conform to the shape of the grasped object. This would appear to make it easier to achieve a stable grasp: consider grasping an object by the tips of two pencils.

Clearly grasping is necessary to pick up an object and manipulate it in space. But in the case of a 2D interactive surface where objects cannot be lifted into the space above the surface, how important is grasping? Consider a typical work desk: when objects are rearranged, how often are they grasped and moved without leaving contact with the desk rather than picked up and moved?

For the purposes of the present work, we may consider grasping as the application of multiple resting contacts on the sides of an object. Each of these resting contacts imparts a normal force. In the case where the sum of all such forces is zero, "mechanical equilibrium" is obtained and the object does not accelerate. If the contacts move, the object may be caused to move to restore mechanical equilibrium, as when, for example, an object is rotated while grasped.

Friction forces play an important role in grasping. Because friction forces are proportional to the normal force (see Figure 1), it is possible to maintain the grip of a slippery object by squeezing it more. Similarly, an object may fall out of grasp if the grip is lessened.

By simulating grasping behavior at the basic level of concepts such as forces and friction, we hope to allow users to exploit the variety of grasping strategies that are familiar in the real world, and for system designers to avoid the need to program multiple interactions in an ad hoc fashion.

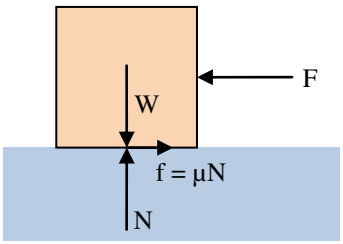



**Figure 1. An object exerts force of weight W. Friction force f opposes force F causing motion, and is proportional to the normal force N.**

## RELATED WORK

While there have been a number of interactive surface systems which have based interactions at least in part using physics concepts (see [14] for a review) there are very few (if any) that treat grasping realistically. More common is the programmatic manipulation of virtual objects (such as in [7]) where an object is rotated, translated and possibly scaled by the movement of contact points on the object. In these cases the object may be considered in the "grasped" state if any contact touches it.

The variety of grasping behavior motivates the dependence on shape information in ShapeTouch [1]. For example, the authors note that hand posture varies drastically with the task, and that even coarse models of shape, such as contact size, may be used to implicitly select the mode of manipulation.

The Responsive Workbench explored the use of multiple inputs with a physics engine in a 3D assembly task [3]. This system is notable for the way that objects are manipulated: virtual springs attached to the corners of a rigid cube are fixed to the object to be manipulated. The rigid frame is then moved with the user's input. Bimanual interactions are supported by superposition of forces from both hands. This combination of forces from multiple inputs is similar to the notion of grasping introduced above.

Grasping has been studied in great detail in the field of robotics, where a classic problem is how to drive a dexterous manipulator to pick up and hold a physical object ("grasp planning").

## PARTICLE PROXIES

The main idea of the particle proxy approach developed in Wilson et al [14] is to insert many small rigid bodies into a real-time physics simulation. These rigid bodies are termed "proxies" because they are positioned to closely approximate the shape of the surface contact (e.g., hand, finger, or other physical object), and so taken together are thought to be a proxy for the actual physical object (see Figure 2).

The particles are moved to match the local motion of the contact observed at the corresponding location in the input image. This is achieved by applying a force of the exact

magnitude and direction to effect the movement of each proxy.

**Figure 2. Particle proxies approximate the shape of multiple contacts. Left: applying friction from the top, and collision from the side to grip a block. Middle: long proxy particle objects (red) illustrated.**

Because each particle is a rigid body in the physics simulation, it participates in collision and friction calculations like any other rigid body. So, for example, when the contour of the hand moves to strike the side of an object, an appropriate collision response is made: the object may move. Similarly, the particles will generate friction forces that may cause the object under them to move.

There are a number of aspects of the particle proxy algorithm that are relevant to the present work.

### Generating particles

Particles are generated to match the contour of the contact. The input image is processed by a Sobel (gradient) operator which highlights the edges of objects on the surface. A particle is generated for every pixel in the Sobel image that is greater than some threshold intensity.

#### *Positioning particles*

Each particle must be positioned in the 3D world of the physics simulation. Because the input is fundamentally 2D and the virtual objects inhabit a 3D world, it is necessary to determine a 3D position for the particle. The position of the particle in the plane of the display is determined naturally from the input image coordinates. The out-of-plane height of the particle's position is determined by finding the first point of intersection with any virtual object by a standard ray casting operation. The particle is placed to be in resting contact with this point.

#### *Particle shape*

The shape of each particle's rigid body is limited only by the collision primitives supported by the physics simulator. A sphere is a good choice because it allows easy collision detection calculations. A capsule that extends all the way from graphics camera near plane to the intersected object is an interesting choice because it would collide with objects that lie above the original intersecting body, such as objects that are stacked, are airborne or have sides that are not flat (e.g., a ball). The capsule proxy may more closely meet users' expectations about how objects should collide with contacts.

#### *Advancing the simulation*

Rather than attempt to track the particle across multiple frames, the algorithm outlined in [14] suggests creating a new set of particles each frame. After the physics simulation is updated, all particles are removed from the simulation. This instantaneous feature of the algorithm avoids complex and error-prone tracking techniques, and fits well with the contour information contained in the Sobel image.

The particle proxy algorithm is summarized as:

```
compute Sobel image from surface input
for each pixel with high spatial gradient:
    raycast into scene to determine initial particle position
    add particle rigid body to physics simulation
    compute contact motion at particle (e.g., from flow)
    compute corresponding tangential motion in scene
    apply force to particle to match scene motion
    apply downward force (gravity) to particle
update physics simulation
destroy all particle rigid bodies
```

## GRASPING WITH PARTICLE PROXIES

The particle proxy approach summarized above does not support grasping primarily because of the instantaneous nature of the algorithm. Consider two finger contacts closing around a virtual object at rest. Each finger is represented by a cluster of particles, each created, positioned, moved, and destroyed every frame. At some point some particles will collide with the object and the object will move in response. Eventually there will be particles in contact on both sides of the object, and the object will have nowhere to go. At this moment we might consider the object to be under grasping control. But it is very likely that the user, in the absence of tactile feedback from the virtual object, will continue to move their fingers together. Shortly thereafter, the subsequently created particles will be placed on top of the object, in accordance with the raycasting placement strategy summarized above.

The net result is that while the user may be able to place particles that are in contact with the side of objects, these particles will never exert any significant normal force on the side of the object (and, subsequently, friction forces) that lead to true grasping behavior. The particles are replaced by new particles that are positioned on top of the object to be grasped. It is as if the fingers slide to the top of the object: the object literally slips out of grasp.

## PERSISTENT PARTICLE PROXIES

The alternative to creating a complete set of particles every frame is to create a set of particles that are preserved frame to frame. As we show in the rest of the paper, this approach

has the advantage of allowing particles to rest on the side of an object even when the corresponding contact has moved deep within the object. A collection of such points distributed along the contour of an input contact can thus conform to the shape of the object and lead to strong normal and friction forces needed for grasping.

The difficulty in this approach is that particles must follow the input contour, and particles must be added and removed to follow the changing size and shape of the input contact. In order to simplify the task of tracking an evenly sampled contour over many frames, we use an explicit contour representation (i.e., a linked list of points) rather than the previous approach of implicitly representing the contour by generating a particle for every bright pixel in the Sobel image. As discussed later, this loses some of the robustness features of the original particle proxy approach.

In the following sections we detail this "persistent particle proxy" approach.

## CONTACT MODEL

Imaging touch screens such as Microsoft Surface and others base the discovery and tracking of input contacts on observing connected components in a binary image that is the result of several image processing steps such as lens distortion correction, level normalization and so on. In this binary image, each "on" pixel corresponds to a tiny region of the screen that is being touched. The calculation of connected components groups these regions into one or more spatially distinct input contacts. Most often the center, bounding box or ellipsoid shape of each contact is made available to subsequent processing.

Wilson et al [14] argues for avoiding the connected component model of contacts on the grounds that it results in a poor representation of contact shape, and that the typical subsequent tracking of connected components is error-prone when the user adopts more complex postures such as the flat of the hand, the whole hand, or any of the myriad postures beyond the tip of the index finger touching the display.

In the persistent particle proxy approach we return to the use of connected components for ease of computation, but compute the exact contour of each connected component using the technique described in [2]. This technique finds all connected components and their contours in one pass of the image (see Figure 3).

Each connected component corresponds to a single input contact. Contacts may be tracked from to frame by matching the connected components found in the current frame to those found in the previous frame. A reasonable heuristic is to assign a connected component to the nearest component in the previous frame, within some threshold distance. Components in the previous frame that have no assignment after this process are marked as removed (the contact has been lifted off). Components in the current frame that have no assignment are marked as new (the contact has just touched the surface).

The use of connected components thus allows the detection and special handling of new input contacts. In particular, in the algorithm presented later, particles corresponding to new contacts are positioned in the scene using raycasting, while those associated with pre-existing contacts are not.

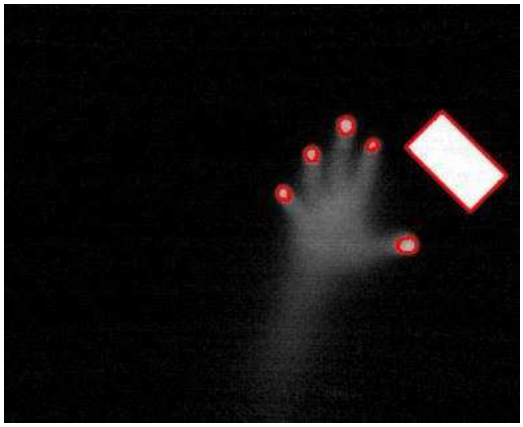


**Figure 3. Example image from diffuse-illumination table showing fingertips in contact and business card (face up) on the table. Extracted contours are in red.**

## KINEMATIC CONTROL

Before presenting the full persistent particle proxy algorithm we outline how rigid bodies may be correctly moved under programmatic control in a physics engine. This is necessary in our case to move a particle to match the position of a point on the contour of a contact.

At their most basic level rigid body physics simulators offer only the ability to apply forces and torques to a rigid body. Setting the position of a rigid body directly is generally not allowed, as this would defeat correct collision and dynamics calculations, as well as generate unrealistic object motion.

A common way to emulate kinematic motion by applying only forces and torques is to apply the amounts necessary to move the rigid body into a given position and orientation. These calculations assume the rigid body is in free space, and neglects the possibility of collisions and friction forces that might interrupt the object's motion.

Given the present (actual) position $\vec{x}$, the force `F` to move an object of mass `m` to position $\vec{x}^*$ is calculated by: $\vec{v}^* = \frac{\vec{x}^* - \vec{x}}{\Delta t}$, $\vec{a}^* = \frac{\vec{v}^* - \vec{v}}{\Delta t}$, $\mathtt{F} = \mathtt{m}\ \vec{a}^*$.

For the purposes of the present work, we need only control angular velocity. Given target angular velocity $\vec{\omega}^*$ and current angular velocity $\vec{\omega}$, apply torque $\vec{\tau}^* = \frac{\vec{\omega}^* - \vec{\omega}}{\Delta t}$.

## ALGORITHM

In this section we detail the persistent particle proxy technique.

As described above, the contact model is able to detect and allow for special handling of new contacts. New contacts are typically generated when the user first touches the surface with a finger, hand or other physical object. To begin the persistent particle proxy technique, each new contact is seeded by a number of particles that approximate the contour of the corresponding connected component. As with the original particle proxy algorithm, these new particles are placed in the 3D scene at (x,y) coordinates to match the position of the corresponding contour point in the input image. The height (z) of the particle is determined by raycasting to find the surface in the 3D scene nearest the graphics camera. Thus if the user touches a virtual object, the particles will be positioned to lie on top of the virtual object.

Existing contacts differ from new contacts in that there is a collection of particles already created to approximate the contact's contour in the previous frame. In the persistent particle proxy technique, this set of particles must be updated to reflect the change in size and shape of the contact's contour from the previous frame to the current frame.

Critical to this update is that particles should move in the same way that the corresponding part of the real object on the surface moves. This is necessary to generate correct friction forces when the particles are in contact with a virtual object, as well as to determine which parts of the contour are new or have been removed.

The motion of a given particle from the previous frame to the current frame is calculated by matching a small part of the contour in the previous frame with the contour in the current frame. Our current prototype uses exhaustive search to find the closest match in the new contour. A point $\vec{x}_{t-1,k}$ at the previous time step is matched to the point $\vec{x}_{t,j^*}$ by summing squared distance over $n$ consecutive points of the contours:

$$j^* = \operatorname*{argmin}_j \sum_{i=-n/2}^{n/2} \left\| \vec{x}_{t-1,k+i} - \vec{x}_{t,j+i} \right\|^2$$

Let us call the best matching point $\vec{x}_{t,j^*}$ on the current contour the previous point's "successor".

As this process continues around the contour, the best match to each point in the current contour is noted with that point. Let us call the current contour point's best match point in the previous contour that point's "predecessor". Note that every point in the previous frame has a successor but not every point in the current frame may have a predecessor.

If a point on the current frame's contour was assigned a predecessor its corresponding particle is marked for reuse: it will be propagated to the next simulation time step.

If the point was not assigned a predecessor, it is considered to be a new point (the contact evidently grew in size). A particle is created for this point and is placed at the position of the particle corresponding with the first neighboring point on the current contour not marked as new. The new particle is added to the physics simulation.

Every point on the previous contour is assigned a successor on the current contour. If that successor does not consider the same point on the previous contour its predecessor, its corresponding particle is marked for deletion and is removed from the physics simulation.

The result of this process is that every point on the current contour has a corresponding particle in the physics simulation that is either new or reused, and each particle is paired with a point on the current frame's contour in such a way as to approximate the local motion at each point of the contour.

As the last step of the algorithm, forces are applied to move the particles to match the position of the paired contour point. This is performed using the technique of kinematic control described earlier, but with one distinction: a constant force is applied in the (z) direction. Like gravity, this tends to keep the particles touching the top face of some virtual object in the scene. To ensure correct friction with the particles and avoid the possibility of spinning particles, the angular velocity of each particle is kinematically controlled to be zero. Finally, we note that the physics simulation should be configured so that particles are not allowed to collide with one another.

## GRASPING FORCES

As noted earlier, the above algorithm supports grasping behavior primarily because particles are preserved frame to frame. Consider a contact that moves to collide with the side of a virtual object. Because each particle is kinematically controlled and will have the same or similar height as its predecessor, particles would rather rest in contact with the side of a virtual object rather than appear on the top of the object. In this way the particles are allowed to conform to the shape of the object much as a real hand might.

Note that the procedure of initializing a new particle is different than the raycasting approach used with a new contact described above. This difference is key to the persistent particle approach: rather than have a new particle

appear on the top of the object, they appear with their neighbors.

Moreover, because the forces applied to kinematically position the particle in the (x,y) plane are proportional to the displacement of the particle from its matched input contour point, the normal force of a particle resting on the side of the object will increase as the contact moves further inside the object, with the attendant increase in friction forces. This allows the user to "press harder" or "squeeze" an object to maintain a firmer grip (see Figure 4).

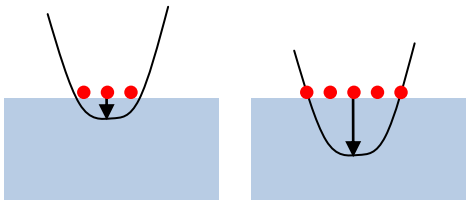


**Figure 4. The force applied to a particle is proportional to its displacement from the input contour.**

Of course, because the object is virtual, there is no real restorative force opposing the user's fingers. Consequently, there is nothing to stop the user from penetrating the object by an amount that cannot be reasonably modeled by the conforming behavior of the particle contours. While this appears to be fundamental limitation of the technology that is difficult to address, we note that the forces applied vary continuously with varying amounts of penetration, so that the subsequent behavior of the object is at least predictable.

## STACKED PARTICLES

In much of our experimentation we have been using spherical particle proxies. These are simple to work with, but can sometimes lead to unusual or unexpected results with certain kinds of 3D geometry. For example, sliding a contact into a virtual ball results in the much smaller particles moving under the ball until they finally hit the underside of the ball near the point at which it contacts the ground floor. Wilson et al [14] suggest using long capsules as proxies to address this problem, and we expect that this approach will work with the persistent particle proxy approach as well.

However, capsules and spheres have the limitation that in most circumstances they will contact a colliding object at just one point. When grasping an object, the height of the point of contact may vary throughout the grasping contours, thus causing forces that may cause the object to rotate or flip over unexpectedly.

In the present work we explore the idea of stacking multiple spherical particles at each point in the grasping contour (see Figure 5). In this scheme, the behavior of the bottommost particle is exactly as before, but the particles above it in the stack are kinematically controlled to lie at fixed heights above the bottommost particle. The result is analogous to using capsule proxies, except with the important distinction that because each particle in a stack is an independent rigid body, the stack conforms to the grasped object's change in shape along the (z) dimension.

The effect of stacked particle proxies during grasping will depend on the geometry of the object under grasp. An object with straight sides will behave no differently, but the ball mentioned earlier will be cradled by the proxies, probably resulting in a more stable grasp.

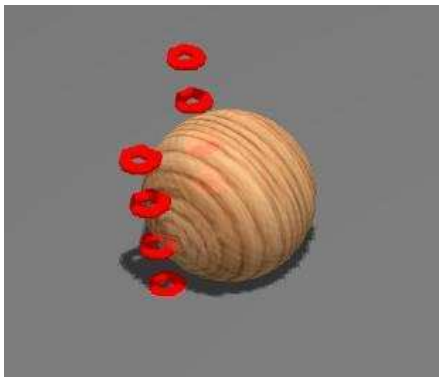


**Figure 5. Stacked particles generated from two contacts.**

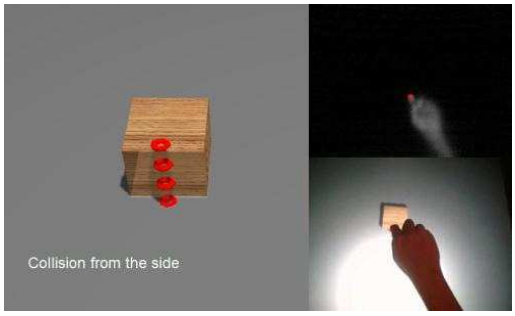
## IMPLEMENTATION

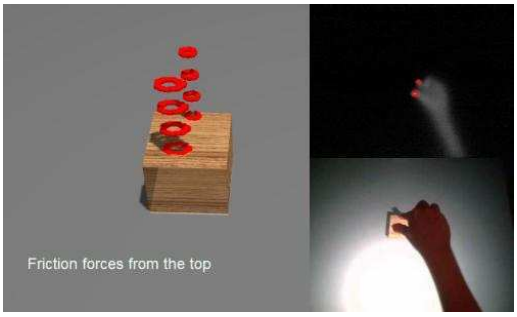
In our experimentation we have used a diffuse-illuminated interactive tabletop system similar to Microsoft Surface. The system's single video camera runs at 60Hz, with images of resolution 640x480. The table projects a 1024x768 image onto a 30" diagonal display. We use standard, well known techniques to correct for lens distortion, projective distortion, level normalization and binarization. For physics simulator, we use Newton 1.53 [6].

The system is driven by a desktop PC with an Intel Core i7 CPU. With the particle proxy approach, CPU consumption increases with more particles. The simple contour matching process is quadratic in the number of points on the contour. While the frame rate of the overall system will dip below the camera frame rate when there are many contacts or large contacts, there has been no concerted attempt to multithread or otherwise optimize the present software implementation.

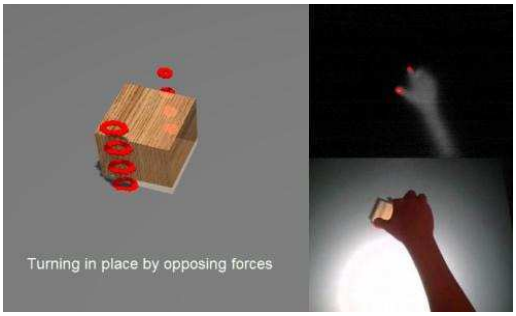
## INTERACTIONS

The persistent particle proxy technique supports the same collision and friction-based interactions of the original particle proxy technique. Collisions from the side cause the object to move, while touching the top of an object may cause it to translate and rotate (we invite the reader to view the accompanying video figure):

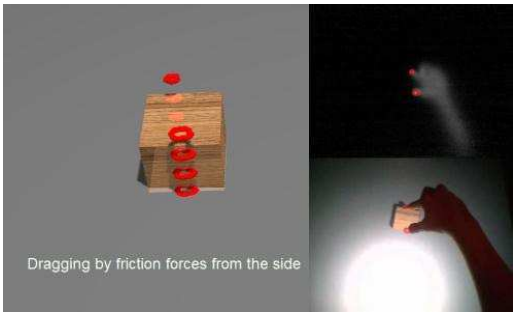


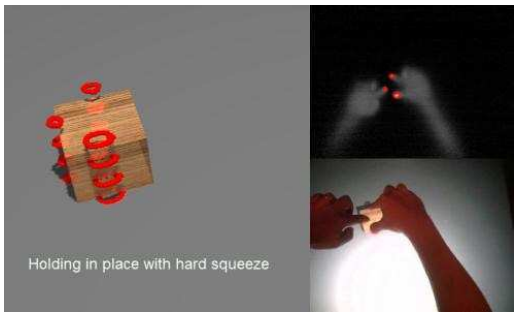

Several new grasping interactions are now possible with the persistent particle technique. Grasping from the side and turning in place sets up opposing forces which cause the object to turn in place:




Grasping from the side can translate the object by friction forces on the side of the object:

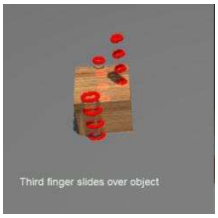


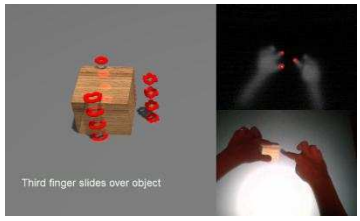

The same friction forces may prevent a colliding object (such as a third finger) from moving the object:



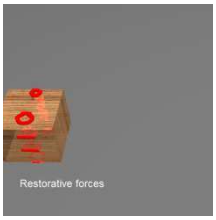

A lighter squeeze might have allowed the object to move.

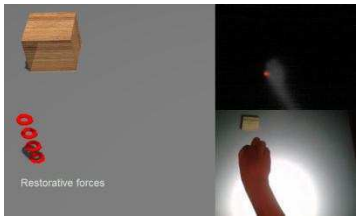
Likewise, a finger from the top may not impart enough friction force to move the object. In this case the finger will slide off the top of the object:




The conforming behavior of the particle proxies will enact a restorative force when an object is let go from grasp:




The "feel" of these interactions will depend on the values of the coefficients of static and kinetic friction, as well as the masses and moments of inertia of the grasped object. The grasp of a slippery or heavy object may be difficult to acquire.

## DISCUSSION

The persistent particle proxy approach addresses one of the main limitations of the particle proxy approach presented in Wilson et al. The present implementation trades away some of the features of the technique that make it robust to tracking and segmentation errors.

Much of the motivation in using connected components and their associated contours is the need to bootstrap new contacts into the simulation.

Another reason for using connected components is that the point matching process used to determine particle motion need only exhaustively search the contour of the tracked component, not all contours.

Unfortunately, errors in the contact tracking process can disrupt the grasping behavior of the persistent particle contour. Consider, for example, a grasping contact with its particles resting on the side of an object. If the contact tracking process confuses this contact with another, it may conclude that it is a new contact, triggering the deletion or relocation of all the contours particles and the placement of new particles on top of the object. The object may fall out of grasp.

If there is no need to support the manipulation of objects by friction forces from the top, it may be preferable to avoid tracking connected components, and instead place all particles on the "ground floor" of the scene, with the provision that new particles are placed so that they do not penetrate any object.

Perhaps an easy way to support top friction forces and grasping is to establish a limit on the penetration that grasping supports. For example, a contact contour that appears near the middle of an object (or is completely contained by the object) would be placed on its top face because to conclude otherwise would imply an unrealistic amount of penetration.

Ultimately these considerations arise from the fact that we are using a 2D sensor to interact with a 3D environment. Hence there is a need to infer (rather than directly measure) the height of a contact.

But we are also very interested in considering interactions that use 3D sensing technology, where the height of a contact may in fact be directly measured.

For example, Wilson [13] shows interactions using a range sensing camera and a co-located projected image. Hilliges et al [5] examine interactions above interactive surfaces, using two hardware configurations that allow sensing the space above the display, including a range sensing camera. In their work, they show a few grasping strategies: first using the technique in [12] to sense when the user brings their thumb and forefinger together. They explore a second approach of tracking fingers directly. 3D finger positions are intersected with rigid bodies in the scene: a five degree of freedom manipulation is enabled on a rigid body when two fingertips are detected to lie within object.

We believe that the persistent particle proxy technique can be extended to work with 3D input devices such as range sensing cameras. The main challenge will be in extending the point motion estimation process to work with meshes rather than contours.

## CONCLUSION

We have shown a variation of the particle proxy approach that supports grasping behavior on imaging interactive surfaces. In addressing one of the main drawbacks of the particle proxy approach, the proposed technique increases the fidelity of the simulation of the physics of interacting with a real object.

## REFERENCES


1. Cao, X., Wilson, A., Balakrishnan, R., Hinckley, K., and Hudson, S. 2008. ShapeTouch: Leveraging Contact Shape on Interactive Surfaces, *Third IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 129-136
2. Chang, F., Chen, C.-J., and Lu, C.-J. 2004. A linear-time component labeling algorithm using contour tracing technique, *Computer Vision and Image Understanding*, vol. 93, no. 2, 206-220.
3. Fröhlich, B., Tramberend, H., Beers, A., Agrawala, M., and Baraff, D. 2000. Physically-based manipulation on the responsive workbench. *IEEE VR Conference 2000*, 5-11.
4. Han, J.Y. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. *UIST '05*, 115-118.
5. Hilliges, O., Izadi, S., and Wilson, A. 2009. Interactions in the air: adding further depth to interactive tabletops, *UIST'09*.
6. Jerez, J., and Suero, A. Newton Physics Engine. http://www.newtondynamics.com.
7. Kruger, R., Carpendale, S., Scott, S., Tang, A. 2005. Fluid integration of rotation and translation. *CHI 2005*, 601-610.
8. MacKenzie, C.L., and Iberall, T. 1994. The grasping hand. Amsterdam, Netherlands: North Holland.
9. Microsoft Corporation. Microsoft Surface. http://www.surface.com. 2007.
10. Wilson, A. 2004. TouchLight: an imaging touch screen and display for gesture-based interaction. *ICMI '04*, 69-76.
11. Wilson, A. 2005. PlayAnywhere: a compact interactive tabletop projection-vision system. *UIST'05*, 83-92.
12. Wilson, A. 2006. Robust computer vision-based detection of pinching for one and two-handed gesture input. *UIST'06*, 255-258.
13. Wilson, A. 2007. Depth-sensing video cameras for 3D Tangible Interaction. *Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 201-204.
14. Wilson, A., Izadi, S., Hilliges, O., Garcia-Mendoza, A. and Kirk, D. 2008. Bringing physics to the surface, *UIST'08*, 67-76.