Higher-Order Energies for Image Segmentation

Jianbing Shen, *Senior Member, IEEE*, Jianteng Peng, Xingping Dong, Ling Shao, *Senior Member, IEEE*, and Fatih Porikli, *Fellow, IEEE*

***Abstract*—A novel energy minimization method for general higher-order binary energy functions is proposed in this paper. We first relax a discrete higher-order function to a continuous one, and use the Taylor expansion to obtain an approximate lower-order function, which is optimized by the quadratic pseudo-boolean optimization (QPBO) or other discrete optimizers. The minimum solution of this lower-order function is then used as a new local point, where we expand the original higher-order energy function again. Our algorithm does not restrict to any specific form of the higher-order binary function or bring in extra auxiliary variables. For concreteness, we show an application of segmentation with the appearance entropy, which is efficiently solved by our method. Experimental results demonstrate that our method outperforms state-of-the-art methods.**

***Index Terms*—Higher-order energy, image segmentation.**

I. Introduction

Discrete optimization problems have been studied extensively in computer vision applications, such as image segmentation [3], [4], [5], [7], [36], [32], saliency [32], [41], [43], [48], [44], [47] and tracking [39], [45], [46], [42]. When a discrete energy function is the sum of terms related to two or fewer variables, it is called a lower-order energy function. In this case, a lot of energy optimization techniques can be used to minimize it efficiently, e.g., graph cuts ([9]), belief propagation ([11]), message passing [40], and random walks ([14], [15], [5]). However, more complicated and useful assumptions of energy formulations for describing natural images have been developed, which cannot be expressed by lower-order functions. As a result, higher-order energy functions have attracted much attention in recent years [27].

Numerous methods for optimizing higher-order energy functions with specific forms have been proposed including the P^n Potts model [18], Cooperative Cut [24], Decomposable submodular functions [22] and Deep Random Field [29]. Kohli *et al.* [18] propose a higher-order energy to encourage variables in one segment to take the same value for local consistency. Jegelka *et al.* [24] and Kohli *et al.* [29] use the cooperative cut and design the higher-order energy to enforce that the number of cut patterns is as small as possible. Gallagher *et al.* [25] introduce an order reduction inference algorithm to select the proper order reduction and minimize the Markov random field (MRF). When a higher-order energy function is transformed into a lower-order one, graph cuts can be used to obtain a satisfactory solution. However, if a lower-order function is not submodular, graph cuts cannot be applied directly. In such cases, the Quadratic Pseudo-Boolean Optimization (QPBO) [6], [16] and Tree-Reweighted Message Passing (TRWS) [13] algorithms can be used to approximately find the minimum value with partial optimality.

When the form of an explicit higher-order energy function is general, how to reduce the order of this energy efficiently is one of the major issues in the field of higher-order energy minimization. A large number of higher-order reduction methods have been proposed recently [8], [19], [20], [21], [26], [31], [30], [34], [33], [35], [38], [28], [23] to solve any order of higher-order binary energy functions in the polynomial form. Ishikawa [19] presents a Higher-Order Clique Reduction (HOCR) method by generalizing the minimum selection reduction method [9] to any order. This method uses auxiliary variables to transform each higher-order term to a lower-order one independently. Furthermore, Fix *et al.* [26] transform a group of higher-order terms into lower-order ones at once. In [33], Ishikawa further reduces higher-order binary terms into lower-order ones without introducing new variables using the excludable local configuration method. However, the objective higher-order functions tackled in [19] and [33] need to be polynomial. In contrast, our minimization method can be applied to both polynomial and non-polynomial functions, and polynomial potentials can be minimized by our method as a degenerate case. Gorelick *et al.* [31] propose an iterative approach called trust region to optimize higher-order energy. They used the first-order Taylor expansion to approximate the original higher-order function, and find its minimum energy in the trust region. Ayed *et al.* [30] present a general auxiliary function as the upper bound of the higher-order energy for segmentation, and minimize it with a bound optimization algorithm. The higher-order terms in [30] described regional properties of the segments. Our method does not require specific forms of the energy functions. Tang *et al.* [35] use parametric pseudo-bounds for a higher-order function, and iteratively optimize the bounds until convergence. However, it requires significant efforts to construct the parametric pseudo-bounds for all the energy functions. In this work, we construct approximate lower-order energy functions instead of lower-order pseudo bounds, which can be found more suitable for explicit general energy functions.

Any higher-order binary term can be transformed to a polynomial form with the sum of finite products of variables. Polynomial higher-order energy is discussed by two represen-

This work was supported in part by the National Basic Research Program of China (973 Program) (No. 2013CB328805), the National Natural Science Foundation of China (61272359), the Australian Research Council's Discovery Projects funding scheme (DP150104645), and the Fok Ying-Tong Education Foundation for Young Teachers. Specialized Fund for Joint Building Program of Beijing Municipal Education Commission.

J. Shen, J. Peng and X. Dong are with the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, P. R. China. (Email: shenjianbing@bit.edu.cn)

L. Shao is with the School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K. (Email: ling.shao@ieee.org)

F. Porikli is with the Research School of Engineering, Australian National University, and NICTA. (Email: fatih.porikli@anu.edu.au)

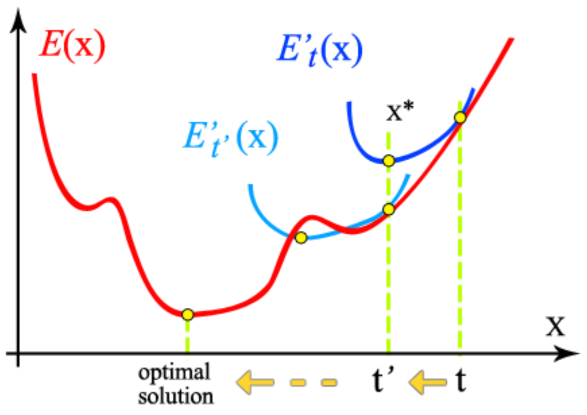


Fig. 1. An illustration of the proposed higher-order energy minimization method. The red curve is the higher-order binary energy function. Two blue curves are the approximate lower-order function at points **t** and **t**′. The minimum value of the dark blue curve at **t** is obtained in the first iteration; and the minimum value of the light blue curve at **t**′ is obtained in the second iteration. Generally, $E(\mathbf{t}') < E(\mathbf{t})$.

TABLE I
ONE EXAMPLE WITH EACH VALUE OF THE HIGHER-ORDER TERM f_h. ALL VALUES OF $f_h(x,y,z) = e^x \cdot |x - yz|$ IN THE WHOLE SOLUTION SPACE $\{0,1\}^3$ ARE PRESENTED IN THIS TABLE.

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
$f_h(x,y,z)$	0	0	0	1	e	e	e	0

tative methods in [19], [33], and can be defined as follows:

$$f_h(x_1, x_2, \dots, x_n) = \sum_{S \subset \{1,2,\dots,n\}} (\alpha_u \cdot \prod_{i \in S} x_i) \qquad u = \mathbb{B}(S) = \sum_{j \in S} 2^{j-1} \tag{1}$$

where $f_h(\cdot)$ is a higher-order term from a higher-order energy function; S is a subset of the index set (which can be empty); $\alpha_u \in \mathbb{R}$ is the coefficient of a product; and $\mathbb{B}(\cdot)$ is a function for mapping a set to a nonnegative integer. If $S = \emptyset$, we then define that $u = 0$ and $\prod_{i \in S} x_i = 1$.

We give an example of transforming an arbitrary higher-order term into the polynomial form in (1). Considering a binary higher-order term $f_h(x,y,z) = e^x \cdot |x - yz|$, we obtain its energies in the whole space $\{x,y,z\} \in \{0,1\}^3$ in TABLE **I**. After obtaining each solution of this term, we have:

$$\begin{aligned} f_h(x,y,z) =& e^x \cdot |x - yz| \\ =& 1 \cdot \bar{x}yz + e \cdot x\bar{y}\bar{z} + e \cdot x\bar{y}z + e \cdot xy\bar{z} \\ =& (-1 - e) \cdot xyz + yz + e \cdot x \end{aligned} \tag{2}$$

where $\bar{x} = 1 - x$.

Existing order reduction approaches accommodate any kind of higher-order binary function in a polynomial form [9], [19], [33]. For non-polynomial functions, it is necessary to transform them to polynomial form first, before applying order reduction methods to optimize them. However, this transformation step entails significant amount of work as pre-processing, which is usually implemented by exhaustively searching the whole solution space of the higher-order term

TABLE II
THE COMPARISON OF EACH HIGHER-ORDER REDUCTION METHODS.

	[19]	[21]	[26]	[30]	[33]	[35]	Ours
polynomial	√	√	√	√	√	√	√
non-polynomial	-	√	√	√	-	√	√
general form	√	-	-	-	√	-	√
many variables	-	√	√	√	√	√	√

(such as the one in TABLE **I**). Furthermore, we can consider a case when the higher-order energy f only contains a higher-order term ($f(\mathbf{x}) = 0 \cdot f_l(\mathbf{x}) + 1 \cdot f_h(\mathbf{x})$, and $f_l(\mathbf{x})$ is a lower-order term). Following the previous minimizing procedure, we need to transform f_h to the polynomial form. After the transformation step, we already obtain each value of f_h in the whole solution space. That is, the minimum value of f can be obtained by *selecting* the smallest value of f_h without *optimizing* f. This step of transforming a higher-order term from non-polynomial to polynomial is a redundant burden, but a necessary process to use existing order reduction methods.

Our method does not require this transformation step and can optimize a larger range of general higher-order energy functions directly. Other order reduction methods [30], [35] are based on iteratively minimizing upper bounds or pseudo bounds of the higher-order energy, which need to be carefully designed. These methods cannot find the upper bounds of general non-polynomial functions easily, and are used in some specific form of energy with finite higher-order assumptions, such as the ones for image segmentation. In contrast, our method reduces the order of a general binary function. Instead of building an upper bound like [30] and [35], our method constructs a lower-order function to approximate the higher-order objective function, which is more easier to construct.

Each of the state-of-the-art order reduction methods has its limitation. [19] and [33] can handle general polynomial higher-order terms, while non-polynomial higher-order terms are required to be converted into polynomial ones before reduction. Furthermore, [19] can not handle the situation where one higher-order term contains more than six variables in a real vision application. [26], [30] and [35] can optimize non-polynomial energies only with some specific forms. The higher-order potentials in [21] need to be concave. Our method can reduce general polynomial and non-polynomial higher-order terms, without the limitation of the number of variables. TABLE **II** shows the comparison of state-of-the-art algorithms and the advantage of our method briefly.

Numerous classic continuous optimization methods, such as the gradient descent method, the Newton-Raphson method and simulated annealing [2], adopt the step by step optimization approach. Similar ideas based on locally fitting and iterative optimizations also appear in the field of submodular optimization [37]. Our method is also based on this iteratively optimization framework. Higher-order energy functions become more and more popular, due to their representation of complex statistical information. Therefore, many applications [17], [18] in the field of computer vision have adopted higher-order functions to model the problem. In this paper, we use image segmentation as an example to demonstrate the effectiveness of our higher-order minimization method.

In our optimization algorithm, we first relax a discrete higher-order function to a continuous one, and reduce its order to a lower-order function at a point, which is approximately equivalent to the higher-order function locally. By minimizing this reduced function, a smaller energy of the original function is obtained, and the optimal solution is obtained through iteratively performing this process. This iterative optimization procedure in our method does not require any auxiliary variables. In each iteration, we construct a lower-order function to approximate the higher-order one by the first or second-order Taylor expansion. If the continuous version of the higher-order function is second order differentiable, we can definitely obtain its lower-order approximation. This paper will focus on considering the higher-order binary function whose continuous version is second order differentiable. Our source code will be available at [1].

Our main contributions are summarized as follows. 1) We propose a novel minimization method for general explicit higher-order energy functions, which can produce accurate solutions for both polynomial and non-polynomial higher-order energy functions. 2) Our method can reduce any order of a general higher-order energy function in a polynomial or non-polynomial form, without auxiliary variables. 3) We solve the application of image segmentation with the appearance entropy by our higher-order minimization method, which demonstrates the superiority of the proposed minimization algorithm.

II. Our approach

A. *Proposed higher-order minimization algorithm*

Our higher-order minimization method for binary energies is based on an iterative framework. Fig. 1 shows our iterative optimization approach, where we denote the higher-order energy function as $E(\mathbf{x})$. The main idea of the proposed algorithm is to approximate a lower-order energy function $E'_{\mathbf{t}}(\mathbf{x})$ at a single point $\mathbf{t}$ on the higher-order objective function. We then find the minimum value of this lower-order function and its solution $\mathbf{x}^*$. $\mathbf{x}^*$ is *generally* closer to the optimal solution than $\mathbf{t}$. After that, a new approximate lower-order function $E'_{\mathbf{t}'}(\mathbf{x})$ is found at the new point $\mathbf{t}' = \mathbf{x}^*$. We minimize these lower-order functions iteratively until convergence. In each iteration, the minimizer can obtain a lower energy until it converges at local or global minima.

Our method can optimize explicit general higher-order binary functions, which can be defined as:

$$E(\mathbf{x}) = E_l(\mathbf{x}) + E_h(\mathbf{x}), \quad where \quad \mathbf{x} = [x_1, x_2, \dots, x_n]^\top \quad (3)$$

where $\mathbf{x}$ is a solution in the discrete space $\{0,1\}^n$.

Given an index set $V = \{1, 2, \dots, n\}$, $\mathbf{x}$ can also be viewed as a variable set $\{x_i | i \in V\}$. In (3), $E_l(\mathbf{x})$ is the lower-order term, which contains unary and pairwise terms. Since the variables in the energy (3) are binary, lower-order terms can be expressed as the sum of product terms with two or

[1]http://github.com/shenjianbing/orderreduction

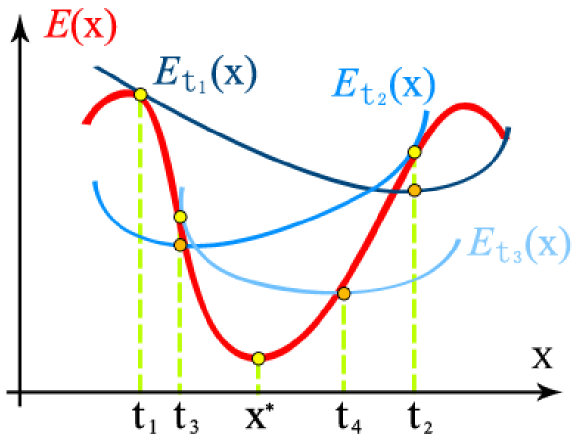


Fig. 2. An illustration of solutions vibrating around the minimum. $\mathbf{x}^*$ is the global minimum of energy $E(\mathbf{x})$ (red curve). Three blue curves $E_{\mathbf{t}_1}(\mathbf{x})$, $E_{\mathbf{t}_2}(\mathbf{x})$ and $E_{\mathbf{t}_3}(\mathbf{x})$ are the lower-order expansions (11) of $E(\mathbf{x})$ at $\mathbf{t}_1$, $\mathbf{t}_2$ and $\mathbf{t}_3$. $\mathbf{t}_2$ and $\mathbf{t}_3$ are the minimums of $E_{\mathbf{t}_1}(\mathbf{x})$ and $E_{\mathbf{t}_2}(\mathbf{x})$. During iterations, the solution is jumping from $\mathbf{t}_1$ to $\mathbf{t}_2$, then to $\mathbf{t}_3$, until it goes to $\mathbf{x}^*$. Therefore, if we do not set the step size, solutions will vibrate around the minima $\mathbf{x}^*$.

fewer variables as follows:

$$E_l(\mathbf{x}) = \sum_{i=1}^{n} u_i x_i + \sum_{1 \le i < j \le n} p_{ij} x_i x_j \quad (4)$$
$$= U^\top \mathbf{x} + \mathbf{x}^\top P \mathbf{x} \quad (5)$$

where $U \in \mathbb{R}^n$ and $P \in \mathbb{R}^{n \times n}$. In addition, u_i and p_{ij} are the elements of vector U and matrix P.

Unlike unary and pairwise terms, we do not fix the higher-order term as the product of variables like (1). We define the higher-order term as follows:

$$E_h(\mathbf{x}) = \sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}^c) \quad (6)$$

where c denotes a subset of the index set V, and $\mathbf{x}^c = \{x_i | i \in c, c \subseteq V\}$. In addition, $\mathcal{C}$ is a set containing all the higher-order cliques. For one term $\psi_c(\mathbf{x}^c)$ in $E_h(\mathbf{x})$, we denote its variables as $\mathbf{x}^c = [x^c_1, x^c_2, \dots, x^c_{|c|}]^\top$ and $\mathbf{x}^c \subseteq \mathbf{x}$. Assuming that we have an initial solution $\mathbf{t}$ of $E(\mathbf{x})$, we can extract its sub-solution $\mathbf{t}^c$ for higher-order term $\psi_c(\mathbf{x}^c)$, which satisfies $\mathbf{t}^c \subseteq \mathbf{t}$. The order of the higher-order term $\psi_c(\mathbf{x}^c)$ is reduced in the following way. We relax the discrete term $\psi_c(\mathbf{x}^c) : \mathbf{x}^c \in \{0,1\}^{|c|}$ to the continuous one $\widetilde{\psi}_c(\mathbf{x}^c) : \mathbf{x}^c \in \mathbb{R}^{|c|}$.

Next, each term in $E_h(\mathbf{x})$ is transformed approximately by the Taylor expansion at the place where $\mathbf{x}^c = \mathbf{t}^c$ ($\mathbf{t}^c = [t_1, t_2, \dots, t_{|c|}]^\top \in \{0,1\}^{|c|}$) as follows:

$$\begin{aligned}\widetilde{\psi}_c(\mathbf{x}^c) &= \widetilde{\psi}_c(\mathbf{t}^c) + \sum_{i=1}^{|c|} \frac{\partial \widetilde{\psi}_c(\mathbf{t}^c)}{\partial x^c_i}(x^c_i - t_i) \\ &+ \frac{1}{2!} \sum_{i,j=1}^{|c|} \frac{\partial^2 \widetilde{\psi}_c(\mathbf{t}^c)}{\partial x^c_i \partial x^c_j}(x^c_i - t_i)(x^c_j - t_j) + o \quad (7) \\ &= \widetilde{\psi}_c(\mathbf{t}^c) + [\nabla \psi_c(\mathbf{t}^c)]^\top (\mathbf{x}^c - \mathbf{t}^c) \\ &+ \frac{1}{2}[\mathbf{x}^c - \mathbf{t}^c]^\top H(\mathbf{t}^c)[\mathbf{x}^c - \mathbf{t}^c] + o\end{aligned}$$

where

$$\nabla\psi_c(\mathbf{t}^c) = \begin{bmatrix} \frac{\partial\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_1^c} & \frac{\partial\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_2^c} & \cdots & \frac{\partial\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_{|c|}^c} \end{bmatrix}^\top \tag{8}$$

$$H(\mathbf{t}^c) = \begin{bmatrix} \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial[x_1^c]^2} & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_1^c\partial x_2^c} & \cdots & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_1^c\partial x_{|c|}^c} \\ \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_2^c\partial x_1^c} & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial[x_2^c]^2} & \cdots & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_2^c\partial x_{|c|}^c} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_{|c|}^c\partial x_1^c} & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_{|c|}^c\partial x_2^c} & \cdots & \frac{\partial^2\widetilde{\psi}_c(\mathbf{t}^c)}{\partial[x_{|c|}^c]^2} \end{bmatrix} \tag{9}$$

and o represents the Peano type remainder of the Taylor expansion. One prerequisite for the expansion in (7) is $\widetilde{\psi}_c(\mathbf{x}^c)$ must be second order differentiable. All the higher-order terms can be expanded, and thus for $E(\mathbf{x})$, we have:

$$\begin{aligned} E(\mathbf{x})|_{\mathbf{x}\in(\mathbf{t}-\epsilon,\mathbf{t}+\epsilon)} &\approx E_l(\mathbf{x}) + \sum_c \Big(\widetilde{\psi}_c(\mathbf{t}^c) \\ &+ [\nabla\psi_c(\mathbf{t}^c)]^\top(\mathbf{x}^c - \mathbf{t}^c) + \frac{1}{2}[\mathbf{x}^c - \mathbf{t}^c]^\top H(\mathbf{t}^c)[\mathbf{x}^c - \mathbf{t}^c]\Big) \\ &= E_l(\mathbf{x}) + \\ &\sum_c \Big([\nabla\psi_c(\mathbf{t}^c)]^\top\mathbf{x}^c + \frac{1}{2}[\mathbf{x}^c - \mathbf{t}^c]^\top H(\mathbf{t}^c)[\mathbf{x}^c - \mathbf{t}^c]\Big) \\ &+ \underline{\sum_c \Big(\widetilde{\psi}_c(\mathbf{t}^c) - [\nabla\psi_c(\mathbf{t}^c)]^\top\mathbf{t}^c\Big)} \\ &= E_l(\mathbf{x}) + \sum_c \Big([\nabla\psi_c(\mathbf{t}^c)]^\top\mathbf{x}^c + \frac{1}{2}[(\mathbf{x}^c)^\top H(\mathbf{t}^c)\mathbf{x}^c \\ &- (\mathbf{t}^c)^\top(H(\mathbf{t}^c)^\top + H(\mathbf{t}^c))\mathbf{x}^c + \underline{(\mathbf{t}^c)^\top H(\mathbf{t}^c)\mathbf{t}^c}]\Big) + \text{const} \\ &= E_l(\mathbf{x}) + \sum_c \Big([\nabla\psi_c(\mathbf{t}^c)]^\top\mathbf{x}^c + \\ &\frac{1}{2}[(\mathbf{x}^c)^\top H(\mathbf{t}^c)\mathbf{x}^c - (\mathbf{t}^c)^\top(H(\mathbf{t}^c)^\top + H(\mathbf{t}^c))\mathbf{x}^c]\Big) + \text{const} \end{aligned} \tag{10}$$

where ϵ is a small value. Two terms with underlines are merged as the constant term.

In this formulation, the objective higher-order function $E(\mathbf{x})$ can be transformed approximately to lower-order at the local area of $\mathbf{t}$ ($\mathbf{x} \in (\mathbf{t}-\epsilon, \mathbf{t}+\epsilon)$). To minimize (10), we only need to minimize the following equation $E_\mathbf{t}(\mathbf{x})$:

$$\begin{aligned} E_\mathbf{t}(\mathbf{x}) &= E_l(\mathbf{x}) + \sum_c [\nabla\psi_c(\mathbf{t}^c)]^\top\mathbf{x}^c + \\ &\frac{1}{2}[(\mathbf{x}^c)^\top H(\mathbf{t}^c)\mathbf{x}^c - (\mathbf{t}^c)^\top(H(\mathbf{t}^c)^\top + H(\mathbf{t}^c))\mathbf{x}^c] \end{aligned} \tag{11}$$

Namely (11) is the same as (10) without constant terms. Therefore, pairwise binary energy minimizing methods can be adopted to obtain minimum of (11) at the local region of $\mathbf{t}$. Existing pairwise binary energy (even non-submodular energy) minimization methods, such as QPBO and TRWS algorithms, can be employed here. We note (10) is an approximate lower-order energy of the original energy $E(\mathbf{x})$ at $\mathbf{t}$. Instead of minimizing $E(\mathbf{x})$ directly, we propose to iteratively minimizing the following formula:

$$L_\mathbf{t}(\mathbf{x}) = E_\mathbf{t}(\mathbf{x}) + \lambda_\mathbf{t}\mathcal{H}(\mathbf{x}, \mathbf{t}) \tag{12}$$

where $\lambda_\mathbf{t} \in \mathbb{R}$ controls the size of the iteration region indirectly. A larger $\lambda_\mathbf{t}$ denotes a smaller iteration step. $\mathcal{H}(\mathbf{x}, \mathbf{t})$ is the Hamming distance:

$$\mathcal{H}(\mathbf{x}, \mathbf{t}) = \sum_{i=1}^n |x_i - t_i| = \sum_{i=1}^n x_i + t_i - 2x_it_i \tag{13}$$

$L_\mathbf{t}(\mathbf{x})$ is still in the lower-order form and $\mathcal{H}(\mathbf{x}, \mathbf{t})$ does not change the submodularity of $E_\mathbf{t}(\mathbf{x})$. Different from [31], we do not find the minimum of the approximate lower-order function in the trust region. Instead, we control the size of descent step by using Hamming distance as lower-order energy and putting it in $L_\mathbf{t}(\mathbf{x})$. $\mathcal{H}(\mathbf{x}, \mathbf{t})$ will cause penalty, if the solution $\mathbf{x}^*$ after minimizing $L_\mathbf{t}(\mathbf{x})$ is different from $\mathbf{t}$. The larger difference between $\mathbf{x}^*$ and $\mathbf{t}$, the higher penalty $\mathcal{H}(\mathbf{x}, \mathbf{t})$ will cause. And thus the descent step $||\mathbf{x}^* - \mathbf{t}||$ is controlled to become smaller for the penalty of $\mathcal{H}(\mathbf{x}, \mathbf{t})$.

After minimizing (12), we use the new solution of $\arg\min_\mathbf{x} L_\mathbf{t}(\mathbf{x})$ as a new $\mathbf{t}'$, then compute the next minimum of $L_{\mathbf{t}'}(\mathbf{x})$ until it reaches convergence. Inspired by the gradient descent method, the value of $\lambda_\mathbf{t}$ is increasing gradually during iterations, which means the weight of Hamming distance will also become larger. This process can 1) ensure convergence since the search area becomes smaller and 2) avoid solutions in each iteration vibrating around minima (Fig. 2). However, simply increasing $\lambda_\mathbf{t}$ has a drawback: the step in each iteration becomes smaller, which decreases the speed of descent. Considering this issue, similar to the Newton-Raphson method, we adopt the Armijo rule in the Wolfe conditions [1] to make sure that our minimizer decreases the energy sufficiently before convergence. Assume two solutions $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ before and after the k-th iteration. Given a descent direction $\mathbf{p}_k$ (we set it to a unit vector in our experiments), we have $\mathbf{x}_{k+1} = \mathbf{x}_k + \beta_k\mathbf{p}_k$, where β_k is the step length. The Armijo rule is the following inequality:

$$E(\mathbf{x}_{k+1}) \le E(\mathbf{x}_k) + b \cdot \beta_k\mathbf{p}_k^\top\nabla E(\mathbf{x}_k) \tag{14}$$

where $b = 0.1$. If the energies before and after this iteration satisfy the above inequality, it means the energy $E(\mathbf{x}_{k+1})$ has been reduced sufficiently by the step length. Therefore, if the energies before and after one iteration do not satisfy the Armijo rule, our strategy is to reduce $\lambda_\mathbf{t}$ to get a larger step size. Finally, we present our method in Algortihm 1.

In Algorithm 1, the parameter $\lambda_\mathbf{t}$ controls the step size of convergence. When $\lambda_\mathbf{t}$ is increased, the step size becomes smaller. In addition, the step parameter λ determines the speed of convergence. We utilize the QPBO method to minimize $L_\mathbf{t}(\mathbf{x})$. QPBO can only find the approximate minimum when the objective function is non-submodular. However, in our iteration process, it is not necessary to obtain the optimal solution of $L_\mathbf{t}(\mathbf{x})$. In each iteration, our method finds a lower energy of $E(\mathbf{x})$ rather than the minimum energy. Thus, a sub-optimal solution by QPBO in each iteration does not prevent our method from reaching the minimum in the end, as it only affects the convergence rate.

Existing general higher-order reduction methods focus on polynomial energy functions. In addition, many higher-order potentials are in polynomial forms. Optimizing polynomial functions can be viewed as degenerate cases of our optimization method. For illustration, we use an intuitive example that considers a higher-order energy term with three variables:

$$E_h(x, y, z) = \alpha \cdot xyz \tag{15}$$

where $\alpha \in \mathbb{R}$ is a non-zero polynomial coefficient.

Algorithm 1: Higher-order energy minimization method

```
1.  Initialize $\mathbf{t} \leftarrow$arbitrary point, $\lambda_\mathbf{t} \leftarrow \lambda$
2.  Loop
3.      Build $E_\mathbf{t}(\mathbf{x})$ by Expanding $E(\mathbf{x})$ at $\mathbf{t}$ as (11) or (7)
4.      Build $L_\mathbf{t}(\mathbf{x})$ as (12)
5.      $\mathbf{x}^* \leftarrow \arg\min_{x\in\{0,1\}^n} L_\mathbf{t}(\mathbf{x})$
6.      If $E(\mathbf{x}^*) = E(\mathbf{t})$
7.          If $\mathbf{x}^* = \mathbf{t}$
8.              Break
9.          else
10.             Do not update $\mathbf{t}$
11.             Increase $\lambda_\mathbf{t}$: $\lambda_\mathbf{t} \leftarrow \lambda_\mathbf{t} + \lambda$
12.     If $E(\mathbf{x}^*) < E(\mathbf{t})$
13.         If $\mathbf{x}^*$ and $\mathbf{t}$ satisfy the Armijo rule
14.             Increase $\lambda_\mathbf{t}$: $\lambda_\mathbf{t} \leftarrow \lambda_\mathbf{t} + \lambda$
15.         else
16.             Decrease $\lambda_\mathbf{t}$: $\lambda_\mathbf{t} \leftarrow \max(\lambda_\mathbf{t} - \lambda, \lambda)$
17.         $\mathbf{t} \leftarrow \mathbf{x}^*$
18.     If $E(\mathbf{x}^*) > E(\mathbf{t})$
19.         Do not update $\mathbf{t}$
20.         Increase $\lambda_\mathbf{t}$: $\lambda_\mathbf{t} \leftarrow \lambda_\mathbf{t} + \lambda$
21. Loop end
22. Output:$E(\mathbf{x}^*)$ and $\mathbf{x}^*$
```

Then we need to expand it by Taylor expansion at point $\mathbf{t} = \{x_t, y_t, z_t\}$. Following the reduction method discussed above, the approximate lower-order term is:

$$\begin{aligned} E_{h(\mathbf{t})}(x,y,z) &= \alpha \cdot \begin{bmatrix} \alpha y_t z_t \\ \alpha x_t z_t \\ \alpha x_t y_t \end{bmatrix}^\top \begin{bmatrix} x - x_t \\ y - y_t \\ z - z_t \end{bmatrix} + \\ \frac{\alpha}{2} &\begin{bmatrix} x - x_t \\ y - y_t \\ z - z_t \end{bmatrix}^\top \begin{bmatrix} 0 & \alpha z_t & \alpha y_t \\ \alpha z_t & 0 & \alpha x_t \\ \alpha y_t & \alpha x_t & 0 \end{bmatrix} \begin{bmatrix} x - x_t \\ y - y_t \\ z - z_t \end{bmatrix} \\ &= \alpha y_t z_t (x - x_t) + \alpha x_t z_t (y - y_t) + \alpha x_t y_t (z - z_t) + \\ \alpha z_t(x-x_t)(y-y_t) &+ \alpha y_t(x-x_t)(z-z_t) + \alpha x_t(y-y_t)(z-z_t) \end{aligned} \tag{16}$$

where (16) is derived from (7), and it can also follow from (11). Now, we extend this example to the general polynomial higher-order term as follows:

$$\begin{aligned} E_h(x_1, x_2, \ldots, x_n) &= \sum_{S \subset V} (\alpha_u \cdot \prod_{i \in S} x_i) \\ u = \mathbb{B}(S) &= \sum_{j \in S} 2^{j-1} \end{aligned} \tag{17}$$

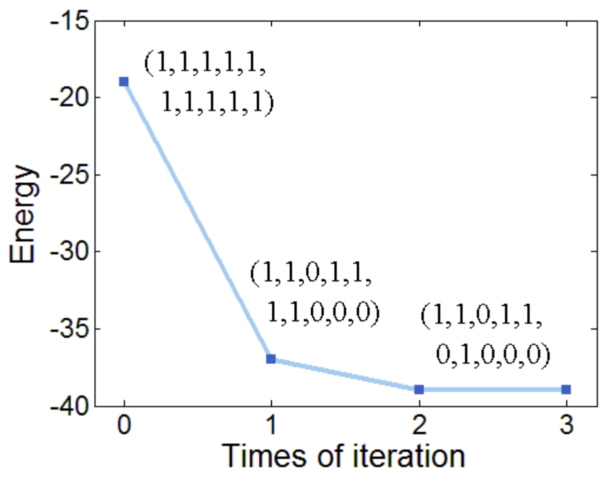


Fig. 3. The convergence curve of the higher-order energy function (25). The solution in each iteration is also shown beside. Only 3 iterations are needed in order to get its optimal solution.

Its approximate lower-order function at $\mathbf{t} = \{t_1, t_2, \ldots, t_n\}$ is:

$$\begin{aligned} &E_{h(\mathbf{t})}(x_1, x_2, \ldots, x_n) = \\ &\sum_{S \subseteq V} \Bigg(\sum_{i \in S, J = S - i} \alpha_u (x_i - t_i) \prod_{j \in J} t_j + \\ &\sum_{i \in S, i' \in S - i, J = S - \{i, i'\}} \alpha_u (x_i - t_i)(x_{i'} - t_{i'}) \prod_{j \in J} t_j \Bigg) \end{aligned} \tag{18}$$

As described in Algorithm 1, $\lambda_\mathbf{t}$ controls the step size and is changed adaptively when the iteration goes on. In each iteration, the descent can be sufficient, which demonstrates that our algorithm can converge efficiently. We set an example to illustrate the convergence of our minimization method. Considering a higher-order function (25) with 10 variables and the 4-th order, here we give its minimum energies of each iteration. The energy curve is shown in Fig. 3, and the minimum value of this energy function converges after three iterations via our optimization method.

B. Higher-order image segmentation

In this section, we demonstrate the effectiveness of the proposed higher-order minimization algorithm by solving a vision problem: image segmentation with appearance entropy. The higher-order function in [35] is employed to describe the image segmentation energy. Adopting our optimization method, we get the segmentation results on two data sets: the Grabcut data set and the BSD data set. And then we compare our performance with Grabcut [10] and pPBC [35] qualitatively and quantitatively.

Some notations for the image segmentation problem are set first. Let $\mathbf{x} = [x_1, x_2, \cdots, x_n]^\top \in \{0,1\}^n$ be the segmentation result of an image I, where n is the number of pixels in I. Variable $x_i = 1$ indicates that pixel s_i belongs to foreground, otherwise, s_i belongs to background. Assume $\mathbf{h^f} \in \mathbb{Z}^m$ and $\mathbf{h^b} \in \mathbb{Z}^m$ are the color histograms of foreground and background, where m is the number of bins of the histograms.

Then we can get an indicating vector $\mathbf{h_i} = [h_{i1}, h_{i2}, \cdots, h_{im}]^\top \in \{0,1\}^m$ for each pixel s_i, where $h_{ik} = 1$, if pixel s_i falls into the k-th bin of the histogram, otherwise, $h_{ik} = 0$. Then we have the following equations:

$$\begin{aligned} \mathbf{h^f} &= \sum_{s_i \in I} \mathbf{h_i} x_i = \mathbf{Hx} \\ \mathbf{h^b} &= \sum_{s_i \in I} \mathbf{h_i}(1 - x_i) = \mathbf{H}(\mathbf{1} - \mathbf{x}) \end{aligned} \tag{19}$$

where $\mathbf{H} = [\mathbf{h_1}, \mathbf{h_2}, \cdots, \mathbf{h_n}]$, and $\mathbf{1} \in \mathbb{R}^n$ is a full vector with value of 1.

Following [10], [35], we obtain the segmentation result by optimizing a higher-order color model fitting energy:

$$\begin{aligned} E(\mathbf{x}, \theta^1, \theta^0) &= E_h(\mathbf{x}) + E_p(\mathbf{x}) \\ &= -\sum_{s_i \in I} \log Pr(s_i|\theta^{x_i}) + |\partial \mathbf{x}| \end{aligned} \tag{20}$$

where E_h and E_p are the higher-order and pairwise term of this energy. θ^1 and θ^0 are the unknown color histograms for the current foreground $\mathbf{x}$ and background $\bar{\mathbf{x}}$, $|\partial \mathbf{x}| = \sum_{\{i,j\} \in \mathrm{N}} p_{ij} |x_i - x_j|$ is a standard pairwise penalty of the segmentation boundary. $E_h(\mathbf{x})$ is a color consistency criterion based on appearance entropy.

The normalized color histograms are used to represent color models in (20) for each pixel s_i, then we have:

$$\begin{aligned} Pr(s_i|\theta^1) &= \mathbf{h_i}^\top \frac{\mathbf{h^f}}{\mathbf{1}^\top \mathbf{x}} = \frac{\mathbf{h_i}^\top \mathbf{Hx}}{\mathbf{1}^\top \mathbf{x}} \\ Pr(s_i|\theta^0) &= \mathbf{h_i}^\top \frac{\mathbf{h^b}}{\mathbf{1}^\top (\mathbf{1} - \mathbf{x})} = \frac{\mathbf{h_i}^\top \mathbf{H}(\mathbf{1} - \mathbf{x})}{\mathbf{1}^\top (\mathbf{1} - \mathbf{x})} \end{aligned} \tag{21}$$

where $\frac{\mathbf{h^f}}{\mathbf{1}^\top \mathbf{x}}$ and $\frac{\mathbf{h^b}}{\mathbf{1}^\top (\mathbf{1}-\mathbf{x})}$ are the normalized foreground and background histograms.

Then the higher-order energy $E_h(\mathbf{x})$ in (20) can be transformed as follows:

$$\begin{aligned} E_h(\mathbf{x}) &= -\sum_{s_i \in I} x_i \log Pr(s_i|\theta^1) + (1 - x_i) \log Pr(s_i|\theta^0) \\ &= -\sum_{s_i \in I} x_i \log \frac{\mathbf{h_i}^\top \mathbf{Hx}}{\mathbf{1}^\top \mathbf{x}} + (1 - x_i) \log \frac{\mathbf{h_i}^\top \mathbf{H}(\mathbf{1} - \mathbf{x})}{\mathbf{1}^\top (\mathbf{1} - \mathbf{x})} \end{aligned} \tag{22}$$

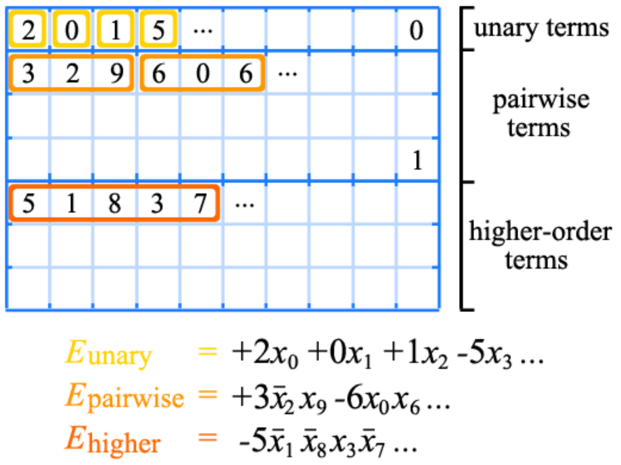
We approximate $E_h(\mathbf{x})$ at a point $\mathbf{t}$ using the first-order Taylor approximation, therefore the approximate lower-order function is: $U_0(\mathbf{x}) = \nabla E_h(\mathbf{t})^\top (\mathbf{x} - \mathbf{t}) + o$. Compared with the first-order expansion, the computational cost of second-order Taylor expansion will increase a lot, since each pixel has one variable and the number of higher-order cliques is also large. For computational simplicity, we use the first order expansion to keep a balance between accuracy and efficiency for image segmentation. Each component of $\nabla E_h(\mathbf{t})$ is:

$$\begin{aligned} \frac{\partial E_h(\mathbf{t})}{\partial x_i} &= \log \mathbf{1}^\top \mathbf{t} - \log \mathbf{h_i}^\top \mathbf{Ht} \\ &\quad - \log \mathbf{1}^\top (\mathbf{1} - \mathbf{t}) + \log \mathbf{h_i}^\top \mathbf{H}(\mathbf{1} - \mathbf{t}) \end{aligned} \tag{23}$$

Details of derivation are provided in *Appendix I*.

With (23), the lower-order expansion (11) for this segmentation problem can be rewritten as:

$$E_{\mathbf{t}}(\mathbf{x}) = E_p(\mathbf{x}) + U_0(\mathbf{x}) \tag{24}$$



2 0 1 5 ··· 0
3 2 9 6 0 6 ···
1
5 1 8 3 7 ···
unary terms
pairwise terms
higher-order terms
$E_{\text{unary}} = +2x_0 + 0x_1 + 1x_2 - 5x_3 \ldots$
$E_{\text{pairwise}} = +3\bar{x}_2 x_9 - 6x_0 x_6 \ldots$
$E_{\text{higher}} = -5\bar{x}_1 \bar{x}_8 x_3 \bar{x}_7 \ldots$

Fig. 4. The first data block B and its energy $E_{\text{mnist}(1)}$ with parameters from the MNIST dataset. Values from small data sets $B[1 \ldots 10]$, $B[11 \ldots 40]$ and $B[41 \ldots 70]$ build the unary, pairwise and higher-order terms.

Then, we can use Algorithm 1 to compute the solution. The proposed order reduction process in Algorithm 1 can use arbitrary initial value to set $\mathbf{t}$. However, when it comes to a specific problem in vision application, a meaningful value depending on the problem would lead to a better result. Thus we initialize $\mathbf{t}$ by a bounding box from user interaction for image segmentation. We will give more discussion about initial $\mathbf{t}$ and segmentation performance in Section *IV*.

III. Experimental Results

In this section, we present two different experiments to show the effectiveness of our minimization method: (a) minimization of 1000 higher-order synthetic functions, and (b) higher-order image segmentation. We compare with two state-of-the-art order reduction methods in this section: HOCR [19] in the first experiment and pPBC [35] in two experiments.

A. Synthetic data experiment

In the first experiment, we build 1000 polynomial functions $E_{\text{mnist}(i)}, i \in [1, 1000]$, whose parameters are taken from the MNIST dataset (http://yann.lecun.com/exdb/mnist). This dataset is originally used for character recognition and contains 70000 handwriting digits and their features. We used those digits (from 0 to 9) to build higher-order functions with 10 variables. By minimizing these synthetic functions, we compare our method with HOCR. Now we introduce how one higher-order function is built. We divide 70K numerical digits into 1000 data blocks, where 70 digits in each block are used to build one function. The details of building one function from one block B are shown in Fig. 4. The first 10 digits $B[1 \cdots 10]$ in block B are used as the coefficients of unary terms (vector U in (5)). Values in $B[11 \cdots 30]$ are extracted for pairwise terms, which build the matrix P in (5). In $B[11 \cdots 30]$, every 3-digits set one pairwise term: the first digit in the triple as coefficient and the last two as subscripts of variables. The higher-order terms are built from the last 30 digits $B[41 \cdots 70]$ in the block. The first digit in the 5-tuple is defined as a higher-order coefficient (such as α_S in (17)), and

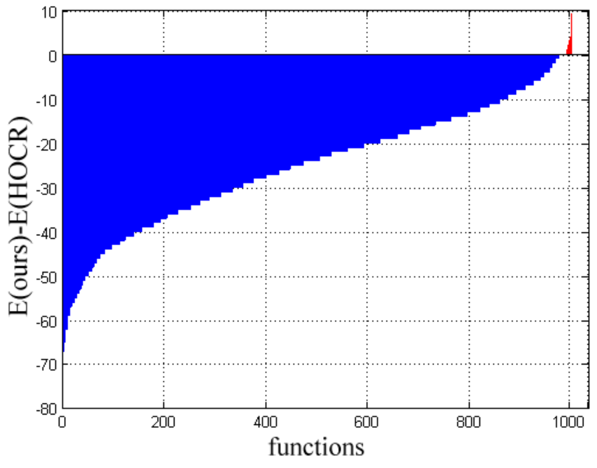


Fig. 5. The statistics of energy difference of minimum energies from 1000 energy functions between HOCR [19] and our method. The blue region is much larger than the red region, which shows that our method achieves more accurate results than HOCR.

the last four digits are denoted as subscripts. For each digit $B[i]$ in the block, its preceding digit $B[i-1]$ sets its symbol as follows:

i) $B[i]$ is used as a coefficient:
If $B[i-1]$ is even, the coefficient will be $B[i]$;
If $B[i-1]$ is odd, the coefficient will be $-B[i]$.

ii) $B[i]$ is used as a subscript:
If $B[i-1]$ is even, the multiplier will be $x_{B[i]}$;
If $B[i-1]$ is odd, the multiplier will be $\overline{x}_{B[i]}$.

Fig. 4 gives an example of building a higher-order energy function with 10 variables and 4 orders from the first block of the MNIST dataset. And this function is :

$$\begin{aligned}E_{\text{mnist}(1)}(x_0, x_1, \ldots, x_9) &= 2x_0 - 0x_1 + 1x_2 - 5x_3 - 9x_4\\ &-1x_5 - 4x_6 + 4x_7 + 7x_8 - 0x_9 - 3\overline{x}_2x_9 - 6x_0x_6\\ &+9\overline{x}_0x_8 + 9\overline{x}_9\overline{x}_1 - 4x_8x_6 + 7\overline{x}_6x_3 - 0x_2x_7 - 2x_3\overline{x}_1\\ &-0x_4x_5 - 8x_4x_1 - 5\overline{x}_1\overline{x}_8x_3\overline{x}_7 - 3\overline{x}_9\overline{x}_5\overline{x}_8x_1\\ &-6x_6x_1\overline{x}_7\overline{x}_8 + 2x_9\overline{x}_6x_7\overline{x}_0 + 4x_4x_2x_0\overline{x}_3 - 5\overline{x}_6x_3\overline{x}_4x_5\end{aligned} \tag{25}$$

In these 1000 functions, the average number of iterations before convergence is 4.5, which proves the fast convergence of our method. As mentioned before, the convergence curve of the function (25) extracted from the first block is already shown in Fig. 3. The task of order reduction is to transform a higher-order function to a lower-order one, but it does not promise that the lower-order function is submodular. Then we use partial optimization methods, such as QPBO, to get an approximate solution. When QPBO cannot decide whether the value of one variable in the energy function is '0' or '1', it labels this variable as '-1', and the solution with '-1' is called partial solution. This situation happens when the lower-order function is non-submodular, and we demonstrate the partial solution of (25). The solutions of the energy (25) are separately obtained by HOCR and the proposed method as follows:

$$\begin{aligned}\text{HOCR} &: (u, u, u, u, u, u, u, u, u, u), u = -1\\ \text{Ours} &: (1, 1, 0, 1, 1, 0, 1, 0, 0, 0)\end{aligned} \tag{26}$$

Both HOCR and our method adopt QPBO as the lower-order optimizer. However, our method generates fewer unlabeled solutions than HOCR. In fact, among all 1000 functions, the percentage of unlabeled variables of HOCR and the proposed method are 92.46% and 12.46%.

In the next comparison, when one variable gets the label '-1', we set its solution to '0'. We count the minimum energies of 1000 functions by these two methods. As shown in Fig. 5, we plot the energy differences of all these 1000 functions from both methods. For instance, we use HOCR [19] and the proposed method to obtain the minimum energies of one certain testing function $E_{\text{mnist}(i)}$ and get the solution $\mathbf{x}_h^*$ and $\mathbf{x}_o^*$. The vertical axis in Fig. 5 (a) denotes the value of $E_{\text{mnist}(i)}(\mathbf{x}_o^*) - E_{\text{mnist}(i)}(\mathbf{x}_h^*)$. When this value is negative, the energy difference is shown in blue; when this difference is larger than 0, the energy difference is denoted by red. Intuitively, the blue region is much larger than the red region, which implies that our method can get lower energies than HOCR. Our method gets smaller energies than HOCR in 974 functions. However, this illustration does not mean that HOCR obtains incorrect solutions. Both of these two order reduction methods get partial solutions with label '-1'. According to the formula derivation of (18), our method can preserve the submodularity of the objective polynomial function, which will be discussed in Section *IV*. Therefore, our method produces fewer unlabeled solutions than HOCR. As a result, the final solutions of HOCR are less accurate than our method, when we simply change the label '-1' to '0'.

B. Image segmentation

Now we demonstrate the viability of our higher-order optimization algorithm on the practical higher-order energies by segmenting images from the Grabcut dataset and the BSD dataset. The GrabCut dataset includes 50 common size images with user interaction (bounding boxes) and ground truth. We run our method on this dataset and compare the results with the other two well-known methods: Grabcut [10] and pPBC [35]. We use error rate to measure the qualities of results from all these three methods, which is the percentage of the misclassified pixels. As shown in Fig. 6, our method achieves better performance than both Grabcut and pPBC, where our segmentation results have smaller error rates. Grabcut minimizes (20) via the block coordinate descent optimization strategy, and it fixes θ^1 and θ^0 with an initial solution to get a new solution of $\mathbf{x}$ with a lower energy. Then it uses this new solution to update the foreground and background histograms θ^1 and θ^0. pPBC [35] builds the parametric pseudo-bound in each iteration and picks out the best parameter through another optimization process, so it runs more slowly than the proposed method. As for the quantitative comparisons, the average error rates of Grabcut, pPBC and the proposed method are 10.0708%, 9.3468% and 7.9254%. The computational times of pPBC and the proposed method are 11.0290 sec and 1.1628 sec. Grabcut costs 0.9325 sec in average. As discussed before, Grabcut uses the block coordinate descent optimization strategy, which is not an actual higher-order reduction process, therefore it costs slightly less than our method.

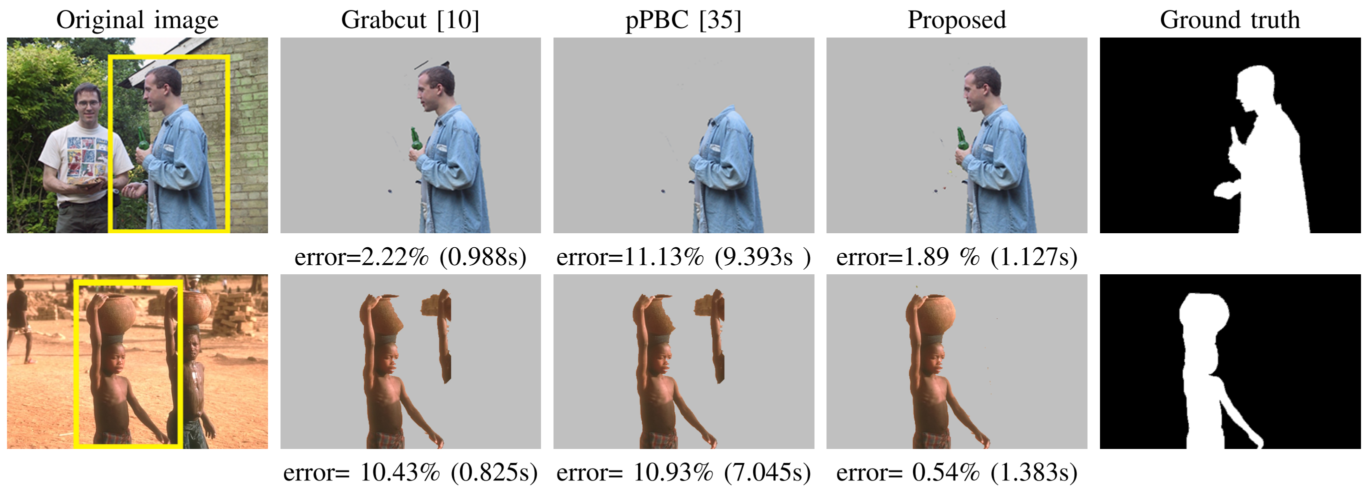


Fig. 6. Comparison of interactive segmentation with Grabcut [10], pPBC [35], and the proposed method using the same user interaction (yellow bounding box). The respective error rates and run time are shown below each image.

In order to further demonstrate the effectiveness of the proposed method, we apply these three segmentation methods on the more challenging BSD dataset. The BSD dataset includes images with a size of 841×321 and ground truth, where 100 images containing foreground objects are randomly selected for testing. We manually add boxes on each of them as user interactions, which indicate the region of foreground objects roughly. We run the optimization procedure with our minimization method on these 100 images, and compare the results with Grabcut and pPBC. The average error rates of Grabcut, pPBC and the proposed method are 24.20%, 18.83% and 15.51% on the BSD data set. And the standard deviation of their error rates are 17.6% (Grabcut), 16.1% (pPBC) and 12.6% (ours). Combining the results on the GrabCut data set, we can conclude that our method gets better segmentation performance than both Grabcut and pPBC. The average number of iterations for our segmentation method for this 100 images is about 5. The average run time for each image of Grabcut, pPBC and the proposed method are 0.63, 8.71 and 0.87 sec. Our method obtains better segmentation performance than both Grabcut and pPBC, and the proposed method yields about **10 times speedup** than the pPBC [35].

IV. More Discussions and Limitations

Now we will discuss the local minima and its relations with initial $\mathbf{t}$. Our optimization framework is similar to the Newton-Raphson method and the gradient descent method. The latter two methods find the gradient at local area in each iteration to obtain a smaller energy. The proposed method builds a lower-order energy each time and continues the search in the direction of decreasing energy until the convergence criteria is satisfied. The main limitation of the proposed algorithm is that this framework can not determine whether the current solution is the global minima when the continuous version of the higher-order energy is non-convex in theory.

As shown in Fig. 7, the red curve is the higher-order energy $E(\mathbf{x})$. We set two different values $\mathbf{t}_1$ and $\mathbf{t}_2$ as initial $\mathbf{t}$. Starting with $\mathbf{t}_1$, the minimizer gets convergence at $\mathbf{x}_1^*$. The minimizer starts at $\mathbf{t}_2$ ends up at $\mathbf{x}_2^*$. And when it converges

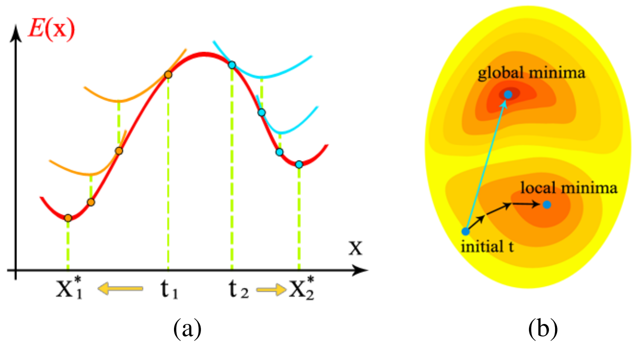


Fig. 7. Illustration of local minima. In (a), $\mathbf{t}_1$ and $\mathbf{t}_2$ are two initial $\mathbf{t}$ for Algorithm 1. After iterations, these two minimizers get convergence at $\mathbf{x}_1^*$ (global minima) and $\mathbf{x}_2^*$ (local minima). Blue dots are solutions in each iteration beginning at $\mathbf{t}_2$, and they are getting closer to $\mathbf{x}_2^*$. These solutions are moving away from the global minima. (b) shows the local minima of gradient descent. The blue arrow points to global minima, which is different from the actual direction of descent (black arrows). Here $E(\mathbf{x}_1^*) < E(\mathbf{x}_2^*)$.

at $\mathbf{x}_2^*$, it is difficult to jump out of this local minima. For the minimizer starting with $\mathbf{t}_2$, it moves away from the optimal solution $\mathbf{x}_1^*$ in each iteration. The phenomenon that minimizer is not getting closer to global minima also happens in gradient descent algorithm, which is shown in Fig. 7 (b). Generally, it would be a useful technique to initialize $\mathbf{t}$ by choosing multiple values to get a lower energy. However, in our synthetic data experiment, we set the initial $\mathbf{t}$ to '0' vector. In the experiment of image segmentation, the initial $\mathbf{t}$ is defined by the bounding box. We only use one value to initialize vector $\mathbf{t}$ for simplicity. In addition, we have made two efforts to jump out of the local minimum at the earlier stage of descent in our implementation. First, the step size is large with a small λ, which can avoid getting stuck in local minimum to a large extent. Second, we adopt Armijo rule in the Wolfe conditions to make sure each descent is sufficient enough.

The step parameter λ has influence on the convergence rate. Fig. 8 gives the example of energy curves about the energy descent with different values of λs. The smaller λ we set, the faster this energy converges, and also the lower energy it

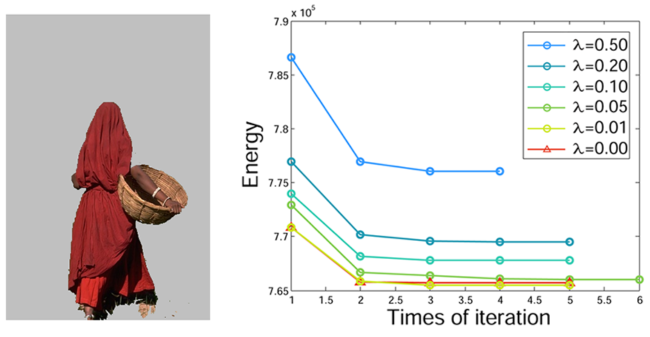


Fig. 8. Comparison of energy curves with different values of λ for the image '80090'. When $\lambda \geq 0.01$, the smaller λ gets, the faster the energy converges and the lower energy it achieves.

reaches. That is because a small λ leads to a small $\lambda_\mathbf{t}$, which represents a large step range in each iteration. Thus, the speed of descent becomes faster. In addition, when the step range is larger, the proposed method has a higher probability to jump out of local minima. Therefore, it can reach a lower energy. However, our experiments show that, when $\lambda < 0.01$, it will not converge faster than the one with $\lambda = 0.01$ (see the red line in Fig. 8). In addition, as mentioned before, if $\lambda = 0$, the minimizer may be vibrating around the minima (Fig. 2). Among the 1000 energies of our synthetic experiment, there are 426 energy functions with vibrations if we set $\lambda = 0$. In these 426 functions, the vibration may cause non-convergence. As a result, we choose $\lambda = 0.01$ in our experiments.

Now we discuss the submodularity of our optimization method. A set function $f : 2^V \rightarrow \mathbb{R}$ is called submodular [12] if it satisfies:

$$f(S) + f(T) \geq f(S \cap T) + f(S \cup T), \forall S, T \subseteq V \qquad (27)$$

We denote the marginal gain in the subset of S by $f(i|S) = f(\{i\} \cup S) - f(S)$. Submodular functions satisfy the property of diminishing gains: $f(i|S) \geq f(i|T)$ for all $S \subseteq T \subseteq V$. Based on our algorithm, we will set a conclusion:

Property: If the higher-order terms of the objective function are submodular, the lower-order terms they are transformed to in our minimization method are also submodular. Our method can preserve submodularity. If a higher-order term $\psi_c(\mathbf{x}^c)$ in (6) is submodular, its lower-order terms in (7) are also submodular. The proof is given in *Appendix II*.

V. Conclusion

We have presented a novel minimization approach for general higher-order binary energy functions. The proposed method adopts first or second-order Taylor expansion to transform a higher-order energy function to a lower-order one at a local region, and then the lower-order function is minimized to get a new solution with smaller energy. Our method obtains the satisfactory solution of the original higher-order function by iteratively optimizing the lower-order function. In the experiments, we solve the image segmentation problem by optimizing the higher-order energy with appearance entropy via our minimization method. The comparison results show that our method achieves better performance than state-of-the-art order reduction methods. And the convergence rate of our method is faster than the previous approaches. In the future work, we expect to explore the application of image denoising by our higher-order reduction method using the Field of Experts (FoE) model as [19] and [33]. Furthermore, we are working on finding more higher-order priors for complex nature scene, which is not in polynomial form. Then we can utilize the advantage of the proposed energy minimization method to solve more computer vision problems.

Appendix I: The Derivation of the Higher-Order Term $E_h(\mathbf{x})$ for Segmentation

As shown in (23) in Section *II*, $E_h(\mathbf{x})$ is built as follows:

$$\begin{aligned} E_h(\mathbf{x}) &= -\sum_{s_i \in I} x_i \log \frac{\mathbf{h_i^T H x}}{\mathbf{1^T x}} + (1 - x_i) \log \frac{\mathbf{h_i^T H (1 - x)}}{\mathbf{1^T (1 - x)}} \\ &= -\sum_{s_i \in I} x_i \log \mathbf{h_i^T H x} + \sum_{s_i \in I} x_i \log \mathbf{1^T x} \\ &\quad - \sum_{s_i \in I} (1 - x_i) \log \mathbf{h_i^T H (1 - x)} \\ &\quad + \sum_{s_i \in I} (1 - x_i) \log \mathbf{1^T (1 - x)} \\ &= A(\mathbf{x}) + B(\mathbf{x}) + C(\mathbf{x}) + D(\mathbf{x}) \end{aligned} \qquad (28)$$

where the bottom number of log is 2. In (28), we split this higher-order term into four parts: $A(\mathbf{x})$, $B(\mathbf{x})$, $C(\mathbf{x})$ and $D(\mathbf{x})$. The process of obtaining the partial derivative of $E_h(\mathbf{x})$ equals to get the partial derivatives of the above four components respectively.

We give some notations for simplifying $\mathbf{h_i^T H x}$. First we define $\mathbf{e_i} \in \mathbb{R}^n$ as an indicating vector. Especially, the $i - th$ element of $\mathbf{e_i}$ equals to 1 and the others equal to 0. Then we define a index set $fallin(i) = \{j | \text{the pixel } s_j \in I \text{ falls in the histogram bin which } s_i \text{ contributes to}\}$.

According to the definition of $\mathbf{h_i}$, it can be found that:

$$\begin{aligned} &\text{If} \quad j \in fallin(i), \text{then} \quad \mathbf{h_i} = \mathbf{h_j}, \mathbf{h_i^T h_j} = 1 \\ &\text{If} \quad j \notin fallin(i), \text{then} \quad \mathbf{h_i^T h_j} = 0 \end{aligned} \qquad (29)$$

As a result, we can rewrite $\mathbf{h_i^T H x}$ as follows:

$$\begin{aligned} \mathbf{h_i^T H x} &= \mathbf{h_i^T (h_1, h_2, \cdots, h_n) x} \\ &= (\mathbf{h_i^T h_1}, \mathbf{h_i^T h_2}, \cdots, \mathbf{h_i^T h_n}) \mathbf{x} \\ &= \left(\sum_{j \in fallin(i)} \mathbf{e_j^T} \right) \mathbf{x} = \sum_{j \in fallin(i)} x_j \end{aligned} \qquad (30)$$

For $k \notin fallin(i)$, $\mathbf{h_k^T H x} = \sum_{j \in fallin(k)} x_j$ does not include x_i. Then we can further split $A(\mathbf{x})$ to two components for each x_i:

$$A(\mathbf{x}) = -\sum_{j \in fallin(i)} x_j \log \mathbf{h_j^T H x} - \sum_{j \notin fallin(i)} x_j \log \mathbf{h_j^T H x} \quad (31)$$

Since the second component of $A(\mathbf{x})$ in (31) does not consist of x_i, we only need to consider the first component to get the partial derivative on x_i. Finally, the derivative of $A(\mathbf{x})$ is defined as follows:

$$
\begin{aligned}
\frac{\partial A(\mathbf{x})}{\partial x_i} &= -\frac{x_i \mathbf{h_i^T H e_i}}{\mathbf{h_i^T H x} \ln 2} - \log \mathbf{h_i^T H x} - \sum_{j \neq i, j \in fallin(i)} \frac{x_j \mathbf{h_j^T H e_j}}{\mathbf{h_j^T H x} \ln 2} \\
&= -\log \mathbf{h_i^T H x} - \sum_{j \in fallin(i)} \frac{x_j \mathbf{h_j^T H e_j}}{\mathbf{h_j^T H x} \ln 2} \\
&= -\log \mathbf{h_i^T H x} - \sum_{j \in fallin(i)} \frac{x_j}{\left(\sum\limits_{k \in fallin(i)} x_k \right) \ln 2} \\
&= -\log \mathbf{h_i^T H x} - \frac{\sum\limits_{j \in fallin(i)} x_j}{\left(\sum\limits_{k \in fallin(i)} x_k \right) \ln 2} \\
&= -\log \mathbf{h_i^T H x} - \frac{1}{\ln 2}
\end{aligned} \tag{32}
$$

Similarly, we can obtain the derivative of $B(\mathbf{x})$:

$$
\begin{aligned}
\frac{\partial B(\mathbf{x})}{\partial x_i} &= \frac{x_i}{\mathbf{1^T x} \ln 2} + \log \mathbf{1^T x} + \sum_{j \neq i} \frac{x_j}{\mathbf{1^T x} \ln 2} \\
&= \log \mathbf{1^T x} + \sum_{j} \frac{x_j}{\mathbf{1^T x} \ln 2} = \log \mathbf{1^T x} + \frac{1}{\ln 2}
\end{aligned} \tag{33}
$$

By defining $\mathbf{y} = \mathbf{1} - \mathbf{x}$, we then obtain $C(\mathbf{x}) = A(\mathbf{y})$, $D(\mathbf{x}) = B(\mathbf{y})$. It is easy to find the derivative of $C(\mathbf{x})$:

$$
\begin{aligned}
\frac{\partial C(\mathbf{x})}{\partial x_i} &= \frac{\partial A(\mathbf{y})}{\partial y_i} \frac{\partial y_i}{\partial x_i} = -\frac{\partial A(\mathbf{y})}{\partial y_i} \\
&= \log \mathbf{h_i^T H (1 - x)} + \frac{1}{\ln 2}
\end{aligned} \tag{34}
$$

And the derivative of $D(\mathbf{x})$ becomes:

$$
\begin{aligned}
\frac{\partial D(\mathbf{x})}{\partial x_i} &= \frac{\partial B(\mathbf{y})}{\partial y_i} \frac{\partial y_i}{\partial x_i} = -\frac{\partial B(\mathbf{y})}{\partial y_i} \\
&= -\log \mathbf{1^T (1 - x)} - \frac{1}{\ln 2}
\end{aligned} \tag{35}
$$

Adding each terms from (32) (33) (34) (35), we can get the derivative of $E_h(\mathbf{x})$:

$$
\begin{aligned}
\frac{\partial E_h(\mathbf{x})}{\partial x_i} &= \log \mathbf{1^T x} - \log \mathbf{h_i^T H x} \\
&\quad - \log \mathbf{1^T (1 - x)} + \log \mathbf{h_i^T H (1 - x)}
\end{aligned} \tag{36}
$$

APPENDIX II: THE PROOF OF THE PROPERTY IN SECTION *IV*

Property: If the higher-order terms of the objective function are submodular, the lower-order terms they transform to in our minimization method are also submodular. Specifically, if a higher-order term $\psi_c(\mathbf{x}^c)$ in (6) is submodular, its lower-order terms with the form of (7) are also submodular.

Proof: According to the property of diminishing gains of a submodular function, if a discrete function $f(x)$ is submodular, then $\forall i$, the discrete derivative $\partial f(x) = f(x + e_i) - f(x)$ is non-increasing in x, and $\frac{\partial^2 \widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_i^c \partial x_j^c}$ in (7) is non-positive. We consider one pairwise term in (7):

$$
P^c(x_i^c, x_j^c) = \frac{1}{2!} \sum_{i,j=1}^{|c|} \frac{\partial^2 \widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_i^c \partial x_j^c} (x_i^c - t_i)(x_j^c - t_j) \tag{37}
$$

Then we have:

$$
\begin{aligned}
&P^c(0,1) + P^c(1,0) - P^c(1,1) - P^c(0,0) \\
&= -\frac{1}{2} \frac{\partial^2 \widetilde{\psi}_c(\mathbf{t}^c)}{\partial x_i^c \partial x_j^c} \geq 0
\end{aligned} \tag{38}
$$

Therefore, the lower-order functions in (7) are submodular, since unary terms are always submodular.

REFERENCES

[1] P. Wolfe, "Convergence conditions for ascent methods," *SIAM Review*, vol. 11, no. 2, pp. pp. 226-235, 1969.

[2] S. Kirkpatrick and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[3] X. Yang, X. Gao, D. Tao, X. Li, "Improving level set method for fast auroral oval segmentation," *IEEE Trans. Image Processing*, vol. 23, no. 7, pp. 2854-2865, 2014.

[4] X. Liu, M. Song, D. Tao, J. Bu, C. Chen, "Random geometric prior forest for multiclass object segmentation," *IEEE Trans. Image Processing*, vol. 24, no. 10, pp. 3060-3070, 2015.

[5] J. Shen, Y. Du, W. Wang and X. Li, "Lazy random walks for superpixel segmentation", *IEEE Trans. on Image Processing*, vol. 23, no. 4, pp. 1451-1462, 2014.

[6] P. L. Hammer, P. Hansen and B. Simeone, "Roof duality, complementation and persistency in quadratic 0-1 optimization," *Mathematical programming*, vol. 28, no. 2, pp. 121-155, 1984.

[7] J. Shen, Y. Du, and X. Li, "Interactive segmentation using constrained Laplacian optimization," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 24, no. 7, pp. 1088-1100, 2014.

[8] J. Peng, J. Shen, and X. Li, "High-order energies for stereo segmentation," *IEEE Trans. on Cybernetics*, vol. 46, no. 7, pp. 1616-1627, 2016.

[9] V. Kolmogorov, and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147-159, 2004.

[10] C. Rother, V. Kolmogorov and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. on Graphics*, vol. 23, no. 3, pp. 309-314, 2004.

[11] T. Meltzer, C. Yanover and Y. Weiss, "Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation," *In Proceedings of IEEE ICCV*, pp. 428-435, 2005.

[12] S. Fujishige, "Submodular functions and optimization," *Elsevier*, 2005.

[13] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568-1583, 2006.

[14] L. Grady, "Random walks for image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768-1783, 2006.

[15] A. K. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm," *In Proceedings of IEEE ICCV*, pp. 1-8, 2007.

[16] C. Rother, V. Kolmogorov, V. Lempitsky and M. Szummer, "Optimizing binary MRFs via extended roof duality," *In Proceedings of IEEE CVPR*, pp. 1-8, 2007.

[17] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr, "Exact inference in multi-label CRFs with higher order cliques", *In Proceedings of IEEE CVPR*, pp. 1-8, 2008.

[18] P. Kohli, L. Ladicky, and P. Torr, "Robust higher order potentials for enforcing label consistency," International Journal of Computer Vision, vol. 82, no. 3, pp. 302-324, 2009.

[19] H. Ishikawa, "Higher-order clique reduction in binary graph cut," *In Proceedings of IEEE CVPR*, pp. 2993-3000, 2009.

[20] C. Rother, P. Kohli, W. Feng and J. Jia, "Minimizing sparse higher order energy functions of discrete variables," *In Proceedings of IEEE CVPR*, pp. 1382-1389, 2009.

[21] S. Vicente, V. Kolmogorov, and C. Rother, "Joint optimization of segmentation and appearance models," *In Proceedings of IEEE ICCV*, pp. 755-762, 2009.

[22] P. Stobbe, A. Krause, "Efficient minimization of decomposable submodular functions," *In Proceedings of NIPS*, pp. 2208-2216, 2010.

[23] F. Kahl and P. Strandmark, "Generalized roof duality for pseudo-boolean optimization," *In Proceedings of IEEE ICCV*, pp. 255-262, 2011.

[24] S. Jegelka and J. Bilmes, "Submodularity beyond submodular energies: coupling edges in graph cuts," *In Proceedings of IEEE CVPR*, pp. 1897-1904, 2011.

[25] A. C. Gallagher, D. Batra and D. Parikh, "Inference for order reduction in Markov random fields," *In Proceedings of IEEE CVPR*, pp. 1857-1864, 2011.

[26] A. Fix, A. Gruber, E. Boros, and R. Zabih, "A graph cut algorithm for higher-order Markov Random Fields," *In Proceedings of IEEE ICCV*, pp. 1020-1027, 2011.

[27] A. Blake, P. Kohli and C. Rother, "Markov random fields for vision and image processing," *Mit Press*, 2011.

[28] A. Delong, O. Veksler, A. Osokin and Y. Boykov, "Minimizing sparse high-order energies by submodular vertex-cover," *In Proceedings of NIPS*, pp. 962-970, 2012.

[29] P. Kohli, A. Osokin and S. Jegelka, "A principled deep random field model for image segmentation," *In Proceedings of IEEE CVPR*, pp. 1971-1978, 2013.

[30] I. B. Ayed, L. Gorelick and Y. Boykov, "Auxiliary cuts for general classes of higher order functionals," *In Proceedings of IEEE CVPR*, pp. 1304-1311, 2013.

[31] L. Gorelick , F. Schmidt and Y. Boykov, "Fast Trust Region for Segmentation," *In Proceedings of IEEE CVPR*, pp. 1714-1721, 2013.

[32] W. Wang, J. Shen, and F. Porikli, "Saliency-aware geodesic video object segmentation," *In Proceedings of IEEE CVPR*, pp. 3395–3402, 2015.

[33] H. Ishikawa, "Higher-order clique reduction without auxiliary variables," *In Proceedings of IEEE CVPR*, pp. 1362-1369, 2014.

[34] A. Fix, W. Chen and R. Zabih, "A Primal-Dual Algorithm for Higher-Order Multilabel Markov Random Fields," *In Proceedings of IEEE CVPR*, pp. 1138-1145, 2014.

[35] M. Tang, I. B. Ayed and Y. Boykov, "Pseudo-bound optimization for binary energies," *In Proceedings of ECCV*, pp. 691-707, 2014.

[36] W. Wang, J. Shen, X. Li, F. Porikli, "Robust video object co-segmentation," *IEEE Trans. on Image Processing*, vol. 24, no. 10, pp. 3137-3148, 2015.

[37] L. Gorelick, Y. Boykov, O. Veksler, I. B. Ayed, and A. Delong, "Submodularization for binary pairwise energies," *In Proceedings of IEEE CVPR*, pp. 1154-1161, 2014.

[38] C. Arora, S. Banerjee, P. Kalra and S. Maheshwari, "Fast Approximate Inference in Higher Order MRF-MAP Labeling Problems," *In Proceedings of IEEE CVPR*, pp. 1338-1345, 2014.

[39] B. Ma, J. Shen, Y. Liu, H. Hu, L. Shao, and X. Li, "Visual tracking using strong classifier and structural local sparse descriptors," *IEEE Trans. on Multimedia*, vol. 17, no. 10, pp. 1818–1828, 2015.

[40] V. Kolmogorov, "A new look at reweighted message passing," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 919-930, 2015.

[41] W. Wang, J. Shen, L. Shao, and F. Porikli, "Correspondence driven saliency transfer," *IEEE Trans. on Image Processing*, vol. 25, no. 11, pp. 5025-5034, 2016.

[42] B. Ma, H. Hu, J. Shen, Y. Liu, and L. Shao, "Generalized pooling for robust object tracking," *IEEE Trans. on Image Processing*, vol. 25, no. 9, pp. 4199–4208, 2016.

[43] D. Zhang, J. Han, and L. Shao, "Co-saliency Detection Based on Intrasaliency Prior Transfer and Deep Intersaliency Mining," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 6, pp. 1163-1176, 2016.

[44] W. Wang and J. Shen, Y. Yu, and K-L. Ma, "Stereoscopic thumbnail creation via efficient stereo saliency detection," *IEEE Trans. on Visualization and Computer Graphics*, vol. 23, no. 8, 2017.

[45] B. Ma, L. Huang, J. Shen, L. Shao, M-H. Yang, and F. Porikli, "Visual tracking under motion blur," *IEEE Trans. on Image Processing*, vol. 25, no. 12, pp. 5867-5876, 2016.

[46] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu, and H. Huang, "Occlusion-aware real-time object tracking," *IEEE Trans. on Multimedia*, vol. 19, no. 4, pp. 763-771, 2017.

[47] W. Wang, J. Shen, R. Yang, and F. Porikli, "Saliency-aware video object segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, in press, doi://10.1109/TPAMI.2017.2662005, 2017.

[48] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of Co-salient Objects by Looking Deep and Wide," *International Journal of Computer Vision*, vol. 120, no. 2, pp. 215-232, 2016.