

Constrained Navigation:

Finding your Way in Complex Environments

Andrew J. Hanson

Computer Science Department

Indiana University

Students: Eric Wernert,
Stephen Hughes

Outline

- **Introduction**
- **Approach:** Define Subspace of Viewing Parameters
- **Origins:** Geometry Visualization Problem
- **Generalizing Animation:** Constraint Manifolds and Viewing Fields
- **Methods for Designers**
- **Interpolation Tasks:** Velocity Problem
- **Examples/Video:** Terrain, Chemistry, Architecture
- **Conclusion:** CHI Tests and Future Work

Introduction

- **Finding one's way is a universal problem.**
"Lost in Space" syndrome: (SGI "dog," CAVE Condor, Math flying, VRML viewers)
- **Need Viewpoint.** *Where to stand.*
- **Need Goal, Object of Gaze.** *Where to look.*
- **Problem: Too much freedom.** User is often not domain expert but *domain client*, doesn't know relevance of scene elements.
- **Solution: Generalize Animation.** Allow larger space of motion, not just single path.
And instead of arbitrary motion in space, make movements *goal-directed*.

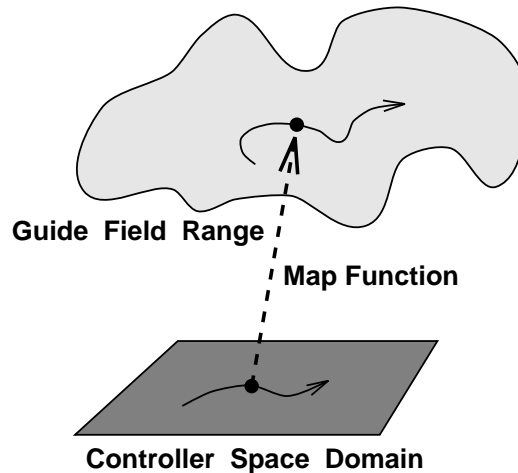
Approach:

Define subspace of viewing parameters

- **Generalized Sidewalk.** Map controller space to an allowed “sidewalk” in real space.
- **Attach Guide Fields.** Each sample point of sidewalk is assigned values (or algorithms) determining camera model.
- **Result: Fiber Bundle.** Rules for patchwork covering, etc., are closely related to classical mathematics of fiber bundles.

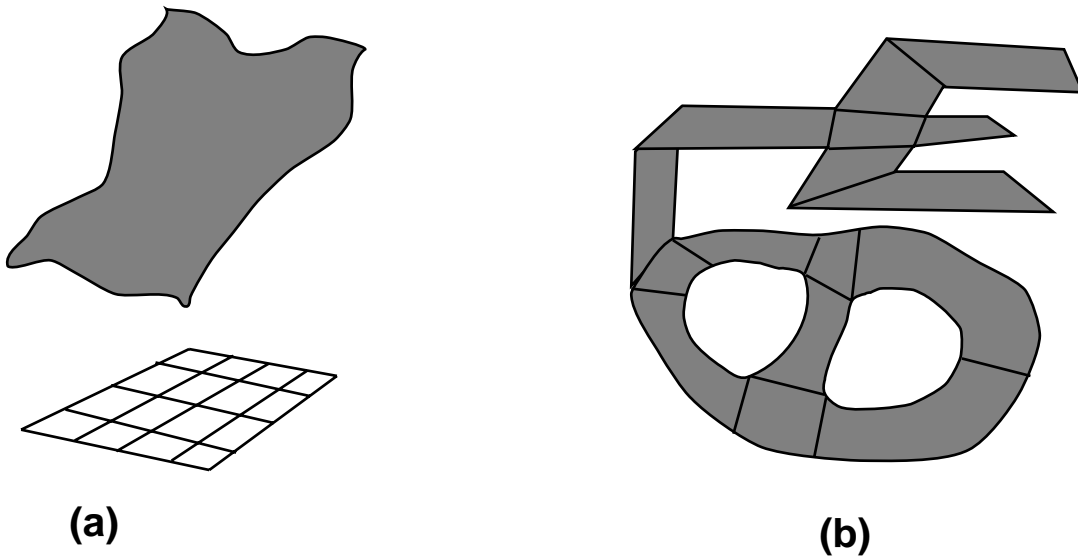
Approach, contd

Fiber Bundle Viewpoint:



- Think of space of *all possible guide field parameters*: ways to view and interact with a scene.
- Define a *projection* from this parameter space to controller space domain (u, v, \dots) .
- Choose a *local section* $G(u, v, \dots)$ with values in the guide field parameter space range.
- *Match* corresponding parameters on boundaries of patches sewn together to form controller space. (*Jumps are possible.*)

Approach, contd

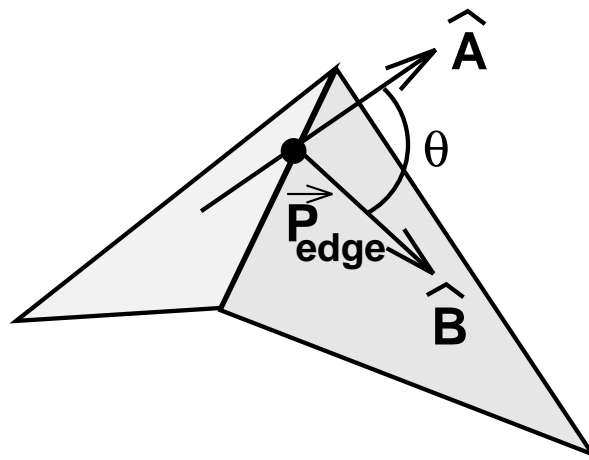


- (a) Simple domains: just map the mouse to rectangular patch.
- (b) Complex domains: require variant of “winged edge” patch definitions to piece together non-trivial manifolds. NO GUARANTEE that scene parameters are smooth and continuous from patch to patch.

Origins: A Geometry Visualization Problem

[Hanson and Ma, "Space Walking," in *Proc. Vis '95*]

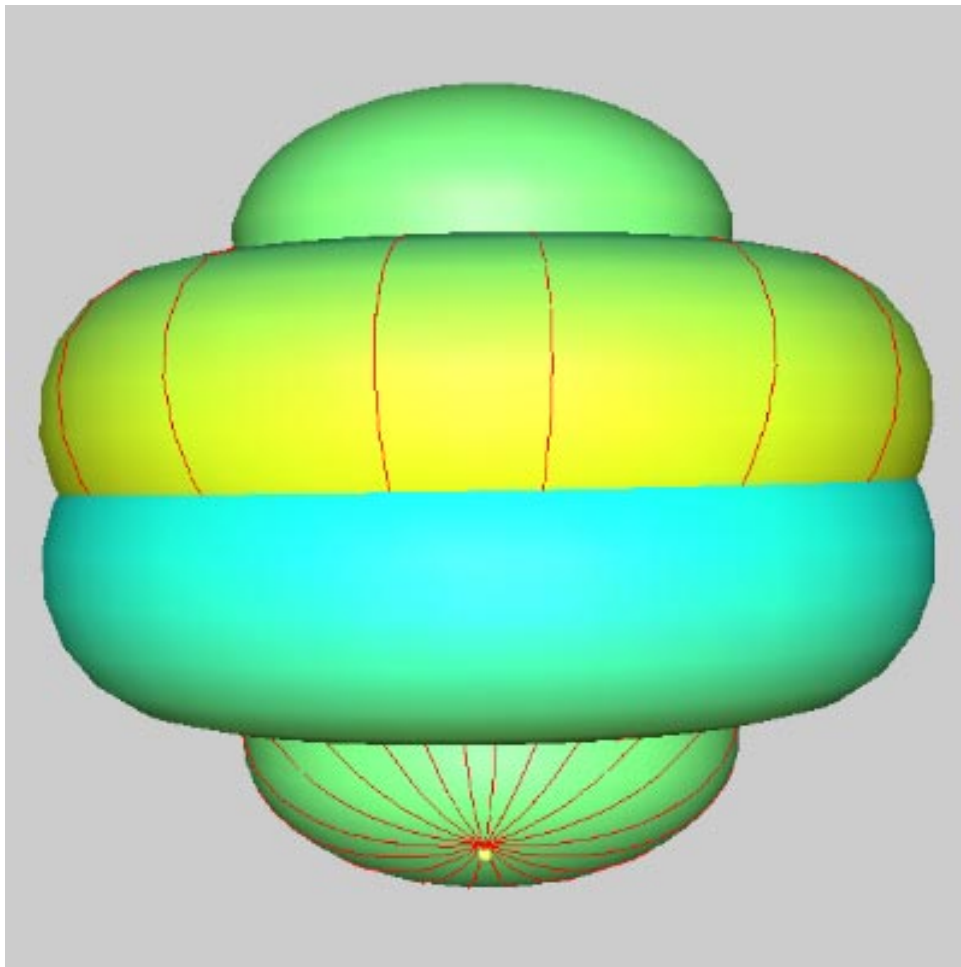
How do you explore a surface? \Rightarrow Make the surface's own geometry work for you.



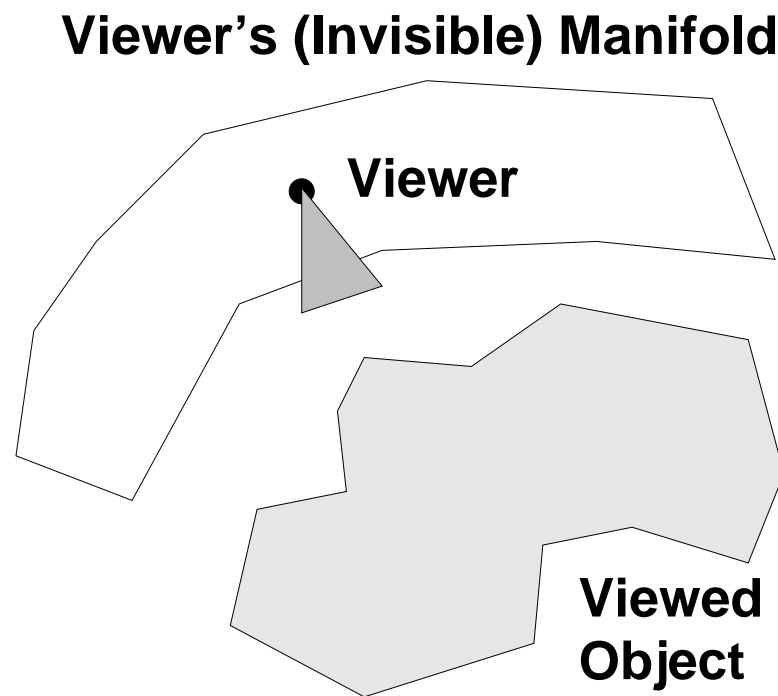
- Follow geodesic "trail." Gaze: perpendicular or parallel to surface.
- **Use surface as "treadmill."** Walking on the surface is generalization of "rolling ball" to "rolling surface."
- **Interesting group properties.** *Rolling ball demo.*
- *Problems:* Simple transition algorithms jerky, subject to "caustic" singularities for parallel gaze.

Example: Walking a knotted sphere

Viewer's camera follows red paths in space walking algorithm, so surface continually "rolls up to meet you."



Realization: Surface could be *Secondary*:



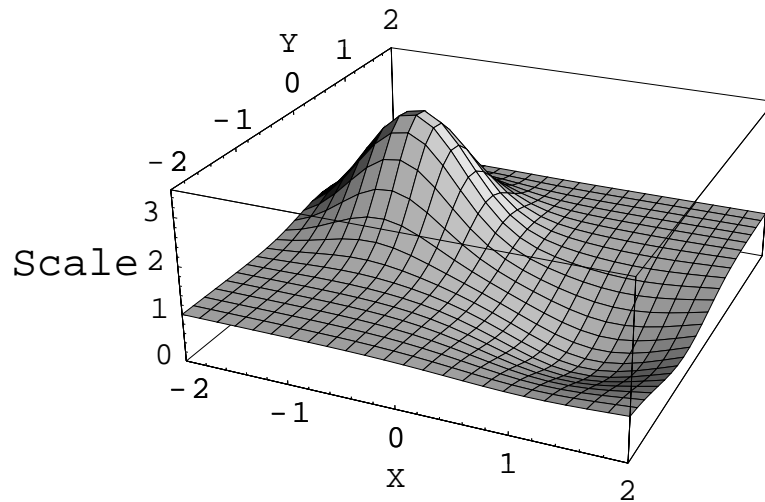
If the surface were transparent, you could still walk on it, but its purpose would be to see *something else* instead of itself.

So *conceal* the surface, and leave the underlying data as a sidewalk of spline control points: also potentially solves jerkiness problems.

Generalizing Animation: Controller Fields

- **1-D slider to N-D controller.** Film = 1-D slider (forwards and backwards in time). *Generalize* to 2D mouse (or higher) “sliding space” $\Rightarrow (u, v, \dots)$
- **Optional: velocity.** Velocities $(\dots, \dot{u}, \dot{v}, \dots)$ add a separate “heading variable” that can be very important in golf-cart style control schemes.
- **Note: controller mapping flexibility.** Usually (u, v) would map to spatial $(x, y, z(x, y))$. But projection is *arbitrary*: could use map to camera (azimuth, elevation). 3D wand could use Euler angles, ignore space, etc.

Trick: Variable Controller Sensitivity



A scaling field for controller sensitivity:

- $S(x, y) > 1$ gives rapid motion.
- $S(x, y) < 1$ gives enhanced sensitivity and control.

This gives effects very similar to classic user interface of Mackinlay, Card, and Robertson, but with a more general designer interface.

Generalizing Animation: Viewing Fields

At each point in the controller parameter space, we provide a point in the space of camera viewing parameters:

- **Camera Position.**
- **Camera Orientation.**
- **Alternative: Spaces of Camera Orientations.**
Application: select from a continuous ring of camera models using velocity.
- **Focal Length.** *Any* point can have its own special camera parameters.
- **Viewing Range.** Fog, context-dependent clipping, depth of field, etc.
- **Spotlight.** Point at things not located at view center.

Methods Available to Designer

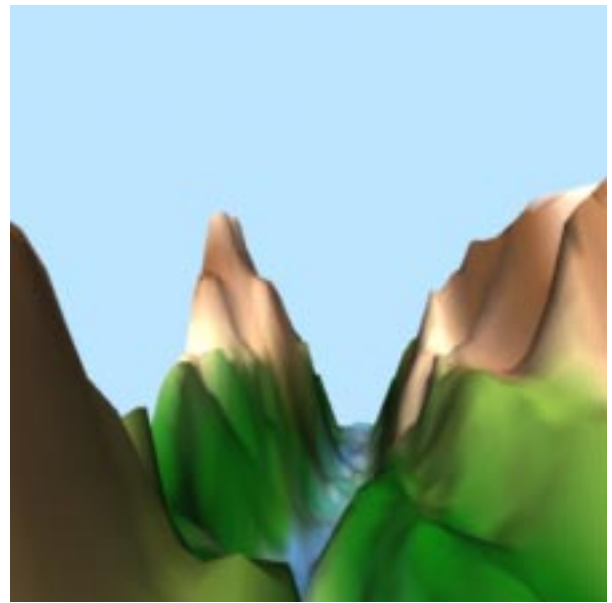
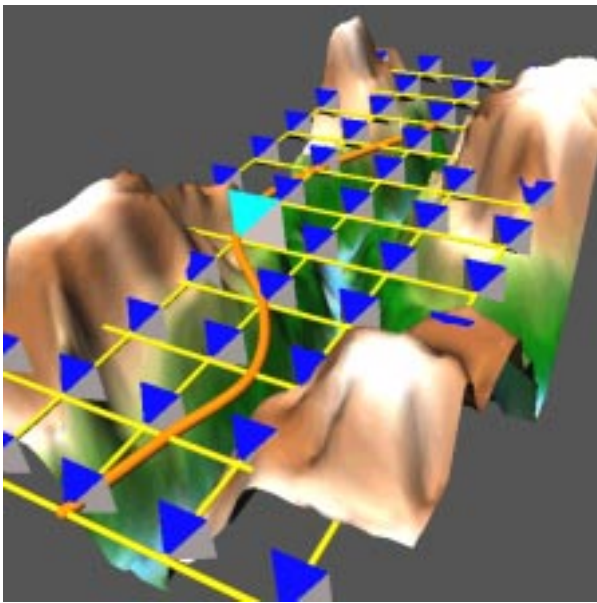
- **Winged Patches.** Arbitrary topological “sidewalk” manifolds, Riemann manifolds, or worse, e.g., soap bubbles, porcupines, are possible.
- **Data Modulation.** Modify default camera by mixing in vectors from data, e.g., terrain gradient.
- **Interest Vectors.** Centers of interest can be defined also as gradients of 3D scalar fields, etc.
- **Fog, Spotlights, Clipping . . .**
- **Vista Points.** Just like roadside tourist stops. Pull up close, and automatically focus on a special object.

Methods for Controller Itself

Bewildering variety of approaches to controller.

- **Fixed fields.** Simplest. Each point on guide manifold provides data for interpolation to a fixed parameter choice.
- **Computed fields.** Harder. Combine a “base” viewing parameter choice with computed modifications, e.g., based on steepness of terrain, “interest field,” or other semantic algorithms.
- **Bare golf cart.** Always look in direction of motion. Can improve by pointing spotlight in direction of interest field.
- **Golf cart, stop and turn.** Align gaze with controller velocity, mix in more of fixed field when slowing, 100% fixed field when stopped.
- **Golf cart with circular view field.** Index into a *ring* of view fields depending on controller velocity direction while moving in golf-cart mode.

Examples of Methods

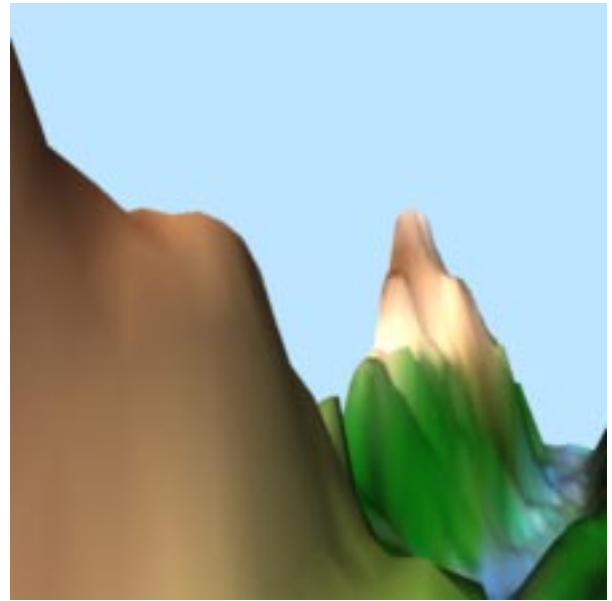
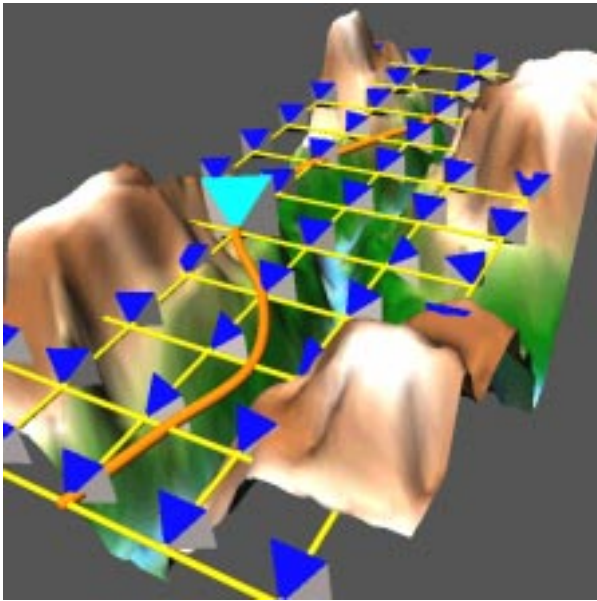


Camera path constrained to plane with fixed camera orientation.

(left) View of path and camera model control points on constraint surface.

(right) View using camera model field at selected point.

Examples of Methods

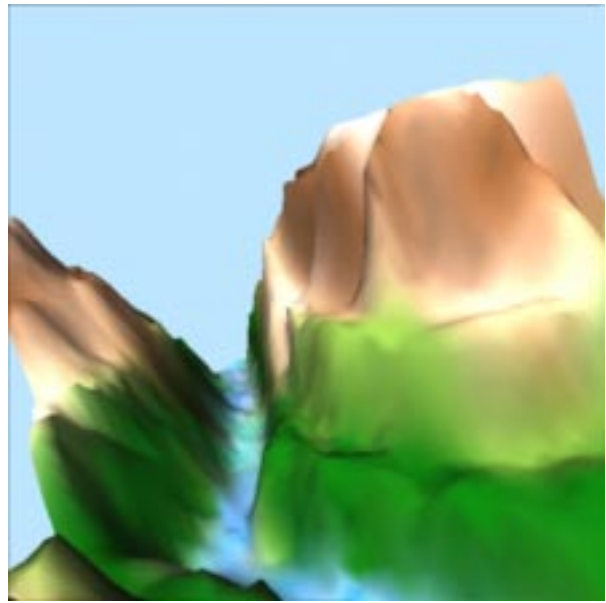
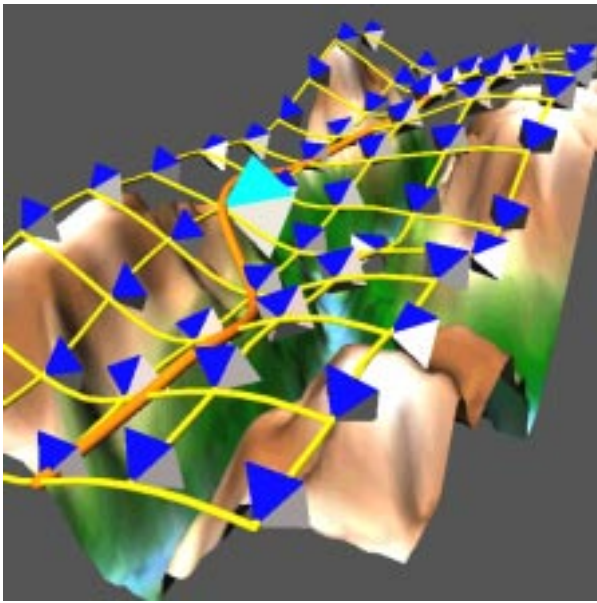


Camera path constrained to plane with camera orientation modulated by terrain gradient.

(left) View of path and camera model control points on constraint surface.

(right) View using camera model field at selected point.

Examples of Methods

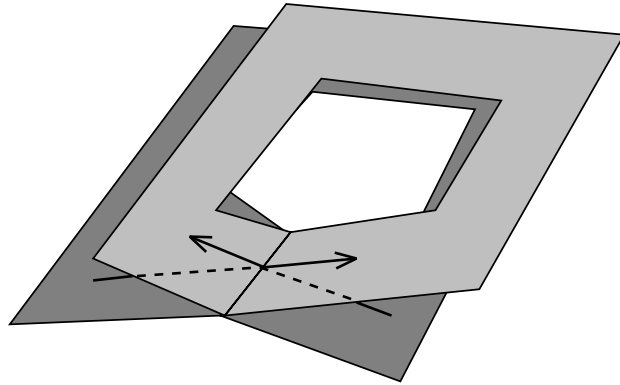


Camera path constrained to conforming surface with default camera orientation matching surface normal, and then modulated by terrain gradient.

(left) View of path and camera model control points on constraint surface.

(right) View using camera model field at selected point.

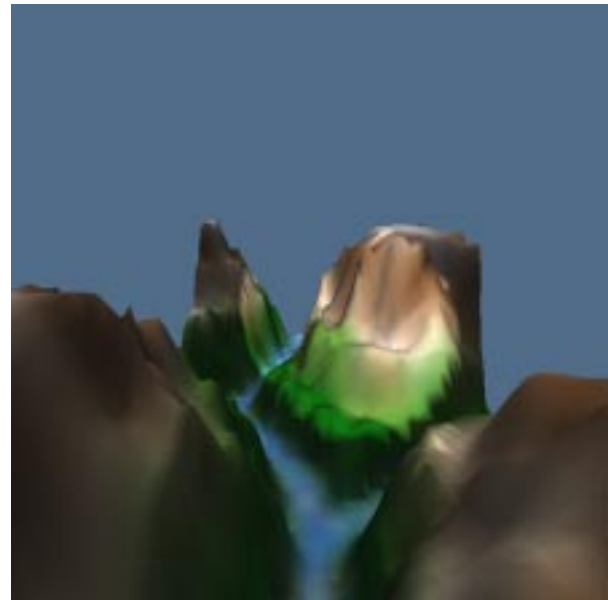
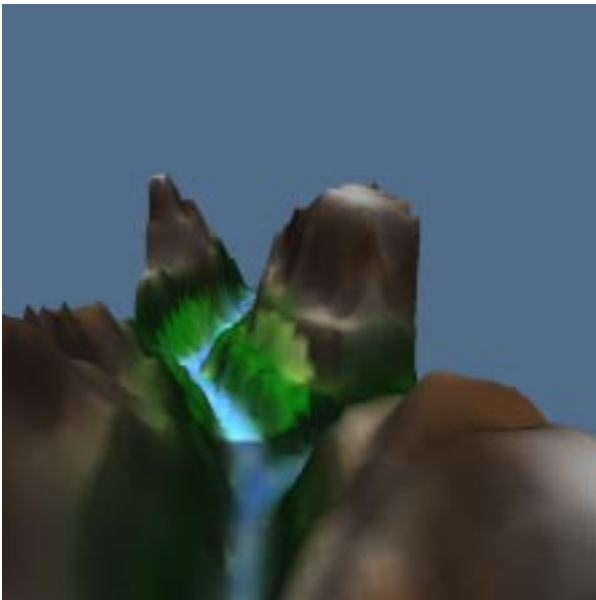
Examples of Methods: Riemann Surfaces



An example of a navigation manifold equivalent to Riemann surface for **square root**.

Idea: Navigation space contains more than one possible layer, hence more than one possible camera model, depending on one's route to the scene.

Examples of Methods: Spotlight



Spotlight focused on an area of interest slightly displaced from gaze and motion directions.

Idea: Better to use spotlight if excess gaze shifting would cause motion sickness.

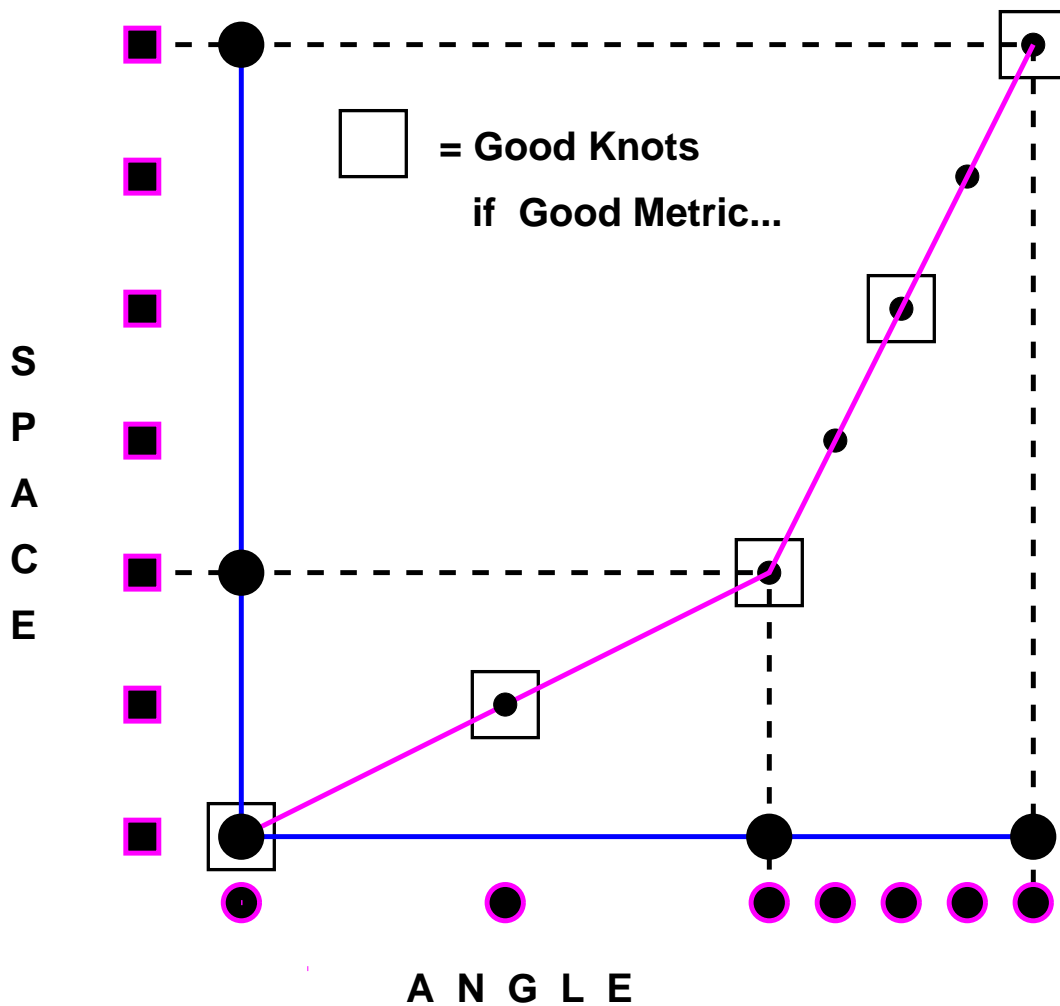
Interpolation Tasks: The Velocity Problem

In practice, designer specifies only key frame parameters at lattice grid points.

- **Normal method:** interpolate variables *separately*: **lerp**'s for Cartesian variables, **slerp**'s for quaternion angular variables.
- **Observation:** Jerky parameter transitions in both space and orientation.
- **No-go Theorem:** You can't fix both using one proper distance, since only *one at a time* is correctly rescaled. (E.g., cylinder.)
- **Interesting Solution:** Concept of *full fiber bundle* rears its head!
Must we interpolate in full bundle space, not separate subspaces?

The Velocity Problem, contd.

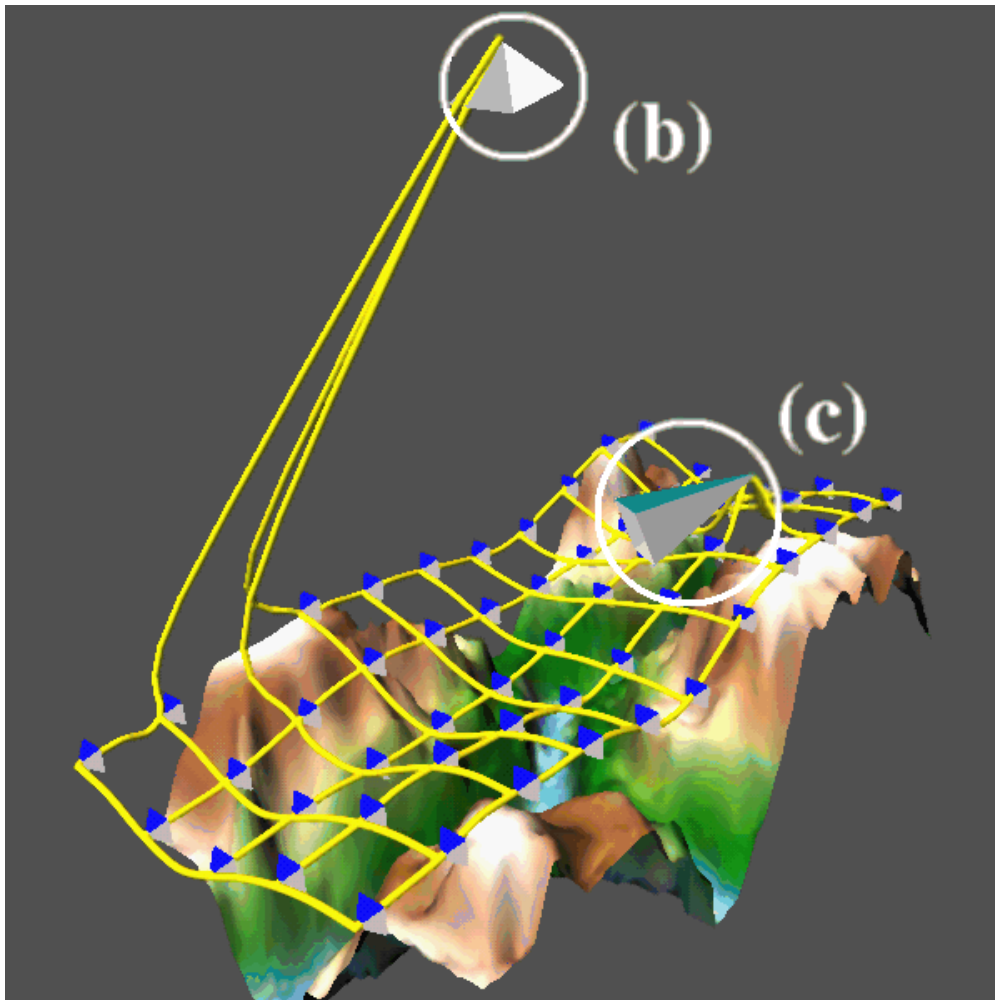
Apparently must adjust parameter normalization using a distance metric in *full bundle*, rather than individual subspaces:



EXAMPLES:

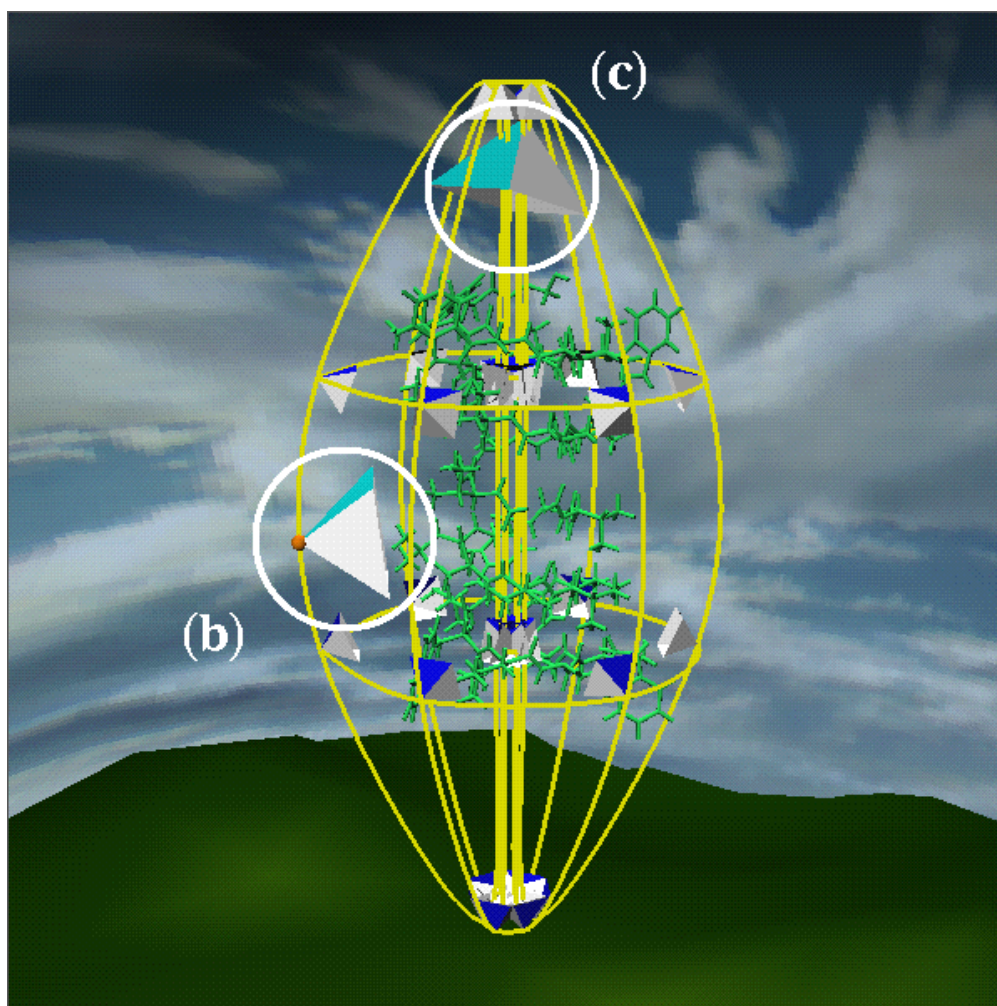
- **Terrain.** Classic application: how to fly an airplane through mountains without worry of crashing!
- **Chemistry.** Where *is* that bonding site, anyway?
- **Architecture.** What did you say you changed?
- **Others:** complex mathematical objects, phase space plots, statistical data in N-dimensions, fluid dynamics, . . .

Terrain



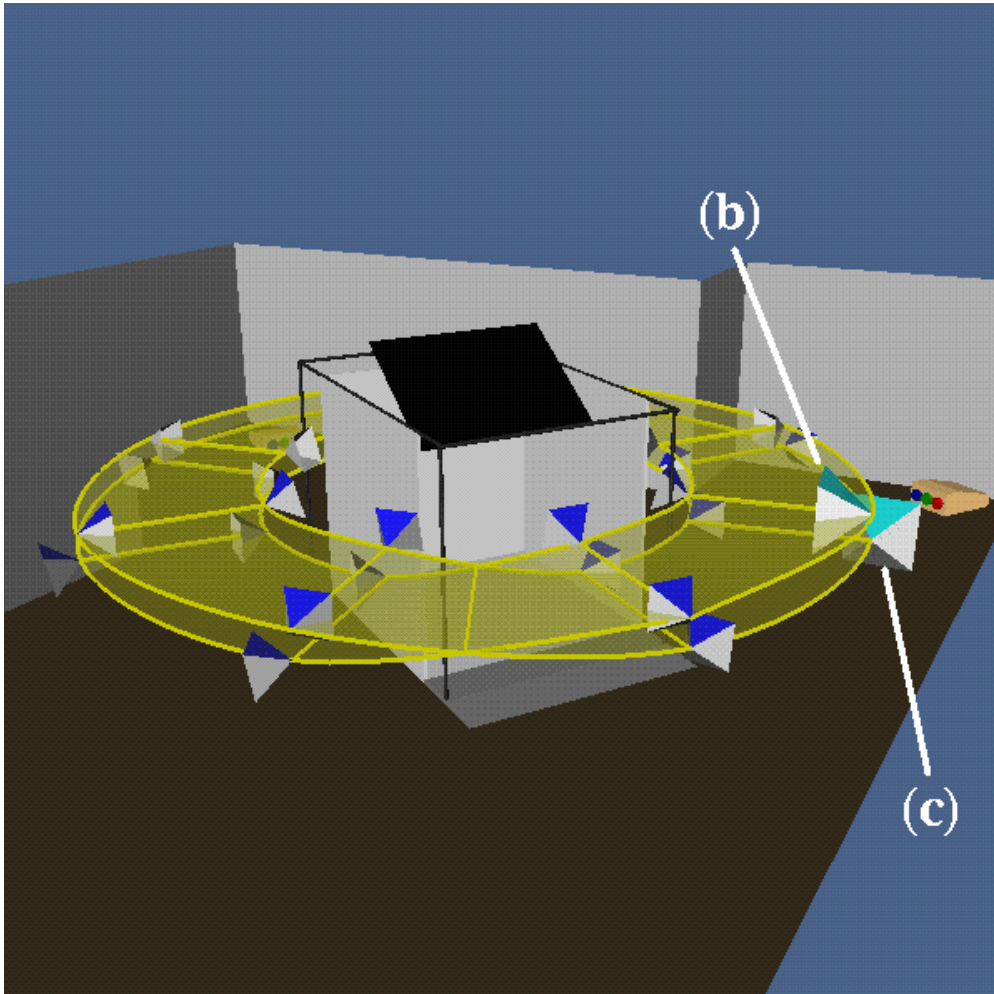
Designer-supplied “vista points” with an “overlook” viewing point at near left and vista point at far right having a telephoto lens pointing back towards base of overlook.

Chemistry



Toroidal constraint surface keeps you looking at the interesting surfaces and the global geometry of the molecule.

Architecture



Multiple valued navigation surface; each layer can have independently chosen camera models.

- **First round:** look at projection screens.
- **Second round:** look at projectors themselves.

Human Interface Results

Master's thesis in progress on human interface properties:

- **Environment.** 3D “apartment” with four rooms containing unique geometric objects.
- **Goal.** Measure effectiveness of object awareness using constrained 2D mouse interface vs free (full 6 DOF Spaceball) interface
- **Results (obvious objects):** Both about 65% acquisition.
- **Results (occlusion):** 55% vs 10% acquisition of occluded objects (singled out by constrained gaze).
- **Results (elevation):** 75% vs 35% acquisition of elevated objects (singled out by constrained gaze).
- *Other Issues:* Very easy to cause motion sickness if parameters not carefully chosen!!

CONCLUSION

- **Problem:** Avoid “Lost in Space” syndrome.
- **Solution:** Constrained Navigation, combining limited freedom of motion with goal-driven selection of viewing parameters.
- **Evaluation:** Initial measurements confirm expected improvements.
- **Future work:** LOTS of open problems:
 - Improve algorithms for knot points in multiple-parameter spaces;
 - Extend 4D algorithms from “Space Walking” framework to arbitrarily high dimensional data;
 - Select “optimal” paths and parameters for both navigation and $ND \Rightarrow 3D$ view projection to get best intuition of ND data sets.
 - Incorporate learned neural net or human expert knowledge.