A Robust Frank-Wolfe Method for MAP Inference

Gang Chen and Ran Xu
Dept. of Computer Science and Engineering
SUNY at Buffalo
Buffalo, NY 14216
{gangchen,rxu2}@buffalo.edu

Abstract

Finding maximum a posterior (MAP) estimation is common problem in computer vision, such as the inference in Markov random fields. However, it is in general intractable, and one has to resort to approximate solutions, e.g. quadratic programming. In this paper, we propose a robust Frank-Wolfe method [6] to do the MAP inference. Our algorithm optimizes the quadratic programming problem by alternating projections between the discrete domain and the continuous domain (relaxed space). If the solution in the discrete domain keeps the energy climbing in the current step in both the discrete and continuous domains, we push the algorithm ahead to that direction in the following steps. Otherwise, we backtrack our algorithm to the continuous domain, which can find a non-discrete solution and improve the quadratic function climbing towards the integer solution. We analyze our algorithm and show the backtrack step under the Frank-Wolfe Method framework can guarantee the energy increasing in the following gradient updating step. We show the advantages of our algorithm by significantly outperforming integer projected fixed point method (IPFP) and other baselines.

1 Introduction

MAP inference is an important problem in computer vision and machine learning, and has been widely used on 2D/3D image segmentation [7], object recognition [10, 5] and label inference [15, 12]. In general, a weighted graph is used to represent the object, and then an integer quadratic function is constructed with unary and pairwise information to measure the relationship between nodes and labels. However, it is NP-hard to do MAP inference because it is an integer optimization problem. Thus, much effort has gone into development of efficient algorithms for this problem. One trend is to directly optimize this problem in discrete space, such as graph cuts, branch and bound methods [9, 4] and Tabu search [1]. Another direction is to relax the problem into continuous domain. The classical optimization algorithms usually find optimal continuous solution of the relaxed problem, and then discretize it, such as belief Propagation (BP), simulated annealing [13], graduated assignment [7] and spectral graph matching [10, 3]. However, these methods like graph cuts and BP, either have restrictions on the clique potentials, or require sparse graph connections. In addition, many quadratic programming methods assume the continuous optimum is close to the discrete global optimum of the original combinatorial problem. Recently, an interesting algorithm, the Integer Projected Fixed Point Method (IPFP) [12] is proposed for MAP Inference and graph matching. This algorithm is an efficient approach, which optimizes in the discrete domain, and guarantee convergence properties. However, it is a kind of greedy approach and can trapped easily into bad local maximum, because it cannot guarantee the continuous solution with backtrace can move towards better discrete solutions. In general, this approach often stop early with bad local maximum.

In this paper, we prefer the relaxation and rounding techniques because it does not need to search the state-space and generally have lower computational complexity. And we introduce a robust Frank-Wolfe Method which can be used for graph matching and MAP inference and solve them

efficiently. Our algorithm first relax the integer quadratic programming into continuous domain, and then optimize it under Frank-Wolfe framework using alternative projection. Basically, we first find the best integer solution, and project it into continuous space if it cannot guarantee the function climbing in the next step. We show that our method always move towards discrete solution and can converge into good local (or global) maximum. Under certain conditions, our method can coverage in $O(\frac{1}{k})$.

2 Overall view

The MAP inference problem is to find a label vector $\mathbf{x}^*$ that maximize a certain quadratic function $f(\mathbf{x}) = (\mathbf{x}^T\mathbf{M}\mathbf{x})$ under discrete constraints, where $\mathbf{M}$ is an matrix measure the comparability between nodes and labels in the graph, and $\mathbf{x}$ is required to be an indicator vector with an entry for each pair of (i, a). If $\mathbf{x}_{ia} = 1$, then the node i assigns the label a, and $\mathbf{x}_{ia} = 0$ otherwise. $\mathbf{M}$ is usually a symmetric matrix with positive elements containing the compatibility score functions, such that $\mathbf{M}_{ia;jb}$ measures how similar the pair of sites (i, j) is compatible with labels (a, b). Note the for the discrete optimization problem, we can always modify $\mathbf{M}$ into a matrix with positive elements, without any changing to the optimal solution [11].

In this part, we first consider the integer quadratic programming, then we relax it into continuous constraints. We take a formula and notations that are similar to [13, 12].

OP1:

$$\mathbf{x}^* = \text{argmax} f(\mathbf{x}), \text{s.t. } \mathbf{A}\mathbf{x} = 1, x \in \{0, 1\}^n \tag{1}$$

In general, we modify OP1, usually by relaxing the constraints on the solution, in order to be able to find efficiently optimal solutions to the new problem.

OP2:

$$\mathbf{x}^* = \text{argmax} f(\mathbf{x}), \text{s.t. } \mathbf{A}\mathbf{x} = 1, x \geq 0, \tag{2}$$

OP2 is also NP-hard, and it becomes a concave minimization problem, equivalent to problem 1, when $\mathbf{M}$ is positive definite. Suppose the feasible region for OP1: $x \in \mathcal{D}_1$, and the feasible region for OP2: $x \in \mathcal{D}_2$. Also note that $\mathcal{D}_1 \subset \mathcal{D}_2$. If the integer quadratic programming OP1 has the optimal discrete solution x^*, then it must be the optimal solution to OP2, but not vice verse.

2.1 The Frank-Wolfe method

The Frank-Wolfe method [6, 8] relax the quadratic programming into linear subproblems, and is widely used for optimization on continuous domain. We denote C_f to be the curvature constant [8], then the framework of Frank-Wolfe method is listed below in algorithm 1.

Algorithm 1 Frank-Wolfe with Approximate Linear subproblems, for Quality $\delta > 0$

1: Let $\mathbf{x}_0 \in \mathcal{D}_2$,
2: **for** $K = 0$ to T **do**
3: Update $\gamma = \frac{2}{k+2}$;
4: Find $\mathbf{s} \in \mathcal{D}$, s.t. $\langle \mathbf{s}, \nabla(f(\mathbf{x}^{(k)}))\rangle \geq \max_{\hat{\mathbf{s}}\in\mathcal{D}}\langle \hat{\mathbf{s}}, \nabla(f(\mathbf{x}^{(k)}))\rangle - \frac{1}{2}\delta C_f$
5: a) performing line search for γ (optional), by maximizing
$\gamma = \max_\gamma f\big(\mathbf{x}^{(k)} + \gamma(\mathbf{s} - \mathbf{x}^{(k)})\big)$
6: b) Update $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \gamma(s - \mathbf{x}^{(k)})$;
7: **end for**
8: Return $\mathbf{x}$;

The IPFP is a kind of Frank-Wolfe method to optimize OP1. Basically, it first finds a discrete solution according to linear programming or Hungarian algorithm, then it backtracks to find γ by performing line search in Alg. 1, which in general leads to continuous solutions. Assume $\mathbf{s} = P_d(\mathbf{M}\mathbf{x}^{(k)})$ is the current solution in the domain $\mathcal{D}_1$, in which P_d is the projection on the one-to-one or many-to-one discrete constraints. And further we denote $C = (\mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{s} - \mathbf{x}^{(k)})$, and $D = (\mathbf{s} - \mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{s} - \mathbf{x}^{(k)})$, which respectively are the first and the second order differences in Taylor series.

Firstly, we argue that we cannot guarantee $C \geq 0$ in each step, because $\mathbf{s}$ maximize $\mathbf{s}^T\mathbf{M}\mathbf{x}^{(k)}$ in the discrete domain $\mathcal{D}_1$, and if $\mathbf{x}^{(k)} \in \mathcal{D}_2$, then it is quite possible $(\mathbf{x}^{(k)})^T\mathbf{M}\mathbf{s} < (\mathbf{x}^{(k)})^T\mathbf{M}\mathbf{x}^{(k)}$. In other words, we may have $C < 0$ when we project the solution from $\mathcal{D}_2$ into $\mathcal{D}_1$.

Secondly, according to the Frank-Wolfe method in Alg. 1, for the new updating $\mathbf{x}^{(k+1)} \in \mathcal{D}_2$, we can express $f(\mathbf{x}^{(k+1)})$ as follows

$$f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)}) + (\mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + O(\delta^2) \tag{3}$$

$$= f(\mathbf{x}^{(k)}) + \gamma(\mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{s} - \mathbf{x}^{(k)}) + \frac{\gamma^2}{2}(\mathbf{s} - \mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{s} - \mathbf{x}^{(k)}) + O(\delta^2) \tag{4}$$

$$= f(\mathbf{x}^{(k)}) + \gamma C + \frac{\gamma^2}{2}D + O(\delta^2) \tag{5}$$

Since $\gamma \in [0, 1]$, the best step for γ can be estimated by setting its gradient to zero

$$\gamma = \min(1, \frac{-C}{D}), \text{s.t. } \gamma \geq 0 \tag{6}$$

However, for the current local optimum $\mathbf{s}$, if $f(\mathbf{s})$ is not better than $f(\mathbf{x}^{(k+1)})$ in Eq. 5, it means that $\mathbf{s}$ guides the OP1 problem into a wrong direction, which cannot guarantee the quadratic function climbing in OP2. Note that the difference between $f(\mathbf{s})$ and $f(\mathbf{x}^{(k+1)})$ can be written with simple algebra manipulation

$$f(\mathbf{x}^{(k+1)}) - f(\mathbf{s}) = \frac{1}{2}(\gamma^2 - 1)D + (\gamma - 1)C \tag{7}$$

where we ignore the high order difference for analysis convenience. Note that $\gamma \in [0, 1]$, thus if $\gamma = 1$ in Eq. 7, then the current $\mathbf{s}$ is also the optimum in the continuous domain $\mathcal{D}_2$ for the OP2 problem. Thus, it can guarantee the objective function climbing in the discrete domain $\mathcal{D}_1$. However, if $\gamma < 1$, we cannot guarantee $f(\mathbf{x}^{(k+1)})$ is better than $f(\mathbf{s})$, which in turn will guide the solution in the following steps into a bad local minimum.

Now Let us go back to Eq. 5. If $D \geq 0$, then $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$ holds, which guarantee the objective climbing in the following steps. Otherwise, If $D < 0$, we cannot guarantee $f(\mathbf{x}^{(k+1)})$ has better score than $f(\mathbf{x}^{(k)})$. Furthermore, if $\frac{-C}{D} < 0$, it will make the case worse. Thus, we propose a the Frank-Wolfe method with alternative projections on the two optimization problems. Our algorithm is listed below in Alg. 2.

2.2 Basic idea

To address this issue, we propose a robust Frank-Wolfe algorithm, which optimizes the quadratic function between discrete and continuous domains. If the discrete solution $\mathbf{x}$ is optimal in OP1, then we trust it and continue its direction to the integer solution. Otherwise, given the current solution $\mathbf{x}$, we optimize the continuous problem OP2, which can always improve the quadratic function $f(\mathbf{x})$ towards to the integer solution. In this case, it is a kind of optimization in the continuous domain, and then we project it back into discrete domain in the following iterations.

If the current discrete solution is trapped in a bad local minimum, our algorithm fell back upon the continuous domain to jump out of the bad case. Thus, we propose the Robust Frank-Wolfe method for integer quadratic programming problem in Alg. 2 below. In our implementation, we use a window size w to computer the average $\mathbf{x}^{(k)} = \sum_{t=k}^{k-w} \mathbf{x}^{(t)}$ in order to make the algorithm stable.

Algorithm 2 Robust Frank-Wolfe method

```
Let x_0 ∈ D_2,
for k = 0 to K do
    Compute gradient ∇(f(x^(k))) given the current solution x^(k);
    Find s ∈ D_1, by maximizing ⟨s, ∇(f(x^(k)))⟩
    Update C = (x^(k))^T M(s − x^(k)), and D = (s − x^(k))^T M(s − x^(k))
    Update γ via Eq. 6;
    if C < 0 then
        (a) x^(k+1) = Backtrack(x^(k), M, β, T);
    else
        if D ≥ 0||γ == 1 then
            (b_1) x^(k+1) = s;
        else
            (b_2) Update x^(k+1) = x^(k) + γ(s − x^(k));
        end if
    end if
end for
if x ∉ D_1 then
    Project x into D_1;
end if
Return x;
```

The backtracking subroutine in our algorithm 2 is to find a solution by monotonically increasing function of $\mathbf{x}^{(k)}$ towards an integer solution. If f is increasing exponentially with β then, by increasing β, we make the assignment at each step similar to a max function such that $\mathbf{x}$ gets closer and closer to the original integral constraints in OP1. We need to design a algorithm, which can not only increasing the objective function, but also guide the current solution into the integer domain. In our backtrack stage, we take a strategy similar to the graduated algorithm [7] and climbing algorithm [11]. Note that our algorithm here is only for backtracking in the right direction, which is different from other methods [7, 11, 2]. In other words, the algorithm only need to move towards the integer solution with just looping a few times, for example $T = 3$, instead of repeating until convergence like graduated assignment or random work [2]. Since every time we visit a site i we only update the values in vector $\mathbf{x}$ corresponding to that site, we can write the objective at that moment t as $f(\mathbf{x})^{(t)} = f(\mathbf{x}_{-i})^{(t)} + f(\mathbf{x}_{i.})^{(t)}$, where $\mathbf{x}_{-i}$ are all the other sites or features except i, and $\mathbf{x}_{i.}$ is the set collected to the site i. Thus $f(\mathbf{x}_{-i})^{(t)}$ is independent of $\mathbf{x}_{ia}$ for any label a. We can show that $f(\mathbf{x}_{i.})^{(t)} \leq f(\mathbf{x}_{i.})^{(t+1)}$, and furthermore, we have $f(\mathbf{x})^{(t)} \leq f(\mathbf{x})^{(t+1)}$. For the proof, refer to [14, 11].

Algorithm 3 Backtrack

Input: similar matrix $\mathbf{M}$, $\mathbf{x}$, β and iterations T;
Output: $\mathbf{x}$;

for $t = 1$ to T **do**
  $\mathbf{v} = \mathbf{M}\mathbf{x}$;
  $\mathbf{v} = \mathbf{x} \cdot \mathbf{v}^{\beta}$; //point-wise product
  for each i **do**
    normalize $\mathbf{v}_{ia} = \frac{\mathbf{v}_{ia}}{\sum_a \mathbf{v}_{ia}}$;
    update $\mathbf{x}_{ia} = \mathbf{v}_{ia}$;
  end for
  Increase β;
end for
Return $\mathbf{x}$;

2.3 Theoretic Analysis

Property 1: The quadratic function $f(\mathbf{x}^{(k)}) = (\mathbf{x}^{(k)})^T\mathbf{M}(\mathbf{x}^{(k)})$ increases at every step k and the sequence of $\mathbf{x}^{(k)}$ converges.

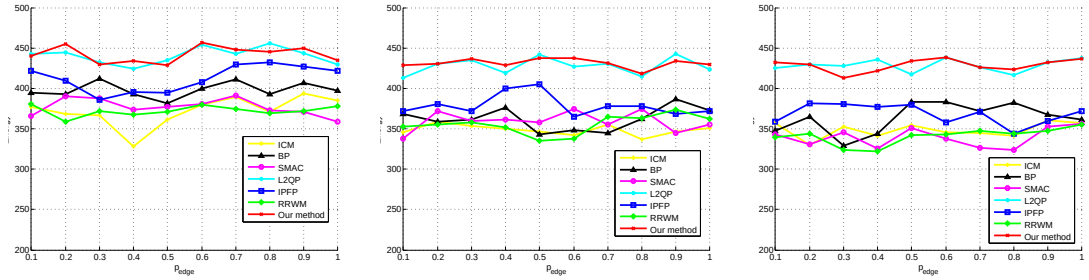
Proof: For a given step k, Algorithm 2 has two conditions to update $\mathbf{x}^{(k+1)}$. If $C < 0$, then we use algorithm 3 to updating the $\mathbf{x}^{(k+1)}$ in the continous domain, which will keep function climbing. If $D \geq 0$ or $\gamma == 1$, then we update $\mathbf{x}^{(k)}$ according to step (b_1) in Algorithm 2. More specifically, if $D \geq 0$, then we $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$ via Eq. 5. Similarly, if $\gamma == 1$, we can yield the same conclusion via Eq. 7. Otherwise, $\mathbf{x}^{(k+1)}$ will be updated according to step (b_2) in Algorithm 2. As we mentioned before, it always improves the function score and move towards to integer solution. Thus, we also have $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$.

Property 2: If $\mathbf{M}$ is positive semidefinite with positive elements, then the algorithm converges in a finite number of iterations to a discrete solution for OP1, which is a maximum of OP2. And further, the algorithm will converge in $O(\frac{1}{k})$ if the step size in the Backtrack subroutine kept in appropriate range.

Proof: Since $\mathbf{M}$ is positive semidefinite, there's global optimum for OP1. According to the monotone convergence theorem, the algorithm must converge in a finite number of steps to a local (or global) maximum, which must be discrete. This result is obviously true for both one-to-one and many-to-one constraints. When $\mathbf{M}$ is positive semidefinite, the algorithm will converge to the global optimum for OP1, which is also true for OP2. Note that $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$. Then there exist γ, which makes $f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)}) + (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T \nabla f\big(\mathbf{x}^{(k)} + \gamma(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})\big)$ hold according to mean value theorem. And if the step side updated in step (b) in the algorithm is small enough, then it will make $\gamma \in [0, 1]$. Then we will have $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) < O(\frac{1}{k})$. For the proof, please refer to [8].

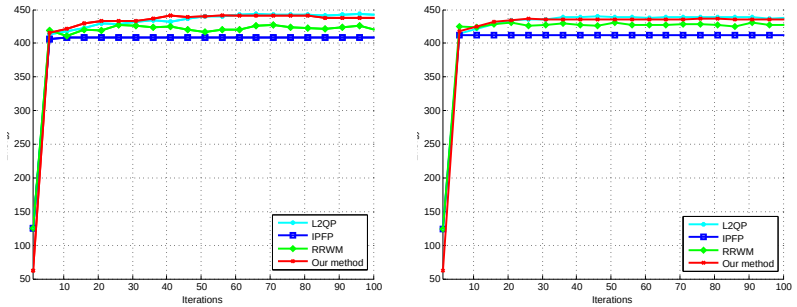
3 Experiments

We compared our algorithm against other methods such as BP, ICM, SMAC [3], IPFP, L2QP [11] and RRWM [2]. For a more thorough analysis, we repeated 5 times to generate synthetic graphs with different nodes and labels, and averaged the output in the experiments. We focused more on the degree of connectedness and the number of labels relative to the number of nodes, and evaluated



(a) (b) (c)

Figure 1: It shows how the energy function changes with p_{edge} for different methods for the same graph with 30 nodes. (a) the result with 5 different labels; (b) the result with 15 labels; (c) indicates the result with 25 different labels.



(a) (b)

Figure 2: It shows how the energy function changes for a given p_{edge} for different methods for the same graph with 30 nodes. It demonstrates that our method yield a higher and even better energy score than baselines. (a) shows how each algorithm converges for the 40% connectivity graph; (b) shows how each algorithm converges for the 80% connectivity graph.

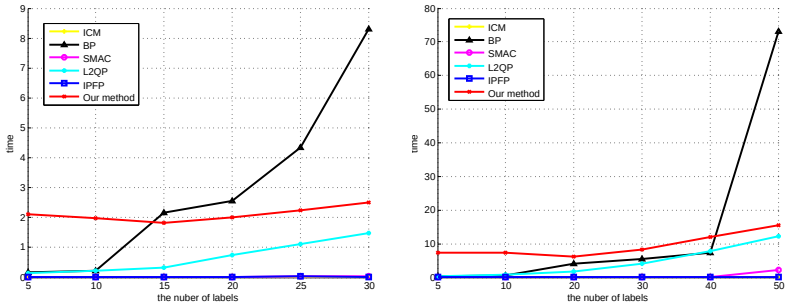
the performance of different methods according to the energy function. Without other specification, we set $p_c = 0.8$, $p_w = 0.4$, $K = 100$ iterations for each algorithm in the experiments.

Similar to [11], we generated a set of graphs by controling the density p_{edge} of connections in $\mathbf{M}$: $p_{edge} \in [0, 1]$. To generate $\mathbf{M}$, we encouraged connections between pairs of nodes with the correct labels (set arbitrarily in advance) to be larger than other connections on average. More specifically, we set $\mathbf{M}_{ia,jb} = log(p/\epsilon)$ with probability p_0, and $\mathbf{M}_{ia,jb} = 0$. If the pair sites (i, j) are connected, then it has higher probability (p_c) to have the same label, otherwise we set a lower probability (p_w).

We generated a graph with 30 nodes and varied the number of labels to test the performance of different methods, and the results are shown in Fig. 1. Note that p_{edge} in Fig. 1 indicates the density of connection in graph. As for p_{edge}, lower value indicates higher density, which is different from [11]. It shows our method yields comparative results, and outperforms all other methods, except L2QP.

We also consider how the energy changes given the density graph. In this experiment, we set the number of nodes to be 30, and the number of labels to be 10. Then, we analyzed how the energy changes with graph density. The result in Fig. 2 shows that for the graph with 80% connectivity, our method can rapidly converge.

Finally, we analyzed the time complexity for our method, shown in Fig. 3. We ignore the result of RRWM because it highest time demanding, with at least a order of magnitude slower slower than other methods. From Figs. 1 and 3, our method is comparable to L2QP in both accuracy and time complexity. And our method can beat most baselines, with a little sacrifice in time complexity.



(a) (b)

Figure 3: It shows how the time complexity changes with the number of labels for different methods. (a) shows the time complexity for different methods on the graph with 30 nodes and 80% edges connectivity; (b) shows how time changes with respect to the number of labels on the graph with 50 nodes and 80% edges connectivity.

4 conclusion

In this paper, we propose a robust Frank-Wolfe method to optimize MAP inference problems. Our algorithm optimizes the quadratic programming problem by alternating projections between the discrete domain and the continuous domain (when necessary). We analyze our algorithm and show the backtrack step under the Frank-Wolfe Method framework can guarantee the energy increasing in the following gradient updating step. In the experiments, we show the advantages of our algorithm by significantly outperforming IPFP and other baselines.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[2] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV*, volume 6315 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2010.

[3] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006.

[4] M. A. Eshera and K. Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):398–408, 1984.

[5] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(1):36–51, 2008.

[6] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

[7] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE TPAMI*, 18(4):377–388, Apr. 1996.

[8] M. Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*, pages 427–435, 2013.

[9] E. L. Lawler and D. E. Wood. Branch-and-bound methods a survey. *Oper. Res.*, 149, 1966.

[10] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489, 2005.

[11] M. Leordeanu and M. Hebert. Efficient MAP approximation for dense energy functions. In *ICML*, pages 545–552, 2006.

[12] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *NIPS*, pages 1114–1122, 2009.

[13] A. Rangarajan. Self annealing and self annihilation: Unifying deterministic annealing and relaxation labeling. In *In Pattern Recognition*, pages 33–635, 2000.

[14] A. Rangarajan, A. Yuille, and E. Mjolsness. Convergence properties of the softassign quadratic assignment algorithm. *Neural Computation*, 11:1455–1474, 1999.

[15] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, pages 596–609, Berlin, Heidelberg, 2008.