SMARTPAPER:

An Interactive and User Friendly Sketching System

Presented by Brian Williamson

Written by: Amit Shesh and Baoquan Chen

Presentation Overview

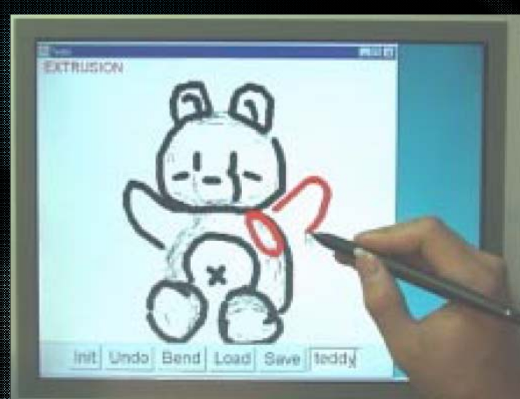
- Introduction to the problem
- 3D rendering techniques using sketch based interfaces
- SMARTPAPER's features
- Processing Pipeline
  - 2D processing
  - 3D Geometry Reconstruction
- Feedback/Cutting/Joining/Rendering
- Conclusion

Introduction

- It is natural to sketch out a 3D object conceptually
- Current methods involve taking a conceptual sketch and using a complex system (such as CAD) to create it
- A better design would be to allow the sketched interface to immediately translate to a 3D object

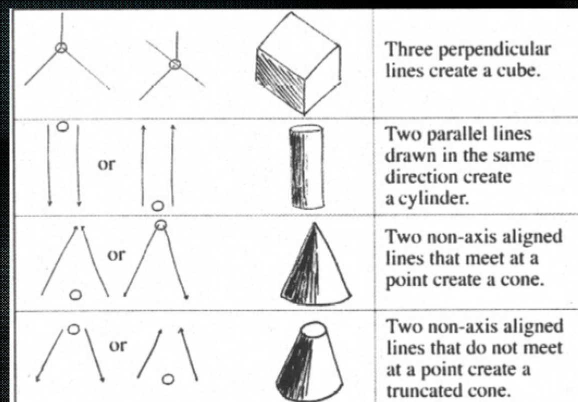
Free Form recognition (TEDDY)

- Draw any design
- Used fewer gestures
- May want simple ways for primitive objects
- Algorithm may create “cartoon looking” models



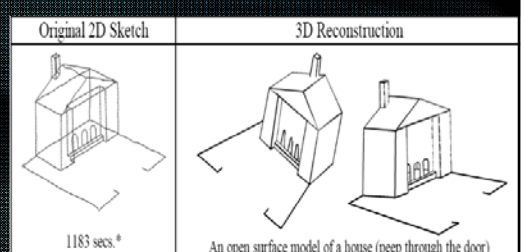
Gesture Rendering (SKETCH)

- Uses several gestures
- Simple to create primitives
- May be unintuitive (depends on gesture set)



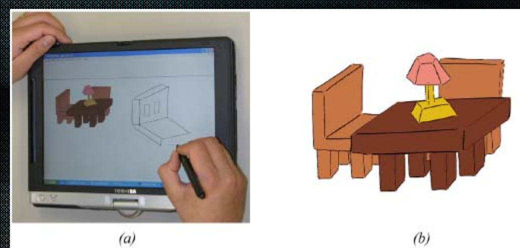
2D Graph Formation

- Proposed by Lipson (1996)
- Scanned in 2D sketches
- Built 2D graph information (vertices, edges)
- Render from graph



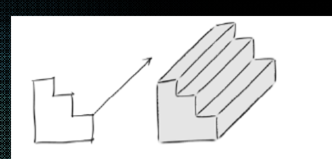
Introducing SMARTPAPER

- A combination of the other techniques
- Allow free form drawing
- Allow gesture manipulation
- Render from 2D graphs

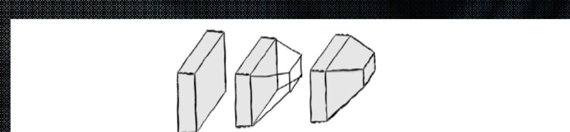


Object Creation

- Can draw an object
- Use an arrow to extrude it out
- Object can be primitive or freeform
- Can perform incremental construction



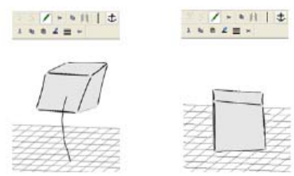
(a) Construction of an object by extrusion.



(e) Incremental construction of a 3D object by sketching directly on an existing 3D object.

Attaching Objects

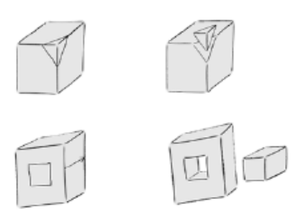
- Can attach an object with a line
- The line determines which face of the object attaches
- Example shows ground plane



(c) Object anchoring by drawing a line between the ground and object. The line also indicates which face of the object touches the ground.

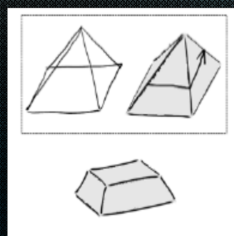
Object Cutting

- Cut an object by drawing into it
- Can draw an extrusion line to completely remove

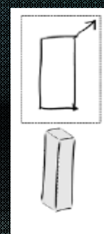


(d) Object cutting by specifying a cutting plane (top) and extrusion (bottom). Figures on the left show sketched input (triangle on top and profile and arrow on the bottom).

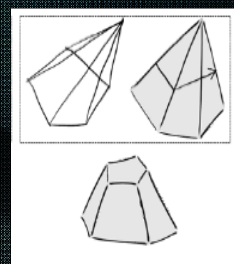
Putting it all together (Lamp example)



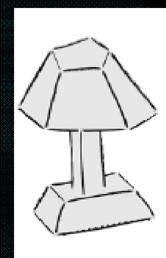
The Base



The Stand

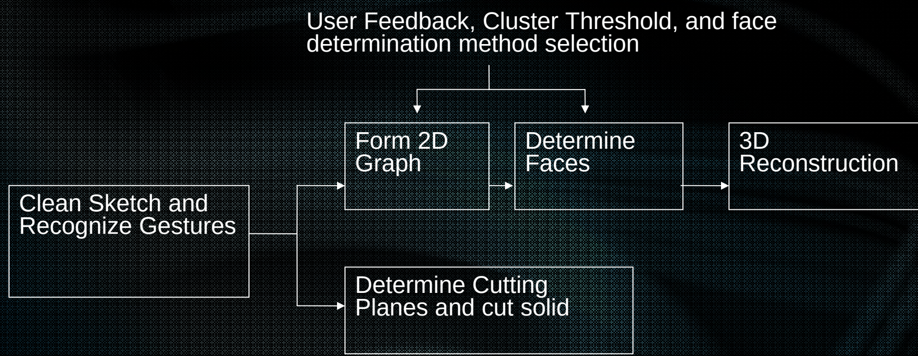


The Shade



The Lamp

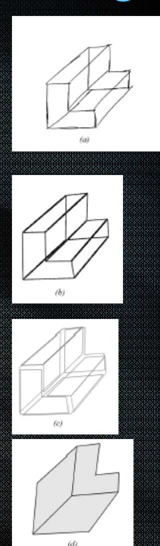
How is this done?



Processing Pipeline

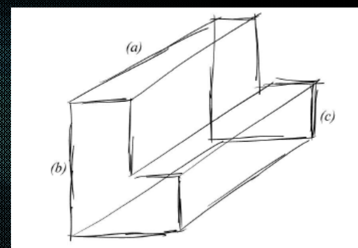
Example Stages

- Input Sketch
  - Rough Drawing
- Cleaned Sketch
  - Remove Overtracing
  - Clean Imperfections
- Recognized Faces
  - Create 2D graph
- Recognized Object
  - Built from 2D graph



Pre-Processing: Over Tracing

- Remove Over Tracing
  - A pair of strokes A,B are over tracing if:
  - They have nearly equal slopes
  - One End Point of A lies in the X and Y ranges of the endpoints of B
- Once found, a connection is made in two passes, A's starting point to B's ending point



Over tracing Example

Pre-Processing: Gesture Recognition

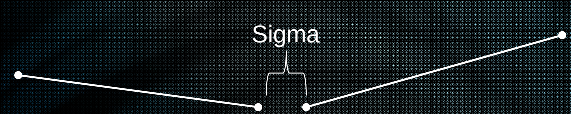
- Arrow was recognized as proof of concept
  - Has to be drawn in two strokes
  - Both recognizer and cutting modules query if this gesture was drawn
  - Either recognizer or cutting module is called depending on the operation

2D Graph Generation

- Graph is generated with a connectivity matrix, each vertex contains (x,y) coordinates
- Imagine taking the endpoints from each line to create vertices
- If performing extrusion, a copy of one graph is made along the direction of the extrusion arrow and edges are connected
- Then use "clustering" to determine proximity and combine graphs
- We assume all sketches are closed objects
  - All vertices must have at least a degree of 3
  - Remove any vertex that does not meet this rule

Clustering Method

- As edges are added to the graph, all end points with a distance of some threshold sigma from an existing vertex are grouped with it.
- Sigma can be changed in the feedback system



3D Reconstruction: Face Determination

- All Faces are Cycles
- Not all cycles are faces though
- Using the closed object assumption
  - All edges of graph G are part of exactly two faces
  - The shortest path of any two vertices V1, V2 is the same length as the path in the face F
  - Proof by contradiction (Available in backup slides)
- Two algorithms are used to determine faces
  - Edge Coherence Algorithm
  - Modified Dijkstra Algorithm

Edge Coherence Algorithm

- 3D objects are not drawn randomly
- Chances are, faces are drawn together
- First Pass: O(e)
  - Can check for cycles in sequential edges for the possibility of a face
  - If two edges do not connect, use a lookahead (1 in the paper) to find a connection
  - With connected edges, attempt to close the temporary face
- Second Pass: O(e1 * n^2)
  - For all unclosed faces and edges not part of two faces
  - Find shortest path of the two vertices
- Works most of the time, but erasing strokes and redrawing strokes will not determine faces

Edge Coherence Pseudo Code

Input: Set of edges in order specified in S

Output: Complete set of faces L and incomplete set of faces V

Pass 1:

//for each edge e in S if the next edge in S is connected to it, add in temporary face F, and if F is closed, add F to L.

//If next edge in S does not have a common vertex with e, look ahead "k" steps for such an edge. If such an edge is found, add in temporary face F and if F is closed, add F to L, else add F to V.

Pass 2:

//Construct sub-graph G', *such that* $V(G') = V(G)$ $E(G') = e : e \in E(G) andeisnotpartof2faces$

//For every edge $e \in E(G')$, *if* $e = (v_1, v_2)$, *then find a shortest* v_1, v_2*-path in* $G' - e$. *This forms a cycle with e in* G'. *If e is now part of two faces, delete e from* G'.

Modified Dijkstra Algorithm

- Take an edge E and find the shortest path from one end point to the other
- Modified in a way that all edges already determined for a face are removed, and cannot be used in the algorithm

Dijkstra Pseudo Code

S: Set of edges that are not part of at least 2 faces R: Set of edges that are part of at least 2 faces M: Matrix having faces as rows and edges as columns.

//Initialize S to all edges that are not part of at least 2 faces. If a current object is being augmented, this set is a proper subset of the edge set of the graph.

while S is not empty do

$e \in S, e = (v_1, v_2)$

//if $e' \in R$ shares an end vertex with e, then mark end vertices of e' as traversed

//Find all edge-disjoint v_1, v_2-paths, mark M accordingly.

//Traverse M column-wise and mark all edges are that are part of more than 2 faces. Then traverse M row-wise to delete all faces consisting of only marked edges.

$G = G - e$ *end while*

3D Reconstruction

- "Inflate" the object by assigning Z values using geometric properties
  - Use planarity of faces, parallelism, etc
- Make use of compliance function
- F(Z) = W * SUM(A)
- Where A is the vector of all factors, W is a weighting function and Z is the z value for all vertices
- This is an N-dimensional optimization problem
  - Used Brent's Minimization technique to reduce to several 1-dimensional optimization problems
  - Go cyclically, Vertex by Vertex, attempting to approach equilibrium
- User can provide hints (dotted lines) to help with reconstruction

More on 3D Reconstruction

- Using hints, create three Z layers
  - One layer consists of vertices to which only hidden edges are incident
  - Second layer is of vertices to which only visible edges are incident
  - Third layer is all other vertices
- These layers give each vertex an initial Z value
- The 3D object is finally represented as a new graph, similar to boundary representation (B-Rep)
  - Boundary Representation is a representation of faces, edges, and vertices. It is currently used in CAD systems.
- Use this graph to reproject when using different viewpoints

Reconstruction Example

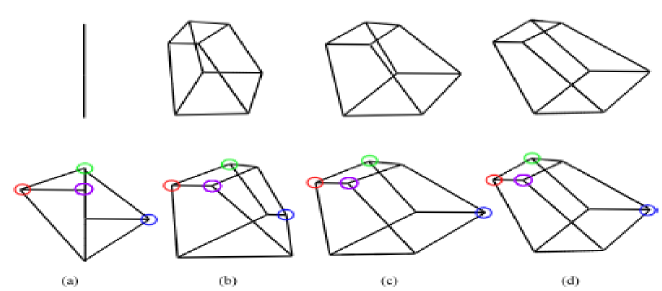
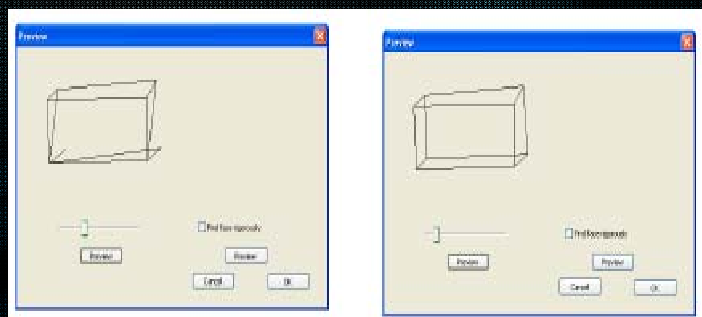


Figure 6: *Inflation of sketch by optimization. Upper and lower rows show how inflation without and with layering respectively: (a) initial condition, (b) and (c) intermediate states, and (d) the final object. The colored circles show how vertices move during inflation.*

Offering User Feedback

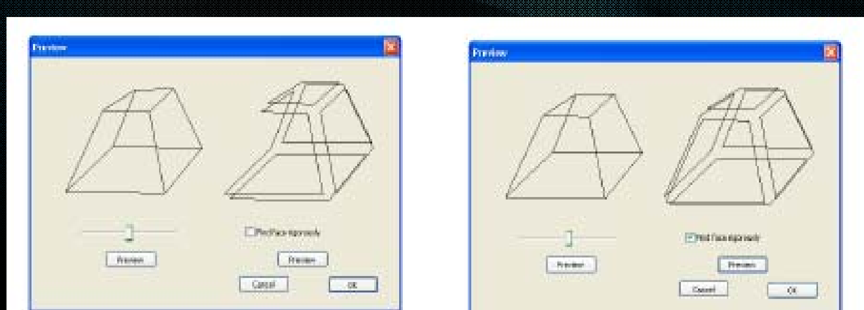
- Clustering may be incorrect
- Leads to incorrect Faces



(a) (b)

More User Feedback

- Can switch face detection algorithm



(c) (d)

Cutting

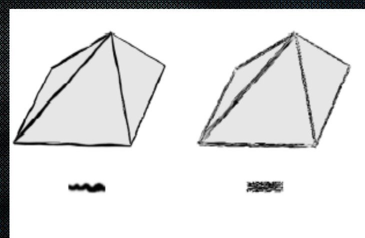
- Performed by drawing on a 3D object
- Convert strokes to 2D graph
- Cast ray from eye position through both end vertices of each edge
- Arrow uses ray casting to determine direction
- Cut from the object once the plane has been generated
- Cutting algorithm discussed in “Boolean Operations of 2-Manifolds through vertex neighborhood classification”
  - Uses Boolean operations to cut from 3D B-Rep graph representation

Joining

- Uses the line to attach two objects
- Perform coordinate transformation to cause the two objects to “stick”
- Any transformation done to one object is automatically performed with the other

Rendering

- Non-photo realistic techniques
- Two pass algorithm
  - First pass, render all faces
  - Render edges as textured quads



User Study

- SMARTPAPER shown to ten students and one faculty member in the Architecture Department
- Appreciated the ease of use and functionality over designing in CAD
- No real data presented

Conclusions

- Seems useful, though not compared with other sketching systems only CAD
- Cannot evaluate curved 3D objects
- Could make use of more gestures
- Thresholds and constraints often used
- Focuses on architectural design techniques

Questions?

BACKUP SLIDES

Proof of Definition A

Definition A: All edges of graph G are part of exactly two faces. Every valid face F of G is such that for all pairs $v_1, v_2 \in V(G)$ that are in F, the shortest v_1, v_2-path in G is of the same length as the v_1, v_2-path in F.

Justification: The first statement is a property of closed, non-laminar, rigid objects. If the second statement is not true, let P be the shortest v_1, v_2-path in G and let P' be the shorter v_1, v_2-path in F. There is at least one edge in P not in P'. PP' thus creates a smaller closed walk and hence a smaller cycle C containing v_1 and v_2. The edge set $E(C) - (E(C) \cap E(F))$ divides face F into two or more different planes, which is a contradiction as F is a valid face and hence is planar.

Modified Dijkstra Example