# Implementation

The system has an offline processing step after which comes the interactive modeling step. The input is assumed to consist of continuous polylines, each representing a sketched curve. During the offline processing step we compute a discrete distance-transform function for each curve over a 512x512 grid. The offline processing step is performed when a sketch is first loaded and its results are saved when the user chooses the “save” option from the menu.

The interactive modeling process maintains a data-model that consists of sketch curves (including their associated distance-transform grids), primitives, feature curves and constraints. Primitives are parametrically defined and the fitting process computes those parameters. For example – a cylinder has center, axis vector and radius. Each primitive has associated feature curves – 3D curves that represent sharp features of the primitive. For example, a cylinder has two circles as its feature curves. Constraints are equations that constrain the same parameters that define the primitives. For example – given a cylinder and a cone, if we want their axes to be perpendicular we will constrain the inner product of those axes to be zero.

When the user drags a primitive over the sketch the system automatically assigns sketch curves to the silhouette curves of that primitive. We use the integral of the distance-transform functions to compute a matching weight, and then run the Hungarian algorithm to find the best matching. This matching gives semantic meaning to the curves on the sketch. For example, a curve can be the top-circle of a cylinder, or its left silhouette. Then we immediately optimize the parameters of the primitive to fit the current assignment of sketch curves. If the user is happy with the fit and releases the mouse button the system attempts to infer additional constraints based on some heuristic rules, and then optimizes again subject to the newly inferred constraints.

As our fitting problem is an optimization problem subject to equality constraints, we use an Augmented-Lagrangian method to solve the optimization problem using a sequence of unconstrained problems. Each unconstrained problem is solved using L-BFGS. Augmented Lagrangian’s main advantage over the simpler penalty method is allowing us to use simple equations for the constraints without the need of “normalizing” them. For example, if we want three points , and to be collinear we can simply use the equation  
without the need to divide by the distances between the points to prevent the optimizer from making the points closer to each other instead of being more collinear. The ability to specify simple equations allows the algorithm to be fast enough to be almost real-time.