PGMO Lecture: Vision, Learning and Optimization

7. Dynamic programming

Thomas Pock

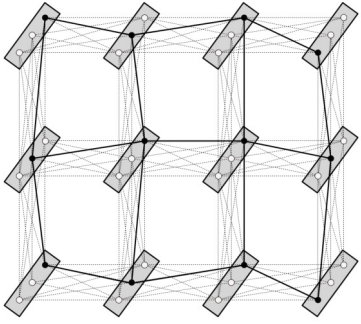
Institute of Computer Graphics and Vision

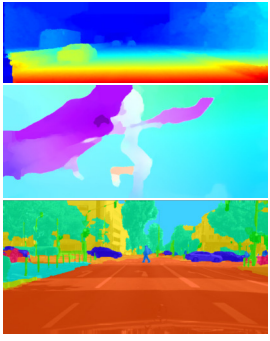
April 3, 2020

Overview

Image labeling

- Many problems in image processing /computer vision can be cast as graph labeling problems [Savchynskyy '19]
- Nodes $i \in \mathcal{V}$ correspond image pixels
- Edges $(i, j) \in \mathcal{E}$ define a neighborhood system
- Each node i can take a label $y_i \in \mathcal{Y} = \{1, 2, ..., K\}$
- Assign an energy to a certain configuration of the labels $y = (y_i)_{i \in \mathcal{V}}$.
- Task: Find the configuration with the lowest energy





Stereo

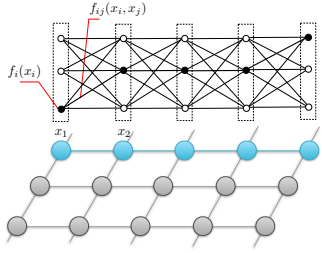
Motion

Segmentation

The energy

- Discrete optimization models can efficiently impose a smoothness prior.
- We consider the following classical labeling model:

$$\min_{y \in \mathcal{Y}^{|\mathcal{V}|}} \left\{ E(y|\theta) := \sum_{i \in \mathcal{V}} \theta_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{i,j}(y_i, y_j) \right\}.$$



- The model depends on unary terms θ_i as well as binary terms $\theta_{i,j}$.

Some state-of-the-art algorithms

- The primal problem can be solved by performing a standard LP relaxation [Schlesinger '76] which in turn can be solved by any large scale LP solver
- Max product believe propagation (BP) [Pearl '88] is exact on trees. For graphs with loops it often yields inferior solutions and it also may not converge
- Move making algorithms based on graph cuts [Boykov, Veksler, Zabih '01], [Komodakis, Tziritas '07]. Solves a sequence of binary problems and hence it may not scale so well with the number of labels
- Tree reweighted message passing (TRW) [Wainwright, Jaakkola, Willsky '05] decomposes graph into trees. BP is used as a subroutine on the trees. It works very well but it may not converge
- TRW-S [Kolmogorov '06] is a sequential version of TRW which monotonically improves the dual energy. Can be seen as a monotonous coordiante ascent on the dual with an efficient sequential implementation

Lifting

- The image labeling problem can be re-cast as a binary optimization problem by means of lifting.
- Lift labels y_i to vectors $x_i = 1_{y_i}$.
- For example for $K = 5$, $y_i = 3$, one has $x_i = (0, 0, 1, 0, 0)$
- The image labeling problem becomes

$$\min_x \left\{ E(x, f) := \sum_{i \in \mathcal{V}} \theta_i^T x_i + \sum_{(i,j) \in \mathcal{E}} \mathrm{tr}(\theta_{i,j}^T (x_i \otimes x_j)) \right\}.$$

- Note that in the lifted representation, the function is linear in the unaries and binaries.

Schlesinger's LP relaxation

- Replace binary variables x_i and $x_i \otimes x_j$ by v_i and $w_{i,j}$.
- A classical LP relaxation is due to [Schlesinger '76]

$$\begin{aligned} \min_{v,w} \quad & \sum_{i \in \mathcal{V}} \theta_i^T v_i \; + \sum_{(i,j) \in \mathcal{E}} \operatorname{tr}(\theta_{i,j}^T w_{i,j}), \\ \text{s.t.} \quad & v_i^T 1 = 1, \quad v_i^l \geq 0, \\ & w_{i,j}^T 1 = v_j, \; w_{i,j} 1 = v_i, \quad w_{i,j} \geq 0. \end{aligned}$$

- Gives favorable integrality gap guarantees [Chekuri et al '04].
- Can be cast as a huge $\mathcal{O}(N^2 K^2)$ LP in standard form
- Example: Image size $N \times N = 1024^2$, $K = 256$ labels, $\mathcal{O}(N^2 K^2) \sim 7 \cdot 10^{10}$ variables.
- Important observation: The dual problem is much smaller, only $\mathcal{O}(N^2 K) \sim 3 \cdot 10^8$ variables.

Markov random fields

- Markov random fields (MRFs) are undirected graphical models that have a Markov property, that is every node only depends on its neighbors.
- The energy of the image labeling problems can be interpreted as a negative log posterior distribution where the unary terms correspond to the data likelihood and the pairwise terms correspond to the prior.

$$p(y|\theta) = \frac{1}{Z} \exp\left(-\frac{E(y|\theta)}{T}\right),$$

where Z is the partition function and $T > 0$ is the temperature (variance) of the distribution.

- One can consider different methods for inference:
- MAP estimation corresponds to the minimization of $E(y|\theta)$.
- Computation of the marginals distributions $p(y_i|\theta)$,

$$p(y_i|\theta) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_n} p(y|\theta)$$

which in turn can be used to compute the expectation (minimum mean squared estimate).

- We will show that both problems can be easily computed on tree-like graphs and well approximated (in practice) on grid-like graphs using dynamic programming.

Overview

Solving labeling problems on a chain

- Let us restrict our image labeling problem to one line (chain) of the image:
- On this chain, we consider a graph of n nodes $\mathcal{V} = \{1, 2, ..., n\}$, representing the image pixels.
- The edge set is given by pairs of neighboring nodes along that line, that is $\mathcal{E} = \{(i, i+1) : \ i = 1, ..., n-1\}$.
- Each node $i \in \mathcal{V}$ can take a label out of a given label set $\mathcal{Y}$.

$$\min_{y_1,\dots,y_n} E(y_1, ..., y_n) := \sum_{i=1}^{n} \theta_i(y_i) + \sum_{i=1}^{n-1} \theta_{i,i+1}(y_i, y_{i+1}).$$

Main observation

- The pairwise structure of the the energy on a chain admits the following recursive definition.

$$E(y_1, ..., y_n) = \min_{y_1,\dots,y_n} \sum_{i=1}^{n-1} \theta_i(y_i) + \sum_{i=1}^{n-1} \theta_{i,i+1}(y_i, y_{i+1}) + \theta_n(y_n)$$

$$= \min_{y_2,\dots,y_n} \left(\underbrace{\min_{y_1} \theta_1(y_1) + \theta_{1,2}(y_1, y_2)}_{F_2(y_2)} \right) + \sum_{i=2}^{n-1} \theta_i(y_i) + \sum_{i=1}^{n-1} \theta_{i,i+1}(y_i, y_{i+1}) + \theta_n(y_n)$$

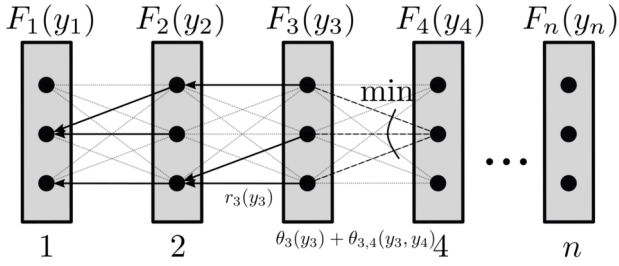
$$= \min_{y_3,\dots,y_n} \left(\underbrace{\min_{y_2} F_2(y_2) + \theta_2(y_2) + \theta_{2,3}(y_2, y_3)}_{F_3(y_3)} \right) + \sum_{i=3}^{n-1} \theta_i(y_i) + \sum_{i=1}^{n-1} \theta_{i,i+1}(y_i, y_{i+1}) + \theta_n(y_n)$$

$$= \dots = \min_{y_n} F_n(y_n) + \theta_n(y_n).$$

Every function $F_i(y_i)$ is a vector that can be computed by taking the minimum over all possible y_{i-1}.

Dynamic Programming

- The recursive definition allows for an efficient dynamic programming scheme



Source: [Savchynskyy '19]

- In the computation of F_n, it turns out that it is convenient to define the functions $F_i : \mathcal{Y} \to \mathbb{R}$:

$$F_1(s) = 0, \quad F_i(s) := \min_{t \in \mathcal{Y}} F_{i-1}(t) + \theta_{i-1}(t) + \theta_{i-1,i}(t, s),$$

which are the so-called forward messages or Bellman functions.

Recursive expression of the energy

- Assume we have computed a function $F_n : \mathcal{Y} \to \mathbb{R}$, such that $F_n(s) + \theta_n(s)$ denotes the minimal cost of E with $s \in \mathcal{Y}$ being the label of the very last node.
- The minimal energy of the overall labeling can be computed as

$$\min_{y_1,\dots y_n} E(y_1, ..., y_n) = \min_{s \in \mathcal{Y}} F_n(s) + \theta_n(s),$$

where the optimal labeling of the last node is given by

$$y_n = \arg\min_{s \in \mathcal{Y}} F_n(s) + \theta_n(s).$$

Dynamic programming algorithm

- Based on our observations, we can define the following dynamic programming algorithm.
- The variables $r_i(s)$ will be needed to recover the optimal labeling.

1. Set $F_1(s) = 0$ for all $y \in \mathcal{Y}$.
2. for $i = 2$ to n do
3. $\quad F_i(s) := \min_{t \in \mathcal{Y}} F_{i-1}(t) + \theta_{i-1}(t) + \theta_{i-1,i}(t, s)$, $\forall s \in \mathcal{Y}$.
4. $\quad r_i(s) := \arg\min_{t \in \mathcal{Y}} F_{i-1}(t) + \theta_{i-1}(t) + \theta_{i-1,i}(t, s)$.
5. end for
6. $E^* = \min_{s \in \mathcal{Y}} F_n(s) + \theta_n(s)$.
7. return E^*, $\{r_i(s) : i = 2, ..., n,\ s \in \mathcal{Y}\}$

Reconstruction of an optimal labeling

- Once, the forward messages have been computed, the optimal labeling can be computed via backtracking starting from the label y_n that achieved the smallest overall energy.

1. $y_n = \arg\min_{s \in \mathcal{Y}} F_n(s) + \theta_n(s)$
2. for $i = n$ to 2 do
3. $\quad y_{i-1} = r_i(y_i)$.
4. end for
5. return $y_1, ..., y_n$.

Complexity of the algorithm

- In each iteration of the dynamic programming algorithm we have to perform $\mathcal{O}(|\mathcal{Y}|)$ minimization steps, each having a complexity of $\mathcal{O}(|\mathcal{Y}|)$ (enumeration).
- Hence the complexity for computing each forward message F_i is $\mathcal{O}(|\mathcal{Y}|^2)$
- We are performing $n-1$ iterations and hence the overall complexity is $\mathcal{O}(|\mathcal{V}||\mathcal{Y}|^2)$.
- The memory to store the model (in the most general case) is also $\mathcal{O}(|\mathcal{V}||\mathcal{Y}|^2)$, and hence the algorithm is linear in the model size.

Reversed algorithm

- The same algorithm can also be implemented in the opposite direction.
- The so-called backward messages are then computed as

$$B_n(s) = 0, \quad B_i(s) = \min_{t \in \mathcal{Y}} B_{i+1}(t) + \theta_{i+1}(t) + \theta_{i,i+1}(s,t)$$

- Observe that the backward messages are computed exactly in the same way as the forward messages, but just processing the graph in reversed order.
- It will turn out that it is useful to compute both the forward and backward messages for extending the dynamic programming concept to grid graphs.

Min-marginals

- The min-marginals at a node $w \in \{1, ..., n\}$ are defined as the energy E_w at the node w which is obtained by "minimizing out" all other nodes, that is

$$E_w(s) = \min_{y \in \mathcal{Y}^{|\mathcal{V}|}, y_w = s} E(y_1, ..., y_n; \theta)$$

- This means that we are fixing the label of $y_w = s$ but choose all other labels such that they minimize the overall energy.
- Using the definitions of forward messages F_i and backward messages B_i one can easily see that the min-marginals are computed as

$$E_i(s) = F_i(s) + B_i(s) + \theta_i(s).$$

- This algorithm is referred to as forward-backward dynamic programming or forward-backward belief propagation on a chain.
- After computing all min-marginals $E_i(s)$, the optimal labels can be simply computed by picking in each node that label with the smallest energy.
- No additional backtracking is needed, as this information is contained already in the forward-backward messages.

Back to Markov random fields

- Let us examine how a similar algorithm can be developed to compute marginals.
- From the Markov property of the energy defined on the chain, we see that the posterior distribution can be written in terms of potential functions $\phi_i(y_i) = \exp(-\theta_i(y_i)/T)$ and $\psi_{i,i+1}(y_i, y_{i+1}) = \exp(-\theta_{i,i+1}(y_i, y_{i+1})/T)$.

$$p(y|\theta) = \frac{1}{Z} \prod_{i=1}^{n-1} \psi_{i,i+1}(y_i, y_{i+1}) \prod_{i=1}^{n} \phi_i(y_i),$$

  where $\psi_{i,i+1}(y_i, y_{i+1})$ can be easily identified with the pairwise terms and $\phi_i(y_i)$ can be identified with the unary terms.
- Let us examine how the true marginals can be computed using dynamic programming.

Marginals

Using the fact that $ab + ac = a(b + c)$ we can rewrite the formula for the marginals as

$$p(y_i|\theta) = \frac{1}{Z} \sum_{y_1} \sum_{y_2} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_n} p(y|\theta) =$$

$$\frac{1}{Z} \underbrace{\left[\sum_{y_{i-1}} \psi_{i-1,i}(y_{i-1}, y_i) \phi_{i-1}(y_{i-1}) \cdots \left[\sum_{y_1} \psi_{1,2}(y_1, y_2) \phi_1(y_1) \right] \right]}_{\tilde{F}_i(y_i)} \cdot \phi_i(y_i) \cdot$$

$$\underbrace{\left[\sum_{y_{i+1}} \psi_{i,i+1}(y_i, y_{i+1}) \phi_{i+1}(y_{i+1}) \cdots \left[\sum_{y_n} \psi_{n-1,n}(y_{n-1}, y_n) \phi_n(y_n) \right] \right]}_{\tilde{B}_i(y_i)},$$

where $\tilde{F}_i$ and $\tilde{B}_i$ denote the forward and backward messages.

Recursive definition

- The forward and backward messages admit a recursive definition based on dynamic programming.

$$\tilde{F}_i(y_i) = \sum_{y_{i-1}} \tilde{F}_{i-1}(y_{i-1})\phi_{i-1}(y_{i-1})\psi_{i-1,i}(y_{i-1}, y_i),$$
$$\tilde{B}_i(y_i) = \sum_{y_{i+1}} \tilde{B}_{i+1}(y_{i+1})\phi_{i+1}(y_{i+1})\psi_{i,i+1}(y_i, y_{i+1}).$$

- For numerical reasons it is advantageous to move to a log-domain, where

$$\log p(y_i|\theta) = \underbrace{\log \tilde{F}_i(y_i)}_{F_i(y_i)} + \underbrace{\log \tilde{B}_i(y_i)}_{B_i(y_i)} + \log \phi_i(y_i) - \log Z$$

- In the log-domain, the messages can be computed as

$$F_i(y_i) = \log\left(\sum_{y_{i-1}} \exp(F_{i-1}(y_{i-1}))\phi_{i-1}(y_{i-1})\psi_{i-1,i}(y_{i-1}, y_i)\right),$$
$$B_i(y_i) = \log\left(\sum_{y_{i+1}} \exp(B_{i+1}(y_{i+1}))\phi_{i+1}(y_{i+1})\psi_{i,i+1}(y_i, y_{i+1})\right).$$

Discussion

- Observe that we never need to know the partition function Z since it only defines a constant offset for the marginals which can be easily accounted for by normalizing $p(y_i|\theta)$ such that

$$\sum_{y_i} p(y_i|\theta) = 1$$

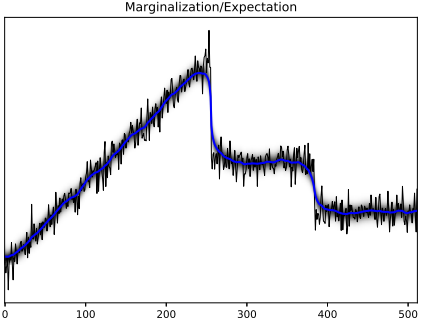
- In the recursive definition of the forward and backward messages in the log-domain, the softmax function appears

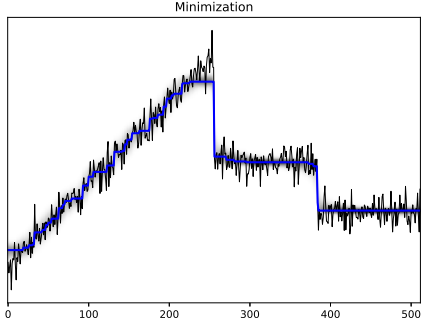
$$x \mapsto \varepsilon \log \sum_{i=1}^{n} \exp(x_i/\varepsilon) \underset{\varepsilon \to 0}{\rightsquigarrow} \max\{x_1, ..., x_n\}$$

- The same is true in the negative log-domain where the softmax function is replaced by the softmin function.

- **As a conclusion, the same dynamic programming algorithm can be used to perform both energy minimization (min) and marginalization (softmin)!**

Marginalization vs. Minimization





- The energy corresponds to the ROF model on a chain.
- The "temperature" parameter was set to $T = 0.02$ (no effect on the minimization).
- The minimization leads to the well-known staircasing effect.
- The marginalization and computation of the expectation leads to more natural results.

Fast message computation

- We have seen that for each node the complexity to compute the message updates requires $\mathcal{O}(|\mathcal{Y}|^2)$ operations.
- Assume, the label set is given by an ordered set, e.g. $\mathcal{Y} = \{0, 1, 2, 3, ..., K-1\}$ and the pairwise functions which are given by

$$\theta_{i-1,i}(y_{i-1}, y_i) = w_{i-1,i}|y_{i-1} - y_i|,$$

  where $w > 0$ is the slope of the absolute function.
- It turns out that in such a case the messages can be computed in linear, that is $\mathcal{O}(|\mathcal{Y}|)$ time.
- This can be see because the message update now becomes

$$F_i(s) = \min_{t \in \mathcal{Y}} \underbrace{F_{i-1}(t) + \theta_{i-1}(t)}_{G_{i-1}(t)} + w_{i-1,i}|t - s|,$$

  which is a so-called min-convolution.

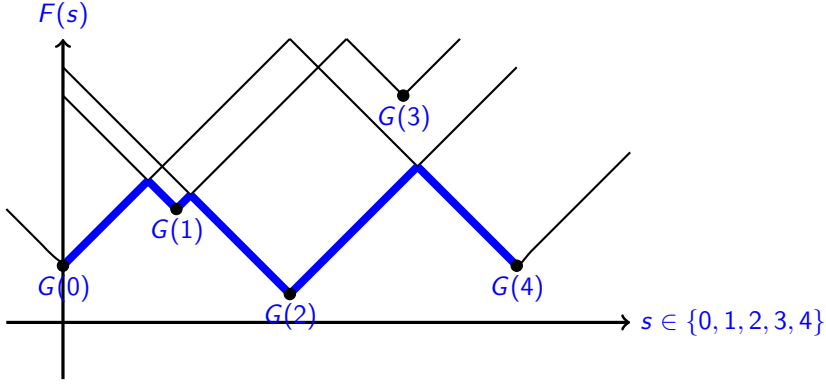
Min-convolution

- Consider the problem

$$F(s) = \min_{t \in \mathcal{Y}} G(t) + w|t - s|,$$

  where $w > 0$ is the slope of the absolute function.

- Geometrically, it amounts to compute the lower bound (blue line):



- Can be computed using a fast distance transform.

Fast distance transform

The fast distance transform can be computed in two passes:

- Initialization: $F(*) = G(*)$

- (i) Left-to-right:

1. for $k = 1$ to $K - 1$ do
2. $\quad F(k) = \min\{F(k), F(k-1) + w\}$.
3. end for

- Right-to-left:

1. for $k = K - 2$ to 0 do
2. $\quad F(k) = \min\{F(k), F(k+1) + w\}$.
3. end for

Truncated pairwise costs

- Assume that the pairwise costs have a constant value T on a subset $\mathcal{Y}_T \subset \mathcal{Y}^2$ of the set of all possible pairwise labels:

$$\theta_{i-1,i}(y_{i-1}, y_i) = T, \quad \forall (y_{i-1}, y_i) \in \mathcal{Y}_T.$$

- This usually occurs in case of truncated pairwise costs, e.g. truncated linear:

$$\theta_{i-1,i}(y_{i-1}, y_i) = \min(T, |y_{i-1} - y_i|)$$

- One can pre-compute the truncated messages $F_i^T(s)$ in $\mathcal{O}(|\mathcal{Y}|)$ time:

$$F_i^T(s) = \min_{(s,t)\in\mathcal{Y}_T} G_{i-1}(t) + \underbrace{\theta_{i-1,i}(s,t)}_{=T} = T + \min_{t\in\mathcal{Y}} G_{i-1}(t)$$

- Then, one has to solve the full message computation problem on the complement $\mathcal{Y}_T^c = \mathcal{Y}^2 \setminus \mathcal{Y}_T$:

$$F_i^{T,c}(s) = \min_{(s,t)\in\mathcal{Y}_T^c} G_i(s) + \theta_{i-1,i}(s,t),$$

  which is still quadratic, i.e. $\mathcal{O}(|\mathcal{Y}_T^c|^2)$ but which is usually much smaller compared to the full complexity $\mathcal{O}(|\mathcal{Y}|^2)$.

- Finally one takes the minimum between $F_i^T(s)$ and $F_i^{T,c}(s)$ that is

$$F_i(s) = \min\{F_i^{T,c}(s), F_i^T(s)\}.$$

Overview

Example: Disparity Estimation

- Given a rectified stereo pair, compute the disparities between the two images
- Rectified stereo pair: Corresponding image pixels lie on the same horizontal line

Example: Disparity Estimation

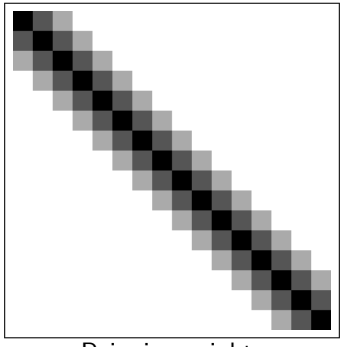
- Given a rectified stereo pair, compute the disparities between the two images
- Rectified stereo pair: Corresponding image pixels lie on the same horizontal line

Energy model

- We consider all chains (horizontal, or vertical) of the left image
- The labels $y_i \in \{0, ..., D-1\}$ correspond to possible disparity values at the pixels
- **Data Term:** The terms $\theta_i(y_i)$ are given by matching some features (SAD, NCC, Census, SIFT, learned ...)
- **Smoothness term:** Truncated linear potential of the form

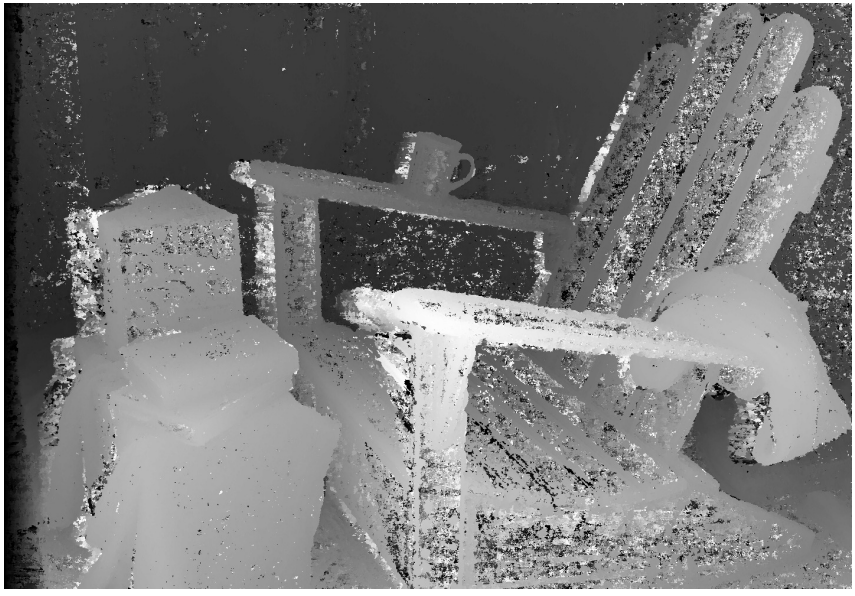
$$\theta_{i-1,i}(y_{i-1}, y_i) = \min(T, |y_{i-1} - y_i|),$$

for some value of T.



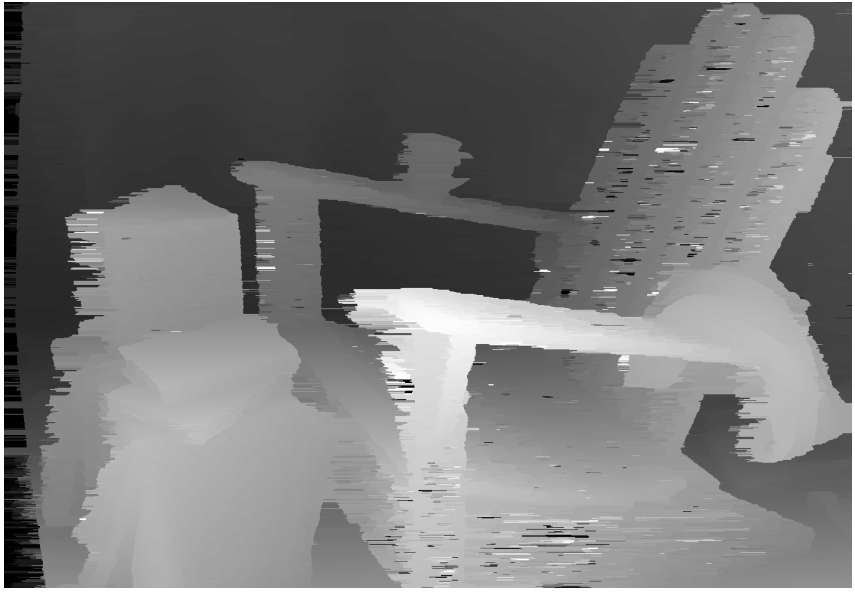
Pairwise weights

Results



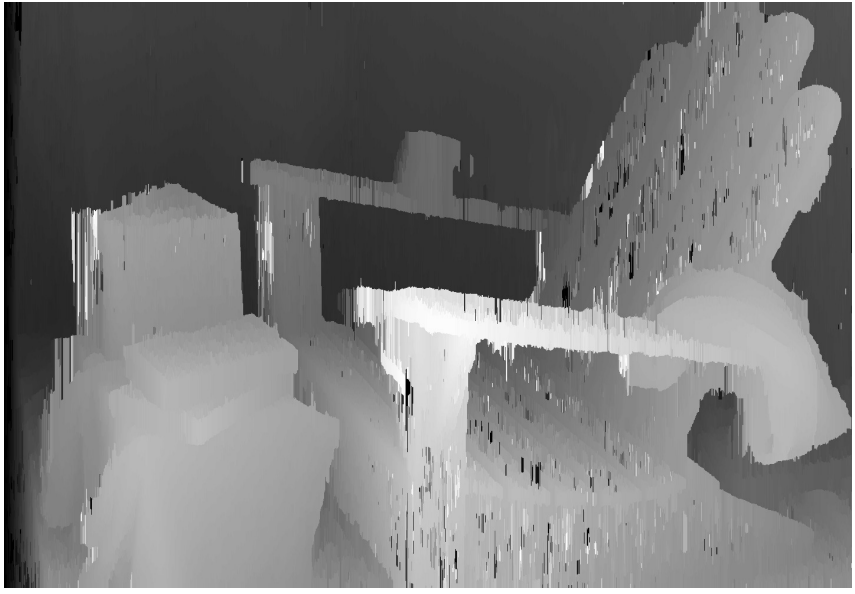
Only unary terms

Results



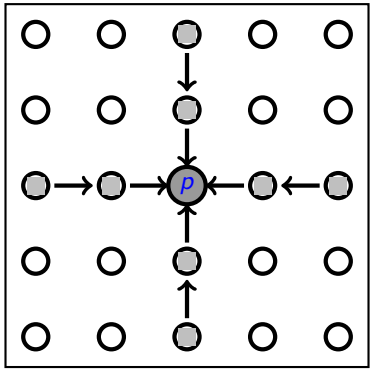
Horizontal scanlines

Results



Vertical scanlines

Semi-global matching (SGM)



- Introduced by Heiko Hirschmüller in 2008 for stereo estimation, (3800 citations).
- The basic idea is to make use of dynamic programming to compute the min-marginals on "+"-like trees graphs as shown in the picture.
- This can be done using forward-backward dynamic programming.
- Can be extended to more directions (16 in practice).
- Iteration of the algorithm leads to improved results.
- Can be easily paralellized on the GPU.

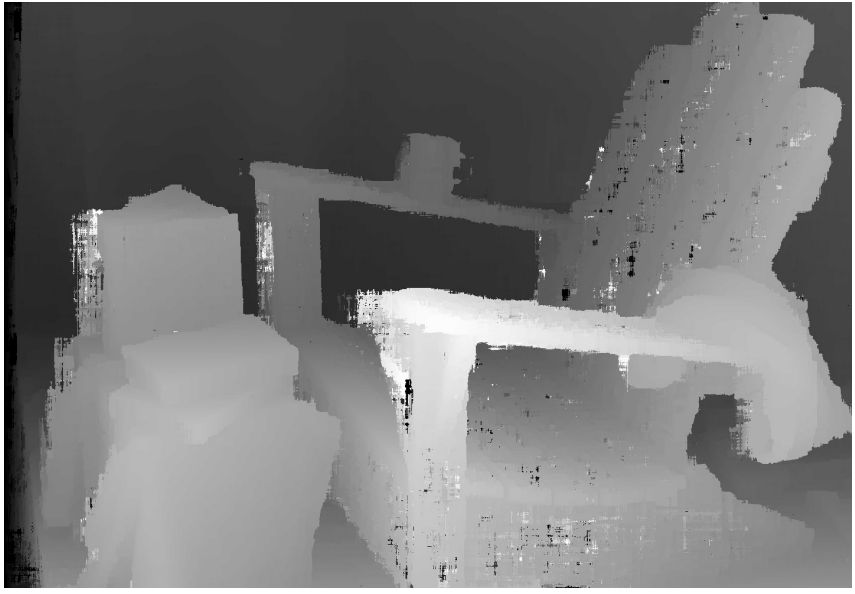
Overview

SGM

- The SGM algorithm is based on computing min-marginals composed of forward- and backward messages from both horizontal and vertical chains
- Let $F_i^h(s)$ and $B_i^h(s)$ be the forward- and backward messages of the horizontal chains and $F_i^v(s)$ and $B_i^v(s)$ be the forward- and backward messages of the vertical chains.

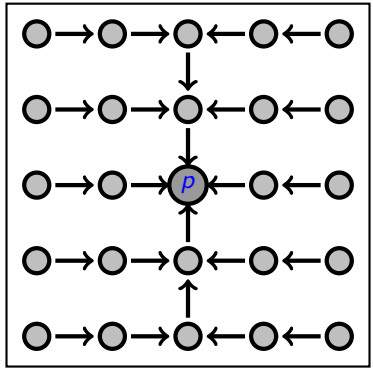
1. Compute $F_i^h(s)$ and $B_i^h(s)$.
2. Compute $F_i^v(s)$ and $B_i^v(s)$.
3. Compute min-marginals $E_i(s) = F_i^h(s) + B_i^h(s) + F_i^v(s) + B_i^v(s) + \theta_i(s)$.
4. Select optimal label: $y_i = \arg\min_s E_i(s)$.

Results



SGM

Belief propagation (BP)



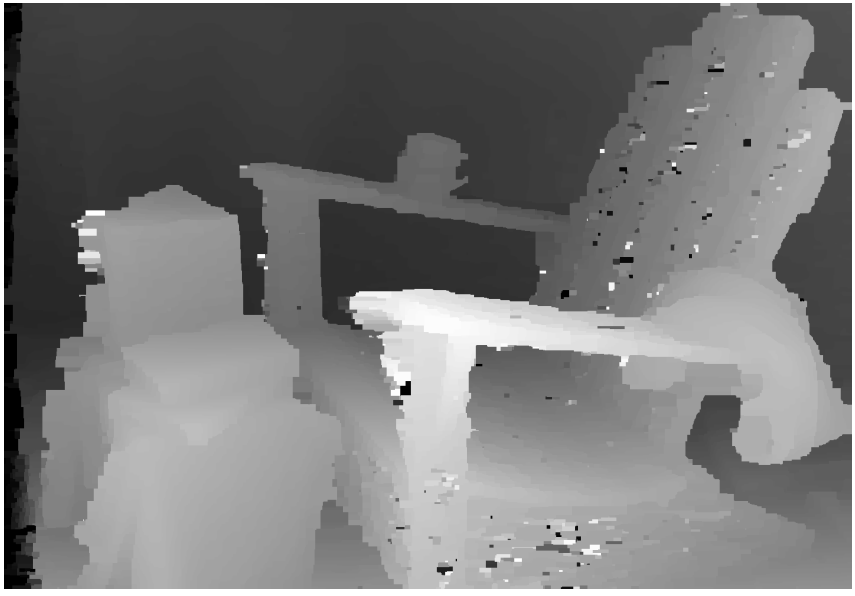
- Introduced 2003 by Tappen and Freeman to approximate the full 2D energy
- Makes use of the largest trees through a particular node.
- The min-marginals can again be computed using forward-backward dynamic programming.
- Iterative application of the algorithm by alternating the type of the tree.

BP

- Itilialize all messages $F_i^h(s), B_i^h(s), F_i^v(s), B_i^v(s)$ with zeros

1. for $it = 1$ to $maxit$ do
2. Compute $F_i^h(s)$, $B_i^h(s)$ with unaries $\tilde{\theta}_i(s) = \theta_i(s) + F_i^v(s) + B_i^v(s)$.
3. Compute $F_i^v(s)$, $B_i^v(s)$ with unaries $\tilde{\theta}_i(s) = \theta_i(s) + F_i^h(s) + B_i^h(s)$.
4. end for
5. Compute min-marginals $E_i(s) = F_i^h(s) + B_i^h(s) + F_i^v(s) + B_i^v(s) + \theta_i(s)$.
6. Select optimal label: $y_i = \arg\min_s E_i(s)$.

Result



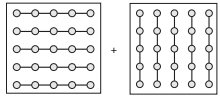
Belief propagation

Overview

Convex relaxation

- The image labeling energy (primal problem) can be naturally split into problems acting on horizontal and vertical chains

$$\min_x P(x) := f_1(x) + f_2(x),$$



(The constraints that $x_i = 1_{y_i}$ is now hidden in the functions)

- Problem is easy to minimize in f_1 or f_2 but not jointly
- Consider the following splitting:

$$\min_x f_1(x) + f_2(x) = \min_{x_1, x_2} f_1(x_1) + f_2(x_2) + \delta_=(x_1, x_2)$$

- The corresponding Fenchel (or Lagrange) dual is given by

$$\begin{aligned} D(\lambda) &= \min_{x_1, x_2} f_1(x_1) + f_2(x_2) + \lambda^T (x_1 - x_2) \\ &= \min_{x_1} \left\{ f_1(x_1) + \lambda^T x_1 \right\} + \min_{x_2} \left\{ f_2(x_2) - \lambda^T x_2 \right\} \\ &= -\left(f_1^*(-\lambda) + f_2^*(\lambda) \right) \end{aligned}$$

- The dual $D(\lambda)$ is a concave, piecewise affine function

Weak Duality

- Weak duality holds

$$\max_{\lambda} \min_{x_1,x_2} f_1(x_1) + f_2(x_2) + \lambda^T(x_1 - x_2) \leq \min_{x_1,x_2} \max_{\lambda} f_1(x_1) + f_2(x_2) + \lambda^T(x_1 - x_2)$$

- It matters who plays first!
- The right hand side is finite only in case $x_1 = x_2$, hence

$$\max_{\lambda} D(\lambda) \leq \min_{x} P(x)$$

- Strong duality "=" holds only under further assumptions such as convexity of f_1 and f_2

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\max_{\lambda} D(\lambda) \quad = \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda))$$

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\begin{aligned} \max_{\lambda} D(\lambda) &= \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda)) \\ &= \max_{\lambda} \lambda^T u - (f_1^*(-\lambda) + f_2^*(\lambda))|_{u=0} \end{aligned} \qquad f^{**}(u) = \max_{\lambda} \lambda^T u - f^*(\lambda)$$

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\begin{aligned}
\max_{\lambda} D(\lambda) &= \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda)) \\
&= \max_{\lambda} \lambda^T u - (f_1^*(-\lambda) + f_2^*(\lambda))|_{u=0} \quad f^{**}(u) = \max_{\lambda} \lambda^T u - f^*(\lambda) \\
&= (f_1^{**} \circ (-1) \,\square\, f_2^{**})(u)|_{u=0} \quad (f^* + g^*)^* = f^{**} \square g^{**} \text{ (infimal convolution)}
\end{aligned}$$

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\begin{aligned}
\max_{\lambda} D(\lambda) &= \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda)) \\
&= \max_{\lambda} \lambda^T u - (f_1^*(-\lambda) + f_2^*(\lambda))|_{u=0} \quad f^{**}(u) = \max_{\lambda} \lambda^T u - f^*(\lambda) \\
&= (f_1^{**} \circ (-1) \,\square\, f_2^{**})(u)|_{u=0} \quad (f^* + g^*)^* = f^{**} \square g^{**} \text{ (infimal convolution)} \\
&= \min_{x_1 + x_2 = 0} f_1^{**}(-x_1) + f_2^{**}(x_2) \quad (f^{**} \square g^{**})(u) = \min_{u=x+y} f^{**}(x) + g^{**}(y)
\end{aligned}$$

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\begin{aligned}
\max_{\lambda} D(\lambda) &= \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda)) \\
&= \max_{\lambda} \lambda^T u - (f_1^*(-\lambda) + f_2^*(\lambda))|_{u=0} && f^{**}(u) = \max_{\lambda} \lambda^T u - f^*(\lambda) \\
&= (f_1^{**} \circ (-1) \,\square\, f_2^{**})(u)|_{u=0} && (f^* + g^*)^* = f^{**} \square g^{**} \text{ (infimal convolution)} \\
&= \min_{x_1 + x_2 = 0} f_1^{**}(-x_1) + f_2^{**}(x_2) && (f^{**} \square g^{**})(u) = \min_{u=x+y} f^{**}(x) + g^{**}(y) \\
&= \min_{x} f_1^{**}(x) + f_2^{**}(x)
\end{aligned}$$

What are we optimizing?

- Maximizing the dual $D(\lambda)$ can be rewritten as

$$\begin{aligned}
\max_{\lambda} D(\lambda) &= \max_{\lambda} -(f_1^*(-\lambda) + f_2^*(\lambda)) \\
&= \max_{\lambda} \lambda^T u - (f_1^*(-\lambda) + f_2^*(\lambda))|_{u=0} && f^{**}(u) = \max_{\lambda} \lambda^T u - f^*(\lambda) \\
&= (f_1^{**} \circ (-1) \,\square\, f_2^{**})(u)|_{u=0} && (f^* + g^*)^* = f^{**} \square g^{**} \text{ (infimal convolution)} \\
&= \min_{x_1 + x_2 = 0} f_1^{**}(-x_1) + f_2^{**}(x_2) && (f^{**} \square g^{**})(u) = \min_{u=x+y} f^{**}(x) + g^{**}(y) \\
&= \min_{x} f_1^{**}(x) + f_2^{**}(x)
\end{aligned}$$

- We are minimizing the sum of the two biconjugate functions $f_1^{**} + f_2^{**}$, which provides a lower bound to the true biconjugate $(f_1 + f_2)^{**}$
- Hence, we are solving a convex relaxation of the original problem

Subgradient Ascent ?

- The most straight-forward approach to maximize the dual problem is subgradient ascent, for this note that

$$x \in \partial f^*(\lambda) \iff \lambda \in \partial f^{**}(x) \iff x \in \arg\min_x -\lambda^T x + f^{**}(x)$$

- A subgradient can be computed by $g(\lambda) = x_1(\lambda) - x_2(\lambda) \in \partial D(\lambda)$, where

$$x_1(\lambda) = \arg\min_x \lambda^T x + f_1^{**}(x), \quad x_2(\lambda) = \arg\min_x -\lambda^T x + f_2^{**}(x),$$

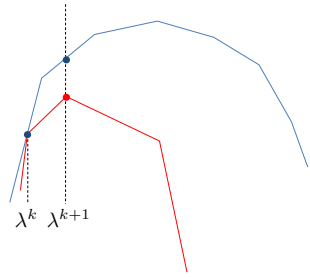
  hence for computing a subgradient, we just need to solve chain problems
- Subgradient ascent [Komodakis et al. '10]

$$\lambda^{k+1} = \lambda^k + h_k\, g(\lambda^k), \qquad \sum_{k=1}^{\infty} h_k = \infty, \ \sum_{k=1}^{\infty} h_k^2 < \infty$$

- Converges to a globally optimal solution but very slowly ...

Overview

Dual minorize-maximize (DMM)



- Consider again our Lagrange dual

$$\max_{\lambda} D(\lambda) = \underbrace{(-f_1^*(-\lambda))}_{D^1(\lambda)} + \underbrace{(-f_2^*(\lambda))}_{D^2(\lambda)}$$

- We propose to maximize the dual by performing a dual minorize-maximize (DMM) algorithm of the form

$$\begin{cases} \lambda^{k+\frac{1}{2}} = \max_{\lambda} \underline{D}^{1,k}(\lambda) + D^2(\lambda) \\ \lambda^{k+1} = \max_{\lambda} D^1(\lambda) + \underline{D}^{2,k}(\lambda), \end{cases}$$

  where $\underline{D}^{1,k}(\lambda) \leq D^1(\lambda)$ for all λ and $\underline{D}^{1,k}(\lambda^k) = D^1(\lambda^k)$ are minorants.
- Per definition yields a monotonically increasing sequence of dual function values

$$D(\lambda^k) \leq D(\lambda^{k+\frac{1}{2}}) \leq D(\lambda^{k+1})$$

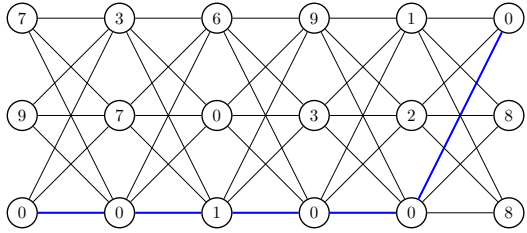
Maximal modular minorants

- In order to make the iterations easily solvable, we are interested in modular minorants $\varphi(x)$ for a function $f(x)$:
$$\varphi(x) \leq f(x), \ \forall x.$$

- We are further interested in the maximal modular minorant φ, i.e. there is no φ' such that
$$\varphi(x') < \varphi'(x')$$
for some x'.
- There are different possibilities to find a maximal modular minorant $\varphi(x)$ by distributing the slacks of the chain problems.

Example: Potts model

- Assume we have given the following Potts problem with edge cost $(\theta_{ij})_{kl} = 1$ if $k \neq l$ and 0 else.
- The graph is given by



- The maximal modular minorant is given by

$$2 + \begin{array}{cccccc} 6 & 4 & 6 & 7 & 1 & 0 \\ 8 & 7 & 1 & 2 & 2 & 7 \\ 0 & 0 & 0 & 0 & 0 & 7 \end{array}$$

The minorants

- Let $\mathcal{M}(f^1 + \lambda^k)(x)$ satisfying for all x the properties:

$$\mathcal{M}(f^1 + \lambda^k)(x) \leq (f^1 + \lambda^k)(x), \quad \min_x \mathcal{M}(f^1 + \lambda^k)(x) = \min_x (f^1 + \lambda^k)(x).$$

- In the DMM algorithm we use the following minorant

$$\underline{D}^{1,k}(\lambda) = \min_x \mathcal{M}(f^1 + \lambda^k)(x) + (\lambda - \lambda^k)^T x,$$

- We need to check the following two conditions

$$\forall \lambda \ \underline{D}^{1,k}(\lambda) \leq D^1(\lambda), \quad \underline{D}^{1,k}(\lambda^k) = D^1(\lambda^k)$$

- Observe that we have for all x and λ

$$\mathcal{M}(f^1 + \lambda^k)(x) + (\lambda - \lambda^k)^T(x) \leq (f^1 + \lambda^k)(x) + (\lambda - \lambda^k)^T(x),$$

  which also holds when taking the minimum w.r.t. x on both sides.
- At λ^k we have

$$\underline{D}^{1,k}(\lambda^k) = \min_x \mathcal{M}(f^1 + \lambda^k)(x) = \min_x \left(f^1 + \lambda^k\right)(x)$$

  since by construction, our minorant preserves the minimum value of $(f^1 + \lambda^k)(x)$

Solving for the iterates

- Using the aforementioned minorant, it turns out that the dual iterations can be solved easily
- Consider the following upper bound

$$\begin{aligned}
&\max_{\lambda} \underline{D}^{1,k}(\lambda) + D^2(\lambda) \\
&= \max_{\lambda} \min_{x} \mathcal{M}(f^1 + \lambda^k)(x) + (\lambda - \lambda^k)^T x + \min_{x} f^2(x) - \lambda^T x \\
&\leq \max_{\lambda} \min_{x} \mathcal{M}(f^1 + \lambda^k)(x) + (\lambda - \lambda^k)^T x + f^2(x) - \lambda^T x \\
&= \min_{x} \mathcal{M}(f^1 + \lambda^k)(x) - x^T \lambda^k + f^2(x)
\end{aligned}$$

- The upper bound is exactly attained by choosing

$$\lambda^{k+\frac{1}{2}} = \lambda^k - \mathcal{M}(f^1 + \lambda^k)$$

- The second update is computed in a similar way

$$\lambda^{k+1} = \mathcal{M}(f^2 - \lambda^{k+\frac{1}{2}}) - \lambda^{k+\frac{1}{2}}$$

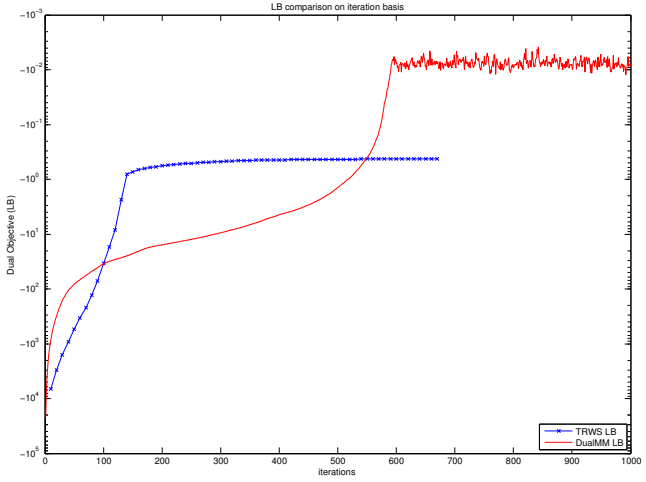
Discussion

- A highly practical maximal modular minorant is obtained from a hierarchical dynamic programming approach on chains that approximately uniformly distributes the slack.
- The computation of the hierarchical minorant is highly efficient, it can be computed in $O(n \log n \log D)$ using D parallel threads.
- The DMM algorithm yields a non-decreasing sequence of dual function values

$$D(\lambda^{k+1}) \geq D(\lambda^{k+\frac{1}{2}}) \geq D(\lambda^{k})$$

- Convergence of a subsequence can be guaranteed, but the algorithm may get stuck in a non-optimal point
- Usually leads to very good results in practice
- The primal solution is obtained by the solutions of the last chain problems
- The overall algorithm can be interpreted as a dual block-coordinate ascent on the dual problem

Comparison with TRW-S



- Comparison between DMM and TRW-S of the dual energy gap per iteration (of approximately equal number of memory accesses)
- DMM is very fast in the beginning and also converges to the globally optimal dual solution
- In practice we use only 5-10 iterations

Comparison with TRW-S

Method / $N^2 \times D$	128^2x16	256^2x32	512^2x64	1024^2x128
TRW-S (baseline)	0.0023s	0.0179s	0.1641s	1.2500s
TRW-S-CPU2	$\times$1.70	$\times$1.56	$\times$1.72	$\times$1.76
TRW-S-CPU4	$\times$2.28	$\times$1.88	$\times$2.29	$\times$2.34
TRW-S-CPU8	$\times$2.19	$\times$1.80	$\times$2.23	$\times$2.29
DMM-GPU	$\times$4.39	$\times$14.78	$\times$27.95	$\times$27.61

Table: Speedup for different problem sizes per iteration

- Parallelization of TRW-S using vectorized operations and multi-CPU saturates due to memory bandwidth limitations
- Parallelization of DMM on a GPU yields a large speedup particularly on larger problems

Overview

Direct algorithms for minimization of the 1D ROF model

- The 1D TV-ℓ_2 (ROF) model is given by

$$\min_x E(x) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| + \frac{\lambda}{2} \sum_{i=1}^{n} (x_i - f_i)^2$$

- This problem has already been studied in the past
- The Taut-String algorithm [Davies, Kovac '01] with $\mathcal{O}(n)$
- The direct algorithm of L. Condat [Condat '13] with $\mathcal{O}(n^2)$
- Has been recently improved to $\mathcal{O}(n)$ [Condat '17]
- The dynamic programming algorithm of N. Johnson [Johnson '13] with $\mathcal{O}(n)$

Total Variation on trees

In [Kolmogorov, P., Rolinek '15] we have extended and generalized the dynamic programming approach of Johnson to a larger class of problems

$$\min_x \sum_{(i,j)\in E} f_{ij}(x_j - x_i) + \sum_{i\in V} f_i(x_i),$$

where f_i are continuous function and $f_{ij}(z) = \min\{w_{ij} \cdot |z|, C_{ij}\}$, $w_{ij} \geq 0$.

- The algorithms we propose are based on efficient implementations of the Viterbi dynamic programming algorithm:

The ROF model

- Consider the simplified setting: $f_i(x_i) = (x_i - f_i)^2/2$, $f_{ij}(x_j - x_i) = |x_{i+1} - x_i|$
- We can rewrite the problem as a recursion of minimization problems

$$\min_{x_1,\dots,x_n} \sum_{i=1}^{n-1} |x_{i+1} - x_i| + \frac{\lambda}{2} \sum_{i=1}^{n} (x_i - f_i)^2 =$$

$$\min_{x_n} (x_n - f_n)^2/2 + \min_{x_{n-1}} (x_{n-1} - f_{n-1})^2/2 + |x_n - x_{n-1}| + \dots + \underbrace{\min_{x_1} (x_1 - f_1)^2/2 + |x_2 - x_1|}_{\phi(x_2)}$$

- Note that each minimization step defines an infimal convolution of the previous function with a $|\cdot|$ function.
- Forward step: Propagate the subgradients of the inf-convolutions from $x_1 \dots x_n$ and solve for x_n.
- Backward step: Solve for $x_{n-1} \dots x_1$ by solving the remaining 1D minimization problems

Propagation of the subgradient

- The first problem to be solved is given by the inf-convolution

$$\phi(x_2) = \min_{x_1}(x_1 - f_1)^2/2 + |x_2 - x_1|$$

- $\phi(x_2)$ is exactly the Huber function

$$\phi(x_2) = \begin{cases} (x_2 - f_1)^2/2 & \text{if} \quad |x_2 - f_1| \leq 1 \\ |x_2 - f_1| - \frac{1}{2} & \text{else} \end{cases}$$

- The derivative $\xi_2(x_2) = \phi'(x_2)$ is given by

$$\xi_2(x_2) = \max\{-1, \min\{1, x_2 - f_1\}\}$$

- In the next step we need to solve the inf-convolution

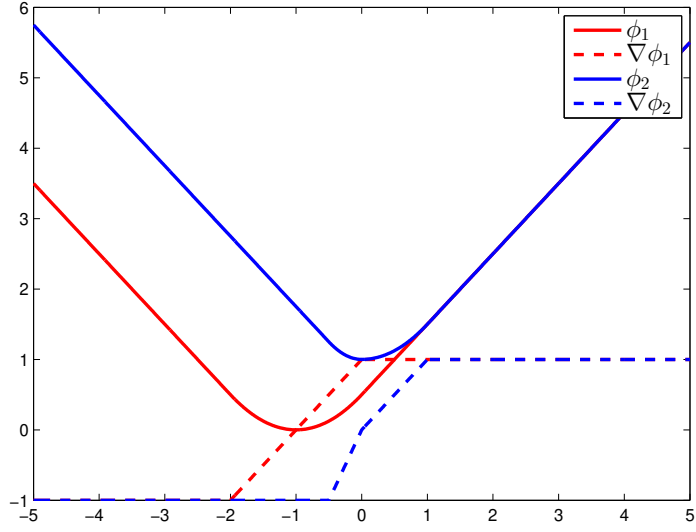
$$\phi(x_3) = \min_{x_2}(x_2 - f_2)^2/2 + \phi(x_2) + |x_3 - x_2|$$

- It is a smooth, piecewise quadratic function, the derivative $\xi_3(x_3) = \phi'(x_3)$ is given by

$$\xi_3(x_3) = \max\{-1, \min\{1, x_3 - f_2 + \xi_2(x_3)\}\}$$

Example

- Consider the two data points $f_1 = -1$, $f_2 = -2$
- $\phi_1(x)$ is a Huber function centered around $f_1 = -1$
- $\phi_2(x)$ is a more complicated piecewise quadratic function

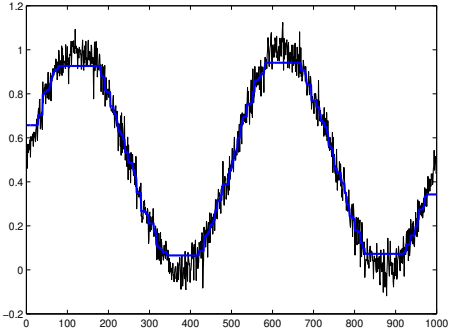


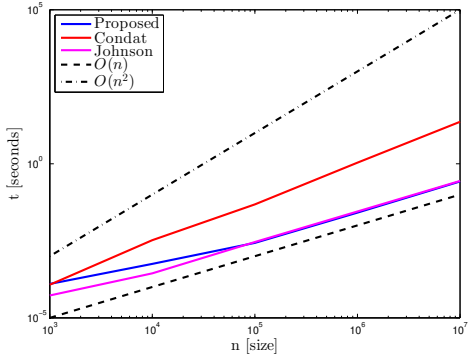
Complexities

Depending on the unary functions we obtain the following worst-case complexities:

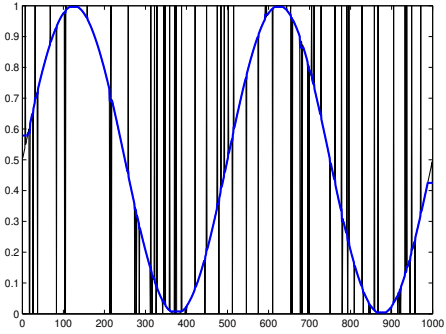
- $C_{ij} = \infty$, f_i are convex quadratic, we obtain a worst case complexity of $\mathcal{O}(n)$
- $C_{ij} = \infty$, f_i are convex piecewise linear, we obtain a worst case complexity of $\mathcal{O}(n \log \log n)$
- $C_{ij} = \infty$, f_i are convex piecewise quadratic, we obtain a worst case complexity of $\mathcal{O}(n \log n)$
- $C_{ij} = \infty$, f_i are non-convex piecewise linear, we obtain a worst case complexity of $\mathcal{O}(n^2)$
- $C_{ij} < \infty$, f_i are non-convex piecewise linear, we obtain an exponential worst case rate

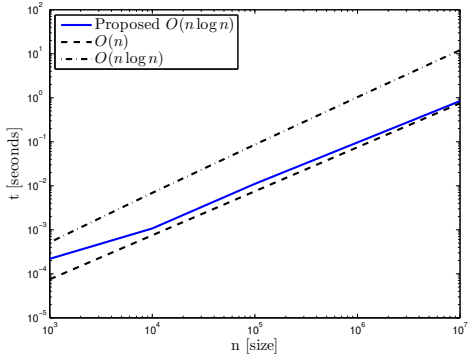
Example: 1D TV-ℓ_2 model



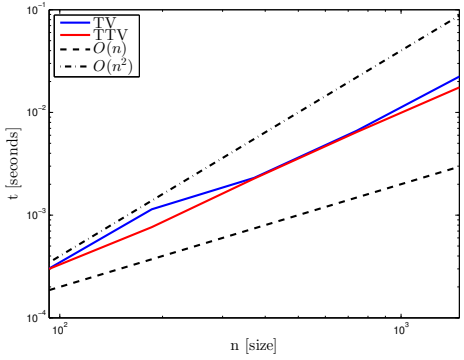


Example: 1D TV-ℓ_1 model





Example: 1D TV-non-convex model



Minimization of the 2D TV-ℓ_2 model

- Let us again consider the 2D TV-ℓ_2 model:

$$\min_u \mathrm{TV}_h(u) + \mathrm{TV}_v(u) + \frac{\lambda}{2}\|u - f\|^2,$$

- We perform the splitting $u = u_1$, $u = u_2$ and consider the Lagrangian function

$$\min_{u_{1,2},u} \max_{p,q} \mathrm{TV}_h(u_1) + \langle p, u - u_1\rangle + \mathrm{TV}_v(u_2) + \langle q, u - u_2\rangle + \frac{\lambda}{2}\|u - f\|^2.$$

- We can identify twice the definition of the convex conjugate which eventually leads to the dual problem

$$\min_{p,q} \mathrm{TV}_h^*(p) + \mathrm{TV}_v^*(q) + \frac{1}{2}\|p + q - \lambda f\|^2$$

- We have a block separable problem with a smooth coupling between p and q

The Moreau identity

- The block minimization algorithm is equivalent to the alternating projection method of Dykstra [Boyle, Dykstra '86]

$$\begin{cases} p^{k+1} = \text{prox}_{TV_h^*}(\lambda f - q^k) \\ q^{k+1} = \text{prox}_{TV_v^*}(\lambda f - p^{k+1}) \end{cases}$$

- The proximal maps wrt to $\text{TV}_{h,v}^*$ can be computed by the proximal maps wrt $\text{TV}_{h,v}$ using the celebrated Moreau identity

$$p = \text{prox}_{TV_h}(p) + \text{prox}_{TV_h^*}(p)$$

- The proximal map with respect to $\text{TV}_{h,v}$ decompose into independent 1D TV-ℓ_2 problem on chains which can be solved by our proposed dynamic programming algorithm in $\mathcal{O}(n)$ time
- The overall algorithm converges with rate $\mathcal{O}(1/k)$ [Beck, Tetruashvili '13]

Acceleration trick

- We can rewrite the last problem as follows:

$$\min_p \underbrace{\mathrm{TV}_h^*(p)}_{G(p)} + \underbrace{\min_q \mathrm{TV}_v^*(q) + \frac{1}{2}\|p + q - \lambda f\|^2}_{F(p)}$$

- The function $F(p)$ is the Moreau-envelope of the function TV_v^*
- Recall that $F(p)$ is smooth and its gradient, $\nabla F(p)$ is 1-Lipsschitz continuous.
- The alternating block minimization is equivalent to the forward-backward splitting, which can be accelerated [Beck, Teboulle '07]
- The accelerated overall algorithm converges with rate $\mathcal{O}(1/k^2)$ [Chambolle, P. '15]

Example

(a) Original image (600×800)

(b) $\lambda = 10$

(c) $\lambda = 5$

(d) $\lambda = 1$

Performance evaluation

Number of iterations to reach a certain primal-dual gap

	(AM)			(AAM)			(AAM-r)		
tol	$\lambda = 10$	$\lambda = 5$	$\lambda = 1$	$\lambda = 10$	$\lambda = 5$	$\lambda = 1$	$\lambda = 10$	$\lambda = 5$	$\lambda = 1$
10^{-1}	80	90	90	30	30	30	30	30	30
10^{-3}	410	380	550	80	80	80	80	80	80
10^{-6}	1810	2220	2830	270	260	300	170	180	190
10^{-9}	3770	10000+	5640	630	790	570	220	320	250

Timings using a multi-core implementation

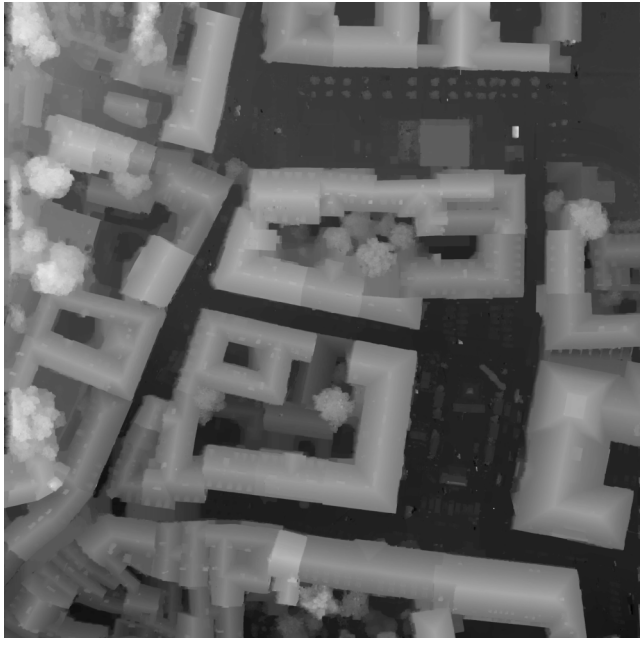
#cores	$\lambda = 10$	$\lambda = 5$	$\lambda = 1$
1	4.13	4.12	4.96
5	1.08	0.94	1.20
10	0.63	0.62	0.75
20	0.48	0.44	0.53
(GC)	3.82	6.37	19.76

Application to globally optimal stereo

- The same approach can be used to solve a globally optimal stereo formulation [Ishikawa '03, P. et al. '08]
- Minimizes the (exact) convex envelope of the non-convex stereo problem in a 3D space
- Can be solved by minimizing a weighted ROF model in 3D
- Accelerated alternating minimization does not work anymore
- We have to split it into two blocks, for one we make a descent, for the other we perform an exact minimization
- 10-20 iterations are sufficient to find an almost optimal solution
- Can be applied to large-scale stereo problems (9MPixel!)

Disparity image of Graz (3000x3000)

Disparity image of Graz (3000x3000)



Minimization of the 2D TV-ℓ_1 model

- Consider the TV-ℓ_1 model

$$\min_u \mathrm{TV}_h(u) + \mathrm{TV}_v(u) + \lambda \|u - f\|_1$$

- We perform the splitting $u = v$

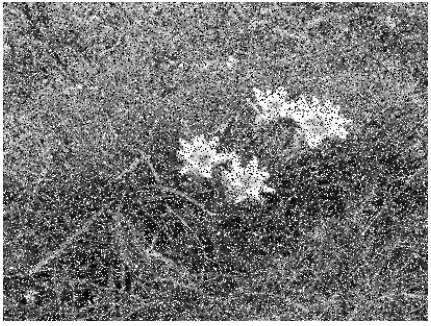
$$\min_{u,v} \max_p \mathrm{TV}_h(u) + \mathrm{TV}_v(v) + \lambda \|u - f\|_1 + \langle u - v, p \rangle$$

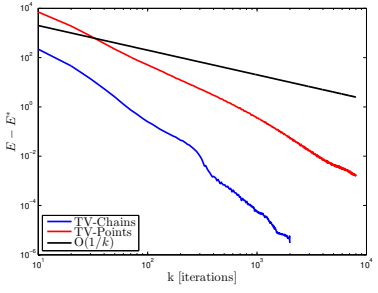
- Using again the definition of the convex conjugate we obtain

$$\min_u \max_p \underbrace{\mathrm{TV}_h(u) + \lambda \|u - f\|_1}_{G(u)} - \underbrace{\mathrm{TV}_v^*(p)}_{F^*(p)} + \langle u, p \rangle$$

- Enters the framework of primal-dual optimization
- The proximal map wrt $G(u)$ can be computed by 1D TV problems subject to piecewise quadratic unaries in $\mathcal{O}(n \log n)$ time.
- The proximal map wrt $F^*(p)$ can be computed by 1D TV problems subject to quadratic unaries in $\mathcal{O}(n)$ time.

Example




10^4
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
$E - E^*$
10^1
10^2
10^3
10^4
k [iterations]
TV-Chains
TV-Points
$O(1/k)$

Minimization of a non-convex TV model

- Consider the following non-convex TV model

$$\min_u \mathrm{TV}_h^C(u) + \mathrm{TV}_v^C(u) + \sum_{i,j} f_{i,j}(u_{i,j}),$$

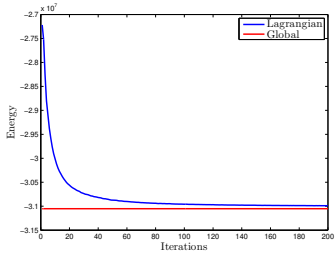
  where $\mathrm{TV}_{h,v}^C$ is the truncated total variation and $f_{i,j}(u_{i,j})$ are non-convex piecewise linear functions

- We again consider a splitting $u = v$ which yields the Lagrangian function

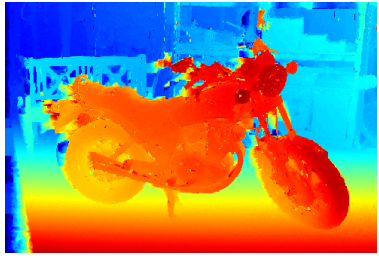
$$\min_{u,v} \max_p \mathrm{TV}_h^C(u) + \frac{1}{2}\sum_{i,j} f_{i,j}(u_{i,j}) + \mathrm{TV}_v^C(v) + \frac{1}{2}\sum_{i,j} f_{i,j}(v_{i,j}) + \langle u - v, p\rangle$$

- Non-convex in u, v but concave in p
- Solves an approximate convex envelope of the original problem
- We consider a straight-forward application of the primal-dual algorithm with varying step sizes
- The proximal maps can be solved with the dynamic programming algorithms for non-convex piecewise linear unaries

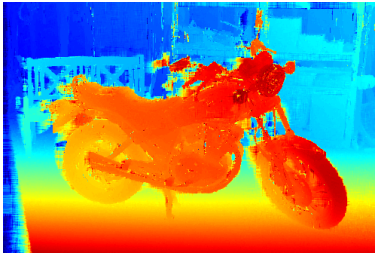
Results



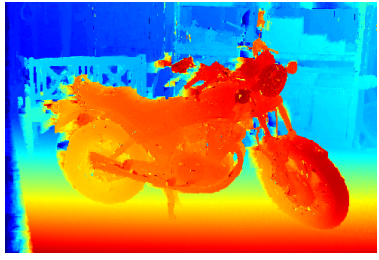
(a) Convergence



(b) Globally optimal



(c) Lagrangian, $k = 1$



(d) Lagrangian, $k = 10$