A MRF Shape Prior for Facade Parsing with Occlusions Supplementary Material

Mateusz Koziński, Raghudeep Gadde, Sergey Zagoruyko, Guillaume Obozinski and Renaud Marlet
Université Paris-Est, LIGM (UMR CNRS 8049), ENPC, F-77455 Marne-la-Vallée

1. Overview

In this document we present:

- Proof of equivalence of the grid pattern prior to the adjacency pattern prior;
- Derivation of the optimization algorithm from section 3;
- Confusion matrices corresponding to the results of experiments presented in the paper.

2. An adjacency pattern equivalent to a given grid pattern

As stated in section 2.1 of the paper, for any grid pattern shape prior, $\mathcal{G} = (\mathcal{C}, \mathcal{R}, \mathcal{H}, \mathcal{V})$ there exists an adjacency pattern $A^{\mathcal{G}} = (S^{\mathcal{G}}, V^{\mathcal{G}}, H^{\mathcal{G}})$ encoding the same set of shapes. The set of pixel classes of the adjacency pattern is $S^{\mathcal{G}} = \mathcal{R} \times \mathcal{C}$. We denote the row-class component of a pixel class $s = (r_s, c_s)$ by $r(s) = r_s$ and its column-class component by $c(s) = c_s$. The sets of allowed classes of adjacent pixels are defined as follows (equation 1 of the main paper):

$$V^{\mathcal{G}} = \big\{(s_1, s_2) | c(s_1) = c(s_2) \wedge \big(r(s_1), r(s_2)\big) \in \mathcal{V}\big\} \quad \text{and} \tag{1a}$$

$$H^{\mathcal{G}} = \big\{(s_1, s_2) | r(s_1) = r(s_2) \wedge \big(c(s_1), c(s_2)\big) \in \mathcal{H}\big\} \, . \tag{1b}$$

We state the equivalence of the two priors formally below.

Lemma 1. *Consider a shape prior encoded as a grid pattern $\mathcal{G} = (\mathcal{C}, \mathcal{R}, \mathcal{H}, \mathcal{V})$, as defined in section 1 of the paper. An adjacency pattern $A^{\mathcal{G}} = (S^{\mathcal{G}}, V^{\mathcal{G}}, H^{\mathcal{G}})$, where $S^{\mathcal{G}} = \mathcal{C} \times \mathcal{R}$ and $V^{\mathcal{G}}$ and $H^{\mathcal{G}}$ are defined according to* (1)*, encodes a prior that is equivalent to the grid pattern. That is, the set of image labelings that conform to the grid pattern $\mathcal{G}$ is the same as the set of labelings that conform to the adjacency pattern $A^{\mathcal{G}}$.*

Proof. First we prove that if a segmentation is consistent with the grid pattern it is also consistent with the adjacency pattern. We need to show that, in a segmentation consistent with the grid pattern, classes of pairs of vertically and horizontally adjacent pixels belong to $V^{\mathcal{G}}$ and $H^{\mathcal{G}}$, respectively. This follows directly from (1).

Next, we show that if a segmentation is not consistent with the grid pattern, then it is necessarily not consistent with the adjacency pattern. There are two possible ways in which a segmentation can be inconsistent with a grid pattern: (1) a row of the segmentation contains two pixels labeled (r, c) and (r', c) where $r \neq r'$, or (2) row classes r and r' are assigned to neighboring rows, while $(r, r') \notin \mathcal{V}$. Analogical conditions can be presented for columns. Condition (1) implies that there exists a sequence of horizontally adjacent pixels, labeled $s_1, \dots s_n$, where $s_1 = (r, c)$ and $s_n = (r', c)$. In consequence there is at least one pair of horizontally adjacent pixels in the sequence that receive classes $s_a = (r_a, c_a)$ and $s_b = (r_b, c_b)$, such that $r_a \neq r_b$ and, according to (1), $(s_a, s_b) \notin H^{\mathcal{G}}$. Therefore the segmentation is not consistent with the adjacency pattern. Condition (2) implies that there exists a pair of vertically neighboring pixels, labeled $s_1 = (r_1, c_1)$ and $s_2 = (r_2, c_2)$ such that $(r_1, r_2) \notin \mathcal{V}$, and therefore $(s_1, s_2) \notin V^{\mathcal{G}}$ according to (1) and the segmentation is inconsistent with the adjacency pattern. □

3. Derivation of the inference algorithm

For readability we provide a quick reference of the used notation and reproduce the problem defined in the paper. Then we derive our algorithm, following the principles of dual decomposition [2, 1].

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

3.1. Notation

- $\mathcal{I}$ the image; a set of pairs (i, j) where i is the row and j is the column index;
- $\mathcal{I}_v$, $\mathcal{I}_h$ the image without the last row, or column, respectively; such image contains only pixels that have a neighbor below or to the right in the original image;
- S^o the set of classes assigned to image pixels
- $\phi_{ij\Psi^o(\sigma)}$ the cost of assigning class σ to pixel (i, j); many classes σ have the same cost since they represent elements of the same semantic class; function $\Psi^o(\sigma)$ maps a pixel class σ to the corresponding semantic class;
- $\theta_{\sigma\sigma'}$ a pairwise potential; σ and σ' represent classes assigned to neighboring pixels;
- V^o, H^o sets of pairs of classes that can be assigned to vertically and horizontally neighboring pixels, respectively;
- $z_{ij\sigma}$ a variable encoding assignment of class σ to pixel (i, j);
- $\mathbf{z}$ a vector of all $z_{ij\sigma}$ for $\forall\sigma \in S^o, \forall(i, j) \in \mathcal{I}$;
- $u_{ij\sigma\sigma'}$ a variable encoding assignment of class σ to pixel (i, j) and class σ' to the pixel immediately above it.
- $\mathbf{u}$ a vector of all $u_{ij\sigma\sigma'}$ for $\forall\sigma, \sigma' \in S^o, \forall(i, j) \in \mathcal{I}_h$;
- $v_{ij\sigma\sigma'}$ a variable encoding assignment of class σ to pixel (i, j) and class σ' to the pixel immediately to the right.
- $\mathbf{v}$ a vector of all $v_{ij\sigma\sigma'}$ for $\forall\sigma, \sigma' \in S^o, \forall(i, j) \in \mathcal{I}_v$;

3.2. The problem

For readability we reproduce here the problem defined in the paper:

$$\min_{\mathbf{z},\mathbf{u},\mathbf{v}} \sum_{\substack{(i,j)\in\mathcal{I}\\ \sigma\in S^o}} \phi_{ij\Psi^o(\sigma)} z_{ij\sigma} + \sum_{\substack{(i,j)\in\mathcal{I}_v\\ \sigma,\sigma'\in S^o}} \theta_{\sigma\sigma'}\, v_{ij\sigma\sigma'} + \sum_{\substack{(i,j)\in\mathcal{I}_h\\ \sigma,\sigma'\in S^o}} \theta_{\sigma\sigma'}\, u_{ij\sigma\sigma'}, \tag{2}$$

where $z_{ij\sigma}, u_{ij\sigma\sigma'}, v_{ij\sigma\sigma'} \in \{0, 1\}$, subject to

$$\forall(i, j) \in \mathcal{I}, \quad \sum_{\sigma\in S^o} z_{ij\sigma} = 1, \tag{3}$$

$$\forall(i, j) \in \mathcal{I}_v, \forall\sigma \in S^o \quad \sum_{\sigma'\in S^o} v_{ij\sigma\sigma'} = z_{ij\sigma}, \qquad \forall(i, j) \in \mathcal{I}_v, \forall\sigma \in S^o \quad \sum_{\sigma'\in S^o} v_{ij\sigma'\sigma} = z_{i+1j\sigma}, \tag{4a}$$

$$\forall(i, j) \in \mathcal{I}_h, \forall\sigma \in S^o \quad \sum_{\sigma'\in S^o} u_{ij\sigma\sigma'} = z_{ij\sigma}, \qquad \forall(i, j) \in \mathcal{I}_h, \forall\sigma \in S^o \quad \sum_{\sigma'\in S^o} u_{ij\sigma'\sigma} = z_{ij+1\sigma}, \tag{4b}$$

and

$$\forall(i, j) \in \mathcal{I}_v, \forall(\sigma, \sigma') \notin V^o, \quad v_{ij\sigma\sigma'} = 0\,, \tag{5a}$$

$$\forall(i, j) \in \mathcal{I}_h, \forall(\sigma, \sigma') \notin H^o, \quad u_{ij\sigma\sigma'} = 0\,. \tag{5b}$$

3.3. Derivation of the inference algorithm

To solve problem (2-5) we adopt the dual decomposition approach [1, 2]. We relax the constraints on binary domain of variables $z_{ij\sigma}$, $u_{ij\sigma\sigma'}$ and $v_{ij\sigma\sigma'}$, obtaining a large linear program. To solve it efficiently, we transform the original objective to a sum of easy to optimize functions and formulate a dual problem that can be solved by iteratively solving the simple objectives. We adopt the most standard decomposition of a 4-connected grid into Markov chains over image rows and columns. The resulting subproblems can be solved independently and efficiently using the Viterbi algorithm. From the

solution to the dual problem, we then extract a primal, binary solution that satisfies the hard constraints. The algorithm is a standard application of dual decomposition to a MRF over a grid of pixels, the details of which can be found in [1, 2].

First we introduce two copies of variable $z_{ij\sigma}$, denoted $z'_{ij\sigma}$ and $z''_{ij\sigma}$. We denote the vector of $z'_{ij\sigma}$ for all $(i,j) \in \mathcal{I}$, $\sigma \in S^o$ by $\mathbf{z}'$ and the vector of all $z''_{ij\sigma}$ by $\mathbf{z}''$. We let the new variables, as well as $u_{ij\sigma\sigma'}$ and $v_{ij\sigma\sigma'}$, vary continuously in the interval $[0,1]$. The objective (2) can be rewritten as

$$\min_{\mathbf{z}',\mathbf{z}'',\mathbf{u},\mathbf{v}} \sum_{(i,j)\in\mathcal{I}} \sum_{\sigma\in S^o} \frac{1}{2}\phi_{ij\Psi^o(\sigma)}(z'_{ij\sigma} + z''_{ij\sigma}) + \sum_{(i,j)\in\mathcal{I}_v} \sum_{\sigma,\sigma'\in S^o} \theta_{\sigma\sigma'} v_{ij\sigma\sigma'} + \sum_{(i,j)\in\mathcal{I}_h} \sum_{\sigma,\sigma'\in S^o} \theta_{\sigma\sigma'} u_{ij\sigma\sigma'} \,, \tag{6}$$

subject to constraints on positivity of the continuous variables

$$\forall (i,j) \in \mathcal{I}, \quad \forall \sigma \in S^o, \quad z'_{ij\sigma} \geq 0, \tag{7a}$$
$$\forall (i,j) \in \mathcal{I}, \quad \forall \sigma \in S^o, \quad z''_{ij\sigma} \geq 0, \tag{7b}$$
$$\forall (i,j) \in \mathcal{I}_v, \forall \sigma,\sigma' \in S^o, \quad v_{ij\sigma\sigma'} \geq 0, \tag{7c}$$
$$\forall (i,j) \in \mathcal{I}_h, \forall \sigma,\sigma' \in S^o, \quad u_{ij\sigma\sigma'} \geq 0, \tag{7d}$$

the simplex constraints (3) reformulated on the new variables

$$\forall (i,j) \in \mathcal{I}, \quad \sum_{\sigma\in S^o} z'_{ij\sigma} = 1 \,, \tag{8a}$$

$$\forall (i,j) \in \mathcal{I}, \quad \sum_{\sigma\in S^o} z''_{ij\sigma} = 1 \,, \tag{8b}$$

and coupling constraints for $z'_{ij\sigma}$ and $z''_{ij\sigma}$,

$$\forall (i,j) \in \mathcal{I}, \sigma \in S^o, \quad z'_{ij\sigma} = z''_{ij\sigma} \,, \tag{9}$$

and a reformulation of constraints (4 and 5) on $z'_{ij\sigma}$ and $z''_{ij\sigma}$ that we present below. Given equation (9), each of constraints (4a, 4b, 5a and 5b) can be defined on just one of the new variables, that is, either on $z'_{ij\sigma}$, or on $z''_{ij\sigma}$. To obtain a formulation which decomposes into horizontal and vertical chains, we require $z'_{ij\sigma}$ to satisfy the constraints on classes of vertical neighbors, and require $z''_{ij\sigma}$ to satisfy the horizontal constraints

$$\forall (i,j) \in \mathcal{I}_v, \ \forall \sigma \in S^o, \qquad \sum_{\sigma'\in S^o} v_{ij\sigma\sigma'} = z'_{ij\sigma}, \qquad \sum_{\sigma'\in S^o} v_{ij\sigma'\sigma} = z'_{i+1\,j\sigma}, \tag{10a}$$
$$\forall (i,j) \in \mathcal{I}_h, \ \forall \sigma \in S^o, \qquad \sum_{\sigma'\in S^o} u_{ij\sigma\sigma'} = z''_{ij\sigma}, \qquad \sum_{\sigma'\in S^o} u_{ij\sigma'\sigma} = z''_{ij+1\sigma}, \tag{10b}$$

$$\forall (i,j) \in \mathcal{I}_v, \forall (\sigma,\sigma') \notin V^o, \quad v_{ij\sigma\sigma'} = 0 \,, \tag{11a}$$
$$\forall (i,j) \in \mathcal{I}_h, \forall (\sigma,\sigma') \notin H^o, \quad u_{ij\sigma\sigma'} = 0 \,. \tag{11b}$$

It is obvious that when constraint (9) is satisfied objective (6) is equivalent to the original objective (2).

We define the feasible set C' as the set of vectors $(\mathbf{z},\mathbf{v})$ that satisfy the vertical constraints (7a, 7c, 8a, 10a, 11a) and the set C'' of vectors $(\mathbf{z},\mathbf{u})$ satisfying the horizontal constraints (7b, 7d, 8b, 10b, 11b). We construct a Lagrangian for problem (6) with respect to constraint (9)

$$\begin{aligned} L_D(\boldsymbol{\lambda}) = \min_{\substack{(\mathbf{z}',\mathbf{v})\in C' \\ (\mathbf{z}'',\mathbf{u})\in C''}} & \sum_{(i,j)\in\mathcal{I}} \sum_{\sigma\in S^o} \frac{1}{2}\phi_{ij\Psi^o(\sigma)}(z'_{ij\sigma} + z''_{ij\sigma}) + \\ & \sum_{(i,j)\in\mathcal{I}_v} \sum_{\sigma,\sigma'\in S^o} \theta_{\sigma,\sigma'} v_{ij\sigma\sigma'} + \sum_{(i,j)\in\mathcal{I}_h} \sum_{\sigma,\sigma'\in S^o} \theta_{\sigma,\sigma'} u_{ij\sigma\sigma'} + \\ & \sum_{(i,j)\in\mathcal{I}} \sum_{\sigma\in S^o} \lambda_{ij\sigma}(z'_{ij\sigma} - z''_{ij\sigma}) \,. \end{aligned} \tag{12}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers $\lambda_{ij\sigma}$ for all $(i,j) \in \mathcal{I}$ and all $\sigma \in S^o$.

We formulate a dual problem

$$\max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) \ . \tag{13}$$

We solve the dual using a subgradient ascent scheme. In each iteration, we calculate the subgradient of the objective and update $\boldsymbol{\lambda}$ by making a step in the direction of the subgradient. We use a fixed sequence of decaying stepsizes. It can be shown (see [1] for details) that the subdifferential of the objective contains the vector $\tilde{\mathbf{z}} = \hat{\mathbf{z}}' - \hat{\mathbf{z}}''$, where the pair $(\hat{\mathbf{z}}', \hat{\mathbf{z}}'')$ minimizes (12). However, the variables are not coupled by any constraints and so we have

$$\hat{\mathbf{z}}' = \operatorname*{argmin}_{(\mathbf{z}',\mathbf{v})\in C'} \sum_{(i,j)\in\mathcal{I}} \sum_{\sigma\in S^o} (\frac{1}{2}\phi_{ij\Psi^o(\sigma)} + \lambda_{ij\sigma}) z'_{ij\sigma} + \sum_{(i,j)\in\mathcal{I}_v} \sum_{\sigma,\sigma'\in S^o} \theta^o(\sigma,\sigma') v_{ij\sigma\sigma'} \tag{14a}$$

and

$$\hat{\mathbf{z}}'' = \operatorname*{argmin}_{(\mathbf{z}'',\mathbf{v})\in C''} \sum_{(i,j)\in\mathcal{I}} \sum_{\sigma\in S^o} (\frac{1}{2}\phi_{ij\Psi^o(\sigma)} - \lambda_{ij\sigma}) z''_{ij\sigma} + \sum_{(i,j)\in\mathcal{I}_h} \sum_{\sigma,\sigma'\in S^o} \theta^o(\sigma,\sigma') u_{ij\sigma\sigma'} \ . \tag{14b}$$

The minimizations (14) are equivalent to finding the most likely configurations of independent Markov chains over image rows and columns, with hard constraints on neighboring labels, and can be solved by running the Viterbi algorithm independently on each row and column.

3.4. The inference algorithm

The resulting algorithm is presented in algorithm (1), where superscript n denotes the iteration number, α^n is the stepsize in n-th iteration, $\mathbf{z}_{\cdot j}$, $\boldsymbol{\phi}_{\cdot j}$ and $\boldsymbol{\lambda}_{\cdot j}$ are vectors of variables and costs corresponding to image column j and $\mathbf{z}_{i\cdot}$, $\boldsymbol{\phi}_{i\cdot}$ and $\boldsymbol{\lambda}_{i\cdot}$ are the vectors corresponding to row i. By $\mathring{z}_{ij\sigma}$ we denote the number of times class σ has been assigned to pixel (i,j) during the operation of the algorithm. We denote the vector of all $\mathring{z}_{ij\sigma}$ by $\mathring{\mathbf{z}}$.

Algorithm 1 Dual Decomposition on a 4-connected grid with slaves solving Markov Chains

$\boldsymbol{\lambda}^0 \leftarrow 0$
$n \leftarrow 1$
$\mathring{\mathbf{z}} \leftarrow 0$
while not converged **do**
  $\forall j \in J \quad \hat{\mathbf{z}}'^n_{\cdot j} \leftarrow \textsc{Viterbi}(\frac{1}{2}\boldsymbol{\phi}_{\cdot j} + \boldsymbol{\lambda}^{n-1}_{\cdot j}, V)$
  $\forall i \in I \quad \hat{\mathbf{z}}''^n_{i\cdot} \leftarrow \textsc{Viterbi}(\frac{1}{2}\boldsymbol{\phi}_{i\cdot} - \boldsymbol{\lambda}^{n-1}_{i\cdot}, H)$
  $\boldsymbol{\lambda}^n \leftarrow \boldsymbol{\lambda}^{n-1} + \frac{\alpha^n}{2}(\hat{\mathbf{z}}'^n - \hat{\mathbf{z}}''^n)$
  $\mathring{\mathbf{z}} \leftarrow \mathring{\mathbf{z}} + \hat{\mathbf{z}}'^n + \hat{\mathbf{z}}''^n$
  $n \leftarrow n + 1$
end while
$\hat{\mathbf{z}} \leftarrow \textsc{GetFinalZ}(\mathring{\mathbf{z}}, V, H)$

After a number of iterations, the variables $\mathbf{z}'$ and $\mathbf{z}''$ agree on the labels for most pixels. Heuristics can then be used to extract labels for the pixels for which $\mathbf{z}'$ and $\mathbf{z}''$ disagree [1]. In algorithm 1, we denote this procedure by GETFINALZ. One possible method is to count how many times a pixel has been assigned each of the possible labels, by accumulating the label frequency $\mathring{\mathbf{z}}$ over a number of iterations, and selecting for each pixel the most frequent label

$$\forall (i,j) \in \mathcal{I},\ \hat{z}_{ij\sigma} = \begin{cases} 1 & \text{if} \quad \sigma = \arg\max_{\sigma'} \mathring{z}_{ij\sigma'} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

However, a solution obtained in this manner would not necessarily be consistent with the constraints on classes of adjacent pixels. To extract a solution consistent with the constraints, we modify this heuristic. We present the concept in algorithm 2. For a randomly drawn row (or column) index $\hat{i}$ we extract its labeling using the Viterbi algorithm with costs corresponding to label frequencies. In algorithm 2, this subprocedure is represented as VITERBIMAX. It finds the labeling of the row that maximizes the total frequency of the labels assigned to pixels in the row, but only assigns to pairs of horizontally neighboring pixels pairs of labels that belong to H. Then we repeat the procedure for the neighboring row. In algorithm 2, this is

represented as VITERBIMAXWITHCONSTRAINTS, which uses the sets of allowed pairs of vertically adjacent classes V and the labels of the previous row to constrain the sets of labels that can be assigned to each pixel in the next row. The process is repeated for consecutive rows until a complete labeling is extracted. We sample a number of labelings initializing the procedure on randomly selected rows and columns, and keep the one with the lowest energy as the final solution.

Algorithm 2 Heuristics used to extract a labeling that satisfies hard constraints on classes of adjacent pixels. The solution is initialized with row $\hat{i}$ of the image. (The same operation can be performed for image columns, given an initial column index $\hat{j}$.)

$\hat{\mathbf{z}}_{\hat{i}\cdot} \leftarrow$ VITERBIMAX$(\mathring{\mathbf{z}}_{\hat{i}\cdot}, H)$
for $i \in \{\hat{i}+1, \dots h\}$ **do**
  $\hat{\mathbf{z}}_{i\cdot} \leftarrow$ VITERBIMAXWITHCONSTRAINTS$(\mathring{\mathbf{z}}_{i\cdot}, H, \hat{\mathbf{z}}_{i-1\cdot}, V)$
end for
for $i \in \{\hat{i}-1, \dots, 1\}$ **do**
  $\hat{\mathbf{z}}_{i\cdot} \leftarrow$ VITERBIMAXWITHCONSTRAINTS$(\mathring{\mathbf{z}}_{i\cdot}, H, \hat{\mathbf{z}}_{i+1\cdot}, V)$
end for

We remark on the theoretical possibility of designing a prior for which algorithm 2 could fail to find a labeling consistent with the constraints on classes of adjacent pixels. Such a situation could emerge if there exists a labeling of all pixels in a row (or column), for which the neighboring row (resp. column) can not be labeled in such a way that the pairs of vertically and horizontally neighboring pixel labels belong to V and H. Since row and column labelings consistent with a prior are defined in terms of constraints on classes of pairs of adjacent pixels, the formal condition for a prior for which algorithm 2 is guaranteed to find a labeling consistent with the constraints can be formulated in terms of pairs of allowed classes of adjacent pixels

$$\forall(\sigma_1, \sigma_2) \in V^o, \exists(\sigma_1', \sigma_2') \in V^o, \text{ s.t. } (\sigma_1, \sigma_1') \in H^o, (\sigma_2, \sigma_2') \in H^o \ , \tag{16a}$$

$$\forall(\sigma_1, \sigma_2) \in H^o, \exists(\sigma_1', \sigma_2') \in H^o, \text{ s.t. } (\sigma_1, \sigma_1') \in V^o, (\sigma_2, \sigma_2') \in V^o \ . \tag{16b}$$

The adjacency patterns for which algorithm 2 can fail to find a labeling satisfying the constraints, do not encode more useful priors than ones that satisfy condition (16). In particular, for any prior for which $\forall s \in S,\ (s, s) \in V$ and $(s, s) \in H$, algorithm 2 always finds a valid labeling. The same holds for adjacency patterns obtained by transforming grid patterns according to (1), and the alphabet of irregular shapes presented in section 2.2 of the main paper. Finally, algorithm 2 can easily be modified to detect a situation when it fails to generate a labeling that respects all the constraints.

The complexity of the Viterbi algorithm used to solve the subproblems can be decreased by exploiting the fact that only some pairs of neighboring classes are allowed. In each step, the algorithm determines the optimal class of the previously processed pixel, given the class of the current one. We can speed up the process by only checking the previous classes that constitute valid pairs with the current class. This is achieved by storing the matrices of allowed classes of neighboring pixels in sparse form. We compare the dependence of the running time on the number of classes for the sparse and dense representations and present the results in figure 1. We vary the number of classes by starting with a complex prior with many classes and then randomly removing classes. The number of classes is a good measure of complexity of shapes represented by a prior. Exploiting the sparsity in the implementation of the Viterbi algorithm results in a speedup of roughly 5 times in this particular example.

4. Confusion matrices

We present the confusion matrices corresponding to experiment results announced in the paper in tables 1, 2, 3, 4, 5.

References

[1] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE Trans. PAMI*, 33(3):531–552, 2011.

[2] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

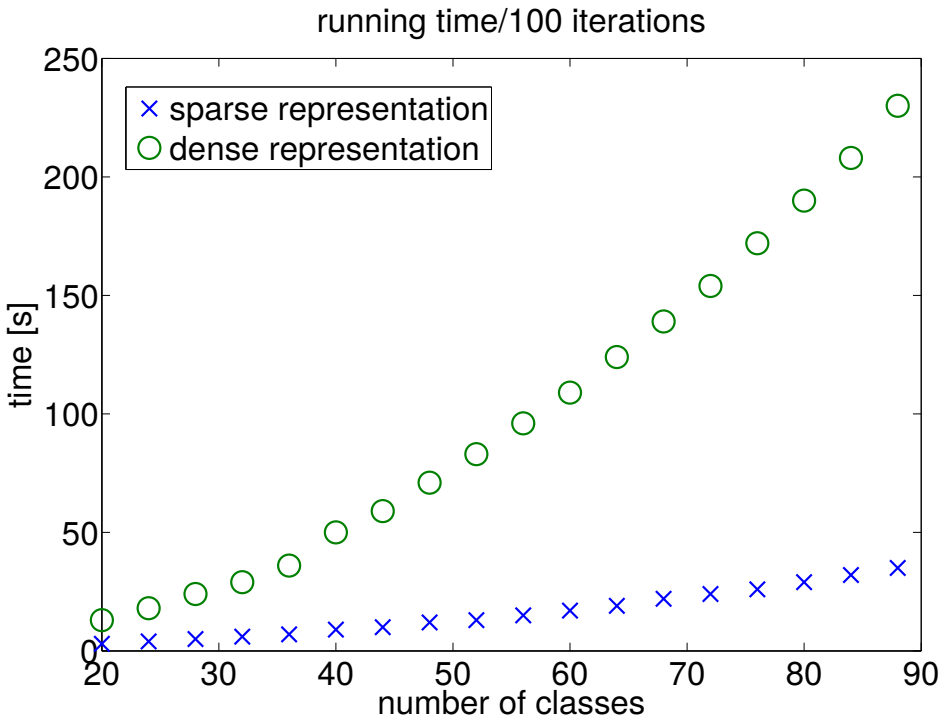

Figure 1. The running time with respect to number of (pre-semantic) classes for the implementation based on sparse and dense representation of a table of allowed neighbor classes. The displayed time contains 100 iterations of the dual decomposition algorithm and extraction of the primal solution. The experiment has been performed on a single image of the ECP dataset. We used a 6-core Core-i7 processor with 3GHz clock.

Table 1. Confusion matrix of results on the ECP dataset with unary potentials obtained using a variant of TextonBoost

	roof	shop	balcony	sky	window	door	wall
roof	91	0	0	2	3	0	4
shop	0	97	0	0	0	0	3
balcony	1	0	91	0	3	0	5
sky	3	0	0	97	0	0	0
window	3	1	4	0	87	0	5
door	0	19	0	0	0	79	3
wall	1	3	2	0	3	1	90

Table 2. Confusion matrix of our results on the ECP dataset using RNN-based unaries.

	roof	shop	balcony	sky	window	door	wall
roof	78	0	1	8	4	0	10
shop	0	90	0	0	0	0	9
balcony	2	0	76	0	6	0	16
sky	5	0	0	94	0	0	0
window	4	1	4	1	67	0	22
door	0	47	0	0	0	44	9
wall	0	3	1	0	1	0	93

Table 3. The confusion matrix of results of the experiment on the Graz50 dataset.

	sky	window	door	wall
sky	93	0	0	6
window	0	84	0	16
door	0	12	60	27
wall	1	3	0	96

Table 4. The confusion matrix of our results on the ArtDeco dataset.

	roof	shop	balcony	sky	window	door	wall
roof	75	0	2	8	5	0	10
shop	0	97	0	0	0	1	3
balcony	1	0	73	0	7	0	19
sky	2	0	0	98	0	0	0
window	3	0	5	0	74	0	17
door	0	58	0	0	0	37	4
wall	1	1	4	0	4	0	90

Table 5. The confusion matrix of our results on the eTrims dataset with RNN-based unaries.

	building	car	door	pavement	road	sky	vegetation	window
building	92	1	0	0	0	0	2	4
car	19	70	0	3	3	0	4	1
door	56	5	20	0	0	0	6	14
pavement	24	12	0	33	30	0	1	0
road	10	6	0	27	56	0	0	0
sky	4	0	0	0	0	96	0	0
vegetation	6	1	0	1	0	1	91	1
window	27	1	0	0	0	0	1	70