Fast MRF Optimization with Application to Depth Reconstruction

Qifeng Chen* Vladlen Koltun†

Abstract

We describe a simple and fast algorithm for optimizing Markov random fields over images. The algorithm performs block coordinate descent by optimally updating a horizontal or vertical line in each step. While the algorithm is not as accurate as state-of-the-art MRF solvers on traditional benchmark problems, it is trivially parallelizable and produces competitive results in a fraction of a second. As an application, we develop an approach to increasing the accuracy of consumer depth cameras. The presented algorithm enables high-resolution MRF optimization at multiple frames per second and substantially increases the accuracy of the produced range images.

1. Introduction

Markov random fields have become a standard representation for dense optimization problems in computer vision, including depth reconstruction, image segmentation, image restoration, image composition, and dense motion estimation [2, 29, 1]. State-of-the-art optimization algorithms for first-order MRFs are both elegant and highly accurate on standard benchmark models [13, 14, 11]. In fact, the accuracy of the best known algorithms sometimes exceeds the demands of the application and the accuracy of the underlying model by a wide margin [29]. On the other hand, these algorithms tend to be slow and are in some cases limited in the types of potentials they can handle. For example, while the advantages of MRF models for stereo reconstruction have been extensively demonstrated by the research community, practical stereo systems shun such models in favor of fast algorithms that do not involve global optimization.

In this paper, we describe a fast algorithm for optimizing Markov random fields over images. We focus on first-order models with large ordered label sets. Such label sets generally describe magnitudes, such as image intensities or depths. They arise naturally in problems such as depth reconstruction and image restoration. Our algorithm, described in Section 2, is not as accurate on standard benchmark problems as state-of-the-art solvers such as TRW [13]. It is, however, extremely simple and trivially parallelizable. In a fraction of a second, the algorithm produces results that have substantially lower energy than the energy of the ground truth labeling [29].

As an application of the presented algorithm, we develop a global optimization approach to improving the accuracy of range images produced by consumer depth cameras. Such cameras have become widely used in computer vision research and applications. The most ubiquitous such devices use speckle patterns. The range image is computed by comparing a stored infrared image of a speckle pattern projected onto a flat calibration surface with an image of the same speckle pattern projected onto the present scene [26]. Tens of millions of depth cameras that operate on this principle have been shipped [19] and range data acquired by these sensors is being used in a variety of computer vision applications [7]. While the data is undoubtedly useful, many concerns about its quality have been raised. Range images produced by current speckle-based sensors suffer from a number of significant defects, including high-frequency noise, quantization, and missing data [28, 12, 21, 6]. For this reason, systems that use these images routinely preprocess them using bilateral filtering [20, 21, 7], second-order smoothness terms [10, 27], or fusion of multiple images [17]. Other systems are specifically tailored to lower their susceptibility to the characteristic defects of the data [6].

In Section 4, we develop a global optimization approach to computing range images from projected speckle patterns. Current speckle-based range cameras follow a winner-take-all approach, associating the most likely range value with each patch in the image. We instead optimize a global objective that balances local correlations of the projected pattern with a robust regularization term. The MRF optimization algorithm described in Section 2 allows this optimization to be performed with very high accuracy at multiple frames per second.

To evaluate the presented depth reconstruction approach, we have acquired ground-truth three-dimensional models for five highly detailed objects using an industrial 3D scanner. Extensive experiments with this ground-truth data demonstrate that our global optimization approach substantially improves the accuracy of range images produced by consumer depth cameras.

*Stanford University
†Adobe Research

2. Fast MRF Optimization

Consider an image of size $W \times H$. For each pixel $p = (x, y)$ in the image, our goal is to assign it a label $l_p = l_{(x,y)}$. Let L be the number of possible labels. For simplicity of exposition, assume that the labels are natural numbers from 0 to $L - 1$. We also restrict the exposition to first-order models over the 4-connected pixel grid, although the presented algorithm can be generalized beyond this setting. The MRF optimization problem can be expressed as follows:

$$\mathbf{l}^\star = \arg\min_{\mathbf{l}} \mathbf{E}(\mathbf{l}) = \arg\min_{\mathbf{l}} \big(\gamma \mathbf{E}_{\text{data}}(\mathbf{l}) + \mathbf{E}_{\text{reg}}(\mathbf{l})\big). \quad (1)$$

Here $\mathbf{l}$ is the set of label assignments for all pixels and $\mathbf{l}^\star$ is the sought after assignment that minimizes the objective $\mathbf{E}(\mathbf{l})$. The objective consists of a data term $\mathbf{E}_{\text{data}}(\mathbf{l})$ and a regularization term $\mathbf{E}_{\text{reg}}(\mathbf{l})$, balanced by a coefficient γ.

The data term expresses the local unary cost incurred by each pixel under a given label assignment. It can be represented as a table $\mathbf{V}$ of size $W \times H \times L$, such that $\mathbf{V}(x, y, l)$ is the cost of assigning label l to pixel (x, y). This table is sometimes referred to as the cost volume.

The regularization term associates costs with pairs of neighboring pixels. These costs are commonly formulated in terms of differences between the assigned labels:

$$\mathbf{E}_{\text{reg}}(\mathbf{l}) = \sum_{\{p,q\} \in \mathcal{N}} w_{pq}\, \rho\big(\ell_p - \ell_q\big), \quad (2)$$

where $\mathcal{N}$ is the 4-connected pixel grid. ρ is a penalty function, such as the L^1 norm, the L^2 norm, the Charbonnier penalty, or truncated versions of such penalties. w_{pq} is a weight associated with the edge $\{p, q\}$.

The problem can be expressed recursively:

$$\ell_p^\star = \arg\min_{\ell_p} \left(\gamma \mathbf{V}(x, y, \ell_p) + \sum_q w_{pq}\, \rho\left(\ell_p - \ell_q^\star\right) \right), \quad (3)$$

where $p = (x, y)$ and the sum is over pixels q such that $\{p, q\} \in \mathcal{N}$. This expression will be useful in the subsequent presentation of our algorithm.

2.1. Block coordinate descent

Our algorithm optimizes the objective $\mathbf{E}(\mathbf{l})$ by block coordinate descent. In each step, $\mathbf{l}$ is fixed outside a complete horizontal or vertical line in the image. The optimal label assignment along this line is computed by dynamic programming, given the fixed assignments along the other lines. Thus each block coordinate descent step performs an optimal update of a complete image row or column. We process the lines in four blocks: even rows, odd rows, even columns, odd columns. This enables trivial parallelization, since the lines in each block are not adjacent and can be processed in parallel. To initialize the algorithm, we produce two label assignments: one generated by optimizing all rows independently and the other by independently optimizing all columns. The initial label of each pixel is chosen randomly from its labels in these two assignments.

Each block coordinate descent step optimizes the complete objective $\mathbf{E}(\mathbf{l})$. While most variables are fixed in any given step, the objective remains the same. Thus the global objective either decreases or remains unchanged in each step. When the objective cannot be decreased by any step, a local minimum has been reached and the algorithm has converged. The algorithm is thus guaranteed to converge.

To optimize the model along a line in the image as required in each step of the algorithm, we use dynamic programming. A naive application of dynamic programming would have complexity $O(VL^2)$ in each step, where V is the number of pixels in the line. However, as observed by Felzenszwalb and Huttenlocher, this can be accelerated to $O(VL)$ without loss of optimality [5]. The Felzenszwalb-Huttenlocher acceleration scheme was presented in the context of belief propagation and was formulated for a number of specific penalty functions ρ. In Section 2.2 we give a general formulation that clarifies that this acceleration scheme is applicable to all penalty functions that satisfy a certain combinatorial criterion. In particular, the acceleration scheme applies to all convex penalties, non-convex robust penalties such as the Lorentzian and the generalized Charbonnier, and to truncated variants of these functions.

2.2. Dynamic programming

To avoid notational clutter in this section, we omit the edge weights w_{pq}, although their incorporation into the presented algorithms is straightforward. We begin by presenting the naive dynamic programming algorithm, which runs in time $O(VL^2)$. Without loss of generality, assume that we are optimizing the label assignments for row y in the image. Thus all assignments outside this line $\big(\{\ell_{(x,y')} : 0 \le x < W, 0 \le y' < H, y' \ne y\}\big)$ are fixed and our goal is to compute the optimal assignments along this line $\big(\{\ell_{(x,y)} : 0 \le x < W\}\big)$. The naive dynamic programming algorithm sweeps the row from left to right and computes the following cumulative cost for each x from 0 to $W - 1$ and for each possible label l:

$$\begin{aligned} \mathbf{C}(x, y, l) \;=\; & \gamma \mathbf{V}(x, y, l) \\ & + \rho\big(l - \ell_{(x,y-1)}\big) + \rho\big(l - \ell_{(x,y+1)}\big) \\ & + \min_{0 \le k < L} \big(\mathbf{C}(x-1, y, k) + \rho(l - k)\big), \end{aligned} \quad (4)$$

with appropriate handling of image boundaries. The algorithm thus computes WL values that fill the two-dimensional slice of the volume $\mathbf{C}$ that corresponds to row y. The procedure is initialized by filling in the L values

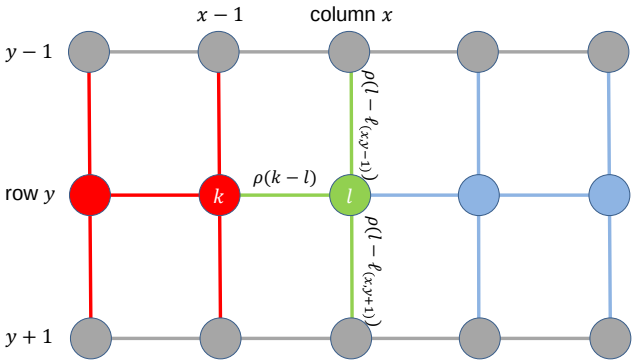


Figure 1. Illustration of the dynamic programming algorithm. Label assignments to the grey nodes are fixed. Cumulative costs $\mathbf{C}$ for the red nodes have been computed and the algorithm is now processing the green node. To compute the cumulative cost $\mathbf{C}(x, y, l)$, the algorithm must minimize over terms involving possible assignments k to the previous node. Blue nodes will be processed subsequently.

for pixel $(0, y)$, for which the recursive term in (4) is inactive. Following this initialization, the L values for pixel $(1, y)$ are filled in, and so forth by induction. This procedure is illustrated in Figure 1. To compute each value $\mathbf{C}(x, y, l)$, the procedure needs to minimize over the L values $\mathbf{C}(x-1, y, k)$ for the previous pixel. Thus the complexity of the naive dynamic programming procedure is $O(WL^2)$.

We now describe how the cumulative costs $\mathbf{C}$ can be computed for all labels l for pixel (x, y) in total time $O(L)$ instead of $O(L^2)$. To this end, let's define the following auxiliary functions:

$$\begin{aligned} f_k(l) &= \mathbf{C}(x-1, y, k) + \rho(l-k), \\ e(l) &= \min_{0 \le k < L} f_k(l). \end{aligned} \tag{5}$$

Substituting (5) into (4) yields

$$\begin{aligned} \mathbf{C}(x, y, l) &= \gamma \mathbf{V}(x, y, l) \\ &+ \rho\big(l - \ell_{(x,y-1)}\big) + \rho\big(l - \ell_{(x,y+1)}\big) \\ &+ e(l). \end{aligned} \tag{6}$$

This implies that if $e(l)$ can be evaluated in time $O(1)$ then the cumulative cost $\mathbf{C}(x, y, l)$ can be computed in time $O(1)$, since all the other terms that make up the cost can be evaluated in constant time.

The function e is the lower envelope of the set of functions $\{f_k\}_k$. Lower envelopes have been extensively studied [24]. In particular, it is known that if each pair of functions (f_i, f_j) intersects at most a constant number of times, the complexity of the lower envelope e is near-linear, rather than quadratic as could be naively supposed. Specifically, the complexity of e is in general $O(L\beta(L))$, where β is an extremely slowly growing function related to the inverse of Ackermann's function [24]. In our case, the situation is even more benign because the functions f_k have special structure: they are all shifted copies of the penalty function ρ. For all penalty functions encountered in the literature, it can be shown that each pair (f_i, f_j) intersects at most twice. This is the case for all convex penalties (L^1, L^2, Huber, Charbonnier, etc.) and for non-convex functions such as the Lorentzian and the generalized Charbonnier penalties. Proofs are provided in supplementary material. In this case, the complexity of the lower envelope e is strictly linear ($O(L)$) without any additional factors. Truncated penalties can be handled easily by observing that

$$\begin{aligned} e(l) &= \min_{0 \le k < L} \Big(\mathbf{C}(x-1, y, k) + \min\big(\rho(l-k), T\big) \Big) \\ &= \min\Big(\min_{0 \le k < L} \big(\mathbf{C}(x-1, y, k) + \rho(l-k) \big), \\ &\qquad T + \min_{0 \le k < L} \mathbf{C}(x-1, y, k) \Big), \end{aligned} \tag{7}$$

where T is the truncation value. Thus the truncated variant of any penalty ρ can be handled with no increase in computational complexity.

The lower envelope e can be computed over its entire domain $0 \le l < L$ in linear time $O(L)$. The algorithm is a simple generalization of a procedure described by Felzenszwalb and Huttenlocher and can be found in our reference implementation. This enables us to compute the cumulative costs $\mathbf{C}(x, y, l)$ for all $0 \le l < L$ in total time $O(L)$ as follows. First we compute the function $e(l)$ for all l in time $O(L)$. The result of this computation is an array of function values that enables evaluating $e(l)$ for any given l in time $O(1)$. As described above, this implies that each cumulative cost $\mathbf{C}(x, y, l)$ can now be evaluated in constant time. Overall, we can compute all $\mathbf{C}(x, y, l)$ for $0 \le l < L$ in time $O(L)$ and thus complete the entire dynamic programming algorithm for row y in time $O(WL)$.

3. Evaluation of MRF Optimization

We evaluate the presented block coordinate descent (BCD) algorithm on the benchmark problems of Szeliski et al. [29] that have large ordered label sets. These are the three stereo matching problems (Tsukuba, Venus, and Teddy) and the two image restoration problems (Penguin and House). (These problems are also used in the recent collection of Kappes et al. [11].) The BCD algorithm is compared to the high-performing algorithms identified by Szeliski et al.: graph cuts with α-expansion moves (`Expansion`) and $\alpha\beta$-swap moves (`Swap`), two implementations of loopy belief propagation (`BP-M` and `BP-S`), and TRW (`TRW-S`). We also include the Fast-PD algorithm

of Komodakis et al. [15], which was developed with an emphasis on speed (FastPD).

The results are shown in Figure 2 (left), following the visualization methodology of Szeliski et al. Note that running time is plotted on a logarithmic scale. In this experiment and in ones reported in Section 5, running times are measured on a desktop machine with 16GB of RAM and a quad-core Intel Core i7-3770K CPU clocked at 3.50GHz. We use a parallelized implementation of BCD. The BCD algorithm is not as accurate at convergence as TRW, but is much faster. On all problems, the BCD algorithm converges to an objective value that is much lower than the objective value of the real-world ground truth solution [29].

The performance of different algorithms is affected by the smoothness terms that are used in the different benchmark problems. Three of the problems (House, Penguin, and Venus) use the L^2 norm and two use the L^1 norm, all with different truncation values. To evaluate the robustness of different algorithms to changes in the smoothness term, we conducted a second experiment in which the accuracy of different algorithms is measured as a function of the truncation distance. The results are shown in Figure 2 (right). The form of the underlying penalty function ρ remains the same: L^1 or L^2, depending on the problem. The corresponding truncated penalty is defined as $\rho'(a) = \min(\rho(a), \rho(d))$. Figure 2 (right) shows the objective value of each algorithm at convergence as a function of the truncation distance d. TRW is the clear winner in terms of accuracy, at the price of substantial running times. On the other hand, a number of algorithms (FastPD, BP−S, and Swap) are clearly sensitive to the precise shape of the smoothness penalty and should not be used without consideration of this aspect of the model. Among the remaining algorithms, BCD and BP−M are both very robust. BP−M is more accurate on average, but parallelized BCD is two orders of magnitude faster.

4. Depth Reconstruction

The problem of estimating the true range information from noisy and incomplete range images has been studied extensively. A common approach is to rely on a color image that is acquired in tandem with the range image, and to use information from the color image to rectify the range image [3, 30, 9, 22, 27, 7, 8, 25]. This approach has a number of drawbacks. First, range cameras are not always accompanied by color cameras. Second, even tightly coupled RGB-D camera pairs do not produce perfectly compatible range and color images: the color image is acquired from a different viewpoint and the shutters are in practice not perfectly synchronized, leading to misalignments between the images. Third, even perfectly aligned color images provide unreliable information on scene geometry.

In this section, we develop a depth reconstruction approach that does not require additional information chan-

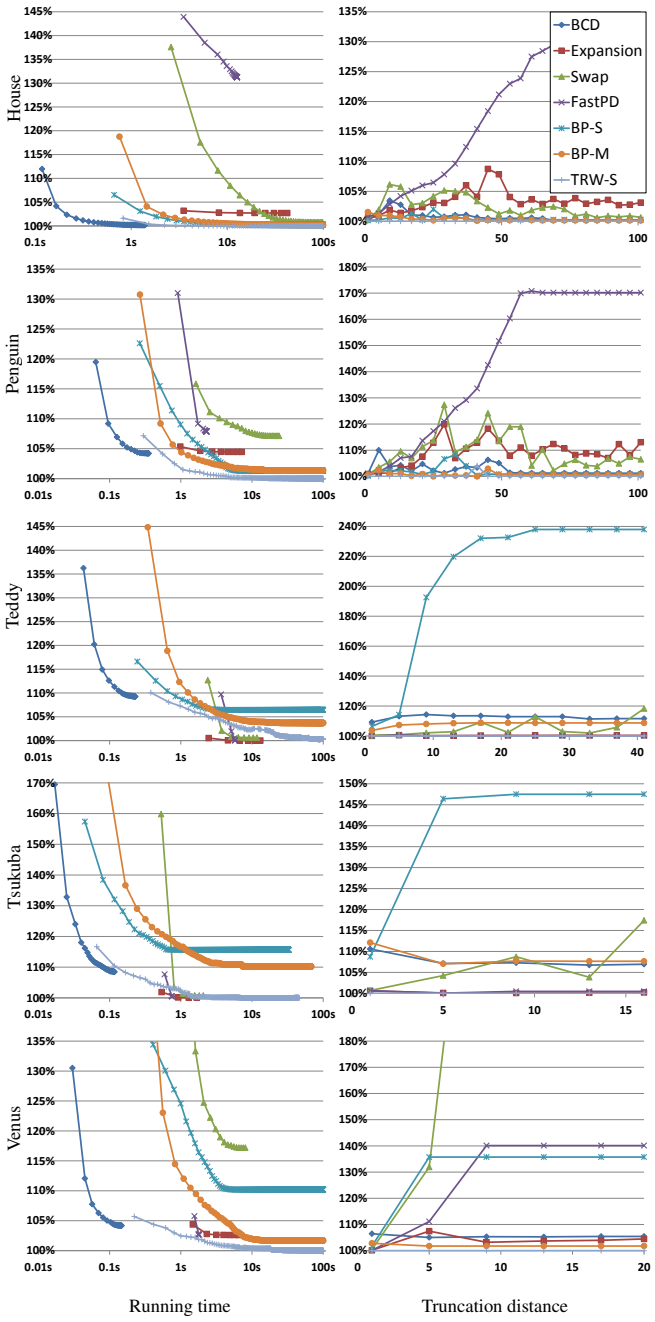


Figure 2. **Left.** The presented BCD algorithm is compared to alternative MRF optimization algorithms on benchmark problems. Objective value of the solution found by each algorithm is plotted over time. Running time in seconds is shown on a logarithmic scale. Objective value is plotted in percentage points, where 100% is set to be the lowest value achieved by any algorithm on the problem. BCD is not as accurate as state-of-the-art solvers such as TRW, but is much faster. **Right.** Evaluation of the robustness of different algorithms to the shape of the smoothness penalty in the model. Objective value of each algorithm at convergence is plotted as a function of the truncation distance of the smoothness term. TRW, BP-M, and BCD are very robust.

nels and operates directly on the infrared speckle pattern image produced by consumer depth cameras [26, 16, 28]. These cameras perform triangulation on a pair of infrared images. Both images are of a fixed speckle pattern projected by the sensor: one is of the pattern projected onto the current scene and the other is a fixed calibration image of the pattern projected onto a flat plane. The goal is to estimate the disparity between the two images for every visible point in the scene. In currently deployed systems, disparity is estimated by correlating horizontally aligned patches in the two images. For concreteness, consider a horizontal search window of 64 pixels, subpixel disparity estimation at $\frac{1}{8}$-pixel resolution, patches of size 9x9, and the normalized cross-correlation measure. These details are consistent with published accounts of deployed systems and are not essential to any aspect of our technique.

The input can be represented as a cost volume $\mathbf{V}$ of size $W \times H \times L$, where $W \times H$ is the image size and L is the number of possible subpixel disparities. The cost volume stores the computed correlation-based costs for all pixels and all possible disparities. The basic disparity estimation algorithm used by deployed systems can be clearly expressed in terms of the cost volume: select the lowest-cost disparity independently for each pixel, $\ell^{\star}_{(x,y)} = \arg\min_{\ell_{(x,y)}} \mathbf{V}(x, y, \ell_{(x,y)})$. In place of this local winner-take-all approach, our model integrates pairwise smoothness terms on neighboring pixels as expressed in equation (3). A number of penalty functions can be used and we evaluate different penalty functions in Section 5. The complete objective is optimized using the algorithm described in Section 2.

We adjust the data term $\mathbf{V}$ in two ways to model specific characteristics of our input. First, we have observed that correlation functions $\mathbf{V}(x, y, \cdot)$ fall into two types. First, there are high-entropy low-confidence pixels for which the correlations for all disparities are low (and the associated costs are high). Second, there are low-entropy high-confidence pixels for which the correlation function has a single dominant peak. This is quite unlike the data terms commonly encountered in passive stereo, where multiple high-correlation peaks and broad high-correlation plateaus are common. This is because the projected speckle pattern has the property that the sub-pattern in any local patch is almost always unique within the search window. Thus the correlation function generally either encounters a single strong match or none at all.

This characteristic of the input casts the role of the regularizer in our model in a new light. The primary role of the regularizer is to propagate information from high-confidence pixels to low-confidence ones. In particular, the influence of the regularizer on the high-confidence pixels themselves should be significantly weaker, since the data term at these pixels is extremely informative. There are different ways to accomplish this. An approach that works well in practice is to simply amplify the high correlation values at the low-entropy pixels.

The second adjustment to the data term involves occlusion regions. These are parts of the infrared image that are occluded from the projected speckle pattern and thus have no projected signal. These regions are easy to detect, since their infrared intensity is considerably lower. If the model is applied without modification within occluded regions, it will lead to smoothing across these regions. A more reasonable assumption is that the depth in the occluded regions should be consistent with the depth of the farther adjacent layer. We thus replace the data term $\mathbf{V}(x, y, \cdot)$ for each occluded pixel (x, y) with a single high-confidence plateau around the depth of the farthest highly confident pixel near a corresponding occlusion boundary.

5. Evaluation of Depth Reconstruction

In this section we evaluate the depth reconstruction approach presented in Section 4. We begin by describing a new dataset of scanned objects that enables us to precisely measure the metric accuracy of different depth reconstruction algorithms. We then evaluate the effect of the MRF smoothness penalty on the accuracy of our approach, compare the performance of the BCD algorithm presented in Section 2 to other MRF optimization algorithms on the depth reconstruction problem, and provide an extensive comparison of our depth reconstruction approach to range rectification algorithms advocated in prior work.

5.1. Data

Ground truth data. We evaluate the presented depth reconstruction approach on a set of images of five objects for which we acquired precise three-dimensional models. The ground-truth three-dimensional models are shown in Figure 3. The objects were purchased from an online retailer. Two objects, the "lion" and the "fox", are decorative sculpted animals, roughly 70cm high. Two objects, the "angel" and the "gargoyle", are decorative sculptures of fantastical creatures. They are 77cm and 60cm high, respectively. The fifth object, the "arch", is a miniature of the Arc De Triomphe, originally intended to be used as an end table. It is 50cm high.

Highly accurate ground truth geometry was acquired with a GOM ATOS II 3D scanner, developed for applications in the aerospace, automotive, and other industries. The ATOS scanner acquires dense depth images with a 0.2 millimeter point spacing and a 25 micron depth accuracy. Individual depth images were precisely aligned using photogrammetry targets. The application of photogrammetry targets is apparent in the corresponding 5mm diameter holes in the ground truth data shown in Figure 3. We did not apply hole filling or any other post processing to the data.

Figure 3. Ground truth models acquired using an industrial 3D scanner. From left to right: angel, arch, fox, gargoyle, and lion.

Test data. To evaluate different components of our approach and to compare our approach to alternatives, we used a PrimeSense Carmine range camera to acquire images of each of the five objects. For each object, the camera was mounted on a tripod and the object was put on a turntable roughly 1m in front of the camera. We imaged each object at 12 angular intervals of approximately 30 degrees each. At each interval, the PrimeSense camera produced three images: a range image, a color image, and an infrared image of the projected speckle pattern.

Each approach we evaluate is used to produce a depth image for each of the 12 poses of each object. To evaluate a given depth image produced by any of the approaches, the ground truth three-dimensional model is aligned to the test depth image using ICP. Points on the ground truth model that are visible from the camera are then automatically identified. For each of these visible points, the Euclidean distance to the closest point on the evaluated depth image is computed. We use the mean of these distances as our primary quantitative evaluation measure, reported in millimeters.

5.2. Experiments

Penalty function. We begin by evaluating different penalty functions ρ that can be used in our model. We evaluate the truncated L^1, L^2, and Charbonnier penalties. For each penalty, we perform five-fold cross-validation on the parameter γ and the truncation threshold T: in each round of cross-validation, we train on four of the objects (e.g., fox, lion, angel, arc) and test on the fifth (e.g., gargoyle). For the Charbonnier penalty we also cross-validate on the parameter ε that is involved in the construction of this function. The quantitative results are shown in Table 1. The results indicate that the robust Charbonnier penalty performs best. We use the truncated Charbonnier penalty henceforth.

Optimization algorithm. We now compare the performance of the BCD algorithm presented in Section 2 to other MRF optimization algorithms on the depth reconstruction problem. We compare BCD to Swap, Expansion, and TRW-S. The results are provided in Table 2.

	time	angel	arch	fox	garg.	lion	*mean*
L^1	0.23 s	3.84	3.46	3.15	4.71	4.42	3.91
L^2	0.28 s	3.93	3.38	3.19	4.79	4.47	3.95
Charbonnier	0.31 s	3.70	2.99	3.08	4.64	4.43	3.77

Table 1. Metric accuracy (in millimeters) obtained with different penalty functions.

	time	angel	arch	fox	garg.	lion	*mean*
Expansion	112 s	4.45	3.74	3.39	4.94	4.81	4.26
Swap	23 m	4.27	3.51	3.20	4.77	4.51	4.05
TRW-S	83 m	**3.71**	3.39	**3.06**	**4.49**	**4.24**	**3.78**
BCD	**0.31 s**	**3.70**	**2.99**	**3.08**	4.64	4.42	**3.77**

Table 2. Metric accuracy (in millimeters) of different MRF optimization algorithms on the depth reconstruction problem.

Comparison to range image rectification. A large number of range image rectification schemes have been advocated in the literature. Some attempt to rectify the original range image produced by the camera by filtering, others make use of a corresponding color image that is assumed to be provided in addition to the original range image. In contrast to these approaches, our algorithm operates on the raw speckle pattern image acquired by the camera and uses a different model to infer the range image from the speckle pattern image. The experiments reported below show that our approach significantly outperforms all range image rectification algorithms, without relying on additional information channels such as color. Some additional range images produced by the presented approach are shown in Figure 4.

The first two approaches we compare to are based on bilateral filtering. It is common in practice to rectify a range image by applying a bilateral filter [20, 21, 7]. This is thus

the first approach we evaluate. We cross-validate on the two standard deviations of the bilateral filter. A more advanced application of bilateral filtering ideas uses a joint bilateral filter [30, 4, 23]. This is the second approach we evaluate. We use the implementation provided by Silberman et al. as part of the NYU Depth dataset and cross-validate on its parameters [27]. The third approach we include is an adaptation of the colorization algorithm of Levin et al. [18] that uses second-order smoothness terms to smooth and extrapolate the available range data. This is the default range rectification algorithm used in the NYU Depth dataset [27]. We use their implementation. The fourth algorithm we compare to is the classical Diebel-Thrun MRF [3]. We use our own implementation and cross-validate on the parameters. The fifth algorithm is the nonlocal filtering approach of Park et al. [22]. The sixth is the Fields of Experts model of Herrera et al. [8]. For the last two approaches we use implementations provided by the authors. The results of the experiment are shown in Table 3.

	time	angel	arch	fox	garg.	lion	*mean*
Original		4.76	3.77	3.55	5.22	5.29	4.52
BF	5.2 s	4.48	3.70	3.27	5.15	4.73	4.27
JBF	1.8 s	4.61	3.61	3.57	5.24	4.91	4.39
Colorization	22 s	4.35	3.45	3.36	5.54	4.57	4.26
DT	3.1 s	4.39	3.47	3.39	5.55	4.57	4.27
Nonlocal	44 s	4.56	3.55	3.50	5.65	4.81	4.42
FoE	79 m	4.22	3.50	3.31	4.95	4.50	4.10
Ours	**0.31 s**	**3.70**	**2.99**	**3.08**	**4.64**	**4.42**	**3.77**

Table 3. Metric accuracy (in millimeters) achieved by range image rectification algorithms and by our approach. The top row shows the accuracy of the original range image produced by the camera. The following rows report the accuracy achieved by bilateral filtering (BF), joint bilateral filtering (JBF), an adaptation of the colorization approach of Levin et al. [18], the MRF model of Diebel and Thrun [3] (DT), the nonlocal filtering approach of Park et al. [22], the Fields of Experts model of Herrera et al. [8] (FoE), and our approach.

6. Conclusion

We presented a simple algorithm for MRF optimization. The presented BCD algorithm is not as accurate as state-of-the-art solvers on standard benchmark problems, but is trivially parallelizable, producing competitive results in a fraction of a second. We then developed a new global optimization approach to improving the accuracy of range images produced by consumer depth cameras. The presented BCD algorithm allows the approach to be applied at multiple frames per second. The approach significantly outperforms all range image rectification algorithms, without relying on additional information channels such as color.

References

[1] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[2] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[3] J. Diebel and S. Thrun. An application of Markov random fields to range sensing. In *NIPS*, 2005.

[4] J. Dolson, J. Baek, C. Plagemann, and S. Thrun. Upsampling range data in dynamic environments. In *CVPR*, 2010.

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70(1), 2006.

[6] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013.

[7] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5), 2013.

[8] D. Herrera, J. Kannala, P. Sturm, and J. Heikkilä. A learned joint depth and intensity prior using Markov random fields. In *3DV*, 2013.

[9] B. Huhle, T. Schairer, P. Jenke, and W. Straßer. Fusion of range and color images for denoising and resolution enhancement with a non-local filter. *Computer Vision and Image Understanding*, 114(12), 2010.

[10] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-D object dataset: Putting the Kinect to work. In *ICCV Workshops*, 2011.

[11] J. H. Kappes et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR*, 2013.

[12] K. Khoshelham and S. O. Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2), 2012.

[13] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10), 2006.

[14] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *PAMI*, 33(3), 2011.

[15] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. In *CVPR*, 2007.

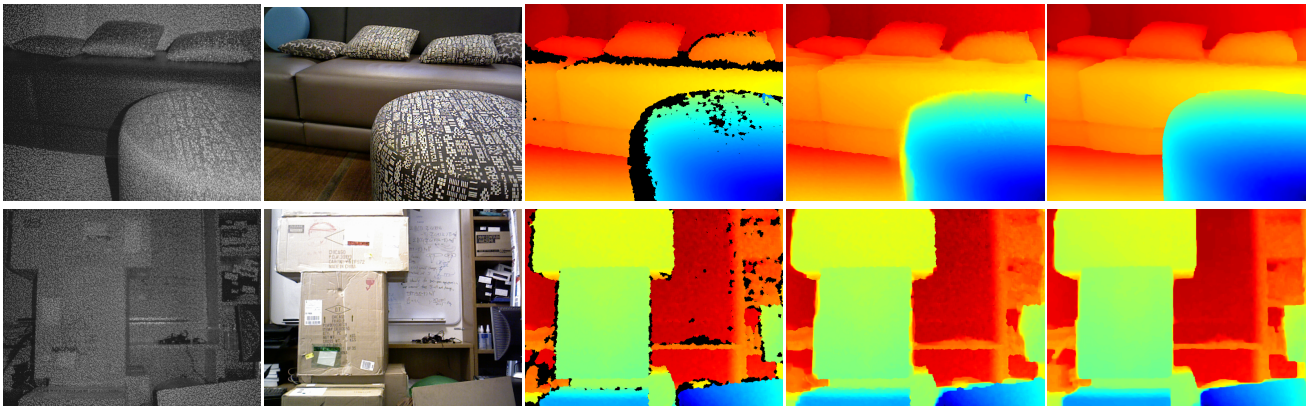
[16] K. Konolige and P. Mihelich. *Technical description of Kinect calibration*. 2012. `http://wiki.ros.org/kinect_calibration/technical`.

[17] K. J. Lee, Q. Zhao, X. Tong, M. Gong, S. Izadi, S. U. Lee, P. Tan, and S. Lin. Estimation of intrinsic image sequences from image+depth video. In *ECCV*, 2012.

[18] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3), 2004.

[19] Microsoft. Kinect. http://www.xbox.com/en-us/kinect, 2010.

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.



(a) Infrared speckle image (b) Color image (c) Original range image (d) Fields of Experts (e) Our result

Figure 4. Depth reconstruction using global optimization. (a) Infrared image of a speckle pattern projected by a consumer depth camera onto a scene, (b) color image of the scene, (c) range image produced by the camera for this scene using the standard winner-take-all algorithm, (d) range image rectified by the Fields of Experts model, which uses the color image [8], (e) range image produced by our MRF optimization approach from the infrared speckle image.

[21] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling Kinect sensor noise for improved 3D reconstruction and tracking. In *3DIMPVT*, 2012.

[22] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I.-S. Kweon. High quality depth map upsampling for 3D-TOF cameras. In *ICCV*, 2011.

[23] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, and C. Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Comput. Graph. Forum*, 31(2), 2012.

[24] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

[25] J. Shen and S.-C. Cheung. Layer depth denoising and completion for structured-light RGB-D cameras. In *CVPR*, 2013.

[26] A. Shpunt and Z. Zalevsky. Three-dimensional sensing using speckle patterns. U.S. Patent 8390821.

[27] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[28] J. Smisek, M. Jancosek, and T. Pajdla. 3D with Kinect. In *ICCV Workshops*, 2011.

[29] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *PAMI*, 30(6), 2008.

[30] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. In *CVPR*, 2007.