13º Encontro Português de Computação Gráfica

12 - 14 Outubro 2005

A Graph-Matching Approach to Sketch-Based Modelling

Filipe Marques Dias Joaquim A. Jorge
Department of Information Systems and Computer Science
INESC-ID/IST/Technical University of Lisbon
R. Alves Redol, 9, 1000-029 Lisboa, Portugal
fil@immi.inesc-id.pt, jorgej@acm.org

Abstract

Automobile designers, when creatively designing a vehicle, have to bind by several shape constraints while also exploring new concepts. A natural and unhindered modelling environment for a designer to start modelling in is thus desired. We present our ongoing modelling system, AutoMake, that enables calligraphic modelling of a simple 3D object that resembles the body shape of a car. Our system provides an innovative way to recognize 3D objects by matching sketches against templates stored in a database, indexed by multiple edge graphs based on various views. Our approach enables users to specify a bounding box providing perspective and size information about model. The user next sketches a model that is matched against a template, using a multidimensional indexing structure.

Keywords

Sketch-based Modelling, Graph Matching.

1. INTRODUCTION

Modelling three-dimensional objects from sketch input has gained importance in the modelling field recently. This fact was motivated by the lack of user-centred concerns in modelling systems, preventing users from pouring their creative power directly upon 3D models without fiddling with bureaucratic modelling interaction concerns. This hampering of creative use forces the separation of tasks. In the industry, this translates into modelling experts working together with designers, causing a need for comprehensive and delaying communication due to misinterpretation and other physical impossibility issues [Dias03]. In order to solve this, the designer needs to build an early stage 3D model that can afterwards be refined and worked from by the modeller. Furthermore, in those early stages of product design, designers sketch creative drawings to explore ideas and produce rough sketches of objects that are usually bound to some general constrained specifications. This led us to the problem of providing a natural and unhindered modelling environment for a designer to start modelling in. We propose a calligraphic approach, based on 3DSketch [Mitani00, Mitani02, Varlet04], where the user sketches a visible-edge representation of one of several possible template models, and is able to manipulate, and edit it by re-sketching over edges as if correcting their curvature. Our approach allows the designer to draw with stylus and tablet or directly on screen, with a digitizing monitor, in a similar fashion as sketching on paper. Our approach applies fewer restrictions to models since unknown information is assumed to be specified by the template stored on the database.

The following section takes a look at other approaches in this area and points out their difficulties in completely solving this problem. The next section describes our approach, explaining how our approach sees the template drawing as a graph, and retrieves it from a data structure. The remaining section refers final considerations, ongoing work aspects, and directions to take for future development.

2. RELATED WORK

The topic of modelling 3D objects interactively from 2D calligraphic input is becoming a more common in the modelling context.

In 1994, IdeS [Branco94] appeared with a modelling scheme based on reconstruction and gesture recognition, for use in a mechanical engineering context. The user was able to draw the solids' visible edges and issue some modelling commands by means of gestures, like extrusions, cuts, and other operations involving more objects. Later, this work saw vast improvements, when context-based modelling aid was introduced in its successor, GIdeS [Pereira00]. GIdeS improved on the previous work in many key areas, the main of which: the user interface, and the addition of context-based functionality to cope with the ambiguity natural to reconstruction. The latter is achieved by an ingenious expectation list that presents several previews of the final result depending on the applicability of operations. The modelling approach present in that system reflects the concerns with user interaction, corresponding to a constructive way of modelling.

SKETCH [Zelesnik96] introduced another gesture-based interface for approximate 3D modelling, with some added

support for animation. With this system, the user follows a few rules to draw specific features of supported primitives. The system then instantiates them in 3D space appropriately. Its modelling methodology is slightly different from the previous GIdeS work, as the model is constructed from separate objects (parts) that are grouped together automatically.

Chateau [Igarashi01] is a system that enables the definition of polygonal shapes and objects, and suggests results of candidate operations based on hints – a similar approach to the aforementioned expectation lists. Chateau highlights entities drawn, to indicate the hints that will be used for suggesting operations. It is up to the user to toggle the highlight status of the entities, to appropriately tailor the scope of possible operations to his desire. Suggestions are generated each time entities change highlight status.

RIBALD [Varley00, Varley04] is a system that is able to produce a boundary representation model from an imperfect line sketch of a manifold polyhedral object. Its approach is based on interpretation, to create solid objects. It starts by labelling lines (concave, convex, or occluding) and junctions; later produces visible geometry, assigning provisory depth values to vertices; and identifies local symmetry features. Additionally, the system tries to classify objects to attempt at speeding the reconstruction using specific case methods. Objects produced are a variety of solid models that can be made from non-perfect line input.

3DSketch [Mitani00, Mitani02, Varley04] reconstructs and renders particular models based on sketch input using non-photorealistic rendering techniques maintaining a sketchy appearance. It works by generating an edge graph from a set of input sketches, reconstructing a model from that graph and some assumptions, and finally rendering it. The system is focused on reconstructing only objects that have 3 perpendicular faces visible, have a flat based and back faces, and possess mirror symmetry. Some of these constraints are necessary for estimating camera parameters due to drawing in perspective. This system also enables de definition of curved edges and surfaces.

These approaches tackle the problem of constructing relatively simple 3D geometry from 2D sketch input, either using a set of commands the user must issue by means of a gesture, or by interpreting the model from a starting set of constraints or existing template. The gesture approach has the drawback of forcing the user to know the commands to issue, and often these do not correspond to the most obvious pen movements. This way of creating faces and surfaces is not appropriate for sketching creative drawings, as interruption is undesired. On one hand, interpretation provides a higher sketched fidelity for visually representing objects. On the other hand, templates, allow restricting the domain of objects to a controllable scope, simplifying reconstruction, and enabling the addition of curvature to an otherwise difficult to reconstruct drawing. However, in Mitani's 3DSketch approach, this scope is limited to a single template.

Our system is based on 3DSketch, in a way that the starting point for modelling is well defined by a template, but allows the existence of a vast number of different templates. Each templates is stored in a database using edge graphs produced from several views of it, so it can be retrieved when a users draws it.

In a context of image-based similarity of models, [Chen03] calculate several views of objects in a database. They use these views to do image-based indexing using Zernike moments and retrieve similar objects. That approach is based in that if 3D models are similar, then they have similar views. They use the vertices' positions of a dodecahedron around the object for placing a camera pointing at it, and render the various views. In our approach we also position a "camera" on a dodecahedron's vertices, but with the difference: we build an edge graph instead of rendering an image. A single template model is stored in an NBTree data structure [Fonseca04] using those graphs as indexes.

3. OUR APPROACH

Our system is based on work by [Mitani00, Mitani02, Varley04] for matching perspective, and the model to graph. However, that approach takes into account a single fixed object template. This drastically limits the usefulness of a modelling application to cope only with an object of a single topology. Among car body shapes, the number of visible vertices on a template can vary depending on the type of vehicle, and this led us to consider more templates in our approach. Our system also has an increased simplicity dealing with more complex templates.

3.1 Overview

A template is a simplified linear-edged model that not only represents the connectivity of vertices on the desired object, but also specifies its rough desired shape, dimensions and some other constraints. A template can be seen as a simplified model, with its key characteristic features specified. Figure 2 shows a template, on the left.

Using our system, the user starts by sketching, in perspective, a bounding box of the object he wishes to draw. This simple step has many uses, as mentioned in the next subsection.

The user then draws the object he intends to represent, by sketching its visible edges. The system sees these edges as edges on a 2D graph, and builds that representation internally. This graph corresponds to the 3D object as viewed from the viewpoint it was drawn in.

This graph is then used for building a numerical vector to query a database, through its adjacency matrix representation [Fonseca04], as described next. This query will be executed on a database to retrieve a set of templates that most resemble the drawn graph. Then the right one is chosen and the user can take part in that choice.

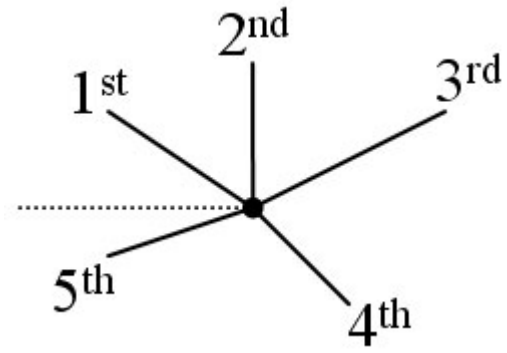

Figure 1 - Order of a node's edges

From this point onwards the user should be able to edit and refine the model interactively.

3.1.1 Bounding box

The simple step of drawing a bounding box has many uses. Its key function is based on the designers' own methodology of drawing, making the user confident and comfortable. Not only will the user be specifying a self guiding method for more accurately express his intent, he will also be less prone to fall into disproportion errors, non-parallel lines, and other unwanted distortions.

As a second advantage, the user will also be revealing certain aspects of the object that the system may use, in a posterior phase, to help cope with ambiguity. Such clues are perspective information and rough proportions. To determine these clues, the system calculates the intersections of line segments for bounding box's edges and their lengths. For perspective information, the linear extensions of the bounding box's edges are used. The intersection of these enables estimation of vanishing points. With these points determined, camera parameters are calculated. The estimation of vanishing points and camera parameters is similar to that described in [Varley00].

A third use is extracted from this bounding box, as well, which is the expected size of the graph, useful for knowing when the drawing is nearing completion. At a later phase, the system must build the graph representation and issue a query with it to a database, to retrieve the template for posterior modelling. The system must either be explicitly issued by the user to do this, or it must estimate the correct time for it. In order to estimate it the system looks for a closed graph, without degenerate edges, that roughly spans the extension visible faces of the bounding box.

3.1.2 Graph Representation

As mentioned, the sketch drawn by the user is stored as an edge graph representing the 2D projection of the 3D object for the given view. Each graph node corresponds to a visible vertex in the object, and each edge to an edge of an object's visible face. The list of edges a graph node has is ordered in a way that edges appear in a clockwise order, as shown in Figure 1.

The purpose of ordering edges is twofold: to make sure the graph is traversed simply in a coherent manner, and to make edges belonging to a same face adjacent in the representation. Consequently, the graph needs only a preprocessing phase to order the edges, enabling all subsequent traversals to be made simpler and faster. However, this also means the graph may require partial re-ordering if the view is changed substantially, in such a way that new edges become visible or invisible, or when a new starting node needs to be calculated for the graph.

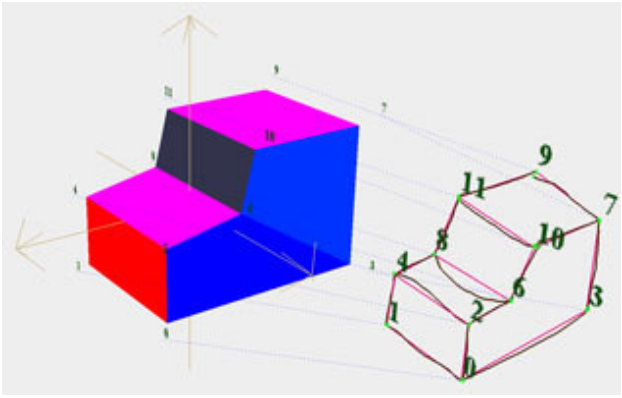

Figure 2 - Vertex numbering and correct match between object graph and template

3.1.3 Template Indexation and Retrieval

In order to store and retrieve template objects, a numerical vector needs to be computed. These vectors index template models, stored along with camera position information, in a multidimensional indexing structure (NBTree [Fonseca04]). A vector is composed of the eigenvalues for the adjacency matrix representation of the drawn edge graph, and the retrieval of such vectors, is a stable problem [Fonseca04]. This means that small variations in the graph used for searching will retrieve results in a vicinity of the original graph.

After a retrieving a set of templates, the system has to choose the best one among them. Our approach allows something similar to an expectation list, like GIdeS [Pereira00] or Chateau [Igarashi01], to be used, enabling the user to take part in more specifically opt for the desired template. Nonetheless, rating possibilities should be made possible using proportion measures based on the drawn edge graph, and perspective hints from the bounding box as well. Though, it is possible that the graph drawn does not correspond to a template stored. In such a case the template returned is the one that is most closely matched. At present, our system assumes that only one best fitting template is returned by the query made to the indexing structure, and that it topologically resembles the drawn graph.

3.1.4 Graph Matching

With a candidate graph selected, the system traverses it, and the graph is built from the user's drawn input, in the same order establishing correspondences between vertices, edges, and faces.

To start traversal, a leading vertex must be chosen for the graph's starting node. Because we assume the object is drawn from an "informative" point of view, and that 3

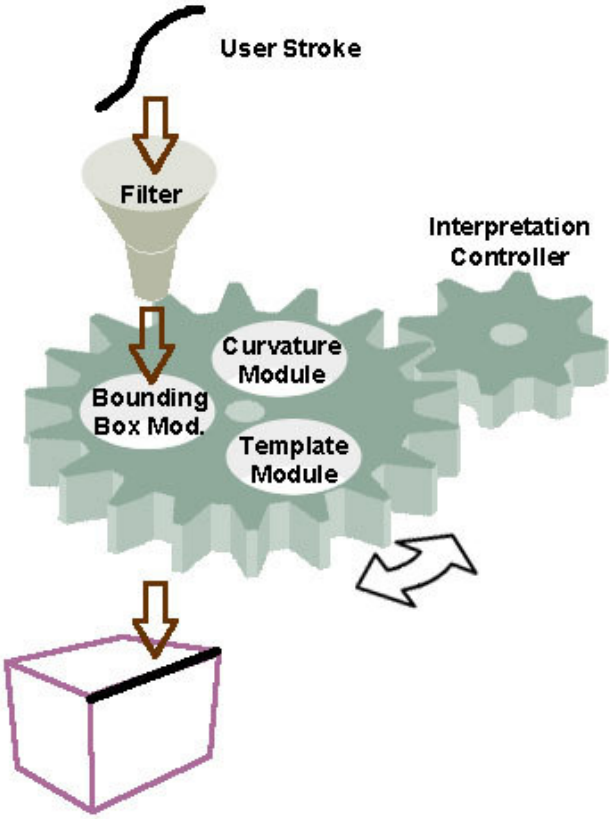

Figure 3 - Modular system architecture for interpreting a stroke depending on modelling context

faces of the bounding box are visible, we simply chose the first vertex to be the leftmost lower vertex (vertex 0 in Figure 2). The system pays more attention to non-dihedral vertices, as these vertices only have two edges. This gives higher flexibility to deal with graphs of slightly different topology, weakening the constraint on both graphs having the same starting node. This means the starting node must be on the same face, instead.

Having picked the starting node, the matching traversal progresses normally, comparing the number of edges for each non-dihedral node and each face. In the end the dihedral nodes are matched. If that match is not possible, the vertex information on the template prevails, if both graphs are successfully matched.

We do not reconstruct an object in the same way 3DSketch or RIBALD do. Instead we use the information stored in the database for specifying the edges. In the future our system will allow the user to change them on the next modelling phase.

3.2 Sketched entities

RIBALD [Varley00, Varley04] assumes the object is drawn from "most informative viewpoint". In our approach, this assumption is replaced by a less restrictive one: a viewpoint that produces an edge graph of the same topology. This is because the user is able to change viewing position after drawing the bounding box. Though, like them, we also assume no through holes are present, and that edges form a single graph. Like [Mitani00] for determining perspective, we assume the bounding box is drawn with 3 perpendicular faces visible. 3DSketch must assume the back face is planar because no invisible vertex or edge is represented. However, with our approach that information comes from the retrieved template object. The user can also navigate around the object and refine it.

Regarding curvature, our system does not currently support it. However, we are planning to follow the same principles described in [Varley04]. As mentioned there, the information needed to define curvature from a single view involves specification of symmetry or other constraints. That information is accounted for inside our template. This would mean curve edges would be estimated at the time of matching, similarly to [Mitani00]. In such cases when curvature estimation is not possible due to lack of information (a symmetrical edge is not visible) our approach will revert to the default edge information saved on the template.

3.3 System Architecture

Our system is organized in modules that wait their turn before acting upon a stroke drawn by the user, depending on context. Figure 3 illustrates this organization.

When the user draws a stroke, it goes through a basic filtering (re-sampling) process to reduce the high number of points. Drawn strokes are then stored in memory to allow posterior access to them, for changes to be made.

Next the filtered stroke is passed to an "interpretation" module assigned by a controller based on context. The module that processes the first strokes is the one responsible for creating the bounding box. This module interprets strokes as linear edges forming a box. This module also estimates de camera parameters mentioned earlier.

When the bounding box is completed and perspective information has been determined, the controller switches the active stroke interpretation module to the Template Module, responsible for building the graph.

After a template has been retrieved from the database, the system can then switch interpretation control over to another module that can interpret strokes differently; for instance, a module for changing curvature of edges.

4. CONCLUSIONS AND FUTURE WORK

In this paper we presented our ongoing calligraphic object modelling approach that matches a drawn graph against a template stored on a database. The user first draws a bounding box for the model, and then sketched the object inside it. Matching these allows the generation of a 3D model that can later be refined.

We showed that the use of the information extracted from the bounding box at the beginning of modelling can be useful for the modelling approach, as it is to the user drawing it and the modelled result. The modelled object is no longer confined to being of a single "boxy" nature, and the user sketches his sketch using his learned drawing techniques.

Regarding the ability to deal with several templates – provided by the use of an indexing structure – we mentioned the stability of retrieving vectors computed from eigenvalues of adjacency matrixes. This encourages us to continue pursuing our approach, improving its modelling capabilities.

We should also find rating solutions for selecting graph similarity based on edge length, dimensions and other features in order to better sort the possibilities returned by a query to the indexing structure.

Our approach allows the user to interact with the model between estimation of camera parameters and definition of the object. This allows us to cope with "not-so-informative" points of view, which RIBALD and 3DSketch are unable to model in. This is an advantage. Having the number of possibilities stored in the database as the limiting factor, our approach has the potential for being more robust. However, this still requires substantial practical testing to confirm.

Another topic to research is the possibility of users specifying invisible edges by using pressure or a lower number of overlapping strokes. Extra information provided by hidden-curve specification may override the default behaviour of depending on the template, and allows for more freedom in choosing a point of view to draw the model – as some invisible faces may be specified.

The curvature in edges is ignored in the current implementation of our system, but it can be added in a similar fashion as in 3DSketch.

When the database is filled with more templates, we believe that we will achieve our goal of allowing a designer to explore his creativity unhindered, using the calligraphic techniques natural to his skills.

5. REFERENCES

[Branco94] Branco, V; Ferreira, F N; Costa, A: "Towards an Intuitive 3D Interaction", *Third Luso-German Computer Graphics Meeting (Conference Proceedings)*, Coimbra, Portugal, 1994.

[Chen03] Chen, Ding; Tian, Xiao; Shen, Yu; Ouhyoung, Ming; "On Visual Similarity Based 3D Model Retrieval", In *Proc. Eurographics*, 2003.

[Dias03] Dias, Filipe; Jorge, Joaquim A. "Task Analysis and Scenario-Based Design of Calligraphic Interfaces", *12º EPCG*, Porto, Portugal, Oct 2003.

[Fonseca04] Manuel J. Fonseca. "Sketch-Based Retrieval in Large Sets of Drawings". *PhD thesis*, IST/UTL, Lisboa, Portugal, July 2004.

[Igarashi01] Takeo Igarashi and John F. Hughes, "A suggestive interface for 3D drawing", *UIST*, 173-181, 2001.

[Mitani00] Jun Mitani, Hiromasa Suzuki, Fumihiko Kimura, "3D Sketch: Sketch based Model Reconstruction and Rendering", *IFIP Workshop Series on Geometric Modeling: Fundamentals and Applications, IFIP Working Group 5.2, Seventh Workshop GEO-7*, University of Parma, Parma, Italy, October 2-4, pp.85-112 (2000).

[Mitani02] Jun Mitani , Hiromasa Suzuki , Fumihiko Kimura, "3D sketch: sketch-based model reconstruction and rendering", *From geometric modeling to shape modeling*, Kluwer Academic Publishers, Norwell, MA, 2002.

[Pereira00] Pereira, J. P.; Jorge, J. A.; Branco, V. A.; Ferreira, F. N.; "Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons", *WSCG - International Conference in Central Europe on Computer Graphics*, *Visualization and Computer Vision*, Feb. 2000.

[Varley00] P. A. C. Varley, H. Suzuki, J. Mitani, R.R. Martin: "Interpretation of Single Sketch Input for Mesh and Solid Models", *International Journal of Shape Modelling* 6(2), 207-241, 2000.

[Varley04] P. A. C. Varley, Y. Takahashi, J. Mitani, H. Suzuki: "A Two-Stage Approach for Interpreting Line Drawings of Curved Objects", *EUROGRAPHICS Workshop on Stekch-Based Interfaces and Modelling 2004*, Grenoble, France, 2004.

[Zelesnik96] Zeleznik, Robert C., Kenneth P. Herndon and John F. Hughes.; "Sketch: An interface for sketching 3d scenes," *Computer Graphics, SIGGRAPH 96 Proceedings*, August 1996, pp. 163-170.