Interactive Cloth Manipulation With Multi-Touch Control

Yue Peng Toh∗  Nancy Pollard†

Carnegie Mellon University

ABSTRACT

We present an interactive system for intuitive manipulation of dynamic cloth using multi-touch technology. The system consists of two main modes that can be toggled anytime, namely cloth creation mode and cloth manipulation mode. In cloth creation mode, the user creates pieces of cloth by drawing each desired shape directly onto the scene using simple swipe gestures. The system automatically generates the best-fit cloth mesh from each drawn shape and renders each cloth on the scene. When cloth manipulation mode is selected, the user can manipulate cloth using a multi-touch pad by placing various fingers at desired locations of the cloth and moving them. Internally, the system seeks the nearest underlying cloth particles under each finger and sticks them to each active finger, while allowing the remaining cloth particles to readjust and move dynamically according to the underlying particle simulation. Our two key interaction techniques, sticky fingers and pinch-lifting, along with the use of Verlet integration and iterative constraint satisfaction, make it possible to control multiple cloth points simultaneously and in varying velocities, allowing the user to intuitively produce a range of complex cloth manipulations such as pinching, draping, tearing and folding.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.5.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.5.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1 INTRODUCTION

Manipulation and control of cloth in 3D scenes can often be non-intuitive and at times tedious because of its non-rigid nature. While many interactive cloth simulation systems allow for real-time manipulation of cloth by enabling the user to drag a single vertex or a subset of vertices using a 2D pointing device, this sort of dragging operation is often inadequate because only one section of the cloth can be dragged, in a single direction in 3D space at any one time.

In fact, for many useful cloth manipulations such as draping, tearing and folding, simultaneous control of different locations of a single cloth in varying directions is almost always mandatory. Precise bi-manual tearing of cloth for instance, involves pinning down sections of the cloth with a number of fingers of one hand, while moving fingers of the working hand in varying velocities on the opposite side of the desired split plane to pull the cloth apart (Figure 2). Likewise, draping and precise adjustment of cloth on scene models often relies on quick shifting (rapid pinning and releasing on the micro-level) of multiple fingers on the cloth surface to achieve the desired cloth configuration with respect to the model surface. Providing such global control of cloth is cumbersome and often non-intuitive with standard 2D pointing devices.

Figure 1: The user first creates a cloth by drawing its shape on the scene. The user then manipulates the cloth using multiple sticky fingers.

In our paper, we address this limitation by providing an intuitive user interface for fast cloth manipulation using a multi-touch input device. The system allows the user to directly control dynamic cloth interactions by means of two main techniques: sticky fingers and pinch-lifting. For the underlying cloth simulation, we adopt a position-based approach with the use of the Verlet integration scheme coupled with iterative constraint satisfaction. The combination of sticky fingers and a position / constraint oriented integration scheme is crucial because it not only yields real-time and stable results for our system, but also empowers us with direct manipulability over particle positions, which is key to making sticky fingers work.

Our first technique, sticky fingers, allows the user to simultaneously control different regions of the cloth with any number of fingers simply by putting their fingers on the touch-screen and moving

Figure 2: In bi-manual tearing, the three sticky fingers on the left serve to immobilize the parts of the cloth while the two fingers of the working hand move in varying velocities away from the split plane to tear the cloth apart.

∗e-mail: yttoh@cs.cmu.edu
†e-mail: nsp@cs.cmu.edu
them around. The nearest underlying cloth particles in contact with a finger are positionally attached to that finger and move dynamically with it. When a particular finger is lifted from the multi-touch device, the cloth region that was previously pinned by that finger is released (Figure 1).

Our second technique, pinch-lifting, on the other hand, allows the user to intuitively lift a cloth region in a direction perpendicular to the initial plane of the cloth simply by pinching that region. The system automatically detects a pinch event whenever the user moves two or more fingers in close proximity to one other. It then maps subsequent finger translations along the y-direction of the multi-touch device to upward movement in the corresponding scene, resulting in vertical movement of the pinched set of fingers and all the cloth particles stuck to them at that instance (Figure 3).

We describe core features of our user interface in Section 3, our cloth simulation model in Section 4, and implementation details of our interaction techniques in Section 5.

Figure 3: Pinch-lifting: The system interprets a pinch event as a cue to lift the cloth.

2 RELATED WORK
Cloth simulation has been a subject of unabated interest within the Computer Graphics community in recent decades. In particular, a great deal of effort has been focused on physically based approaches to cloth modeling. As a small sample of influential research in this area, Terzopoulos and his colleagues [16] introduced cloth simulation techniques to the graphics community. Breen et al. [3] performed early research on particle models with focus on physical parameters. Baraff and Witkin [2] used an implicit integration method to stably simulate realistic-looking cloth with large time steps. Choi and Ko [5] used a semi-implicit integration method to produce stable yet responsive cloth by handling post-buckling problems in cloth. Bridson and his colleagues [4] provided an efficient approach that robustly handled friction and contact responses. Their model also took into account cloth thickness.


Many commercial 3D applications such as garment-design programs and animation software require real-time manipulation of virtual cloth. Wojtan et al. [17] proposed a system that allows keyframe control of cloth. However his system is not real-time. Igarashi and Hughes [6] created an interactive system that allows the user to wrap cloth around a virtual character and adjust cloth vertices by dragging cloth vertices along the surface of the underlying geometry. Meng, Mok and Jin [10] produced a virtual try-on system that allows pre-positioning and draping of clothes on virtual bodies by means four basic manipulations: fix, drag, move and pin. While the last two systems mentioned are real-time, each operation can only be carried out one at a time, via a solitary mouse cursor. Haptics have also been explored for manipulation of three-dimensional geometries. Notably, Dachille et al. [7] proposed a haptic system that allows manipulation of physics-based B-spline surfaces with haptic feedback.

In contrast with previous systems, this paper develops techniques for real-time multi-touch control of cloth, allowing multi-point motions such as tearing, pinch lifting, and controlled bunching, rotation, and smoothing.

3 USER INTERFACE FEATURES
3.1 Cloth Creation
The system directly allows the user to initialize cloths of different shapes and sizes onto the scene in cloth creation mode. In this mode, the user simply draws cloth pieces on the multi-touch device, one after another using simple swipe gestures. Any valid cloth shape drawn is automatically converted into an appropriate cloth mesh at the scene location where the cloth is drawn. Gestures that do not resemble any supported shape such as incomplete enclosed shapes are otherwise rejected. In the event that part of a new piece of cloth is drawn over an existing cloth, the system automatically recognizes that part of the cloth to be physically on top of the corresponding area of the underlying cloth (Figure 4).

Figure 4: Cloth Creation: Newer cloths are automatically created on top of older ones in the event of overlaps.

3.2 Cloth Manipulation
3.2.1 Sticky Fingers
Once a piece of cloth has been created, the user can manipulate the cloth using sticky fingers (Figure 5). The user does so first by placing any number of fingers on the multi-touch screen. Each active finger that is in contact with a part of a virtual cloth becomes sticky. All underlying cloth vertices/particles within the finger contact area are automatically stuck to that finger and are dragged along with the finger whenever the finger moves. When the finger is removed from the touch surface, the same set of particles are freed from the particular finger. By default, the fingers are positioned on a default floor plane \(y = 0\) in the scene and slide over the floor plane when moved. Since cloth pieces are also drawn on the floor by default, each finger that is in contact with a cloth effectively pins the cloth onto the underlying floor surface and drags that part of the cloth along the floor surface when moved.

Our dragging mechanism markedly differs from the typical vertex dragging operation in many real-time cloth simulation systems in that various sections of the cloth can be dragged and released simultaneously in varying velocities. The typical single vertex dragging operation allows the user to select a vertex or a set of vertices using a single 2D input cursor for uni-directional displacement.
Drag a single vertex tends to induce large initial deformations near the vertex and it may take a substantial amount of time for the change to propagate to the remaining vertices. Therefore, while this technique is often useful for minute local adjustments of the cloth, it is ill-suited for global manipulations such as twisting a piece of cloth around a model surface during draping.

Moreover, many manipulations such as cloth adjustment on models also require rapid shifting of fingers to pin down varying areas of the cloth while applying dragging changes to other portions of the cloth (Figure 6). Such precise adjustments are difficult to control with a solitary 2D input cursor. Technically, this process can be replicated by rapidly inserting and removing pins to constrain and free specified locations of the cloth while making adjustments to the rest of the cloth. However, such a technique is obviously non-intuitive and extremely tedious to use. Some systems actually allow simultaneous translation of multiple vertices, but this process collectively propagates the vertices in a single direction, which is clearly inadequate when we require different portions of cloth to move in varying velocities at different instances. In fact, many basic cloth manipulations such as pinching and flattening require simultaneous and multi-directional control.

Our multi-touch sticky fingers technique allows the user to independently control different parts of the cloth at different velocities. This technique also allows the user to quickly shift his/her fingers rapidly to precisely determine which parts of the cloth to pin, and which parts of the cloth should be immobilized or simply left alone to be influenced by natural forces such as gravity and drag. Unlike most conventional interactive systems, our system does not need to explicitly specify discrete basic cloth operations, such as dragging, rotation, pinning and global translation. The process of using sticky fingers implicitly supports all these features in a manner that feels intuitive and natural to the user. In addition, it automatically supports bi-manual cloth operations such as dual-pinch folding and tearing (Figure 2).

3.2.2 Sticky Finger Size

The size of each sticky finger directly determines the number of underlying particles that can be stuck to the finger. Currently, the system does not take into account the exact shape and area of finger contact on the multi-touch screen and uses a simple 2D circle to represent its area of influence on the cloth. The user may however find it useful sometimes to resize the area of the finger via an option in the user interface even though we find that the default circular area is often adequate for many manipulations.

In fact, resizing the circular area has an effect of determining how much cloth area each finger is pinning down and any one time. A small finger radius pins down fewer particles and vice versa. Planar tearing manipulation for instance, requires some number of immobilizing fingers to be holding down portions of the cloth while moving the other fingers away from the desired split plane to achieve a tearing effect. Logically, the larger the area of individual immobilizing fingers, the larger the collective area of the cloth that is immobilized. This tends to make it easier to tear a particular cloth because the user is pinning down many particles and constraints while letting the desired constraints nearer the opposite side of the split plane to elongate as far as they can until they snap instead of letting many constraints elongate bit by bit throughout the cloth (Figure 7).

3.2.3 Cloth Selection

In the event that a sticky finger comes into contact with two overlapping pieces of cloth, the finger automatically pins downs and sticks both underlying areas of the cloths to itself. This behavior is very useful if the user needs to manipulate both layers of the cloth in unison, such as tearing stacked cloths together, or simply moving a piece of cloth that has been folded to become two layers. It could represent a common situation where the friction between cloth layers is high relative to that with the underlying surface. However, the user can choose to control only the first layer of cloth that the finger encounters by enabling a special mode known as the stick to top only mode (bottom left of Figure 8). This mode is particularly useful for picking out and separating overlapping cloths.

3.2.4 Lifting With Sticky Fingers

Often, the user may find the need to lift sections of the cloth vertically, especially in important manipulations such as folding and draping. The user can achieve that in two ways: directly by sticky fingers or by pinch-lifting. To lift a piece of cloth by sticky fingers, the user invokes a hot-key that remaps upward motion on the
physical touch pad to movement of each sticky finger up along the y-direction in the scene space. Individual fingers can then lift the cloth without having to explicitly pinch the cloth by having the particles stuck underneath them move up with them. While this is a less physically realistic maneuver, it is particular useful in situations where the user finds himself/herself running out of fingers. One example of such a situation is when the user wants to pinch all four corners of a cloth and lift the cloth. This is not possible with just two hands, so another way of achieving this operation is to use four sticky fingers, one at each corner of the cloth, to execute a upward lifting operation (Figure 9). The second method of lifting is realized by pinch-lifting, which is described in the next section (Section 3.3).

3.3 Pinch-lifting

While the first method of lifting a piece of cloth via sticky fingers as described in Section 3.2.4 can be convenient, it tends to be less intuitive and also less physically natural. The system hence provides the user a more intuitive way of cloth lifting by interpreting a pinch gesture made at a portion of the cloth as an indication that the user is about to lift that bit of the cloth. When the user moves two or more fingers close enough towards each other at a point on the cloth, the system automatically switches the mode of control to vertical movement in the scene (in similar fashion to the process of invoking the hot-key for sticky fingers lifting). Subsequently the user can slide the pinched set of fingers upwards on the touch-pad, and this movement is now interpreted as moving the cloth upwards in the scene instead of inwards in the direction parallel to the floor plane. Indeed, most of our testers find this particular mode of lifting more natural than lifting with sticky fingers, especially if the task is as simple as pinching two ends of the cloth and waving it about. However for more complex tasks such as precise folding and draping, most testers find it easier and more efficient to simply use sticky fingers for lifting.

3.4 Pinning

Selected cloth sections can be explicitly pinned by using the pinning sub-tool offered in cloth manipulation mode. The user does so by tapping on the appropriate area to push a pin through the cloth at the required location. To remove the pin, the user invokes a “delete pin” hotkey in pinning mode and taps on the appropriate pin to remove it.

3.5 Adjusting the Capture Region

The user can dynamically control the size and location of the capture region, a 2D rectangular zone in scene space that represents the area of the physical multi-touch surface mapped directly to scene space (Figure 10). The location of this capture region determines the exact scene area where any active finger can exert its influence. This applies to both cloth creation and cloth manipulation.

Figure 8: Cloth Selection: Top: A finger contacts a region with two layers of cloth. Bottom Left: When the finger moves, the default response is to move both layers together in unison. Bottom Right: However, it is sometimes useful to move only the top layer that is directly in contact with the finger.

Figure 9: Lifting With Sticky Fingers: Each desired region of the cloth can be lifted with one finger. Compare this to pinch-lifting in Figure 3. The latter would require at least two fingers at each lifting point.

Figure 10: The capture region: a user-adjustable rectangular zone of control in the scene that corresponds to the area of the physical touch-pad.

The user manipulates a set of directional hot keys to dynamically resize or re-position the capture region. The default 2D capture region in the scene lies on the x-y plane and is centered about the origin (Figure 10). The ratio of the width to the height of the capture region is clamped by default to the aspect ratio of the physical multi-touch surface area. This is done to ensure an intuitive and more natural mapping of the physical finger movement on the multi-touch surface to that within the 3D scene. The user can over-ride this limitation by manually re-specifying the scaling factor. Re-sizing the capture region can be especially useful if the user requires very fine-tuned control over a small portion of the cloth, such as a precise pinch. In this case, the capture zone can be quickly shrunk and re-positioned at the desired area of influence on the cloth. The capture region is now represented by a larger physical multi-touch surface, allowing any finger movement on the surface to correspondingly map to a relatively finer adjustment on the virtual cloth in the scene.

4 Cloth Simulation Model

It is imperative that we use a fast and stable integration scheme since our system seeks to be interactive in nature. We first model cloth as a simple grid of particles connected to one another by stick-like constraints (Figure 11). For the particle simulation, we choose to use the velocity-less Verlet integration scheme with iterative constraint satisfaction [8] due to the speed and relative stability that this approach provides. In particular, this method works directly
on positions, which is key to making our technique of sticky fingers work. In this section, we give a brief technical overview of the Verlet scheme used in the context of our system. For more in-depth background about basic Verlet integration with corrective constraint satisfaction, we would like to refer the reader to Jakobsen’s work [8].

4.1 Cloth Model

We represent cloth as a simple regular grid of particles interconnected by various stick-like constraints (Figure 11). Particles immediately adjacent to each other are connected by a stretch constraint. Particles that share the same shortest diagonal are connected by a shear constraint. Finally, particles that are two neighbors away from each other in the cloth grid are connected by a bend constraint. While traditional mass-spring models often use spring-damper forces to represent constraints between particles, such an approach tends to produce “super-elastic” cloth [14], which often looks un-realistic. Our position-based approach does not use spring forces to maintain distance constraints. Instead, we avoid the problem of over-elongation by attempting to correct the positions of particles that have moved out too far beyond their allowed constraints. We achieve this by a process known as iterative constraint satisfaction (Section 4.3). Our approach allows us to generate rather stiff “stick-like” constraints between cloth particles, which results in more realistic-looking and less stretchy cloth.

4.2 Particle Integration

In most particle integration schemes, a basic particle typically maintains three variables: position \( x \), velocity \( v \) and acceleration \( a \). In Verlet Integration however, the current velocity is approximated by the difference between the current position of the particle, \( x_{\text{cur}} \), and its position in the previous time-step, \( x_{\text{old}} \). Therefore, velocities are now implicitly updated via direct manipulation of positions and need not be explicitly stored. Instead it becomes necessary to always maintain and update \( x_{\text{old}} \) at each time-step. The main Verlet update rule is shown in equation (1). In equation (2), the old position, \( x_{\text{old}} \), is updated with the value of the current particle position \( x_{\text{cur}} \), so that it can be used in the next time step.

\[
\begin{align*}
\text{x}_{\text{new}} &= 2\text{x}_{\text{cur}} - \text{x}_{\text{old}} + a\Delta t^2 \\
\text{x}_{\text{old}} &= \text{x}_{\text{cur}}
\end{align*}
\]

4.3 Iterative Constraint Satisfaction

The final requirement for our interactive system is to be relatively stable. This is achieved via iterative constraint satisfaction. After the Verlet integration step, particles attain new positions. Multiple pairs of particles may have been over-stretched beyond their desired rest distance. To keep cloth structure intact while allowing the cloth to exhibit desirable properties such as shearing and bending, we attempt to satisfy all the constraints mentioned in 4.1 by relaxation. We iterate through all the distance constraints a number of times and attempt to restore pairs of particles back to their rest states by applying suitable position corrections to each particle. For any constrained pair of particles with positions \( x_1 \) and \( x_2 \), and initial rest distance \( r \) between the particles, the amount of correction for each particle is computed using formulas (3) and (4) as proposed by Jakobsen. In (3), we compute the total displacement that particle \( x_1 \) alone has to make to be one rest distance apart from particle \( x_2 \).

\[
\Delta x_1 = \frac{1}{2} \left( |x_2 - x_1| - r \right) \frac{x_2 - x_1}{|x_2 - x_1|} \tag{3}
\]

\[
\Delta x_2 = -\Delta x_1 \tag{4}
\]

4.4 Integration with Sticky Fingers

Integrating sticky fingers with our simulation model is generally seamless but does require a little book-keeping. Specifically, during iterative constraint satisfaction, we need to ensure that any constraint that involves a cloth particle that is already stuck to a sticky finger does not try to move the particle. On the other hand, the particle on the other “end” of the constraint should attempt to obey that constraint by moving towards the stuck particle, only if it is not stuck to a finger as well. To handle this situation, each particle is flagged whenever it is stuck to a sticky finger. During iterative constraint satisfaction, for each constraint that contains a flagged particle \( x_1 \) and unflagged particle \( x_2 \), we compute the total correction vector as in 4.3 but distribute the full correction vector entirely to \( x_2 \) to ensure that the stuck particle is fixed to the finger while only the unflagged particle attempts to move towards the stuck finger. Essentially, we are setting the mass of stuck particle to infinity, since its motion is clamped to that of the finger and should never be allowed to obey any other constraint until the finger is lifted. Formally, the corrections are expressed as follows:

\[
\begin{align*}
\Delta x_1 &= 0 \\
\Delta x_2 &= -\left( |x_2 - x_1| - r \right) \frac{x_2 - x_1}{|x_2 - x_1|} \tag{5}
\end{align*}
\]

In the event that both particles are being stuck to fingers, both correction values should be set to zero. In the case where both particles are not stuck, equations (3) and (4) from the previous section should be applied.

Explicit pinning of cloth regions by the user is handled exactly in the same manner; These particles are flagged as being “stuck” whenever they are pinned, and unflagged when the pin is removed. It is worth mentioning that our equations are identical in spirit to the constraint projection equations that involve inverse masses [12]. To stick particles to moving objects, Muller zero-ed the inverse mass of the particle and weighed the respective correction vectors accordingly [12]. We found it slightly easier though to flag “stick particles” and treat the three possible cases individually instead of explicitly using and modifying inverse masses.
We use the TUIO protocol [9] for multi-touch integration with our system. This protocol allows the transmission of touch events in the form of UDP packets from a suitable finger tracker application to our system. On start-up, our system listens on a UDP port for these touch messages. A basic touch message typically consists of the position, velocity and area of each individual finger that is in contact with the multi-touch surface. Our system constantly keeps track of position information of each physical finger which is given in normalized touch-screen coordinates, and calculates its corresponding virtual finger position on the capture region in the scene (Figure 10). When a physical finger moves, this calculation is updated accordingly based on the new position. Finally, when a physical finger is lifted from the touch surface, we simply delete the virtual finger from our system. Depending on the multi-touch hardware used, the area and shape of each finger that is in contact with the multi-touch surface may be available to the system. This information could produce more accurate physical results because the underlying cloth area of influence under the virtual finger can then be shaped according to the physical size of the actual area of contact instead of being fixed as a round finger diameter. However, our current hardware implementation does not return area information, so our system works with adjustable finger diameters instead. In fact, this setting generally works very well for all our manipulations.

Currently, we use the Apple Ipad as our multi-touch device because of its good screen resolution, reliability and portability. We also rely on a TUIO-compatible finger-tracker application for Ipad, MSA Remote [1], to track finger positions on the Ipad touch-screen and send these touch messages wirelessly to our system. Prior to choosing the Apple Ipad, we built a simple DIY multi-touch device based on diffused illumination [15] and tracked finger shadows using a camera. Using an open-source image processing library, TouchLib [13], we were able to feed the relevant finger positions into our system and use them to manipulate cloth. However, the ability to achieve satisfactory results often depend on the surrounding lighting conditions. The touch data received were sometimes noisy, which often resulted in “ghost fingers”. We eliminated these noise issues and the hassle of constant re-calibration by switching to the capacitive touch-screen of the Ipad, which allows us to achieve consistent and robust results.

5.2 Simple Shape Gesture Recognition

In creation mode, the user creates a cloth by drawing the cloth shape on the multi-touch screen with a single and continuous finger stroke. Our system currently performs simple axis-aligned rectangle gesture recognition (Figure 13). Whenever a finger delete event occurs, we check whether the removed finger was the only active finger in the previous session. A session is defined as a time period during which the number of active fingers on the multi-touch surface does not change. If the removed finger was indeed the only active finger, the user must have executed a single and continuous swipe gesture.

Assuming that the user has attempted to draw something resembling a square or a rectangle, we test the drawn finger path for proper closure. If the beginning and the end of the stroke are too far away from each other, we reject the gesture as an incomplete shape or an accidental stroke. Otherwise, we proceed to compute an axis-aligned bounding box from the minimum and maximum x and y values seen from the points in the finger trajectory. We then calculate the ratio of the dimensions of the resulting bounding box to the dimensions of the touch-screen, derive the scene dimensions of the cloth and initialize the cloth within the capture region on the scene. We find that this simple axis aligned bounding box approximation works surprisingly well.

![Figure 12: Simple mesh tearing is achieved by removing constraints that are “over-stretched” beyond an allowed distance threshold](image12)

4.5 Tearing

We allow tearing of a cloth mesh by proceeding with a simplified approach of removing constraints containing particles "stretched" apart by a certain allowable distance threshold (Figure 12). While many other robust tearing methods such as the particle-splitting method [12] exist, our simplified model is easy to implement and yields satisfactory visual results for higher resolution meshes. The feel of the cloth can also be varied by the user simply by specifying the maximum tearing threshold for each kind of constraint. If the threshold values are generally low with respect to the initial rest distances between cloth particles, the cloth exhibits highly brittle behavior, even upon mere contact. Conversely, if the thresholds are high, the resulting cloth is very hard to tear and may have to be stretched by a large distance before the first instance of tearing occurs.

5 Implementation Details

The system is implemented in C++ and uses OpenGL for rendering. The system is tested on a Intel Core 2 Duo laptop at 2.0 Ghz. The exact number of particles depends on the dimensions of the cloth as drawn by the user but a cloth mesh with a typically high resolution can consist of about 1000 particles on average. Constraints are satisfied 5 times per frame. Even with 1000-particle cloth meshes, we are able to attain highly satisfactory frame rates for most of the manipulations carried out. The cloth structure is quite robust in terms of stability due to the process of iterative constraint satisfaction at each time step. If the user somehow manages to blow up the cloth or cause a large number of self-intersections, the cloth automatically recovers and unravels itself neatly when left alone for a short amount of time.

5.1 Multi-Touch Integration

The system is tested on a Intel Core 2 Duo laptop at 2.0 Ghz. The system is implemented in C++ and uses OpenGL for rendering. The system currently performs simple axis-aligned rectangle gesture recognition (Figure 13). Whenever a finger delete event occurs, we check whether the removed finger was the only active finger in the previous session. A session is defined as a time period during which the number of active fingers on the multi-touch surface does not change. If the removed finger was indeed the only active finger, the user must have executed a single and continuous swipe gesture.

Assuming that the user has attempted to draw something resembling a square or a rectangle, we test the drawn finger path for proper closure. If the beginning and the end of the stroke are too far away from each other, we reject the gesture as an incomplete shape or an accidental stroke. Otherwise, we proceed to compute an axis-aligned bounding box from the minimum and maximum x and y values seen from the points in the finger trajectory. We then calculate the ratio of the dimensions of the resulting bounding box to the dimensions of the touch-screen, derive the scene dimensions of the cloth and initialize the cloth within the capture region on the scene. We find that this simple axis aligned bounding box approximation works surprisingly well.

![Figure 13: Simple Rectangle Gesture Recognition: The gesture is first tested for reasonable closure. Then an axis-aligned bounding box is constructed around the gesture. Finally the cloth is created based on the bounding box.](image13)
5.3 Collision Response

To maintain our system at interactive rates, cloth self-collision is not implemented. For collision detection and response with simple scene geometries, we use a simple projection scheme that tests whether each cloth particle lies within the model. If a particle is found to be in that state, we simply push the particle onto the nearest point on the object surface. We also keep the cloth a small distance away from the surface to prevent depth-related rendering artifacts as well as the possibility of edges penetrating the underlying surface. Typically though, edge penetration seldom happens because our cloth model is of a sufficiently high resolution. As for friction, we compute the tangential component of the collision response from the vector $x_{new} - x_{old}$ and the surface normal. After projection to the surface, we then scale this tangential component by a suitable frictional coefficient and attenuate $x_{new}$ with the scaled tangential component to produce a suitable frictional response between the cloth and the surface.

Our dragging operation is unconstrained in that if the user moves his fingers too quickly into the geometry of a scene object, the cloth may also move into the object. However, our system automatically corrects this violation by projecting the cloth to the surface of the object in the following time step the moment the user releases his finger (Figure 14).

![Figure 14: Collision Handling by Projection. The system automatically corrects any geometric violation in next time step after the user releases his finger.](image)

6 Results and Evaluation

Figures 15 and 16 show the basic actions of bunching, spreading, translation, and rotation performed using our system.

We evaluated our system by conducting informal user tests. We designed each test as a set of simple manipulation tasks for the user to achieve. The test begins with a “basic functionality” task. For this task, users were requested to accomplish a series of simple planar operations, namely bunching, spreading, translation and rotation in this specific order. Generally, the average user took about 25 to 30 seconds on the first try to get a good feel of the system and complete this sequence. On subsequent attempts, most users were able to perform the required sequence under 10 seconds.

The next task involves planar tearing of a piece of cloth into two halves. We wanted to gauge how quickly and intuitively it is for the user to grasp the feel of the cloth material and also how well the user can control the tearing process. In the first few attempts, users generally have difficulties tearing a cloth into exactly two halves. They tend to produce many small scraps in the process. This is actually because the system can only utilize fingertip contact areas. In real tearing, many parts of the cloth are immobilized by large areas of the hand such as entire length of fingers, or the palm, while cloth particles near the desired split plane are unconstrained and allowed to be moved apart. Therefore, users often produce small scraps the size of their finger tips, because these are the exact areas where the cloths are not allowed to be split. When we enlarge the finger size, users tend to have higher success rates. Also, some users adapted to this limitation by interleaving rapid movement of fingers in the direction parallel to the desired split plane with perpendicular tearing away from the split plane to control the tearing process.

When we vary the cloth material to make the cloth harder to tear, users were able to “feel” the difference in material after an initial attempt to pull it apart. We observed an interesting common strategy that emerged: Users tend to drag cloth particles away from the desired split plane then quickly shift their fingers to get a re-grip on cloth particles nearer the split plane to repeat the process, until the cloth tears. This “clawing” strategy seems to be particularly effective because a single drag may not produce significant extensions near the split plane if the tearing thresholds are high. Multiple drags are often required to produce the overall elongation required to stretch particle distances, especially those near the split plane, beyond these thresholds.

The remaining tasks, namely draping and folding, involve explicit lifting of the cloth from the underlying plane. Users were encouraged to experiment with both pinch-lifting and lifting by sticky fingers to see which method suits them the best for each of the tasks. Most users have no problems making a simple fold or drape with no particular preference for either method. This seems to be so because the manipulation can typically be done by lifting at most two points of the cloth. When we enforced the requirement that the draping must be done precisely so that the cloth is positioned in a precise configuration with respect to the underlying geometry, many users tend to prefer lifting by sticky fingers because they could lift up the cloth completely with multiple fingers and position it precisely above the geometry before placing it carefully on top of the model. But a problem commonly faced with this mode of lifting is the need to trigger the toggle key for lifting. This broke the “momentum” of the entire manipulation process, especially for users who were using sticky fingers from both hands; they find themselves putting down the cloth for a moment to trigger the toggle, before continuing. On the other hand, pinch-lifting felt more natural for most users even though they tend to do a lot more adjustments to get the cloth in the desired configuration once they manage to get it onto the model.

In general, our testers liked the novelty of our system. They also commented that cloth control felt natural and physically realistic; they were able to relate the effects of our supported operations to real cloth manipulation. However, some testers pointed out that the system would be more appealing if cloths of various shapes can be created. Other testers also felt that pinch-lifting is visually more believable than lifting with sticky fingers, even though the latter may in some cases be easier to use. In addition, a tester remarked that our interaction techniques would be very suitable for games that involve manipulation of flexible virtual objects, possibly by multiple users, since our system does not place any restriction on the number of sticky fingers one can place on a cloth.

![Figure 15: Planar manipulations. Top row: Bunching, spreading and translation. Bottom row: Three frames of rotation](image)
7 Limitations and Future Work

We have presented a novel real-time system that allows the user to intuitively control and manipulate cloth with multiple fingers using a multi-touch device. Our techniques of sticky fingers and pinch-lifting, coupled with the use of Verlet Integration with iterative constraint satisfaction, enable the user to generate a range of interesting cloth interactions that are often not attainable with the conventional mouse device. We believe that our system will be very useful for interactive 3D scene design involving multiple cloths, virtual garment design as well as cloth folding applications. In addition, our interaction techniques can potentially be used to generate realistic animations of dynamic cloth manipulations, which can often be challenging to attain manually.

Our approach has several limitations. Firstly, since the nature of the multi-touch screen is two-dimensional, cloth can only be controlled along a single plane at any instance. Fingers cannot be lifted off the surface. While planar manipulations by sticky fingers can allow the user to generate rather interesting cloth behavior that would previously be challenging to generate, such as pinching, bunching, flattening, global cloth rotation and translation, many complex cloth manipulations like folding and draping count on the ability to lift the cloth off the underlying plane. We attempt to provide "2.5-dimensional control" by introducing two modes of lifting: namely sticky-finger lifting and pinch-lifting. For sticky-finger lifting, we explicitly provide a toggle key to switch the control plane from the default x-z plane to the y-z plane or x-y plane while for pinch-lifting, we switch these axes of control automatically. While we find these modes sufficient for a wide range of manipulations, a user may want to control cloth along a user-defined arbitrary plane other than the three preset ones. Perhaps, allowing manipulation of the control plane by interactively moving the camera might be more intuitive to the user.

Cloth-cloth collisions, both with itself, and with other cloths, are not implemented for performance reasons. While we think this decision is crucial to keep the system at real-time, perhaps an optional collision detection scheme such as basic particle-particle repulsion might appeal to certain users seeking for physical correctness over manipulability and real-time control. In fact, the proposed repulsion scheme may suffice for some interesting effects such as the ability to drape clothes over a laundry line modeled as a thin rope.

We are also considering the support of manipulations such as cutting and stitching, which we believe will be easy to incorporate into our current framework. With these operations, the user will be able to cut or sew cloth by simple finger swipe gestures along cloth surfaces. When combined with our supported operations such as folding and tearing, the user can achieve many interesting outcomes with the cloth.

Currently, our system permits only rectangular cloths. In future, we would also like to provide support for both gesture recognition and modeling of arbitrarily-shaped cloth. Instead of representing cloth as a regular grid of vertices, polygonal manifold meshes can be used to represent various cloth geometries more accurately.

In addition, our tearing model only supports cloth wire meshes and does not conserve total cloth area due to the direct removal of constraints, which visually represents the "wire" between each particle during wireframe rendering. A more accurate model of tearing would allow us to render shaded cloth properly as well as conserve total area [12]. Also, if we can stick particles based on exact touch contact areas instead of assuming circular finger area contacts, the user will be able to tear cloth more realistically because the exact cloth area that is immobilized during tearing can be modeled.

We are also exploring the concept of "non-sticky fingers". Instead of sticking to sections of the cloth and moving them, such fingers slide along the surface of the cloth without adhering to any part of the cloth. In particular, "non-sticky fingers" could be useful for smoothing out wrinkles on a cloth.

Acknowledgements

We wish to thank: the Carnegie Mellon Graphics Lab for their comments and suggestions during the Graphics Retreat, the Hands Group for the constructive discussions, as well as all those who helped test the system.

References
