Project description

Contents

[External libraries 3](#_Toc306985570)

[Installed 3](#_Toc306985571)

[Part of the project 3](#_Toc306985572)

[System overview 3](#_Toc306985573)

[Input and preprocessing 4](#_Toc306985574)

[Input 4](#_Toc306985575)

[Distance transform 5](#_Toc306985576)

[Categorization 5](#_Toc306985577)

[Interactive modeling 5](#_Toc306985578)

[Overview 5](#_Toc306985579)

[Primitive categories 6](#_Toc306985580)

[New primitives 6](#_Toc306985581)

[Snapped primitives 6](#_Toc306985582)

[Matching 6](#_Toc306985583)

[Snapping 7](#_Toc306985584)

[The target function 7](#_Toc306985585)

[The constraints 7](#_Toc306985586)

[Optimization algorithm 7](#_Toc306985587)

[Primitive types 8](#_Toc306985588)

[Cylinder 8](#_Toc306985589)

[Cone 11](#_Toc306985590)

[Sphere 11](#_Toc306985591)

[Straight generalized cylinder 11](#_Toc306985592)

[Building blocks 12](#_Toc306985593)

[Cases for objective function construction 14](#_Toc306985594)

[Annotations 15](#_Toc306985595)

[New primitives **Error! Bookmark not defined.**](#_Toc306985596)

[Snapped primitives **Error! Bookmark not defined.**](#_Toc306985597)

[References 17](#_Toc306985598)

# External libraries

The project consists of two kinds of external libraries – those that need to be installed, and those that are part of the project. The code is written in the C# language for the .NET platform.

## Installed

* Prism library. The library allows easier development of UI applications using WPF. <http://compositewpf.codeplex.com/releases/view/55576>
* Reactive extensions for .NET:  
  <http://www.microsoft.com/download/en/details.aspx?id=26649>
* WPF 3D Tools:  
  <http://3dtools.codeplex.com/releases/view/2058>

## Part of the project

* AutoDiff library for .NET. Used for numeric optimization. The library uses reverse-mode AD algorithm to quickly compute gradients of functions. [1, pp. 31-37].  
  <http://autodiff.codeplex.com/>
* AlgLib – a library for numeric computing in C#. Used for its implementation of L-BFGS algorithm [2].  
  <http://www.alglib.org>

# System overview

The system operates in two stages – a pre-processing stage and an interactive modeling stage. The system’s input is a sketch given as a set of polylines. In the pre-processing stage products are produced:

* A distance transform image for every polyline. The polylines are sampled on a 512x512 grid representing the sketch. Then a discrete distance transform function is computed on this grid.
* Categorization of the polylines to feature and silhouette curves. This stage is manual and the user performs it.

The following illustration describes this process:

Polyline 1

…

Polyline n

Sketch

Distance transforms

DT(Polyline 1)

…

DT(Polyline n)

Feature/silhouette

|  |  |
| --- | --- |
| Polyline 1 | Feature |
| Polyline 2 | Silhouette |
| … | … |
| Polyline n | Feature |

Next, an interactive modeling process occurs. The system uses the polylines, the distance transforms and the categorization for the interactive modeling process.

During the modeling process the user can perform the following operations:

* Add and position a new primitive over the sketch
* Snap the primitive to the sketch
* Annotate the already-snapped primitives with 3D information for correct 3D alignment.
* Duplicate an already-snapped primitive. The duplicate behaves like a new primitive, which can be edited and positioned before snapping.

# Input and preprocessing

## Input

The system’s input is a collection of polylines in an SVG file. If the SVG file contains path objects, the paths are sampled and converted to polylines. The sketch is normalized 2D coordinates in the rectangle . We define to be the sketch – a set of polylines. We assume that their vertices are normalized to the above coordinate system.

The sketch is assumed to be orthographically projected. The system does not deal with perspective and will assume that smaller objects on the sketch represent actually smaller objects (and not further-away objects).

## Distance transform

A grid of 512x512 pixels is defined over the rectangle . Every polyline is sampled over this grid to produce a binary image. Then, a distance transform of this binary image is computed using chamfer distance transform algorithm [3].

As a result of this process, we have distance transform images for the polylines. The pixels in each image define an approximate distance to the closest point on the matching polyline.

## Categorization

The user is asked to categorize the polylines into two sets – feature lines and silhouette lines. Feature lines represent sharp feature of the sketched 3D model. Silhouette lines represent the boundary of the object’s silhouette defined by the viewing direction.

The following example illustrates this categorization. The black lines are feature lines. The gray lines are silhouette lines.

# Interactive modeling

## Overview

The modeling process consists of the following stages:

* The user adds and edits a primitive over the sketch. At this stage the user can freely modify all the parameters of the primitive.
* During the primitive modification the system automatically computes a matching between the 3D curves of the primitive to the 2D curves of the sketch. This matching is displayed color-coded to the user (a curve on the primitive and a matching curve on the sketch have the same color).
* If the matching doesn’t satisfy the user – he can change this matching manually using the mouse.
* When the user is satisfied with the computed matching, he presses the “snap” button. The system uses the matching to deform the primitive’s shape to match the sketch.
* The user can select curves of the snapped primitives and annotate them with constraints (such as “parallel” or “collinear”).

## Primitive categories

The system has two categories of primitives – new and snapped.

### New primitives

New primitives are the kind of primitives the user can edit and manipulate over the sketch. Each new primitive has a parametric representation and a set of feature and silhouette curves. New primitives do not snap to the sketch. However, every edit operation results in automatic re-computation of the matching between the curves of the primitive and the curves of the sketch. See the “Matching” section for more details.

### Snapped primitives

Snapped primitives are kind of primitives that are optimized to match a part of the sketch. Such primitives cannot be manipulated directly by the user. Snapped primitives have a parametric representation using variables from the AutoDiff library. This parametric representation corresponds to the parametric representation of the corresponding new primitive. For example, a new cylinder might have a radius. A snapped cylinder will have a variable object that represents this radius.

In addition, snapped primitives also have feature curves that are represented using the object’s parameters. That is, a feature curve is represented using symbolic expressions that involve variables in the AutoDiff library. Those feature curves are also displayed to the user, and the user can annotate them with constraints. Those constraints serve as hard constraints in the optimization process that fits the snapped primitives to the sketch.

Finally, each snapped primitive has its own set of constraints on its own parameters. For example, if a cylinder is represented using its radius , a center point , a length value and an axis vector , we will have the constraint . All such constraints are also hard constraints for the final optimization process. We will describe each primitive in detail later, including every primitive’s parameters and intrinsic constraints.

## Matching

The purpose of the matching process is to find the best matching of a new primitive’s curves to the sketch curves. Let be the curves of the sketch let be the curves of a primitive to be matched. First, a weight function is computed. Higher values represent better matching. Then the Hungarian algorithm [4] is run to find the best matching of a primitive’s curves to the sketch curves. The algorithm is run separately for feature curves and for silhouette curves.

The function is the value of the integral of along the line . We assume that the lower the distance transform integral is, the better the matching is, and the function will give higher values (note that the integrated function is the negated distance transform). Because in our case we computed the discrete version of the distance transform function, we use numeric integration (Simpson’s method) to approximate the integral.

The result of the matching is displayed to the user using color-coding. That is, a curve on the primitive and a curve on the sketch will have the same color if they match.

## Snapping

Snapping is a numeric optimization process that optimizes a target function subject to equality constraints.

### The target function

Let be the set of snapped primitives. For each primitive we define a function such that its minimum describes a best-fit of the primitive to the sketch in some sense. The target function of the whole optimization process is .

### The constraints

As stated before, each primitive might have a set of intrinsic constraints that constrain its parameters. And finally, every annotation also defines a set of constraints. Let be the set of annotations defined by the user. Let be the set of constraints induced by the annotation. The final set of constraints is

### Optimization algorithm

The system uses an Augmented-Lagrangian method [5] for the optimization. The AL method minimizes a target function subject to hard constraints by solving a sequence of unconstrained optimization problems.

The main advantage of using AL over a penalty method that is usually used in computer graphics applications is that we do not need to normalize the constraints. Therefore, computing the gradient of the target function is a much faster procedure and does not require careful construction of the constraints such that they do not affect the optimization procedure.

We will demonstrate this advantage with a simple 2D example. Let , and be 2D points. We would like to constraint those points to be collinear. That is the vectors and need to be parallel. We can express the parallelism using the following equation:

When we use it as a soft constraint in a penalty method we add the following term to the objective function:

However, if we look carefully we can see that minimizing the above term can be done both by making the points more “collinear” or just closer to each other. The result of the optimization process will divert us from our desired solution by making the points closer to each other, instead of best-fitting the sketch.

In order to avoid this issue and force the optimization procedure to make them just collinear we need to normalize this constraint such that positioning the points closer to each other will not minimize the term. This can be done by replacing this term by the following term:

However, we just made our objective function far more complicated than necessary. The Augmented Lagrangian method allows us to avoid this complexity and define the constraints in the most straight-forward way.

Note that there can be many ways to express collinearity. However all of them require some kind of normalization in order to avoid the “making the points closer” problem when using penalty methods.

## Primitive types

This section describes the types of primitives the system can currently handle. For each primitive we describe its parameters, its intrinsic constraints, its feature and silhouette curves and the objective functions for sketch-matching. Each primitive has more than one objective function to choose from, depending on the matching computed for its defining curves.

### Cylinder

A cylinder is defined using the following parameters:

* An axis vector
* The “bottom” center point
* A length parameter
* A radius parameter

The terms “top”, “bottom”, “left” and “right” used above and from now on are relative to the direction of the axis if we imagine that points up.

A cylinder has one intrinsic constraint: and four curves – top circle, bottom circle, left silhouette and right silhouette. The following illustration depicts the parameters:

* The blue bold point at the bottom is the “Bottom center”
* The radius and length are self-explanatory.
* The unit vector from the bottom center to the top center is the axis (and not vice versa!)
* The purple line is the left silhouette
* The orange line is the right silhouette.

The bottom feature curve is the 3D circle centered at having radius and normal direction . The top feature curve is a 3D circle centered at having the radius and normal direction .

In the next sub-sections we describe the objective function for the cases handled by the system.

#### Full matching

In this case all the curves of the 3D cylinder have a matching sketch curves. In this case, the system does not use the sketched silhouette lines at all and only uses the matching feature curves on the sketch.

The objective function is the average of the projection fit of the top feature curve and the projection fit of the bottom feature curve.

Let us define the “projection fit” function for a feature curve centered at , having normal direction and radius . We assume that the feature curve is matched to a 2D sketch curve P. We uniformly sample P at points to obtain a set of 2D samples . The projection fit of the circle to the 2D curve is the average of the projection fits of all sample points.

The projection fit function for a single sample point is given by the following formula:

The above formula, which seems quire scary, was derived from the following observation:

A 2D point is a projection of a point on a 3D circle if for some Z coordinate it resides on the plane defined by the center and the normal, and its distance from the center is the radius.

The rest is just simple algebra. We also use the intrinsic constraint in order to derive the above formula. Without this constraint the formula becomes much more complicated. Remember that we don’t actually need to differentiate the above term, only express it. AutoDiff does the tedious gradient computation work for us.

#### Two silhouettes single feature

This is the case when we have only 3 sketch curves matched to the primitive – two silhouette curves and one feature curve.

We will use the following illustration to explain the objective function:

P

Q

The top curve that looks like an ellipse contributes a “projection fit” term to the matched feature curve. Projection fit was explained above.

Using the two points P,Q we define the following terms:

Where is the 2D projection of the center of the relevant primitive feature curve. The contribution to the objective function is . The final objective function is the average of both contributions.

The objective function expresses the fact that the top sketch feature curve resides on the 3D circle defined by the matching primitive feature curve, and the bottom points are anti-polar points on the 3D circle defined by the other primitive feature curve.

#### Two silhouettes no feature

In the case of two silhouettes – we use the same idea as in the previous subsection. On every side the end-points contribute an objective function saying that the points are anti-polar points on the relevant 3D circle.

### Cone

Cone is similar to a cylinder, except having two radii – top radius and bottom radius. Thus a cone has one more variable. The matching and objective functions are the same as for the cylinder case.

### Sphere

A sphere is defined by the following parameters;

* A center
* A radius .

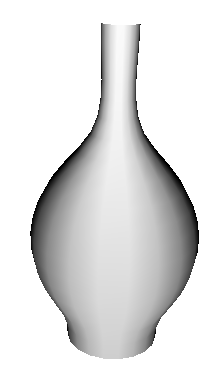
It has one silhouette curve – the great circle that is parallel to the viewing plane. This silhouette curve is matched against the sketch in the matching process so that the system knows which sketch curve belongs to the sphere.

The objective function is simple. Given a sketch curve, we sample it on points . The objective functions attempts make all the points to be at distance from the center . That is, the objective function is:

The above formula simply implies that we would like to make the X and Y coordinates of all the sketch points lie as close as possible to the great circle on the sphere that is parallel to the viewing plane.

### Straight generalized cylinder

Straight generalized cylinders (SGC) are a generalization of the cone. It has a straight spine (a line segment) but the radius around every point of the line may vary. The following image illustrates this kind of primitive.



Theoretically, the variation of the radius along the spine can be smooth. However, represent the varying radius using a discrete set of component. Each component is defined by its “progress” along the spine (0 means at one end, 1 means at the other end, 0.5 means at the middle) and a radius at this position. So the parametric representation of an SGC consists of the following parameters:

* An axis vector
* The “bottom” center point
* A length parameter
* A set of components, where component is defined by:
  + Progress . This parameter is constant, and not a variable being optimized. We have ranging from in . The components are sorted by in ascending order.
  + Radius .

Again, we have the intrinsic constraint .

Like cylinders and cones, an SGC has two feature curves and two silhouette curves. Their definition is the same as in the case of cylinders and cones.

The following diagram explains the objective function construction process. The axis approximation and spine approximation steps vary between the different cases, depending on the available matched sketch curves. The objective function construction step is the same in all cases.

Before we describe the construction of the objective function we describe the basic building blocks that are used for this construction – **ellipse orientation extraction**, **radii computation** and **objective function construction**. Afterwards we describe the exact method of axis approximation and spine approximation for the various cases.

### Building blocks

#### Ellipse orientation extraction

This step is used as a building block in the axis approximation process. Given a 2D curve on a sketch that looks like a part of an ellipse, we use the algorithm described in [6] to find the best fitting ellipse. After fitting the parameters of this ellipse we use them to extract the parameters of the 3D circle that projects to this ellipse. The system assumes that elliptical curves represent projections of 3D circles.

Let us denote by and the major and the minor axis vectors of the ellipse. The value is the major radius, and is the minor radius. The radius of the 3D circle is . Next, we compute two basis vectors for the plane where the circle resides

Using the above vectors we can compute a (normalized) normal vector of the circle’s plane, which is the 3D orientation of the ellipse:

#### Spine approximation

The “spine” of the generalized cylinder is defined by two end-points and , and a set of radii associated with different progress points between and (progress is in range ). The set of progresses are given as input to the algorithm (usually points uniformly distributed between 0 and 1).

We have various spine approximation algorithms for various cases and they differ in the input they receive and all product , and the set as their output.

#### Generic objective function

The objective function for snapping is constructed given the following information:

* Approximate axis - . This vector must be normalized.
* Approximate 2D projections of the spine – ,
* A list of values where is the approximate radius for the th component. The final objective function has the following form:

Where:

* – Orientation term. Makes the axis approach the approximate axis. We have:  
  Because the approximate axis is normalized, and we have an intrinsic constraint that forces to be normalized too, this distance actually makes sense.
* – Radii Approximation Term. Makes the radii approach the provided approximate radii. We have:
* – Radii Smooth Term. We assume that the radii vary smoothly along the spine of the SGC. Therefore, this is a simple Laplacian smoothness term, that is:
* – Start-point Term. Makes the projection of the spine’s start-point be as close as possible to the bottom center b. That is:
* – End-point Term. Makes the projection of the spine’s end-point be as close as possible to the top center. That is:

We can see that the information provided to the objective function construction procedure is the information computed in the axis approximation and spine approximation stages.

### Cases for objective function construction

In this section we will describe exactly how the input to the “Generic objective function” process is computed in various cases.

#### Full info – two silhouettes two features

##### Axis approximation

We use the two sketch curves matched to the primitive’s feature curves. We fit ellipses and extract the circle orientation of both sketch curves. Let us denote by the normal vector of the circle extracted from the top sketch curve and the normal vector extracted from the bottom sketch curve.

If the perimeter of the ellipse fitted for the top curve is larger than the bottom, we output as the axis approximation. Otherwise we output . The perimeter cannot be exactly computed, but can be approximated very well. We use the formula denoted as “Infinite series 2” in [7] and truncated it after four terms:  
Where:  
and , are the major and minor radii of the ellipse.

##### Spine approximation

The algorithm’s input consists of two polylines and , the set of component progresses along the spine - , a 2D projection of a point on the spine and a 2D direction vector of the spine . The last two inputs are provided by the axis approximation stage as the center and normal (projected to 2D) of the chosen circle. The polylines are part of the sketch and the progress set is a part of the primitive’s representation.

The first step is projecting all the points of and on the line defined by and . The projection gives us two end-points of the 2D projection of the spine. The following illustration depicts this process:

The bold blue lines are and . The central thin line is the line defined by and . The dashed lines are the projection directions (orthogonal to the central thin line). The big blue points are the start and end-points of the spine. That is, those are and in the output.

After we found and we use linear interpolation using as the weights to sample points along the spine and project them back on the polylines and to get the radii. More specifically, the radius corresponding to component is the average of the distances of the point to the polylines and .

#### Single silhouette two features

…

## Annotations

Annotations are objects that constrain **feature curves of** snapped **primitives**. The constraints defined by the annotations participate in the optimization process that snaps primitives to the sketch and allow the user to specify the correct 3D alignment of objects.

We assume that all feature curves of snapped primitives are planar objects. That is, the whole curve lies in one plane. This is not an actual restriction because all the feature curves of the primitives described above are planar. In addition, the current software version only supports feature curves that are circles. That is, each curve is defined by three parameters where is the center, is the radius and is the normal vector.

### Parallelism

The parallelism annotation constrains the planes of two or more feature curves to be parallel. Let be the normal vectors of feature curves, the parallelism annotation adds the following constraints:

Where

### Cocentrality

This annotation constrains the centers of all selected feature curves to be the same point. Given two or more feature curves centers we define the following set of constraints:

That is, we constrain all the centers to be the same point.

### Collinear centers

This annotation constrains the centers of the selected feature curves to lie on a single line. Given three or more feature curve centers we define the following set of vectors:

Now we constrain the vectors to be parallel. That is, we add the following set of constraints:

The function is defined in the parallelism annotation.

#### Coplanarity

This annotation constrains two or more feature curves to lie on a single plane. Because the curves are planar, it just means that the planes of all of the feature curves are actually the same plane.

Given normal and centers of two or more feature curves - and we construct the following equations:

The above equations express parallelism of the normal vectors, and the fact that the centers of two consecutive feature curves lie on each other’s planes.

# References

|  |  |
| --- | --- |
| [1] | A. Griewank and A. Walther, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd ed., SIAM, 2008. |
| [2] | D. C. a. N. J. Liu, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming,* vol. 45, no. 1, pp. 503-528, 1989. |
| [3] | G. Borgefors, "Distance transformations in digital images," *Comput. Vision Graph. Image Process.,* vol. 34, no. 3, pp. 344--371, June 1986. |
| [4] | H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly,* vol. 2, pp. 83-97, 1955. |
| [5] | J. Nocedal and S. Wright, "Penalty and Augmented Lagrangian methods," in *Numerical Optimization* , 2nd ed., Springer, 2006, pp. 497-528. |
| [6] | R. Hal and J. Flusser, "Numerically stable direct least squares fitting of ellipses," *The Sixth International Conference in Central Europe on Computer Graphics and Visualization ,* vol. 21, no. 5, pp. 125-132, 1998. |
| [7] | "Perimeter of an Ellipse," [Online]. Available: http://www.mathsisfun.com/geometry/ellipse-perimeter.html. |