EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Geosemantic Snapping for Sketch-Based Modeling: Optimization Details

A. Shtof[1] and A. Agathos[2] and Y. Gingold[3] and A. Shamir[4] and D. Cohen-Or[1]

[1]Tel Aviv University, Tel Aviv, Israel
[2]West University of Timişoara, Timişoara, Romania
[3]George Mason University, Fairfax, USA
[4]The Interdisciplinary Center, Herzliya, Israel

1. Introduction

In this supplemental document, we describe our optimization problem in detail and our implementation of the augmented Lagrangian method we use to solve it.

2. Optimization problem

The paper describes the optimization problem's ingredients: its individual objective and constraint functions. We complement this description by explicitly stating the optimization problems solved by our system.

Each primitive p in our system has a set of parameters x_p. Fitting a primitive p to the sketch is done by minimizing an objective function $\phi_p(x_p)$ subject to the set of internal structure constraints $C_p(x_p) = 0$. In the paper we detailed x_p, ϕ_p and C_p for each kind of primitive.

The inter-primitive relationships of the model are specified by a set of geosemantic relationships G. For a relationship $g \in G$ we define x_g to be the set of parameters that participate in this relationship. For example, if g is an "orthogonal" relationship between two circles, then x_g is a vector of six components: the normal vectors of both circles. For each $g \in G$ we define a constraint equation $\psi_g(x_g) = 0$. The constraint equations for each relationship are described in the paper. Note that the components of x_g are a subset of $\bigcup_p x_p$.

While the user drags a primitive p, the optimization problem we solve is:

$$\begin{aligned} min: \quad & \phi_p(x_p) \\ s.t.: \quad & C_p(x_p) = 0 \end{aligned} \tag{1}$$

This problem is solved using the augmented Lagrangian method [NW06], with the user's specified position and orientation as the starting point. By solving this problem while dragging, the user is shown a real-time "preview" of the snapped state of the primitive. Let x_p^* be the solution of this optimization problem.

Let $P = \{p_1, ..., p_m\}$ be the set of primitives in the modeled object and let $x = (x_{p_1}, ..., x_{p_m})$ be the vector of all primitives' parameters. When the user finishes dragging and releases a primitive p, the system uses the last obtained x_p^* to infer geosemantic relations (constraints) and add them to G. Then, the primitive is added to P and the following optimization problem is solved:

$$\begin{aligned} \min_x: \quad & \textstyle\sum_{p \in P} \phi_p(x_p) & \\ s.t.: \quad & C_p(x_p) = 0 & \forall p \in P \\ & \psi_g(x_g) = 0 & \forall g \in G \end{aligned} \tag{2}$$

Let $x^*_{P_{prev}}$ be the state of the primitives in the model before p was added. We define $x_{start} = (x^*_{P_{prev}}, x_p^*)$ and use it as the starting point to an augmented Lagrangian optimization algorithm for solving Equation 2. The intuition behind x_{start} is that the snapped state of the newly added primitive without geosemantic constraints is a good initial guess for its final state once geosemantic constraints are included.

3. Augmented Lagrangian optimization

To solve our equality-constrained optimization problems (Equations 1 and 2), we employ the augmented Lagrangian method. This section provides an intuitive description of the technique—see Nocedal and Wright [NW06] for a thorough presentation—as well as pseudocode, implementation details, and motivation.

An equality-constrained optimization problem can be expressed in the following form:

$$\begin{aligned} min: \quad & f(x) \\ s.t.: \quad & c_1(x) = 0 \\ & c_2(x) = 0 \\ & \cdots \\ & c_m(x) = 0 \end{aligned}$$

The Augmented Lagrangian method, like many optimization methods, proposes to solve this problem by solving a sequence of unconstrained problems.

Given a constant μ and set of constants $\lambda = \{\lambda_1, ..., \lambda_m\}$, define the following unconstrained objective function:

$$L_{aug}(x;\mu,\lambda) = f(x) + \underbrace{\sum_{i=1}^{m} \lambda_i \cdot c_i(x)}_{\text{Lagrangian term}} + \underbrace{\frac{\mu}{2} \cdot \sum_{i=1}^{m} |c_i(x)|^2}_{\text{Penalty term}}$$

Intuitively, we can see that this function resembles both the Lagrange function and a penalty function. Intuitively, we can see that minimizing this function over x attempts to approximate the KKT conditions:

$$\begin{aligned} \nabla_x L &= 0 \\ c_i(x) &= 0 \end{aligned}$$

With this intuition in mind, pseudocode for an augmented Lagrangian algorithm is:

1: **procedure** AUGMENTED-LAGRANGIAN($x^{(0)}$, $\lambda^{(0)}$, μ)
2: $i \leftarrow 0$
3: **while** not converged **do**
4: $x^{(i+1)} \leftarrow \operatorname{argmin}_y L_{aug}(y;\mu,\lambda^{(i)})$ starting from $x^{(i)}$
5: **for** $j = 1 \rightarrow m$ **do**
6: $\lambda_j^{(i+1)} \leftarrow \lambda_j^{(i)} + \mu \cdot c_j(x^{(i+1)})$
7: **end for**
8: $i \leftarrow i + 1$
9: **end while**
10: **end procedure**

Line 4 minimizes the unconstrained objective function L_{aug}. Lines 5–7 update λ to better approximate the actual Lagrange multipliers. This ensures that the next iteration of line 4 will better approximate the KKT conditions. This simple update formula is based on the observation that the gradient of L_{aug} and of the actual Lagrange function are very close when x is close to constraint satisfaction.

Implementation details. We use L-BFGS [LN89] to solve the unconstrained optimization problem on line 4. We evaluate gradients using reverse-mode automatic differentiation [GW08]; the library that we developed for this purpose has been made available to others as an open source project[†].

More sophisticated implementations of the augmented Lagrangian method update μ to accelerate convergence, but it is not required. (Under certain conditions it can be shown that for a given problem there exists some μ_0 such that the algorithm will converge for all $\mu > \mu_0$.) Our implementation does include one such method to accelerate convergence. The pseudocode for our method is:

1: **procedure** AUGMENTED-LAGRANGIAN-ACCEL($x^{(0)}$, $\lambda^{(0)}$, μ_0, μ_{max}, α, β)
2: $i \leftarrow 0$
3: $\mu \leftarrow \mu_0$
4: $C_{max} \leftarrow \frac{1}{\mu^\alpha}$
5: **while** not converged **do**
6: $x^{(i+1)} \leftarrow \operatorname{argmin}_y L_{aug}(y;\mu,\lambda^{(i)})$ starting from $x^{(i)}$
7: **for** $j = 1 \rightarrow m$ **do**
8: $\lambda_j^{(i+1)} \leftarrow \lambda_j^{(i)} + \mu \cdot c_j(x^{(i+1)})$
9: **end for**
10: **if** $\sum_{i=1}^{m} |c_i(x^{(i+1)})|^2 < C_{max}$ **then**
11: $C_{max} \leftarrow \frac{C_{max}}{\mu^\alpha}$
12: **else**
13: $\mu \leftarrow min(\mu \cdot \beta, \mu_{max})$
14: $C_{max} \leftarrow \frac{1}{\mu^\alpha}$
15: **end if**
16: $i \leftarrow i + 1$
17: **end while**
18: **end procedure**

(Changes from AUGMENTED-LAGRANGIAN are highlighted in red.) The intuition behind AUGMENTED-LAGRANGIAN-ACCEL is that if the constraint violation is not reduced enough, we increase μ. We use $\mu_0 = 10$, $\mu_{max} = 1000$, $\alpha = 0.5$, and $\beta = 2$; we initialize $\lambda_i^0 = 0$.

Augmented Lagrangian vs. Penalty methods. The penalty method, too, solves a sequence of unconstrained problems. Given a parameter μ, penalty methods optimize the following unconstrained objective function:

$$L_{pen}(x;\mu) = f(x) + \frac{\mu}{2} \cdot \sum_{i=1}^{m} |c_i(x)|^2$$

In theory, as μ goes to infinity the solution obtained by minimizing $L_{pen}(x;\mu)$ converges to the solution of the constrained optimization problem. In practice, one needs to minimize multiple times with increasing values of μ. During development, we were not able to find a simple strategy for updating μ that converges quickly without getting stuck in a visually distorted local minima. In contrast, the augmented Lagrangian method converges quickly and is more "resilient" to the non-convexity of our optimization problem.

References

[GW08] GRIEWANK A., WALTHER A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial Mathematics, 2008. 2

[LN89] LIU D. C., NOCEDAL J.: On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45* (1989), 503–528. 10.1007/BF01589116. 2

[NW06] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*. Springer, 2006. 1

[†] http://autodiff.codeplex.com