Efficient Relaxations for Dense CRFs with Sparse Higher-Order Potentials[*]

Thomas Joy[†], Alban Desmaison[†], Thalaiyasingam Ajanthan[†], Rudy Bunel[†],
Mathieu Salzmann[‡], Pushmeet Kohli[§], Philip H. S. Torr[†], and M. Pawan Kumar[†]

Abstract. Dense conditional random fields (CRFs) have become a popular framework for modeling several problems in computer vision such as stereo correspondence and multiclass semantic segmentation. By modeling long-range interactions, dense CRFs provide a labeling that captures finer detail than their sparse counterparts. Currently, the state-of-the-art algorithm performs mean-field inference using a filter-based method but fails to provide a strong theoretical guarantee on the quality of the solution. A question naturally arises as to whether it is possible to obtain a maximum a posteriori (MAP) estimate of a dense CRF using a principled method. Within this paper, we show that this is indeed possible. Specifically, we will show that, by using a filter-based method, continuous relaxations of the MAP problem can be optimized efficiently using state-of-the-art algorithms. Specifically, we will solve a quadratic programming relaxation using the Frank–Wolfe algorithm and a linear programming relaxation by developing a proximal minimization framework. By exploiting labeling consistency in the higher-order potentials and utilizing the filter-based method, we are able to formulate the above algorithms such that each iteration has a complexity linear in the number of classes and random variables. The presented algorithms can be applied to any labeling problem using a dense CRF with sparse higher-order potentials. In this paper, we use semantic segmentation as an example application as it demonstrates the ability of the algorithm to scale to dense CRFs with large dimensions. We perform experiments on the Pascal dataset to indicate that the presented algorithms are able to attain lower energies than the mean-field inference method.

Key words. CRFs, MRFs, optimization, semantic segmentation, energy minimization

AMS subject classification. 90C35

DOI. 10.1137/18M1178104

1. Introduction. Conditional random fields (CRFs) are a popular framework for modeling several problems in computer vision. The energy function of the CRF consists of a sum of three types of terms: *unary energies* that depend on the label for one random variable; *pairwise energies* that depend on the labels of two random variables; and *higher-order energies* that depend on a collection of random variables. Notable works such as [2, 10, 19] focus on just the unary and pairwise energies, leaving out the higher-order energies for computational efficiency.

The popularity of CRFs led to a considerable research effort in efficient energy minimization algorithms. One of the biggest successes of this effort was the development of several

accurate continuous relaxations of the underlying discrete optimization problem [14, 28]. An important advantage of such relaxations is that they lend themselves easily to analysis, which allows us to compare them theoretically [28], as well as establish bounds on the quality of their solutions [6]. However, despite the successes of continuous relaxations, the algorithms used to solve such relaxations fail to scale well with the number of pairwise connections. To combat this deficiency, traditional energy minimization methods employed sparse connectivity structures, such as 4 or 8 connected grid CRFs.

By using a mean-field inference method [17], Krähenbühl and Koltun [19] were able to solve CRFs with many pairwise connections. It was shown that the use of dense pairwise connections achieved a more accurate labeling. In order to operationalize dense CRFs, Krähenbühl and Koltun [19] made two key observations. First, the pairwise potentials used in computer vision typically encourage similar labeling. This enabled them to restrict themselves to the special case of Gaussian pairwise potentials introduced by Tappen et al. [32]. Second, the message computation required at each iteration of mean-field can be carried out in $\mathcal{O}(N)$ operations using the filtering approach of Adams, Baek, and Abraham [1], where N is the number of random variables (of the order of hundreds of thousands). Vineet, Warrell, and Torr [34] made use of this filter-based method to perform mean-field inference on a dense CRF with sparse higher-order potentials, which provided a further improvement in segmentation accuracy.

While the mean-field algorithm does not provide any theoretical guarantees on the quality of the solution, the use of a richer model, namely, dense CRFs with sparse higher-order potentials, still allows us to obtain a significant improvement in the accuracy of several computer vision applications compared to sparse models [19, 34]. However, this still leaves open the intriguing possibility that the same filtering approach—which enabled the efficient mean-field algorithm—can also be used in conjunction with the principled methods of energy minimization with continuous relaxations. In this work we show that this is indeed possible; specifically, the main contributions of this paper are as follows:

(1) We are the first to combine dense CRFs with higher-order potentials when using a continuous relaxation of the maximum a posteriori (MAP) problem. Specifically, we formulate the energy function as both a quadratic programming (QP) and a linear programming (LP) relaxation and go on to show that both can be optimized efficiently using the filter-based method. As a novel contribution, we then extend the energy minimization algorithms of our existing work [2, 10] to deal with these higher-order potentials, while maintaining a complexity that is linear in the number of labels and random variables at each iteration.

(2) We provide novel relaxations of the higher-order terms which are based on the P^n-Potts model [15]. These formulations have been tailored to suit the specific relaxation in a way that is amenable to the continuous relaxation of the MAP problem. We also ensure the formulation allows efficient energy minimization using the two frameworks mentioned above.

In more detail, we make two contributions to the problem of energy minimization in dense CRFs with sparse higher-order potentials. First, we show that the conditional gradient of a QP relaxation [28] can be computed in a complexity linear in the number of labels and random variables. Together with our observation that the optimal step size of a descent direction can be computed analytically, this allows us to minimize the QP relaxation efficiently using the Frank–Wolfe algorithm [12]. Second, we introduce an iterative LP minimization algorithm which has a complexity at each iteration that is also linear in the number of labels and random

variables. To this end, instead of relying on a standard subgradient technique, we propose to make use of the proximal method [27]. The resulting proximal problem has a smooth dual, which can be efficiently optimized using block coordinate descent. We show that each block of variables can be optimized efficiently. Specifically, for one block, the problem decomposes into significantly smaller subproblems, each of which is defined over a single pixel. For the other block, the problem can be optimized via the Frank–Wolfe algorithm [12, 23]. We show that the conditional gradient required by this algorithm can be computed efficiently. In particular, we modify the filtering method of [1] such that the conditional gradient can be computed in a complexity linear in the number of labels and random variables. Besides this linear complexity, our approach has two additional benefits. First, it can be initialized with the solution of a faster, less accurate algorithm, such as mean-field [19], thus speeding up convergence. Second, the optimal step size of our iterative procedure can be obtained analytically, thus overcoming the need to rely on an expensive line search procedure.

There are preliminary versions of this work available, and the interested reader is encouraged to visit [2, 10]. This work can be considered as a unified view of our previous work [2, 10] with a novel addition of higher-order potentials. To keep this paper self-contained all relevant information and findings are detailed in this paper. Specifically, our contribution is a QP and LP relaxation for dense CRF with sparse higher-order potentials and their associated energy minimization frameworks.

2. Related work. Krähenbühl and Koltun popularized the use of densely connected CRFs at the pixel level [19], resulting in significant improvements both in terms of the quantitative performance and in terms of the visual quality of their results. By restricting themselves to Gaussian pairwise potentials, they made the computation of the message passing in mean-field feasible. This was achieved by formulating message computation as a convolution in a higher-dimensional space, which enabled the use of an efficient filter-based method [1]. Recent works leverage deep learning to achieve deep embeddings of the pairwise potential [5]; however, [5] acts on the patch space, whereas in this work we act on the pixel space.

While the initial work by [19] used a version of mean-field that is not guaranteed to converge, their follow-up paper [20] proposed a convergent mean-field algorithm for negative semi definite label compatibility functions. Recently, Baqué et al. [4] presented a new algorithm that has convergence guarantees in the general case. Vineet, Warrell, and Torr [34] extended the mean-field model to allow the addition of higher-order terms on top of the dense pairwise potentials, enabling the use of co-occurence potentials [24] and P^n-Potts models [15].

Independently from the mean-field work, Zhang and Chen [37] designed a different set of constraints that lends itself to a QP relaxation of the original problem. Their approach is similar to ours in that they use continuous relaxation to approximate the solution of the original problem but differs in the form of the pairwise potentials. The algorithm they propose to solve the QP relaxation has linearithmic[1] complexity, while ours is linear in the number of labels and random variables. Furthermore, it is not clear whether their approach can be easily generalized to tighter relaxations such as the LP.

Wang, Shen, and van den Hengel [35] derived a semidefinite programming relaxation of the energy minimization problem, allowing them to reach lower energies than mean-field. Their

---

[1]An algorithm is said be linearithmic if it runs in a computational complexity of $\mathcal{O}(N\log(N))$.

approach has the advantage of not being restricted to Gaussian pairwise potentials. Inference is made feasible by performing a low-rank approximation of the Gram matrix of the kernel, instead of using the filter-based method. However, in theory, the complexity of their algorithm is the same as our quadratic program, but in practice, the runtime is significantly higher.

The success of the inference algorithms naturally led to an interest in methods for learning the parameters of dense CRFs; while learning the parameters is orthogonal to this work, we include a brief review for completeness. Combining them with fully convolutional neural networks [26] has resulted in high performance on semantic segmentation applications [7]. Several works [29, 38] showed independently how to jointly learn the parameters of the unary and pairwise potentials of the CRF.

In this paper, we use the same filter-based method [1] as the one employed in mean-field. We use it to solve continuous relaxations of the original problem that have both convergence and quality guarantees. Our work can be viewed as a complementary direction to previous research trends in dense CRFs. While [4, 20, 34] improved mean-field and [29, 38] learned the parameters, we focus on the energy minimization problem.

3. Problem formulation. While CRFs can be used for many different applications, we use semantic segmentation as an illustrative example. As will be seen shortly, by using the appropriate choice of random variables, labels, and potentials, our model provides an intuitive framework for semantic segmentation.

3.1. Dense CRF energy function. We define a dense CRF over a set of N random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$, where each random variable X_a takes a single label from the set of M labels $\mathcal{L} = \{l_1, \ldots, l_M\}$. To formalize this labeling, a vector $\mathbf{x} \in \mathcal{L}^N$ is introduced such that the element x_a of $\mathbf{x}$ holds the label associated with the random variable X_a. Before proceeding to the energy function, it will prove useful to define a *clique* and its relationship to the sparse higher-order potentials. Formally, a clique is defined as a fully connected subgraph. In the context of this work, a clique with three or more random variables represents a higher-order potential and a clique with two random variables is represented by a pairwise potential. A given clique S_p is a subset of $\mathcal{X}$ and the set of cliques containing higher-order potentials $\mathcal{S}$ is defined below:

$$(3.1) \qquad\qquad\qquad\qquad \mathcal{S} = \{S_1, \ldots, S_R\},$$

$$(3.2) \qquad\qquad\qquad\qquad \text{s.t. } S_p \in \{S \subseteq \mathcal{X} \mid |S| > 2\}.$$

Here, R represents the total number of cliques in the set $\mathcal{S}$. It will also prove useful to introduce another set $\mathcal{R}_p$, which represents the set of indexes for the random variables in the clique S_p; this can be formally expressed as $\mathcal{R}_p = \{a \in \{1, \ldots, N\} \mid X_a \in S_p\}$. With the introduction of $\mathbf{x}_p$, which is a vector of more than two elements, containing the labels of the random variables in the clique S_p, the energy function can be defined as

$$(3.3) \qquad\qquad E(\mathbf{x}) = \sum_{a=1}^{N} \phi_a(x_a) + \sum_{a=1}^{N} \sum_{\substack{b=1 \\ b \neq a}}^{N} \psi_{a,b}(x_a, x_b) + \sum_{p=1}^{R} \theta_p(\mathbf{x}_p),$$

where $\phi_a(x_a)$ denotes the unary potential, $\psi_{a,b}(x_a, x_b)$ denotes the pairwise potential, and $\theta_p(\mathbf{x}_p)$ denotes the clique potential. The unary potential represents the cost of assigning the

random variable X_a the label x_a. The pairwise potential represents the cost of assigning the random variables X_a and X_b the labels x_a and x_b, respectively. The clique potential represents the cost of assigning all random variables in S_p the labels $\mathbf{x}_p$ and embodies the higher-order potentials.

So far, the dense CRF has been described using random variables and their associated labels. In this work we use semantic segmentation as a tangible application to demonstrate the energy minimization of the proposed methods. In detail, a random variable corresponds to a pixel and the associated labels correspond to a semantic class. A *superpixel*—which is a collection of homogeneous spatially adjacent pixels—is represented by a higher-order clique. The optimal solution to this energy function forms an optimization problem over the variable $\mathbf{x}$ and can be compactly written as

$$(3.4) \qquad \mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^{\mathbf{N}}} E(\mathbf{x}).$$

In the general case this minimization problem is NP-hard [18] and hence cannot be solved in polynomial time. To this extent, efficient methods will be introduced in sections 4 and 5 that compute approximate solutions for this minimization problem.

3.1.1. Unary potentials. The unary potentials for this formulation can be arbitrary but generally provide a rough initial labeling solution. In this work we employ unary potentials which are derived from TextonBoost [25, 30]. More detail on the generation of the unary potentials are given in section 6.1.

3.1.2. Gaussian pairwise potentials. We follow the work of [19] by using Gaussian pairwise potentials, taking the form of

$$(3.5) \qquad \psi_{a,b}(x_a, x_b) = \mu(x_a, x_b) \mathrm{K}_{ab},$$

$$\text{s.t. } \mathrm{K}_{ab} = \sum_m w^{(m)} k^{(m)} \big(\mathbf{f}_a^{(m)}, \mathbf{f}_b^{(m)}\big),$$

where $\mu(x_a, x_b)$ is a scalar representing the *label compatibility*, K_{ab} is the *pixel compatibility* function, which is defined in the next paragraph, $w^{(m)}$ is a scalar weighting factor, and $k^{(m)}\big(\mathbf{f}_a^{(m)}, \mathbf{f}_b^{(m)}\big)$ are Gaussian kernels taking the form of

$$(3.6) \qquad k^{(m)}(\mathbf{f}_a^{(m)}, \mathbf{f}_b^{(m)}) = \exp\bigg(-\frac{||\mathbf{f}_a^{(m)} - \mathbf{f}_b^{(m)}||^2}{2\sigma_{(m)}^2}\bigg).$$

The terms $\mathbf{f}_a^{(m)}$ and $\mathbf{f}_b^{(m)}$ are feature vectors containing the spatial and color information of the image with pixel indices a and b, respectively.

Pixel compatibility. For multiclass semantic segmentation problems, the pixel compatibility function takes the form of contrast-sensitive two-kernel potentials, defined as

$$(3.7) \qquad \mathrm{K}_{ab} = w^{(1)} \exp\bigg(-\frac{|p_a - p_b|^2}{2\sigma_{(1)}^2} - \frac{|I_a - I_b|^2}{2\sigma_{(2)}^2}\bigg) + w^{(2)} \exp\bigg(-\frac{|p_a - p_b|^2}{2\sigma_{(3)}^2}\bigg),$$

with I_a, I_b and p_a, p_b representing the color information and spatial information of pixels a and b, respectively, and $\sigma^2_{(m)}$ is the kernel bandwidth. The first term corresponds to the *bilateral kernel* and is inspired by the observation that pixels of similar color and position are likely to take the same label, and the second term corresponds to a *spatial kernel* which penalizes small isolated regions. The parameters $w^{(1)}$, $w^{(2)}$, $\sigma^2_{(1)}$, $\sigma^2_{(2)}$, and $\sigma^2_{(3)}$ are obtained via cross-validation; more detail is given in section 6.1.

Label compatibility. The *label compatibility* function $\mu(x_a, x_b)$ forms part of the cost of assigning the random variables X_a and X_b the labels corresponding to the value of x_a and x_b, respectively. The label compatibility function used for this work is the Potts model [10] and is specified as

$$(3.8) \qquad \mu_{Potts}(x_a, x_b) = \mathbb{1}[x_a \neq x_b],$$

where $\mathbb{1}[\cdot]$ is the Iverson bracket. While other label compatibility functions exist, such as metric or semimetric functions [14], the Potts model was chosen as it enables more sophisticated minimisation algorithms to be leveraged, which will be discussed in sections 4 and 5.

3.1.3. Higher-order potentials. In this work, the higher-order terms are represented as a *clique potential*. We formulate the higher-order potential using the P^n-Potts models [15]. Specifically, if all of the random variables in S_p do not take the same label, the clique potential introduces a constant cost, which we set to be proportional to the variance of the color information of the super pixel. The clique potential is defined by

$$(3.9) \qquad \theta_p(\mathbf{x}_p) = \begin{cases} 0 & \text{if } x_c = x_d, \ \forall c, d \in \mathcal{R}_p \\ \Gamma \exp\left\{\frac{-\sigma_p^2}{\eta}\right\} & \text{otherwise,} \end{cases}$$

where Γ and η are cross-validated parameters and σ_p^2 represents the variance of the pixel color values within the clique S_p. To this extent, the set of random variables which form the clique S_p must be carefully chosen. Hence, by context of the image, all of the corresponding pixels in the clique S_p must represent the same object.

Generating cliques. For this work, a clique represents a superpixel of an oversegmented image. A superpixel is a collection of adjoining pixels that share similar color information. The cliques were generated using the mean-shift algorithm [9], which is a semiparametric method of segmenting an image into superpixels. We used the mean-shift algorithm due to its simplicity; however, in practice any algorithm that generates oversegmentations can be used. Evaluating the quality of the oversegmentation is beyond the scope of this work; however, we cross-validate the size of the superpixels to ensure we use higher-order potentials that match the problem. Representing superpixels by higher-order potentials introduces an implicit constraint on S_p, that is, $S_p \cap S_q = \varnothing \ \forall p, q \neq p$, as every pixel is assigned to exactly one superpixel. While this is not a necessary constraint (as the algorithms can deal with arbitrary sizes of cliques), we will make use of this later on to ensure that the complexity of each iteration is linear in the number of labels and pixels.

3.1.4. Filtering method. The pixel compatibility function defined in (3.7) was chosen to take a Gaussian form due to the fact that it allows a filter-based method [1] to be utilized.

This filter-based method exploits the *permutohedral lattice* to achieve efficient computation of operations featuring Gaussian kernels; specifically it approximates the following:

$$(3.10) \qquad \forall a \in \{1, \ldots, N\}, \qquad v'_a = \sum_{b=1}^{N} k(\mathbf{f}_a, \mathbf{f}_b) v_b,$$

where $v'_a, v_b \in \mathbb{R}$, $b \in \{1, \ldots, N\}$, and $k(\mathbf{f}_a, \mathbf{f}_b)$ is a Gaussian kernel described in section 3.1.2. A naïve approach would take $\mathcal{O}(N^2)$ operations. However, the use of the filtering method enables this operation to be computed in approximately $\mathcal{O}(N)$ operations. Krähenbühl and Koltun [19] employed this filter-based method to compute the message passing step of the mean-field inference algorithm efficiently. We investigated the accuracy of the filter-based method [1] with differing values for the variances of (3.7) in our preliminary work [10]. The results indicate that the filtering method introduces an error scaling factor, which for large values of N tends to 0.6. The interested reader is referred to Appendix A of [10] for more information. This scaling factor will be propagated into the gradient, but it is implicitly accounted for when the optimal step size is computed and hence does not have an adverse affect on the algorithms as the direction of the gradient is exact.

3.1.5. Integer program formulation.
We now formulate the energy minimization function (3.3) as an integer program (IP). To this end, a vector $\mathbf{y} \in \mathbb{R}^{NM}$ is introduced, such that its elements $y_{a:i} \in \{0,1\}$ are binary variables indicating whether the random variable X_a takes the label l_i. The vector $\mathbf{y}_p = \{y_{c:i} | c \in \mathcal{R}_p, i \in \mathcal{L}\}$ is introduced which holds the vectors of indicator variables for $\mathbf{x}_p$. With this new notation the energy minimization function can be defined as

$$(3.11) \qquad \min_{\mathbf{y}} \sum_{a=1}^{N} \sum_{i \in \mathcal{L}} \phi_a(i) y_{a:i} + \sum_{a=1}^{N} \sum_{\substack{b=1 \\ b \neq a}}^{N} \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} \psi_{a,b}(i,j) y_{a:i} y_{b:j} + \sum_{p=1}^{R} \overline{\theta}_p(\mathbf{y}_p)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{L}} y_{a:i} = 1 \quad \forall a \in \{1, \ldots, N\},$$
$$y_{a:i} \in \{0,1\} \quad \forall a \in \{1, \ldots, N\}, \quad \forall i \in \mathcal{L},$$

$$(3.12) \qquad \overline{\theta}_p(\mathbf{y}_p) = \begin{cases} 0 & \text{if } y_{c:i} = y_{d:i}, \forall c, d \in \mathcal{R}_p, c \neq d, \forall i \in \mathcal{L}, \\ \Gamma \exp\left\{\frac{-\sigma_p^2}{\eta}\right\} & \text{otherwise.} \end{cases}$$

The first set of constraints ensure that each random variable has to be assigned exactly one label, while the second constraint ensures that the labeling is binary. It is important to note that $\overline{\theta}_p(\cdot)$ is a polynomial with an order equal to the number of random variables within the clique S_p. Normally the manipulation of $\overline{\theta}_p(\cdot)$ would exhibit an intractable complexity; however, by exploiting labeling consistency in the sparse higher-order potentials, it will be shown that this higher-order polynomial can be reformulated in a tractable manner.

3.2. Relaxations.
It is worth noting that the IP in (3.11) is NP-hard [18] and hence cannot be solved in polynomial time. We address this issue by relaxing the integral constraint to approximate the IP, enabling us to formulate an energy minimization problem. Specifically we formulate a QP relaxation and an LP relaxation given in sections 4 and 5, respectively.

4. Quadratic program. We are now ready to demonstrate how the filter-based method [1] can be used to optimize our first continuous relaxation, namely, the QP relaxation.

4.1. Notation and formulation. The unary and pairwise potentials of the IP given in (3.11) can be neatly summarized in vector form with linear algebra operations. To this extent, the unary potential can be concisely written as the dot product between the vector $\mathbf{y} \in \mathbb{R}^{NM}$ and the vector of unary terms denoted $\boldsymbol{\phi} \in \mathbb{R}^{NM}$. The pairwise potential is a little more complex and will require the use of the label compatibility matrix $\boldsymbol{\mu}_{Potts} \in \mathbb{R}^{M \times M}$, which in this case is the Potts model described in (3.8). For the pixel compatibility function, each kernel (3.6) is represented by the Gram matrix $\boldsymbol{K}^{(m)} \in \mathbb{R}^{N \times N}$. The element of $\boldsymbol{K}^{(m)}$ at index (a, b) corresponds to the value of $k^{(m)}\big(\mathbf{f}_a^{(m)}, \mathbf{f}_b^{(m)}\big)$. The matrix $\boldsymbol{\Psi} \in \mathbb{R}^{NM \times NM}$ represents the pairwise terms and is defined as

$$(4.1) \qquad \boldsymbol{\Psi} = \boldsymbol{\mu}_{Potts} \otimes \sum_m w^{(m)} \big(\boldsymbol{K}^{(m)} - \boldsymbol{I}_N \big),$$

where $\otimes$ is the Kronecker product and $\boldsymbol{I}_N$ is the identity matrix of size $N \times N$. Similarly to [20], $\boldsymbol{K}^{(m)}$ has a unit diagonal and hence the identity matrix $\boldsymbol{I}_N$ is introduced for completeness. The objective function of the IP for the unary and pairwise potentials is given in vectorized form as

$$(4.2) \qquad \min_{\mathbf{y}} \boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y},$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{L}} y_{a:i} = 1 \quad \forall a \in \{1, \ldots, N\},$$
$$y_{a:i} \in \{0, 1\} \quad \forall a \in \{1, \ldots, N\}, \quad \forall i \in \mathcal{L}.$$

In the general case, a clique potential forms a high-order polynomial with an order equal to the number of random variables in each clique. However, by exploiting labeling consistency, we are able to reformulate this high-order polynomial as a lower-order one. To this end, a binary auxiliary variable $z_{p:i}$ is introduced which indicates whether all of the random variables in the clique S_p take the label $\mathcal{L}_i$. The auxiliary variable $z_{p:i}$ is given as

$$(4.3) \qquad z_{p:i} = \begin{cases} 0 & \text{if } y_{c:i} = 1 \quad \forall c \in \mathcal{R}_p, \\ 1 & \text{otherwise.} \end{cases}$$

In other words if all random variables in the clique S_p take the same label, then $z_{p:i} = 0$. Before proceeding to the definition of the clique potential for the QP it will be beneficial to introduce an additional term $H_p(a)$, which is used to indicate if the random variable X_a belongs to the clique S_p. Formally $H_p(a) = 1$ if $a \in \mathcal{R}_p$ and $H_p(a) = 0$ otherwise. With the addition of the auxiliary variable $z_{p:i}$ and the indicator term $H_p(a)$, the clique potential forms a quadratic polynomial in $z_{p:i}$ and $y_{a:i}$, which is given below. The clique potential is given as

$$(4.4) \qquad f_c := \sum_{p=1}^{R} \sum_{i \in \mathcal{L}} C_p \Big(z_{p:i} + \big((1 - z_{p:i}) \sum_{a=1}^{N} H_p(a)(1 - y_{a:i}) \big) \Big).$$

It is worth noting that the last term will always evaluate to zero. However, once the binary constraints on $z_{p:i}$ and $y_{a:i}$ are relaxed, the latter term provides a coupling between $z_{p:i}$ and $y_{a:i}$.

More detail will be given on this in section 4.1.1. The vectorized version of $z_{p:i}$ is $\mathbf{z} \in \mathbb{R}^{MR}$. The values of $H_p(a)$ form the matrix $\boldsymbol{H} \in \mathbb{R}^{MR \times NM}$, which is a sparse matrix of ones, such that the elements are in the correct order to perform the summations. The matrix $\boldsymbol{H}$ is purely provided for illustrative purposes, and due to its sparse nature, in the implementation it is not stored as a matrix of size $MR \times NM$. Instead, R arrays are instantiated with each array containing the indexes of the pixels within the clique. With the addition of $\mathbf{z}$ and $\boldsymbol{H}$, the IP can be concisely written in vector form as

$$(4.5) \qquad \min_{\mathbf{y}, \mathbf{z}} f(\mathbf{y}, \mathbf{z}) = \min_{\mathbf{y}, \mathbf{z}} \left(\boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y} + \mathbf{c}^T \mathbf{z} + (\mathbf{1}_\mathbf{z} - \mathbf{z})^T \mathbf{C} \boldsymbol{H} (\mathbf{1}_\mathbf{y} - \mathbf{y}) \right),$$

where $\mathbf{c} \in \mathbb{R}^{MR}$ is a vector containing the constants C_p in the appropriate order. The matrix $\mathbf{C} \in \mathbb{R}^{MR \times MR}$ is the diagonal matrix of the vector $\mathbf{c}$. The vectors $\mathbf{1}_\mathbf{z} \in \mathbb{R}^{MR}$ and $\mathbf{1}_\mathbf{y} \in \mathbb{R}^{NM}$ are vectors of all ones.

4.1.1. Relaxations. The IP introduced in (4.5) is an NP-hard problem. To overcome this difficulty, it is proposed to relax the binary constraints on the indicator variable $y_{a:i}$ and the auxiliary variables $z_{p:i}$, allowing them to take fractional values between 0 and 1. Formally, with these relaxations, the feasible set for $\mathbf{y}$ and $\mathbf{x}$ becomes

$$(4.6) \qquad \mathcal{Y} = \left\{ \mathbf{y} \; \middle| \; \begin{array}{ll} \sum_{i \in \mathcal{L}} y_{a:i} = 1, & a \in \{1, \dots, N\} \\ y_{a:i} \geq 0, & a \in \{1, \dots, N\}, i \in \mathcal{L} \end{array} \right\},$$

$$(4.7) \qquad \mathcal{Z} = \left\{ \mathbf{z} \; \middle| \; 0 \leq z_{p:i} \leq 1, \quad S_p \in \mathcal{S}, i \in \mathcal{L} \right\}.$$

Thus, the QP relaxation can be formally defined as

$$(4.8) \qquad \min_{\mathbf{y}, \mathbf{z}} f(\mathbf{y}, \mathbf{z}) = \min_{\mathbf{y}, \mathbf{z}} \left(\boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y} + \mathbf{c}^T \mathbf{z} + (\mathbf{1}_\mathbf{z} - \mathbf{z})^T \mathbf{C} \boldsymbol{H} (\mathbf{1}_\mathbf{y} - \mathbf{y}) \right)$$
$$\text{s.t. } \mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}.$$

4.2. Minimization. The minimization of the objective function is achieved via the Frank–Wolfe algorithm [12], which is advantageous for two reasons: First, the Frank–Wolfe algorithm is projection free, and second, the conditional gradient can be computed in a complexity linear in the number of pixels and labels. The objective function of (4.8) can be solved in several ways; however, even though (4.8) is non–convex, we choose to obtain a local minimum using the Frank–Wolfe algorithm, as we are able to take advantage of the aforementioned qualities. While the Frank–Wolfe algorithm normally optimizes convex objectives, it has been proven to find a stationary point at a rate of $\mathcal{O}(1/\sqrt{t})$ of a nonconvex objective function over a convex compact set,[2] where t is the number of iterations [22]. The key steps of the algorithm are shown in Algorithm 4.1. To utilize the Frank–Wolfe algorithm effectively, three steps need to be taken: obtaining the gradient of the objective function (step 3), efficient conditional gradient computation (step 4), and the optimal step size calculation (step 5). All three of these requirements are achieved in a feasible manner and details are given in this section.

---

[2] A topological space $\mathcal{X}$ is called compact if every open cover has a finite subcover. Furthermore, a set $\mathcal{D} \subset \mathbb{R}^N$ is compact if and only if it is closed and bounded [13].

Algorithm 4.1. QP minimization algorithm

1: $\mathbf{y}^0 \in \mathcal{Y}, \mathbf{z}^0 \in \mathcal{Z}$ ▷ Initialize
2: **while** not converged **do**
3:      $\mathbf{g}^t \leftarrow \nabla f(\mathbf{y}^t, \mathbf{z}^t)$ ▷ Compute the gradient
4:      $\left(\mathbf{s_y^t}, \mathbf{s_z^t}\right)^T \leftarrow \text{argmin}_{\mathbf{s_y} \in \mathcal{Y}, \mathbf{s_z} \in \mathcal{Z}} \left\langle \left(\mathbf{s_y}, \mathbf{s_z}\right)^T, \mathbf{g}^t \right\rangle$ ▷ Compute the conditional gradient
5:      $\delta \leftarrow \text{argmin}_{\delta \in [0,1]} f(\mathbf{y}^t + \delta(\mathbf{s_y^t} - \mathbf{y}^t), \mathbf{z}^t + \delta(\mathbf{s_z^t} - \mathbf{z}^t))$ ▷ Compute the optimal step size
6:      $\left(\mathbf{y}^{t+1}, \mathbf{z}^{t+1}\right) \leftarrow \left(\mathbf{y}^t + \delta(\mathbf{s_y^t} - \mathbf{y}^t), \mathbf{z}^t + \delta(\mathbf{s_z^t} - \mathbf{z}^t)\right)$ ▷ Update

---

4.2.1. Gradient computation. The Frank–Wolfe algorithm requires efficient computation of the gradient, which can easily be achieved for this problem. Formally, the gradient of f is defined as

$$(4.9) \qquad \nabla f(\mathbf{y}, \mathbf{z}) = \begin{pmatrix} \phi + 2\boldsymbol{\Psi}\mathbf{y} + \boldsymbol{H}^\mathbf{T}\mathbf{C}(\mathbf{z} - \mathbf{1_z}) \\ \mathbf{c} + \mathbf{C}\boldsymbol{H}(\mathbf{y} - \mathbf{1_y}) \end{pmatrix}.$$

Specific attention is drawn to the complexity of the gradient in the $\mathbf{y}$ direction. The unary term is left as a constant and hence scales linearly with the number of labels and random variables. Computing the value of the pairwise potential in the naïve way would result in a complexity of the order $\mathcal{O}((MN)^2)$, which for dimensions of an image is intractable. However, due to the elements of $\boldsymbol{\Psi}$ containing Gaussian kernels, this expensive computation of the pairwise potential can be performed in linear time using the filter-based method; more detail on this filter-based method is given in section 3.1.4.

Due to the fact that $\boldsymbol{H}$ performs a summation over random variables and labels and there is no intersection between cliques $S_p \cap S_p = \varnothing$, the resulting complexity of the clique potential is of the order $\mathcal{O}(NM)$ as for each clique we perform a sum over only the labels and pixels within that clique.

4.2.2. Low-cost gradient computation. We observe that the gradient introduced in section 4.2.1 need not be explicitly computed at every iteration. Instead the gradient can be incremented from its initial value using the update equations, which are given as

$$(4.10) \qquad \begin{pmatrix} \mathbf{y}^{t+1} \\ \mathbf{z}^{t+1} \end{pmatrix} = \begin{pmatrix} \mathbf{y}^t + \delta(\mathbf{s_y}^t - \mathbf{y}^t) \\ \mathbf{z}^t + \delta(\mathbf{s_z}^t - \mathbf{z}^t) \end{pmatrix},$$

where s_y and s_z are the conditional gradients of $f(\mathbf{y}, \mathbf{z})$. The expensive operations of $2\boldsymbol{\Psi}\mathbf{y}$ and $\boldsymbol{H}^T\mathbf{C}\mathbf{z}$ in (4.9) can be avoided by using the values of $2\boldsymbol{\Psi}(\mathbf{s_y} - \mathbf{y})$ and $\boldsymbol{H}^T\mathbf{C}(\mathbf{s_z} - \mathbf{z})$—which are both computed as part of the optimal step size—and by using the update equations. By multiplying the update equation for $\mathbf{y}$ by $2\boldsymbol{\Psi}$ and multiplying the update equation for $\mathbf{z}$ by $\boldsymbol{H}^T\mathbf{C}$, the updated terms can be given as

$$(4.11) \qquad 2\boldsymbol{\Psi}\mathbf{y}^{t+1} = 2\boldsymbol{\Psi}\mathbf{y}^t + 2\delta\boldsymbol{\Psi}(\mathbf{s_y}^t - \mathbf{y}^t),$$

$$(4.12) \qquad \boldsymbol{H}^T\mathbf{C}\mathbf{z}^{t+1} = \boldsymbol{H}^T\mathbf{C}\mathbf{z}^t + \delta\boldsymbol{H}^T\mathbf{C}(\mathbf{s_z}^t - \mathbf{z}^t).$$

Thus, allowing the explicit computation of $2\boldsymbol{\Psi}\mathbf{y}$ and $\boldsymbol{H}^T\mathbf{C}\mathbf{z}$ to be avoided. Instead their values can be incremented from their previous state. Hence the updated gradient in $\mathbf{y}$ is also

an increment from the previous step via the addition of $2\delta\boldsymbol{\Psi}(\mathbf{s_y}^t - \mathbf{y}^t) + \delta\boldsymbol{H}^T\mathbf{C}(\mathbf{s_z}^t - \mathbf{z}^t)$ and is more formally given as

$$(4.13) \qquad \nabla_y f(\mathbf{y}^{t+1}, \mathbf{z}^{t+1}) = \nabla_y f(\mathbf{y}^t, \mathbf{z}^t) + 2\delta\boldsymbol{\Psi}(\mathbf{s_y}^t - \mathbf{y}^t) + \delta\boldsymbol{H}^T\mathbf{C}(\mathbf{s_z}^t - \mathbf{z}^t).$$

A similar approach can be taken for $\nabla_z f(\mathbf{y}^{t+1}, \mathbf{z}^{t+1})$. Incrementing the gradients reduces the operational complexity by a constant factor of two. This is due to the fact that the filter-based method does not need to be called when computing the gradient and the product of $\boldsymbol{H}^T\mathbf{C}\mathbf{z}$ does not need to be computed either.

4.2.3. Conditional gradient computation.
Computing the conditional gradient is an essential step in the Frank–Wolfe algorithm, and we show that it can be computed in a complexity linear in the number of labels and pixels. The conditional gradient $\left(\begin{smallmatrix}\mathbf{s_y}\\\mathbf{s_z}\end{smallmatrix}\right)$, with $\mathbf{s_y} \in \mathcal{Y}, \mathbf{s_z} \in \mathcal{Z}$, of the objective function f is obtained by solving

$$(4.14) \qquad \begin{pmatrix}\mathbf{s_y}\\\mathbf{s_z}\end{pmatrix} \in \operatorname{argmin}_{\mathbf{s_y}\in\mathcal{Y},\mathbf{s_z}\in\mathcal{Z}} \left\langle \begin{pmatrix}\mathbf{s_y}\\\mathbf{s_z}\end{pmatrix}, \nabla f(\mathbf{y}, \mathbf{z}) \right\rangle.$$

Minimizing (4.14) with dimensions proportional to that of an image would normally be an expensive operation. However, the reader's attention is drawn to the fact that the feasible set $\mathcal{Y}$ is linearly separable into N subsets as follows: $\mathcal{Y} = \prod_a \mathcal{Y}_a$, where $\mathcal{Y}_a = \{y_{a:i}| \sum_{i\in\mathcal{L}} y_{a:i} = 1, y_{a:i} \geq 0, i \in \mathcal{L}\}$. Exploiting this constraint enables the minimization problem to be broken down into N smaller minimization problems for each of the random variables in $\mathcal{X}$. Minimizing $\langle \mathbf{s_y}, \nabla f(\mathbf{y}, \mathbf{z})\rangle$ with respect to $\mathbf{s_y}$ is thus achieved via N linear searches with the search space restricted to the number of labels. The resulting computational complexity of the conditional gradient is $\mathcal{O}(NM)$ and is more formally defined as

$$(4.15) \qquad s_{a:i}^{(y)} = \begin{cases} 1 & \text{if } i = \operatorname{argmin}_{i\in\mathcal{L}} \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial y_{a:i}}, \\ 0 & \text{otherwise.} \end{cases}$$

For the case where $\operatorname{argmin}_{i\in\mathcal{L}} \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial y_{a:i}}$ yields multiple values, we arbitrarily assign $s_{a:i}^{(y)} = 1$ for only one of the given minima. The feasible set $\mathcal{Z}$ is also separable and can be decomposed as follows: $\mathcal{Z} = \prod_{p,i} \mathcal{Z}_{p:i}$. Thus the minimization for $\mathbf{s_z} = \operatorname{argmin}_{i\in\mathcal{Z}}\langle \mathbf{s_z}, \nabla_z f(\mathbf{y}, \mathbf{z})\rangle$ can be performed via a linear search through all MR elements. With the constraints on the set $\mathcal{Z}_{p:i} = \{z|0 \leq z_{p:i} \leq 1 \ \forall p, \forall i \in \mathcal{L}\}$, the conditional gradient $\mathbf{s_z}$, is given as

$$(4.16) \qquad s_{p:i}^{(z)} = \begin{cases} 1 & \text{if } \nabla_z f(\mathbf{y}, \mathbf{z}) < 0, \\ 0 & \text{otherwise.} \end{cases}$$

The complexity of $\mathbf{s_z}$ will always be significantly less than the complexity of $\mathbf{s_y}$ due to $R \ll N$. Hence, the computational complexity of calculating the conditional gradient is $\mathcal{O}(NM)$ as computing $\mathbf{s_y}$ requires the most floating point operations.

4.2.4. Optimal step size calculation.
Traditionally, the step size to the Frank–Wolfe algorithm is achieved via line search. However, for this problem the optimal step size can be computed via minimizing a quadratic function over a single variable. This quadratic function

has a closed form solution and the minimum can be calculated analytically. The optimal step size for the Frank–Wolfe algorithm is obtained by solving

$$(4.17) \qquad \delta = \operatorname{argmin}_{\delta \in [0,1]} f(\mathbf{y} + \delta(\mathbf{s_y} - \mathbf{y}), \mathbf{z} + \delta(\mathbf{s_z} - \mathbf{z})).$$

A closed form solution of the optimal step size is given in Appendix A.1. Obtaining the optimal step size will result in faster convergence and hence yield an efficient algorithm.

4.3. Summary. The above procedure remains linear in the number of pixels and labels at each iteration, despite introducing higher-order potentials which would normally cause intractability within the algorithm. This is achieved via exploiting the filter-based method [1], labeling consistency within a clique and enforcing the intersection between cliques to be an empty set. It is worth noting that the filter-based method is called only once per iteration, resulting in an efficient QP minimization algorithm.

5. Linear program. In this section we introduce the LP relaxation, our second continuous relaxation. To this end, relaxations will be applied to the objective function (3.11) and dual variables will be introduced, allowing the Lagrange dual problem to be formulated. An optimal solution can then be found via the use of the proximal minimization algorithm [27] which guarantees a monotonic decrease in the objective function.

5.1. Linear programming relaxation. In a similar manner to the QP, we also relax the binary indicator variables $y_{a:i}$, and due to the use of the Potts model and the P^n-Potts model, we can write the relaxation of (3.11) as a piecewise linear function, defined as

$$(5.1) \quad \min_{\mathbf{y}} \tilde{E}(\mathbf{y}) = \sum_{a=1}^{N} \sum_{i \in \mathcal{L}} \phi_{a:i} y_{a:i} + \sum_{a=1}^{N} \sum_{\substack{b=1 \\ b \neq a}}^{N} \sum_{i \in \mathcal{L}} \mathrm{K}_{ab} \frac{|y_{a:i} - y_{b:i}|}{2} + \sum_{p=1}^{R} C_p \max_{i \in \mathcal{L}} \max_{\substack{c,d \in \mathcal{R}_p \\ c \neq d}} |y_{c:i} - y_{d:i}|$$

$$\text{s.t. } \mathbf{y} \in \mathcal{Y},$$

where K_{ab} is the pixel compatibility function defined in (3.7). For integer labelings, the objective $\tilde{E}(\mathbf{y})$ has the same value as the IP objective $E(\mathbf{y})$ and is known to provide the best theoretical bounds [14]. Using standard solvers to minimize this LP would require the introduction of $\mathcal{O}((NM)^2)$ variables (see (5.2)), making it intractable. Therefore the nonsmooth objective of (5.1) has to be optimized directly. This was handled using projected subgradient descent in our previous version [10], which also turns out to be inefficient in practice. In this paper, we extend the algorithm introduced in [2] to handle higher-order potentials while maintaining linear scaling at each iteration in both space and time complexity.

The piecewise linear functions $|y_{a:i} - y_{b:i}|$, in the pairwise and clique potentials, can be reformulated as piecewise maximum functions $\max\{y_{a:i} - y_{b:i}, y_{b:i} - y_{a:i}\}$, and then subsequently replaced by auxiliary variables $v_{ab:i}$ and w_p in the standard way. The auxiliary variables and their constraints enable the minimization problem to be defined without the piecewise maximum operators. With the introduction of these auxiliary variables the primal minimization problem can be written as an LP relaxation and is given as

$$(5.2) \qquad \min_{\mathbf{y},\mathbf{v},\mathbf{w}} \sum_{a=1}^{N} \sum_{i\in\mathcal{L}} \phi_{a:i} y_{a:i} + \sum_{a=1}^{N} \sum_{\substack{b=1\\b\neq a}}^{N} \sum_{i\in\mathcal{L}} \frac{\mathrm{K}_{ab}}{2} v_{ab:i} + \sum_{p=1}^{R} C_p w_p + \frac{1}{2\lambda}\|\mathbf{y}-\mathbf{y}^k\|^2$$

$$\text{s.t. } v_{ab:i} \geq y_{a:i} - y_{b:i} \qquad \forall a,b \in \{1,\dots,N\} \qquad a \neq b \qquad \forall i \in \mathcal{L},$$
$$v_{ab:i} \geq y_{b:i} - y_{a:i} \qquad \forall a,b \in \{1,\dots,N\} \qquad a \neq b \qquad \forall i \in \mathcal{L},$$
$$w_p \geq y_{c:pi} - y_{d:pi} \quad \forall c,d \in \mathcal{R}_p \qquad c \neq d \qquad \forall i \in \mathcal{L} \qquad \forall p \in \{1,\dots,R\},$$
$$w_p \geq y_{d:pi} - y_{c:pi} \quad \forall c,d \in \mathcal{R}_p \qquad c \neq d \qquad \forall i \in \mathcal{L} \qquad \forall p \in \{1,\dots,R\},$$
$$y_{a:i} \geq 0 \qquad\qquad \forall a \in \{1,\dots,N\} \qquad \forall i \in \mathcal{L},$$
$$\sum_{i\in\mathcal{L}} y_{a:i} = 1 \qquad\qquad \forall a \in \{1,\dots,N\}.$$

In the next section we present our minimization strategy for the above LP relaxation.

5.2. Minimization. In this section we present our efficient minimization strategy, which uses the proximal method [27]. The complexity of each iteration of our implementation remains linear in the number of labels and pixels.

5.2.1. Proximal minimization for LP relaxation. Our goal is to design an efficient minimization strategy for the LP relaxation in (5.1). In our previous version [10], we utilized projected subgradient descent to minimize an LP similar to (5.2); however, this method resulted in a significantly high runtime and a complexity that scales at $\mathcal{O}(MN\log(N))$ at each iteration. To this end, we propose to use the proximal minimization algorithm [27]. The additional quadratic regularization term makes the dual problem smooth, enabling the use of more sophisticated optimization methods. Furthermore, this method guarantees a monotonic decrease in the objective value, enabling us to leverage faster methods for initialization. In the remainder of this paper, we detail this approach and show that each iteration has a complexity linear in the number of labels and pixels. In practice, our algorithm converges in a small number of iterations, thereby making the overall approach computationally efficient. The proximal minimization algorithm [27] is an iterative method that, given the current estimate of the solution $\mathbf{y}^k$, solves the problem

$$(5.3) \qquad\qquad \min_{\mathbf{y}} \tilde{E}(\mathbf{y}) + \frac{1}{2\lambda}\|\mathbf{y}-\mathbf{y}^k\|^2$$
$$\text{s.t. } \mathbf{y} \in \mathcal{Y},$$

where λ influences the weighting of the quadratic regularizer. In this section we introduce a new algorithm that is tailored to this problem. In particular, we solve the Lagrange dual of (5.3) in a blockwise fashion.

5.2.2. Dual formulation. To write the LP relaxation (5.1) as the dual function, four vectors of dual variables will be introduced for each constraint:

$$(5.4) \qquad \boldsymbol{\alpha} = \{\alpha_{ab:i}^1, \alpha_{ab:i}^2 |\ a \in \{1,\dots,N\}, b \in \{1,\dots,N\}, a \neq b, i \in \mathcal{L}\},$$

$$(5.5) \qquad \boldsymbol{\mu} = \{\mu_{cd:pi}^1, \mu_{cd:pi}^2 |\ c,d \in \mathcal{R}_p, c \neq d, i \in \mathcal{L}, p \in \{1,\dots,R\}\},$$

$$(5.6) \qquad \boldsymbol{\gamma} = \{\gamma_a | a \in \{1,\dots,N\}, i \in \mathcal{L}\}\},$$

$$(5.7) \qquad \boldsymbol{\beta} = \{\beta_a | a \in \{1,\dots,N\}\},$$

where (5.4) is for the constraints on $v_{ab:i}$, (5.5) for the constraints on w_p, (5.6) for the non-negativity of $y_{a:i}$, and (5.7) for the labeling of $y_{a:i}$, respectively. The dimensions of these vectors are $\boldsymbol{\alpha} \in \mathbb{R}^{2N(N-1)M}$, $\boldsymbol{\mu} \in \mathbb{R}^{2N(N-1)M}$, $\boldsymbol{\beta} \in \mathbb{R}^{N}$, and $\boldsymbol{\gamma} \in \mathbb{R}^{NM}$. Clearly when dealing with images, the dimensions of $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ are intractable. It will be shown that these vectors need not be stored explicitly; instead they can be stored in a compact form. To this extent, three matrices are introduced: $\mathbf{A} \in \mathbb{R}^{NM \times 2N(N-1)M}$, $\mathbf{U} \in \mathbb{R}^{NM \times 2N(N-1)M}$, and $\mathbf{B} \in \mathbb{R}^{NM \times N}$, such that

$$(5.8) \qquad (\mathbf{A}\boldsymbol{\alpha})_{a:i} = \sum_{\substack{b=1 \\ a \neq b}}^{N} (\alpha_{ab:i}^2 - \alpha_{ab:i}^1 - \alpha_{ba:i}^2 + \alpha_{ba:i}^1),$$

$$(5.9) \qquad (\mathbf{U}\boldsymbol{\mu})_{c:pi} = \sum_{\substack{d \in \mathcal{R}_p \\ c \neq d}} (\mu_{cd:pi}^2 - \mu_{cd:pi}^1 - \mu_{dc:pi}^2 + \mu_{dc:pi}^1),$$

$$(5.10) \qquad (\mathbf{B}\boldsymbol{\beta})_{a:i} = \beta_a.$$

As will be seen shortly, only the products of $(\mathbf{A}\boldsymbol{\alpha}) \in \mathbb{R}^{NM}$, $(\mathbf{U}\boldsymbol{\mu}) \in \mathbb{R}^{NM}$ need to be stored, enabling an efficient implementation. It is also worth defining two of the properties of the matrix $\mathbf{B}$, the product of $\mathbf{B}^T\mathbf{y} = \mathbf{1}$, where $\mathbf{y} \in \mathcal{Y}$ and $\mathbf{1}$ is a vector of all ones. The second property of $\mathbf{B}$ is that $\mathbf{B}^T\mathbf{B} = M\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and M is the number of labels. With the dual variables introduced it is now possible to proceed to the formation of the dual problem of (5.2).

Proposition 5.1 (formation of the Lagrange dual).
  1. *The Lagrange dual of* (5.2) *is given as*

(5.11)

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\lambda}{2}\|\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}\|^2 + \langle \mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}, \mathbf{y}^k \rangle$$
$$- \langle \mathbf{1}, \boldsymbol{\beta} \rangle$$
$$\text{s.t. } \gamma_{a:i} \geq 0 \; \forall a \in \{1,\dots,N\} \; \forall i \in \mathcal{L},$$
$$\boldsymbol{\alpha} \in \mathcal{A} = \left\{ \boldsymbol{\alpha} \; \middle| \; \begin{array}{ll} \alpha_{ab:i}^1 + \alpha_{ab:i}^2 = \frac{K_{ab}}{2} & a,b \in \{1,\dots,N\}, a \neq b, i \in \mathcal{L} \\ \alpha_{ab:i}^1, \alpha_{ab:i}^2 \geq 0 & a,b \in \{1,\dots,N\}, a \neq b, i \in \mathcal{L} \end{array} \right\},$$
$$\boldsymbol{\mu} \in \mathcal{U} = \left\{ \boldsymbol{\mu} \; \middle| \; \begin{array}{ll} \sum_{i \in \mathcal{L}} \sum_{\substack{c,d \in \mathcal{R}_p \\ c \neq d}} \mu_{cd:pi}^1 + \mu_{cd:pi}^2 = C_p & p \in \{1,\dots,R\} \\ \mu_{cd:pi}^1, \mu_{cd:pi}^2 \geq 0 & c,d \in \mathcal{R}_p, c \neq d, i \in \mathcal{L}, p \in \{1,\dots,R\} \end{array} \right\}.$$

  2. *The primal variable* $\mathbf{y}$ *satisfies the following:*

$$(5.12) \qquad \mathbf{y} = \lambda(\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}) + \mathbf{y}^k.$$

Proof. A detailed formulation of the Lagrangian and the dual is given in Appendix A.2. ∎

Algorithm 5.1. Proximal minimisation of LP

1: $\mathbf{y}^0 \in \mathcal{Y}$ ▷ Initialise
2: **for** $k \leftarrow 0...K$ **do**
3:    $\mathbf{A}\boldsymbol{\alpha}^0 \leftarrow \mathbf{0}, \mathbf{U}\boldsymbol{\mu}^0 \leftarrow \mathbf{0}, \mathbf{B}\boldsymbol{\beta}^0 \leftarrow \mathbf{0}, \boldsymbol{\gamma}^0 \leftarrow \mathbf{0}$ ▷ Initialise
4:    **for** $t \leftarrow 0...T$ **do**
5:      $(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \leftarrow \mathrm{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\, g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}, \boldsymbol{\gamma})$ ▷ Optimise $\boldsymbol{\beta}^t$ and $\boldsymbol{\gamma}^t$
6:      $\tilde{\mathbf{y}}^t = \lambda(\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t + \mathbf{B}\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}) + \mathbf{y}^k$ ▷ Update feasible solution
7:      $\left(\mathbf{s}_\alpha^t, \mathbf{s}_\mu^t\right) \leftarrow \mathrm{argmin}_{\mathbf{s}_\alpha \in \mathcal{A}, \mathbf{s}_\mu \in \mathcal{U}} \left\langle \left(\mathbf{s}_\alpha, \mathbf{s}_\mu\right), \nabla g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\right\rangle$ ▷ Conditional gradient
8:      $\delta \leftarrow \mathrm{argmin}_\delta\, g(\boldsymbol{\alpha}^t + \delta(\mathbf{s}_\alpha^t - \boldsymbol{\alpha}^t), \boldsymbol{\mu}^t + \delta(\mathbf{s}_\mu^t - \boldsymbol{\mu}^t), \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ ▷ Optimal step size
9:      $\left(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\mu}^{t+1}\right) \leftarrow \left(\boldsymbol{\alpha}^t + \delta(\mathbf{s}_\alpha^t - \boldsymbol{\alpha}^t), \boldsymbol{\mu}^t + \delta(\mathbf{s}_\mu^t - \boldsymbol{\mu}^t)\right)$ ▷ Update
10:    $\mathbf{y}^{k+1} \leftarrow P_{\mathcal{Y}}(\tilde{\mathbf{y}}^t)$ ▷ Project the primal solution onto the feasible set $\mathcal{Y}$

5.2.3. LP minimization algorithm. The dual problem (5.11), in its standard form, can only be tackled using projected gradient descent. However, by separating the variables based on the type of the feasible domains, we are able to formulate an efficient block coordinate descent approach. Each of these blocks is amenable to more sophisticated optimization methods, resulting in a computationally efficient algorithm. As the dual problem is strictly convex and smooth, the optimal solution is still guaranteed. The variables are separated as follows: $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ into one block and $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ into another block, with each block being amenable to more sophisticated optimization algorithms. For $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ the problem decomposes over the random variables. Then with the optimal values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, the minimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ is over a compact domain and can be efficiently tackled using the Frank–Wolfe algorithm [12]. The complete algorithm is summarized in Algorithm 5.1.

Optimizing over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. The values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are efficiently optimized in linear time with the variables $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ fixed as $\boldsymbol{\alpha}^t$ and $\boldsymbol{\mu}^t$. This is achieved via the use of simultaneous equations and the QP minimization algorithm detailed in [36]. Due to the unconstrained nature of $\boldsymbol{\beta}$, the minimum value of the dual objective g is obtained when $\nabla_\beta g(\boldsymbol{\alpha^t}, \boldsymbol{\mu^t}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = 0$ and hence $\boldsymbol{\beta}$ can be derived as a function of $\boldsymbol{\gamma}$.

Proposition 5.2 (optimal for $\boldsymbol{\beta}$).
1. *The optimal value for $\boldsymbol{\beta}$ forms a compact expression given as*

(5.13) $$\boldsymbol{\beta} = -\frac{\mathbf{B}^T}{M}(\mathbf{A}\boldsymbol{\alpha^t} + \mathbf{U}\boldsymbol{\mu^t} + \boldsymbol{\gamma} - \boldsymbol{\phi}).$$

Proof. A detailed formulation of the optimal expression for $\boldsymbol{\beta}$ is given in Appendix A.3. ∎

By substituting the expression for $\boldsymbol{\beta}$ into the dual objective (5.11), a quadratic optimization problem over $\boldsymbol{\gamma}$ is formed. Interestingly, the resulting problem can be optimized independently for each pixel, with each subproblem being an M dimensional QP with nonnegativity constraints.

Proposition 5.3 (optimizing $\boldsymbol{\gamma}$).
1. *The optimal value for $\boldsymbol{\gamma}_a$ is obtained by minimising the following QP:*

(5.14)
$$\min_{\gamma_a} \frac{1}{2}\boldsymbol{\gamma^T}_a \mathbf{Q}\boldsymbol{\gamma}_a + \langle \mathbf{Q}((\mathbf{A}\boldsymbol{\alpha^t}_a) + (\mathbf{U}\boldsymbol{\mu^t})_a - \boldsymbol{\phi}_a) + \mathbf{y}^k, \boldsymbol{\gamma}_a\rangle$$
$$s.t. \; \boldsymbol{\gamma}_a \geq \mathbf{0}.$$

Proof. The derivation of (5.14) is given in Appendix A.4. ∎

Here, $\boldsymbol{\gamma}_a$ denotes the vector $\{\gamma_{a:i}|i \in \mathcal{L}\}$ and $\mathbf{Q} = \lambda(\mathbf{I} - \frac{1}{M})$. For notational simplicity it will be beneficial to write the quadratic program above (5.14) in the following way:

(5.15)
$$\min_{\gamma_a} \quad \frac{1}{2}\boldsymbol{\gamma^T}_a \mathbf{Q}\boldsymbol{\gamma}_a - \langle \mathbf{h}_a, \boldsymbol{\gamma}_a\rangle,$$
$$\mathbf{h}_a = -\mathbf{Q}((\mathbf{A}\boldsymbol{\alpha^t}_a) + (\mathbf{U}\boldsymbol{\mu^t})_a - \boldsymbol{\phi}_a) - \mathbf{y^k}.$$

We optimize each of these subproblems using the iterative method given in [36], as it enables the optimization to remain linear in the number of labels at each iteration. The key stage of the algorithm is the elementwise update equation, which is given by

(5.16)
$$\gamma_{a:i} = \gamma_{a:i}\left[\frac{2(\mathbf{Q}^-\gamma_a)_i + h_{a:i}^+ + c}{(|\mathbf{Q}|\gamma_a)_i + h_{a:i}^- + c}\right],$$

where $\mathbf{Q}^- = \max(-\mathbf{Q}, \mathbf{0})$, $|\mathbf{Q}| = \text{abs}(\mathbf{Q})$, $h_{a:i}^+ = \max(h_{a:i}, 0)$, $h_{a:i}^- = \max(-h_{a:i}, 0)$, and $0 < c \ll 1$. Once an optimal value for $\boldsymbol{\gamma}$ is obtained, the value of $\boldsymbol{\beta}$ can be calculated via (5.13). Note that even though the matrix $\mathbf{Q}$ has M^2 elements, the multiplication by $\mathbf{Q}$ can be performed in $\mathcal{O}(M)$. In particular, the multiplication by $\mathbf{Q}$ can be decoupled to a multiplication by an identity matrix and a matrix of all ones, both of which can be performed in linear time. Similar observations can be made for the matrices $\mathbf{Q}^-$ and $|\mathbf{Q}|$, hence the time complexity of the above update is $\mathcal{O}(M)$. The interested reader is referred to [36] for more information.

Once values for $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are obtained, the values for $\boldsymbol{\gamma}^t$ and $\boldsymbol{\beta}^t$ are fixed as $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Due to the fact that optimization of $\boldsymbol{\gamma}$ decomposes over the number of pixels, and the optimization of each subproblem is linear in the number of labels, the total complexity of the optimization of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ is linear in the number of labels and random variables at each iteration.

Optimizing over $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$. We now turn to the problem of optimizing over $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ given $\boldsymbol{\beta}^t$ and $\boldsymbol{\gamma}^t$. To this end, we use the Frank–Wolfe algorithm [12], which has the advantage of being projection free. Furthermore, we show that the conditional gradient can be computed in a linear complexity and that the step size can be obtained analytically. The time complexity of each iteration of this method is linear in the number of pixels and labels. In practice, the Frank–Wolfe algorithm is only run for a fixed number of iterations.

Conditional gradient computation. With the dual variables fixed at $\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t$ the conditional gradient $\left(\begin{smallmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{smallmatrix}\right)$ is obtained by solving the following:

(5.17)
$$\begin{pmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{pmatrix} \in \underset{\mathbf{s}_\alpha \in \mathcal{A}, \mathbf{s}_\mu \in \mathcal{U}}{\text{argmin}} \left\langle \begin{pmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{pmatrix}, \begin{pmatrix}\nabla_{\boldsymbol{\alpha}}g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \\ \nabla_{\boldsymbol{\mu}}g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\end{pmatrix} \right\rangle.$$

Minimizing this equation to obtain the conditional gradients $\mathbf{s}_\alpha$ and $\mathbf{s}_\mu$ can be neatly summarized by exploiting the properties of the matrices $\mathbf{A}$ and $\mathbf{U}$ given in (5.8) and (5.9), respectively.

Proposition 5.4 (conditional gradient computation).
1. *The conditional gradient* $\mathbf{s}_\alpha$ *is given by*

(5.18)
$$(\mathbf{A}\mathbf{s}_\alpha)_{a:i} = \sum_{b=1}^{N}(K_{ab}\mathbb{1}[\tilde{y}_{a:i}^t \leq \tilde{y}_{b:i}^t] - K_{ab}\mathbb{1}[\tilde{y}_{b:i}^t \leq \tilde{y}_{a:i}^t]).$$

2. *The conditional gradient* $\mathbf{s}_\mu$ *is given by*

(5.19)
$$(\mathbf{U}\mathbf{s}_\mu)_{c:pi} = \begin{cases} C_p & \text{if } \tilde{y}_{c:pi}^t \leq \tilde{y}_{d:pj}^t \ \forall d \in \mathcal{R}_p \backslash c, \ \forall j \in \mathcal{L}, \\ -C_p & \text{if } \tilde{y}_{c:pi}^t \geq \tilde{y}_{d:pj}^t \ \forall d \in \mathcal{R}_p \backslash c, \ \forall j \in \mathcal{L}, \\ 0 & \text{otherwise}, \end{cases}$$

where $\tilde{\mathbf{y}}^\mathbf{t}$ *is the current (infeasible) solution computed using* (5.12).

Proof. Full derivations of the conditional gradients are given in Appendix A.5. ∎

Note that the conditional gradient in (5.18) takes the same form as the subgradient in equation (20) of [10]. This is not a surprising result, as there has been a proven duality relationship between subgradients and conditional gradients for certain problems [3]. The conditional gradient $\mathbf{s}_\alpha$ is obtained via the use of a modified version of the advanced filter-based method with more detail given in Appendix A.7, which reduces (5.18) to a linear complexity. The conditional gradient $\mathbf{s}_\mu$ is obtained via a linear search through all the elements of each clique to find the minimum and the maximum values for $\tilde{y}_{c:i}^t$ in each clique and setting the values to C_p and $-C_p$, respectively. Hence, the resulting complexity of the conditional gradient is linear in the number of variables and labels.

Optimal step size. The performance of any gradient descent based algorithm is fundamentally dependant on the choice of step size. Here, the optimal step size can be computed via minimizing a quadratic function over a single variable, which has a closed form solution. This further improves the efficiency of this method. The optimal step size for the Frank–Wolfe algorithm is obtained by solving

(5.20)
$$\delta = \underset{\delta \in [0,1]}{\arg\min} \, g(\boldsymbol{\alpha}^t + \delta(\mathbf{s}_\alpha^t - \boldsymbol{\alpha}^t), \boldsymbol{\mu}^t + \delta(\mathbf{s}_\mu^t - \boldsymbol{\mu}^t), \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t),$$

which can be obtained analytically and has a closed form solution.

Proposition 5.5 (optimal step size calculation). *The optimal step size to the Frank–Wolfe algorithm is given as*

(5.21)
$$\delta = P_{[0,1]}\left[\frac{\langle \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{A}\mathbf{s}_\mu^t - \mathbf{U}\mathbf{s}_\alpha^t, \mathbf{y}^\mathbf{t} \rangle}{\lambda||\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{A}\mathbf{s}_\mu^t - \mathbf{U}\mathbf{s}_\alpha^t||^2}\right],$$

where $P_{[0,1]}$ *indicates the truncation of the quotient into the interval* $[0,1]$.

Proof. Full derivations of the conditional gradients are given in Appendix A.6. ∎

5.3. Summary. To summarize, our method has the following desirable qualities of an efficient iterative algorithm. With our choice of a quadratic proximal term, the dual of the proximal problem can be efficiently optimized in a blockwise fashion. Specifically, the dual variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are computed efficiently by minimizing a small QP (of dimensions equal the number of labels) for each pixel independently. The remaining dual variables $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ are optimized using the Frank–Wolfe algorithm, where the conditional gradients are computed in linear time, and the optimal step size is obtained analytically. Overall, the time complexity of one iteration of our algorithm is $\mathcal{O}(NM)$ and has no dependence on the number of cliques or their size. This is achieved via again exploiting the filter-based method [1], labeling consistency within a clique and enforcing the intersection between cliques to be an empty set. To the best of our knowledge, this constitutes the first LP minimization algorithm for dense CRFs with sparse higher-order potentials, with a complexity linear in the number of labels and pixels per iteration.

6. Evaluation. This section details the evaluation of the QP and LP implementation outlined in the previous sections; specifically we provide details on datasets, methods, and results. We use semantic segmentation as an example application to demonstrate the low energies achieved by the optimization methods and their ability to tackle dense CRFs whose dimensions match those of images. While we are aware that current state-of-the-art methods focus on increasing the intersection over union (IoU) score [8, 38], we consider this avenue to be tangential to our work and focus primarily on energy minimization to evaluate the methods. The experiments were conducted on the Pascal VOC 2010 dataset [11] and the MSRC dataset [16], which are both standard benchmarks for semantic segmentation.

Pascal contains 1928 color images with dimensions of approximately 500×400 pixels, and 20 classes excluding the background. We split the data in the same way as in [19], that is, 40% for training, 15% for validation, and 45% for testing. We also use the unary potentials from [19] which were trained using the 40% training set. To obtain the kernel and higher-order potential parameters, we use the 15% validation set, with the evaluation performed on the 45% testing set.

MSRC contains 591 images with 21 classes; the dimensions of the images are 320×213 pixels. The labeling ground truths provided in the MSRC dataset are of poor quality as regions around the object are left unlabeled and the boundaries are inaccurate. Hence, the current dataset is not sufficient for performance evaluation; to overcome this Krähenbühl and Koltun [19] manually produced accurate segmentations for 94 images. It is this smaller dataset with accurate ground truths on which we perform the cross validation and tests. We also use the unary potentials from [19] which were not trained on any of the images with accurate ground truths.

We denote the QP and LP implementations as $\mathbf{QP_{clique}}$ and $\mathbf{LP_{clique}}$, respectively. We also performed experiments for the QP and LP without introducing higher-order potentials, i.e., the objective function just consists of a unary and a pairwise potential, which we denote as $\mathbf{QP}$ and $\mathbf{LP}$, respectively. To provide a standard benchmark, we compare our methods against methods that optimize a dense CRF model, namely, the mean-field algorithm [19] and its higher-order variant [34]. We denote these methods as $\mathbf{MF5}$ and $\mathbf{MF5_{clique}}$, which were both run for five iterations, as is often done in practice. All experiments were conducted on a 3.60GHz Intel i7-6850K processor. No GPU parallelization was utilized and the experiments

were performed within a single processing thread. The initial starting points for the algorithms are obtained by minimizing the unary potentials.

6.1. Methods.

Training of unary potentials. The unary potentials were trained using the JointBoost algorithm [33] by Krähenbühl and Koltun [19]. To train the unary potentials for both datasets, 45% of each of the original datasets were used.

Generating cliques. To generate the higher-order potentials we used the mean-shift algorithm [9]. To obtain higher-order potentials that match the problem, we cross validate the minimum region size. We set a spatial and range resolution to 8 and 4, respectively, to avoid cross validating a large number of parameters.

Cross validation of parameters. We use cross validation to find a set of parameters that best represent the semantic segmentation problem. We consider the choice of optimal parameters to be beyond the scope of this work. Due to the long runtime of the LP, we only performed cross validation on the QP and mean-field algorithms. For the $\mathbf{QP_{clique}}$ and $\mathbf{MF5_{clique}}$, eight parameters had to be cross validated—five for the pixel compatibility function (3.7) and then three for the clique potential (3.9). For the $\mathbf{QP}$ and $\mathbf{MF5}$, only five parameters had to be cross validated for the pixel compatibility function (3.7). This was achieved using the Spearmint package [31], which uses Bayesian inference to obtain a set of suitable parameters. The cross-validated parameters are given in Appendix A.8.

6.2. Results.
The collected results provide a quantitative measure of accuracy, energy, and IoU. Accuracy is measured as a percentage of correctly labeled pixels. Energy is the value of the energy function for the resultant labeling. For $\mathbf{QP}$, $\mathbf{LP}$, and $\mathbf{MF5}$ the assignment energy is calculated using only the unary and pairwise terms of (3.11), while $\mathbf{QP_{clique}}$, $\mathbf{LP_{clique}}$, and $\mathbf{MF5_{clique}}$ take the energy function of (3.11). The IoU gives a representation of the proportion of correctly labeled pixels to all pixels taking that class. The optimization process relies on relaxing the constraints on the variables, allowing them to take fractional values. To manifest the fractional solution as an integral solution the, argmax rounding scheme is used, specifically $x_a = \operatorname{argmax}_i(y_{a:i})$.

In order to compare energy values, for $\mathbf{QP}$, $\mathbf{LP}$, and $\mathbf{MF5}$ we used the parameters tuned to $\mathbf{QP}$. For $\mathbf{QP_{clique}}$, $\mathbf{LP_{clique}}$, and $\mathbf{MF5_{clique}}$ the parameters were tuned to $\mathbf{QP_{clique}}$; Table 1 gives the results for all algorithms and Figure 1 shows a decrease in energy at runtime. While $\mathbf{LP}$ and $\mathbf{LP_{clique}}$ could be initialized with a faster algorithm such as $\mathbf{QP}$, we chose to present the results in an "as is" fashion; the interested reader is encouraged to visit [2] for an example of when $\mathbf{LP}$ is initialized with a faster algorithm. Qualitative methods can be seen in Figure 2.

The results given in Table 1 clearly show that the LP relaxations achieve lower energies when compared to their QP counterparts; this is not surprising as the LP relaxation used in this paper is known to give a tighter relaxation than QP [21]. As is consistent with our previous work [2], $\mathbf{LP}$ also obtains lower energies than $\mathbf{MF5}$. For consistency we also performed a set of experiments with the parameters tuned to $\mathbf{MF5_{clique}}$ and $\mathbf{MF5}$, given in Appendix A.9. In this setting, the same pattern is observed where $\mathbf{LP_{clique}}$ achieves lower energies than its $\mathbf{QP}$ and $\mathbf{MF5}$ counterparts. In summary, $\mathbf{QP_{clique}}$ achieves fast initial energy minimization but converges to a local minimum and fails to reach the low energies achieved by $\mathbf{LP_{clique}}$. However, while using continuous relaxations clearly achieves lower energies, it is not apparent as to whether continuous relaxations improve segmentation accuracy. Similarly to our previous

Table 1

Table displaying the average energy, timings, accuracy and IoU, when the parameters are tuned to $\boldsymbol{QP}_{clique}$ and $\boldsymbol{QP}$. It is shown that the lowest energies are achieved by $\boldsymbol{LP}_{clique}$ and $\boldsymbol{LP}$. Interestingly the inclusion of higher-order terms reduces the pixel accuracy but provides a slight increase in IoU score.

Algorithm	Avg.E ($\times 10^7$)	Time(s)	Acc(%)	IoU(%)
		Pascal		
MF5	2.92	**0.7**	79.42	22.21
QP	0.97	9.8	79.51	22.19
LP	**0.62**	236.5	**79.84**	21.80
MF5$_{clique}$	5.20	1.2	79.44	22.22
QP$_{clique}$	3.78	12.4	79.54	22.21
LP$_{clique}$	**2.19**	254.2	79.80	**22.22**
		MSRC		
MF5	58.9	**0.27**	83.79	57.16
QP	29.2	1.06	**83.93**	57.80
LP	**13.8**	54.0	82.93	57.30
MF5$_{clique}$	73.6	0.475	83.404	57.81
QP$_{clique}$	46.1	1.75	83.56	**57.81**
LP$_{clique}$	**44.1**	49.3	81.49	55.81

works [2, 19], the segmentation performance is heavily dependent on the choice of parameters. As such, further work would include investigating the learning of such parameters, possibly in a deep-learning setting as in [38].

7. Discussion. The primary contributions of this paper are a QP and an LP relaxation for minimizing a dense CRF with sparse higher-order potentials. Due to the use of Gaussian pairwise potentials and enforcing labeling consistency in the higher-order terms, each iteration of both algorithms exhibits a time complexity linear in the number of labels and random variables. It is the tightness of the relaxations coupled with the sophistication of the optimization techniques that allows both approaches to achieve lower energies than state-of-the-art methods. Further work would include incorporating the methods into an end-to-end learning framework [38], which would focus on achieving accurate segmentation results as well as low energies.

Appendix A.

A.1. Optimal step size for the Frank–Wolfe algorithm. For an efficient Frank–Wolfe algorithm, an optimal step size is essential and forms one of the three key steps defined in Algorithm 4.1. This section details how the optimal step size is calculated; the optimal step size to the Frank–Wolfe algorithm is achieved by solving

$$(A.1) \qquad \delta^* = \underset{\delta \in [0,1]}{\operatorname{argmin}} f(\mathbf{y} + \delta(\mathbf{s_y} - \mathbf{y}), \mathbf{z} + \delta(\mathbf{s_z} - \mathbf{z})),$$

where

$$(A.2) \qquad \min_{\mathbf{y},\mathbf{z}} f(\mathbf{y},\mathbf{z}) = \min_{\mathbf{y},\mathbf{z}} \left(\boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y} + \mathbf{c}^T \mathbf{z} + (\mathbf{1_z} - \mathbf{z})^T \boldsymbol{CH}(\mathbf{1_y} - \mathbf{y}) \right).$$
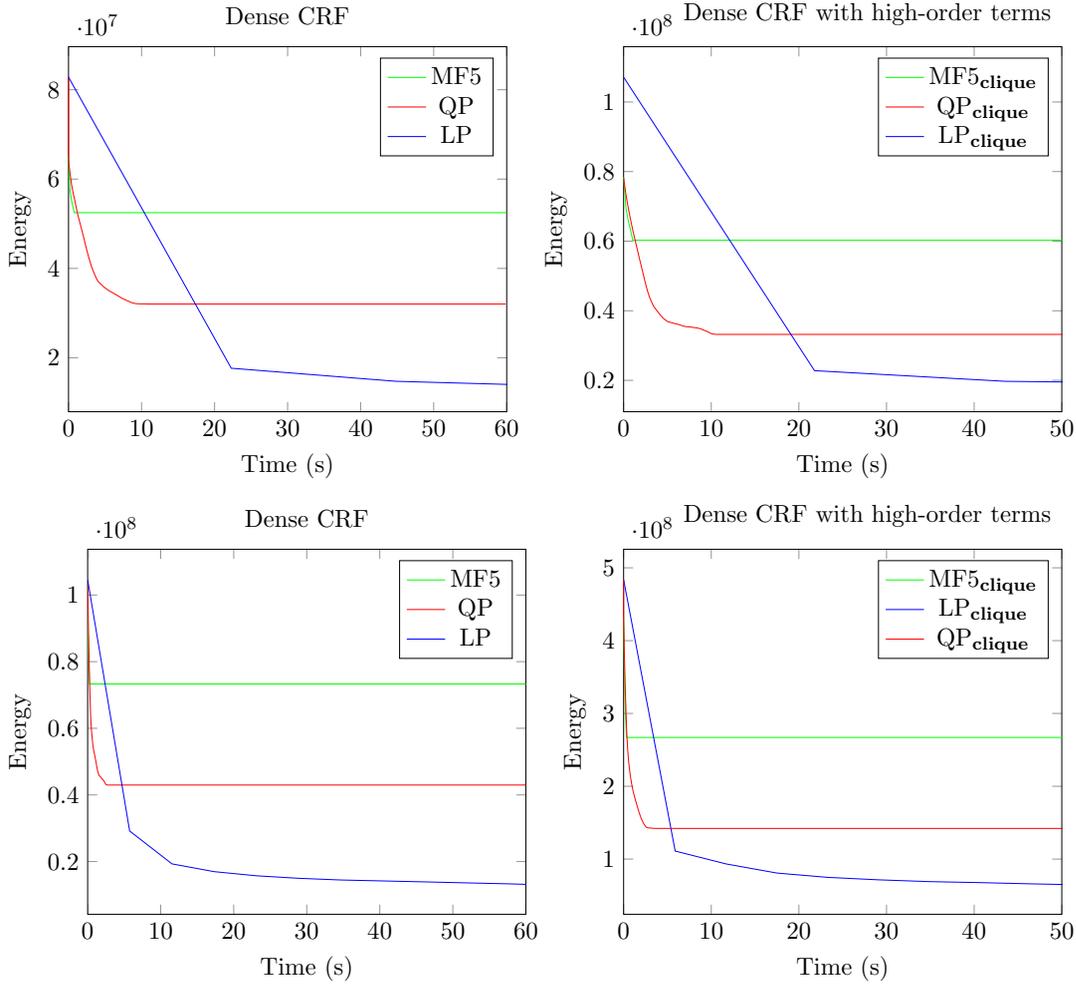
Figure 1. *Assignment energies for the sheep image from Pascal (top row) and small plane image from MSRC (bottom row) as a function of time when the parameters are tuned to $QP_{\mathbf{clique}}$ and QP. The left graphs show the assignment energy calculated using only the unary and pairwise potentials; the right images show the assignment energy of* (3.11), *which consists of the unary, pairwise, and higher-order potentials. It is worth noting the first iteration of $LP_{\mathbf{clique}}$ and LP achieves a lower energy than the final energy of $QP_{\mathbf{clique}}$ and QP, respectively, further highlighting the sophistication of the LP minimization.*

Expanding out (A.1) and collecting terms of δ yields

$$
\begin{aligned}
\delta^* = \operatorname*{argmin}_{\delta \in [0,1]} \Bigg(&\delta^2 \Big((\mathbf{s_y} - \mathbf{y})^T \boldsymbol{\Psi} (\mathbf{s_y} - \mathbf{y}) + (\mathbf{s_z} - \mathbf{z})^T \mathbf{CH} (\mathbf{s_y} - \mathbf{y}) \Big) \\
&+ \delta \Big(\boldsymbol{\phi}^T (\mathbf{s_y} - \mathbf{y}) + \mathbf{2} (\mathbf{s_y} - \mathbf{y})^T \boldsymbol{\Psi} \mathbf{y} + \mathbf{c}^T (\mathbf{s_z} - \mathbf{z}) \\
&\quad - (\mathbf{1_z} - \mathbf{z})^T \mathbf{CH} (\mathbf{s_y} - \mathbf{y}) - (\mathbf{s_z} - \mathbf{z})^T \mathbf{CH} (\mathbf{1_y} - \mathbf{y}) \Big) \\
&+ \Big(\boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y} + \mathbf{c}^T \mathbf{z} + (\mathbf{1_z} - \mathbf{z})^T \mathbf{CH} (\mathbf{1_y} - \mathbf{y}) \Big) \Bigg).
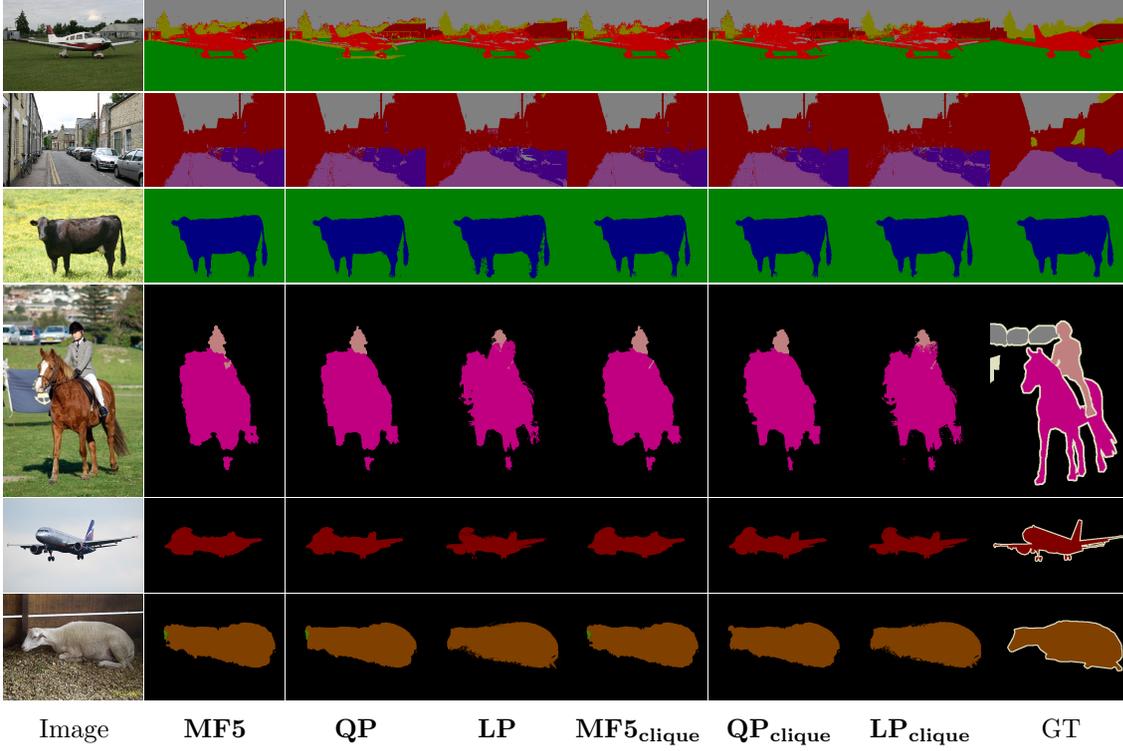\end{aligned}
$$
(A.3)

| Image | **MF5** | **QP** | **LP** | **MF5_{clique}** | **QP_{clique}** | **LP_{clique}** | GT |

Figure 2. *Qualitative results for MSRC (top three) and Pascal (bottom three) with the parameters tuned for* $\mathbf{QP_{clique}}$ *and* $\mathbf{QP}$*. For Pascal,* $\mathbf{LP}$ *achieves the most accurate segmentations; however, this is not the case for MSRC.*

This equation is quadratic in δ and hence the minimum value has a closed form solution given as

$$\delta^* = P_{[0,1]}\left[-\frac{1}{2}\frac{\boldsymbol{\phi}^T(\mathbf{s_y}-\mathbf{y}) + \mathbf{2}(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T(\mathbf{s_z}-\mathbf{z})}{(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}(\mathbf{s_y}-\mathbf{y}) + (\mathbf{s_z}-\mathbf{z})^T\mathbf{CH}(\mathbf{s_y}-\mathbf{y})} \right.$$

(A.4)

$$\left. + \frac{1}{2}\frac{(\mathbf{1_z}-\mathbf{z})^T\mathbf{CH}(\mathbf{s_y}-\mathbf{y}) - (\mathbf{s_z}-\mathbf{z})^T\mathbf{CH}(\mathbf{1_y}-\mathbf{y})}{(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}(\mathbf{s_y}-\mathbf{y}) + (\mathbf{s_z}-\mathbf{z})^T\mathbf{CH}(\mathbf{s_y}-\mathbf{y})} \right],$$

and for the Frank–Wolfe algorithm, is the optimal step size. $P_{[0,1]}$ indicates that if the value of δ^* falls outside of the range $[0,1]$, then the optimal step size is truncated to lie within this range.

A.2. Formulation of the Lagrange dual for the LP. Starting with the primal problem, which is given as

$$(A.5) \quad \min_{\mathbf{y},\mathbf{v},\mathbf{w}} \sum_{a=1}^{N}\sum_{i\in\mathcal{L}}\phi_{a:i}y_{a:i} + \sum_{a=1}^{N}\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}\frac{\mathrm{K}_{ab}}{2}v_{ab:i} + \sum_{p=1}^{R}C_p w_p + \frac{1}{2\lambda}||\mathbf{y}-\mathbf{y}^k||^2$$

$$\begin{aligned}
\text{s.t. } v_{ab:i} &\geq y_{a:i}-y_{b:i} & \forall a,b\in\{1,\ldots,N\} \quad & a\neq b \quad & \forall i\in\mathcal{L},\\
v_{ab:i} &\geq y_{b:i}-y_{a:i} & \forall a,b\in\{1,\ldots,N\} \quad & a\neq b \quad & \forall i\in\mathcal{L},\\
w_p &\geq y_{c:pi}-y_{d:pi} & \forall c,d\in\mathcal{R}_p \quad & c\neq d \quad & \forall i\in\mathcal{L} \quad \forall p\in\{1,\ldots,R\},\\
w_p &\geq y_{d:pi}-y_{c:pi} & \forall c,d\in\mathcal{R}_p \quad & c\neq d \quad & \forall i\in\mathcal{L} \quad \forall p\in\{1,\ldots,R\},\\
y_{a:i} &\geq 0 & \forall a\in\{1,\ldots,N\} \quad & & \forall i\in\mathcal{L},\\
\sum_{i\in\mathcal{L}}y_{a:i} &= 1 & \forall a\in\{1,\ldots,N\}. &
\end{aligned}$$

The associated Lagrangian can thus be written as

$$(A.6) \quad \max_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma}}\min_{\mathbf{y},\mathbf{w},\mathbf{v}} L(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v})$$

$$= \sum_{a=1}^{N}\sum_{i\in\mathcal{L}}\phi_{a:i}y_{a:i} + \sum_{a=1}^{N}\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}\frac{\mathrm{K}_{ab}}{2}v_{ab:i} + \sum_{p=1}^{R}C_p w_p + \frac{1}{2\lambda}||\mathbf{y}-\mathbf{y}^k||$$

$$- \sum_{a=1}^{N}\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}\alpha^1_{ab:i}(y_{b:i}-y_{a:i}+v_{ab:i}) - \sum_{a=1}^{N}\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}\alpha^2_{ab:i}(y_{a:i}-y_{b:i}+v_{ab:i})$$

$$- \sum_{p=1}^{R}\sum_{\substack{c,d\in\mathcal{R}_p\\c\neq d}}\sum_{i\in\mathcal{L}}\mu^1_{cd:pi}(y_{d:i}-y_{c:i}+w_p) - \sum_{p=1}^{R}\sum_{\substack{c,d\in\mathcal{R}_p\\c\neq d}}\sum_{i\in\mathcal{L}}\mu^2_{cd:pi}(y_{c:i}-y_{d:i}+w_p)$$

$$+ \sum_a \beta_a\left(1-\sum_{i\in\mathcal{L}}y_{a:i}\right) - \sum_a\sum_{i\in\mathcal{L}}\gamma_{a:i}y_{a:i}$$

$$\begin{aligned}
\text{s.t. } \quad \alpha^1_{ab:i},\alpha^2_{ab:i} &\geq 0 & \forall a,b\in\{1,\ldots,N\} \quad & a\neq b \quad & \forall i\in\mathcal{L},\\
\mu^1_{cd:pi},\mu^2_{cd:i} &\geq 0 & \forall c,d\in\mathcal{R}_p \quad & c\neq d \quad & \forall i\in\mathcal{L} \quad \forall p\in\{1,\ldots,R\},\\
\gamma_{a:i} &\geq 0 & \forall a\in\{1,\ldots,N\} \quad & & \forall i\in\mathcal{L}.
\end{aligned}$$

Here $\alpha^1_{ab:i}$, $\alpha^2_{ab:i}$, $\mu^1_{cd:pi}$, $\mu^2_{cd:pi}$, β_a, and $\gamma_{a:i}$ are the Lagrange variables. To obtain the dual problem, the Lagrangian needs to be minimized over the primal variables $\mathbf{y},\mathbf{w},\mathbf{v}$. When the derivatives of the Lagrangian with respect to $\mathbf{w}$ and $\mathbf{v}$ are nonzero, the problem is unbounded and hence the minimization yields a value of $-\infty$. To this extent, the derivatives of the Lagrangian with respect to $\mathbf{w}$ and $\mathbf{v}$ must be zero for a bounded solution. These conditions are instrumental in obtaining constraints on the Lagrange multipliers $\alpha^1_{ab:i},\alpha^2_{ab:i}$, and $\mu^1_{cd:pi},\mu^2_{cd:pi}$. By rearranging $\nabla_{\mathbf{v}}L(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v})=0$ and $\nabla_L\mathbf{w}(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v})=0$ the constraints for the Lagrange multipliers $\alpha^1_{ab:i},\alpha^2_{ab:i}$, and $\mu^1_{cd:pi},\mu^2_{cd:pi}$ are obtained and given respectively as

(A.7) $$\alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{K_{ab}}{2} \qquad \forall a,b \in \{1,\dots,N\} \qquad a \neq b \qquad \forall i \in \mathcal{L},$$

(A.8) $$\sum_{i\in\mathcal{L}}\sum_{c,d}\mu^1_{cd:pi} + \mu^2_{cd:pi} = C_p \qquad \forall c,d \in \mathcal{R}_p \qquad c \neq d \qquad \forall i \in \mathcal{L}.$$

By differentiating the Lagrangian with respect to $\mathbf{y}$ and setting the derivative to zero, an equation for the primal variables can be obtained. Before solving $\nabla_{\mathbf{y}} L(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = 0$, it is beneficial to reorder the terms in the Lagrangian, using (A.7) and (A.8), we can arrange the Lagrangian as follows:

(A.9) $$L(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = \sum_{a=1}^{N}\sum_{i\in\mathcal{L}}(\phi_{a:i}-\beta_a-\gamma_a)y_{a:i} + \frac{1}{2\lambda}\sum_{a=1}^{N}\sum_{i\in\mathcal{L}}(y_{a:i}-y^k_{a:i})^2 + \sum_{a=1}^{N}\beta_a$$

$$+ \sum_{a=1}^{N}\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}(\alpha^1_{ab:i}-\alpha^2_{ab:i}-\alpha^1_{ba:i}+\alpha^2_{ba:i})y_{a:i}$$

$$+ \sum_{p=1}^{R}\sum_{\substack{c,d\in\mathcal{R}_p\\c\neq d}}\sum_{i\in\mathcal{L}}(\mu^1_{cd:pi}-\mu^2_{cd:pi}-\mu^1_{dc:pi}+\mu^2_{dc:pi})y_{c:i}.$$

From this, differentiating the Lagrangian with respect to $\mathbf{y}$ is a trivially achieved:

(A.10) $$\frac{1}{\lambda}(y_{a:i}-y^k_{a:i}) = -\sum_{\substack{b=1\\b\neq a}}^{N}\sum_{i\in\mathcal{L}}(\alpha^1_{ab:i}-\alpha^2_{ab:i}-\alpha^1_{ba:i}+\alpha^2_{ba:i}) + \beta_a + \gamma_{a:i}$$

$$- \sum_{p=1}^{R}\sum_{\substack{d\in\mathcal{R}_p\\a\neq d}}(\mu^1_{ad:pi}-\mu^2_{ad:pi}-\mu^1_{da:pi}+\mu^2_{da:pi}) - \phi_{a:i}.$$

By utilizing the matrices introduced in (5.8), (5.9), and (5.10) this expression can be concisely written in vector form:

(A.11) $$\mathbf{y} = \lambda(\mathbf{A}\boldsymbol{\alpha}+\mathbf{U}\boldsymbol{\mu}+\mathbf{B}\boldsymbol{\beta}+\boldsymbol{\gamma}-\boldsymbol{\phi}) + \mathbf{y}^k.$$

Substituting this equation into the Lagrangian defined in (A.9) yields the following Lagrange dual problem:

(A.12)

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\lambda}{2}\|\mathbf{A}\boldsymbol{\alpha}+\mathbf{U}\boldsymbol{\mu}+\mathbf{B}\boldsymbol{\beta}+\boldsymbol{\gamma}-\boldsymbol{\phi}\|^2 + \langle\mathbf{A}\boldsymbol{\alpha}+\mathbf{U}\boldsymbol{\mu}+\mathbf{B}\boldsymbol{\beta}+\boldsymbol{\gamma}-\boldsymbol{\phi},\mathbf{y}^k\rangle$$

$$- \langle\mathbf{1},\boldsymbol{\beta}\rangle$$

$$\text{s.t.} \quad \gamma_{a:i} \geq 0 \ \forall a \in \{1,\dots,N\} \ \forall i \in \mathcal{L},$$

$$\boldsymbol{\alpha} \in \mathcal{A} = \left\{\boldsymbol{\alpha} \ \middle| \ \begin{array}{ll} \alpha^1_{ab:i}+\alpha^2_{ab:i}=\frac{K_{ab}}{2}, & a,b\in\{1,\dots,N\}, a\neq b, i\in\mathcal{L} \\ \alpha^1_{ab:i},\alpha^2_{ab:i}\geq 0, & a,b\in\{1,\dots,N\}, a\neq b, i\in\mathcal{L}\end{array}\right\},$$

$$\boldsymbol{\mu} \in \mathcal{U} = \left\{\boldsymbol{\mu} \ \middle| \ \begin{array}{ll} \sum_{i\in\mathcal{L}}\sum_{\substack{c,d\in\mathcal{R}_p\\c\neq d}}\mu^1_{cd:pi}+\mu^2_{cd:pi}=C_p, & p\in\{1,\dots,R\} \\ \mu^1_{cd:pi},\mu^2_{cd:pi}\geq 0, & c,d\in\mathcal{R}_p, c\neq d, i\in\mathcal{L}, p\in\{1,\dots,R\}\end{array}\right\}.$$

A.3. Closed form expression for $\boldsymbol{\beta}$. Due to the unconstrained nature of $\boldsymbol{\beta}$, the minimum value of the dual objective g is obtained when $\nabla_{\boldsymbol{\beta}} g(\boldsymbol{\alpha^t}, \boldsymbol{\mu^t}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = 0$ and hence $\boldsymbol{\beta}$ can be derived as a function of $\boldsymbol{\gamma}$. Using the fact that $\mathbf{B}^T \mathbf{y}^k = \mathbf{1}$, $\nabla_{\boldsymbol{\beta}} g(\cdot)$ can be written as

$$(A.13) \qquad \nabla_{\boldsymbol{\beta}} g(\boldsymbol{\alpha^t}, \boldsymbol{\mu^t}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \lambda \mathbf{B}^T (\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}).$$

Using $\mathbf{B}^T \mathbf{B} = M\mathbf{I}$ and the fact that λ is a constant, an expression for $\boldsymbol{\beta}$ can be given as

$$(A.14) \qquad \boldsymbol{\beta} = -\frac{\mathbf{B}^T}{M}(\mathbf{A}\boldsymbol{\alpha^t} + \mathbf{U}\boldsymbol{\mu^t} + \boldsymbol{\gamma} - \boldsymbol{\phi}).$$

A.4. Quadratic program for $\boldsymbol{\gamma}_a$. By substituting the expression for $\boldsymbol{\beta}$ into the dual objective (5.11), a quadratic optimisation problem over $\boldsymbol{\gamma}$ is formed:

$$(A.15) \qquad \min_{\boldsymbol{\gamma}} g(\boldsymbol{\alpha^t}, \boldsymbol{\mu^t}, \boldsymbol{\gamma}) = \frac{\lambda}{2} ||\mathbf{D}\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \boldsymbol{\gamma} - \boldsymbol{\phi})||^2 + \langle \mathbf{D}(\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \boldsymbol{\gamma} - \boldsymbol{\phi}), \mathbf{y}^k \rangle$$
$$+ \frac{\mathbf{B}^T}{M} \langle \mathbf{1}, \mathbf{A}\boldsymbol{\alpha^t} + \mathbf{U}\boldsymbol{\mu^t} + \boldsymbol{\gamma} - \boldsymbol{\phi} \rangle,$$

where $\mathbf{D} = \boldsymbol{I} - \frac{\mathbf{B}\mathbf{B}^T}{M}$. Using the fact that $\mathbf{B}^T \mathbf{y}^k = \mathbf{1}$, the identity $\mathbf{D}\mathbf{D}^T = \mathbf{D}$, and removing constant terms, the optimization problem over $\boldsymbol{\gamma}$ can be simplified:

$$(A.16) \qquad \min_{\boldsymbol{\gamma}} \quad \frac{1}{2}\boldsymbol{\gamma}^T \lambda \mathbf{D}\boldsymbol{\gamma} + \langle \lambda \mathbf{D}((\mathbf{A}\boldsymbol{\alpha^t}) + (\mathbf{U}\boldsymbol{\mu^t}) - \boldsymbol{\phi}) + \mathbf{y}^k, \boldsymbol{\gamma} \rangle$$
$$\text{s.t.} \quad \boldsymbol{\gamma} \geq \mathbf{0}.$$

Due to the fact that $\boldsymbol{D}$ is a block diagonal, the resulting problem can be written as a sum of quadratic programs:

$$(A.17) \qquad \min_{\boldsymbol{\gamma} \geq 0} g(\boldsymbol{\alpha^t}, \boldsymbol{\mu^t}, \boldsymbol{\gamma}) = \sum_a \min_{\boldsymbol{\gamma} \geq 0} \frac{1}{2}\boldsymbol{\gamma}^T{}_a \mathbf{Q}_a \boldsymbol{\gamma}_a + \langle \mathbf{Q}_a((\mathbf{A}\boldsymbol{\alpha^t})_a + (\mathbf{U}\boldsymbol{\mu^t})_a - \boldsymbol{\phi}_a) + \mathbf{y}^k_a, \boldsymbol{\gamma}_a \rangle$$
$$\text{s.t.} \quad \boldsymbol{\gamma}_a \geq \mathbf{0},$$

which can be optimized independently:

$$(A.18) \qquad \min_{\boldsymbol{\gamma}_a} \quad \frac{1}{2}\boldsymbol{\gamma}^T{}_a \mathbf{Q}_a \boldsymbol{\gamma}_a + \langle \mathbf{Q}_a((\mathbf{A}\boldsymbol{\alpha^t}_a) + (\mathbf{U}\boldsymbol{\mu^t})_a - \boldsymbol{\phi}_a) + \mathbf{y}^k_a, \boldsymbol{\gamma}_a \rangle$$
$$\text{s.t.} \quad \boldsymbol{\gamma}_a \geq \mathbf{0}.$$

Here, $\boldsymbol{\gamma}_a$ denotes the vector $\{\gamma_{a:i} | i \in \mathcal{L}\}$ and $\mathbf{Q} = \lambda(\boldsymbol{I} - \frac{1}{M}) \in \mathbb{R}^{M \times M}$. Thus the resulting optimization problem decomposes to N subproblems, with each subproblem being an M dimensional QP.

A.5. Derivation of the conditional gradient for $\mathbf{s}_\alpha$ and $\mathbf{s}_\mu$. As previously stated, efficient conditional gradient computation is critical to a well-performing Frank–Wolfe algorithm. This appendix details the method for computing the conditional gradient in linear time. Attention is drawn to the computation of the conditional gradient $\mathbf{s}_\alpha$, in which a modified version of the advanced filtering method detailed in section 3.1.4 is used. A summary is provided in Appendix A.7 but the interested reader is referred to [2] for more information.

Derivation of the conditional gradient for $\mathbf{s}_\alpha$. With the dual variables fixed at $\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t$, the conditional gradient with respect to $\boldsymbol{\alpha}$ is obtained via solving the following:

$$(A.19) \qquad \mathbf{s}_\alpha = \underset{\mathbf{s}_\alpha \in \mathcal{A}}{\operatorname{argmin}} \langle \mathbf{s}_\alpha, \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \rangle.$$

By using (A.11), $\nabla_{\boldsymbol{\alpha}} g(.)$ is given as

$$(A.20) \qquad \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \mathbf{A}^T \tilde{\mathbf{y}}^t.$$

Note that the feasible set $\mathcal{A}$ is separable and can be written as $\mathcal{A} = \prod_{a, b \neq a, i} \mathcal{A}_{ab:i}$, where $\mathcal{A}_{ab:i} = \{(\alpha^1_{ab:i}, \alpha^2_{ab:i}) | \alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{1}{2} K_{ab}, \alpha^1_{ad:i}, \alpha^2_{ab:i} \geq 0\}$. It is possible to exploit this separability and compute the conditional gradient $\mathbf{s}_\alpha$ for each Lagrange multiplier as follows:

$$(A.21) \qquad \min_{s_{\alpha^1_{ab:i}}, s_{\alpha^2_{ab:i}}} s_{\alpha^1_{ab:i}} \nabla_{\alpha^1_{ad:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) + s_{\alpha^2_{ab:i}} \nabla_{\alpha^2_{ad:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$$

$$\text{s.t.} \qquad s_{\alpha^1_{ab:i}}, s_{\alpha^2_{ab:i}} \in \mathcal{A}_{ab:i}.$$

The derivatives $\nabla_{\alpha^1_{ad:i}} g(\cdot)$ and $\nabla_{\alpha^2_{ad:i}} g(\cdot)$ can be easily computed to yield the following:

$$(A.22) \qquad \nabla_{\alpha^1_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{b:i} - \tilde{y}^t_{a:i},$$

$$(A.23) \qquad \nabla_{\alpha^2_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{a:i} - \tilde{y}^t_{b:i},$$

where the reader is reminded that $\tilde{y}^t_{a:i}$ represents the current infeasible solution, as detailed in step 6 of Algorithm 5.1. Hence, the minimum is given as

$$(A.24) \qquad s_{\alpha^1_{ab:i}} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \geq \tilde{y}^t_{b:i}, \\ 0 & \text{otherwise,} \end{cases}$$

$$(A.25) \qquad s_{\alpha^2_{ab:i}} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}, \\ 0 & \text{otherwise,} \end{cases}$$

which by utilizing matrix $\mathbf{A}$, introduced in (5.8), can be concisely written as

$$(A.26) \qquad (\mathbf{A} \mathbf{s}_\alpha)_{a:i} = \sum_b (K_{ab} \mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}] - K_{ab} \mathbb{1}[\tilde{y}^t_{b:i} \leq \tilde{y}^t_{a:i}]).$$

Derivation of the conditional gradient $\mathbf{s}_\mu$. Similarly to $\mathbf{s}_\mu$ the conditional gradient of $\boldsymbol{\mu}$ at $\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t$ is obtained by via solving the following:

$$(A.27) \qquad \mathbf{s}_\mu = \underset{\mathbf{s}_\mu \in \mathcal{U}}{\operatorname{argmin}} \langle \mathbf{s}_\mu, \nabla_{\boldsymbol{\mu}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \rangle.$$

By using (A.11), $\nabla_{\boldsymbol{\mu}} g(\cdot)$ is given as

$$(A.28) \qquad \nabla_{\boldsymbol{\mu}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \mathbf{U}^T \tilde{\mathbf{y}}^t.$$

The set $\mathcal{U}$ can only be separated according to the number of cliques, $\mathcal{U} = \prod_p \mathcal{U}_p$, where $\mathcal{U}_p = \{(\mu^1_{cd:pi}, \mu^2_{cd:pi}) | \sum_{i \in \mathcal{L}} \sum_{c,d} \mu^1_{cd:pi} + \mu^2_{cd:pi} = C_p, \mu^1_{cd:pi}, \mu^2_{cd:pi} \geq 0, \forall c, d \neq c, \forall i \in \mathcal{L}\}$. The conditional gradient for each set $\mathcal{U}_p$ can be written as

$$(A.29) \qquad \min_{s_{\mu^1_{cd:pi}}, s_{\mu^2_{cd:pi}}} s_{\mu^1_{cd:pi}} \nabla_{\mu^1_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) + s_{\mu^2_{cd:pi}} \nabla_{\mu^2_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$$

$$\text{s.t.} \qquad s_{\mu^1_{cd:pi}}, s_{\mu^2_{cd:pi}} \in \mathcal{U}_p.$$

The derivatives $\nabla_{\mu^1_{cd:pi}} g(.)$ and $\nabla_{\mu^2_{cd:pi}} g(.)$ can be easily computed to yield the following:

$$(A.30) \qquad\qquad\qquad \nabla_{\mu^1_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{d:pi} - \tilde{y}^t_{c:pi},$$

$$(A.31) \qquad\qquad\qquad \nabla_{\mu^2_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{c:pi} - \tilde{y}^t_{d:pi},$$

where the reader is reminded that $\tilde{y}^t_{a:i}$ represents the current infeasible solution, as detailed in step 6 of Algorithm 5.1. Given $s_{\mu^1_{cd:pi}}, s_{\mu^2_{cd:pi}} \in \mathcal{U}_p$, the conditional gradients are thus given as

$$(A.32) \qquad s_{\mu^1_{cd:pi}} = \begin{cases} C_p/2 & \text{if } \tilde{y}^t_{c:pi} = \max_{c \in p, i \in \mathcal{L}} \tilde{y}^t_{c:pi} \text{ and } \tilde{y}^t_{d:i} = \min_{d \in p, i \in \mathcal{L}} \tilde{y}^t_{d:pi}, \\ 0 & \text{otherwise}, \end{cases}$$

$$(A.33) \qquad s_{\mu^2_{cd:pi}} = \begin{cases} C_p/2 & \text{if } \tilde{y}^t_{c:pi} = \min_{c \in p, i \in \mathcal{L}} \tilde{y}^t_{c:pi} \text{ and } \tilde{y}^t_{d:i} = \max_{d \in p, i \in \mathcal{L}} \tilde{y}^t_{d:pi}, \\ 0 & \text{otherwise}. \end{cases}$$

By utilizing the matrix $\mathbf{U}$ the conditional gradient of $\mathbf{s}_\mu$ can be written as

$$(A.34) \qquad (\mathbf{U}\mathbf{s}_\mu)_{c:pi} = \begin{cases} C_p & \text{if } \tilde{y}^t_{c:pi} \leq \tilde{y}^t_{d:pj} \qquad \forall d \in p \qquad \forall j \in \mathcal{L}, \\ -C_p & \text{if } \tilde{y}^t_{c:pi} > \tilde{y}^t_{d:pj} \qquad \forall d \in p \qquad \forall j \in \mathcal{L}, \\ 0 & \text{otherwise}. \end{cases}$$

A.6. Optimal step size. We need to find the step size δ that gives the maximum decrease in the objective function g given the descent direction s^t. This can be formalized as the following optimization problem:

$$(A.35) \qquad \min_\delta \quad \frac{\lambda}{2} \left\| \mathbf{A}\boldsymbol{\alpha}^t + \delta(\mathbf{A}\mathbf{s}^t_\alpha - \mathbf{A}\boldsymbol{\alpha}^t) + \mathbf{U}\boldsymbol{\mu}^t + \delta(\mathbf{U}\mathbf{s}^t_\mu - \mathbf{U}\boldsymbol{\mu}^t) + \mathbf{B}\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}^t \right\|^2$$

$$+ \left\langle \mathbf{A}\boldsymbol{\alpha}^t + \delta(\mathbf{A}\mathbf{s}^t_\alpha - \mathbf{A}\boldsymbol{\alpha}^t) + \mathbf{U}\boldsymbol{\mu}^t + \delta(\mathbf{U}\mathbf{s}^t_\mu - \mathbf{U}\boldsymbol{\mu}^t) + \mathbf{B}\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}^t, \mathbf{y}^k \right\rangle$$

$$- \langle \mathbf{1}, \boldsymbol{\beta}^t \rangle$$

$$\text{s.t.} \quad \delta \in [0, 1] .$$

Note that the above function is optimized over the scalar variable δ and the minimum is attained when the derivative is zero. Hence setting the derivate to zero,

(A.36)

$$0 = \left\langle \mathbf{y}^k, \mathbf{As}_\alpha^t - \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{Us}_\mu^t - \mathbf{U}\boldsymbol{\mu}^t \right\rangle$$

$$+ \lambda \left\langle (1-\delta)\mathbf{A}\boldsymbol{\alpha}^t + \delta\mathbf{As}_\alpha^t + (1-\delta)\mathbf{U}\boldsymbol{\mu}^t + \delta\mathbf{Us}_\mu^t + \mathbf{B}\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}^t, \mathbf{As}_\alpha^t - \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{Us}_\mu^t - \mathbf{U}\boldsymbol{\mu}^t \right\rangle$$

$$\delta = \frac{\langle \mathbf{As}_\alpha^t - \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{Us}_\mu^t - \mathbf{U}\boldsymbol{\mu}^t, \lambda\left(\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t + \mathbf{B}\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}^t + \mathbf{y}^k\right)\rangle}{\lambda\|\langle \mathbf{As}_\alpha^t - \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{Us}_\mu^t - \mathbf{U}\boldsymbol{\mu}^t\|^2},$$

$$\delta = P_{[0,1]}\left[\frac{\langle \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{As}_\mu^t - \mathbf{Us}_\alpha^t, \mathbf{y}^t\rangle}{\lambda\|\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{As}_\mu^t - \mathbf{Us}_\alpha^t\|^2}\right].$$

In fact, if the optimal δ is out of the interval $[0,1]$ it is simply projected back.

A.7. Filtering method appendix.

Original filtering algorithm. Let us first introduce some notation below. We denote the set of lattice points of the original permutohedral lattice with $\mathcal{P}$ and the neighboring feature points of lattice point l by $N(l)$. Also we denote the neighboring lattice points of a feature point a by $\overline{N}(a)$. In addition, the barycentric weight between the lattice point l and feature point b is denoted with w_{lb}. Furthermore, the value at feature point b is denoted with v_b and the value at lattice point l is denoted with $\overline{v}_l$. The pseudocode of the algorithm is given in Algorithm A.1

Algorithm A.1. Original filtering algorithm [1]

Require: Permutohedral lattice $\mathcal{P}$
  for all $l \in P$ **do**
    $\overline{v}_l \leftarrow \sum_{b \in N(l)} w_{lb} v_b$ ▷ Splatting
  $\overline{V}' \leftarrow k \otimes \overline{V}$ ▷ Blurring
  for all $a \in \{1, \dots, N\}$ **do**
    $v_a' \leftarrow \sum_{l \in \overline{N}(a)} w_{la} \overline{v}_l'$ ▷ Slicing

---

Modified filtering algorithm. As mentioned in the main paper the interval $[0,1]$ is discretized into H discrete bins. Note that each bin $h \in \{0 \dots H\}$ is associated with a probability interval which is identified as $\left[\frac{h}{H-1}, \frac{h+1}{H-1}\right)$. To this end, the bin h_b of the feature point b satisfy the following inequality: $\left[\frac{h_b}{H-1} \leq y_b < \frac{h_b+1}{H-1}\right]$.

Furthermore, at the splatting step, the values v_b are accumulated to its neighboring lattice point only if the lattice point is above or equal to the feature point level. Formally, the barycentric interpolation at lattice point l at level h can be written as

(A.37)
$$\overline{v}_{l:h} = \sum_{\substack{b \in N(l) \\ h \geq h_b}} w_{lb} v_b = \sum_{b \in N(l)} w_{lb} v_b \mathbb{1}\left[\frac{h}{H-1} \geq y_b\right],$$

where h_b is the level of feature point b and w_{lb} is the barycentric weight between lattice point l and feature point b. Then blurring is performed independently at each discrete level h. Finally, at the slicing step, the resulting values are interpolated at the level of the feature point. Our modified algorithm is given in Algorithm A.2.

Algorithm A.2. Modified filtering algorithm

Require: Permutohedral lattice $\mathcal{P}$, discrete levels H

  for all $l \in \mathcal{P}$ **do**   ▷ Splatting

    for all $h \in \{0 \ldots H-1\}$ **do**

$$\overline{v}_{l:h} \leftarrow \sum_{b \in N(l)} w_{lb}\, v_b \mathbb{1}\left[\frac{h}{H-1} \geq y_b\right]$$

  for all $h \in \{0 \ldots H-1\}$ **do**

$$\overline{V}'_h \leftarrow k \otimes \overline{V}_h \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ▷ \text{Blurring}$$

  for all $a \in \{1, \ldots, N\}$ **do**   ▷ Slicing

$$v'_a \leftarrow \sum_{l \in \overline{N}(a)} \sum_{h=0}^{H-1} w_{la}\, \overline{v}'_{l:h} \mathbb{1}\left[\frac{h}{H-1} \leq y_a < \frac{h+1}{H-1}\right]$$

Note that the above algorithm is given for the $\mathbb{1}[y_a \geq y_b]$ constraint; however, it is fairly easy to modify it for the $\mathbb{1}[y_a \leq y_b]$ constraint. In particular, one needs to change the interval identified by the bin h to $\left(\frac{h-1}{H-1}, \frac{h}{H-1}\right]$. Using this fact, one can easily derive the splatting and slicing equations for the $\mathbb{1}[y_a \leq y_b]$ constraint.

A.8. Cross-validated parameters. A table of the cross validated parameters is given in Table 2.

A.9. Additional results. In this section we present the results for when the algorithms are tuned to $\mathbf{MF5_{clique}}$ and $\mathbf{MF}$, displayed in Table 3. As is consistent with our previous results, it can be seen that $\mathbf{QP}$ achieves lower energies than $\mathbf{MF5}$ but fails to reach the low energies of $\mathbf{LP}$. A similar pattern can be seen for the higher-order potentials, where $\mathbf{LP_{clique}}$ obtains lower energies than $\mathbf{QP_{clique}}$. Visual results can be seen in Figure 3.

Table 2

Table of cross-validated parameters for each of the above algorithms.

Algorithm	$\sigma^{(3)}$	$w^{(2)}$	$\sigma^{(1)}$	$\sigma^{(2)}$	$w^{(1)}$	*cliquesize*	Γ	η
				MSRC				
MF5	4.10	77.047	47.79	4.69	100	-	-	-
QP	60.5	6.53	22.89	48.73	4.13	-	-	-
MF5$_\mathbf{clique}$	6.53	4.46	50	9.74	11.56	10	54.88	876.08
QP$_\mathbf{clique}$	3.74	17.67	39.76	9.49	54.56	100	19.71	109.0
				Pascal				
MF5	1.00	29.19	17.82	6.14	32.56	-	-	-
QP	1.00	100	19.11	6.08	55.19	-	-	-
MF5$_\mathbf{clique}$	1.20	76.38	16.32	38.10	1.45	27	20.71	467.36
QP$_\mathbf{clique}$	1.00	99.53	13.30	7.89	100.00	97	100	139.70

Table displaying the average energy, timings, accuracy, and IoU when the parameters are tuned to $MF5_{clique}$ and $MF5$. It is shown that the lowest energies are achieved by LP_{clique} and LP. Interestingly, LP obtains the greatest segmentation accuracy for Pascal, despite tuning the parameters for $MF5_{clique}$ and $MF5$.

Algorithm	**Avg.E** $(\times 10^7)$	**Time(s)**	**Acc(%)**	**IoU(%)**
		Pascal		
MF5	5.26	**0.35**	79.54	**22.23**
QP	4.13	1.06	79.63	22.23
LP	**1.17**	54.0	**79.84**	21.91
MF5$_{clique}$	5.20	1.75	79.26	22.22
QP$_{clique}$	3.78	1.75	79.25	22.21
LP$_{clique}$	**2.19**	79.67	79.67	21.35
		MSRC		
MF5	17.7	**0.37**	83.79	57.16
QP	12.2	0.58	83.93	57.80
LP	**0.39**	67.9	82.93	57.30
MF5$_{clique}$	0.34	0.62	84.30	**60.53**
QP$_{clique}$	0.30	1.13	**84.41**	60.32
LP$_{clique}$	**0.12**	46.9	81.89	55.66

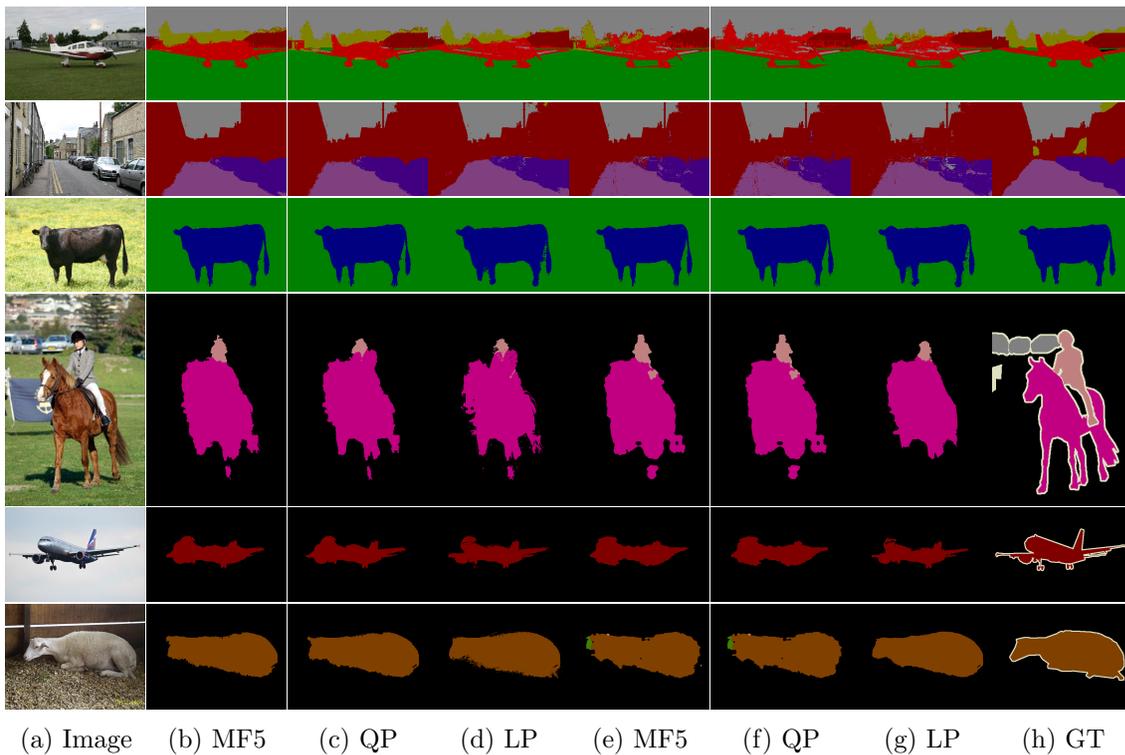(a) Image   (b) MF5   (c) QP   (d) LP   (e) MF5   (f) QP   (g) LP   (h) GT

Figure 3. Qualitative results with the parameters tuned for **MF5$_{clique}$** and **MF5**. As can be clearly seen, even though the parameters have been tuned to **MF5**, the **LP** algorithm produces competitive segmentations when compared to **MF5**.

REFERENCES

[1] A. ADAMS, J. BAEK, AND M. ABRAHAM, *Fast high-dimensional filtering using the permutohedral lattice*, in Proceedings of Eurographics, 2010.

[2] T. AJANTHAN, A. DESMAISON, R. BUNEL, M. SALZMANN, P. H. S. TORR, AND M. P. KUMAR, *Efficient linear programming for dense CRFs*, in Proceedings of CVPR, 2017.

[3] F. R. BACH, *Duality between subgradient and conditional gradient methods*, SIAM J. Optim., 25 (2015), pp. 115–129.

[4] P. BAQUÉ, T. BAGAUTDINOV, F. FLEURET, AND P. FUA, *Principled parallel mean-field inference for discrete random fields*, in Proceedings of CVPR, 2016.

[5] S. CHANDRA, N. USUNIER, AND I. KOKKINOS, *Dense and low-rank gaussian crfs using deep embeddings*, in Proceedings of International Conference on Computer Vision, Vol. 2, 2017.

[6] C. CHEKURI, S. KHANNA, J. NAOR, AND L. ZOSIN, *Approximation algorithms for the metric labeling problem via a new linear programming formulation*, in Proceedings of SODA, 2001.

[7] L. CHEN, G. PAPANDREOU, I. KOKKINOS, K. MURPHY, AND A. L. YUILLE, *Semantic image segmentation with deep convolutional nets and fully connected CRFs*, in Proceedings of ICLR, 2015.

[8] L.-C. CHEN, G. PAPANDREOU, I. KOKKINOS, K. MURPHY, AND A. L. YUILLE, *DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs*, IEEE Trans. Pattern Anal. Mach. Intell., 40 (2018), pp. 834–848.

[9] D. COMANICIU AND P. MEER, *Mean shift: A robust approach toward feature space analysis*, IEEE Trans. Pattern Anal. Mach. Intell., 24 (2002), pp. 603–619.

[10] A. DESMAISON, R. BUNEL, P. KOHLI, P. H. S. TORR, AND M. P. KUMAR, *Efficient continuous relaxations for dense CRF*, in Proceedings of ECCV, 2016.

[11] M. EVERINGHAM, L. VAN GOOL, C. K. I. WILLIAMS, J. WINN, AND A. ZISSERMAN, *The PASCAL Visual Object Classes Challenge* 2010 *(VOC*2010*) Results*, http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[12] M. FRANK AND P. WOLFE, *An algorithm for quadratic programming*, Naval Res. Logistics Q., 3 (1956), pp. 95–110.

[13] J. L. KELLEY, *General Topology*, Grad. Texts in Math. 27, Springer, New York, 1975.

[14] J. KLEINBERG AND E. TARDOS, *Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields*, J. ACM, 49 (2002), pp. 616–639.

[15] P. KOHLI, P. KUMAR, AND P. TORR, *P3 & beyond: Solving energies with higher order cliques*, in Proceedings of CVPR, 2007.

[16] P. KOHLI, T. MINKA, AND J. WINN, *MSRC Dataset*, https://www.microsoft.com/en-us/research/project/image-understanding (2017).

[17] D. KOLLER AND N. FRIEDMAN, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, Cambridge, UK, 2009.

[18] V. KOLMOGOROV AND R. ZABIH, *What energy functions can be minimizedvia graph cuts?*, IEEE Trans. Pattern Anal. Mach. Intell., 2 (2004), pp. 147–159.

[19] P. KRÄHENBÜHL AND V. KOLTUN, *Efficient inference in fully connected CRFs with gaussian edge potentials*, in Proceedings of NIPS, 2011.

[20] P. KRÄHENBÜHL AND V. KOLTUN, *Parameter learning and convergent inference for dense random fields*, in Proceedings of ICML, 2013.

[21] M. P. KUMAR, V. KOLMOGOROV, AND P. H. S. TORR, *An analysis of convex relaxations for map estimation of discrete mrfs*, J. Mach. Learn. Res., 10 (2008), pp. 71–106.

[22] S. LACOSTE-JULIEN, *Convergence Rate of Frank-Wolfe for Non-Convex Objectives*, arXiv:1607.00345, 2016.

[23] S. LACOSTE-JULIEN, M. JAGGI, M. SCHMIDT, AND P. PLETSCHER, *Block-coordinate Frank-Wolfe optimization for structural SVMs*, in Proceedings of ICML, 2012.

[24] L. LADICKY, C. RUSSELL, P. KOHLI, AND P. TORR, *Graph cut based inference with co-occurrence statistics*, in Proceedings of ECCV, 2010.

[25] L. LADICKY, C. RUSSELL, P. KOHLI, AND P. H. S. TORR, *Associative hierarchical CRFs for object class image segmentation*, in Proceedings of IEEE 12th Conference on Computer Vision, 2009, pp. 739–746.

[26] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, in Proceedings of CVPR, 2015.

[27] N. Parikh and S. Boyd, *Proximal algorithms*, Found. Trends Optim., 1 (2014), pp. 123–131.

[28] P. Ravikumar and J. Lafferty, *Quadratic programming relaxations for metric labeling and Markov random field map estimation*, in Proceedings of ICML, 2006.

[29] A. Schwing and R. Urtasun, *Fully Connected Deep Structured Networks*, arXiv:1503.0235, 2015.

[30] J. Shotton, J. Winn, C. Rother, and A. Criminisi, *Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context*, Int. J. Comput. Vis., 81 (2009), pp. 2–23.

[31] J. Snoek, H. Larochelle, and R. Adams, *Practical bayesian optimization of machine learning algorithms*, in Proceedings of NIPS, 2012.

[32] M. Tappen, C. Liu, E. Adelson, and W. Freeman, *Learning gaussian conditional random fields for low-level vision*, in Proceedings of CVPR, 2007.

[33] A. Torralba, K. P. Murphy, and W. T. Freeman, *Sharing visual features for multiclass and multiview object detection*, IEEE Trans. Pattern Anal. Mach. Intel., 29(2007), pp. 854–869.

[34] V. Vineet, J. Warrell, and P. H. S. Torr, *Filter-based mean-field inference for random fields with higher-order terms and product label-spaces*, in Computer Vision: ECCV 2012, Lecture Notes in Comput. Sci. 7576, Springer, New York, pp. 31–44.

[35] P. Wang, C. Shen, and A. van den Hengel, *Efficient SDP inference for fully-connected CRFs based on low-rank decomposition*, in Proceedings of CVPR, 2015.

[36] X. Xiao and D. Chen, *Multiplicative iteration for nonnegative quadratic programming*, Numer. Linear Algebra Appl., submitted.

[37] Y. Zhang and T. Chen, *Efficient inference for fully-connected CRFs with stationarity*, in Proceedings of CVPR, 2012.

[38] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, *Conditional random fields as recurrent neural networks*, in Proceedings of ICCV, 2015.