Convex Cardinal Shape Composition and Object Recognition in Computer Vision

Alireza Aghasi and Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332-0250
Email: {alireza.aghasi, jrom}@ece.gatech.edu

Abstract—This work mainly focuses on the segmentation and identification of objects present in an image, where the geometry sought is composed of given prototype shapes. Given a dictionary of prototype shapes, we define our problem as selecting a limited number of dictionary elements and geometrically composing them through basic set operations to characterize desired regions in an image. Aside from imaging applications such as shape-based characterization and optical character recognition, this problem is closely linked to the geometric packing problem. A recent work proposes a convex relaxation to this combinatorial problem [1], and the main focus of this paper is to computationally address the proposed convex program. We consider an alternating direction method of multipliers (ADMM) scheme, which suits a parallel processing framework and supports large-scale problems.

I. INTRODUCTION

The focus of this paper is a problem, instance of which may arise in various areas of imaging [2], [3], [4], computer vision [5], [6], [1], operations research and integer programming [7], [8], [9]. The main idea is to efficiently represent objects of interest as a concise composition of some prototype shapes. For instance, in an image segmentation application, the goal might be to perform the segmentation task, while making sure that the reconstructed segments are efficiently composed of a fixed number of prototype shapes [1]. This is simply because it often happens that the geometry of interest is composed of simpler building-blocks. Moreover, if we have a fixed collection of simple prototype shapes, being composed of few prototypes may be a way of regularizing the problem. The shape composition here would exploit set operative interactions (such as the union and possibly set differences), simply because objects usually interact through such operators (overlapping modeled by the sets union and occlusion modeled by the set difference). Depending on the application, formally addressing problems of this kind can be extremely hard, mainly because of their combinatorial nature. A good example is the geometric packing problem concerned with the arrangement of given objects inside larger containers. Many instances of this problem are known to be NP-complete [8], [9].

Speaking of the image segmentation problem, which will be the main focus of this paper, a general goal is often to label and aggregate pixels that are visually or statistically related. Specifically, in the binary case, a given image \( D \) needs to be partitioned into disjoint regions \( \Sigma \) and \( D \setminus \Sigma \), such that “similar” pixels lie on the same segments.

For a given dictionary of prototype shapes, \( \Sigma_1, \Sigma_2, \ldots, \Sigma_n \), an interesting segmentation problem is to characterize the target object \( \Sigma \) as a composition of selected elements of the dictionary. A flexible composition rule proposed in [3], [1] is

\[
\mathcal{R}_{I_\theta, I_\delta} \triangleq \left( \bigcup_{j \in I_\delta} \Sigma_j \right) \setminus \left( \bigcup_{j \in I_\delta} \Sigma_j \right),
\]

which allows interaction of the prototype shapes in the form of overlapping and occlusion. To promote simpler geometries (i.e., geometrically regularizing the problem), a cardinality restriction applies to the target sets \( I_\theta, I_\delta \in \{1, 2, \ldots, n_s\} \).

A formal formulation of this problem was first proposed in [1] as a challenging combinatorial problem. To address the problem, the authors proposed a convex surrogate, and provided a detailed analysis about the performance of the proposed proxy. As the main focus in [1] was a theoretical justification of the convex model, the computational aspects of the work mainly remained undiscussed.

In this paper, we first provide a quick overview of the shape composition problem presented in [1], and discuss the mechanics of the underlying convex model. We then propose a computational scheme based on the alternating direction method of multipliers (ADMM) to address the proposed program. We derive a closed-form expression of the proximal operator associated with the main components of the convex objective. Such closed-form expression plays a key role in the computational efficiency of our solver. The solver well suits a parallel processing framework, and allows addressing large scale imaging problems. We finally discuss some experiments and provide the concluding remarks.

A. Background

For an imaging domain \( D \subset \mathbb{R}^d \) with pixel values \( u(x) \), \( x \in D \), a binary segmentation corresponds to partitioning \( D \) into two disjoint regions \( \Sigma \) and \( D \setminus \Sigma \), where each region encompasses similar pixels. A well-known variational model is determining \( \Sigma \) via the minimization

\[
\Sigma^* = \arg \min_{\Sigma} \gamma(\Sigma) + \int_{\Sigma} \Pi_{in}(x) \, dx + \int_{D \setminus \Sigma} \Pi_{ex}(x) \, dx, \tag{2}
\]

where \( \gamma(\Sigma) \) is a regularization term promoting a desired structure, and \( \Pi_{in}(\cdot) \geq 0 \) and \( \Pi_{ex}(\cdot) \geq 0 \) are some image-dependent inhomogeneity measures.

A widely-used measure is the one proposed by Chan and Vese [10], which takes \( \Pi_{in}(x) = (u(x) - \bar{u}_{in})^2 \) and \( \Pi_{ex}(x) = \]
search domain, consider selecting shape redundancy. It often happens that \( \Pi_{in} \) and \( \Pi_{ex} \) are fixed or can be estimated a priori. For instance, in the case of classic Chan–Vese model, applicable to images of almost binary nature with bimodal histograms, the quantities \( \tilde{u}_{in} \) and \( \tilde{u}_{ex} \) can be roughly estimated from the histogram. In such situation

\[
\int_{D \setminus \Sigma} \Pi_{ex}(x) \, dx = \int_D \Pi_{ex}(x) \, dx - \int_{\Sigma} \Pi_{ex}(x) \, dx,
\]

and since the first term on the right-hand side is constant with respect to \( \Sigma \), an equivalent formulation of (2) becomes

\[
\Sigma^* = \arg \min_{\Sigma} \gamma(\Sigma) + \int_{\Sigma} (\Pi_{in}(x) - \Pi_{ex}(x)) \, dx.
\]  

In this convex model the shape interactions are modeled by the combination of characteristic functions. The number of active shapes are controlled by the \( \ell_1 \) restriction on \( \alpha \). Among the active components of the solution, the components with positive values identify the \( I_B \) set and the negative values represent \( I_G \). The parameter \( \tau \) is a free parameter and often an integer quantity. In fact in many interesting scenarios this value can simply be taken to be the same as \( s \). The interested reader is referred to [1] for a more detailed discussion.

II. Addressing the Convex Program

The main focus in [1] is the convex analysis of (5) and characterizing its minimizers, for which the computational aspects of the program remained undiscussed. In this section we propose a computational scheme based on the ADMM technique [12] to computationally address (5). The ADMM has received considerable attention from various areas of science and engineering, mainly in the context of machine learning and distributed optimization (see [12], [13] for a comprehensive review of the algorithm).

A main component of the ADMM is the evaluation of the proximity operator. Given a convex function \( g(\alpha) : \mathbb{R}^n \to \mathbb{R} \), the proximal operator of \( g \) scaled by a factor \( \xi \) is defined by

\[
\text{prox}_\xi g(\rho) = \arg \min_{\alpha} g(\alpha) + \frac{1}{2\xi} \|\alpha - \rho\|^2.
\]  

The ADMM often applies to convex problems of the form

\[
\min_{\alpha} g(\alpha) + f(\alpha),
\]  

where both \( g \) and \( f \) are allowed to be non-smooth convex functions. To handle constrained convex problems, the function \( f \) is chosen to be the indicator of the constraint set.

Based on an equivalent reformulation of (9) as

\[
\min_{\alpha, \rho} g(\alpha) + f(\rho), \quad \text{s.t.} \quad \alpha = \rho,
\]  

the ADMM converges to a solution by alternating between a proximal update on \( \alpha \) and a proximal update on \( \rho \) (see §4.4 of [13] for details). The method would be of special interest when closed-form expressions for the proximal operators of \( g \) and \( f \) are available.

To proceed with an ADMM formulation of the proposed convex model, we consider an image which consists of \( N \) pixels \( x_1, x_2, \ldots, x_N \) (or voxels in 3D). A discrete formulation of (5) takes the following form:

\[
\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{N} \max \left( \alpha_i^T \mathbf{a}_i, b_i \right), \quad \|\alpha\|_1 \leq \tau.
\]
for $j = 1, \ldots, n_s$, the $j$-th entry of the vector $a_i$ is $\Delta(x_i) x_{S_j}(x_i)$, and $b_i = \Delta(x_j)^{-}$. For the quantized problem in (11), the objective may be cast as

$$\min_{\alpha} \sum_{i=1}^{N} g_i(\alpha) + f(\alpha),$$

(12)

where $g_i(\alpha) = \max(a_i^T \omega, b_i)$ and $f(\alpha) = \mathbb{I}_{\mathcal{C}_\tau}(\alpha)$ represents the indicator function of the convex set $\mathcal{C}_\tau = \{ \alpha \in \mathbb{R}^{n_s} : \|\alpha\|_1 \leq \tau \}$. That is,

$$\mathbb{I}_{\mathcal{C}_\tau}(\alpha) = \begin{cases} 0 & \|\alpha\|_1 \leq \tau \\ +\infty & \|\alpha\|_1 > \tau \end{cases}.$$  (13)

Now, problem (12) can be classified as an instance of the global variable consensus problem with regularization (see §7.1.1 of [12]), which is equivalent to the convex program

$$\min_{\alpha_1, \ldots, \alpha_N, \rho} \sum_{i=1}^{N} g_i(\alpha_i) + f(\rho), \quad \text{s.t.} \quad \alpha_1 = \cdots = \alpha_N = \rho.$$  (14)

As elaborated in [12], an ADMM iterative process to solve (14) takes the following alternating form:

$$\alpha_i^{k+1} = \text{prox}_{\xi g_i}(\rho^k - \omega_i^k),$$  (15)

$$\rho^{k+1} = \text{prox}_{(\xi/N)f}(\alpha^{k+1} + \omega^k),$$  (16)

$$\omega_i^{k+1} = \omega_i^k + \alpha_i^{k+1} - \rho^{k+1}.$$  (17)

Here, the superscript $k$ denotes the iteration index, the variable $\omega_i$ measures the deviation of $\alpha_i$ and $\rho$, and the variables with the bar denote the mean vectors, i.e., $\bar{\omega}^k = \frac{1}{N} \sum_{i=1}^{N} \omega_i^k$ and $\bar{\alpha}^k = \frac{1}{N} \sum_{i=1}^{N} \alpha_i^k$. Often times all variables are initialized as zero vectors, which might be a good initialization when the optimal $\alpha$ is a sparse vector. In general, a warm start can help the algorithm converge faster.

The quantity $\xi$ can be simply set to 1 throughout the iterations. However, as noted in §3.4 of [12], replacing $\xi$ with $\xi^k$ and iteratively updating $\xi$ is capable of enhancing the convergence rate compared to the case of fixing $\xi$ in the entire process. For the purpose of this paper, however, the focus will be on the simple case of $\xi = 1$.

The proposed iterative scheme can be efficiently carried out in a distributed computing framework. Each subsystem $i$ would be in charge of computing the proximal update on $\alpha_i$ and an evaluation of $\omega_i$, as suggested by (15) and (17). Once these quantities are calculated, they will be averaged and passed to a single processing unit to perform the proximal calculation pointed out in (16).

We would like to note two facts about the proximal calculations in (15) and (16), which highlight the motivation behind using an ADMM scheme. First, it is straightforward to see that for $f(\alpha) = \mathbb{I}_{\mathcal{C}_\tau}(\alpha)$, the proximity function in (16) reduces to

$$\text{prox}_{(\xi/N)f}(\rho) = P_{\mathcal{C}_\tau}(\rho),$$

where $P_{\mathcal{C}_\tau}$ denotes the projection onto the convex set $\mathcal{C}_\tau$. For a fast implementation of the $\ell_1$ projection we suggest using the approach proposed in [14], which has a worst-case complexity of $O(n_s \log n_s)$, noting that the average complexity for many practical problems is much less.

In the regularized form of the proposed convex problem which is formulated as

$$\min_{\alpha} \int_D \max\{\Delta(x) \sum_{j=1}^{n_s} \alpha_j x_{S_j}(x), \Delta(x)^{-}\} \, dx + \lambda \|\alpha\|_1,$$  (18)

for $f(\alpha) = \lambda \|\alpha\|_1$, the proximity function reduces to the soft thresholding operator (cf. §6.5 in [13]):

$$\text{prox}_{\xi g_i}(\cdot) = \begin{cases} \alpha_j - \frac{\xi \lambda}{N} & \alpha_j > \frac{\xi \lambda}{N} \\ 0 & \alpha_j \leq \frac{\xi \lambda}{N} \quad \alpha_j \leq -\frac{\xi \lambda}{N} \end{cases}.$$  

Second, a main motivation to use the proposed ADMM scheme is the possibility of deriving a closed form expression for $\text{prox}_{\xi g_i}(\cdot)$. More specifically,

$$\text{prox}_{\xi g_i}(\rho) = \begin{cases} \rho - \frac{\xi a_i}{\|a_i\|^2} \rho + \frac{b_i}{\|a_i\|^2} a_i & a_i^T \rho \in (b_i + \xi \|a_i\|^2, \infty) \\ \rho & a_i^T \rho \in (-\infty, b_i) \end{cases}.$$  (19)

This derivation is presented in Appendix A. Basically, the proximal updates on the vectors $\alpha_i$ can be performed very cheaply, and the proposed scheme can be conveniently applied to large scale problems.

### III. Experiments and Discussions

In this section, through an example, we discuss the performance of the proposed ADMM scheme. For a comprehensive set of experiments related to the general performance of the convex proxies (5) and (18), the reader is referred to §5 in [1].

Similar to the gradient descent scheme applicable to a large class of optimization problems with smooth objectives, the ADMM can serve as a standard tool for addressing convex problems in a distributed fashion. Despite its wide applicability and general-purpose nature, the ADMM can be slow to converge to high accuracy, especially in addressing large-scale problems.

While the computational time within each iteration can be very short (especially thanks to the parallel processing feature), in order to attain an accurate solution, the algorithm may require a large number of iterations [12]. Therefore, without considering any modifications such as iteratively updating the penalty parameter $\xi$, or combining the algorithm with other post-processing schemes, the present ADMM scheme best suits the regularized version of the proposed convex proxy (18). This is mainly because the focus in (18) is finding a concise representation for an object in the image, while in using (5) we are often expecting a sparse $\alpha^*$ solution with integer nonzero components. Converging to exact integer values can take many iterations for an ADMM scheme. More discussions and suggestions will be provided in the context of the following example.

We specifically focus on one of the optical character recognition (OCR) examples in [1], where the primary objective is the identification of letters appeared in an image. We
consider the identification of the letters inside an image of size $63 \times 160$ pixels (shown in Figure 1) which represents the word “Sample”. The letters in the image undergo different case, size, rotation and level of overlap. To build up the character dictionary, yet maintaining a reasonable-sized dictionary, we use characters of a similar font, but in various uppercase/lowercase formats. The character shapes are placed throughout the image in different size and orientations. With this setup, we build up a dictionary of $n_s = 2056$ character elements.

Specifically for $\tau = 5$, we have shown the output of (5) using CVX [15]. As could be observed, the nonzero components of $\alpha^*$ correspond to five main characters appeared in the image. Increasing the value of $\tau$ by unit steps, from $\tau = 1$ to $\tau = 6$, allows us to identify the main characters within the image one after the other (see [1] for a complete set of experiments).

In this example, there are two characters “m” in the dictionary which closely match the letter “m” within the image. The cost difference that selecting one of these letters over the other causes is negligible and by referring to the example in [1] we can see that for $\tau = 1, 2, 3$ one of the letters “m” is identified and for $\tau = 4, 5, 6$ the solution identifies the other.

Figure 2(a) shows the outcome of an ADMM scheme at different iterations. As a matter of fact the letters “a”, “p”, “L” and “E” are identified at earlier iterations (before $k = 1200$) and through several hundred iterations the algorithm tries to increase the weight of one of the letters “m” to 1, and suppress the other one, to ultimately recover the solution in Figure 1. The rather slow progression of the ADMM scheme is due to the negligible cost difference between two solutions, each with one of the letters “m” being active. For instance, if one of the two letters is removed from the dictionary, the ADMM scheme converges to something close to the true solution in less than 1200 iterations, as shown in Figure 2(b).

Figures 3(a) and 3(b) show the values of $\|\hat{\alpha}^k - \alpha^*\|$ and the cost at different iterations for the two aforementioned cases, one with both letters “m” present in the dictionary and one with a letter removed. While a low cost and sparse solution can be attained within a few hundred iterations, accurate solutions require more iterations. In the case of having two similar target elements in the dictionary (black curve in Figure 3(a)), after approximately 1100 iterations the algorithm starts a linear convergence to $\alpha^*$ with a moderately small slope, justifying the small improvement in balancing the coefficients between the two letters “m” in Figure 2(a).

Since after this point the main updates of the ADMM will be over the support of $\hat{\alpha}^k$, the off-support components of $\alpha$ can be neglected and the ADMM can be carried out solely on the support for a significant computation reduction and speed enhancement. A reasonable combined scheme starts with the entire elements of the dictionary and after reaching an almost steady-state, where the majority of the zero components of $\alpha^*$ are identified, the ADMM is continued solely on the nonzero components. Using this scheme within only a few seconds a convergence to the global minimizer $\alpha^*$ is attained (the outcome is very similar to Figures 1 and 2(b) and eliminated

IV. Conclusion

An ADMM scheme to address the convex cardinal shape composition problem in [1] is proposed. Closed-form expressions for the underlying proximal operators along with the parallel processing feature allow us to consider the scheme for large-scale problems. The main limitations of the core ADMM scheme are discussed and suggestions based on the suitability of the algorithm for the regularized version of the convex problem are stated. In the constraint form of the problem, where the ADMM can exhibit slow convergence, combining the core ADMM framework with a pruning scheme allows a significant computation reduction and achieving the global minimizer at a significantly lower computational cost.

Appendix A

Derivation of the Proximal Operator

The focus of this section is the proximal calculation for

$$g(\alpha) = \max(a^T \alpha, b),$$

based on the definition in (8). Let’s denote the minimizer of (8) by $\rho^*$. We consider breaking the $\alpha$-domain into three regions relative to the hyperplane $a^T \alpha = b$: (a) points where $a^T \alpha > b$; (b) points where $a^T \alpha < b$; (c) points that lie on the hyperplane.
We then discuss the conditions under which $\rho^*$ lands in each region.

(a) $a^T\alpha > b$: In this region $g(\alpha) = a^T\alpha$ and the proximal calculation reduces to

$$\rho^* = \arg \min_\alpha a^T\alpha + \frac{1}{2\xi}||\alpha - \rho||^2.$$ 

The minimizer of the resulting quadratic program is simply $\rho^* = \rho - \xi a$. The feasibility of $\rho^*$ (i.e., the minimizer being in the designated region) requires $a^T(\rho - \xi a) > b$ or simply $a^T\rho > b + \xi a^Ta$.

(b) $a^T\alpha < b$: In this region the proximal program reduces to

$$\rho^* = \arg \min_\alpha b + \frac{1}{2\xi}||\alpha - \rho||^2,$$

which trivially yields $\rho^* = \rho$. The feasibility of $\rho^*$ in this case requires

$$a^T\rho < b.$$ 

(c) $a^T\alpha = b$: In the case that $b \leq a^T\rho \leq b + \xi a^Ta$, neither regions $a^T\alpha > b$ or $a^T\alpha < b$ could be in hold of $\rho^*$, and the point must lie on the hyperplane $a^T\alpha = b$. Subsequently,

$$\rho^* = \arg \min_\alpha b + \frac{1}{2\xi}||\alpha - \rho||^2 \text{ s.t. } a^T\alpha = b.$$ 

To explicitly find $\rho^*$, we neglect the constant term and the $\xi^{-1}$ factor, and form the Lagrangian as

$$L(\alpha, \mu) = \frac{1}{2}||\alpha - \rho||^2 + \mu(a^T\alpha - b).$$

We denote $\mu^*$ as the optimal multiplier, where

$$(\rho^*, \mu^*) = \arg \min_{(\alpha, \mu)} L(\alpha, \mu).$$

Imposing the optimality conditions yields

$$\frac{\partial L}{\partial \alpha}(\alpha, \mu) = (\rho^* - \rho + \mu^* a) = 0, \quad (20)$$

and

$$\frac{\partial L}{\partial \mu}(\alpha, \mu) = a^T\rho^* - b = 0. \quad (21)$$

Simultaneously solving (20) and (21) for $(\rho^*, \mu^*)$ results in

$$\mu^* = \frac{a^T\rho - b}{||a||^2}, \quad \rho^* = \rho - \frac{aa^T}{||a||^2} + \frac{b}{||a||^2} a.$$ 

We can now combine the conditions and the minimizer evaluations for each region to summarize the proximal operator as in (19).

REFERENCES