Chapter 2
Multi-finger Grasps in a Dynamic Environment

William Harwin and Alastair Barrow

Abstract Most current state-of-the-art haptic devices render only a single force, however almost all human grasps are characterised by multiple forces and torques applied by the fingers and palms of the hand to the object. In this chapter we will begin by considering the different types of grasp and then consider the physics of rigid objects that will be needed for correct haptic rendering. We then describe an algorithm to represent the forces associated with grasp in a natural manner. The power of the algorithm is that it considers only the capabilities of the haptic device and requires no model of the hand, thus applies to most practical grasp types. The technique is sufficiently general that it would also apply to multi-hand interactions, and hence to collaborative interactions where several people interact with the same rigid object. Key concepts in friction and rigid body dynamics are discussed and applied to the problem of rendering multiple forces to allow the person to choose their grasp on a virtual object and perceive the resulting movement via the forces in a natural way. The algorithm also generalises well to support computation of multi-body physics

2.1 Introduction

Our ability to manipulate our environment underlies our intelligence, and the role of hands in this manipulation is self evident. The machinery behind the hand, both mechanical (muscular skeletal) and computational (neuronal) has been an inspiration for research and design. Hooks and artificial hands are the currency of upper-limb prosthetics and an early example was the iron hand made for the soldier Götz von Berlichingen in the 15th Century. In the early part of the 20th Century the value of remote handling was realised with the first master-slave telemanipulators [41].

W. Harwin (✉)
School of Systems Engineering, University of Reading, Reading RG6 6AY, UK
e-mail: w.s.harwin@reading.ac.uk

A. Barrow
Imperial College, London, UK
e-mail: a.barrow@imperial.ac.uk

I. Galiana, M. Ferre (eds.), *Multi-finger Haptic Interaction*,
Springer Series on Touch and Haptic Systems, DOI 10.1007/978-1-4471-5204-0_2,

Fig. 2.1 Remote handling master hand interface mechanism. Courtesy of Oxford Technologies Ltd. (www.oxfordtechnologies.co.uk)

The early master-slave telemanipulators were directly coupled so the forces encountered in the environment were reflected into the hands. Modern telemanipulators are now coupled though an information channel but the value of the force reflection is still realised in modern implementations to give the operator greater control (Fig. 2.1). These ideas were combined with the concepts of simulated environments and virtual realities and led to the idea of telepresence. The interest in haptics then emerged as a mechanism to facilitate our interaction with computers and visual information [6, 7], but with a much devalued concept of manipulation. Multi-finger concepts provide a means to realise manipulation in virtual and simulated environments.

Most modern haptic devices associate interactions between the person's hand and a virtual environment which can be optionally colocated (an example of co-location is shown in Fig. 2.14). The high cost of haptic device technology has resulted in most of these interactions occurring via a single point of contact which could be a thimble as in the Phantom,[1] or via a tool surrogate. Surrogates include primitive shapes such as the spheres used by the Falcon,[2] HapticMaster,[3] and Omega.3;[4] or tools such as pens/styli, dental and laproscopic instruments, catheters and needles. Although interaction with surrogate tools may simplify hand-world interactions, the problem is that the tool may need to be changed for each context. So, for example, a training workstation used to teach dentistry skills may use a surrogate of the dental hand piece to remove filling or decayed tooth (caries), but this tool surrogate will need to be exchanged for a thinner and lighter stylus to represent a probe for diagnosing decay, a periodontal probe to measure pocket depth or a periodontal scaler for removing calculus [12, 44].

Allowing force or force+torque (wrench) interactions to happen at the level of the individual contacts between the person and the virtual object allows more flexibility in the individual applications, in particular it eliminates the need for applica-

[1] Sensable.com, USA.

[2] Novint, USA.

[3] Moog FCS, The Netherlands.

[4] ForceDimension, Switzerland.

tion dependent surrogate objects. There are two reasons that this is a less favoured approach, firstly cost since it increases the number of degrees of freedom needed, and second the complexity, since problems emerge in terms of reducing the effective workspace over a simpler device and the unintended self collisions between the linkages of the haptic device, or between the device and the person's fingers, palm etc.

However there is a growing interest multi-finger and multi-contact haptics. Work has been done on kinematic design of multi-finger interactions [4, 48], and on providing multi-contact haptics in large workspaces [5, 35, 50]. Work reported elsewhere in this book includes the Spidar8 and Spidar-hand by Kumazawa and Sato [21]. The latter project succeeded in providing 8 contact points to 4 fingers on each hand using a total of 24 cables (3 per finger). The HIRO series of robot hands [15] is also reported elsewhere in this book and provides an innovative method to deliver forces to the fingers and thumbs of each hand. Reach and grasp has been studied also in terms of stroke rehabilitation and work by Loureiro et al. [27, 28] has shown that there are benefits in retraining a reach-grasp-transport-release cycle for people with upper limb hemiplegia. The Gentle/G system used for the study relied on the HapticMaster admittance controlled haptic device for the gross movements, and a custom built three-axis admittance controlled hand exoskeleton for the grasp-release cycles.

There has been less work on assessment of multi-finger contacts, but McKnight [29] has shown that the addition of multi-finger haptic feedback leads to better positioning accuracy in 5 degrees of freedom.

2.2 Grasp Analysis

Although the algorithm now reported does not require knowledge of hand anatomy, physiology or motor control, these concepts form an important backbone for the design and evaluation of any haptic device so are discussed here in outline.

There are many taxonomies for hand grasps and work has been done in areas such as prosthetics [26], psychology [16], occupational and physical therapy [45], ergonomics, robotics [14], haptics [8], accessibility education and employment.

One classification is shown in Fig. 2.2 (US access board) which shows a broad classification mechanism into pinch, lateral pinch, power, etc. However if you consider the palm as a reference frame it is possible to identify 16–20 DoF as described in Table 2.1. Degrees of freedom are often linked, or the range of movement is only a few degrees hence the uncertainty in defining the independent degrees of freedom. However as shown in Fig. 2.3, even the simple act of holding a pencil or stylus can elicit a range of acceptable grasps. More recently in-hand actions have become prominent due to the rise of mobile phone texting, and a typical grasp would lay the keyboard of the phone across the fingers to allow texting with the thumb as shown in Fig. 2.4.

We consider grasp taxonomy in terms of the physics needed for a stable grasp, that is a set of contact points or areas between the person's hand and the object

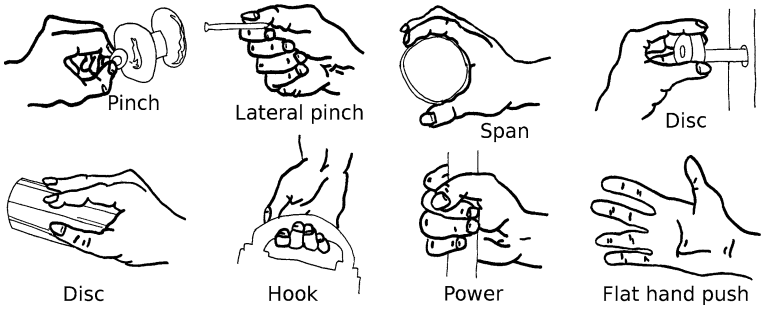

Fig. 2.2 Some typical Grip configurations (Reprinted from monograph by Edward Steinfeld, Hands-on Architecture [42])

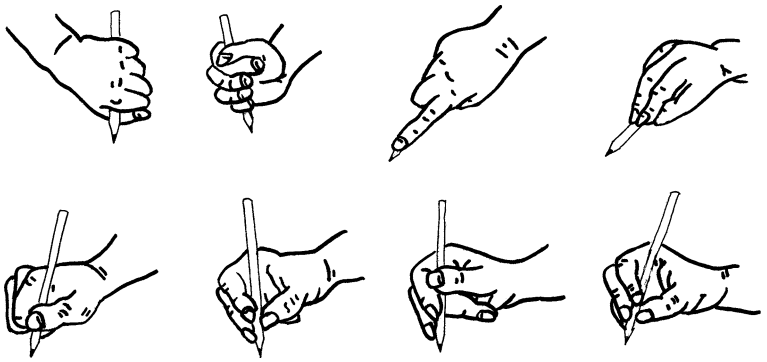

Fig. 2.3 The classic stylus or pencil grasp is called a 'tripod grasp'. In practice there is a large variation of stylus grasps. Adapted from [40]

Table 2.1 Principal degrees of freedom within the hand

Joint	DoF
Metacarpal proximal phalengeal (fingers)	8
Proximal to mid phalengeal (finger)	4
Mid to distal phalengeal (finger)	4
Carpal to metacarpal (thumb)	2
Metacarpal to proximal phalengeal (thumb)	1
Proximal to distal (thumb)	1

that allow the object to be accelerated without danger of losing control over the trajectory of the object. An approximate hierarchy of complexity is

- Single point of contact: Static stability requires object mass to lie below point of contact. Example would be a finger hooked to hold a cup or mug by its handle.

Fig. 2.4 Typical grasp while texting with the thumb

- Single area of contact: Stability requires the object centre of mass to rise if the object is perturbed. Example would be holding a plate on the palm.
- Multiple points of contact on the finger(s) and thumb. Stability is discussed below. Examples are pinch grips, and stylus grips.
- Multiple points or areas of contact that could include finger(s), thumb and palm. All grasps could be considered in terms of areas of contact that might change over the duration of the grasp. The definition would now include manipulating a mobile phone to do texting, as well as what are some times known as power grasps, where the hand encloses the object, for example the handle of a hammer, or a can of soft-drink.

A further requirement is that the contacting fingers have the span to reach to the considered contact points as well as the capacity to exert the necessary force. This is sometimes defined as grasp isotropy [14].

Assuming that grasp isotropy can be achieved for a particular individual the requirement for a stable grasp is then force closure. Force closure is the condition where the net forces on the object from the fingers, gravity, any other objects or constraints, and due to any accelerations must be zero. The robustness of the grasp can then be considered as the perturbation of all the forces applied to the object that still maintain the object within the person's grasp. This condition will be used as the basis for the multi-finger haptic rendering described below.

A second, and related condition required for a stable grasp is torque closure. Point forces applied via the fingers to the object result in a torque around the object's centre of gravity. Where a contact area is involved in the grasp then an additional torque is applied as being the sum of the integral of the pressure taken as a moment around the centre of force. The discussion below will only consider the torques due to point forces, and will assume that the torque due to the pressure distribution over the contact area can be ignored, however extending this algorithm to include the latter should not be difficult.

If a contact area is being considered then the point equivalent force $\underline{F}$ can be calculated as

$$\underline{F} = \int_A P(\underline{r})\, d\underline{a}$$

where $\underline{P}$ is the pressure field at point $\underline{r}$ integrated over contact area A. The location of this force is the centroid of the pressure distribution. This force must be combined with a torque calculated as

$$\underline{T} = \int_A P(\underline{r})\, \underline{r} \times d\underline{a}$$

The concepts of force closure and torque closure can be used to define a stable grasp. The concept is modified slightly in haptic rendering so the residual forces and torques applied to the (virtual) object are used to compute an acceleration and hence to update the location of the object. Constraints such as a working surface like a table top are managed either as a force applied through a stiff element or as an impulse condition.

For most grasps friction forms an important and stabilising component of the force closure. Without friction the grasp would have to be naturally stable. This can be done by hanging a cup or mug by the handle with a hooked finger so that the centre of gravity can rest below the point of contact. In the absence of a convenient handle then at least 4 contact points are required to ensure that the object is restrained in three dimensional space. (Four points are a necessary condition given that the fingers can only apply a compressive force on the object). By including a consideration of the frictional forces of a grip, it is possible to manipulate the object with 5 degrees of freedom using only two points of contact. The degree of freedom that is not controlled is the rotation around an axis joining these two points of contact. A three finger grasp then becomes sufficient to constrain motion of the object within all 6 degrees of freedom.

2.2.1 Internal Models of Movement—Perceptions and Grasp

An important aspect of haptic rendering is the person's perception and how that can be used to promote realism. The concept of visual dominance is well documented [36, 37]. Giving visual and auditory cues can do much to add to the illusion of permanence in a haptic situation, for example, the authors noted that in the dental training workstation developed for the Haptel project (www.haptel.kcl.ac.uk) the variation of the pitch of the hand-piece as pressure was increased on the burr was a strong surrogate for the vibrations (that were missing) that would have been present in an operational drill.

These phenomena are explained by the fact that humans do a large degree of a-priory modelling about the nature of the object they are about to grasp, and in addition anticipate the loads during the manipulation [25]. Thus explaining perceptual phenomena such as

A-priory modelling

- Preshaping of grasp [20]
- Grasp illusions such as size-weight, texture-weight, grasp aperture-weight illusions (judgements of an objects weight are made well before movements start)

During manipulation

- Anticipation of object dynamics (grip force adjusts on lifting vs lowering an object) [17, 49]
- Correlation between grip force and load force. The grip force is modulated to be 'just enough' [23]

2.3 Friction Models for One Point Contacts

Friction is a complex and nonlinear phenomena with a corresponding plethora of approaches to modelling its effect. Most undergraduate engineering textbooks will only discuss linear (viscous) friction in any depth where the force due to the relative movement of the frictional surfaces is assumed to be proportional to the relative velocity. That is $f = Bv$ where B is the constant of proportionality. Although this model allows techniques such as Laplace transforms to be applied to the analysis, it is not a sufficiently accurate model of friction to describe grasp and support the haptic rendering of multi-finger systems requiring object manipulations.

The classic Coulomb friction model (shown in part in Fig. 2.5) provides a better representation of friction for haptic rendering. The model is characterised by a single value and is most commonly expressed as

$$F = F_c \operatorname{sgn}(v)$$

where the function $\operatorname{sgn} v$ computes the sign of the relative velocity. The frictional force F is assumed to oppose relative movement and the defining constant, F_c may, in practice, differ for positive and negative directions of sliding.

In certain situations it becomes necessary to consider the discontinuity of Coulomb friction around the origin. Thus for example, the frictional force is indeterminate at $v = 0$ lying somewhere between $-F_c$ and F_c. The LuGre friction model [2] includes microslip states to model microslip phenomena when there is an applied force that is less than the limit of static friction. The LuGre model is a relatively sophisticated model and a more intuitive realisation is the bristle model [9] which considers micro movements during periods of zero relative velocity, to be modelled as the interaction of two bristle surfaces. The bristle model reduces to a simpler stress model known as the Dahl model [19].

A well known method for computations involving coulomb friction is to generate a friction cone (Fig. 2.6) based on the limits of static friction and to consider the resultant force on an object that is subject to friction. The cone of friction provides a easy method to determine if the object is sliding by observing the resultant force with respect to the cone. If the resultant is contained within the cone then the object

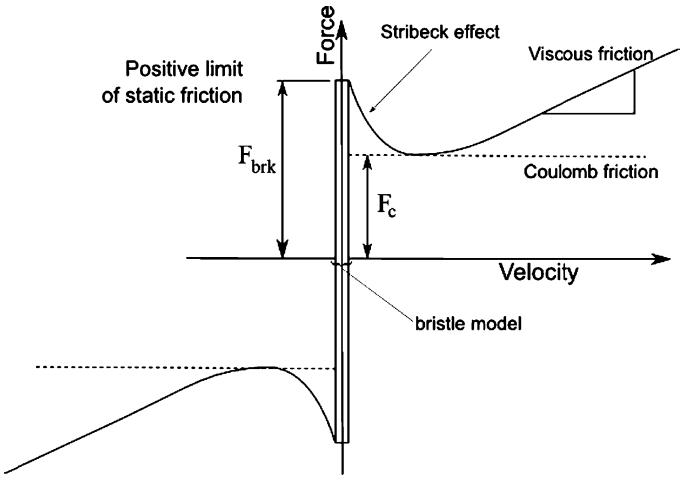

Fig. 2.5 Friction models shown as a function of velocity with a discontinuity at the origin to illustrate position phenomena

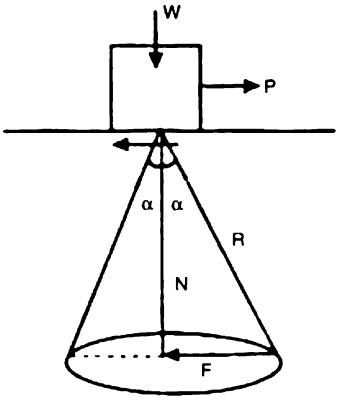

Fig. 2.6 Use of a cone of friction to determine if the object is sliding

is assumed to be at rest, if outside then it is assumed to be sliding. This concept is developed with respect to rendering of multi-finger contact systems in the following two sections.

2.4 Friction Cone Algorithm

The friction cone algorithm allows simple modelling of the friction between the fingers and the object to be built into the haptic rendering algorithm. It has the benefit of sufficient simplicity to be quick and efficient to implement, and sufficiently detailed to give the individual a good perception of manipulating common objects based on the friction needed to form simple multi-fingered grasps.

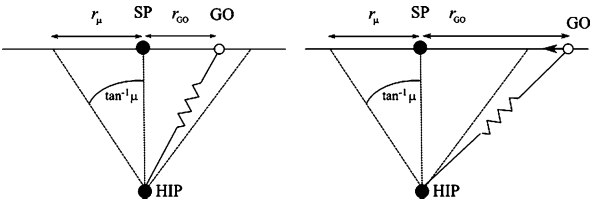

Fig. 2.7 Relationship between the god-object ($\overline{\text{GO}}$) and the bounding circle defined by the friction cone [31]. *Left*: shows the god-object within the circle and thus it remains static on the surface. *Right*: shows the god-object outside the bounding circle so it should be moved to the nearest point on the perimeter

The algorithm is based on the Salisbury and Zilles 'god-object' concept [51] which uses a point based god-object to maintain the history of contact between the person's fingers and the object.

The algorithm is described in detail in [31, 34] so is given in outline here.

A haptic interface point ($\overline{\text{HIP}}$) is defined as the representation of the end-point of the haptic interface in the virtual environment. (The overline convention is used where several letters are used to represent a vector variable such as $\overline{\text{HIP}}$.) In the case of finger contacts this is assumed to be one of the fingers involved in the manipulation of the object. A collision detection algorithm is needed to identify when the $\overline{\text{HIP}}$ is within an object, and at this stage it may be assumed that contact has been made. On first contact between the $\overline{\text{HIP}}$ and the object a notional god-object ($\overline{\text{GO}}$) is placed on the surface of the object and from thereon a force vector is assumed to be opposing the contact with a spring like characteristic. That is a force is created on the haptic interface that is proportional to the vector $\underline{\text{v}} = \overline{\text{GO}} - \overline{\text{HIP}}$. The position of the god-object is then used to model friction and this can be best described by considering the traditional friction cone with an apex half angle of $\tan^{-1}\mu$ but inverted so the apex is at the $\overline{\text{HIP}}$. This representation can be seen in Fig. 2.6. The god-object remains on the surface, but the additional constraint is now that it must also remain within the boundary of the circle formed by the intersection between the surface polygon and the friction cone.

As the $\overline{\text{HIP}}$ moves within the object the god-object may move outside the circle boundary. At this point the object is sliding over the fingers and this is handled within the algorithm by moving the god-object to the closest point on the circumference of the circle Fig. 2.7. A subsequent calculation may move the object with respect to the $\overline{\text{HIP}}$ and this would cause the surface point and the circle to move. Slip may then continue if this movement causes the god-object ($\overline{\text{GO}}$) to move outside the friction circle.

2.4.1 Extended Friction Cone Algorithm (xFCA)

Our initial work used a customised algorithm to identify collisions between the haptic interface point ($\overline{\text{HIP}}$) and the contacted object. The advantage of the god-object

(GO) method, and therefore the friction cone algorithm (FCA), is that once a collision is identified, and for as long as contact remains (the HIP is inside the object), no further collision detection passes are required as the GO can traverse the surface of the object. If the topology of the object is known (for the common triangle mesh this is a pre-computed edge/vertex connectivity graph) then on each update of the HIP position, the GO is moved along the surface in the direction which will minimise the distance between GO and HIP. With efficient data structures the complexity of the mesh which can be traversed is bounded only by the available memory, and not by the processing speed. This method assumes that the initial collision is detected sufficiently quickly so that the HIP has not penetrated too deeply into the object. Often this calculation can occur at update rates significantly lower than typical for haptic rendering (0.8–1 KHz). See Melder and Harwin [32–34] and Melder [30] for a description of such a surface traversal algorithm.

However, it is often desirable that the HIP is not a single point and in fact has some volume. This reduces the problems of the point escaping through cracks between poorly generated surface triangles and facilitates the use of convex-convex collision detection libraries [11, 38, 46]. In this case surface traversal cannot be easily computed. If the two colliding objects can be approximated as convex hulls, then fast and predictable solutions for surface traversal can be implemented. One such method is to compute the Minkowski Sum of the two convex objects and proceed to traverse this combined surface as a single point [38], however, computing the Minkowski Sum becomes cumbersome for non trivial shapes undergoing rotations.

As efficient algorithms for collision detection between convex polyhedral are now widely available though well supported software libraries, such as SOLID [47] (dtecta.com) and Bullet [11] (bulletphysics.org) we prefer a more generalisable method of rendering convex-convex haptic interaction which we refer to as the extended friction cone algorithm xFCA. While the convenience of not having to perform repeated collision passes during haptic rendering is lost, convex-convex collision detection is efficient even for very high polygon count models if the convex hull is computed [46].

The xFCA algorithm can be summarised as follows:

1. When the HIP is in free space, the haptic cursor is mapped to that position, Fig. 2.8-1. At each simulation update the haptic cursor is moved to the new HIP position and a full collision pass is performed looking for any contact between the haptic cursor and other objects.
2. If a contact is found, Fig. 2.8-2, the god-object is created for that HIP-Object pair and the main xFCA loop begins. The haptic cursor remains with the GO on the surface.
3. During the same simulation frame, the penetration depth and normal returned from the collision detection system is used to move the GO to the surface of the object, Fig. 2.8-3. The new surface position of the GO is stored, along with the surface normal and plane at the point of contact.
4. The simulation now proceeds to the next frame. If the HIP has moved with respect to the previous frame, the position of the GO must be updated to reflect the new surface position. Before performing a collision pass, the GO is moved

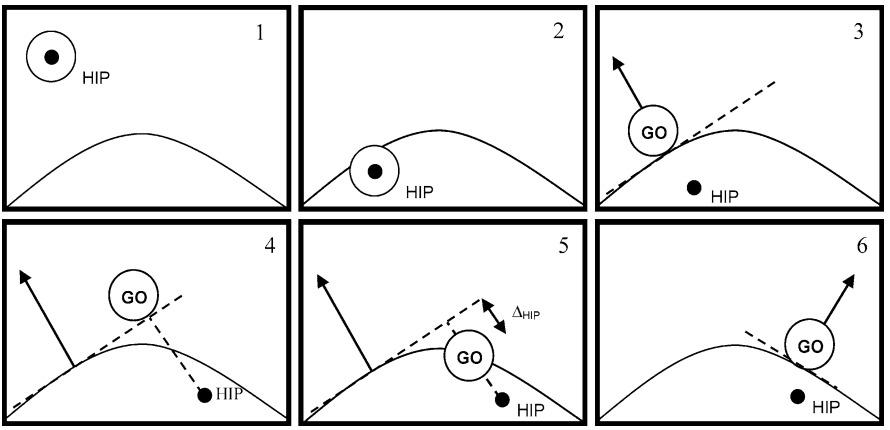

Fig. 2.8 The progression of the xFCA moving the god-object (GO) along a surface and the haptic interaction point ($\overline{\text{HIP}}$) into the object

along the surface plane to the point at which it is closest to the $\overline{\text{HIP}}$, Fig. 2.8-4, by removing that portion of $\underline{\text{v}}$ (the $\overline{\text{GO}}$ to $\overline{\text{HIP}}$ vector) which is in the direction of the surface normal $\mathbf{n}_s$:

$$\overline{P}^{+}_{GO} = \overline{P}_{GO} - (\underline{\text{v}} \cdot \mathbf{n}_s)\mathbf{n}_s$$

where $\underline{\text{v}} = \overline{\text{HIP}} - \overline{\text{GO}}$.

5. Next, the $\overline{\text{GO}}$ is moved down towards the $\overline{\text{HIP}}$ by a fixed amount, Δ_{HIP}, in order that the $\overline{\text{GO}}$ should be beneath the object's surface so the collision pass will return a positive contact, Fig. 2.8-5. There is more to this step which will be discussed next.
6. A collision pass is now performed to return a new penetration depth and contact normal for the GO-Object collision which is then used to reposition the $\overline{\text{GO}}$ on the surface of the object, Fig. 2.8-6.

Stages 3–6 continue until either, the $\overline{\text{HIP}}$ crosses the surface plane, or the collision pass fails to find contact between the $\overline{\text{GO}}$ and the object. Both of these conditions are assumed to indicate that the $\overline{\text{HIP}}$ has lost contact with the object. At the end of each xFCA cycle, once the $\overline{\text{GO}}$ is positioned on the true object surface, the distance between the $\overline{\text{GO}}$ and the $\overline{\text{HIP}}$ is used to calculate a spring force which is applied to the haptic device and the virtual object.

The reason the $\overline{\text{GO}}$ is not moved the full distance to the $\overline{\text{HIP}}$ is that for thin objects or large forces this would result in push through. However, if it is too small compared to the radius of curvature of the surface then the $\overline{\text{GO}}$ may fail to retain contact even though the $\overline{\text{HIP}}$ is still within the object, Fig. 2.9, left.

If the maximum distance that the $\overline{\text{HIP}}$ will move in a single time step, d_{max}, is known, it is possible to approximate the minimum Δ_{HIP} necessary to guarantee sustained contact on a surface with radius of curvature r_c, Fig. 2.9, right. This results

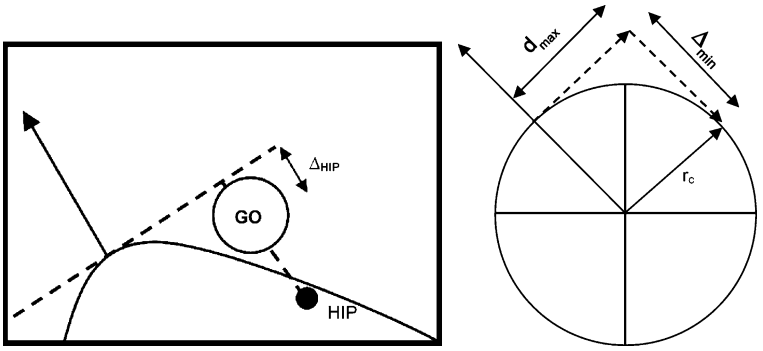

Fig. 2.9 *Left*: Contact may be unintentionally lost if the distance the $\overline{\text{GO}}$ is moved towards the $\overline{\text{HIP}}$ is too small. *Right*: The minimum distance the $\overline{\text{GO}}$ should be moved towards the $\overline{\text{HIP}}$ as a function of surface curvature and the maximum distance the $\overline{\text{HIP}}$ can move in a single simulation step

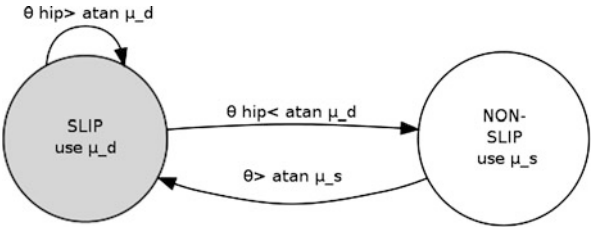

Fig. 2.10 Finite state machine to track transitions between slipping (dynamic coefficient of friction) and static conditions

in the following relationship:

$$\Delta_{HIP} = \sqrt{2r_c^2 - (d_{max})^2}$$

The value of Δ_{HIP} can also be used, along with the smallest dimension of the haptic cursor, to define the minimum depth an object can be penetrated before push-through becomes possible. In practice, the common haptic update rate of 1000 KHz or more results in very small values of d_{max}, even at high velocities.

2.4.1.1 Modelling Dynamic and Static Friction

Implementing static friction requires a slight modification from the original FCA. The main difference is that, if the friction cone is large compared to d_{max} when it is exceeded, the requirement for a bounded dmax is broken and subsequently the $\overline{\text{GO}}$ may be moved a large distance away from the surface causing a loss of contact. To solve this we introduce the concept of the Surface Object (SO). The SO represents the theoretical position of the $\overline{\text{GO}}$ if there were no friction, is updated at every time step and is assumed not to lose contact unless the $\overline{\text{HIP}}$ leaves the surface. However, as it is the GO which represents the true current surface position, contact is only lost

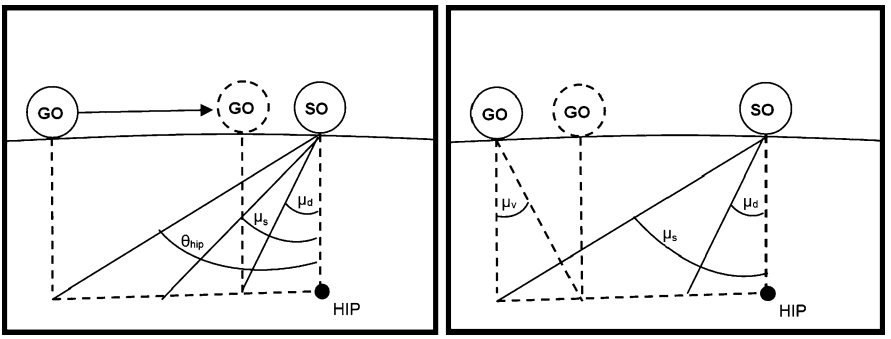

Fig. 2.11 *Left*: If using Coulomb friction, when the $\overline{\text{GO}}$ exits the static friction cone it is placed on the edge of the, smaller, Coulomb friction cone. *Right*: To create a force proportional to speed (viscous friction) the $\overline{\text{GO}}$ is moved a maximum distance proportional to penetration depth at each time step bounded by the Coulomb friction radius

if the $\overline{\text{HIP}}$ crosses the GO's plane and not that associated with the SO. Should the $\overline{\text{SO}}$ lose contact at any time, it remains positioned at the last known point of contact on the surface until either another surface position is found or the $\overline{\text{HIP}}$ transitions the GO's contact plane.

An easy adaption to this algorithm allows a simplified version of Stribeck's effect to be implemented. If the finger is not sliding with respect to the object the algorithm is computed based on a static coefficient of friction, but if slipping is occurring then a dynamic coefficient of friction is used. The finite state machine shown in Fig. 2.10 is used to track the friction state and considers only the two conditions, slipping or static.

When the angle θ_{HIP} exceeds the static friction angle the GO is moved to the edge of the Coulomb friction cone defined by μ_d where $\mu_d < \mu_s$, Fig. 2.11, left. At each subsequent time step, if the $\overline{\text{GO}}$ is outside the Coulomb friction cone it is moved back to the nearest edge. If the $\overline{\text{GO}}$ is found inside the Coulomb friction cone, it is left in place. This gives a stick-slip effect with a force proportional to normal force during surface slip.

Viscous friction can also be neatly integrated into the same framework. Viscous friction is proportional to the surface velocity as well as normal force and requires that the $\overline{\text{GO}}$ is moved a fixed distance relative to its own position, not the $\overline{\text{SO}}$ position. From the friction model in Fig. 2.7 it is clear that Coulomb friction defines a base level of friction during surface slip and viscous friction increases it above this level relative to velocity. To mimic this, during the slip phase of the friction model, the $\overline{\text{GO}}$ is moved towards the edge of the Coulomb friction circle but only as far as the edge of the viscous friction circle centred on the GO. This means that, at each time step, if the $\overline{\text{SO}}$ is moved away from the $\overline{\text{GO}}$ by less than the radius of viscous friction only Coulomb friction will be felt. If the surface velocity increases, the position of the $\overline{\text{GO}}$ will begin to trail behind creating a greater frictional force, Fig. 2.11, right.

The algorithm is simple and effective. Simple because changing between static and dynamic coefficients of friction simply alters the diameter of the bounding circle formed by the intersection between the friction cone and the object surface. Effective because it allows the person to reduce their grip force to a minimum needed to support and move the object.

2.4.1.2 Handling Non-convex Hulls

As efficient algorithms exist to decompose non-convex objects into separate convex pieces any non-convex object can be rendered haptically in this manner by treating it as multiple separate HIP-Object contacts, each with its own local $\overline{GO}$ stored to calculate forces arising due to penetration depth. However, it should be noted that it becomes possible to slide into gaps between convex pieces due to the force from opposing $\overline{GO}$s cancelling each other out, though this can be minimised with careful convex-decomposition.

Certain classes of object do not decompose easily into large convex pieces, smooth concave curves for example (coffee mug handle). If rendering quality is to be preserved in such cases, or discontinuities such as slipping into cracks between convex pieces are to be removed entirely it is necessary to perform rendering directly between non-convex objects. For the xFCA this would mean computing the penetration depth of intersecting non-convex polyhedra, a non-trivial task which can be, in the worst case, $O(nm)$ for the number of polygons in each mesh [22].

It is possible to make use of the fact that distance calculation between non-overlapping non-convex objects is achievable in real time and hence some researchers employ a haptic rendering algorithm based on repulsion rather than penetration to avoid the situation of overlapping non-convex objects [24]. Without very high stiffness, this technique results in the haptic cursor not visibly contacting the object which may not be desirable in realistic simulation.

The technique used by the authors for general non-convex haptic rendering is similar to this repulsive technique but is in fact closer to the constraint based methods used in animation and games physics engines and in fact may be considered an intermediary step between simply coupling haptic interfaces directly into constraint based physics engines and removing a separate haptic rendering stage entirely. Although this is straightforward algorithmically it is not yet widely employed because, at the time of writing, computing power is not yet sufficient to allow common game physics engines, such as the Bullet library, to run at real time speeds with haptic time steps ($<$1 mS) even for a seemingly simple virtual environment.

The principle of the technique is now briefly described. Using this method, the haptic cursor is treated as a very light mass object coupled to the $\overline{HIP}$ by a spring with a high stiffness and a damper and its position is updated each time step based on standard Newton dynamics equations as described in Sect. 2.5.1. As the mass is so light, and the spring stiffness comparatively high, when in free space no forces are perceived by the user as the haptic cursor does not lag far behind the $\overline{HIP}$ (some

tuning of the damping constant is required to prevent vibrations). The haptic cursor is continually checked against other objects in the environment for a collision. Should a collision occur, a frictionless constraint based linear complimentary problem (LCP) model is formed and solved to calculate the forces and torques which, when applied to the haptic cursor, will exactly cancel out acceleration in the direction of further penetration and effectively cause the objects to slide over each other. Formulation of this type of LCP for object dynamics is a well studied problem, see the original work by Baraff [3] for a simple explanation and formulation, though many more efficient implementations have since been proposed [1, 43].

Depending on the requirements of the simulation, the force based interaction between the haptic cursor and contacted object may be entirely decoupled from the wider physics engine which governs movement of dynamic objects in the scene. From the point of view of the haptic rendering constraint model it is operating on an object which is fixed in space. From the point of view of the physics engine, the haptic cursor does not exist except as a disturbance force added into the global force matrix. The result is that even if the physics engine slows temporarily, the haptic rendering proceeds uninterrupted and the perception of the user is of the inertia of the contacted object(s) increasing.

Implementing a combined penalty and constraint based haptic rendering algorithm in this manner is practical because solving the LCP resulting from contact between only two objects where one is stationary is achievable within the haptic time steps even when many points of intersection are found.

However the following points should be taken into consideration:

- Calculation of friction between the contacted object and haptic cursor is approximated using the FCA as the addition of the extra constraints into the LCP contact problem results in unpredictable solution times.
- It is believed that newer techniques for approximating friction in LCP based object dynamics may solve this but it is untested by the authors.
- To maintain haptic update rates it is preferred that the haptic cursor should be decomposable into convex parts or the collision detection between complex non-convex objects will be too slow for haptic rendering

2.5 Haptic Rendering and Manipulation of Virtual Objects

Multi-finger contact in a virtual environment allows complex and direct manipulation of physical objects. Any haptic rendering algorithm requires a two stage physics engine. Stage 1 is to identify the forces within the system of objects due to any points of contact or collisions, stage 2 is to apply the appropriate physical laws to compute the response of these objects. Stability of the haptic device requires rapid computation of the response of the simulated physics to the applied forces to those objects with a direct attachment to the person (through the haptic device). The following discussion will be restricted to rigid object manipulation where the object shape, mass and inertia remains unchanged through the manipulation. Extending the work

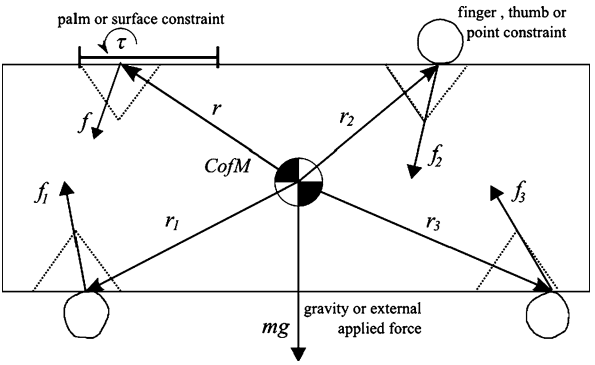

Fig. 2.12 Residual forces and torques on a grasped object

to rendering flexible objects with visco-elastic properties and the ability to reshape the object through cutting, wearing and drilling during the rendering remains a difficult challenge. However for the most part this application area is sufficiently well covered by haptic interfaces using a single point of contact via a surrogate tool, so has thus far not demanded academic attention.

The friction cone algorithm described naturally provides a key piece of information for multi-finger manipulation of rigid physical objects, that is the force vector at each contact point, that is to say each finger. This force vector is directed from the $\overline{\text{HIP}}$ to the $\overline{\text{GO}}$ and is proportional to the length of this vector. The proportional constant is directly related to the perceived stiffness of the surface of the rigid object. Manipulation of the object now becomes, in essence, the application of Newton-Euler dynamics to the virtual object.

Consideration of the manipulation of the object is most conveniently done by separating translation and rotation of the object, and applying Newton's law to the former and Newton-Eulers law to the latter case. If the object is at rest the sum of all force vectors on the object must be zero. Should any of the forces applied to the object deviate, this will give rise to a small unbalanced residual force which, once determined, can be used to calculate the acceleration of the object and hence by integration, the updated velocity and position. Forces on the object can arise from finger contacts, contact between the object and other objects, or forces due to gravity. The algorithm will of course still work if the author chooses to represent gravity as an upward acceleration of the object, but the normal convention is to consider gravity as a constant force vector with the direction fixed in the universal frame.

2.5.1 Computation of Translation from the Residual Forces

Considering the forces shown in Fig. 2.12 we can sum the forces on the virtual object. To implement Newton's law we must give the object a notional (real) mass m, and this mass can be considered to be entirely located at the object centre of

gravity which can be located within a coordinate frame attached to the object. It is usually convenient to make the centre of mass the origin of this coordinate frame for both the translational and rotational calculations. From Fig. 2.12 it can be seen that the residual force is calculated as the sum of forces on the object, in this illustration this is

$$\underline{\mathrm{F}}_{\text{residual}} = \sum_{i=1}^{n} \underline{\mathrm{f}}_i + \underline{\mathrm{g}}$$

Newton's equation is now used to compute the acceleration so

$$\underline{\mathrm{a}} = \underline{\mathrm{F}}_{\text{residual}}/m$$

and from the acceleration the velocity and position are readily calculated by Euler integration as

$$\underline{\mathrm{v}}_{new} = \underline{\mathrm{v}}_{old} + \underline{\mathrm{a}}\Delta_t \tag{2.1}$$

$$\underline{\mathrm{p}}_{new} = \underline{\mathrm{p}}_{old} + \underline{\mathrm{v}}_{new}\Delta_t \tag{2.2}$$

Δ_t is the loop time of the control algorithm. In general Δ_t needs to be small since large computational delays result in haptic instabilities [10]. In practice most hardware dictates that Δ_t is of the order 1 to 2 milliseconds.

2.5.2 Computation of Rotations from the Residual Torques

A similar but slightly more complex approach is used to compute and update the rotation parameters for the object coordinate frame. A variety of methods are available to specify the rotation of the object coordinate frame with respect to the base frame. A common method in computer graphics and robotics is to specify rotations as a 3×3 orthogonal matrix that can be used on a 3×1 vector or n vectors expressed as a $3 \times n$ matrix of vectors. This rotation matrix will rotate the components of vectors expressed in the object's coordinate frame into in components in the base frame. This rotation matrix has a dual use, the first to express this coordinate frame rotation, the second (and related use) is to convert the objects inertial matrix J from the local to the base frame.

The full Newton-Euler relationship is

$$\underline{\mathrm{T}} = J\underline{\dot{\omega}} + (\underline{\omega} \times J\underline{\omega})$$

where T is the opposed torque, $\times$ represents the cross product, and J the object inertia matrix. However the second term in this equation $(\underline{\omega} \times J\underline{\omega})$ contributes the 'gyroscopic term' and in practice can be ignored. Using this simplified Newton-Euler relationship we can rearrange the equation to compute the angular acceleration from the object inertia and the computed torque. That is

$$\underline{\dot{\omega}} = \underline{\mathrm{T}}J^{-1} \tag{2.3}$$

A similar technique to the calculation of the residual forces can be used to compute the residual torque on the object, that is—to use the vector between the haptic interface point ($\overline{\text{HIP}}$) and the god-object ($\overline{\text{GO}}$) to represent the forces applied to by the fingers. These forces represent a set of torques, conventionally computed around the centre of gravity, that can be summed to estimate the residual torque causing the angular acceleration, and hence the angular velocity and ultimately the rotational terms expressed in the matrix R.

Other sources of torque such as area of contacts, can be added to the torques due to the finger contacts, before the computation of angular acceleration is made, thus the residual torque ($\underline{\text{T}}_{\text{residual}}$) is

$$\underline{\text{T}}_{\text{residual}} = \sum_i r_i \times f_i + \sum_i \tau_i$$

where $r \times f$ are the torques due to individual point contacts, usually assumed to be point representations of the finger contacts, and τ are the torques applied through area contacts.

An inertia matrix is needed to complete the calculation in (2.3). This is most conveniently retained as the inertia along the three principal axes of the object, and these are often aligned with the local object coordinate frame. The inertia is stored in a matrix form that is

$$A = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

A prior estimate of the rotation matrix is needed to convert this inertia matrix into the matrix representation the instantaneous rotation, that is

$${}^{A}J = RAR^{T}$$

The full calculation for the angular acceleration is thus

$$\dot{\underline{\omega}} = {}^{A}J^{-1}\left(T - \underline{\omega} \times {}^{A}J\underline{\omega}\right)$$

but is simplified to ignore the 'gyroscopic term' to be

$$\dot{\underline{\omega}} \approx RJ_p^{-1}R^T T$$

$\dot{\underline{\omega}}$ represents the angular acceleration around the three axes of the base coordinate frame, and these can be treated as independent for the first integration to compute the instantaneous angular velocity. That is

$$\underline{\omega}_{new} = \underline{\omega} + \dot{\underline{\omega}}\Delta_t$$

However the second integration needs to compute the absolute position of the object, and to this end is calculated in matrix form. The first stage is to compute the

skew matrix S

$$S = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

S is a convenient method to represent instantaneous angular velocities since it can be shown that $\dot{R} = SR$.

The derivation of this expression follows by differentiating the definition of orthogonal matrix $RR^T = I$ and a good discussion of this is given in Craig [13]. If integration happens over a sufficiently small interval then the rotation matrix can be written as

$$\dot{R} \approx \Delta R = I + S\Delta_t$$

The Euler integration is then

$$R_{new} = R\Delta R = R(I + S\Delta t)$$

where the new rotation matrix is R_{new} and the new vector of rotational speeds is $\underline{\omega}_{new}$.

2.6 Multi-body Dynamics for Haptic Rendering

Realistic simulation of multiple moving bodies for natural object manipulation with haptics is computationally challenging. A full discussion of implementation issues in multi-body dynamics simulation is out of the scope of this chapter, although the following discussion is based on Baraff [3], Anitescu and Potra [1] and Stewart and Trinkle [43]. The following points should be considered when implementing multi-body physics simulation for haptics:

- At haptic update rates, even the most efficient algorithms are unlikely to be fast enough to maintain real-time solutions for large numbers of objects
- The haptic sense is highly sensitive to force discontinuities
- The common shortcuts used in games physics engines are likely to be manifested as force discontinuity and as such will be detected and perceived by the user as an inconsistency
- The haptic sense is sensitive to all objects in physical contact, thus the haptic rendering must consider the energy coupling through all objects connected to the physical haptic device. An example would be the haptic rendering of a tray carrying a tea service.

Considering these points, a framework for a multi-body haptics simulation can be formulated to maximise the use of computing resources and minimise discontinuities perceptible to the user.

The current computing trend towards parallel computing architectures means that highly multi-threaded software solutions can be effectively employed to help

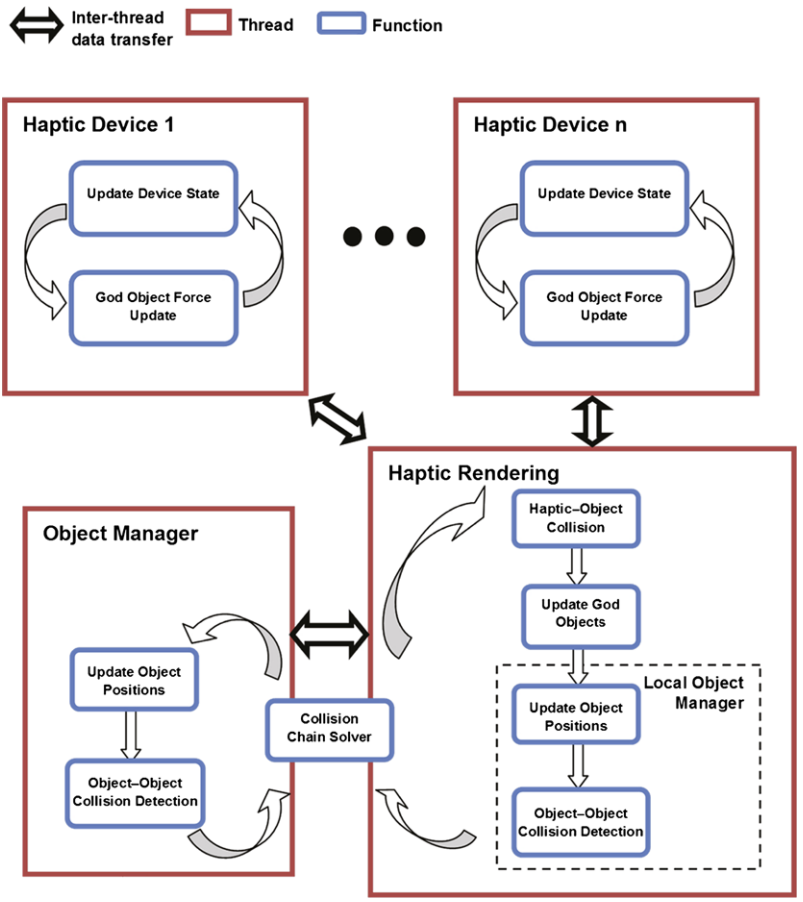

Fig. 2.13 Example threading model and information interchange for a multi-body multi-finger haptic simulator

achieve the real-time needs of a haptically enabled simulation. It is common to separate graphics rendering into its own or multiple threads and haptics and physics rendering can be treated in a similar manner. An example multi-threading structure for a multi-body haptic simulator is shown in Fig. 2.13.

2.6.1 Multi-threaded Haptic Renderer

In this model, a strict threading priority hierarchy is maintained and an important aspect of the implementation is that during interchange of thread safe information, no thread is ever made to wait on a lower priority thread for a non-deterministic length of time, usually the time required to make a copy of a thread safe data structure.

Each haptic device has its own dedicated thread, the highest priority, which communicates with the device, updates the force commanded based on the current stored location of any god-objects associated with it but does not update the god-objects themselves. This means that should the haptic rendering ever slow, stability of the haptic device is not compromised, but an increase in dynamic friction may be perceived.

The next highest priority threads are those associated with haptic rendering, i.e. collision detection between haptic cursors and scene objects and the movement of the god-object along the surface of a contacted object using the FCA. The haptic rendering threads may also be tasked with updating a small number of contacted objects, the reasons for this are discussed next.

2.6.2 Multi-threaded Object Manager

The object manager threads are responsible for the motion of dynamic objects. There are three parts to this: the integrator, which calculates future velocities and positions of objects based on the accumulated forces and torques acting on them, the collision detection system, which reports the points of contact, and the constraint handler, which calculates appropriate forces, torques and impulses so, on the next integrator pass, the motion of objects does not result in further interpenetration.

The rigid-body dynamics algorithm has the following steps:

1. Apply all external forces acting on objects (haptics, gravity etc.)
2. Process all objects and find all points of contact
3. Group all contacts into collision chains
4. Solve each collision chain to find new accelerations and velocities
5. Update each object's state vector and repeat

External forces are applied first so that they are taken into account when the contact matrices are formulated and solved. Once all the points of contact have been identified they are grouped into collision chains. Contact points are part of the same collision chain if they are connected via a movable object. Two movable objects sitting apart but on the same static floor would be part of separate collision chains. If the movable objects were touching they would be part of the same collision chain.

The purpose of separating objects into collision chains is that each chain is an independent system and can be solved in parallel. Once all contact points are grouped into chains they are assigned to separate threads to be solved. A pool of pre-existing threads is used to remove the overhead associated with creating and destroying them.

As the object dynamics is decoupled from the haptic rendering it can be updated at a different rate. Ideally, the system will attempt to maintain as high a frame rate for the physics as possible by using a variable length time step. However, if the contact solver begins to slow too much, the step size will grow large enough for the user to feel objects judder instead of move smoothly, this is likely to happen long before it can be detected visually. To prevent this, the time step is capped

at 0.002 ms (500 Hz). If the previous time step took longer, the simulation will fall behind real-time rather than take larger and larger steps. The advantage of this method is that neither the objects nor the haptic rendering will become unstable, if the object update slows significantly then the user will perceive an increase in the apparent inertia of the object but will still be free to haptically explore the object.

Once a haptic device makes contact with an object, if required the rendering thread can take over responsibility for updating it. Part of the haptic rendering loop now includes the same functions as the object manager and any objects assigned to run at the haptic update rate are processed here. If the object being updated by the haptic thread makes contact with another object, it too becomes the responsibility of the haptic thread. When the number of objects in the collision chain assigned to the haptic thread exceeds a predefined safe limit (usually very low, 1 to 3 objects) for guaranteeing update rate, the whole chain reverts to the responsibility of the Object Manager.

The further advantage of this technique is that stability problems associated with grasps involving objects with very low rotational inertia can be minimised by updating their position at haptic update rates. If multiple objects are in contact so that the chain cannot be updated at haptic rates, the increased inertia and energy loss due to the stack of objects are still likely be sufficient to remove and instabilities.

2.7 Discussion

The evidence for providing good multi-finger multi-contact haptic interactions is clear, although many technical challenges remain. The transparency of the haptic device is important, and the approach of using low inertia haptic interfaces helps to achieve this. However this is at the expense of workspace so it has proved difficult to produce transparency in a large working volume. The approach by Barrow and Harwin [5] partially addresses the issue of transparency, workspace and high achievable stiffness but further development along this line would introduce higher costs and greater need to ensure the operator's safety. Smaller multi-finger workspaces are more readily achievable, in particular with the potential fall in cost of high quality devices such as the W3D and W5D produced by Entact Robotics (entactrobotics.com), the Novint Falcon and the Sensible Omni, although all three would require the tool surrogate to be replaced by a more appropriate interaction point such as a thimble. A strong limitation is self interference of the multiple haptic interface and although some work has been done in this area, more work is needed to allow rapid evaluations of the trade-off between the movements required by a range of tasks, and the capabilities of the device. For example most kinematics would not allow the operator to 'cross fingers'. In our own work on stroke rehabilitation, the natural workspace expected by the therapists included a full range of movement that included shoulder rotation. No device has yet achieved this capability, while still allowing individual free movement of the individual fingers and thumbs.

The algorithm discussed in this chapter provides a good method for doing multi-contact multi-finger interactions that is independent of device kinematics and will

Fig. 2.14 Illustration of multi-point contacts using two Phantom 1.5 haptic interfaces

work across a wide range of haptic devices. The threading approach ensures that high update speeds can occur on those parts of the system that require rapid information for stability, while still maintaining good visual rendering and object dynamics for the objects that are not contacted by the haptic device and operator. The method is robust to contact instability (due to threading) in the sense that dynamics of objects and haptic rendering are not decoupled but not reliant on one another. Should the object update slow it is simply perceived as an increase in the inertia. The method requires no models, and is such that objects held in a grasp with any number of contact points tend to centre in the grasp via the dynamics of the object. Thus transportation of the objects appears natural. The experimental setup used in this work is shown in Fig. 2.14 and there are also videos of the device published on Youtube under the user ReadingTHRIL.

As the technology moves forward these algorithms must be adapted to multi-contact interactions where there is significant delay, such as Internet delayed signal paths, as well as allow more complex material interactions such as soft body physics including the ability to cut, tear, drill and probe simulated materials such as tendons, muscles, bone, teeth, clays, woods, metals etc.

2.8 Conclusions

There are still many problems to be addressed in providing good multi-finger multi-contact haptic interactions. These include cost, workspace, transparency and device stiffness. However if these can be addressed there are likely to be a wide range of applications, ranging from education through to data interaction and computer interfacing. There is a need to understand the mechanisms and neuroscience behind grasp to ensure that the device is transparent to the tasks the operator is trying to perform and thus a brief overview of this was presented. As computing advances

so must the computer interfaces and metaphors for user interactions must change. In the words of Gauldie, Wright and Shillito, *3D modelling is not for wimps* [18, 39]. The algorithms discussed in this chapter allow for interactions to occur in 3D. Indeed the method allows a world where all digits of both hands can be used to manipulate and interact with rigid objects.

Acknowledgements Many individuals have contributed to this work, including technical and academic staff, as well as members of the tHRIL laboratory. The authors are pleased to acknowledge in particular the contributions made by Dr Nic Melder and Mr Sebastian McKnight.

References

1. Anitescu, M., & Potra, F. A. (1997). Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, *14*(3), 231–247.
2. Åström, K. J., & de Wit, C. C. (2008). Revisiting the LuGre friction model: stick-slip motion and rate dependence *IEEE Control Systems Magazine*, *28*(6), 101–114.
3. Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on computer graphics and interactive techniques* (pp. 23–34). New York: ACM.
4. Barbagli, F., Salisbury, K. Jr, & Devengenzo, R. (2003). Enabling multi-finger, multi-hand virtualized grasping. In *Proceedings of the 2003 IEEE international conference on robotics and automation, ICRA'03* (Vol. 1, pp. 809–815). New York: IEEE Press.
5. Barrow, A., & Harwin, W. (2008). High bandwidth, large workspace haptic interaction: flying phantoms. In *Haptic interfaces for virtual environments and teleoperator systems*.
6. Brooks, F. P. Jr. (1999). What's real about virtual reality? *IEEE Computer Graphics and Applications*, *19*(6), 16–27.
7. Brooks, F. P. Jr., Ouh-Young, M., Batter, J. J., & Kilpatrick, P. J. (1990). Project GROPEHaptic displays for scientific visualization. In *ACM SIGGraph computer graphics* (Vol. 24, pp. 177–185). New York: ACM.
8. Bullock, I. M., & Dollar, A. M. (2011). Classifying human manipulation behavior. In *IEEE international conference on rehabilitation robotics (ICORR 2011)* (pp. 532–537). New York: IEEE Press.
9. Canudas de Wit, C., Olsson, H., Astrom, K. J., & Lischinsky, P. (1995). A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, *40*(3), 419–425.
10. Colgate, J. E., & Schenkel, G. G. (1994). Passivity of a class of sampled-data systems: application to haptic interfaces. In *American control conference* (pp. 3236–3240).
11. Coumans, E., et al. (2006). Bullet physics library. Open source: bulletphysics.org.
12. Cox, M. J., Quinn, B. F., Newton, J. T., Banerjee, A., & Woolford, M. (2012). Researching haptics in higher education: the complexity of developing haptics virtual learning systems and evaluating its impact on students' learning. *Computers & Education*, *59*(1), 156–166. doi:10.1016/j.compedu.2011.11.009.
13. Craig, J. J. (1989). *Introduction to robotics: mechanics and control*. Reading: Addison–Wesley. ISBN 0-201-09528-9. UR call 629.892-CRA.
14. Cutkosky, M. R. (1989). On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, *5*(3), 269–279.
15. Endo, T., Kawasaki, H., Mouri, T., Ishigure, Y., Shimomura, H., Matsumura, M., & Koketsu, K. (2011). Five-fingered haptic interface robot: HIRO III. *IEEE Transactions on Haptics*, *4*(1), 14–27.
16. Flanagan, J. R., & Wing, A. M. (1990). The stability of precision grip forces during cyclic arm movements with a hand-held load. *Experimental Brain Research*, *105*(3), 455–464.

17. Flanagan, J. R., & Wing, A. M. (1997). The role of internal models in motion planning and control: evidence from grip force adjustments during movements of hand-held loads. *The Journal of Neuroscience*, *17*(4), 1519.
18. Gauldie, D., Wright, M., & Shillito, A. M. (2004). 3D modelling is not for WIMPS part ii: stylus/mouse clicks. In *Proceedings of eurohaptics* (pp. 182–189).
19. Haessig, D. A. Jr, & Friedland, B. (1991). On the modeling and simulation of friction. *Journal of Dynamic Systems, Measurement, and Control*, *113*(3), 354–362.
20. Haggard, P., & Wing, A. (1995). Coordinated responses following mechanical perturbation of the arm during prehension. *Experimental Brain Research*, *102*, 483–494.
21. Hyun, J.-W., Kumazawa, I., & Sato, M. (2007). A new measurement methodology of multi-finger tracking for handheld device control using mixed reality. In *Annual conference, SICE 2007* (pp. 2315–2320). New York: IEEE Press. doi:10.1109/SICE.2007.4421375.
22. Jiménez, P., Thomas, F., & Torras, C. (2001). 3d collision detection: a survey. *Computers & Graphics*, *25*(2), 269–285.
23. Johansson, R. S., & Westling, G. (1984). Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental Brain Research*, *56*(3), 550–564.
24. Johnson, D. E., & Willemsen, P. (2003). Six degree-of-freedom haptic rendering of complex polygonal models. In *Proceedings of the 11th symposium on haptic interfaces for virtual environment and teleoperator systems, HAPTICS 2003* (pp. 229–235). New York: IEEE Press.
25. Körding, K. P., & Wolpert, D. M. (2004). Bayesian integration in sensorimotor learning. *Nature*, *427*, 244–247.
26. Light, C. M., Chappell, P. H., & Kyberd, P. J. (2002). Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: normative data, reliability, and validity. *Archives of Physical Medicine and Rehabilitation*, *83*, 776–783.
27. Loureiro, R. C. V., & Harwin, W. S. (2007). Reach and grasp therapy: design and control of a 9-dof robotic neuro-rehabilitation system. In *Proceedings of the 2007 IEEE 10th international conference on rehabilitation robotics*, 12–15 June 2007 (pp. 757–763).
28. Loureiro, R. C. V., Lamperd, B., Collin, C., & Harwin, W. S. (2009). Reach and grasp therapy: effects of the gentle/g system assessing sub-acute stroke whole-arm rehabilitation. In *IEEE international conference on rehabilitation robotics* (pp. 755–760).
29. McKnight, S., Melder, N., Barrow, A. L., Harwin, W. S., & Wann, J. P. (2005). Perceptual cues for orientation in a two finger haptic grasp task. In *First joint eurohaptics conference and symposium on haptic interfaces for virtual environment and teleoperator systems*, 18–20 March (pp. 549–550). New York: IEEE Press.
30. Melder, N. (2011). *Multi-finger manipulation physics for haptic rendering*. PhD thesis, University of Reading. Call number, THESIS-R10821.
31. Melder, N., & Harwin, W. (2002). Improved rendering for multi-finger manipulation using friction cone based god-objects. In S. Wall, M. Wright, & A. M. Shillito (Eds.), *Proceedings of eurohaptics conference* (pp. 82–85). Edinburgh College of Art and University of Edinburgh.
32. Melder, N., & Harwin, W. S. (2004). Extending the friction cone algorithm for arbitrary polygon based haptic objects. In *Haptic interfaces for virtual environment and teleoperator systems* (pp. 234–241). New York: IEEE Press.
33. Melder, N., & Harwin, W. S. (2005). Force shading and bump mapping using the friction cone algorithm. In *First joint eurohaptics conference and symposium on haptic interfaces for virtual environment and teleoperator systems* (pp. 573–575). New York: IEEE Press.
34. Melder, N., Harwin, W., & Sharkey, P. (2003). Translation and rotation of multi-point contacted virtual objects. In *Proceedings of eurohaptics conference* (pp. 218–227). Trinity College Dublin and Media Lab Europe.
35. Pawar, V. M. (2013). *Exploring the influence of haptic force feedback on 3D selection strategies*. PhD thesis, University College London.
36. Rock, I., & Harris, C. S. (1967). Vision and touch. *Scientific American*, *216*(5), 96–104. doi:10.1038/scientificamerican0567-96.

37. Rock, I., & Victor, J. (1964). Vision and touch: an experimentally created conflict between the two senses. *Science*, *143*(3606), 594–596.
38. Ruspini, D. C., Kolarov, K., & Khatib, O. (1997). The haptic display of complex graphical environments. In *Proceedings of the 24th annual conference on computer graphics and interactive techniques* (pp. 345–352). New York/Reading: ACM/Addison–Wesley
39. Scali, S., Wright, M., & Shillito, A. M. (2003). 3D modelling is not for WIMPs. In *Proceedings of HCI international*.
40. Schneck, C. M., & Henderson, A. (1990). Descriptive analysis of the developmental progression of grip position for pencil and crayon control in nondysfunctional children. *The American Journal of Occupational Therapy*, *44*(10), 893–900.
41. Sheridan, T. (1992). *Telerobotics, automation and human supervisory control*. Cambridge: MIT Press. (Background) UR call 620.46-SHE.
42. Steinfeld, E. (1986). Hands-on architecture. http://www.access-board.gov/research/handsonarch/html/recommendations%20for%20standards.htm. December 1986. Adaptive Environments Laboratory, Department of Architecture, SUNY/Buffalo.
43. Stewart, D., & Trinkle, J. C. (2000). An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Proceedings of the IEEE international conference on robotics and automation, ICRA'00* (Vol. 1, pp. 162–169). New York: IEEE Press.
44. Tse, B., Harwin, W., Barrow, A., Quinn, B., San Diego, J., & Cox, M. (2010). Design and development of a haptic dental training system—haptel. In *Lecture notes in computer science* (Vol. 6192, Part II, pp. 101–108).
45. Turner, A. (Ed.) (1981). *The practice of occupational therapy: an introduction to the treatment pf physical dysfunction*. London: Churchill Livingstone.
46. van den Bergen, G. (1999). A fast and robust gjk implementation for collision detection of convex objects. *Journal of Graphics Tools*, *4*(2), 7–25.
47. van den Bergen, G. (2001). Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*.
48. Wall, S., & Harwin, W. (2001). Design of a multiple contact point haptic interface. In *Proceedings of eurohaptics conference* (pp. 146–148).
49. Wing, A. M., & Flanagan, J. R. (1998). Anticipating dynamic loads in handling objects. In *Proceedings of the ASME dynamic systems and control division* (Vol. 64, pp. 139–143). New York: ASME.
50. Wood, J., Magennis, M., Arias, E. F. C., Gutierrez, T., Graupp, H., & Bergamasco, M. (2003). The design and evaluation of a computer game for the blind in the GRAB haptic audio virtual environment. In *Proceedings of eurohaptics*.
51. Zilles, C., & Salisbury, K. (1995). A constraint-based god-object method for haptic display. In *IROS, international conference on intelligent robots and systems*.