Efficient Linear Programming for Dense CRFs - Supplementary Material

Thalaiyasingam Ajanthan[1], Alban Desmaison[2], Rudy Bunel[2], Mathieu Salzmann[3], Philip H.S. Torr[2], and M. Pawan Kumar[2,4]

[1]Australian National University & Data61, CSIRO [2]Department of Engineering Science, University of Oxford
[3]Computer Vision Laboratory, EPFL [4]Alan Turing Institute

A. Proximal minimization for LP relaxation

In this section, we give the detailed derivation of our proximal minimization algorithm for the LP relaxation.

A.1. Dual formulation

Let us first write the proximal problem (5) in the main paper in the standard form by introducing auxiliary variables $z_{ab:i}$.

$$\min_{\mathbf{y},\mathbf{z}} \quad \sum_a \sum_i \phi_{a:i}\, y_{a:i} + \sum_{a,b\neq a} \sum_i \frac{K_{ab}}{2} z_{ab:i} + \frac{1}{2\lambda}\|\mathbf{y}-\mathbf{y}^k\|^2 \,, \tag{15a}$$

$$\text{s.t.} \quad z_{ab:i} \geq y_{a:i} - y_{b:i} \quad \forall\, a, b\neq a \quad \forall\, i \in \mathcal{L} \,, \tag{15b}$$

$$z_{ab:i} \geq y_{b:i} - y_{a:i} \quad \forall\, a, b\neq a \quad \forall\, i \in \mathcal{L} \,, \tag{15c}$$

$$\sum_i y_{a:i} = 1 \quad \forall\, a \in \{1\ldots n\} \,, \tag{15d}$$

$$y_{a:i} \geq 0 \quad \forall\, a \in \{1\ldots n\} \quad \forall\, i \in \mathcal{L} \,. \tag{15e}$$

Here, the shorthand notation $\sum_{a,b\neq a}$ denotes the summation over both variables a and b such that $a \neq b$, writing it explicitly: $\sum_a \sum_{b\neq a}$. Furthermore, the shorthand $\forall\, a, b \neq a$ denotes for all $a \in \{1\ldots n\}$ and $b \in \{\{1\ldots n\}, b \neq a\}$. Note that, these shorthand notations are consistent with the main paper, and we mention it here to make it clearer.

We introduce three blocks of dual variables. Namely, $\boldsymbol{\alpha} = \{\alpha^1_{ab:i}, \alpha^2_{ab:i} \mid a, b \neq a, i \in \mathcal{L}\}$ for the constraints in Eqs. (15b) and (15c), $\boldsymbol{\beta} = \{\beta_a \mid a \in \{1\ldots n\}\}$ for Eq. (15d) and $\boldsymbol{\gamma} = \{\gamma_{a:i} \mid a \in \{1\ldots n\}, i \in \mathcal{L}\}$ for Eq. (15e), respectively. Now we can write the Lagrangian associated with this primal problem [2]:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \min_{\mathbf{y},\mathbf{z}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) = & \sum_a \sum_i \phi_{a:i}\, y_{a:i} + \sum_{a,b\neq a} \sum_i \frac{K_{ab}}{2} z_{ab:i} + \frac{1}{2\lambda} \sum_a \sum_i \left(y_{a:i} - y^k_{a:i}\right)^2 \\ & + \sum_{a,b\neq a} \sum_i \alpha^1_{ab:i} \left(y_{a:i} - y_{b:i} - z_{ab:i}\right) + \sum_{a,b\neq a} \sum_i \alpha^2_{ab:i} \left(y_{b:i} - y_{a:i} - z_{ab:i}\right) \\ & + \sum_a \beta_a \left(1 - \sum_i y_{a:i}\right) - \sum_a \sum_i \gamma_{a:i}\, y_{a:i} \,, \\ \text{s.t.} \quad & \alpha^1_{ab:i}, \alpha^2_{ab:i} \geq 0 \quad \forall\, a, b\neq a \quad \forall\, i \in \mathcal{L} \,, \\ & \gamma_{a:i} \geq 0 \quad \forall\, a \in \{1\ldots n\} \quad \forall\, i \in \mathcal{L} \,. \end{aligned} \tag{16}$$

Here the vector $\boldsymbol{\alpha}$ has $p = 2n(n-1)m$ elements.

Note that the dual problem is obtained by minimizing the Lagrangian over the primal variables $(\mathbf{y}, \mathbf{z})$. With respect to $\mathbf{z}$, the Lagrangian is linear and when $\nabla_{\mathbf{z}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) \neq 0$, the minimization in $\mathbf{z}$ yields $-\infty$. This situation is not useful as the dual function is unbounded. Therefore we restrict ourselves to the case where $\nabla_{\mathbf{z}} L(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{y}, \mathbf{z}) = 0$. By differentiating with respect to $\mathbf{z}$ and setting the derivatives to zero, we obtain

$$\alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{K_{ab}}{2} \quad \forall\, a, b \neq a \quad \forall\, i \in \mathcal{L}\,. \tag{17}$$

The minimum of the Lagrangian with respect to $\mathbf{y}$ is attained when $\nabla_{\mathbf{y}} L(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{y}, \mathbf{z}) = 0$. Before differentiating with respect to $\mathbf{y}$, let us rewrite the Lagrangian using Eq. (17) and reorder the terms:

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{y}, \mathbf{z}) = &\sum_a \sum_i (\phi_{a:i} - \beta_a - \gamma_{a:i})\, y_{a:i} + \frac{1}{2\lambda} \sum_a \sum_i \left(y_{a:i} - y^k_{a:i}\right)^2 + \sum_{a,b\neq a} \sum_i \left(\alpha^1_{ab:i} - \alpha^2_{ab:i}\right) y_{a:i} \\ &+ \sum_{a,b\neq a} \sum_i \left(\alpha^2_{ba:i} - \alpha^1_{ba:i}\right) y_{a:i} + \sum_a \beta_a\,. \end{aligned} \tag{18}$$

Now, by differentiating with respect to $\mathbf{y}$ and setting the derivatives to zero, we get

$$\frac{1}{\lambda}\left(y_{a:i} - y^k_{a:i}\right) = -\sum_{b\neq a} \left(\alpha^1_{ab:i} - \alpha^2_{ab:i} + \alpha^2_{ba:i} - \alpha^1_{ba:i}\right) + \beta_a + \gamma_{a:i} - \phi_{a:i} \quad \forall\, a \in \{1 \ldots n\} \quad \forall\, i \in \mathcal{L}\,. \tag{19}$$

Note that, here we used the notation $\sum_{b\neq a}$ to denote the summation over b such that $b \neq a$. This is not to be confused with the shorthand notation $\sum_{a,b\neq a}$, for example in Eq. (15a). Writing the above equation in vector form yields

$$\frac{1}{\lambda}\left(\mathbf{y} - \mathbf{y}^k\right) = A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}\,, \tag{20}$$

where $A \in \mathrm{I\!R}^{\mathrm{nm}\times\mathrm{p}}$ and $B \in \mathrm{I\!R}^{\mathrm{nm}\times\mathrm{n}}$, with

$$\begin{aligned} (A\boldsymbol{\alpha})_{a:i} &= -\sum_{b\neq a} \left(\alpha^1_{ab:i} - \alpha^2_{ab:i} + \alpha^2_{ba:i} - \alpha^1_{ba:i}\right)\,, \\ (B\boldsymbol{\beta})_{a:i} &= \beta_a\,. \end{aligned} \tag{21}$$

Proposition A.1 (Properties of matrix A)**.** Let $\mathbf{x} \in \mathrm{I\!R}^{\mathrm{nm}}$. Then, for all $a \neq b$ and $i \in \mathcal{L}$,

$$\begin{aligned} \left(A^T\mathbf{x}\right)_{ab:i^1} &= x_{b:i} - x_{a:i}\,, \\ \left(A^T\mathbf{x}\right)_{ab:i^2} &= x_{a:i} - x_{b:i}\,. \end{aligned}$$

Here, the index $ab : i^1$ denotes the element corresponding to $\alpha^1_{ab:i}$.

Proof. This can be easily proved by inspecting the matrix A. □

Proposition A.2 (Properties of matrix B)**.** The matrix $B \in \mathrm{I\!R}^{\mathrm{nm}\times\mathrm{n}}$ defined in Eq. (21) satisfies the following properties:

1. Let $\mathbf{x} \in \mathrm{I\!R}^{\mathrm{nm}}$. Then, $\left(B^T\mathbf{x}\right)_a = \sum_{i\in\mathcal{L}} x_{a:i}$ for all $a \in \{1 \ldots n\}$.
2. $B^TB = mI$, where $I \in \mathrm{I\!R}^{\mathrm{n}\times\mathrm{n}}$ is the identity matrix.
3. BB^T is a block diagonal matrix, with each block $B_a = \mathbf{1}$ for all $a \in \{1 \ldots n\}$, where $\mathbf{1} \in \mathrm{I\!R}^{\mathrm{m}\times\mathrm{m}}$ is the matrix of all ones.

Proof. Note that, from Eq. (21), the matrix B simply repeats the elements β_a m times. In particular, for $m = 3$, the matrix B has the following form:

$$B = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \vdots & & & & \vdots \\ 1 & 0 & \cdots & \cdots & \cdots & \vdots \\ 0 & 1 & & & & \vdots \\ \vdots & 1 & & & & \vdots \\ \vdots & 1 & & & & \vdots \\ \vdots & 0 & & & & \vdots \\ \vdots & \vdots & & & & 0 \\ \vdots & \vdots & & & & 1 \\ \vdots & \vdots & & & & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{22}$$

Therefore, multiplication by B^T amounts to summing over the labels. From this, the other properties can be proved easily. □

Now, using Eqs. (17) and (20), the dual problem can be written as:

$$\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \; & g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\lambda}{2}\|A\boldsymbol{\alpha}+B\boldsymbol{\beta}+\boldsymbol{\gamma}-\boldsymbol{\phi}\|^2 + \left\langle A\boldsymbol{\alpha}+B\boldsymbol{\beta}+\boldsymbol{\gamma}-\boldsymbol{\phi}, \mathbf{y}^k\right\rangle - \langle \mathbf{1},\boldsymbol{\beta}\rangle \,, \\
\text{s.t.} \quad & \gamma_{a:i} \geq 0 \quad \forall\, a \in \{1\dots n\} \quad \forall\, i \in \mathcal{L}\,, \\
& \boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{K_{ab}}{2},\, \forall\, a, b \neq a,\, \forall\, i \in \mathcal{L} \\ \alpha^1_{ab:i}, \alpha^2_{ab:i} \geq 0,\, \forall\, a \neq b,\, \forall\, i \in \mathcal{L} \end{array} \right\} .
\end{aligned} \tag{23}$$

Here, $\mathbf{1}$ denotes the vector of all ones of appropriate dimension. Note that we converted our problem to a minimization one by changing the sign of all the terms. This is exactly the dual problem given in (7) in the main paper.

A.2. Optimizing over β and γ

In this section, for a fixed value of $\boldsymbol{\alpha}^t$, we optimize over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. To this end, since the dual variables $\boldsymbol{\beta}$ are unconstrained, the minimum value of the dual objective g is attained when $\nabla_{\boldsymbol{\beta}} g(\boldsymbol{\alpha}^t,\boldsymbol{\beta},\boldsymbol{\gamma}) = 0$. Hence, by differentiating with respect to $\boldsymbol{\beta}$ and setting the derivatives to zero, we obtain

$$\boldsymbol{\beta} = -B^T(A\boldsymbol{\alpha}^t + \boldsymbol{\gamma} - \boldsymbol{\phi})/m \,. \tag{24}$$

Note that, from Proposition A.2, $B^T\mathbf{y}^k = \mathbf{1}$ since $\mathbf{y}^k \in \mathcal{M}$ (defined in Eq. (4) in the main paper), and $B^TB = mI$. Both these identities are used to simplify the above equation. By substituting $\boldsymbol{\beta}$ in the dual problem (23) with the above expression, the optimization problem over $\boldsymbol{\gamma}$ takes the following form:

$$\begin{aligned}
\min_{\boldsymbol{\gamma}} \; & g(\boldsymbol{\alpha}^t,\boldsymbol{\gamma}) = \frac{\lambda}{2}\|D(A\boldsymbol{\alpha}^t+\boldsymbol{\gamma}-\boldsymbol{\phi})\|^2 + \left\langle D(A\boldsymbol{\alpha}^t+\boldsymbol{\gamma}-\boldsymbol{\phi}), \mathbf{y}^k\right\rangle + \frac{1}{m}\left\langle \mathbf{1}, A\boldsymbol{\alpha}^t+\boldsymbol{\gamma}-\boldsymbol{\phi}\right\rangle \,, \\
\text{s.t.} \quad & \boldsymbol{\gamma} \geq \mathbf{0}\,,
\end{aligned} \tag{25}$$

where $D = I - \frac{BB^T}{m}$.

Proposition A.3 (Properties of matrix D)**.** The matrix $D = I - \frac{BB^T}{m}$ satisfies the following properties:

1. D is block diagonal, with each block matrix $D_a = I - \mathbf{1}/m$, where $I \in \mathbb{R}^{m\times m}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^{m\times m}$ is the matrix of all ones.

2. $D^TD = D$.

Proof. From Proposition A.2, the matrix BB^T is block diagonal and therefore D is block diagonal with each block matrix $D_a = I - \mathbf{1}/m$. Note that the block matrices D_a are identical. The second property can be easily proved using simple matrix algebra. □

Note that, since $\mathbf{y}^k \in \mathcal{M}$, from Proposition A.2, $B^T\mathbf{y}^k = \mathbf{1}$. Using this fact, the identity $D^TD = D$, and by removing the constant terms, the optimization problem over $\boldsymbol{\gamma}$ can be simplified:

$$\begin{aligned}
\min_{\boldsymbol{\gamma}} \; & g(\boldsymbol{\alpha}^t,\boldsymbol{\gamma}) = \frac{\lambda}{2}\boldsymbol{\gamma}^TD\boldsymbol{\gamma} + \langle \boldsymbol{\gamma}, \lambda D(A\boldsymbol{\alpha}^t - \boldsymbol{\phi}) + \mathbf{y}^k\rangle \,, \\
\text{s.t.} \quad & \boldsymbol{\gamma} \geq \mathbf{0}\,.
\end{aligned} \tag{26}$$

Furthermore, since D is block diagonal, we obtain

$$\min_{\boldsymbol{\gamma}\geq\mathbf{0}} g(\boldsymbol{\alpha}^t,\boldsymbol{\gamma}) = \sum_a \min_{\boldsymbol{\gamma}_a\geq\mathbf{0}} \frac{\lambda}{2}\boldsymbol{\gamma}_a^TD_a\boldsymbol{\gamma}_a + \langle \boldsymbol{\gamma}_a, \lambda D_a\left((A\boldsymbol{\alpha}^t)_a - \boldsymbol{\phi}_a\right) + \mathbf{y}^k_a\rangle \,, \tag{27}$$

where the notation $\boldsymbol{\gamma}_a$ denotes the vector $\{\gamma_{a:i} \mid i \in \mathcal{L}\}$. Each of these m dimensional quadratic programs (QP) are optimized using the iterative algorithm [7]. Before we give the update equation, let us first write our problem in the form used in [7]. For a given $a \in \{1\dots n\}$, this yields

$$\min_{\boldsymbol{\gamma}_a\geq\mathbf{0}} \frac{1}{2}\boldsymbol{\gamma}_a^TQ\boldsymbol{\gamma}_a - \langle \boldsymbol{\gamma}_a, \mathbf{h}_a\rangle \,, \tag{28}$$

where

$$Q = \lambda \left(I - \frac{\mathbf{1}}{m} \right) , \tag{29}$$
$$\mathbf{h}_a = -Q \left((A\boldsymbol{\alpha}^t)_a - \boldsymbol{\phi}_a \right) - \mathbf{y}_a^k .$$

This is exactly the QP given in (10). Hence, at each iteration, the element-wise update equation has the following form:

$$\gamma_{a:i} = \gamma_{a:i} \left[\frac{2\,(Q^- \boldsymbol{\gamma}_a)_i + h^+_{a:i} + c}{(|Q|\boldsymbol{\gamma}_a)_i + h^-_{a:i} + c} \right] , \tag{30}$$

where $Q^- = \max(-Q, 0)$, $|Q| = \text{abs}(Q)$, $h^+_{a:i} = \max(h_{a:i}, 0)$ and $h^-_{a:i} = \max(-h_{a:i}, 0)$ and $0 < c \ll 1$. These max and abs operations are element-wise. We refer the interested reader to [7] for more detail on this update rule.

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 into a multiplication by the 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 $\mathcal{O}(m)$. Once the optimal $\boldsymbol{\gamma}$ for a given $\boldsymbol{\alpha}^t$ is computed, the corresponding optimal $\boldsymbol{\beta}$ is given by Eq. (24).

A.3. Conditional gradient computation

The conditional gradient with respect to $\boldsymbol{\alpha}$ is obtained by solving the following linearization problem:

$$\mathbf{s}^t = \underset{\hat{\mathbf{s}} \in \mathcal{C}}{\text{argmin}} \left\langle \hat{\mathbf{s}}, \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \right\rangle , \tag{31}$$

where

$$\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = A^T \tilde{\mathbf{y}}^t , \tag{32}$$

with $\tilde{\mathbf{y}}^t = \lambda \left(A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi} \right) + \mathbf{y}^k$ using Eq. (20).

Note that the feasible set $\mathcal{C}$ is separable, *i.e.*, it can be written as $\mathcal{C} = \prod_{a,\, b \neq a,\, i \in \mathcal{L}} \mathcal{C}_{ab:i}$, with $\mathcal{C}_{ab:i} = \left\{ (\alpha^1_{ab:i}, \alpha^2_{ab:i}) \mid \alpha^1_{ab:i} + \alpha^2_{ab:i} = K_{ab}/2, \alpha^1_{ab:i}, \alpha^2_{ab:i} \geq 0 \right\}$. Therefore, the conditional gradient can be computed separately, corresponding to each set $\mathcal{C}_{ab:i}$. This yields

$$\begin{aligned} \min_{\hat{s}^1_{ab:i}, \hat{s}^2_{ab:i}} \quad & \hat{s}^1_{ab:i} \nabla_{\alpha^1_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) + \hat{s}^2_{ab:i} \nabla_{\alpha^2_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) , \\ \text{s.t.} \quad & \hat{s}^1_{ab:i} + \hat{s}^2_{ab:i} = K_{ab}/2 , \\ & \hat{s}^1_{ab:i}, \hat{s}^2_{ab:i} \geq 0 , \end{aligned} \tag{33}$$

where, using Proposition A.1, the gradients can be written as:

$$\nabla_{\alpha^1_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{b:i} - \tilde{y}^t_{a:i} , \tag{34}$$
$$\nabla_{\alpha^2_{ab:i}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \tilde{y}^t_{a:i} - \tilde{y}^t_{b:i} .$$

Hence, the minimum is attained at:

$$s^1_{ab:i} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \geq \tilde{y}^t_{b:i} \\ 0 & \text{otherwise} , \end{cases} \tag{35}$$
$$s^2_{ab:i} = \begin{cases} K_{ab}/2 & \text{if } \tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i} \\ 0 & \text{otherwise} . \end{cases}$$

Now, from Eq. (21), $A\mathbf{s}^t$ takes the following form:

$$\begin{aligned} \left(A\mathbf{s}^t \right)_{a:i} &= -\sum_{b \neq a} \left(\frac{K_{ab}}{2} \mathbb{1}[\tilde{y}^t_{a:i} \geq \tilde{y}^t_{b:i}] - \frac{K_{ab}}{2} \mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}] + \frac{K_{ba}}{2} \mathbb{1}[\tilde{y}^t_{b:i} \leq \tilde{y}^t_{a:i}] - \frac{K_{ba}}{2} \mathbb{1}[\tilde{y}^t_{b:i} \geq \tilde{y}^t_{a:i}] \right) , \\ &= -\sum_{b} \left(K_{ab} \mathbb{1}[\tilde{y}^t_{a:i} \geq \tilde{y}^t_{b:i}] - K_{ab} \mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{b:i}] \right) . \end{aligned} \tag{36}$$

Here, we used the symmetry of the kernel matrix K to obtain this result. Note that the second equation is a summation over $b \in \{1 \dots n\}$. This is true due to the identity $K_{aa} \mathbb{1}[\tilde{y}^t_{a:i} \geq \tilde{y}^t_{a:i}] - K_{aa} \mathbb{1}[\tilde{y}^t_{a:i} \leq \tilde{y}^t_{a:i}] = 0$ when $b = a$. This equation is exactly the conditional gradient provided in Eq. (12) in the main paper.

A.4. Optimal step size

We need to find the step size δ that gives the maximum decrease in the objective function g given the descent direction $\mathbf{s}^t$. This can be formulated as the following optimization problem:

$$\begin{aligned}\min_{\delta} \quad & \frac{\lambda}{2}\left\|A\boldsymbol{\alpha}^t + \delta\left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right) + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}\right\|^2 + \left\langle A\boldsymbol{\alpha}^t + \delta\left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right) + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}), \mathbf{y}^k\right\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ , \\ \text{s.t.} \quad & \delta \in [0,1] \ .\end{aligned} \tag{37}$$

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

$$\begin{aligned} 0 &= \lambda\left\langle \delta\left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right) + A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}, A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right\rangle + \left\langle \mathbf{y}^k, A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right\rangle \ , \\ \delta &= \frac{\left\langle A\boldsymbol{\alpha}^t - A\mathbf{s}^t, \lambda\left(A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\phi}\right) + \mathbf{y}^k\right\rangle}{\lambda\|A\boldsymbol{\alpha}^t - A\mathbf{s}^t\|^2} \ , \\ \delta &= \frac{\langle A\boldsymbol{\alpha}^t - A\mathbf{s}^t, \tilde{\mathbf{y}}^t\rangle}{\lambda\|A\boldsymbol{\alpha}^t - A\mathbf{s}^t\|^2} \ . \end{aligned} \tag{38}$$

In fact, if the optimal δ is out of the interval $[0,1]$, the value is simply truncated to be in $[0,1]$.

B. Fast conditional gradient computation

In this section, we give the technical details of the original filtering algorithm and then our modified filtering algorithm. To this end, we consider the following computation

$$\forall\, a \in \{1 \dots n\}, \qquad v'_a = \sum_b k(\mathbf{f}_a, \mathbf{f}_b)\, v_b\, \mathbb{1}[y_a \geq y_b] \ , \tag{39}$$

with $y_a, y_b \in [0,1]$ for all $a, b \in \{1 \dots n\}$. Note that the above equation is the same as Eq. (14) in the main paper, except for the multiplication by the scalar v_b. In Section 4 in the main paper, the value v_b was assumed to be 1, but here we consider the general case where $v_b \in \mathbb{R}$.

B.1. Original filtering algorithm

Let us first introduce some notations below. We denote the set of lattice points of the original permutohedral lattice with $\mathcal{P}$ and the neighbouring feature points of lattice point l by $N(l)$. This neighbourhood is shown in Fig. 5. Furthermore, we denote the neighbouring lattice points of a feature point a by $\bar{N}(a)$. In addition, the barycentric weight between the lattice point l and feature point b is denoted with w_{lb}. Furthermore, the value at feature point b is denoted by v_b and the value at lattice point l is denoted by $\bar{v}_l$. Finally, the set of feature point scores is denoted by $\mathcal{Y} = \{y_b \mid b \in \{1 \dots n\}\}$, their set of values is denoted by $\mathcal{V} = \{v_b \mid b \in \{1 \dots n\}\}$ and the set of lattice point values is denoted by $\bar{\mathcal{V}} = \{\bar{v}_l \mid l \in \mathcal{P}\}$. The pseudocode of the algorithm is given in Algorithm 2.

Algorithm 2 Original filtering algorithm [1]

```
Require: Permutohedral lattice $\mathcal{P}$, set of feature point values $\mathcal{V}$
  $\mathcal{V}' \leftarrow \mathbf{0} \quad \bar{\mathcal{V}} \leftarrow \mathbf{0} \quad \bar{\mathcal{V}}' \leftarrow \mathbf{0}$        ▷ Initialization
  for all $l \in \mathcal{P}$ do                                                              ▷ Splatting
      for all $b \in N(l)$ do
          $\bar{v}_l \leftarrow \bar{v}_l + w_{lb}\, v_b$
  $\bar{\mathcal{V}}' \leftarrow k \otimes \bar{\mathcal{V}}$                                  ▷ Blurring
  for all $a \in \{1 \dots n\}$ do                                                             ▷ Slicing
      for all $l \in \bar{N}(a)$ do
          $v'_a \leftarrow v'_a + w_{la}\, \bar{v}'_l$
```

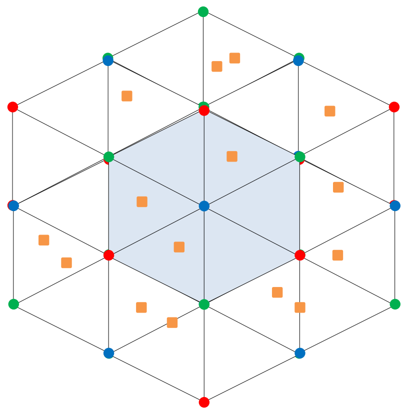


Figure 5: *A* 2*-dimensional hyperplane tessellated by the permutohedral lattice. The feature points are denoted with squares and the lattice points with circles. The neighborhood of the center lattice point is shaded and, for a feature point, the neighbouring lattice points are the vertices of the enclosing triangle.*

B.2. Modified filtering algorithm

As mentioned in the main paper, the interval $[0, 1]$ is discretized into H bins. Note that each bin $h \in \{0 \ldots H-1\}$ is associated with an interval which is identified as: $\left[\frac{h}{H-1}, \frac{h+1}{H-1}\right)$. Note that the last bin (with bin id $H-1$) is associated with the interval $[1, \cdot)$. Since $y_b \leq 1$, this bin contains the feature points whose scores are exactly 1. Given the score y_b of the feature point b, its bin/level can be identified as

$$h_b = \lfloor y_b * (H-1) \rfloor \ , \tag{40}$$

where $\lfloor \cdot \rfloor$ denotes the standard floor function.

Furthermore, during splatting, the values v_b are accumulated to the neighbouring lattice point only if the lattice point is above or equal to the feature point level. We denote the value at lattice point l at level h by $\bar{v}_{l:h}$. Formally, the barycentric interpolation at lattice point l at level h can be written as

$$\bar{v}_{l:h} = \sum_{\substack{b \in N(l) \\ h_b \leq h}} w_{lb} \, v_b \ . \tag{41}$$

Then, blurring is performed independently at each discrete level h. Finally, during slicing, the resulting values are interpolated at the feature point level. Our modified algorithm is given in Algorithm 3. In this algorithm, we denote the set of values corresponding to all the lattice points at level h as $\bar{\mathcal{V}}_h = \{v_{l:h} \mid l \in \mathcal{P}\}$.

Algorithm 3 Modified filtering algorithm

```
Require: Permutohedral lattice $\mathcal{P}$, set of feature point values $\mathcal{V}$, discrete levels $H$, set of scores $\mathcal{Y}$
  $\mathcal{V}' \leftarrow \mathbf{0} \quad \bar{\mathcal{V}} \leftarrow \mathbf{0} \quad \bar{\mathcal{V}}' \leftarrow \mathbf{0}$          ▷ Initialization
  for all $l \in \mathcal{P}$ do                                         ▷ Splatting
    for all $b \in N(l)$ do
      $h_b \leftarrow \lfloor y_b * (H-1) \rfloor$
      for all $h \in \{h_b \ldots H-1\}$ do            ▷ Splat at the feature point level and above
        $\bar{v}_{l:h} \leftarrow \bar{v}_{l:h} + w_{lb} \, v_b$
  for all $h \in \{0 \ldots H-1\}$ do $\bar{\mathcal{V}}'_h \leftarrow k \otimes \bar{\mathcal{V}}_h$     ▷ Blurring at each level independently
  for all $a \in \{1 \ldots n\}$ do                                     ▷ Slicing
    $h_a \leftarrow \lfloor y_a * (H-1) \rfloor$
    for all $l \in \bar{N}(a)$ do
      $v'_a \leftarrow v'_a + w_{la} \, \bar{v}'_{l:h_a}$                     ▷ Slice at the feature point level
```

Note that the above algorithm is given for the constraint $\mathbb{1}[y_a \geq y_b]$ (Eq. (14) in the main paper). However, it is fairly easy to modify it for the $\mathbb{1}[y_a \leq y_b]$ constraint. In particular, one needs to change the interval identified by the bin h to:

Dataset	Algorithm	$w^{(1)}$	σ_1	$w^{(2)}$	$\sigma_{2:s}$	$\sigma_{2:c}$
MSRC	MF	7.467846	1.000000	4.028773	35.865959	11.209644
	DC$_{\text{neg}}$	2.247081	3.535267	1.699011	31.232626	7.949970
Pascal	MF	100.000000	1.000000	74.877398	50.000000	5.454272
	DC$_{\text{neg}}$	0.500000	3.071772	0.960811	49.785678	1.000000

Table 3: *Parameters tuned for MF and DC$_{neg}$ on the MSRC and Pascal validation sets using Spearmint [5].*

$\left(\frac{h-1}{H-1}, \frac{h}{H-1}\right]$. Using this fact, one can easily derive the splatting and slicing equations for the $\mathbb{1}[y_a \leq y_b]$ constraint. The algorithm given above introduces an approximation to the gradient computation that depends on the number of discrete bins H. However, this approximation can be eliminated by using a dynamic data structure which we briefly explain in the next section.

B.2.1 Adaptive version of the modified filtering algorithm

Here, we briefly explain the adaptive version of our modified algorithm, which replaces the fixed discretization with a dynamic data structure. Effectively, discretization boils down to storing a vector of length H at each lattice point. Instead of such a fixed-length vector, one can use a dynamic data structure that grows with the number of different scores encountered at each lattice point in the splatting and blurring steps. In the worst case, *i.e.*, when all the neighbouring feature points have different scores, the maximum number of values to store at a lattice point is

$$H = \max_l |N^2(l)| \,, \tag{42}$$

where $N^2(l)$ denotes the union of neighbourhoods of the lattice point l and its neighbouring lattice points (the vertices of the shaded hexagon in Fig. 5). In our experiments, we observed that $|N^2(l)|$ is usually less than 100, with an average around 10. Empirically, however, we found this dynamic version to be slightly slower than the static one. We conjecture that this is due to the static version benefiting from better compiler optimization. Furthermore, both the versions obtained results with similar accuracy and therefore we used the static one for all our experiments.

C. Additional experiments

Let us first explain the pixel compatibility function used in the experiments. We then turn to additional experiments.

C.1. Pixel compatibility function used in the experiments

As mentioned in the main paper, our algorithm is applicable to any pixel compatibility function that is composed of a mixture of Gaussian kernels. In all our experiments, we used two kernels, namely spatial kernel and bilateral kernel, similar to [3, 4]. Our pixel compatibility function can be written as

$$K_{ab} = w^{(1)} \exp\left(-\frac{|\mathbf{p}_a - \mathbf{p}_b|^2}{\sigma_1}\right) + w^{(2)} \exp\left(-\frac{|\mathbf{p}_a - \mathbf{p}_b|^2}{\sigma_{2:s}} - \frac{|\mathbf{I}_a - \mathbf{I}_b|^2}{\sigma_{2:c}}\right) \,, \tag{43}$$

where $\mathbf{p}_a$ denotes the (x, y) position of pixel a measured from top left and $\mathbf{I}_a$ denotes the (r, g, b) values of pixel a. Note that there are 5 learnable parameters: $w^{(1)}, \sigma_1, w^{(2)}, \sigma_{2:s}, \sigma_{2:c}$. These parameters are cross validated for different algorithms on each data set. The final cross validated parameters for MF and DC$_{neg}$ are given in Table 3. To perform this cross-validation, we ran Spearmint for 2 days for each algorithm on both datasets. Note that, due to this time limitation, we were able to run approximately 1000 Spearmint iterations on MSRC but only 100 iterations on Pascal. This is due to bigger images and a larger validation set on the Pascal dataset. Hence, it resulted in less accurate energy parameters.

C.2. Additional segmentation results

In this section we provide additional segmentation results.

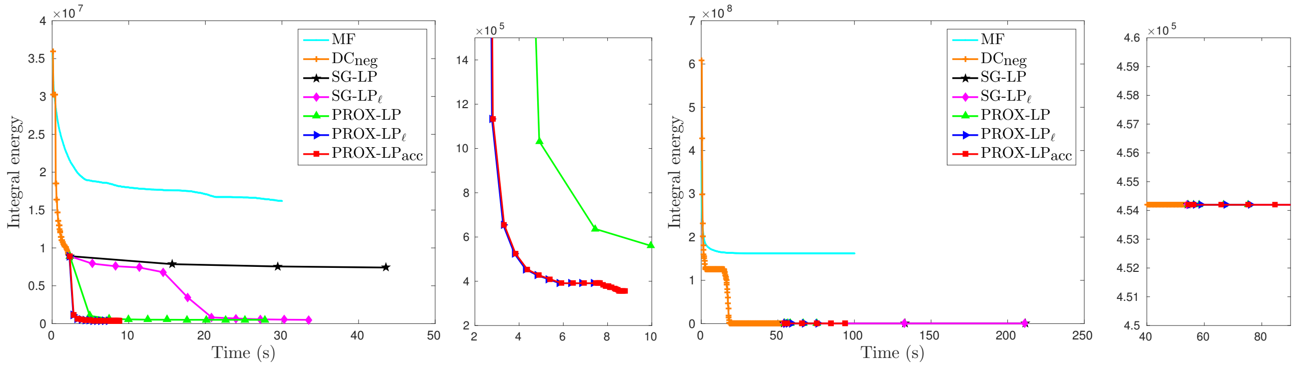


Figure 6: *Assignment energy as a function of time with the parameters tuned for MF for an image in (**left**) MSRC and (**right**) Pascal. A zoomed-in version is shown next to each plot. Except for MF, all the algorithms were initialized with DC$_{neg}$. For the MSRC image, PROX-LP clearly outperforms SG-LP$_\ell$ by obtaining much lower energies in fewer iterations, and the accelerated versions of our algorithm obtain roughly the same energy as PROX-LP but significantly faster. For the Pascal image, however, no LP algorithm is able to improve over DC$_{neg}$. Note that, in the Pascal dataset, for the MF parameters, DC$_{neg}$ ended up classifying all pixel in most images as background (which yields low energy values) and no LP algorithm is able to improve over it.*

C.2.1 Results on parameters tuned for MF

The results for the parameters tuned for MF on the MSRC and Pascal datasets are given in Table 4. In Fig. 6, we show the assignment energy as a function of time for an image in MSRC (the tree image in Fig. 7) and for an image in Pascal (the sheep image in Fig. 7). Furthermore, we provide some of the segmentation results in Fig. 7.

Interestingly, for the parameters tuned for MF, even though our algorithm obtains much lower energies, MF yields the best segmentation accuracy. In fact, one can argue that the parameters tuned for MF do not model the segmentation problem accurately, but were tuned such that the inaccurate MF inference yields good results. Note that, in the Pascal dataset, when tuned for MF, the Gaussian mixture coefficients are very high (see Table 3). In such a setting, DC$_{\text{neg}}$ ended up classifying all pixel in most images as background. In fact, SG-LP$_\ell$ was able to improve over DC$_{\text{neg}}$ in only 1% of the images, whereas all our versions improved over DC$_{\text{neg}}$ in roughly 25% of the images. Furthermore, our accelerated versions could not get any advantage over the standard version and resulted in similar run times. Note that, in most of the images, the *uncertain* pixels are in fact the entire image, as shown in Fig. 7.

C.2.2 More results on parameters tuned for DC$_{\text{neg}}$

In Fig. 8, we show assignment energy as a function of time for more images on MSRC for the parameters tuned for DC$_{\text{neg}}$. The same behaviour as in Fig. 2 in the main paper is observed. For this parameter setting, more qualitative results are shown in Fig. 9. Furthermore, some failure examples of PROX-LP$_{\text{acc}}$ on MSRC are shown in Fig. 10.

C.2.3 Summary

We have evaluated all the algorithms using two different parameter settings. Therefore, we summarize the best segmentation accuracy obtained by each algorithm and the corresponding parameter setting in Table 5. Note that, on MSRC, the best parameter setting for DC$_{\text{neg}}$ corresponds to the parameters tuned for MF. This is a strange result but can be explained by the fact that, as mentioned in the main paper, cross-validation was performed using the less accurate ground truth provided with the original dataset, but evaluation using the accurate ground truth annotations provided by [4].

Furthermore, in contrast to MSRC, the segmentation results of our algorithm on the Pascal dataset are not the state-of-the-art, even with the parameters tuned for DC$_{\text{neg}}$. This may be explained by the fact, that due to the limited cross-validation, the energy parameters obtained for the Pascal dataset are not accurate. Therefore, even though our algorithm obtained lower energies that was not reflected in the segmentation accuracy. Similar behaviour was observed in [3, 6].

		MF5	MF	DC$_{\text{neg}}$	SG-LP$_\ell$	PROX-LP	PROX-LP$_\ell$	PROX-LP$_{\text{acc}}$	Avg. E ($\times 10^4$)	Avg. T (s)	Acc.	IoU
MSRC	MF5	-	0	0	0	0	0	0	2366.6	**0.2**	81.14	54.60
	MF	95	-	18	15	2	1	2	1053.6	13.0	**83.86**	**59.75**
	DC$_{\text{neg}}$	95	77	-	0	0	0	0	812.7	2.8	83.50	59.67
	SG-LP$_\ell$	95	80	48	-	2	0	1	800.1	37.3	83.51	59.68
	PROX-LP	95	93	95	93	-	35	46	265.6	27.3	83.01	58.74
	PROX-LP$_\ell$	95	94	94	94	59	-	43	**261.2**	13.9	82.98	58.62
	PROX-LP$_{\text{acc}}$	95	93	93	93	49	46	-	295.9	7.9	83.03	58.97
Pascal	MF5	-	-	1	1	0	0	0	40779.8	**0.8**	80.42	28.66
	MF	93	-	3	3	0	0	1	20354.9	21.7	**80.95**	**28.86**
	DC$_{\text{neg}}$	93	87	-	0	0	0	0	2476.2	39.1	77.77	14.93
	SG-LP$_\ell$	93	87	1	-	0	0	0	2474.1	414.7	77.77	14.92
	PROX-LP	94	90	24	24	-	4	9	1475.6	81.0	78.04	15.79
	PROX-LP$_\ell$	94	90	24	24	5	-	9	**1458.9**	82.7	78.04	15.79
	PROX-LP$_{\text{acc}}$	94	89	28	27	18	18	-	1623.7	83.9	77.86	15.18

Table 4: *Results on the MSRC and Pascal datasets with the parameters tuned for MF. We show: the percentage of images where the row method strictly outperforms the column one on the final integral energy, the average integral energy over the test set, the average run time, the segmentation accuracy and the intersection over union score. Note that all versions of our algorithm obtain much lower energies than the baselines. However, as expected, lower energy does not correspond to better segmentation accuracy, mainly due to the less accurate energy parameters. Furthermore, the accelerated versions of our algorithm are similar in run time and obtain similar energies compared to PROX-LP.*

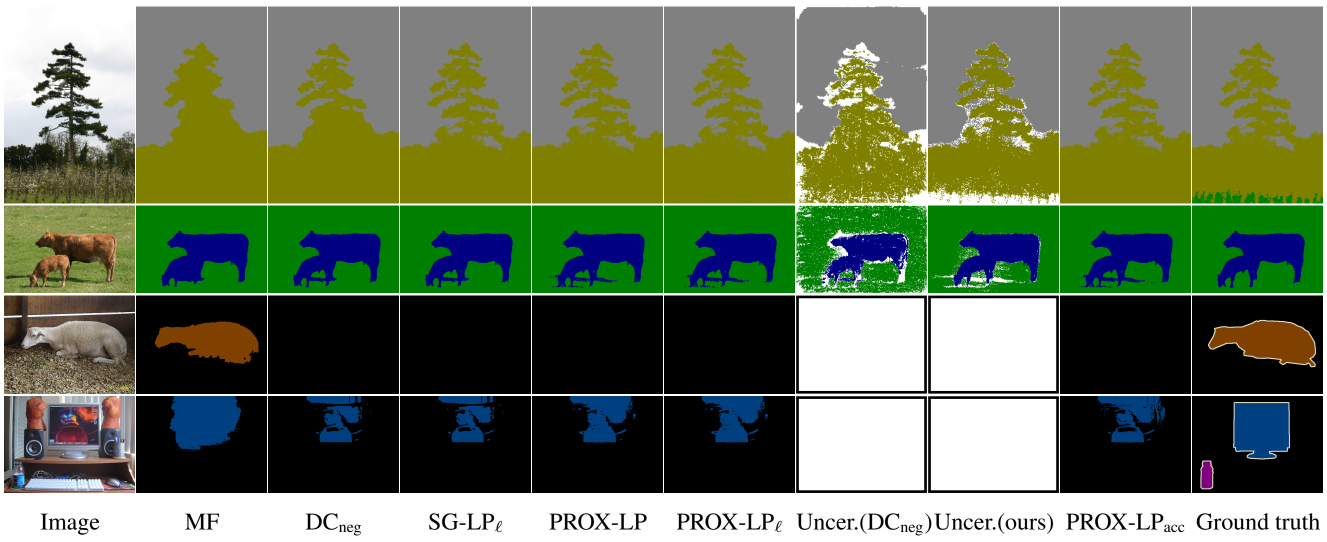


Figure 7: *Results with the parameters tuned for MF for images in (**top two rows**) MSRC and (**bottom two rows**) Pascal. The uncertain pixels identified by DC$_{neg}$ and PROX-LP$_{acc}$ are marked in white. Note that, in MSRC all versions of our algorithm obtain visually good segmentations similar to MF (or better). In Pascal, the segmentation results are poor except for MF, even though we obtain much lower energies. We argue that, in this case, the energy parameters do not model the segmentation problem accurately.*

C.3. Effect of proximal regularization constant

We plot the assignment energy as a function of time for an image in MSRC (the tree image in Fig. 9) by varying the proximal regularization constant λ. Here, we used the parameters tuned for DC$_{\text{neg}}$. The plot is shown in Fig. 11. In summary, for a wide range of λ, PROX-LP obtains similar energies with approximately the same run time.

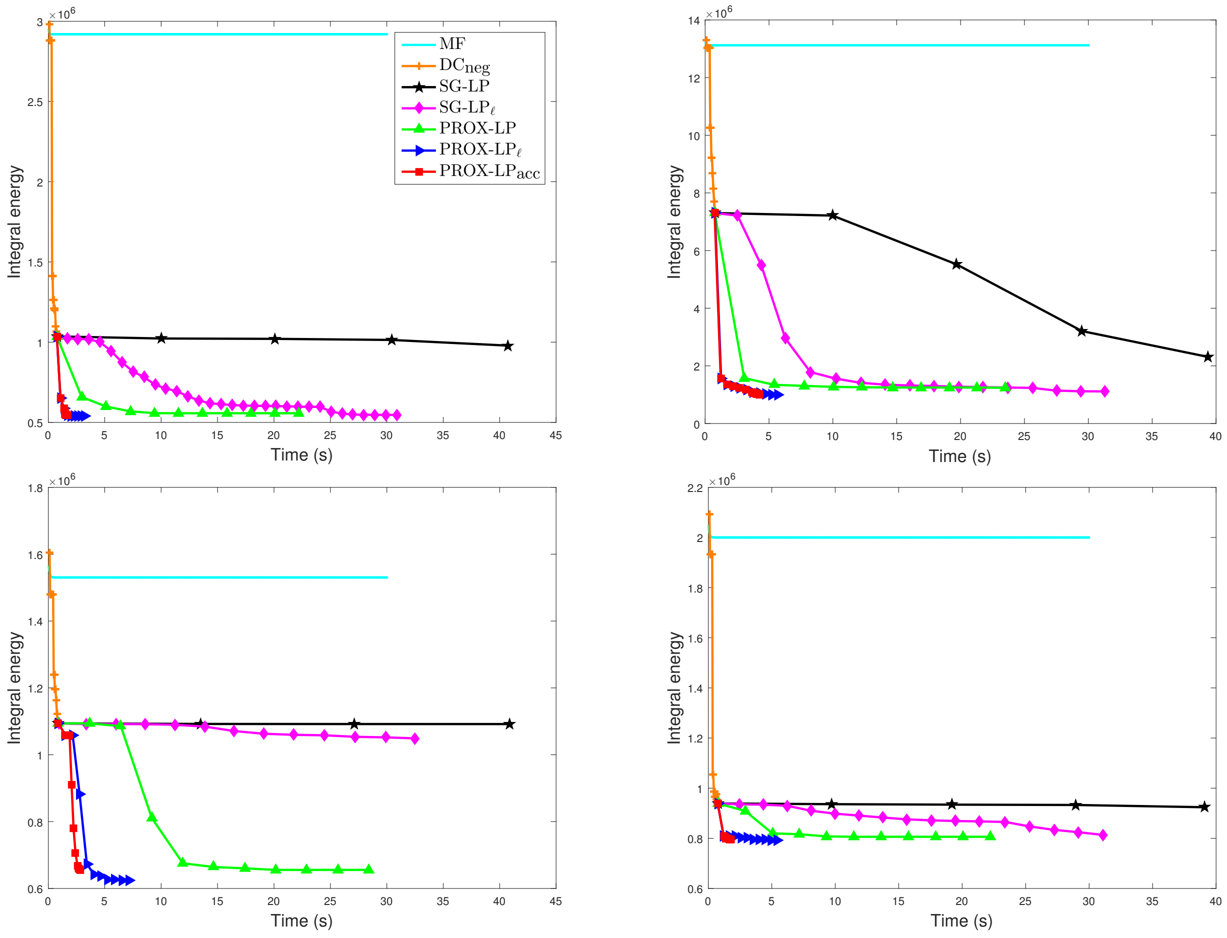


Figure 8: *Assignment energy as a function of time with the parameters tuned for DC$_{neg}$ for some images in (cow, building, person and car images shown in Fig. 9) MSRC. Note that PROX-LP clearly outperforms SG-LP$_\ell$ by obtaining much lower energies in fewer iterations. Furthermore, the accelerated versions of our algorithm obtain roughly the same energy as PROX-LP but significantly faster. In short the same behaviour as in Fig. 2 in the main paper is observed.*

Algorithm	MSRC			Pascal		
	Parameters	Avg. T (s)	Acc.	Parameters	Avg. T (s)	Acc.
MF5	MF	**0.2**	81.14	MF	**0.8**	80.42
MF	MF	13.0	83.86	MF	21.7	**80.95**
DC$_{neg}$	MF	2.8	83.50	DC$_{neg}$	3.7	80.43
SG-LP$_\ell$	MF	37.3	83.51	DC$_{neg}$	84.4	80.49
PROX-LP	DC$_{neg}$	23.5	83.99	DC$_{neg}$	106.7	80.63
PROX-LP$_\ell$	DC$_{neg}$	6.3	83.94	DC$_{neg}$	22.1	80.65
PROX-LP$_{acc}$	DC$_{neg}$	3.7	**84.16**	DC$_{neg}$	14.7	80.58

Table 5: *Best segmentation results of each algorithm with their respective parameters, the average time on the test set and the segmentation accuracy. In MSRC, the best segmentation accuracy is obtained by PROX-LP$_{acc}$ and in Pascal it is by MF. Note that, on MSRC, the best parameter setting for DC$_{neg}$ corresponds to the parameters tuned for MF. This is due to the fact that cross-validation was performed on the less accurate ground truth but evaluation on the accurate ground truth annotations provided by [4]. Furthermore, the low segmentation performance of our algorithm on the Pascal dataset is may be due to less accurate energy parameters resulted from limited cross-validation.*

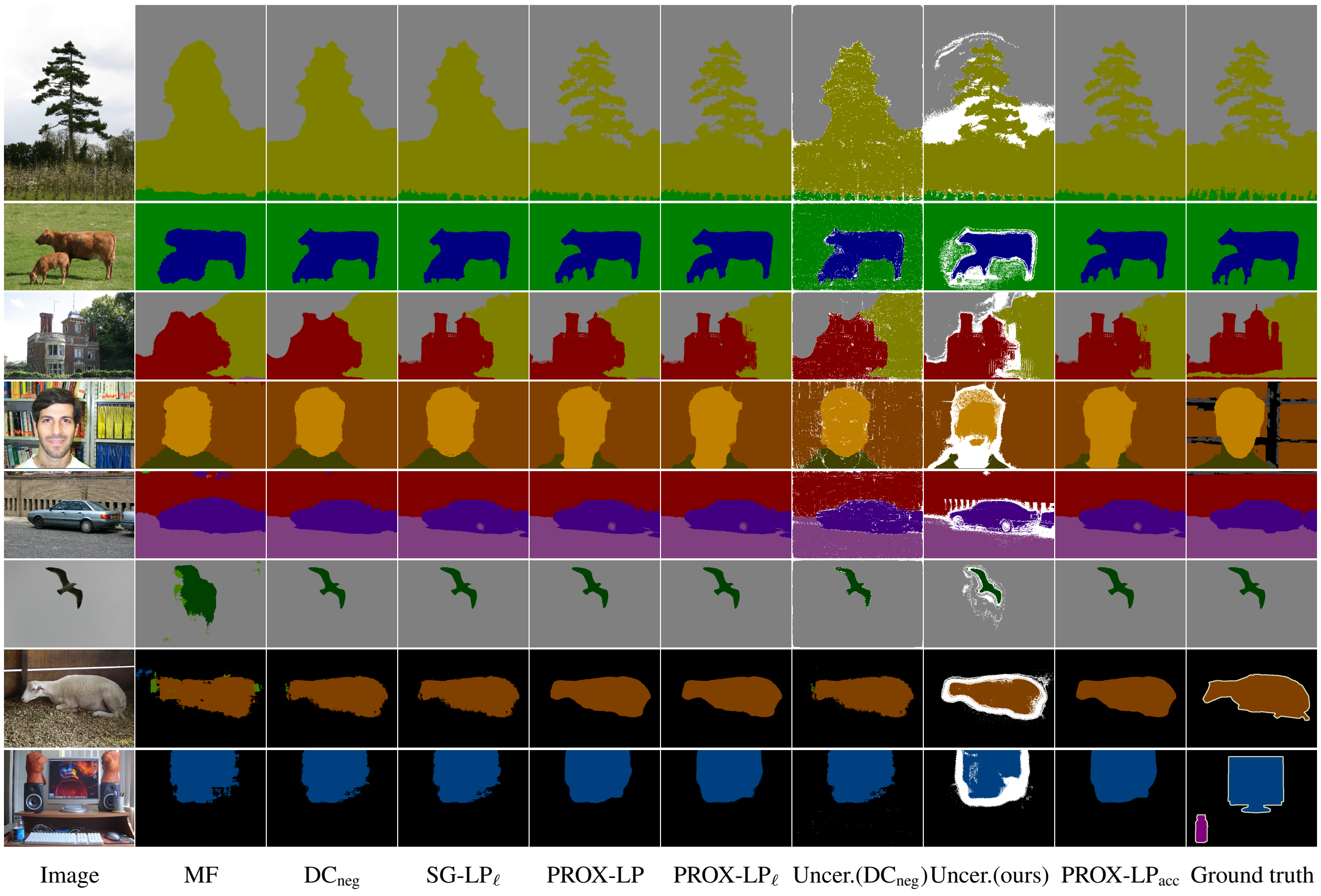


Figure 9: *Results with the parameters tuned for DC$_{neg}$ for images in (**top six rows**) MSRC and (**bottom two rows**) Pascal. The uncertain pixels identified by DC$_{neg}$ and PROX-LP$_{acc}$ are marked in white. Note that all versions of our algorithm obtain visually good segmentations. In addition, even though DC$_{neg}$ is less accurate (the percentatge of uncertain pixels for DC$_{neg}$ is usually less than 1%) in predicting uncertain pixels, our algorithm marks most of the crucial pixels (object boundaries and shadows) as uncertain. Furthermore, in the MSRC images, the improvement of PROX-LP$_{acc}$ over the baselines is clearly visible and the final segmentation is virtually the same as the accurate ground truth.*

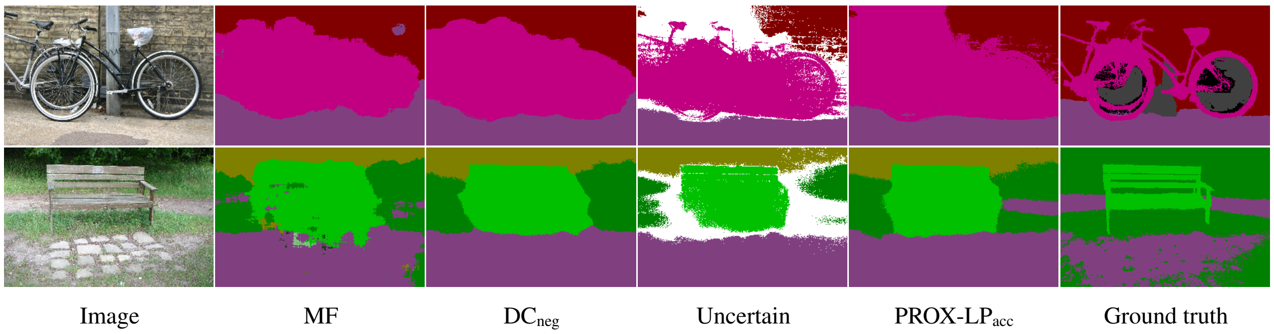


Figure 10: *Failure examples for PROX-LP$_{acc}$ with the parameters tuned for DC$_{neg}$ in MSRC.*

C.4. Modified filtering algorithm

We compare our modified filtering method, described in Section 4, with the divide-and-conquer strategy of [3]. To this end, we evaluated both algorithms on one of the Pascal VOC test images (the sheep image in Fig. 9), but varying the image

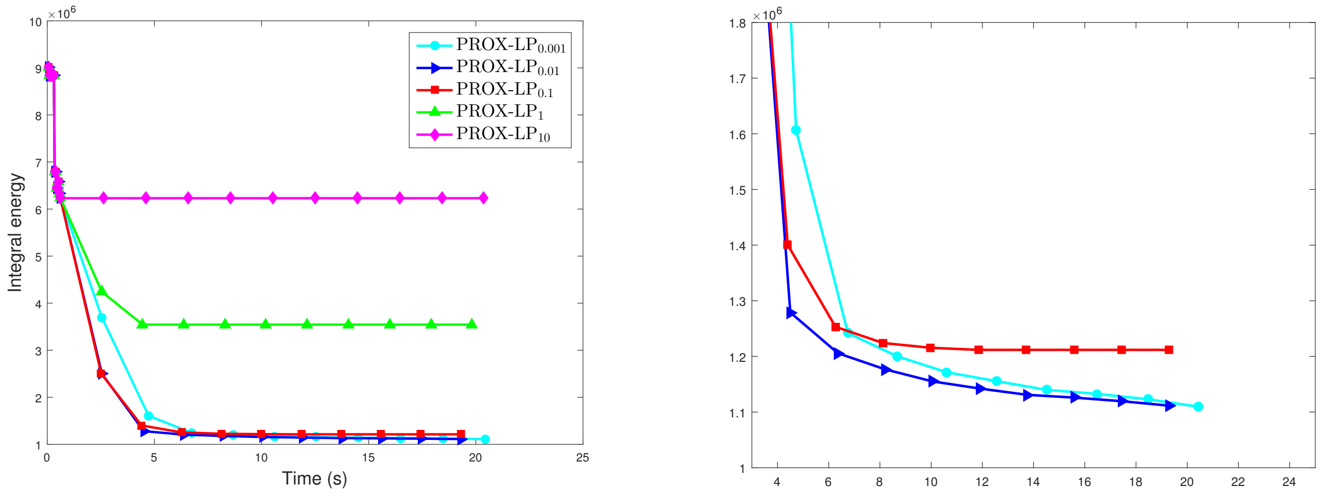


Figure 11: *Assignment energy as a function of time for an image in MSRC, for different values of λ. The zoomed plot is shown on the right. Note that, for $\lambda = 0.1, 0.01, 0.001$, PROX-LP obtains similar energies in approximately the same amount of time.*

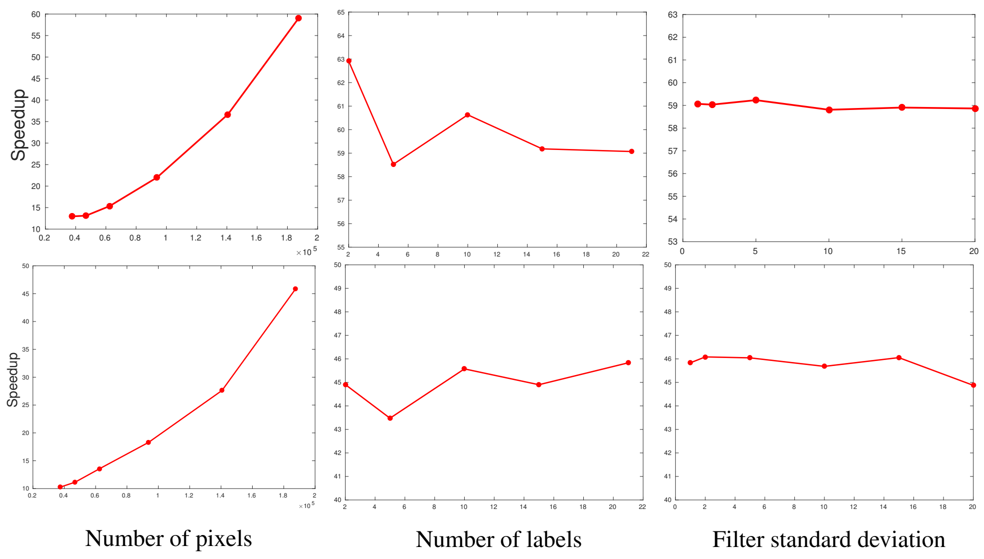
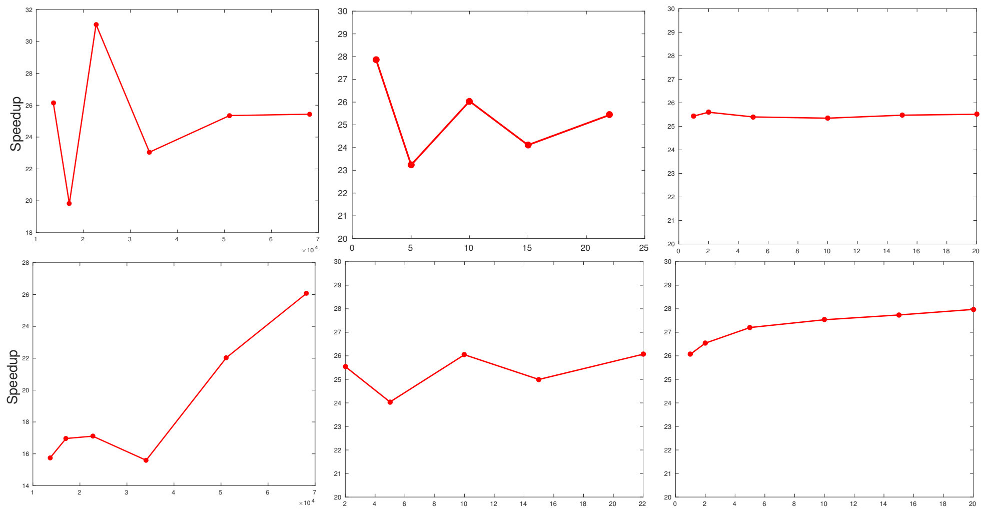


Figure 12: *Speedup of our modified filtering algorithm over the divide-and-conquer strategy of [3] on a Pascal image,* ***top:*** *spatial kernel (d = 2),* ***bottom:*** *bilateral kernel (d = 5). Note that our speedup grows with the number of pixels and is approximately constant with respect to the number of labels and filter standard deviation.*

size, the number of labels and the Gaussian kernel standard deviation. The respective plots are shown in Fig. 12. Note that, as claimed in the main paper, speedup with respect to the standard deviation is roughly constant. Similar plots for an MSRC image (the tree image in Fig. 9) are shown in Fig. 13. In this case, speedup is around $15 - 32$, with around $23 - 32$ in the operating region of all versions of our algorithm.



Number of pixels   Number of labels   Filter standard deviation

Figure 13: *Speedup of our modified filtering algorithm over the divide-and-conquer strategy of [3] on a MSRC image,* ***top:*** *spatial kernel (d = 2),* ***bottom:*** *bilateral kernel (d = 5). Note that our speedup grows with the number of pixels and is approximately constant with respect to the number of labels and filter standard deviation.*

References

[1] A. Adams, J. Baek, and M. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 2010. 5

[2] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2009. 1

[3] A. Desmaison, R. Bunel, P. Kohli, P. Torr, and P. Kumar. Efficient continuous relaxations for dense CRF. *ECCV*, 2016. 7, 8, 11, 12, 13

[4] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. *NIPS*, 2011. 7, 8, 10

[5] J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. *NIPS*, 2012. 7

[6] P. Wang, C. Shen, and A. van den Hengel. Efficient SDP inference for fully-connected CRFs based on low-rank decomposition. *CVPR*, 2015. 8

[7] X. Xiao and D. Chen. Multiplicative iteration for nonnegative quadratic programming. *Numerical Linear Algebra with Applications*, 2014. 3, 4