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) with Gaussian pairwise potentials have become a popular framework for modelling several problems in computer vision such as stereo correspondence and multi-class semantic segmentation. By modelling long-range interactions, dense CRFs provide a more detailed labelling compared to their sparse counterparts. Currently the state-of-the-art algorithm performs mean-field inference using a filter-based method to obtain accurate segmentations, but fails to provide strong theoretical guarantees on the quality of the solution. Whilst the underlying model of a dense CRF provides enough information to yield well defined segmentations, it lacks the richness introduced via higher order potentials. The mean-field inference strategy was also extended to incorporate higher order potentials, but again failed to obtain a bound on the quality of the solution. To this extent, we show that a dense CRF can be aggregated with sparse higher order potentials in a way that is amenable to continuous relaxations. We will then show that, by using a filter-based method, these continuous relaxations can be optimised efficiently using state-of-the-art algorithms. Specifically we will solve a quadratic programming (QP) relaxation using the Frank-Wolfe algorithm and a linear programming (LP) relaxation by developing a proximal minimisation framework. By exploiting labelling consistency in the higher order potentials and utilising 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 experiments are performed on the standard publicly available MSRC data set and demonstrate the low energies achieved from the minimisation and the accuracy of the resulting segmentations.

1. Introduction.

Conditional random fields (CRFs) with sparse higher order potentials are a popular framework for modelling several problems in computer vision. In order to use them in practice, one requires an energy minimisation algorithm that obtains the most likely output for a given input. The energy function consists of a sum of three types of terms: *unary potentials* that depend on the label for one random variable; *pairwise potentials* that depend on the labels of two random variables; and *higher order potentials* that depend on a collection of random variables. Many works such as [2, 8, 14] just focus on the unary and pairwise potentials, leaving out the higher order potential due to it's high complexity.

To combat the high complexity of the pairwise potential, traditional computer vision methods employed sparse connectivity structures, such as 4 or 8 connected grid CRFs. Their popularity led to a considerable research effort in efficient energy minimisation algorithms. One of the biggest successes of this effort was the development of several accurate continuous relaxations of the underlying discrete optimisation problem [10, 24]. An important advantage of such relaxations is that they lend themselves easily to analysis, which allows us to compare them theoretically [24], as well as establish bounds on the quality of their solutions [5].

Recently, the influential work of Krähenbühl and Koltun [14] has popularised the use of dense CRFs, where each pair of random variables is connected by an edge. Dense CRFs capture useful long-range interactions, thereby providing finer details on the labelling. However, modeling long-range interactions comes at the cost of a significant increase in complexity. In

*Indicates equal contribution

order to operationalise dense CRFs, Krähenbühl and Koltun [14] made two key observations. First, the pairwise potentials used in computer vision typically encourage smooth labelling. This enabled them to restrict themselves to the special case of Gaussian pairwise potentials introduced by Tappen et al. [28]. Second, for this special case, it is possible to obtain a labelling efficiently by using the mean-field algorithm [13]. Specifically, the message computation required at each iteration of mean-field can be carried out in $\mathcal{O}(N)$ operations, where N is the number of random variables (of the order of hundreds of thousands). This is in contrast to a naïve implementation that requires $\mathcal{O}(N^2)$ operations. The significant speed-up is made possible by the fact that the messages can be computed using the filtering approach of Adams et al. [1]. Vineet et al. [30] made use of this filter based method to perform mean-field inference on a dense CRF with sparse higher order potentials, which provided an improvement in segmentation accuracy.

While the mean-field algorithm does not provide any theoretical guarantees on the quality of the solutions, 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 [14, 30]. However, this still leaves open the intriguing possibility that the same filtering approach that enabled the efficient mean-field algorithm can also be used to speed-up energy minimisation algorithms based on continuous relaxations. In this work, we show that this is indeed possible.

In more detail, we make two contributions to the problem of energy minimisation in dense CRFs with sparse higher order potentials. First, we show that the conditional gradient of a quadratic programming (QP) relaxation [24] 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 minimise the QP relaxation efficiently using the Frank-Wolfe algorithm [9]. Second, we introduce an iterative linear programming (LP) minimization algorithm which has a complexity 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 [23]. 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 [9, 19]. 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 [14], thus speeding up convergence. Second, the optimal step size of our iterative procedure can be obtained analytically, thus preventing 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, 8]. However, all relevant information and findings are detailed in this work. Specifically, our contribution is a QP and LP relaxation for dense CRF with sparse higher order potentials. The effectiveness of both algorithms are demonstrated on a selection of the

MSRC data set for which accurate ground truth segmentations are available.

2. Related works. Krähenbühl and Koltun popularised the use of densely connected CRFs at the pixel level [14], 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].

While the original work [14] used a version of mean-field that is not guaranteed to converge, their follow-up paper [15] 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 et al. [30] 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 [20] and P^n-Potts models [11].

The success of the inference algorithms naturally led to research in learning the parameters of dense CRFs. Combining them with fully convolutional neural networks [22] has resulted in high performance on semantic segmentation applications [6]. Several works [25, 34] showed independently how to jointly learn the parameters of the unary and pairwise potentials of the CRF. These methods resulted in significant improvements on various computer vision applications such as semantic segmentation.

Independently from the mean-field work, Zhang et al. [33] 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 differ in the form of the pairwise potentials. The algorithm they propose to solve the QP relaxation has linearithmic complexity while ours is linear in the number of labels and random variables. Furthermore, it is not clear whether their approach can be easily generalised to tighter relaxations such as the LP.

Wang et al. [31] derived a semi-definite programming relaxation of the energy minimisation problem, allowing them to reach lower energies than mean-field. Their approach has the advantage of not being restricted to Gaussian pairwise potentials. Inference is made feasible by performing 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 QP, but in practice the runtime is significantly higher.

In this paper, we use the same filter-based method [1] as the one employed in mean-field. We build on 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, 15, 30] improved mean-field and [25, 34] learnt the parameters, we focus on the energy minimisation 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 $\mathcal{X} = \{X_1, ..., X_N\}$ where each random variable X_a takes a single label from the set of M labels $\mathcal{L} = \{l_1, ..., l_M\}$. To formalise this labelling, 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 containing two or more vertices. In the context of this work, a clique with three or more random variables represents a higher order potential and a clique with two or more random variables is represented by a pairwise potential. A given clique S_p is a subset of $\mathcal{X}$ and the set of cliques $\mathcal{S}$ is defined below:

$$\mathcal{S} = \{S_p \,|S_p \ \subseteq \mathcal{X} \text{ s.t. } p \subseteq \{1, ..., N\}\}\,. \tag{3.1}$$

Here, R represents the total number of cliques in the set $\mathcal{S}$. It will prove useful to introduce an additional vector $\mathbf{x}_p \in \{x_c | \forall c \in p\}$, which is a vector of more than two elements, containing the labels of the random variables in the clique S_p. With the introduction of $\mathbf{x}_p$, the energy function can be defined as:

$$E(\mathbf{x}) = \sum_a \phi_a(x_a) + \sum_a \sum_{a,b \neq a} \psi_{a,b}(x_a, x_b) + \sum_{S_p \in \mathcal{S}} \theta_p(\mathbf{x}_p), \tag{3.2}$$

where $\phi_a(x_a)$ denotes the unary potential, $\psi_{a,b}(x_a, x_b)$ denotes the pairwise potential, $\theta_p(\mathbf{x}_p)$ 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 the context of semantic segmentation, which is the goal of this work, a random variable corresponds to a pixel and the associated labels correspond a semantic class. A clique is represented by a *superpixel*, which is a collection of homogeneous spatially adjacent pixels. The optimal solution to this energy function forms an optimisation problem over the variable $\mathbf{x}$ and can be compactly written as:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{L}^\mathbf{N}}{\operatorname{argmin}}\, E(\mathbf{x}). \tag{3.3}$$

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

3.1.1. Unary Potentials. The unary potentials for this formulation can be arbitrary, but generally provide a rough initial labelling solution. In this work we employ unary potentials which are derived from TextonBoost [21, 26]. More detail on the generation of the unary potentials is given in section 6.2.

3.1.2. Gaussian Pairwise Potentials. Following the work of [14], the form of the pairwise potential is given by:

$$\psi_{a,b}(x_a, x_b) = \mu(x_a, x_b) K_{ab}, \tag{3.4}$$

$$\text{s.t } K_{ab} = \sum_m \omega^{(m)} k^{(m)}(\mathbf{f}_a, \mathbf{f}_b) \tag{3.5}$$

where $\mu(x_a, x_b)$ represents the *label compatibility*, K_{ab} is the *pixel compatability* function which is defined in the next paragraph and $k^{(m)}(\mathbf{f}_a, \mathbf{f}_b)$ are gaussian kernels taking the form of:

$$k^{(m)}(\mathbf{f}_a, \mathbf{f}_b) = \exp\left(- \frac{||\mathbf{f}_a - \mathbf{f}_b||^2}{2\sigma^2} \right) \tag{3.6}$$

The terms $\mathbf{f}_a$ and $\mathbf{f}_b$ are feature vectors containing the spatial and colour information of the image with pixel indices a and b respectively.

Pixel Compatibility. For multi-class semantic segmentation problems, the pixel compatibility function takes the form of a contrast-sensitive two-kernel potential, defined as:

$$K_{ab} = w^{(1)} \exp\left(- \frac{|p_a - p_b|^2}{2\sigma^2_{(1)}} - \frac{|I_a - I_b|^2}{2\sigma^2_{(2)}} \right) + w^{(2)} \exp\left(- \frac{|p_a - p_b|^2}{2\sigma^2_{(3)}} \right), \tag{3.7}$$

with I_a, I_b and p_a, p_b representing the colour information and spatial information of pixels a and b respectively. The first term corresponds to the *appearance kernel* and is inspired by the observation that pixels of similar colour and position are likely to take the same label, the second term corresponds to a *smoothness kernel* which penalises 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 6.2.

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 and is specified as:

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

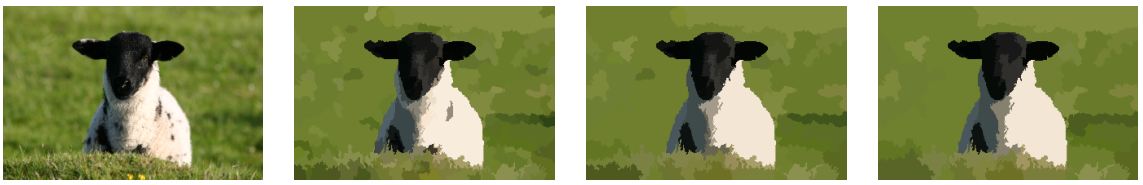
Whilst other label compatibility function exists, such as metric or semi-metric functions [10], the Potts model was chosen as it enables more sophisticated minimisation algorithms to be leveraged which will be discussed in section 4 and 5.

3.1.3. Higher Order Potentials. In this work, the higher order terms are represented as a *clique potential*. The specific role of the clique potential, which is introduced in equation (3.2), is to minimise the associated cost if all of the random variables within the clique S_p take the same label. Specifically, if all of the random variables in S_p do not take the same label, the clique potential introduces a cost proportional to the variance of the colour information of the super pixel. The clique potential is defined by:

$$\theta_p(\mathbf{x}_p) = \begin{cases} 0 & \text{if } x_c = x_d,\ \forall c, d \in p \\ C_p & \text{otherwise,} \end{cases} \tag{3.9}$$

$$\text{s.t } C_p = \Gamma \exp\left\{ \frac{-\sigma_p^2}{\eta} \right\}, \tag{3.10}$$

where Γ and η are learned parameters and σ_p^2 represents the variance of the pixel colour 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.



Original Image | Min Region = 100 | Min Region = 250 | Min Region = 400

Figure 1. *Visual examples of the Mean-Shift algorithm with varying minimum region values. As the minimum region size increases, the consistency of the underlying pixels reduces, this can be seen by comparing the bodies of the sheep for the different over segmentations, where the dark spots are incorporated into the larger superpixel. Best viewed in colour.*

Generating Cliques. For this work, a clique represents a super-pixel of an over segmented image. A superpixel is a collection of adjoining pixels who share similar colour information. The cliques were generated using the mean-shift algorithm [7] which is a semiparametric method of segmenting an image in to superpixels. We used the mean-shift algorithm due to its simplicity, however in practice any algorithm can be used for generating the over segmentations. The mean-shift algorithm takes three input parameters to control the segmentation, the parameters are: spatial bandwidth, resolution bandwidth and minimum region size. The spatial bandwidth controls the dimensions of the kernel for filtering and the resolution bandwidth controls the colour and spectral information. The minimum region size, provides a lower bound on the quantity of pixels contained within the super-pixel. For more information the reader is encouraged to consult the paper by Comaniciu and Meer [7]. Visual examples of over-segmentations are given in Figure 1. 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 super pixel. Whilst 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. Advanced Filtering Method. The pixel compatibility function defined in (3.7), was chosen to take a Gaussian formation due to the fact that it allows an advanced filter-based method [1] to be utilised. This filter based method exploits the *permutohedral lattice* to achieve efficient computation of operations featuring Gaussian kernels, specifically it approximates the following:

$$\forall a \in \{1, ..., N\}, \qquad v'_a = \sum_b k(\mathbf{f}_a, \mathbf{f}_b) v_b, \tag{3.11}$$

where $v'_a, v_b \in \mathbb{R}$, $b \in \{1, ..., N\}$ and $k(\mathbf{f}_a, \mathbf{f}_b)$ is a Gaussian kernel described in section 3.1.2. The naïve approach to this operation 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 [14] employed this filter-based method to compute the mean-field

inference algorithm efficiently. We investigated the accuracy of the filter-based method [1] with differing values for the variances of equation (3.7) in our preliminary work [8]. 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 [8] 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.

3.1.5. Integer Program Formulation. We now formulate the energy minimisation function (3.2) as an integer program (IP). From this IP, continuous relaxations will be applied to form the QP relaxation and the LP relaxation, which are introduced in sections 4 and 5 respectively. To this end, a new vector $\mathbf{y} \in \mathbb{R}^{NM}$ is introduced, such that its elements $y_{a:i} \in \{0,1\}$ are binary variables indicating whether or not the random variable X_a takes the label l_i. The vector $\mathbf{y}_p \in \{y_{c:i} | \forall c \in p \ , \forall i \in \mathcal{L}\}$ is introduced which holds the vectors of indicator variables for $\mathbf{x}_p$. With this new notation the energy minimisation function can be defined as:

$$\begin{aligned} (3.12) \qquad \min_{y} \sum_{a} \sum_{i} \phi_a(i) y_{a:i} + \sum_{a} \sum_{a,b \neq a} \sum_{i,j} \psi_{a,b}(i,j) y_{a:i} y_{b:j} + \sum_{p} \sum_{i} \overline{\theta}_p(\mathbf{y}_p), \\ \text{s.t} \quad \textstyle\sum_{i \in \mathcal{L}} y_{a:i} = 1 \quad \forall a \in \{1,...,N\}, \\ y_{a:i} \in \{0,1\} \quad \forall a \in \{1,...,N\}, \quad \forall i \in \mathcal{L}, \end{aligned}$$

where $\overline{\theta}_p(\mathbf{y}_p) \equiv \theta_p(\mathbf{x}_p)$. The first set of constraints ensure that each random variable has to be assigned exactly one label, whilst the second constraint ensures that the labelling is binary. It is important to note that $\theta_p(.)$ is a polynomial with an order equal to number of random variables within the clique S_p. Normally the manipulation of $\theta_p(.)$ would exhibit an intractable complexity, however by exploiting labelling 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 (3.12) is still NP-hard and hence cannot be solved in polynomial time. We address this issue by applying continuous relaxations to approximate the IP, so that we can formulate an energy minimisation problem and establish a bound on the quality of the solution. 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 optimise our first continuous relaxation, namely the QP relaxation.

4.1. Notation and Formulation. The unary and pairwise potentials of the IP given in equation (3.12) can be neatly summarised 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 equation (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 $k^{(m)}(\mathbf{f}_a, \mathbf{f}_b)$. The matrix $\boldsymbol{\Psi} \in \mathbb{R}^{NM \times NM}$

represents the pairwise terms and is defined as:

$$\boldsymbol{\Psi} = \boldsymbol{\mu}_{Potts} \otimes \sum_{m} \omega^{(m)} \big(\boldsymbol{K}^{(m)} - \boldsymbol{I}_N \big), \tag{4.1}$$

where $\otimes$ is the Kronecker product, $\boldsymbol{I}_N$ is the identity matrix of size $\mathbb{R}^{N \times N}$. Similarly to [15], $\boldsymbol{K}^{(m)}$ 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:

$$\begin{aligned} &\min_{\mathbf{y}} \boldsymbol{\phi}^T \mathbf{y} + \mathbf{y}^T \boldsymbol{\Psi} \mathbf{y}, \\ \text{s.t} \quad &\textstyle\sum_{i \in \mathcal{L}} y_{a:i} = 1 \quad \forall a \in \{1, ..., N\}, \\ &y_{a:i} \in \{0, 1\} \quad \forall a \in \{1, ..., N\}, \quad \forall i \in \mathcal{L}. \end{aligned} \tag{4.2}$$

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 labelling 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 or not all of the random variables in the clique S_p take the label i. The auxiliary variable $z_{p:i}$ is given as:

$$z_{p:i} = \begin{cases} 0, & \text{if } y_{a:i} = 1, \ \forall a \in p \\ 1, & \text{otherwise.} \end{cases} \tag{4.3}$$

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 ath pixel belongs to the clique S_p. Formally $H_p(a) = 1$ if $a \in p$ 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:

$$f_c := \sum_{p} \sum_{i} C_p \left[z_{p:i} + [(1 - z_{p:i}) \sum_{a} H_p(a)(1 - y_{a:i})] \right]. \tag{4.4}$$

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 vectorised 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 is not stored as a matrix of size $MR \times NM$. Instead, similar to a list of lists data structure R arrays are instantiated which represent the cliques and contain indexes of the pixels within the corresponding clique. With the addition of $\mathbf{z}$ and $\boldsymbol{H}$, the IP can be concisely written in vector form as:

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

Algorithm 4.1 QP Minimisation Algorithm

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

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 $\mathbf{1_z} \in \mathbb{R}^{MR}$ and $\mathbf{1_y} \in \mathbb{R}^{NM}$ are vectors of all ones.

4.1.1. Relaxations. The Integer Program introduced in equation (4.8) 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:

$$\mathcal{Y} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{cc} \sum_i y_{a:i} = 1 & \forall a \in \{1, ..., N\} \\ y_{a:i} \geq 0 & \forall a \in \{1, ..., N\}, \forall i \in \mathcal{L} \end{array} \right\}, \tag{4.6}$$

$$\mathcal{Z} = \left\{ \mathbf{z} \;\middle|\; 0 \leq z_{p:i} \leq 1 \quad \forall p, \forall i \in \mathcal{L} \right\}. \tag{4.7}$$

Thus, the QP relaxation can be formally defined as:

$$\begin{aligned} \min_{\mathbf{y},\mathbf{z}} f(\mathbf{y}, \mathbf{z}) &= \boldsymbol{\phi}^T\mathbf{y} + \mathbf{y}^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T\mathbf{z} + (\mathbf{1_z} - \mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{1_y} - \mathbf{y}), \\ \text{s.t } \mathbf{y} &\in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}. \end{aligned} \tag{4.8}$$

4.2. Minimisation. The minimisation of the objective function is achieved via the Frank-Wolfe algorithm [9]. 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 observe that the conditional gradient can be computed in a complexity linear in the number of labels and pixels. Whilst the Frank-Wolfe algorithm normally optimises convex objectives, it has been proven to find a stationary point at a rate of $\mathcal{O}(1/\sqrt{t})$ of a non-convex objective function over a convex compact set [18]. The key steps of the algorithm are shown in Algorithm 4.1. To utilise the Frank-Wolfe algorithm effectively, three steps need to be taken: obtain 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.

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:

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

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 results 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 $\boldsymbol{H}$ is implemented as a list of lists data structure 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 the clique. However, whilst this complexity is inline with the rest of the algorithm, we will show that the gradients can be updated by using the update equations - removing the need for explicit computation. Further detail on this will be explained in Section 4.2.2.

4.2.2. Low Cost Gradient Computation. In section 4.2.1 the reader's attention was drawn to the fact that the gradient need not be explicitly computed at every iteration. Instead the gradient can be incremented from its initial value using the update equations. The expensive operations of $2\boldsymbol{\Psi}\mathbf{y}$ and $\boldsymbol{H}^T\mathbf{C}\mathbf{z}$ in equation (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 which are given as:

$$\begin{bmatrix} \mathbf{y}^{t+1} \\ \mathbf{z}^{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{y}^t + \delta(\mathbf{s_y}^t - \mathbf{y}^t) \\ \mathbf{z}^t + \delta(\mathbf{s_z}^t - \mathbf{z}^t) \end{bmatrix}. \tag{4.10}$$

Hence only one call to the filter based method is required per iteration. 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:

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

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

Thus, allowing the explicit computation of the operations 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:

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

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 need to be called when computing the gradient and the product of $\boldsymbol{H}^T\mathbf{Cz}$ 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 $\begin{pmatrix}\mathbf{s_y}\\ \mathbf{s_z}\end{pmatrix}$ with $\mathbf{s_y} \in \mathcal{Z}, \mathbf{s_z} \in \mathcal{Y}$, of the objective function f is obtained by solving:

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

Minimising equation (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 like so $\mathcal{Y} = \prod_a \mathcal{Y}_a$, where $\mathcal{Y}_a = \{y_{a:i} | \sum_i y_{a:i} = 1, y_{a:i} \geq 0, i \in \mathcal{L}\}$. Exploiting this constraint enables the minimisation problem to be broken down into N smaller minimisation problems. Minimising $\langle \mathbf{s_y}, \nabla f(\mathbf{y}, \mathbf{z})\rangle$ 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:

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

The feasible set $\mathcal{Z}$ is also linear separable and can be decomposed like so: $\mathcal{Z} = \prod_{p,i} \mathcal{Z}_{p:i}$. Thus the minimisation for $\mathbf{s_z} = \text{argmin}_{i\in\mathcal{Z}} \langle \mathbf{s_z}, \nabla_z f(\mathbf{y}, \mathbf{z})\rangle$ can be performed via a linear search through all MR elements. With the constraints on the set $\mathcal{Z}_{p:i} = \{z | 0 \leq z_{p:i} \leq 1\}$, the conditional gradient $\mathbf{s_z}$, is given as:

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

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 optimal step size to the Frank-Wolfe algorithm is achieved via line search. However, for this problem the optimal step size can be computed via minimising 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:

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

A closed form solution of the optimal step size is given in appendix A.1. Obtaining the optimal step size will result in a faster convergence rate and hence provide 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 normally cause intractability within the algorithm. This is achieved via exploiting the filter based method [1], labelling 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 minimisation 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 (3.12) 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 minimisation algorithm [23] 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, we define the LP relaxation of (3.12) as

$$\min_{\mathbf{y}} \tilde{E}(\mathbf{y}) = \sum_{a} \sum_{i \in \mathcal{L}} \phi_{a:i} y_{a:i} + \sum_{a, b \neq a} \sum_{i} \mathrm{K}_{ab} \frac{|y_{a:i} - y_{b:i}|}{2} + \sum_{p} C_p \max_{i} \max_{\substack{c,d \in p \\ c \neq d}} |y_{c:i} - y_{d:i}|, \tag{5.1}$$
$$\text{s.t } \mathbf{y} \in \mathcal{Y},$$

where K_{ab} is the sum of Gaussian kernels and is defined in equation (3.7). For integer labellings, the LP 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 [10]. Using standard solvers to minimize this LP would require the introduction of $\mathcal{O}((NM)^2)$ variables (see equation (5.3)), making it intractable. Therefore the non-smooth objective of equation (5.1) has to be optimized directly. This was handled using projected subgradient descent in our previous version [8], which also turns out to be inefficient in practice. In this paper, we extend the algorithm introduced in [2] to handle higher order potentials, whilst maintaining linear scaling in both space and time complexity.

5.2. Minimisation. In this section we present our efficient minimisation strategy, which uses the proximal method [23]. The complexity of each iteration of our implementation remains linear in the number of labels and pixels.

5.2.1. Proximal Minimisation for LP Relaxation. Our goal is to design an efficient minimization strategy for the LP relaxation in (5.1). In our previous version [8], we utilised projected subgradient descent to minimise a similar LP to (5.1), however this method resulted in a significantly high runtime, and a complexity that scales at $\mathcal{O}(MN\log(N))$. To this end, we propose to use the proximal minimization algorithm [23]. The additional quadratic regularization term makes the dual problem smooth, enabling the use of more sophisticated optimization methods. Furthermore, this method guarantees monotonic decrease in the objective value, enabling us to leverage faster, less accurate 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 [23] is an iterative method that, given the current estimate of the solution $\mathbf{y}^k$, solves the problem

$$\begin{aligned} &\min_{\mathbf{y}} \tilde{E}(\mathbf{y}) + \frac{1}{2\lambda}||\mathbf{y} - \mathbf{y}^k||^2, \\ &\text{s.t } \mathbf{y} \in \mathcal{Y}, \end{aligned} \tag{5.2}$$

where λ influences the weighting of the quadratic regulariser. 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 in the standard way. In this section we introduce a new algorithm that is tailored to this problem. In particular, we solve the Lagrange dual of (5.2) in a block-wise fashion.

5.2.2. Dual Formulation. The first stage of forming the dual of the proximal minimisation problem (5.2) is to re-write the proximal minimisation problem with the introduction of auxiliary variables $v_{ab:i}$ and w_p. The auxiliary variables and their constraints enable the minimisation problem to be defined without the piecewise maximum operators. With the introduction of these auxiliary variables the primal minimisation problem is given as:

$$\begin{aligned} \min_{\mathbf{y},\mathbf{v},\mathbf{w}} &\sum_a \sum_{i \in \mathcal{L}} \phi_{a:i} y_{a:i} + \sum_{a,b \neq a} \sum_i \frac{\mathrm{K}_{ab}}{2} v_{ab:i} + \sum_p C_p w_p + \frac{1}{2\lambda}||\mathbf{y} - \mathbf{y}^k||, \\ \text{s.t } & v_{ab:i} \geq y_{a:i} - y_{b:i} \quad \forall a, b \in \{1, ..., N\} \quad a \neq b \quad \forall i \in \mathcal{L}, \\ & v_{ab:i} \geq y_{b:i} - y_{a:i} \quad \forall a, b \in \{1, ..., N\} \quad a \neq b \quad \forall i \in \mathcal{L}, \\ & w_p \geq y_{c:pi} - y_{d:pi} \quad \forall c, d \in p \quad c \neq d \quad \forall i \in \mathcal{L} \quad \forall p, \\ & w_p \geq y_{d:pi} - y_{c:pi} \quad \forall c, d \in p \quad c \neq d \quad \forall i \in \mathcal{L} \quad \forall p, \\ & y_{a:i} \geq 0 \quad \forall a \in \{1, ..., N\} \quad \forall i \in \mathcal{L}, \\ & \sum_i y_{a:i} = 1 \quad \forall a \in \{1, ..., N\}. \end{aligned} \tag{5.3}$$

As is often done in practice, the auxiliary variables $v_{ab:i}$ and w_p are introduced to replace the piecewise maximum functions described in the previous section, and their constraints ensure that their resulting values are a maximum. Four vectors of dual variables will now be introduced. Namely, $\boldsymbol{\alpha} = \{\alpha^1_{ab:i}, \alpha^2_{ab:i} | \ \forall a \in \{1, ..., N\}, \forall b \in \{1, ..., N\} \backslash \{a\}, \forall i \in \mathcal{L}\}$ for the constraints on $v_{ab:i}$; $\boldsymbol{\mu} = \{\mu^1_{cd:pi}, \mu^2_{cd:pi} | \ \forall p, \ \forall c \in p, \forall d \in p \backslash \{c\}, \ \forall i \in \mathcal{L}\}$ for the constraints on w_p; $\boldsymbol{\beta} = \{\beta_a | a \in \{1, ..., N\}\}$ for the labelling constraint on $y_{a:i}$ and $\boldsymbol{\gamma} = \{\gamma_a | a \in \{1, ..., N\}, i \in \mathcal{L}\}\}$ for the non negativity constraint on $y_{a:i}$. 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:

$$(\mathbf{A}\boldsymbol{\alpha})_{a:i} = \sum_{a \neq b} (\alpha^2_{ab:i} - \alpha^1_{ab:i} - \alpha^2_{ba:i} + \alpha^1_{ba:i}) \tag{5.4}$$

$$(\mathbf{U}\boldsymbol{\mu})_{c:pi} = \sum_{\substack{c,d \in p \\ c \neq d}} (\mu^2_{cd:pi} - \mu^1_{cd:pi} - \mu^2_{dc:pi} + \mu^1_{dc:pi}) \tag{5.5}$$

$$(\mathbf{B}\boldsymbol{\beta})_{a:i} = \beta_a, \tag{5.6}$$

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 equation (5.3).

Proposition 5.1. Formation of the Lagrange Dual

1. *The Lagrange dual of equation (5.3) is given as:*

(5.7)

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

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

$$\mathbf{y} = \lambda(\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}) + \mathbf{y}^k \tag{5.8}$$

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

5.2.3. LP Minimisation Algorithm. The dual problem (5.7), 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 propose an efficient block coordinate descent approach. Each of these blocks are amenable to more sophisticated optimization methods, resulting in a computationally efficient algorithm. Since the different sets of variables in (5.7) have different types of feasible domains, we propose to optimize (5.7) in a block-wise fashion. 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 optimisation algorithms. For $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ the problem

Algorithm 5.1 Proximal minimisation of LP

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

decomposes over the pixels. Then with the optimal values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, the minimisation of $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ is over a compact domain, and can be efficiently tackled using the Frank-Wolfe algorithm [9]. The complete algorithm is summarised in Algorithm 5.1.

Optimising over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. The values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are efficiently optimised 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 minimisation algorithm detailed in [32]. 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 $\boldsymbol{\gamma}$. Using the fact that $\mathbf{B}^T\mathbf{y}^k = \mathbf{1}$ and $\mathbf{B}^T\mathbf{B} = M\mathbf{I}$, an expression for $\boldsymbol{\beta}$ can be given as:

$$\boldsymbol{\beta} = -\frac{\mathbf{B}^T}{M}(\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t + \boldsymbol{\gamma} - \boldsymbol{\phi}), \tag{5.9}$$

where the reader is reminded that M is the total number of labels. A proof to a similar problem is available in Appendix A.2 of our preliminary version of this work [2]. By substituting the expression for $\boldsymbol{\beta}$ into the dual objective (5.7), a quadratic optimisation problem over $\boldsymbol{\gamma}$ is formed. Interestingly, the resulting problem can be optimized independently for each pixel, with each subproblem being an M dimensional quadratic program (QP) with nonnegativity constraints, where M is the number of labels. For a pixel a, this QP has the form:

$$\begin{aligned} \min_{\boldsymbol{\gamma}_a} \quad & \frac{1}{2}\boldsymbol{\gamma}^{\boldsymbol{T}}{}_a\mathbf{Q}\boldsymbol{\gamma}_a + \langle \mathbf{Q}((\mathbf{A}\boldsymbol{\alpha}^{\boldsymbol{t}}{}_a) + (\mathbf{U}\boldsymbol{\mu}^{\boldsymbol{t}})_a - \boldsymbol{\phi}_a) + \mathbf{y}^k, \boldsymbol{\gamma}_a \rangle, \\ \text{s.t} \quad & \boldsymbol{\gamma}_a \geq \mathbf{0}. \end{aligned} \tag{5.10}$$

Here, $\boldsymbol{\gamma}_a$ denotes the vector $\{\gamma_{a:i} | i \in \mathcal{L}\}$ and $\mathbf{Q} = \lambda(\boldsymbol{I} - \frac{\mathbf{1}}{M}) \in \mathbb{R}^{M \times M}$, with $\boldsymbol{I}$ being the identity matrix and $\mathbf{1}$ being a matrix of all ones. A detailed proof to a similar problem is given in Appendix A.2 of [2]. For notational simplicity it will be beneficial to write the quadratic program above (5.10) in the following way:

$$\begin{aligned} \min_{\boldsymbol{\gamma}_a} \quad & \frac{1}{2}\boldsymbol{\gamma}^{\boldsymbol{T}}{}_a\mathbf{Q}\boldsymbol{\gamma}_a - \langle \mathbf{h}_a, \boldsymbol{\gamma}_a \rangle, \\ & \mathbf{h}_a = -\mathbf{Q}((\mathbf{A}\boldsymbol{\alpha}^{\boldsymbol{t}}{}_a) + (\mathbf{U}\boldsymbol{\mu}^{\boldsymbol{t}})_a - \boldsymbol{\phi}_a) - \mathbf{y}^{\mathbf{k}}. \end{aligned} \tag{5.11}$$

We optimise each of these sub-problems using the iterative method given in [32]. The key stage of the algorithm is the element-wise update equation, which is given by:

$$\gamma_{a:i} = \gamma_{a:i}\left[\frac{2(\mathbf{Q}^{-}\gamma_a)_i + h^{+}_{a:i} + c}{(|\mathbf{Q}|\gamma_a)_i + h^{-}_{a:i} + c}\right], \tag{5.12}$$

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 equation (5.9). Note that, even though the matrix Q has M^2 elements, the multiplication by Q can be performed in $\mathcal{O}(M)$. In particular, the multiplication by 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 Q^{-} and $|Q|$, hence the time complexity of the above update is $\mathcal{O}(M)$. The interested reader is referred to [32] for more information.

Due to the fact that optimisation of $\boldsymbol{\gamma}$ decomposes over the number a pixels, and the optimisation of each subproblem is linear in the number of lables, the total complexity of the optimisation of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ is linear in the number of labels and pixels.

Optimising 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 [9], 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.

Conditional Gradient Computation. With the dual variables fixed at $\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t$ the conditional gradient $\begin{pmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{pmatrix}$ is obtained by solving the following:

$$\begin{pmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{pmatrix} = \underset{\mathbf{s}_\alpha \in \mathcal{A}, \mathbf{s}_\mu \in \mathcal{U}}{\text{argmin}} \left\langle \begin{pmatrix}\mathbf{s}_\alpha \\ \mathbf{s}_\mu\end{pmatrix}, \nabla_{\boldsymbol{\alpha},\boldsymbol{\mu}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \right\rangle. \tag{5.13}$$

Minimising this equation to obtain the conditional gradients $\mathbf{s}_\alpha$ and $\mathbf{s}_\mu$ can be neatly summarised by exploiting the properties of the matrices $\mathbf{A}$ and $\mathbf{U}$ given in equations (5.4) and (5.5) respectively.

Proposition 5.2. Conditional gradient computation

1. *The conditional gradient $\mathbf{s}_\alpha$ is given by:*

$$(\mathbf{A}\mathbf{s}_\alpha)_{a:i} = \sum_b (K_{ab}\mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}] - K_{ab}\mathbb{1}[\tilde{y}^t_{b:i} \leq \tilde{y}^t_{a:i}]), \tag{5.14}$$

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

$$(\mathbf{U}\mathbf{s}_\mu)_{c:pi} = \begin{cases} C_p & \text{if } \tilde{y}^t_{c:pi} \leq \tilde{y}^t_{d:pj} \ \forall d \in p \ \forall j \in \mathcal{L} \\ -C_p & \text{if } \tilde{y}^t_{c:pi} \geq \tilde{y}^t_{d:pj} \ \forall d \in p \ \forall j \in \mathcal{L} \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

where $\tilde{\mathbf{y}}^{\mathbf{t}}$ is the current (infeasible) solution computed using equation (5.8).

Proof. Full derivations of the conditional gradients are given in Appendix A.3.

Note that the conditional gradient in (5.14) takes the same form as the subgradient in equation (20) of [8]. This is not a surprising result, as there has been proven a 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.4, which reduces equation (5.14) 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 $\tilde{y}^t_{c:i}$ 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. Performance of any gradient descent based algorithms are fundamentally dependant on the choice of step size. Here, the optimal step size can be computed via minimising a quadratic function over a single variable, which has a closed form solution. This further improve the sophistication of this method. The optimal step size for the Frank-Wolfe algorithm is obtained by solving:

$$\delta = \underset{\delta \in [0,1]}{\operatorname{argmin}}\, g(\boldsymbol{\alpha}^t + \delta(\mathbf{s}^t_\alpha - \boldsymbol{\alpha}^t), \boldsymbol{\mu}^t + \delta(\mathbf{s}^t_\mu - \boldsymbol{\mu}^t), \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \tag{5.16}$$

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

Proposition 5.3. Optimal step size calculation

The optimal step size to the Frank-Wolfe algorithm is given as:

$$\delta = P_{[0,1]}\left[\frac{\langle \mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{A}\mathbf{s}^t_\mu - \mathbf{U}\mathbf{s}^t_\alpha, \tilde{\mathbf{y}}^\mathbf{t}\rangle}{\lambda||\mathbf{A}\boldsymbol{\alpha}^t + \mathbf{U}\boldsymbol{\mu}^t - \mathbf{A}\mathbf{s}^t_\mu - \mathbf{U}\mathbf{s}^t_\alpha||^2}\right], \tag{5.17}$$

where $P_{[0,1]}$ indicates the truncation of the quotient into the interval $[0, 1]$. Obtaining the optimal step size results in a faster convergence rate and hence yields an efficient algorithm. The reader is encouraged to visit [2] for a proof of a similar proposition.

5.3. Summary. To summarise, 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 block-wise fashion. Specifically, the dual variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are computed efficiently by minimising 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], labelling 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 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 **QP** and **LP** respectively. To provide a standard benchmark, we include **MF5**, which is the mean-field algorithm [14] run for 5 iterations as is often done in practice. All experiments were conducted on a 3.60GHz Intel i7-6850K processor. No GPU parallelisation was utilised and the experiments were performed within a single processing thread. All experiments used the unary potentials provided by Krähenbühl and Koltun [14] which can be found at the following location [16]. The initial starting points for the algorithms are obtained by minimizing the unary potentials.

6.1. MSRC Dataset. The experiments were conducted on the MSRC [12] data set which is a standard benchmark for semantic segmentation. The data set contains 591 images with 21 classes, the dimensions of the images are 320×213 pixels. The labelling ground truths provided in the MSRC data set are of poor quality as regions around the object are left unlabelled and the boundaries are inaccurate. Hence, the current data set is not sufficient for performance evaluation, to overcome this Krähenbühl and Koltun [14] 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.

6.2. Methods.

Training of Unary Potentials. The unary potentials were trained using the JointBoost algorithm [29] by Krähenbühl and Koltun [14] for the MSRC data set. To train the unary potentials, 45% of the original data set was used. None of the images used to train the unary potentials are present in the 94 images with accurate ground truths.

Generating Cliques. To generate the higher order potentials we used the mean-shift algorithm [7]. Poor proposals of the superpixels will lead to errors and a decrease in accuracy within the methods. To mitigate the effect this will have, three sets of over segmentations were generated for each image. Whilst the constraint is enforced that the intersection between cliques produces an empty set, this is only to ensure the computational complexity scales linearly with labels and pixels at each iteration. Thus, introducing three sets of over-segmented images, will only affect the complexity by a constant factor of three. We set a spatial and range resolution to 8 and 4 respectively for all three of the over segmented images, and varied the minimum region size to 100, 250 and 400 pixels.

Cross Validation of Parameters. We performed cross validation on each of the four algorithms. For the $\mathbf{QP_{clique}}$ and $\mathbf{LP_{clique}}$, eleven parameters had to be cross validated - five for the pixel compatibility function (3.7) and then two for each set of over segmentations for the clique potential (3.9). For the **QP** and **LP**, only five parameters had to be cross validated for the pixel compatibility function (3.7). This was achieved using the Spearmint package [27], which uses Bayesian inference to obtain optimal parameters. The cross-validated parameters are given in Table 1.

6.3. Results. The collected results provide a quantitative measure of: accuracy, energy and IoU. Accuracy is measured as a percentage of correctly labelled pixels. Energy is the value of the energy function for the resultant labelling. For **QP**, **LP** and **MF5** the assignment energy

Algorithm	$\omega^{(1)}$	$\sigma^{(1)}$	$\sigma^{(2)}$	$\omega^{(2)}$	$\sigma^{(3)}$	$\Gamma^{(1)}$	$\Gamma^{(2)}$	$\Gamma^{(3)}$	$\eta^{(1)}$	$\eta^{(2)}$	$\eta^{(3)}$
MF5	4.16	77.05	47.79	4.69	100	-	-	-	-	-	-
QP	2.36	22.89	48.73	6.53	60.50	-	-	-	-	-	-
LP	1.0	81.71	10.73	15.77	100	-	-	-	-	-	-
$\mathbf{QP_{clique}}$	3.74	17.67	39.76	9.49	54.56	19.71	0	0	10000	109	9757
$\mathbf{LP_{clique}}$	1.0	100	27.71	8.64	99.98	100	0	0	100	6886	110

Table 1

Table of cross validated parameters for each of the above algorithms. It is interesting to note that values of $\Gamma^{(2)}$ and $\Gamma^{(3)}$ are 0, indicating that the super pixels with minimum regions of 250 and 400 pixels are poor proposals of higher order potentials which exhibit labelling consistency.

is calculated using only the unary and pairwise terms of (3.12), whilst $\mathbf{QP_{clique}}$ and $\mathbf{LP_{clique}}$ take the energy function of (3.12). The IoU (Intersection over Union) gives a representation of the proportion of correctly labelled pixels to all pixels taking that class. The optimisation 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 $x_a = \text{argmax}_i(y_{a:i})$. In order to compare energy values, for **QP**, **LP** and **MF5** we used the parameters tuned to **LP** and for $\mathbf{QP_{clique}}$ and $\mathbf{LP_{clique}}$ the parameters were tuned to $\mathbf{LP_{clique}}$, Table 2 gives the results for all algorithms and Figure 2 shows a decrease in energy at runtime. Whilst **LP** and $\mathbf{LP_{clique}}$ could be initialised with a faster algorithm such as **QP**, we chose to presents the results in an "as is" fashion, the interested reader is encouraged to visit [2] for en example of when **LP** is initialised with a faster algorithm.

Algorithm	**Avg.E** ($\times 10^7$)	**Time(s)**	**Acc(%)**	**IoU(%)**
MF5	54.9	**0.27**	79.70	50.66
QP	33.6	2.04	81.03	52.07
LP	**13.8**	64.0	82.59	55.56
$\mathbf{QP_{clique}}$	95.1	4.21	82.55	54.50
$\mathbf{LP_{clique}}$	**44.1**	64.6	**83.25**	**57.25**

Table 2

*Table displaying the average energy, timings, accuracy and IoU, when the parameters for **QP**, **LP** and **MF5** were tuned to **LP** and for $\mathbf{QP_{clique}}$ and $\mathbf{LP_{clique}}$ the parameters were tuned to $\mathbf{LP_{clique}}$. It is shown that the lowest energies are achieved by $\mathbf{LP_{clique}}$ and **LP**. The highest segmentation accuracy and IoU score are obtained by $\mathbf{LP_{clique}}$, which outperforms all other methods.*

The results given in Table 2 show a clear increase in segmentation accuracy and IoU score, when introducing the higher order potentials. It can clearly be seen 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 then QP [17]. As is consistent with our previous work [2] **LP** also obtains lower energies then **MF5**. It also worth noting that $\mathbf{LP_{clique}}$ provides the best segmentation accuracy and IoU score. For consistency we also performed a set of experiments with the parameters tuned to $\mathbf{QP_{clique}}$ and **QP**,

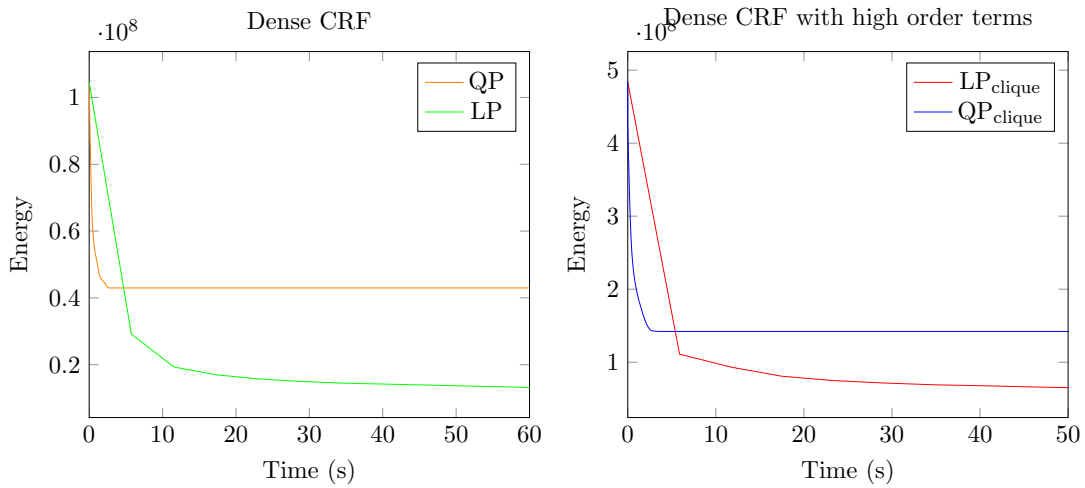


Figure 2. *Assignment energies for the plane image as a function of time when the parameters tuned to* $\boldsymbol{LP_{clique}}$ *and* $\boldsymbol{LP}$*. The left graph shows the assignment energy calculated using only the unary and pairwise potentials, the right image shows the assignment energy of (3.12), which consists of the unary, pairwise and higher order potentials. It is worth noting the first iteration of* $\boldsymbol{LP_{clique}}$ *and* $\boldsymbol{LP}$*, achieves a lower energy then the final energy of* $\boldsymbol{QP_{clique}}$ *and* $\boldsymbol{QP}$ *respectively, further highlighting the sophistication of the LP minimisation.*

given in Appendix A.5. In this setting, the same pattern is observed where $\mathbf{LP_{clique}}$ achieves lower energies then its **QP** and **MF5** counterparts. Interestingly, the highest score of IoU corresponds to $\mathbf{QP_{clique}}$, suggesting that the best performing algorithms (in terms of IoU scores) are the ones in which we cross validate the parameters for.

In summary, $\mathbf{QP_{clique}}$ achieves fast initial energy minimisation but converges to a local minimum and fails to reach the low energies achieved by $\mathbf{LP_{clique}}$.

7. Discussion. The primary contributions of this paper are a quadratic programming and a linear programming relaxation for minimising a dense CRF with sparse higher order potentials. Due to the use of Gaussian Pairwise potentials and enforcing labelling consistency in the higher order terms, the algorithms exhibit a time complexity linear in the number of labels and pixels. It is the tightness of the relaxations coupled with the sophistication of the optimisation techniques that allows both approaches to achieve lower energies than state of the art methods. Furthermore, with a correct set of parameters, accurate segmentations can be obtained without exploiting deep learning. Further work would include incorporating the methods into an end-to-end learning framework [34].

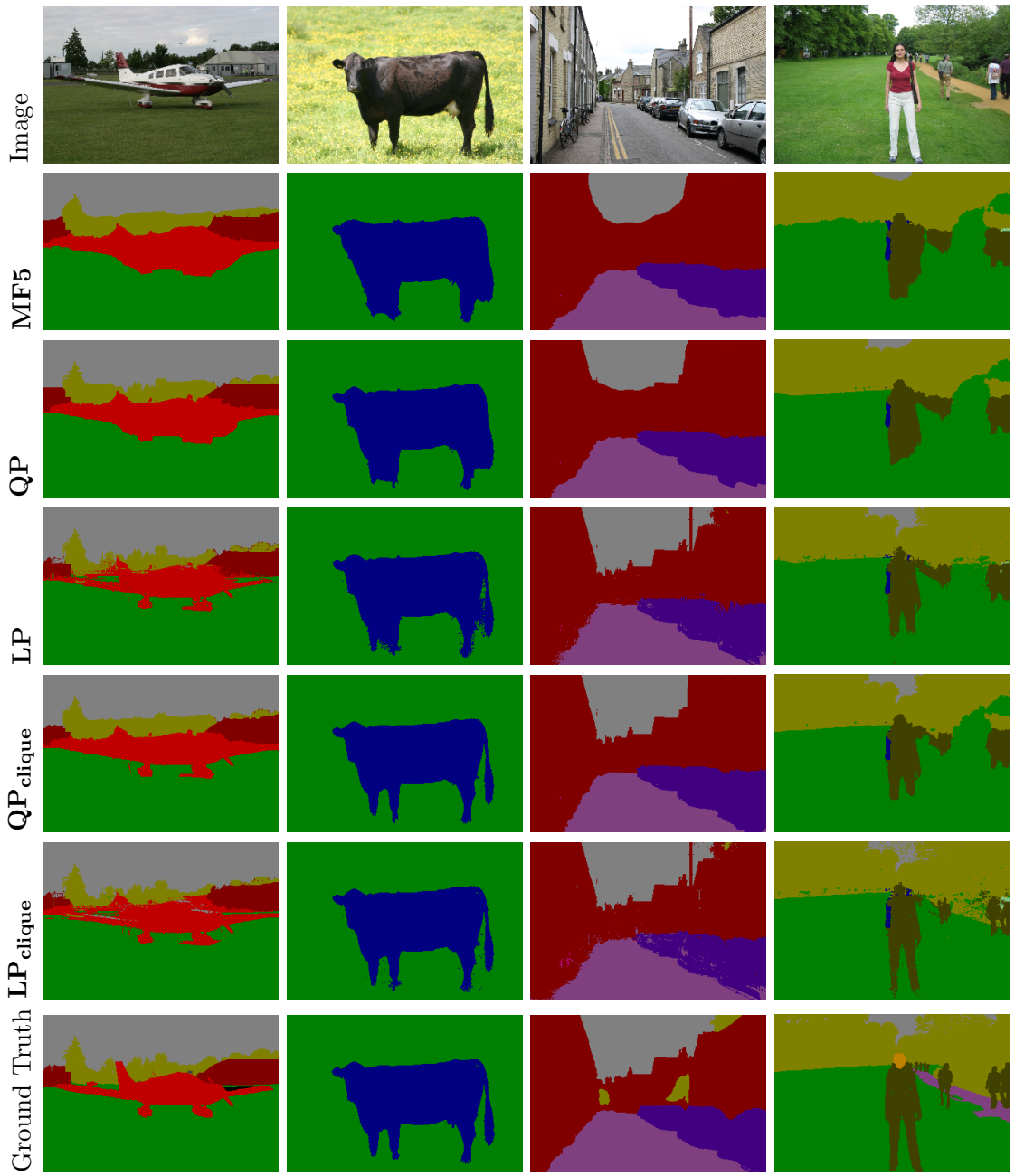


Figure 3. *Segmentation results for the parameters tuned to* $\boldsymbol{LP}_{clique}$ *and* $\boldsymbol{LP}$*. All of our methods provide visually pleasing segmentations, but particular attention is drawn to the accuracy of the edges for* $\boldsymbol{LP}_{clique}$ *and* $\boldsymbol{LP}$*, where the roofs of the building give sharp edges, which is not the case with* $\boldsymbol{QP}_{clique}$ *and* $\boldsymbol{QP}$*.*

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:

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

where:

$$\min_{\mathbf{y},\mathbf{z}} f(\mathbf{y},\mathbf{z}) = \boldsymbol{\phi}^T\mathbf{y} + \mathbf{y}^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T\mathbf{z} + (\mathbf{1_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{1_y}-\mathbf{y}). \tag{A.2}$$

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

$$\begin{aligned}\delta = \operatorname{argmin}_\delta \Big(&\delta^2\Big[(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}(\mathbf{s_y}-\mathbf{y}) + (\mathbf{s_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{s_y}-\mathbf{y})\Big]\\ &+\delta\Big[\boldsymbol{\phi}^T(\mathbf{s_y}-\mathbf{y}) + \mathbf{2}(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T(\mathbf{s_z}-\mathbf{z})\\ &-(\mathbf{1_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{s_y}-\mathbf{y}) - (\mathbf{s_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{1_y}-\mathbf{y})\Big]\\ &+\Big[\boldsymbol{\phi}^T\mathbf{y} + \mathbf{y}^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T\mathbf{z} + (\mathbf{1_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{1_y}-\mathbf{y})\Big]\Big).\end{aligned} \tag{A.3}$$

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

$$\begin{aligned}\delta^* = P_{[0,1]}\Bigg[&-\frac{1}{2}\frac{\boldsymbol{\phi}^T(\mathbf{s_y}-\mathbf{y}) + \mathbf{2}(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}\mathbf{y} + \mathbf{c}^T(\mathbf{s_z}-\mathbf{z})}{(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}(\mathbf{s_y}-\mathbf{y}) + (\mathbf{s_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{s_y}-\mathbf{y})}\\ &+\frac{1}{2}\frac{(\mathbf{1_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{s_y}-\mathbf{y}) - (\mathbf{s_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{1_y}-\mathbf{y})}{(\mathbf{s_y}-\mathbf{y})^T\boldsymbol{\Psi}(\mathbf{s_y}-\mathbf{y}) + (\mathbf{s_z}-\mathbf{z})^T\mathbf{C}\boldsymbol{H}(\mathbf{s_y}-\mathbf{y})}\Bigg],\end{aligned} \tag{A.4}$$

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:

$$\begin{aligned}\min_{\mathbf{y},\mathbf{w},\mathbf{v}} &\sum_a\sum_{i\in\mathcal{L}}\phi_{a:i}y_{a:i} + \sum_{a,b\neq a}\sum_i\frac{\mathrm{K}_{ab}}{2}v_{ab:i} + \sum_p C_p w_p + \frac{1}{2\lambda}||\mathbf{y}-\mathbf{y}^k||,\\ \text{s.t } &v_{ab:i} \geq y_{a:i} - y_{b:i} \quad \forall a,b\in\{1,...,N\} \quad a\neq b \quad \forall i\in\mathcal{L},\\ &v_{ab:i} \geq y_{b:i} - y_{a:i} \quad \forall a,b\in\{1,...,N\} \quad a\neq b \quad \forall i\in\mathcal{L},\\ &w_p \geq y_{c:pi} - y_{d:pi} \quad \forall c,d\in p \quad c\neq d \quad \forall i\in\mathcal{L} \quad \forall p,\\ &w_p \geq y_{d:pi} - y_{c:pi} \quad \forall c,d\in p \quad c\neq d \quad \forall i\in\mathcal{L} \quad \forall p,\\ &y_{a:i} \geq 0 \quad \forall a\in\{1,...,N\} \quad \forall i\in\mathcal{L},\\ &\sum_i y_{a:i} = 1 \quad \forall a\in\{1,...,N\}.\end{aligned} \tag{A.5}$$

The associated Lagrangian can thus be written as:

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

$$\begin{aligned}
&\sum_a \sum_i \phi_{a:i} y_{a:i} + \sum_{a,b\neq a} \sum_i \frac{\mathrm{K}_{ab}}{2} v_{ab:i} + \sum_p C_p w_p + \frac{1}{2\lambda}||\mathbf{y}-\mathbf{y}^k|| \\
&- \sum_{a,b\neq a} \sum_i \alpha^1_{ab:i}(y_{b:i} - y_{a:i} + v_{ab:i}) - \sum_{a,b\neq a} \sum_i \alpha^2_{ab:i}(y_{a:i} - y_{b:i} + v_{ab:i}) \\
&- \sum_p \sum_{\substack{c,d\in p \\ c\neq d}} \sum_i \mu^1_{cd:pi}(y_{d:i} - y_{c:i} + w_p) - \sum_p \sum_{\substack{c,d\in p \\ c\neq d}} \sum_i \mu^2_{cd:pi}(y_{c:i} - y_{d:i} + w_p)
\end{aligned} \tag{A.7}$$

$$+ \sum_a \beta_a \Big(1 - \sum_i y_{a:i}\Big) - \sum_a \sum_i \gamma_{a:i} y_{a:i}, \tag{A.8}$$

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

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 minimised 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 non-zero, the problem is unbounded and hence the minimisation 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 $\nabla_{\mathbf{v}}(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = 0$ and $\nabla_{\mathbf{w}}(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = 0$ 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:

$$\alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{K_{ab}}{2} \qquad \forall a \neq b \qquad \forall i \in \mathcal{L}, \tag{A.9}$$

$$\sum_i \sum_{c,d} \mu^1_{cd:pi} + \mu^2_{cd:pi} = C_p \qquad \forall c,d \in p \qquad \forall i \in \mathcal{L}. \tag{A.10}$$

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 $\nabla_{\mathbf{y}}(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = 0$, it is beneficial to reorder the terms in the Lagrangian, using equations (A.9) and (A.10), we can arrange the Lagrangian like so:

(A.11)

$$\begin{aligned}
L(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{w},\mathbf{v}) = &\sum_a \sum_i (\phi_{a:i} - \beta_a - \gamma_a) y_{a:i} + \frac{1}{2\lambda} \sum_{a,b\neq a} \sum_i (y_{a:i} - y^k_{a:i})^2 + \sum_a \beta_a \\
&+ \sum_{a,b\neq a} \sum_i (\alpha^1_{ab:i} - \alpha^2_{ab:i} - \alpha^1_{ba:i} + \alpha^2_{ba:i}) y_{a:i} \\
&+ \sum_p \sum_{c,d} \sum_i (\mu^1_{cd:pi} - \mu^2_{cd:pi} - \mu^1_{dc:pi} + \mu^2_{dc:pi}) y_{a:i}
\end{aligned}$$

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

$$\frac{1}{\lambda}(y_{a:i} - y^k_{a:i}) = -\sum_{a,b\neq a}\sum_{i}(\alpha^1_{ab:i} - \alpha^2_{ab:i} - \alpha^1_{ba:i} + \alpha^2_{ba:i}) + \beta_a + \gamma_{a:i} - \sum_{p}\sum_{c,d}\sum_{i}(\mu^1_{cd:pi} - \mu^2_{cd:pi} - \mu^1_{dc:pi} + \mu^2_{dc:pi}) - \phi_{a:i}, \tag{A.12}$$

By utilising the matrices introduced in equations (5.4), (5.5) and (5.6) this expression can be concisely written in vector form:

$$\mathbf{y} = \lambda(\mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}) + \mathbf{y}^k. \tag{A.13}$$

Substituting this equation into the Lagrangian defined in (A.11) yields the following Lagrange dual problem:

(A.14)

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

$$+ \langle \mathbf{A}\boldsymbol{\alpha} + \mathbf{U}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}, \mathbf{y}^k\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \tag{A.15}$$

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

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

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

A.3. 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.4 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:

$$\mathbf{s}_\alpha = \underset{\mathbf{s}_\alpha \in \mathcal{A}}{\operatorname{argmin}} \langle \mathbf{s}_\alpha, \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\rangle. \tag{A.16}$$

By using equation(A.13), $\nabla_{\boldsymbol{\alpha}} g(.)$ is given as:

$$\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \mathbf{A}^T \tilde{y}^t. \tag{A.17}$$

Note that the feasible set $\mathcal{A}$ is separable and can be written as $\mathcal{A} = \prod_{a,b,i} \mathcal{A}_{ab:i}$, where $\mathcal{A}_{ad:i} = \{(\alpha^1_{ad:i}, \alpha^2_{ab:i}) | \alpha^1_{ab:i} + \alpha^2_{ab:i} = K_{ab}, \alpha^1_{ad:i}, \alpha^2_{ab:i} \geq 0\}$. It is possible exploit this separability

and compute the conditional gradient $\mathbf{s}_\alpha$ for each Lagrange multiplier like so:

$$\min_{s_{\alpha^1_{ab:i}}, s_{\alpha^2_{ab:i}}} s_{\alpha^1_{ab:i}} \nabla_{\alpha^1_{ad:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) + s_{\alpha^2_{ab:i}} \nabla_{\alpha^2_{ad:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \tag{A.18}$$
$$\text{s.t} \quad s_{\alpha^1_{ab:i}}, s_{\alpha^2_{ab:i}} \in \mathcal{A}_{ab:i}.$$

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

$$\nabla_{\alpha^1_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{b:i} - \tilde{y}^t_{a:i} \tag{A.19}$$
$$\nabla_{\alpha^2_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{a:i} - \tilde{y}^t_{b:i}, \tag{A.20}$$

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:

$$s_{\alpha^1_{ab:i}} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \geq \tilde{y}^t_{b:i} \\ 0 & \text{otherwise,} \end{cases} \tag{A.21}$$

$$s_{\alpha^2_{ab:i}} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i} \\ 0 & \text{otherwise.} \end{cases} \tag{A.22}$$

Which by utilising matrix $\mathbf{A}$, introduced in equation (5.4), can be concisely written as:

$$(\mathbf{A}\mathbf{s}_\alpha)_{a:i} = \sum_b (K_{ab} \mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}] - K_{ab} \mathbb{1}[\tilde{y}^t_{b:i} \leq \tilde{y}^t_{a:i}]) \tag{A.23}$$

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:

$$\mathbf{s}_\mu = \underset{\mathbf{s}_\mu \in \mathcal{U}}{\operatorname{argmin}} \langle \mathbf{s}_\mu, \nabla_{\boldsymbol{\mu}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \rangle. \tag{A.24}$$

By using equation(A.13), $\nabla_{\boldsymbol{\mu}} g(.)$ is given as:

$$\nabla_{\boldsymbol{\mu}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \mathbf{U}^T \tilde{y}^t. \tag{A.25}$$

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

$$\min_{s_{\mu^1_{cd:pi}}, s_{\mu^2_{cd:pi}}} s_{\mu^1_{cd:pi}} \nabla_{\mu^1_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) + s_{\mu^2_{cd:pi}} \nabla_{\mu^2_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \tag{A.26}$$
$$\text{s.t} \quad s_{\mu^1_{cd:pi}}, s_{\mu^2_{cd:pi}} \in \mathcal{U}_p.$$

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

$$\nabla_{\mu^1_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{d:pi} - \tilde{y}^t_{c:pi}, \tag{A.27}$$
$$\nabla_{\mu^2_{cd:pi}} g(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{c:pi} - \tilde{y}^t_{d:pi}, \tag{A.28}$$

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:

$$s_{\mu^1_{cd:pi}} = \begin{cases} C_p/2 & \text{if } \tilde{y}^t_{c:pi} = \max_{c\in p, i\in\mathcal{L}} \tilde{y}^t_{c:pi} \text{ and } \tilde{y}^t_{d:i} = \min_{d\in p, i\in\mathcal{L}} \tilde{y}^t_{d:pi} \\ 0 & \text{otherwise,} \end{cases} \tag{A.29}$$

$$s_{\mu^2_{cd:pi}} = \begin{cases} C_p/2 & \text{if } \tilde{y}^t_{c:pi} = \min_{c\in p, i\in\mathcal{L}} \tilde{y}^t_{c:pi} \text{ and } \tilde{y}^t_{d:i} = \max_{d\in p, i\in\mathcal{L}} \tilde{y}^t_{d:pi} \\ 0 & \text{otherwise.} \end{cases} \tag{A.30}$$

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

$$(\mathbf{U}\mathbf{s}_\mu)_{c:pi} = \begin{cases} C_p & \text{if } \tilde{y}^t_{c:pi} \leq \tilde{y}^t_{d:pj} \quad \forall d \in p \quad \forall j \in \mathcal{L} \\ -C_p & \text{if } \tilde{y}^t_{c:pi} > \tilde{y}^t_{d:pj} \quad \forall d \in p \quad \forall j \in \mathcal{L} \\ 0 & \text{otherwise.} \end{cases} \tag{A.31}$$

A.4. Advanced Filtering Method Appendix

Original filtering method Using the filter based method introduced by Adams et al. [1], the computational complexity of the pairwise potential computation can be reduced from $\mathcal{O}(N^2)$ down to $\mathcal{O}(N)$. The filter based method requires the operations to contain Gaussian kernels and approximates:

$$\forall a \in \{1, ..., N\}, \qquad v'_a = \sum_b k(\mathbf{f}_a, \mathbf{f}_b) v_b, \tag{A.32}$$

where $v'_a, v_b \in \mathbb{R}$, $b \in \{1, ..., N\}$ and $k(\mathbf{f}_a, \mathbf{f}_b)$ is a Gaussian kernel described in section 3.1.2 and $\mathbf{f}_a$ and $\mathbf{f}_b$ represent the feature vectors for the random variable X_a and X_b respectively. The filter based method utilises the permutohedral lattice to perform fast high dimensional Gaussian filtering. The key steps to the high dimensional Gaussian filtering method are given in this section, however the interested reader is referred to [1] for more information.The first stage of the filtering method is to construct the permutohedral lattice, this is achieved via embedding the feature vectors $\mathbf{f}_a \in \mathbb{R}^d$ into a d-dimensional hyperplane. The next stage involves *splatting* the input values onto the vertices of the permutohedral lattice using barycentric-interpolation. Once the splatting stage is complete, the values of the vertices are *blurred* using a truncated Gaussian filter along each dimension of the d-dimensional hyperplane. The final stage, known as *slicing*, propagates the vertices back to the feature vectors using the same barycentric weights used in splatting. The computational complexity of this filtering method is given as $\mathcal{O}(dN)$ [1, 14].

Modified filtering method The filtering method detailed above is not suitable in its current form as it has no mechanism to handle the ordering constraint $\mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}]$ imposed by the conditional gradient $\mathbf{s}_\alpha$. To this extent we modified the filtering method above by uniformly discretising the continuous interval $[0, 1]$ into H bins, with each feature point belonging to one of these bins. H permutohedral lattices are then instantiated, one for each level $h \in \{0, ..., H-1\}$. The feature points belonging to the bin q are then splated onto the permutohedral lattices corresponding to the levels $q \leq h < H$. Blurring is then performed independently on all

lattices. This is advantageous as it enables a feature point b to influence the feature point a if $y_a \geq y_b$. The slicing stage recovers the feature point residing in bin q from the q^{th} permutohedral lattice. The computational complexity of this method is $\mathcal{O}(dNH)$, and in practice we used a value of H as small as 10. For a more in depth explanation, please consult our previous work [2].

A.5. Additional Results In this section we present the results for when the algorithms are tuned to $\mathbf{LP_{clique}}$ and $\mathbf{LP}$, displayed in Table 3. As is consistent with our previous results, it can be seen that $\mathbf{QP}$ achieves lower energies then $\mathbf{MF5}$, 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 then $\mathbf{QP_{clique}}$. This is not an astonishing result as the LP relaxation is known to achieve a tighter relaxation then the QP [17]. An interesting observation is $\mathbf{QP}$ obtaining a slightly higher segmentation accuracy then $\mathbf{QP_{clique}}$, but $\mathbf{QP_{clique}}$ obtaining a slightly higher IoU. A plausible explanation is a poor estimation of cliques (from mean-shift) causes inaccurate boundaries and pixels on the edges to be labelled incorrectly. Visual results can be seen in Figure 4 and a plot of the decrease in energy can be seen in Figure 2.

Algorithm	**Avg.E** ($\times 10^7$)	**Time(s)**	**Acc(%)**	**IoU(%)**
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
$\mathbf{QP_{clique}}$	46.1	1.75	83.56	**57.81**
$\mathbf{LP_{clique}}$	**44.1**	49.3	81.49	55.81

Table 3

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 $\boldsymbol{LP}$. $\boldsymbol{QP_{clique}}$ and $\boldsymbol{QP}$ obtain the greatest segmentation accuracy, which is expected, as the highest performing algorithms are the ones in which we tune the parameters for.

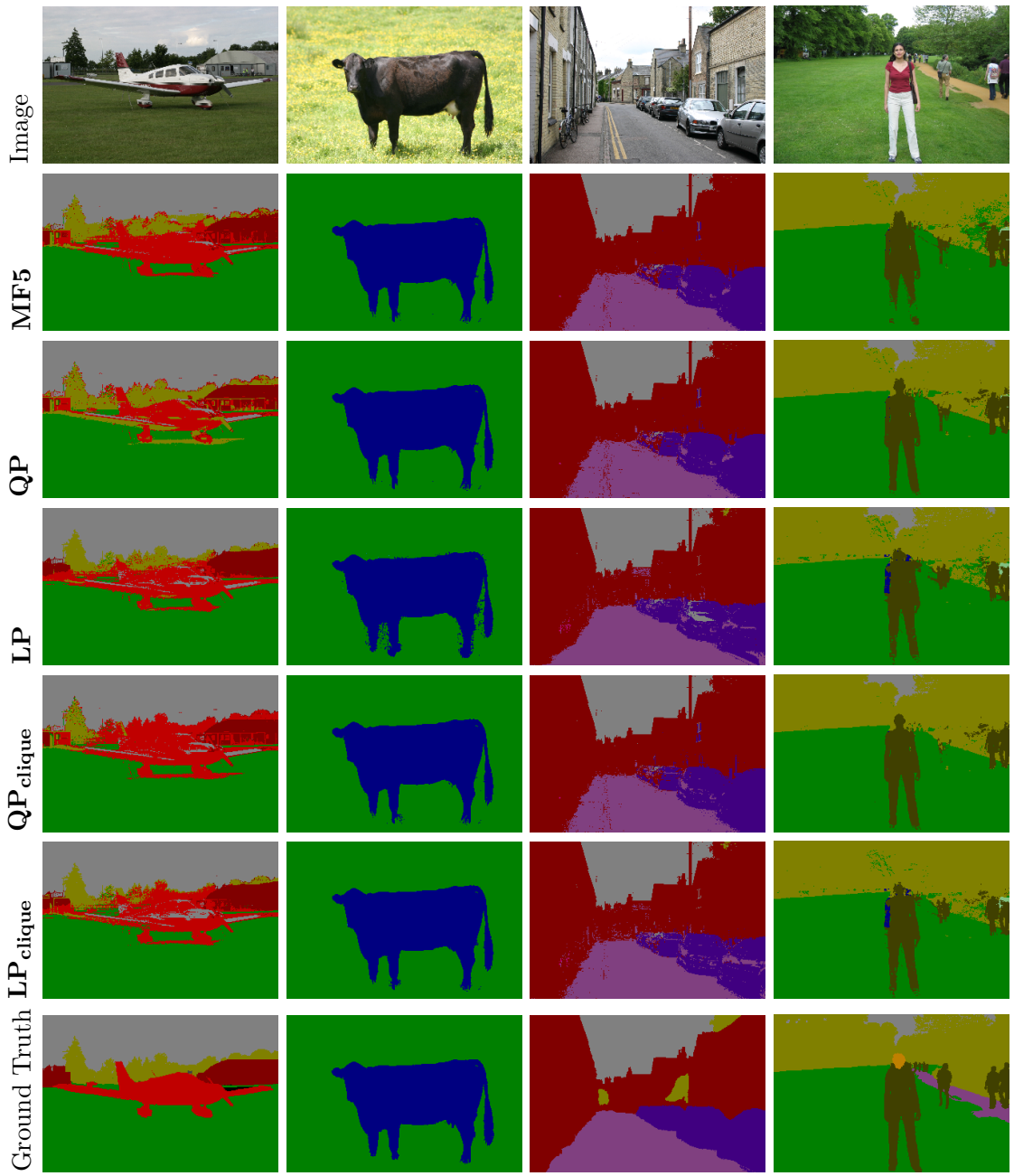


Figure 4. *Segmentation results for the parameters tuned to* $\boldsymbol{QP}_{\boldsymbol{clique}}$ *and* $\boldsymbol{QP}$*. All of our methods provide visually good segmentations, but particular attention is drawn to the quality of the segmentations for* $\boldsymbol{QP}_{\boldsymbol{clique}}$ *and* $\boldsymbol{QP}$*, which obtain the best segmentations.*

References

[1] A. Adams, J. Baek, and M. Abraham. Fast high-dimensional filtering using the permutohedral lattice. *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 *CVPR*, 2017.

[3] F. R. Bach. Duality between subgradient and conditional gradient methods. *SLAM Journal on Optimization*, 2015.

[4] P. Baqué, T. Bagautdinov, F. Fleuret, and P. Fua. Principled parallel mean-field inference for discrete random fields. In *CVPR*, 2016.

[5] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *SODA*, 2001.

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[8] A. Desmaison, R. Bunel, P. Kohli, P. H. S. Torr, and M. P. Kumar. Efficient continuous relaxations for dense CRF. *ECCV*, 2016.

[9] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 1956.

[10] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM*, 2002.

[11] P. Kohli, P. Kumar, and P. Torr. P3 & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.

[12] P. Kohli, T. Minka, and J. Winn. Msrc dataset. https://www.microsoft.com/en-us/research/project/image-understanding/. Accessed: 2017-03-14.

[13] D. Koller and N. Friedman. Probabilistic graphical models: principles and techniques. *MIT press*, 2009.

[14] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011.

[15] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *ICML*, 2013.

[16] P. Krähenbühl, V. Koltun, and S. University. Trained unary potentials. Stanford University, https://graphics.stanford.edu/projects/densecrf/unary/. Accessed: 2017-02-27.

[17] M. P. Kumar, V. Kolmogorov, and P. H. S. Torr. An analysis of convex relaxations for map estimation of discrete mrfs. *Journal of Machine Learning Reaserch*, 10:71–106, 2008.

[18] S. Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[19] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate frank-wolfe optimization for structural SVMs. *ICML*, 2012.

[20] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Graph cut based inference with co-occurrence statistics. In *ECCV*, 2010.

[21] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical CRFs for object class image segmentation. pages 739–746, 2009.

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
[23] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2014.
[24] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. *ICML*, 2006.
[25] A. Schwing and R. Urtasun. Fully connected deep structured networks. *CoRR*, 2015.
[26] 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. *IJCV*, 81(1):2–23, Jan. 2009.
[27] J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. *In NIPS*, 2012.
[28] M. Tappen, C. Liu, E. Adelson, and W. Freeman. Learning gaussian conditional random fields for low-level vision. In *CVPR*, 2007.
[29] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.
[30] 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. *International Journal of Computer Vision*, 2014.
[31] P. Wang, C. Shen, and A. van den Hengel. Efficient SDP inference for fully-connected CRFs based on low-rank decomposition. In *CVPR*, 2015.
[32] X. Xiao and D. Chen. Multiplicative iteration for nonnegative quadratic programming. *Numerical Linear Algebra with Applications*, 2014.
[33] Y. Zhang and T. Chen. Efficient inference for fully-connected CRFs with stationarity. In *CVPR*, 2012.
[34] 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. *ICCV*, 2015.