Min Norm Point Algorithm for Higher Order MRF-MAP Inference

Ishant Shanu Chetan Arora
IIIT Delhi, India

Parag Singla
IIT Delhi, India

Abstract

Many tasks in computer vision and machine learning can be modelled as the inference problems in an MRF-MAP formulation and can be reduced to minimizing a submodular function. Using higher order clique potentials to model complex dependencies between pixels improves the performance but the current state of the art inference algorithms fail to scale for larger clique sizes. We adapt a well known Min Norm Point algorithm from mathematical optimization literature to exploit the sum of submodular structure found in the MRF-MAP formulation. Unlike some contemporary methods, we do not make any assumptions (other than submodularity) on the type of the clique potentials. Current state of the art inference algorithms for general submodular function takes many hours for problems with clique size 16*, and fail to scale beyond. On the other hand, our algorithm is highly efficient and can perform optimal inference in few seconds even on clique size an order of magnitude larger. The proposed algorithm can even scale to clique sizes of many hundreds, unlocking the usage of really large size cliques for MRF-MAP inference problems in computer vision. We demonstrate the efficacy of our approach by experimenting on synthetic as well as real datasets.*

1. Introduction

Many problems in computer vision and machine learning can be reduced to assigning a label on each pixel. The label may denote different quantities depending upon the application, for example depth, in the stereo matching problem or pixel intensity in the image denoising problem. Finding the best labeling configuration can be formulated as finding the minimum of the following energy equation:

$$E(\mathbf{l}_{\mathcal{P}}) = \min_{\mathbf{l}_{\mathcal{P}}} \sum_{\mathsf{c}\in\mathcal{C}} W_{\mathsf{c}}(\mathbf{l}_{\mathsf{c}}). \quad (1)$$

Here, c, also called a *clique*, is defined as the set of pixels whose labels are contextually dependent on each other. A labeling configuration on a clique c is denoted as $\mathbf{l}_{\mathsf{c}}$. $\mathcal{P}$ denotes the set of all pixels and $\mathcal{C}$ denotes the set of all cliques. The size of the maximal clique $k = \max_{\mathsf{c}} |\mathsf{c}|$ is known as the *order* of the problem. Each term, $W_{\mathsf{c}}(\mathbf{l}_{\mathsf{c}})$, also called the *clique potential*, measures the cost of the labeling configuration $\mathbf{l}_{\mathsf{c}}$ of a clique c depending on how consistent the labeling is with respect to the observation and prior knowledge. The formulation is popularly known as MRF-MAP inference in computer vision and structured prediction in the machine learning community. Over the last decade many computer vision problems ranging from image restoration [14], segmentation of videos [5] and images [40], super resolution [30], texture synthesis [28], stereo matching [3] to object detection [16] have been formulated as the MRF-MAP inference problems.

The focus of this paper is on efficient inference in MRF-MAP problems with 2 labels. Apart from having applications in its own right, inference for multi label problems is often formulated as a series of inference on 2-label problems [3, 10]. Optimal inference even for a 2-label MRF-MAP problem is NP hard in general. However, given the importance of the problem, researchers have focussed on various subsets of the problems for which efficient inference is possible. *Submodularity* is one such property which can naturally capture commonly occurring constraints in problems from computer vision, machine learning and signal processing. A set function $f : 2^V \rightarrow \mathbb{R}$ on a set V is called submodular if for all subsets $S, T \subseteq V : f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$. Note that for a 2-label problem each clique potential function can also be seen as a set function where a label 1 implies inclusion in the set and 0 implies exclusion. Hence, the 2-label case can be seen as the problem minimizing a sum of submodular functions.

Our focus in this paper is on problems with higher order clique potentials ($k > 2$) which can encode various structural and complex dependencies between pixels. It has been adequately shown by various authors [17, 22, 27, 38, 39, 47] that using such complex dependencies greatly improves the solution quality.

There have been many algorithms proposed for inference in higher order problems during last few years. Some notable work in this area includes Message Passing and variations of Dual Decomposition [7, 15, 20, 24, 27, 34, 36, 43, 45] or reduction techniques [8, 11, 13, 17, 18, 21, 26, 37, 39] which convert higher order cliques to pairwise ones and then use QPBO [25] to solve the reduced problem. Despite the best efforts, efficient inference for general sum of submodular functions has been shown to be possible only for clique size up to 16 [1].

Considering the complexity of inference for general sum of submodular functions, researchers have suggested efficient algorithms for various special subsets of submodular functions. Jegelka et al.[20] applied two well known (block coordinate descent and Douglas Rachford splitting) dual decomposition procedures to solve sum of special submodular functions for which $\min_{S\subseteq V}(f(S) - a(S))$ can be computed efficiently, where $a \in R^V$ is a constant vector. Ene and Nguyen [7] focus on the functions for which $f + w$ can be computed efficiently, where w is a linear function. Stobbe and Krause [44] use Lovasz extension for problems with concave clique potentials $f(S) \propto |S||V - S|$, where V is set of all the nodes. Ramalingam et al.[37] use monotonic functions and Rother et al.[39] use sparse potentials having non-zero costs only for a few labeling configurations.

It is useful to note that the sum of submodular functions is also a submodular function and techniques from mathematical optimization for submodular function minimization (hereon SFM) can be potentially applied for solving (1) as well. However, with a high order polynomial complexity of n^5 or higher (n is the number of pixels), these techniques [12, 19, 29, 33, 41] fail to scale for problem sizes typically found in computer vision. Kolmogorov [23] has suggested adapting submodular flow technique proposed by Iwata et al.[19] for sum of submodular problems. However, the contribution is essentially theoretical with no implementation or application shown.

We propose an algorithm for inference in 2-label higher order MRF-MAP problem (1), when the clique potentials are submodular. The paper has following specific contributions:

1. **Structure:** Our algorithm uses the ideas from min-norm-point algorithm [12] and adapts them for exploiting the sum of submodular structure in our problems.

2. **General:** Unlike contemporary approaches, the proposed algorithm can give optimal inference for general submodular functions with no extra conditions on the type of clique potential.

3. **Scalable:** In a significant improvement over state of the art, our algorithm can easily scale to problems with clique sizes ranging up to many hundreds compared to 16 for current state of the art [1].

4. **Efficient:** When the earlier state of the art can take hours for inference on clique size 16, our experiments show that the proposed algorithm converges with optimal inference in a matter of few seconds on problems with order of magnitude larger clique sizes.

2. Background

In this section we discuss some results from the standard SFM literature which will be useful for this paper. We extend the results to sum of submodular functions in the next section when we describe our algorithm. We will use the abbreviation SoS to refer to the phrase Sum of Submodular. To maintain inter-operability, we will use the notation as is common in the SFM literature. The objective is to find a minimizer set, $S^* = \min_{S\subseteq V} f(S)$, of a submodular function f, where V is the set of all the elements and S is any subset of V. Without loss of generality, we will assume that $f(\{\}) = 0$. There are two polyhedra in $\mathbb{R}^V$ associated with f, the *submodular polyhedron*, $P(f)$, and *base polyhedron*, $B(f)$, defined as follows:

$$P(f) = \{x \mid x \in \mathbb{R}^V,\ \forall\, U \subseteq V : x(U) \leq f(U)\},$$
$$B(f) = \{x \mid x \in P(f), x(V) = f(V)\},$$

where $x(U) = \sum_{v\in U} x(v)$. Here, $x(v)$ denotes the element at index v in the vector x. A vector in the base polyhedron $B(f)$ is called a *base*, and an extreme point of $B(f)$ an *extreme base*. Edmond's greedy algorithm gives a simple procedure to create an extreme base, $b^{\prec}$, given a total order $\prec$ of elements of V such that: $\prec : v_1 \prec \ldots \prec v_n$. Denoting by $i_{\prec}$ the set of elements $\{1, \ldots, i\}$, the algorithm initializes the first element as $b^{\prec}(1) = f(\{v_1\})$ and rest of the elements as $b^{\prec}(k) = f(k_{\prec}) - f((k-1)_{\prec})$.

For any vector $x \in \mathbb{R}^V$, we denote by x^- the vectors in $\mathbb{R}^V$ defined by $x^-(v) = \min\{0, x(v)\}$ for $v \in V$. It is easy to see that for any base x and a subset U: $x^-(V) \leq x(U) \leq f(U)$. The Min Max Theorem as given below shows that the inequalities are tight at the maximum and minimum values of $x^-(V)$ and $f(U)$ respectively.

Theorem 2.1 (Min Max Theorem)**.** *Given a submodular function $f : 2^V \to R$, we have*

$$\max\{x^-(V) \mid x \in B(f)\} = \min\{f(U) \mid U \subseteq V\}.$$

Most of the algorithms for SFM suggested in combinatorial optimization community actually solve this dual problem. The earliest strongly polynomial time algorithm for SFM was given by Schrijver [41], which maintains a set of extreme bases, b_i, and a solution vector, x, as a convex combination of these extreme bases. The algorithm then tries to find a new x with higher $x^-(V)$, by inserting/deleting extreme bases using an operation known as *exchange moves*. The complexity of the algorithm is $O(n^8)$. Schrijver's algorithm tends to waste a lot of time in some exchange moves which ultimately get reversed in a later stage. Iwata et al.[19] improved the strategy by accumulating the exchange moves using a flow like strategy. They have given a strongly polynomial time algorithm with time complexity of $n^7 \log n$. The current fastest combinatorial algorithm using a similar strategy for SFM is due to Orlin et al.[33] with a time complexity of n^6.

2.1. Min Norm Point Algorithm [12]

Min Norm Point algorithm reduces the problem of finding the minimum of f to finding a base x with minimum norm in the convex hull of all the extreme bases. The following lemma establishes the equivalence of the problem:

Lemma 2.2 (Min Norm Equivalence)**.** *Suppose x^* is a minimizer of the problem:*

$$\min \|x\|^2 \textit{ subject to } x \in B(f).$$

Then a minimizer S^* for f can be obtained as follows:

$$S^* = \{u \in V \mid x^-(u) \leq 0\}.$$

Using the above equivalence, in the rest of the paper, we will focus on the task of finding $\min_{x\in B(f)} \|x\|^2$ and equivalently, we would have solved the SFM problem. A popular strategy to find the minimum norm point is using Wolfe's algorithm as described below.

2.2. Wolfe's Algorithm [12, 46]

Given a finite set P of points $p_i \in \mathbb{R}^n$ $(i \in I)$, Wolfe's algorithm gives an efficient way to find the minimum-norm point x^* in the convex hull $\hat{P}$ of points p_i $(i \in I)$. The algorithm can be described as follows [12]:

Procedure 1 Wolfe's Algorithm

```
Input: A finite set $P$ of points $p_i$ $(i \in I)$ in $\mathbb{R}^n$.
Output: The minimum-norm point $x^*$ in the convex hull of $P$.
1: Choose any point $p \in P$ and initialize $S := \{p\}$ and $x := p$.
2: Find a point $\hat{p}$ in $P$ that minimizes the dot product $\hat{p} = \arg\min_{p\in P} \langle x, p\rangle = \arg\min_{p\in P} \sum_{k=1}^{n} x(k)p(k)$. If $\|x\|^2 \leq \langle x, \hat{p}\rangle + \epsilon$, then return $x^* = x$ and halt; else put $S := S \cup \{\hat{p}\}$.
3: Find the minimum-norm point $y$ in the affine hull of points in $S$. If $y$ lies in the relative interior of the convex hull of $S$, then put $x := y$ and go to Step 2.
4: Let $z$ be the point that is the nearest to $y$ among the intersection of the convex hull of $S$ and the line segment $[y, x]$ between $y$ and $x$. Also let $S' \subset S$ be the unique subset of $S$ such that $z$ lies in the relative interior of the convex hull of $S'$. Put $S := S'$ and $x := z$. Go to Step 3.
```

The base polyhedron, $B(f)$, for a submodular function f, is the convex hull of all the extreme bases. Therefore, Wolfe's algorithm can be used to find the min norm point, $\min_{x\in B(f)} \|x\|^2$, by setting the set of points P as the set of extreme bases associated with $B(f)$. The n, in this case, is equal to the dimension of each of the points, i.e., $|V|$. Since the number of extreme bases can be $n!$, the key to the application of Wolfe's algorithm is step 2, which requires that the computation of $\min_{p\in P} \langle x, p\rangle$ be done efficiently. Fortunately, the Edmond's algorithm [12] gives a procedure to do this step in $n\log(n)$ steps as given in Procedure 2.

Procedure 2 Edmond's Algorithm

```
Input: A set of extreme bases $b \in B_f$. A base vector $x$.
Output: An extreme base $\hat{b} = \arg\min_b \langle x, b\rangle$.
1: Define an ordering $\prec$ on the indices, such that $x_1 \leq x_2 \ldots \leq x_n$.
2: Denote by $i_\prec$ the set of elements $\{1, \ldots, i\}$. Set $\hat{b}(i) = f(i_\prec) - f((i-1)_\prec)$.
```

Chakrabarty et al.[4] have shown that an approximate min-norm point x satisfying $\|x\| \leq \|x^*\| + \epsilon$ can be reached in $O(n^2F^2/\epsilon^2)$ number of iterations of Wolfe's algorithm. Here, $F = \max_{U,v} |f(U \cup \{v\}) - f(U)|$, where $U \subseteq V, v \in V$ i.e., the maximum increase in the value of the function after adding (deleting) an element to (from) a set. Correspondingly, they give a time complexity bound of $O((n^5 EO + n^7)F^2)$ for finding the minimizer of f [1]. Here, EO denotes the time complexity of an oracle call to evaluate the function f at any given subset $U \subseteq V$.

3. Proposed Algorithm: SoS Min Norm

The key to the scalability of the algorithms for computer vision problems is their ability to exploit the structure present in such problems. Specifically, the MRF-MAP problem (1) is minimizing the sum of submodular functions. Although, sum of submodular functions is also a submodular function and traditional SFM techniques can be applied, their usage is highly inefficient without exploiting the sum structure. In this work, we adapt the min norm point algorithm for exploiting the sum of submodular structure. We start with the theoretical foundations for our work, followed by the algorithm description. We show the effectiveness of our algorithm by running it on real computer vision problems in the next section.

3.1. Theoretical Results

Let f be a submodular set function on a set V of the form: $f(S) = \sum_{\mathsf{c}\in\mathcal{C}} f_\mathsf{c}(S \cap \mathsf{c})$, where $\mathcal{C} \subseteq 2^V$ is a set of subsets of V, and $f_\mathsf{c} : 2^\mathsf{c} \rightarrow \mathbb{R}$ are submodular functions. Our objective is to find the minimizer set $S^* = \arg\min_S f(S) = \arg\min_S \sum_\mathsf{c} f_\mathsf{c}(S \cap \mathsf{c})$. Since each f_c is submodular, we can associate, with each f_c, a

[1]The time complexity bound is applicable when f is integer valued; any function can be made integer valued by appropriate scaling [4].

base polyhedron given by:

$$B(f_{\mathsf{c}}) := \Big\{ y_{\mathsf{c}} \in \mathbb{R}^{\mathsf{c}} \;\mid\; y_{\mathsf{c}}(U) \leq f_{\mathsf{c}}(U), \quad \forall\, U \subseteq \mathsf{c}; \\ y_{\mathsf{c}}(\mathsf{c}) = f_{\mathsf{c}}(\mathsf{c}) \Big\}.$$

As defined earlier, y_{c} denotes a vector of scalars for every element in c: $y_{\mathsf{c}}(U) := \sum_{v \in U} y_{\mathsf{c}}(v), \quad \forall\, U \subseteq \mathsf{c}$.

We define a total order $\prec^{j}_{\mathsf{c}}$ for any $\mathsf{c} \in \mathcal{C}$ and correspondingly define the extreme base $q_{\mathsf{c},j} \in \mathbb{R}^{\mathsf{c}}$ using Edmond's algorithm. It follows that any convex combination of the extreme bases i.e., $y_{\mathsf{c}} = \sum_{j=1}^{k} \lambda_{\mathsf{c},j} q_{\mathsf{c},j}$, lies on the submodular polyhedron $B(f_{\mathsf{c}})$. We can now state the following:

Lemma 3.1. *Let $x(S) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap S)$ where each y_{c} lies on the submodular polyhedron $B(f_{\mathsf{c}})$. Then, the vector x lies on the base polyhedron $B(f)$.*

Proof. Consider $U \subseteq V$. To prove the lemma, it suffices to show that $x(U) \leq f(U)$ and $x(V) = f(V)$. It is easy to see that:

$$x(U) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap U) \leq \sum_{\mathsf{c}} f_{\mathsf{c}}(\mathsf{c} \cap U) = f(U).$$

The inequality follows from the fact that each x_{c} lies on the submodular polyhedron $B(f_{\mathsf{c}})$. Further,

$$x(V) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap V) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c}) = \sum_{\mathsf{c}} f_{\mathsf{c}}(\mathsf{c}) \\ = \sum_{\mathsf{c}} f_{\mathsf{c}}(\mathsf{c} \cap V) = f(V).$$

Hence, proved. □

Lemma 3.2. *Let x be a vector belonging to the base polyhedron $B(f)$. Then, x can be expressed as the sum: $x(S) = \sum_{\mathsf{c}} y_{\mathsf{c}}(S \cap \mathsf{c})$, where each y_{c} belongs to the submodular polyhedron $B(f_{\mathsf{c}})$ i.e., $y_{\mathsf{c}} \in B(f_{\mathsf{c}}) \forall \mathsf{c}$.*

Proof. Let us first prove the lemma for the case when x is an extreme base of $B(f)$. Using Edmond's algorithm, $\exists$ an ordering $\prec$ over the variables such that $x(i) = f(i_{\prec}) - f((i-1)_{\prec})$ (see Procedure 2 for details). Then, using the submodular decomposition of f, we have $x(i) = \sum_{\mathsf{c}} \left(f_{\mathsf{c}}(i_{\prec} \cap \mathsf{c}) - f((i-1)_{\prec} \cap \mathsf{c}) \right)$. Given a clique c, define $y_{\mathsf{c}}(i) = f_{\mathsf{c}}(\mathsf{c} \cap i_{\prec}) - f_{\mathsf{c}}(\mathsf{c} \cap (i-1)_{\prec})$. It is easy to see that y_{c} is an extreme base for $B(f_{\mathsf{c}})$, thus implying $x(i) = \sum_{\mathsf{c}} y_{\mathsf{c}}(i)$ for some $y_{\mathsf{c}} \in B(f_{\mathsf{c}}), \forall c$. The same can be extended to show that $x(S) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap S)$ by adding the corresponding equations for each $i \in S$.

Next, let us consider the case when $x \in B(f)$ but may not be an extreme base. Since, every vector in the base polyhedron can be expressed as a convex combination of extreme bases, we have $x(S) = \sum_i \lambda_i b_i(S)$ where b_i is an extreme base of $B(f)$. Using the earlier proof for extreme bases, there exist $q_{\mathsf{c},i} \in B(f_{\mathsf{c}})$ such that $b_i(S) = \sum_{\mathsf{c}} q_{\mathsf{c},i}(\mathsf{c} \cap S)$. This means that $x(S) = \sum_i \lambda_i \sum_{\mathsf{c}} q_{\mathsf{c},i}(\mathsf{c} \cap S) = \sum_{\mathsf{c}} \sum_i \lambda_i q_{\mathsf{c},i}(\mathsf{c} \cap S)$. The inner sum is a convex combination of extreme bases $q_{\mathsf{c},i}$ and hence, lies inside the base polyhedron $B(f_{\mathsf{c}})$. Let $y_{\mathsf{c}} = \sum_i \lambda_i q_{\mathsf{c},i}(\mathsf{c} \cap S)$. Therefore, $x(S) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap S)$ where $y_{\mathsf{c}} \in B(f_{\mathsf{c}})$. Hence, proved. □

3.2. Algorithm

Our goal in this section is to devise a strategy for minimizing $\|x\|^2$ subject to $x \in B(f)$. From lemma 3.2, every vector $x \in B(f)$ can be expressed as a sum $x(S) = \sum_{\mathsf{c}} y_{\mathsf{c}}(\mathsf{c} \cap S)$, where each $y_{\mathsf{c}} \in B(f_{\mathsf{c}})$. In other words, every vector x can be written as a sum of vector y_{c}'s which lie on respective base polyhedrons. Therefore, we can devise a strategy for minimizing $\|x\|^2$ by applying block coordinate descent where the blocks are represented by the variables y_{c}'s as defined above.

Let x_{c} denote the restriction of the base vector x to clique c. Further, let $\bar{x}_{\mathsf{c}}$ denote the restriction of x to variables $\{v \in V \mid v \notin \mathsf{c}\}$. By definition, $\|x\|^2 = \|x_{\mathsf{c}}\|^2 + \|\bar{x}_{\mathsf{c}}\|^2$. Further, it is clear that y_{c} contributes to the to vector x only through x_{c}. Therefore, when trying to minimize $\|x\|^2$ with respect to the block y_{c}, we can simply focus on the component x_{c}. Let a_{c} denote the contribution of cliques other than c to the component x_{c}. Since, we are doing block coordinate descent over variables in y_{c}, a_{c} can be treated as a constant. Further, noting that $y_{\mathsf{c}} = \sum_j \lambda_j q_{\mathsf{c},j}$, we have:

$$x_{\mathsf{c}} = y_{\mathsf{c}} + a_{\mathsf{c}} = \sum_j \lambda_{\mathsf{c},j} q_{\mathsf{c},j} + a_{\mathsf{c}} = \sum_j \lambda_{\mathsf{c},j} \left(q_{\mathsf{c},j} + a_{\mathsf{c}} \right)$$

The last equality follows from the fact that $\sum_j \lambda_{\mathsf{c},j} q_{\mathsf{c},j}$ is a convex combination and hence, $\sum_j \lambda_{\mathsf{c},j} = 1$. Therefore, in minimizing $\|x_{\mathsf{c}}\|^2$ with respect to the variables in the block y_{c}, we are looking for a convex combination of the points of the form $p = q_{\mathsf{c},j} + a_{\mathsf{c}}$ where each $q_{\mathsf{c},j} \in B(f_{\mathsf{c}})$ is an extreme base of $B(f_{\mathsf{c}})$ and a_{c} is a constant. Hence, we can use Wolfe's algorithm (see Section 2.2) for carrying out this minimization. The key question is how to efficiently carry out the step 2 of the algorithm. It is easy to see that:

$$\operatorname*{arg\,min}_{(q+a_{\mathsf{c}}):q \in B(f_{\mathsf{c}})} \langle \hat{x}, (q + a_{\mathsf{c}}) \rangle = a_{\mathsf{c}} + \operatorname*{arg\,min}_{q \in B(f_{\mathsf{c}})} \langle \hat{x}, q \rangle. \quad (2)$$

Equation (2) suggests we use Edmond's algorithm (Procedure 2) to select the extreme base $\hat{q}$ which minimizes the RHS above, followed by a translation $\hat{q}$ with a_{c}. Other steps of the Wolfe's algorithm can remain as is.

Procedure 3 describes our algorithm for finding the min norm point. This in turn requires Procedure 4 which minimizes $\|x_{\mathsf{c}}\|^2$ with respect to the variables in the block y_{c}. We ensure that $y_{\mathsf{c}} \in B(f_{\mathsf{c}})$ during each minimization. Every time $\|x_{\mathsf{c}}\|^2$ is minimized, the set of extreme bases S_{c}

Procedure 3 Min Norm Point Algorithm for Sum of Submodular Functions

```
Input: $\{f_c\}$ such that $f = \sum f_c$
Output: $x = \arg\min \|x\|^2$ subject to $x \in B(f)$.

  # Initialize
 1: for all ($c \in \mathcal{C}$) do
 2:     $q_c \leftarrow$ Take any extreme base of $f_c$;
 3:     $S_c := \{q_c\}$;
 4:     $y_c := q_c$
 5: end for
 6: $x := \sum_c y_c$;

  # Perform Block Coordinate Descent with blocks specified by Cliques
 7: while ($\|x\|$ decreases by more than $\delta$) do
 8:     for all ($c \in \mathcal{C}$) do
 9:         MinNormOverAClique($f_c$,$S_c$,$x_c$,$y_c$)    # Proc. (4)
10:     end for
11: end while
```

involved in the minimiation (see Procedure 4) is stored and used to initialize the future iteration minimizing x_c. This is important so that we do not waste computations done during the previous minimization steps. We note that the coordinate descent updates for non-overlapping blocks (i.e., y_c's) can be done in parallel and exploiting this is a direction for future work.

3.3. Convergence

Since we are optimizing a convex differentiable function ($\|x\|^2$) over a bounded convex set (i.e., $x \in (B(f))$), block coordinate descent is guaranteed to converge to the minima of the function [2]. In each coordinate descent step, we optimize the function over a subset of variables (x_c) using Wolfe's algorithm. For each descent step, we can obtain an approximate solution satisfying $x_c \leq \|x_c^*\| + \epsilon$ in $O(|c|^2 F_c/\epsilon^2)$ number of iterations. Here, $F_c = \max_{U,v} |f_c(U \cup \{v\}) - f(U)|$, where $Y \subseteq c, v \in c$ (bounds are based on Chakrabarty et al. [4]; see Section 2). Existing work [2] provides convergence bounds for block coordinate descent when the blocks of variables are non-overlapping with each other. In our case, c's can be overlapping and extending the convergence bounds for our context is a direction for future work.

4. Experiments

In this section, we report our experiments comparing our proposed SoS MinNorm algorithm with the state of the art. The implementation of our algorithm in C++, is available at http://www.iiitd.edu.in/~ishants/sosminnorm.html. All the tests have

Procedure 4 MinNormOverAClique

```
Input: Clique function: $f_c$
Input: Set of extreme bases selected in last iteration: $S_c$
Input: Restriction of current solution vector $x$ on c: $x_c$
Input: Current clique vector: $y_c$
Output: Clique vector $y_c^* \in B(f_c)$ minimizing $\|x_c\|^2$
Output: Updated set $S_c^*$ of extreme bases

 1: while (TRUE) do
 2:     Find new translation $a_c := x_c - y_c$;
 3:     Find extreme base $\hat{q}_c := \arg\min_{q_c \in B_{f_c}} \langle x_c, q_c \rangle$ using Edmond's algorithm as given in Procedure (2)
 4:     Find translated extreme base $\hat{p}_c = \hat{q}_c + a_c$.
  #     Same as step 2 in Proc. 1
 5:     if ($\|x_c\|^2 \leq \langle x_c, \hat{p} \rangle + \epsilon$) then
 6:         break;
 7:     end if
 8:     $S_c := S_c \cup \hat{q}_c$;
 9:     $P_c = \{\hat{q_c} + a_c | q_c \in S_c\}$
10:     Find $x_c$ in affine hull of $P_c$ similar to Proc. 1 step 3
11:     If $x_c$ is not in convex hull $P_c$, translate to nearest point in convex hull and update $S_c$ similar to Proc. 1 step 4.
12: end while
```

been performed on a standard workstation with 3.0 GHz CPU and 8 GB RAM running Ubuntu 14.04.

4.1. Experimental Setup

Using the terminology in the standard vision literature, we refer to our problems as 0/1 labeling problems. Each pixel can be assigned a value of 0 or 1 corresponding to exclusion and inclusion in a set, respectively. The total labeling cost is defined as the sum of labeling costs over pixels appearing in each clique. Labeling costs for a clique are referred to as clique potentials which directly correspond to the functions f_c in the sum we want to minimize. Therefore, if the clique potentials are submodular, then the problem of finding the minimum cost labeling configuration can be expressed as an SoS minimization problem.

We compare with the following algorithms:

1. Standard MinNorm point (MinNorm) which does not use sum of submodular property. We did not find any public implementation of the MinNorm and have used or own implementation in C++ for comparison.

2. Generic Cuts (GC) [1]: a flow based approach exploiting sum of submodular structure. We have used the implementation available on authors' website.

3. Jegelka et al. [20] approach using decomposition strategy. This algorithm is restricted to a subclass of sub-

	Problem Size				
	16	36	64	100	144
SoS-MinNorm	0.000	0.001	0.003	0.005	0.006
MinNorm [12]	0.002	0.023	0.073	0.221	0.776

Table 1: Comparing performance of SoS-MinNorm with the vanilla Min Norm Point algorithm [12]. We keep clique size = 2×2 with edge based costs for this experiment. The numbers denote the time taken in seconds. The proposed algorithm clearly outperforms the vanilla appproach.

modular functions for which $\min_{S \subseteq V} f(S) - a(S)$ can be computed efficiently, where $a \in R^V$ is a constant vector. The assumption is exploited in an inner loop of their algorithm. Since we would like to experiment with a general class of submodular functions, we have replaced this step dealing with specialized functions with a more general QP solver routine from cvxopt library [6].

4. For object detection experiments we have used the TextonBoost [42] approach to generate per pixel confidence. To generate simple baseline we use confidence directly for prediction (without any MRF structure), which we refer to as output from TextonBoost.

5. We also compare with pairwise cliques formulation using QPBO [25, 31] for the inference.

The proposed algorithm as well as the compared algorithm [1, 20] give optimal inference for submodular functions tested in this paper. The focus of the experiments in this paper is therefore on scalability and efficiency.

Fix et al.[9] have suggested an algorithm to learn submodular functions for the inference problems they have considered. In our experiments the approach fails to scale beyond clique size 9. Therefore we have used simple hand tuned clique potentials. It may be noted that the clique potentials are not the focus of this paper and are merely used as a proxy for real world potentials normally seen in computer vision problems. One of the future directions of our research is to use our algorithm to learn submodular clique potentials for large cliques.

Given a clique C of size $k_w \times k_h$, we consider the following potentials:

- **Edge Based Costs:** As described by Arora et al. [1], we generate a submodular potential over a 2×2 clique c by defining $f_{\mathsf{c}}(S \cap \mathsf{c})$ as the square root of the number of edges where an edge is a pair of neighboring vertices (top,down,left and right nodes) assigned different labels. In order to generate functions of size greater than 4, we translate the template in a non-overlapping fashion and add the costs from various templates.

	Clique Size			
	2×2	4×2	4×4	4×6
SoS-MinNorm	0.01	0.01	0.02	0.02
GC [1]	0.00	0.26	731.67	DNR
SoS-Jegelka [20]	861.07	12217.37	TO	TO

Table 2: Comparing SoS-MinNorm with GC [1] and SoS-Jegelka [20] on varying cliques using edge based potential. The problem size was fixed at 400. The numbers denote the time taken in seconds. DNR shows that the algorithm crashed on the test. TO denotes a time out for decomposition approach after 4 hours of running. SoS-MinNorm significantly outperforms both the existing algorithms, none of which can scale beyond clique size 16.

	Problem Size			
	100	400	900	1600
SoS-MinNorm	0.01	0.01	0.04	0.09
GC [1]	36	467	2744	2868

Table 3: Comparing SoS-MinNorm with GC [1] for varying problem sizes. Clique size is fixed at 16. We use edge based potentials for the experiment. Our approach significantly outperforms both the other approaches at all the problem sizes. SoS-Jegelka [20] had a timeout (time more than 4 hours) for all values.

- **Count Based Costs:** These potentials are inspired by the ones used by Stobbe and Krause [44]. We define a submodular potential over c as $f_{\mathsf{c}}(S) = |S \cap \mathsf{c}||\mathsf{c} \setminus S|$. For a fixed c, $f_{\mathsf{c}}(S)$ is a concave function of the number of pixels in c labeled 1. Uniform labeling is favored while equal number of pixels with $1's$ and $0's$ are penalized the most.

As is the standard for several formulations, we also incorporated additional per pixel costs, called the unary potentials. The overall function can be defined as follows:

$$f(S) = \sum_{v_i \in V} w_i \mathcal{I}[v_i \in S] + \alpha \sum_{\mathsf{c} \in \mathcal{C}} f_{\mathsf{c}}(\mathsf{c} \cap S).$$

Unary potentials can be seen as encoding pixel-wise evidence, whereas clique potentials represent labeling priors. Note that unary costs can equivalently be absorbed in the cost of a clique, and hence, do not lead to any additional complications in the model.

4.2. Experiments on Synthetic Problems

We perform our evaluations in two parts. In the first part, we experiment with synthetically generated submodular potentials. Our synthetic potentials are inspired by those used in real world vision applications. Synthetic data allows us to test the scalability of our approach with varying clique as well image sizes. Synthetic problems are generated over

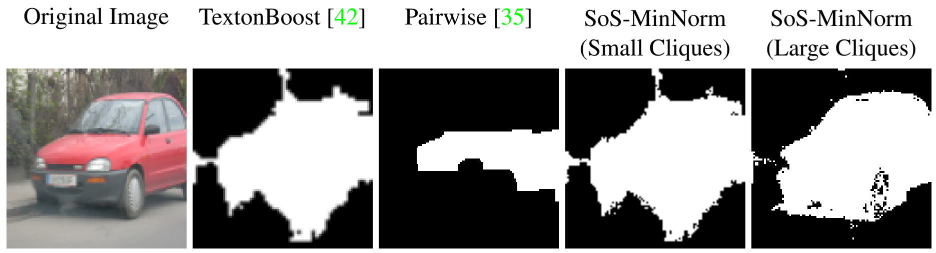

Figure 1: Pixel level object detection: We generate per pixel confidence using the probabilities generated by TextonBoost [42]. The second column titled TextonBoost has been generated based upon these confidence alone. The third column, shows the results using cliques of size of 2 only [35]. For generating higher order cliques to be used with our algorithm, we use region growing as suggested by Stobbe and Krause [44]. The fourth column shows the results using our algorithm with region growing restricted to 50 resulting in average clique size of 31. For testing with larger cliques, we allowed region growing until size 300 generating cliques with average size of 230. We used count based cost. The image size is 100 $\times$ 100 and the time for small and large cliques is 0.6 and 53 seconds respectively. The quality of results seems to improve with increasing clique size.

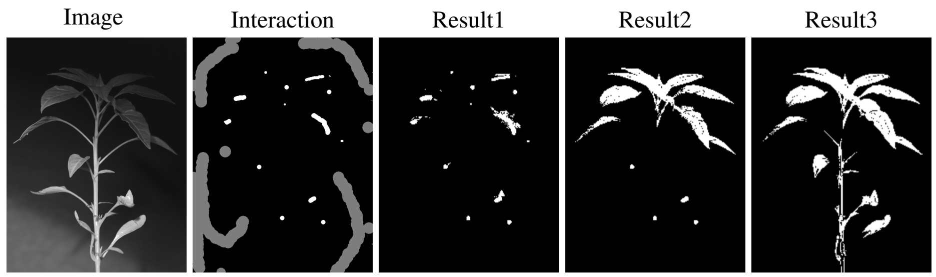

Figure 2: Interactive object segmentation: The unary cost for the inference is based upon user interaction, whereas the higher order cliques have been generated using the region growing as is done for the object detection problem. First and second columns show the input image and user inputs respectively, whereas columns third to fifth show the results using our approach with increasing average clique size. The image size is 103 $\times$ 132 and the time for the 3 results shown are 1.15,0.47 and 0.11 seconds respectively. Similar to the object detection problem, our experiments show improved visual quality with increasing clique size.

grid graphs of size $n = n_w \times n_h$. Cliques represent sub-grids of size $k = k_w \times k_h$. We vary n and k in our experiments. Unary costs have been generated randomly.

Table 1 compares SoS-MinNorm with vanilla Min Norm Point algorithm [12]. Clearly, we significantly outperform the vanilla approach by virtue of exploiting the sum of submodular property.

Next, we compare our approach with GC [1] and SoS-Jegelka [20]. Both the algorithms are state of the art and exploit the sum of submodular property of the underlying function. Table 2 shows the comparison results. Our proposed algorithm clearly outperforms SoS-Jegelka which did not scale well in our experiments. GC required few hours to solve the problems with clique size 16. Our algorithm could solve these problems in less than a minute. Further, the proposed algorithm can easily scale to problems of clique size 32 whereas GC can not go beyond clique size 16.

We also compared our algorithm with GC and SoS-Jegelka using different problem sizes. We fixed the clique size at $k = 16$ (maximum possible to which GC can scale). The image size was varied from 100 to 1600. Table 3 shows the results. Our approach is at least an order of magnitude faster than GC at all problem sizes. SoS-Jegelka timed out (process killed externally after 4 hours) at this clique size for all problem sizes.

4.3. Comparison on Real Datasets

The contribution of this paper is essentially algorithmic in nature and our algorithm can be used for any problem formulated as binary MRF-MAP or structured prediction problem with sum of submodular structure. However, we have done some indicative experiments with pixel level object detection and interactive object segmentation problems to show the efficacy of our approach on real datasets.

We experimented with pixel level object detection using the dataset provided by [32]. We generate per pixel confidence using the probabilities generated by TextonBoost [42]. We use these probabilities for setting unary potentials in our formulation. We compare with the formulation using unary cost alone (thresholding) and pairwise cliques approach [35]. We experimented with multiple values for the relative weighing of clique potential for the pairwise approach and chose the one which gave the best visual results. For generating higher order cliques to be used with

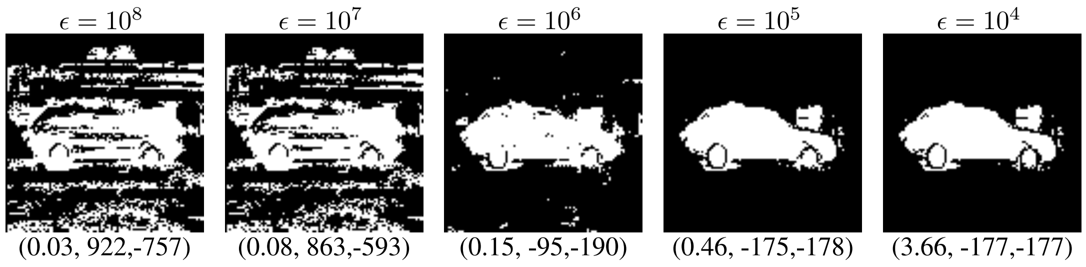

Figure 3: Our algorithm can be used for faster approximate inference by changing the value of ϵ in Step 5 Procedure 4. The number below the each figure shows time taken (in seconds), followed by primal and dual values (in thousands). We have used count based clique potential with clique size ~ 250. The approximation strategy should be useful for applications with limited time budget.

our algorithm, we first grow regions using HSV channels as suggested by Stobbe and Krause [44]. The image was divided into 5×5 grids and each grid intersection was taken as a seed for region growing. To cover any remaining regions, we then generate 50 random seeds and grow regions from them. Any seed appearing in already grown region was ignored. To show the improvement using higher order clique we use cliques of two different sizes. For generating smaller cliques, the region growing was restricted to 50 resulting in average clique size of 31. For larger cliques, region growing was allowed until 300, generating cliques with average size of 230. We use count based cost for this experiment. Figure 1 compares the results of TextonBoost, Pairwise Cliques and our algorithm using small and large cliques. Not only, our algorithm can scale to such large sized cliques, the quality of results seems to improve with increasing clique size.

Next we have experimented with interactive object segmentation as used by Jegelka et al.. [20]. The setup resembles the method proposed by Rother et al.[40]. From inference perspective, we generate the higher order cliques in the same way as described for object detection. However, the unary costs are now based upon the user interaction. Figure 2 shows the result. Similar to object detection, we see an improvement with increasing clique size with our method.

4.4. Approximation Strategy

Our algorithm is guaranteed to converge to the optimum for all submodular functions. However, the convergence may be slow for large problems. The algorithm can be potentially used as an approximation algorithm also. For optimal inference ϵ in Step 5 Procedure 4 should be a very small value. Increasing the value of ϵ can lead to faster convergence but may cause algorithm to terminate before optimality is reached. We have experimented with various values of ϵ for the pixel level object detection experiment. Figure 3 shows the results. As expected the time taken for the inference improves as we increase the value of ϵ. Correspondingly, we observe increasing primal dual gap with increasing values of ϵ. Interestingly, the visual quality of results also degrades gradually. This indicates the possibility of using ϵ as a tunable parameter for controlling quality vs time taken in problems with limited time budget.

5. Conclusion

Many problems in computer vision modelled as MRF-MAP labeling problems can be reduced to minimizing a sum of submodular functions. The state of the art algorithms suggested in computer vision scale well with image size but not with clique size. On the other the algorithms proposed in mathematical optimization community scales well with clique size but not with image size. In this paper we have tried to take the best of both the worlds. We suggest a new algorithm which adapts Min Norm Point algorithm for minimizing a sum of submodular functions. Being based upon min norm, the algorithm scales well with clique size, whereas by exploiting sum of submodular structure, the algorithm works well with large problem sizes also. In our experiments, we have run it for inference problems with number of nodes running into many thousands and clique size of multiple hundreds. The algorithm achieves state of the art accuracy both in terms of efficiency (time taken for the inference) as well as scalability (with image and clique size).

Recent research in computer vision has shown the potential of the MRF-MAP formulation using submodular functions learnt from the training samples [9]. The techniques do not scale to large clique sizes. Learning higher order clique potentials is an area of our future research.

Acknowledgement

We would like to thank anonymous reviewers for their useful comments and suggestions towards improving some of the theoretical results presented in the paper.

References

[1] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari. Generalized flows for optimal inference in higher order MRF-MAP.

IEEE TPAMI, 37(7):1323–1335, 2015. 2, 5, 6, 7

[2] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013. 5

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001. 1

[4] D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using wolfe's algorithm. In *Proc. of NIPS*, pages 802–809, 2014. 3, 5

[5] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *Proc. of CVPR*, pages 53–60, 2006. 1

[6] J. Dahl and L. Vandenberghe. Cvxopt, 2007. http://mloss.org/software/view/34/. 6

[7] A. Ene and H. L. Nguyen. Random coordinate descent methods for minimizing decomposable submodular functions. *arXiv preprint arXiv:1502.02643*, 2015. 1, 2

[8] A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for higher-order markov random fields. In *Proc. of ICCV*, pages 1020–1027, 2011. 1

[9] A. Fix, T. Joachims, S. M. Park, and R. Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Proc. of ICCV*, pages 3104–3111, 2013. 6, 8

[10] A. Fix, C. Wang, and R. Zabih. A primal-dual algorithm for higher-order multilabel markov random fields. In *Proc. of CVPR*, pages 1138–1145, 2014. 1

[11] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proc. of CVPR*, pages 939–946, 2005. 1

[12] S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7:3–17, 2011. 2, 3, 6, 7

[13] A. C. Gallagher, D. Batra, and D. Parikh. Inference for order reduction in markov random fields. In *Proc. of CVPR*, pages 1857–1864, 2011. 1

[14] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE TPAMI*, 6(6):721–741, 1984. 1

[15] T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *Information Theory*, 56(12):6294–6316, 2010. 1

[16] R. J. ian and T. S. Huang. Object detection using hierarchical MRF and MAP estimation. In *Proc. of CVPR*, pages 186–192, 1997. 1

[17] H. Ishikawa. Transformation of general binary MRF minimization to the first-order case. *IEEE TPAMI*, 33(6):1234–1249, 2011. 1

[18] H. Ishikawa. Higher-order clique reduction without auxiliary variables. In *Proc. of CVPR*, pages 1362–1369, 2014. 1

[19] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001. 2

[20] S. Jegelka, F. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. In *Proc. of NIPS*, pages 1313–1321, 2013. 1, 2, 5, 6, 7, 8

[21] F. Kahl and P. Strandmark. Generalized roof duality for pseudo-boolean optimization. In *Proc. of ICCV*, pages 255–262, 2011. 1

[22] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009. 1

[23] V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(15):2246–2258, 2012. 2

[24] V. Kolmogorov. A new look at reweighted message passing. *IEEE TPAMI*, 37(5):919–930, 2015. 1

[25] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE TPAMI*, 29(7):1274–1279, 2007. 2, 6

[26] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 26(2):147–159, 2004. 1

[27] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proc. of CVPR*, pages 2985–2992, 2009. 1

[28] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *Proc. of SIGGRAPH*, pages 277–286, 2003. 1

[29] Y. T. Lee, A. Sidford, and S. C.-w. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. *arXiv preprint arXiv:1508.04874*, 2015. 2

[30] U. Mudenagudi, S. Banerjee, and P. K. Kalra. Space-time super-resolution using graph-cut optimization. *IEEE TPAMI*, 33(5):995–1008, 2011. 1

[31] A. C. Müller and S. Behnke. pystruct - learning structured prediction in python. *Journal of Machine Learning Research*, 15:2055–2060, 2014. 6

[32] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE TPAMI*, 28(3):416–431, 2006. 7

[33] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009. 2, 3

[34] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014. 1

[35] W. Pieczynski and A.-N. Tebbache. Pairwise markov random fields and segmentation of textured images. *Machine Graphics and Vision*, 9(3):705–718, 2000. 7

[36] B. Potetz and T. S. Lee. Efficient belief propagation for higher-order cliques using linear constraint nodes. *CVIU*, 112(1):39–54, Oct. 2008. 1

[37] S. Ramalingam, C. Russell, L. Ladicky, and P. H. Torr. Efficient minimization of higher order submodular functions using monotonic boolean functions. *arXiv preprint arXiv:1109.2304*, 2011. 1, 2

[38] S. Roth and M. J. Black. Fields of experts. *IJCV*, 82(2):205–229, 2009. 1

[39] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proc. of CVPR*, pages 1382–1389, 2009. 1, 2

[40] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004. 1, 8

[41] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000. 2

[42] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. of ECCV*, pages 1–15, 2006. 6, 7

[43] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254, 2011. 1

[44] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Proc. of NIPS*, pages 2208–2216, 2010. 2, 6, 7, 8

[45] D. Tarlow, I. E. Givoni, and R. S. Zemel. Hop-map: Efficient message passing with high order potentials. In *Proc. of AISTATS*, 2010. 1

[46] P. Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11, 1976. 3

[47] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *Proc. of CVPR*, pages 1–8, 2008. 1