Energy Constraints On Parameterized Models

Andrew Witkin†
Kurt Fleischer†
Alan Barr‡
†Schlumberger Palo Alto Research, 3340 Hillview Avenue, Palo Alto, CA 94304
‡California Institute of Technology, Pasadena, CA 91125

Abstract

A simple but general approach to imposing and solving geometric constraints on parameterized models is introduced, applicable to animation as well as model construction. Constraints are expressed as energy functions, and the energy gradient followed through the model's parameter space. Intuitively, energy constraints behave like forces that pull and parametrically deform the parts of the model into place. A wide variety of geometric constraints are amenable to this formulation, and may be used to influence arbitrary model parameters. A catalogue of basic constraints is presented, and results are shown.

Keywords — Constraints, Modeling, Animation

1 Introduction

A widely-used approach to modeling is to combine geometric primitives—such as cylinders, blocks, and bicubic patches—with a variety of operators—such as translations, rotations, booleans, and deformations—to form a model hierarchy. The task of constructing a model within this framework has two parts: building the hierarchy, and setting the internal parameters of the primitives and operators. Experience tells us that the second task is usually by far the more difficult and time-consuming, particularly when complex operators such as deformations are used. As the complexity of the model increases, the number of parameters becomes large, and they tend to interact in ways that make the model difficult to control.

The user knows at the outset what the objects being modeled are supposed to look like—how the pieces are supposed to fit together and move. The difficulty lies in finding settings of the parameters that achieve the desired effect. The utility of hierarchic modeling systems would be greatly enhanced if this tedious process could be performed automatically, permitting the user to state in terms of constraints the properties the model is supposed to have, without the need to manually adjust parameters to give it those properties.

In this paper we present a simple but general approach to expressing and solving constraints on parameterized model hierarchies. We formulate constraints as *"energy"* functions on the model's parameter space, non-negative functions with zeroes at points satisfying the constraints. We then sum the constraints' energy functions to create a single scalar function of the parameters, and move through parameter space to minimize the energy.

We refer to the constraint functions as energy functions not because they always model the energy of actual physical systems, but because they play a role similar to that of physical energy functions during the constraint solving process. For example, an energy constraint attaching points on the surfaces of two objects acts much like a spring that pulls the objects together. However, in addition to translating and rotating, the objects are free to vary their internal parameters, so, for example, a cylinder may vary its length or radius to meet the constraint. Although no familiar physical material deforms in this stylized way in response to applied forces, it is easy enough to imagine an unphysical material that does. Since we are using the energy analogy as a mechanism for building models, rather than for simulating physical phenomena, this kind of non-physical behavior poses no problem.

Our approach provides:

- *Self-assembling* models, whose parts move and deform parametrically from an initial configuration to one that satisfies the specified constraints.
- *Animated* models that, once assembled, may move in response to time-varying constraints while continuing to satisfy static ones.
- Generality and modularity: we can formulate a

wide range of constraints as energy functions, and use them to influence arbitrary model parameters. To implement a new constraint we need not know the details of other constraints, nor of the primitives and operators to which they will be applied. Similarly, new parameterized primitives and operators may be implemented without modifying existing constraints.

- Additivity: Energy functions compose by addition. The solution to a system of constraints is the solution to a single equation, the sum of the energy terms. This property is particularly valuable in dealing with overdetermined systems. While conventional algebraic methods return no solution to an overdetermined system, the energy minimum is a "compromise" that is sometimes acceptable, nearly always informative, and often easily repaired.

- Interactive control: since we satisfy constraints by moving through a curve in parameter space, the initial constraint solving process can itself be animated, permitting the user to assist the solver in escaping local energy minima, resolving ambiguities, etc.

A significant body of work in constraint-based and dynamics modeling for computer graphics is concerned with the the specialized problem of animating articulated bodies, particularly human and animal forms. These include Armstrong and Green, [1], Girard and Maciejewski, [6], and Wilhelms and Barksy [13]. Nelson's *Juno* editor employs non-linear geometric constraints in the context of a 2-D image editor. Dynamic models of elastic bodies for computer graphics are treated by Terzopoulos *et al.* in [12]. Closest to ours in approach are Barzel and Barr's dynamic constraints [3], although their current work focuses on constraining the motion of rigid bodies.

2 Energy constraints

A model hierarchy is a tree that defines the model's geometry through a collection of mathematical functions, three of which will concern us here. These are a *parametric position* function, $\mathbf{P}(u,v)$ that returns a 3-space point for each (u,v) pair, a *surface normal* function, $\mathbf{N}(u,v)$, that returns a surface normal vector, and an *implicit* or inside-outside function, $I(\mathbf{X})$, that returns a scalar given a 3-space point, such that $I=0$ for points on the object's surface, $I<0$ for points inside the object, and $I>0$ for points outside the object. One such collection of functions is defined for each leaf in the tree, and represents the combined effects of the primitive at the leaf and of all the operators on the path from that leaf to the root. Definitions of these functions are given in Appendix A.

In general, each primitive or operator possesses some real-valued parameters, for instance the radius of a sphere, a translation vector, or the bend angle of a parameterized deformation. The position, normal and implicit functions each depend on these parameters: when they change, objects' surfaces move. We refer to the union of all these parameters as $\mathbf{\Psi}$. Once the hierarchic structure is fixed, the state of the model is completely determined by the value of $\mathbf{\Psi}$. The notion of parameter spaces, and of motion along curves through parameter space, is familiar in computer graphics in the context of keyframe interpolation (see, e.g. [11].)

We will express geometric constraints in terms of the functions $\mathbf{P}$, $\mathbf{N}$, and I. For instance, if we wish to attach two surface points $\mathbf{P}_1(u_1,v_1)$ and $\mathbf{P}_2(u_2,v_2)$, at least one condition we must satisfy is $\mathbf{P}_1=\mathbf{P}_2$. A solution is a value of $\mathbf{\Psi}$ such that all the imposed conditions are met.

Rather than solving the constraint equations algebraically, we formulate constraints as energy functions, and move through parameter space according to the energy gradient. In general, to formulate an energy constraint, we must construct a non-negative smooth function $E(\mathbf{\Psi})$ such that $E(\mathbf{\Psi})=0$ at all and only values of $\mathbf{\Psi}$ for which the constraint is satisfied. The solutions to a set of n constraints are values of $\mathbf{\Psi}$ such that

$$E(\mathbf{\Psi})=\sum_{i}^{n} E_i(\mathbf{\Psi})=0,$$

so to combine constraints, the corresponding energy terms are simply summed. We are free to express E in terms of position, normal, or implicit functions, or any other quantities that may be extracted from the model tree.

Intuitively, energy constraints may be viewed as "forces" that pull the parts of the model into the desired configuration and hold them there, although they are not necessarily intended to be physically realistic forces. For instance, a simple attachment constraint might be implemented as a spring connecting the points, in which case we have

$$E_{spring}=\kappa\left|P_1(u_1,v_1)-P_2(u_2,v_2)\right|^2,$$

where κ is a spring constant, and the "force" vector in parameter space is ∇E_{spring}.

From an initial condition $\mathbf{\Psi}_0$, the energy E is minimized by numerically solving the differential equation

$$d\mathbf{F}(t)/dt = \nabla E, \mathbf{F}(0) = \mathbf{\Psi}_0,$$

for a fixed point of the parameter-space curve $\mathbf{F}(t)$, i.e. a point at which $\nabla E = 0$. The solution is a local minimum, although it is not guaranteed to be a zero. This is a steepest descent method for solving $\nabla E = 0$.

A variety of standard numerical methods may be used to solve the energy equation. The simplest of these, Euler's method, has

$$F(t_{i+1}) = F(t) + h\nabla E,$$

with h the step size. More sophisticated methods, such as Gear's method [5] should be used to obtain accurate, reliable results. Any solution method requires evaluation of ∇E. To do so, the summed energy must be differentiated with respect to each model parameter, which may be done numerically by varying each parameter in turn, re-evaluating E, and taking differences. The structure of the tree may be used to avoid the needless expense of differentiating energy terms with respect to parameters on which they do not depend.

A limitation of the method is that it may be trapped at spurious (i.e. non-zero) local minima of E. Our solution to this problem is user interaction. Such minima are usually easy to interpret geometrically, e.g. a single part has become stuck or been turned backwards. Presented with a bad answer, the user can often correct the situation by manually repositioning a part. In fact, with fast enough rendering to view the evolving solution dynamically, the user can literally push or pull on parts of the model with a pointing device, introducing a time-varying energy term, E_{user}, into the equation, to bump it out of local minima. This style of user interaction has been used to good effect in interactive image interpretation [7]. A different form of user control is obtained by selectively freezing and un-freezing model parameters, to decompose a large problem into a sequence of smaller ones.

3 A catalogue of useful constraints

A wide range of geometric constraints may be cast in the form of energy functions. Here we list a few basic constraints, most of which are used in the examples to follow. Many are subject to alternative formulations. Each energy function is multiplied by a weighting factor; these factors are not shown below.

- **Attachment to a fixed point in space:** The energy term

$$E = \left|\mathbf{P}^a(u_a, v_a) - \mathbf{Q}\right|^2$$

attaches a specific point on the surface of object a, defined by parameter point (u_a, v_a), to a specific point in space, $\mathbf{Q}$.

- **Surface-to-surface attachment:** Place specified points on two surfaces in contact. To acheive contact, the points must coincide. In addition, their tangent planes at those points must coincide, and the surfaces should not (locally) interpenetrate. These conditions can be encoded by

$$\begin{aligned} E &= \left|\mathbf{P}^a(u_a, v_a) - \mathbf{P}^b(u_b, v_b)\right|^2 + \\ &\quad \mathbf{N}^a(u_a, v_a) \cdot \mathbf{N}^b(u_b, v_b) + 1, \end{aligned}$$

where P^a and P^b are the positions at the two attachment points, and N^a and N^b are the unit surface normals at those points. This function is zero when the points coincide and the dot product of the normals is -1.

- **Floating attachment:** Attach a specified point on an object to *some* point on a second object, allowing the point of contact to slide freely on the second object. Our implementation of this constraint is similar to that for simple surface-to-surface attachment, but uses the second object's implicit function instead of its parametric position function:

$$\begin{aligned} E &= I^b(\mathbf{P}^a(u_a, v_a))^2 + \\ &\quad \mathbf{N}^a(u_a, v_a) \cdot \frac{\nabla I^b}{\left|\nabla I^b\right|} + 1, \end{aligned}$$

noting that $\nabla I^b / \left|\nabla I\right|$ is a unit normal to surface b where $I^b = 0$. A useful variation is double floating attachment, in which both points float on their respective surfaces.

- **Slider constraint:** Constrain a specified point on an object to a line in space:

$$E = PLD(\mathbf{P}^a(u_a, v_a), \mathbf{P}_1, \mathbf{P}_2)^2,$$

where PLD is the point-to-line distance function, and $\mathbf{P}_1$ and $\mathbf{P}_2$ are points on the line. Useful variations constrain the point to a line *segment*, or constrain two line segments to be colinear and to overlap (like segments of a telescope.)

- **Collision using the implicit function:** For objects possessing inside-outside functions, interference or collision constraints may be imposed without calculating surface intersections. The implicit function is zero everywhere on the object's surface, and, by our convention, negative inside and positive outside the object. To impose an anti-interference constraint, we transform the implicit function into a thin repulsive "force field" surrounding the object. At a single point, $P^a(u_a, v_a)$ on object a, a suitable energy function is

$$E = e^{-\kappa I^b(P^a(u_a,v_a))},$$

  where κ is a positive scale factor. This makes the energy high inside object b, tending to zero far from object b. Thus a point inside the object is repelled, the repulsion approaching zero off the object's surface. To implement a general interference constraint, this function must be integrated over u and v. An efficient implementation must make use of intelligent sampling, hierarchic bounding boxes, etc., which are beyond the scope of this paper.

- **Direct constraints on parameters:** It is sometimes useful to impose constraints directly on model parameters, for instance to establish default values, or to constrain relations among parameters. If α is some model parameter, then a defaulting constraint may be written as $E = (\alpha - \alpha_0)^2$, where α_0 is the default value. If α and β are two model parameters, then a linear relational constraint may be written as $E = (\alpha - k_1\beta - k_2)^2$, which says that the linear relation $\alpha = k_1\beta + k_2$ should hold between α and β. For instance, if we want one rod to be twice as long as another, we have $E = (l_1 - 2l_2)^2$, where l_1 and l_2 are the lengths of the rods.

4 Examples

This section presents three examples of the use of energy constraints to build and animate models. The first, a relatively simple one, is described in detail as a concrete illustration of the method. The second and third are described more briefly.

Pipefitting: A simple example illustrating self-assembly and adjustment of a variety of model parameters is shown in figure 1. A cylindrical pipe, subjected to a translation, a rotation, and a parameterized bend is to be fitted to other pipes at either end. All other parts of the model have already been assembled, and their parameters frozen.

The model hierarchy for the adjustable tube is described schematically by $Trans(Rot(Bend(Tube)))$:

- A tube is a primitive, whose axis is coincides with the x-axis in model space, and whose dimensions are controlled by three parameters —length, radius, and thickness.
- A bend (after [2]) is a parameterized deformation that maps the x-axis into a circular arc in the x, z plane. Its effect on shape is controlled by three parameters—start, stop, and amount. Start and stop determine the "tightness" of the bend, and amount is the angle between the ends of the curved axis.
- A quaternion rotation is specified by a 4-vector. Although any rotation may be specified by three Euler angles, we use quaternions to avoid gimbal lock.
- A translation is specified by a 3-vector of x, y, and z displacements.

The union of these parameters, $\boldsymbol{\Psi}$, is a 13-dimensional vector. Motion along a curve through $\boldsymbol{\Psi}$-space corresponds to some combination of bending, translating, rotating, and changing the tube's dimensions. Despite the simplicity of the example, this means that we have to come up with 13 independent numbers to pin down the model in the desired configuration. Doing this manually, or writing a special purpose program to do it, would have been unpleasant.

For our purposes, two pipes are attached if their axes join smoothly and they have the same radius and thickness. An "attach-pipes" (AP) energy constraint may be built by combining a surface-to-surface attachment constraint, as defined in the previous section, with relational constraints that make the radii and thicknesses agree. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be the endpoints of the two pipes' axes, i.e. the points to be joined, and let $\mathbf{N}_1$ and $\mathbf{N}_2$ be two unit vectors extending outward from the axis endpoints in a direction tangent to the axes. Then we want $P_1 = P_2$ and $N_1 \cdot N_2 = 0$. Additionally, we want $r_1 = r_2$ and $t_1 = t_2$ for the radii and thicknesses. In energy form, this gives us

$$E_{AP} = |\mathbf{P}_1 - \mathbf{P}_2|^2 + \mathbf{N}_1 \cdot \mathbf{N}_2 + 1 + (r_1 - r_2)^2 + (t_1 - t_2)^2.$$

To attach both ends of the moving pipe, we have as the total energy the sum of two terms of this form.

Prior to bending, rotation, and translation, the endpoints of the moving tube's axis are situated at the points $\mathbf{P}_1 = (length/2, 0, 0)$, and $\mathbf{P}_2 = (-length/2, 0, 0)$, with unit normals $\mathbf{N}_1 = (1,0,0)$ and $\mathbf{N}_2 = (-1,0,0)$ pointing outward from the ends. The two fixed pipes to which the moving one is to be attached each have a similarly defined endpoint and normal, which are constants. The transformed endpoints are $Trans(Rot(Bend(\mathbf{P}_1)))$ and $Trans(Rot(Bend(\mathbf{P}_2)))$, respectively, so their positions in model space depend on all the model parameters except radius and thickness. Using the fact that surface normals transform under deformations by multiplication with the inverse transpose of the deformation's jacobian ([2]), the transformed normals are $Rot(\mathbf{J}_{Bend}^{-1T}\mathbf{N}_1)$ and $Rot(\mathbf{J}_{Bend}^{-1T}\mathbf{N}_2)$, where J_{Bend}^{-1T} is the inverse transpose of the bend operator's jacobian, evaluated at P_1 and P_2 respectively (see [2] for the formula.)

To solve the constraints, we solve the equation $\frac{d}{dt}\mathbf{\Psi}(t) = \nabla E$ from a starting point $\mathbf{\Psi}(t_0)$, until we reach a point at which $\nabla E \leq \epsilon$, where ϵ is a small tolerance value. At any point $\mathbf{\Psi}$ in parameter space, we can evaluate $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{N}_1$, and $\mathbf{N}_2$, and hence the energy function defined in terms of them. To compute ∇E numerically, we first evaluate E at the current $\mathbf{\Psi}$, then add a small Δ to each parameter (i.e. each component of $\mathbf{\Psi}$) in turn, re-evaluate E and subtract the central value of E, obtaining the corresponding component of ∇E. Using Euler's method the new value of $\mathbf{\Psi}$ is given by $\mathbf{\Psi}_{t+1} = \mathbf{\Psi}_t + h\nabla E$, where h is a step size.

Figure 1 shows the model at the initial condition, at several steps toward the solution, and at the solution. Finally, we show a new solution obtained when one of the fixed pipes is moved.

Oldham linkage: An oldham linkage is used to transfer rotation between shafts that are offset in a plane perpendicular to their axes, by means of a system of tongues and grooves. Figure 2 shows several frames from the self-assembly sequence for an oldham linkage, and several frames from an animation sequence. The specification of this mechanism involved a variety of constraints, including sliders, surface-to-surface contact, and position and orientation constraints. This example illustrates the use of energy constraints both for assembly and for animation of the assembled model.

Cam and rocker arm: This example (figure 3) shows an already-assembled cam and rocker arm at several points in its cycle. This is a working model: the pressure of the spring on the valve head pushes the follower against the cam via the fulcrum. The follower "feels" the cam using an implicit-function interference constraint, so that if the cam were reshaped, the motion of the arm would change accordingly.

5 Conclusion

Energy constraints were shown to provide an effective means of building and controlling parameterized models. A principle advantage of the energy method is its generality: it does not depend on the details of the constraints used or the models to which they are applied. It is tolerant of over- and under-determined systems, and amenable to user interaction. Among its disadvantages, it can be numerically intensive—particularly when the equations become stiff—and it can be trapped in local minima. The second difficulty is largely overcome by effective user interaction.

Appendix A

In this appendix we define the position, normal and implicit functions in terms of the structure of the model tree. Our definition describes the SPAR modeling testbed [4], but is typical of hierarchic modelers in most relevant respects. Each leaf in the tree represents a primitive. The root is a camera, and the intermediate nodes represent operators. We denote the leaves by L^i, and the n nodes on the ascending path from leaf L^i to the camera as $O^{i,j}$, where $O^{i,1}$ is L^i, $O^{i,2}$ is the first operator above L^i, and O^i, n is the camera. (When we refer to the objects comprising the tree without regard to the paths they lie on, we index them as O_i.) For the purpose of sampling and rendering, each such path defines a distinct object, characterized by a coordinated bundle of mathematical functions. These include:

- A parametric position function, $\mathbf{P}^i(u,v)$, $\Re^2 \rightarrow \Re^3$, defined recursively by

$$\begin{aligned}\mathbf{P}^i(u,v) &= \mathbf{P}^{i,n}(u,v),\\ \mathbf{P}^{i,j}(u,v) &= \mathbf{T}^{i,j}(\mathbf{P}^{i,j-1}(u,v)), j \neq 1,\\ \mathbf{P}^{i,1}(u,v) &= \mathbf{P}^{L^i}(u,v),\end{aligned}$$

  where $\mathbf{T}^{i,j}$, $\Re^3 \rightarrow \Re^3$ is associated with operator $O^{i,j}$, defining the transformation it performs, and $\mathbf{P}^{L^i}$ is the primitive position function associated with L^i. By convention, the domain of $\mathbf{P}$ is the unit square $0 \leq u \leq 1$, $0 \leq v \leq 1$. In words, the primitive L^i generates 3-space positions on the object's surface as a function of u and v, and each operator transforms those positions.

- A parametric normal function, $\mathbf{N}^i(u,v)$, similar to the position function but generating surface normals. This is defined by

$$\begin{aligned}
\mathbf{N}^{i}(u,v) &= \mathbf{N}^{i,n}(u,v), \\
\mathbf{N}^{i,j}(u,v) &= \mathbf{N}^{i,j-1}(u,v)\mathbf{J}^{-1\mathrm{T}}(\mathbf{T}^{i,j}), j \neq 1, \\
\mathbf{N}^{i,1}(u,v) &= \frac{\partial \mathbf{P}^{L^i}(u,v)}{\partial u} \times \frac{\partial \mathbf{P}^{L^i}(u,v)}{\partial v},
\end{aligned}$$

where $\mathbf{J}^{-1\mathrm{T}}$ is the inverse transpose of the Jacobian matrix [2].

- An implicit (inside-outside) function, $I^i(\mathbf{X})$, $\Re^3 \to \Re^1$, such that the solution to $I^i(\mathbf{X}) = 0$ is the same surface defined by $\mathbf{P}^i$. The implicit function is defined recursively by

$$\begin{aligned}
I^{i}(\mathbf{X}) &= I^{i,n}(\mathbf{X}), \\
I^{i,j}(\mathbf{X}) &= I^{i,j-1}((T^{i,j})^{-1}(\mathbf{X}), j \neq 1, \\
I^{i,1}(\mathbf{X}) &= I^{L^i}(\mathbf{X}),
\end{aligned}$$

where $\mathbf{T}^{-1}$ is the inverse of $\mathbf{T}$, and I^{L^i} is the primitive implicit function associated with L^i.

References

[1] William W. Armstrong and Mark W. Green, *The dynamics of articulated rigid bodies for purposes of animation,* in *Visual Computer,* Springer-Verlag, 1985, pp. 231-240.

[2] Alan H. Barr, *Global and Local Deformations of Solid Primitives*, Proc. SIGGRAPH, 1984, pp. 21–29,.

[3] Ronen Barzel and Alan H. Barr, *Dynamic Constraints*, to appear.

[4] Kurt Fleischer and Andrew Witkin, *The SPAR modeling testbed,* to appear.

[5] William C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations,* Prentice-Hall, Englewood Cliffs, NJ, 1971

[6] Michael Girard and Anthony a Maciejewski, *Computataional Modeling for the Computer Animation of Legged Figures,* Proc. SIGGRAPH, 1985, pp. 263–270

[7] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, *Snakes: Active Contour Models,* Proc. International Conference on Computer Vision, 1987.

[8] V. C. Lin, D. C. Gossard, and R. A. Light, *Variational Geometry in Computer-Aided Design,* Proc. SIGGRAPH, 1981, pp. 171–178

[9] Greg Nelson, *Juno, a constraint-based graphics system* Computer Graphics, Vol. 19 No. 3, July 1985, pp. 235-243.

[10] Thomas W. Sederberg and Scott R. Parry, *Free-Form Deformation of Solid Geometric Models*, Proc. SIGGRAPH, 1986, pp. 151–160.

[11] Scott N. Steketee and Norman I. Badler, *Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control*, Proc. SIGGRAPH, 1985, pp. 255–262.

[12] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer, *Elastically Deformable Models*, Proc. SIGGRAPH, 1987.

[13] Jane Wilhelms and Brian Barsky, *Using Dynamic Analysis To Animate Articulated Bodies Such As Humans and Robots*, Graphics Interface, 1985.

Acknowledgements

All images were rendered by Kurt Fleischer using his modeling testbed system (M-BLORB) on a Symbolics LISP Machine. Michael Kass helped design some of the constraints, and provided helpful comments on early verions of this paper. We thank Demetri Terzopoulos, John Platt, and Michael Kass for valuable discussions and goop.

Figure Captions

Figure 1: Pipefitting. A pipe subjected to a parametric bend is attached to other pipes at either end using surface-to-surface attachments. Starting with the upper left we see the initial configuration, three steps in the assembly process, and the final solution. At the lower right is shown a new solution obtained by moving one of the fixed pipes.

Figure 2: An Oldham's linkage being assembled and moved using a variety of energy constraints including sliders. The system of tongues and grooves serves to transfer rotation through the floating disc while allowing translations of either shaft in the plane of the disc.

Figure 3: A working model of a cam and rocker arm, specified using energy constraints, shown at several points in its cycle. The spring creates a torque around the fulcrum, pushing the follower against the cam. The follower "feels" the cam using an implicit-function interference constraint.