High Performance Graphics (2016)
Ulf Assarsson and Warren Hunt (Editors)

Supplemental Material for "A Fast, Massively Parallel Solver for Large, Irregular Pairwise Markov Random Fields"

D. Thuerck[1,2], M. Waechter[1], S. Widmer[1,2], M. von Buelow[1], P. Seemann[1], M. E. Pfetsch[1,2] and M. Goesele[1,2]

[1]TU Darmstadt [2]Graduate School of Computational Engineering; TU Darmstadt

This supplemental material is organized as follows:

In Section A we give more detailed descriptions of our datasets, a table of their key properties, all solver's final energies, as well as plots for all experiments (in contrast to the main paper, where we could only show a selection for space considerations) in larger, printable form.

Afterwards, in Section B we formally prove some of our paper's claims, mostly from Section 4.

The source code for our paper is available at `www.gcc.tu-darmstadt.de/home/proj/mapmap`.

A. Extended Dataset Description and Experimental Results

In the main paper, we kept the presentation of our used datasets concise in favor of brevity; the same holds for the discussion of experimental results. In this part of the supplemental material, we augment both sections with more details. Apart from an extended textual description, Table A.1 lists the key properties of all datasets that we used for evaluation.

A.1. Extended Dataset Description

In the following we give a short description of what applications our datasets – except for those originating from Kappes' benchmark [KAH*15] – arise from.

Plane Sweep. Plane sweeping is a standard method in stereo reconstruction. Given a base image and a set of reference images with corresponding camera parameters, the reference images are iteratively projected onto planes parallel to the base image at different depths. A photometric difference metric between these projections and the base view yields the unary costs. For regularization, one commonly assumes a fronto-parallel, piecewise-planar prior, which can be expressed by a truncated linear model. To improve runtime, the unary cost matrix can be sparsified. In our dataset, we used the "Arts" images from the Middlebury Stereo benchmark [SP07]. The unary costs are given by the cost volume, discretized into 100 labels. On average each node has 32 feasible labels. As binary costs, a truncated linear term $V_{\{i,j\}}(\ell_1,\ell_2) = \min(|\ell_1 - \ell_2|, 2)$ was used. As such, these datasets are the prototypical example of pixel-based grid topologies which were the workhorse for parallel solvers in recent years.

3D Model Texturing. Waechter *et al.* [WMG14] texture 3D reconstructions with 3D-registered input photos by letting MRFs assign photos to mesh triangles. Mesh triangles are turned into MRF nodes, mesh edges are MRF edges, input photos are labels, and the photos are assigned to triangles subject to the following: The unary costs aim at assigning a suitable (*i.e.*, ideally orthogonal, close-up, in focus) photo to a triangle and the binary costs (Potts model) punish adjacent triangles being textured from different photos. The resulting MRFs are neither planar nor a grid and they exceed 10^7 variables and 500 labels. A small example was given in Figure 1. While most nodes have a degree of 3, no regular structure can be assumed, especially since the input meshes where not hand-crafted but obtained with image-based modeling (multi-view stereo plus surface reconstruction). We use the public source code and datasets [Gra16] of Waechter *et al.*. To obtain smaller MRFs for testing, we simplified the meshes. The number after the dataset (*e.g.*, Citywall-40) is a simplification parameter: 100 means full quality, smaller means less triangles / MRF variables.

Mesh Segmentation. Segmenting a 3D mesh into different, meaningful parts is another source of irregular MRFs. According to Chen *et al.* [CGF09], Shapira *et al.*'s shape diameter function [SSCO08] is a state of the art technique. While Shapira *et al.* determine the appropriate number of labels by repeated optimization with an increasing number of labels, we minimize the label count by using label costs. Apart from that, we use the costs as determined by the shape diameter function. As meshes, we use the Stanford Dragon and Bunny [Sta], a scaled version of the Dolphin accompanying the iVRML viewer [ivr], and CGAL's shape diameter function code [CGA15].

Graph Coloring. In contrast to most computer vision applications, some applications may have anti-metric or arbitrary binary costs. In graph coloring (used, *e.g.*, in robotics [DEdW09]) the bi-

Category	Dataset	Properties							Final energies							
		Nodes	Edges	Size	Labels	Avg. labels	Topology structure	Binary cost type	Ours	BP	TRW-S	GCO	GridGCO	DGCO	FastPD	CK-BCD
From Kappes *et al.* [KAH*15]	Teddy	168,750	336,675	10,125,000	60	60	square grid	Truncated linear	713,118	711,638	**711,216**	713,708	718,347	716,563	719,860	714,997
	Tsukuba	110,592	220,512	1,769,472	16	16	square grid	Truncated linear	102,900	103,011	101,344	**101,337**	101,768	101,684	101,634	105,814
	Venus	166,222	331,627	3,324,440	20	20	square grid	Truncated linear	791,638	790,932	**790,732**	791,809	797,383	795,601	797,700	793,170
	Brain_9mm	785,540	2,309,383	3,927,700	5	5	3D grid (N6)	Potts	8,039,810	8,039,806	**8,039,800**	8,039,802	-	8,039,800	8,039,810	-
	Brain_5mm	1,413,972	4,188,311	7,069,860	5	5	3D grid (N6)	Potts	14,084,900	**14,084,892**	14,084,900	**14,084,892**	-	14,085,584	14,084,900	-
	Brain_3mm	2,356,620	7,006,703	11,783,100	5	5	3D grid (N6)	Potts	8,039,810	8,039,806	**8,039,800**	8,039,802	-	8,040,216.1	8,039,810	-
	Knott-3D-150	651	2939	423,801	651	651	irregular	Weighted Potts	**-5,461.08**	-	-5,362.28	-3,892	-	-	-	-
	Knott-3D-300	4,228	26,278	17,875,984	4,228	4,228	irregular	Weighted Potts	**-27,318.2**	-	-	-23,026	-	-	-	-
	Knott-3D-450	15,393	97,275	236,944,449	15,393	15,393	irregular	Weighted Potts	**-72,318.8**	-	-	-62,485	-	-	-	-
Sparse plane sweep	Planesweep_320_256_96	81,920	163,264	4,669,440	96	57	square grid	Truncated linear	194,538	195,852.6	194,344	194,052	197,186	197,166.2	**194,048**	196,118.5
	Planesweep_640_511_96	327,040	652,929	21,911,680	96	67	square grid	Truncated linear	502,923	505,876.7	503,479	499,599	506,515	506,496.7	**499,588**	504,034.2
	Planesweep_1280_1022_96	1,308,160	2,614,018	87,646,720	96	67	square grid	Truncated linear	1,286,720	1,310,563.8	1,293,290	1,273,796	1,299,470	1,300,772	**1,273,620**	1,287,551.5
Mesh segmentation [CGA15]	Bunny	69,451	104,065	347,255	5	5	irregular	Weighted Potts	10,190.2	10,186.2	10,195.7	**10,186**	-	104,158.3	10,186.2	-
	Dolphin	836	1,254	4,180	5	5	irregular	Weighted Potts	283.9	283.9	283.9	**283**	-	-	283.9	-
	Dragon	871,306	1,306,959	4,356,530	5	5	irregular	Weighted Potts	149,316	149,165.1	149,168.0	**149,144**	-	1,119,423.6	149,144	-
	Dragon + LC	871,306	1,306,959	4,356,530	5	5	irregular	Weighted Potts + LC	**206,541**	-	-	214,277	-	-	-	-
Texturing [WMG14]	Citywall-20	733,917	1,094,069	35,228,016	561	48	irregular	Potts	693,212	698,046.9	707,733	**688,753**	-	723,578.2	688,755	-
	Citywall-40	2,431,185	3,639,981	141,008,730	561	58	irregular	Potts	2,145,690	-	2,248,830	**2,136,762**	-	2,402,473.7	-	-
	Citywall-100	8,164,823	12,240,438	661,350,663	561	81	irregular	Potts	**6,703,580**	-	-	6,826,897	-	7,965,659	-	-
	Reader-20	1,222,233	1,835,041	50,332,921	539	41	irregular	Potts	1,083,290	1,075,844.8	1,073,610	**1,070,690**	-	1,214,938.1	**1,070,690**	-
	Reader-40	5,749,316	8,625,993	298,348,004	539	52	irregular	Potts	4,672,550	-	4,752,550	**4,659,336**	-	5,675,864.1	-	-
	Reader-100	12,463,814	18,697,742	697,973,584	539	56	irregular	Potts	**9,932,820**	-	-	10,331,097	-	12,452,136	-	-
Graph Coloring	Gra._Col._10M_10D_4L	10,000,000	47,437,632	40,000,000	4	4	irregular	Anti-Potts	**605,184**	2,339,447	2,339,450	-	-	-	-	-
	Gra._Col._10M_10D_8L	10,000,000	47,431,197	80,000,000	8	8	irregular	Anti-Potts	**0**	15	15	-	-	-	-	-
	Gra._Col._10M_10D_16L	10,000,000	47,404,420	160,000,000	16	16	irregular	Anti-Potts	**0**	**0**	**0**	-	-	-	-	-
	Gra._Col._15M_10D_4L	15,000,000	71,135,959	60,000,000	4	4	irregular	Anti-Potts	**980,709**	3,514,112	3,514,110	-	-	-	-	-
	Gra._Col._15M_10D_8L	15,000,000	71,138,591	120,000,000	8	8	irregular	Anti-Potts	**0**	95	95	-	-	-	-	-
	Gra._Col._15M_10D_16L	15,000,000	71,129,488	240,000,000	16	16	irregular	Anti-Potts	**0**	**0**	**0**	-	-	-	-	-
	Gra._Col._15M_10D_16L + LC	15,000,000	71,129,488	240,000,000	16	16	irregular	Anti-Potts + LC	**0.9**	-	-	-	-	-	-	-

Table A.1: List of evaluated datasets, their key properties, and the final energies achieved by the tested solvers. "+ LC" means "with label costs". Size is the average number of valid labels per node times the number of nodes.

nary costs are *anti-Potts* models: $V_{\{i,j\}}(\ell_1,\ell_2) = w_{i,j}[\ell_1 = \ell_2]$. Additionally, we can use small label costs (*e.g.* $M_\ell = 0.1$) to minimize the number of labels used for the coloring. Our datasets are based on irregular, undirected, connected random graphs where the node degrees were sampled with a Gaussian distribution with mean 10.

We show results for all datasets in Figures A.1, A.2 and A.3. As in the main paper, energies (y-axis) are relative to the best found solution for the problem. In addition, we give the absolute energies in Table A.1. For each dataset/method pair that is feasible, the table contains the energy of the best feasible solution a method found in the time span given until manual or automatic termination.

A.2. Extended Experimental Evaluation and Discussion

Kappes' Benchmark. The benchmark results are shown in Figures A.1a–f and in Figure A.3. In the stereo datasets our performance is as expected: While the energy is comparable (difference $< 0.1\,\%$) to other solvers, there is no advantage in terms of time-to-solution. This is also due to the size of the datasets: With 16 or 20 labels and $< 200{,}000$ nodes they are relatively small and the GPU's processing groups are underutilized. We expect that an improved load balancing or node grouping for processing groups would be beneficial for this kind of datasets. For the even smaller brain instances we achieve almost the same energy as all other methods, albeit slower. With only 5 labels per node, these MRFs resemble the characteristics of the stereo datasets and thus share their explanation.

The Knott-3D datasets pose a different challenge: A relatively small number of nodes and edges, but a huge number of labels makes these datasets difficult for GPU acceleration. Due to the high number of labels, the dynamic programming table cannot be cached in shared memory, hence all computations are executed on global memory, severely hurting performance. Nevertheless, the GPU dominates the other algorithms in terms of solution quality and speed. This is in part due to the fact that a high number of labels results in a higher amount of readily-available parallelism, thereby countering the quadratic complexity in the number of labels per node. In addition, these datasets are the only datasets included that have positive and negative costs, expressed as edge weights. In practice, this is no limitation for our solver.

Plane Sweep. The results in Figures A.1g–i show that the solver's performance does not vary much between the different dataset sizes. Since the label count per node is roughly constant among the three versions, all statements about solution quality and speed hold for all three datasets. As parallel solvers such as GRIDGCO are tuned for this type of input data, we did not expect our general-purpose solver to make any significant win here. At least for the largest plane sweep case, there is enough work to keep all processing elements busy; hence in terms of potential, we are already approaching the upper bound as determined by the algorithm and hardware. Hence, only a completely different algorithm might have a shot at being a leap forward for these regular datasets. The energy differences after convergence between our and the best solver (GCO) are not crucial for the application. Meltzer *et al.* [MYW05] analyzed depth maps in a similar setting and concluded that instead of the "last percent" in energy one should rather strive to improve the stereo model. A visual comparison in Section A.3 confirms this observation. Besides, stereo belongs to a group of applications (for more, see Kappes *et al.* [KAH*15]) that suffer from weak LP relaxations, hence the advantage for primal solvers.

Mesh Segmentation. The *Dragon* dataset's size is similar to Kappes' *Venus*. As Figure A.1j shows, we outperform all solvers in speed and quality except for BP and TRW-S, who reach a slightly better quality in comparable time. Since only 5 labels are used, the GPU is heavily underutilized. Again, fine-grained load balancing could result in a big speedup. *Bunny* and *Dolphin* yield even smaller MRFs. This reflects on the results (Figures A.1k–l): Whereas for the *Dragon* our solution was slightly worse than the competitors' but in roughly the same time, we perform worse speed-wise for *Bunny* and *Dolphin*. Profiling the GPU version with NVIDIA's tools showed that our solver's overhead (*e.g.*, GPU setup time) dominates algorithm runtime and our approach seems unsuitable for very small datasets.

Texturing. The sizes of the different texturing datasets vary by more than one order of magnitude. Regarding their node count, *Reader-20* and *Citywall-20* are comparable with Kappes' benchmark datasets and our plane sweep data. However, their number of labels exceeds those by at least a factor of 5 whereas their cost matrix is very sparse. This combination of topology size, huge number of labels and sparsity is challenging for most solvers. Citywall-20 and Reader-20 can still be handled by all solvers handling irregular topologies. Here, our GPU and CPU versions deliver solutions that compare favorably to most solvers in quality (with the exception of GCO and FastPD) about twice as fast, see Figures A.2a and d.

FastPD and BP are unable to handle the larger datasets *Reader-40* and *Citywall-40*, so only GCO and TRW-S are considered. The quality of the solutions mimic the behavior on the smaller datasets (see Figures A.2b and e): Our solver is slightly inferior to GCO, but delivers its comparable solution about 4 times faster. The difference in solution quality is not visible in the final, textured mesh. TRW-S' performance is significantly worse than our solvers' and GCO's.

For *Reader-100* and *Citywall-100* only our solvers and GCO were applicable due to enormous memory requirements. These datasets are ideal candidates for our algorithm: The huge node and label count offers enough parallelism to saturate the hardware, and both models prefer large, homogeneous regions. As Figures A.2c and f show, our solvers excel in both cases: Our GPU solver's solution after 1 minute is often already better than GCO's final solution after 1.5 hours. GCO's larger energy here stands in contrast to the smaller datasets. We suspect (among others) numerical issues as a potential cause: Since floating point numbers are used as costs, adding up all costs to form the objective value is subject to numerical instabilities due to lack of precision. In our parallel implementation, this addition is done as a parallel reduction, which is numerically more stable. However, in practice the energy discrepancy between GCO and our solvers may be irrelevant for the results' visual appearance; see the supplemental material's Section A.3 for a visual comparison.

Graph Coloring. Graph coloring uses anti-metric costs, which only TRW-S and BP support. Figures A.2g–l give results. As al-

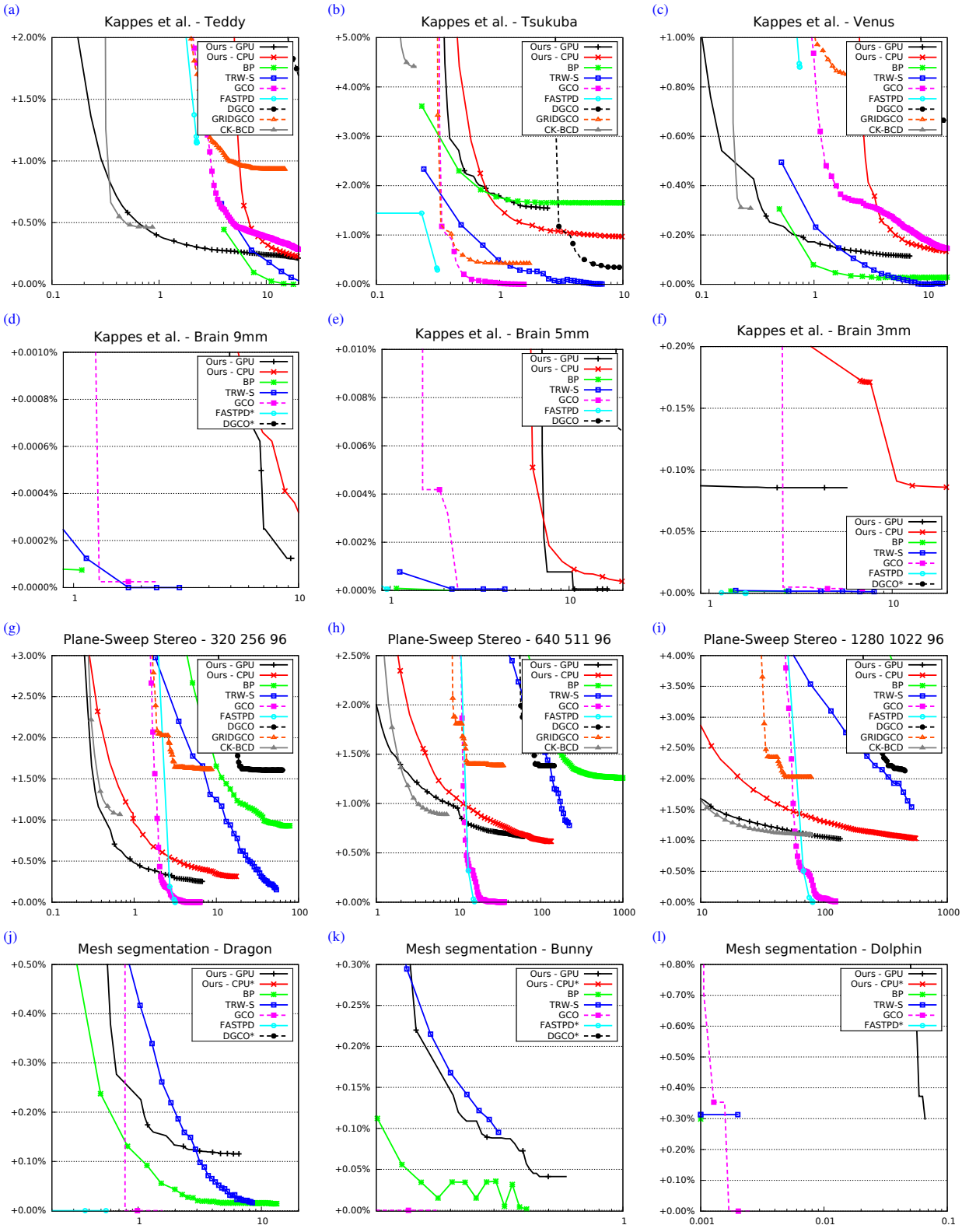

Figure A.1: Relative difference to lowest energy over logarithmic time (in seconds) for different datasets.

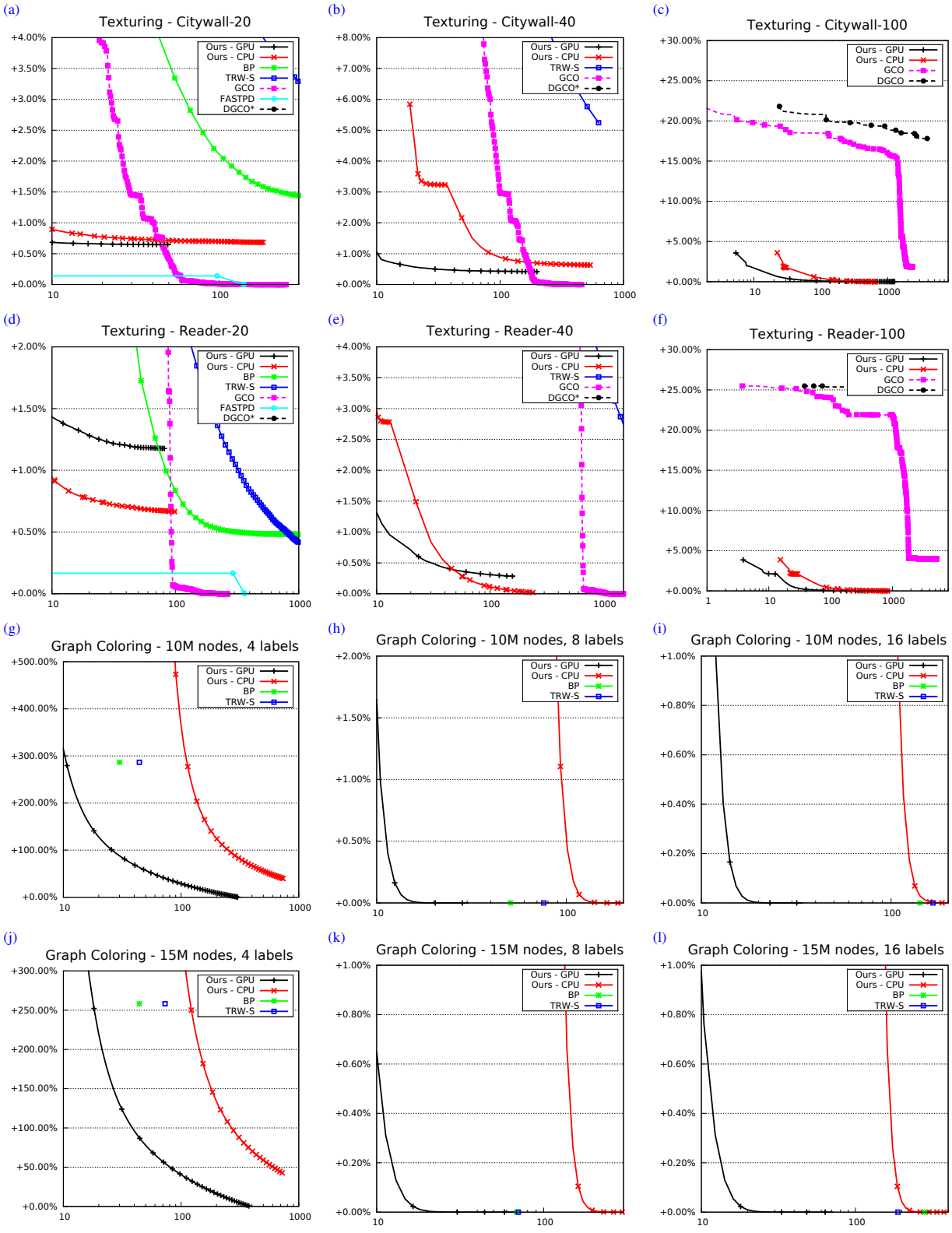

Figure A.2: Relative difference to lowest energy over logarithmic time (in seconds) for different datasets.

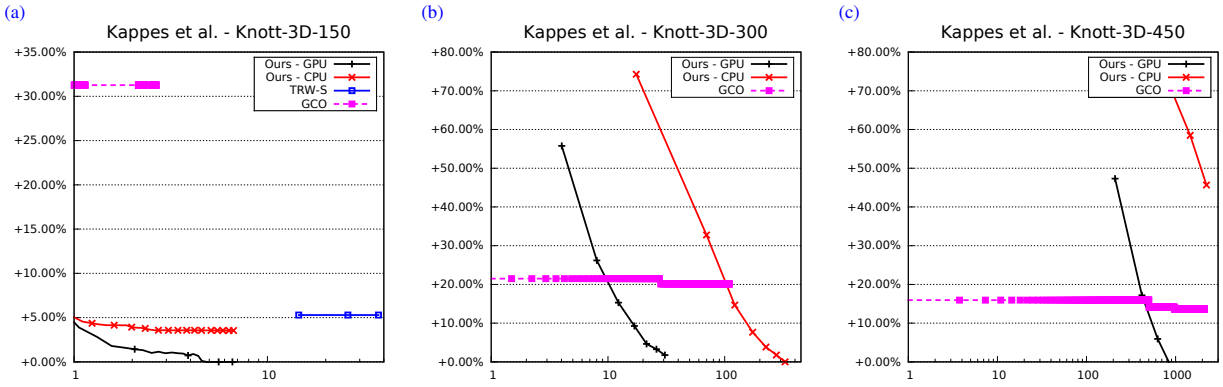

Figure A.3: Relative difference to lowest energy over logarithmic time (in seconds) for different datasets.

ready mentioned, our solver detects this cost type and skips the region graph heuristic. Instead we found our BCD scheme without heuristics to be very effective. Our solver found the best solution in all cases. Especially for the large datasets there is a good speedup. If TRW-S and BP do not find the optimal solution 0, they often converge after one iteration with a poor solution. Note that both solvers continued to run for more iterations, slowly improving the lower bound, but did not round that into a primal solution of smaller objective; for a fair comparison, we treated this case as if the solvers had terminated after finding the first primal solution. Despite the superiority of our solver in this case, there is still room for improvement: By handling multiple nodes per processing group, we could increase the number of parallel tasks available at any time as already suggested for Kappes' datasets.

A.3. Visual Quality of Results in Applications

In the plane sweep datasets, our solver's final energy was slightly inferior to other solvers such as GCO. As Meltzer [MYW05] argues, in computer vision applications one should generally prefer to tune the model instead of squeezing out the last bit of energy close to the global optimum. We want to support this view by two comparisons of solution quality. In Figure A.4, we compared the solution of our solver with GCO's. Despite the small energy difference, the solutions are both acceptable for application purposes. Note that our solver generates the solution in a matter of seconds.

Additionally, we show an example of a texturing result in Figure A.5: A comparison between GCO's result at convergence after more than 30 minutes and our result (with a lower energy than GCO) after 1 minute using the GPU. Our energy is a bit smaller than GCO's, but the differences are hardly visible in the final textured object, which supports our point from above.

B. Proofs

In this section we prove some of our paper's claims. Though it seems intuitively clear that BCD with any nonempty coordinate set that obeys the partitioning requirement leads to a sequence of solutions with monotonous decreasing energy, a proof clearly shows the consequence of not adhering to the partitioning requirement.

In the following, we first show that larger coordinate sets lead to a potentially larger energy decrease (Section B.2) and deduce the monotonicity of BCD (Section B.3) from there. Lastly, we give a short proof of the upper bound on our treatment of label costs (Section B.4).

During these proofs we concentrate on the case of a single set of feasible labels $\mathcal{L}$ shared by all nodes to keep the representation short and readable. The other cases can be proven as follows:

- For sparse cost tables: instead of summing over $\mathcal{L}$ for every node i, we sum over $\mathcal{L}_i$ for node i;
- For label costs: after each (part of the) objective add the label cost term; the sets of used labels include free and fixed variables for a given coordinate set. Note that using these proofs algorithmically with label costs would require an exact solver for label costs on trees.

B.1. Notation

First, we introduce some short-hand notation that makes the formulation of the optimization problems in the following proofs more compact and easier to read.

Sum Notation. For index sets $S'_1 \subseteq S_1$, $S'_2 \subseteq S_2$, …, $S'_n \subseteq S_n$ and a multi-array $\nu \in \{0,1\}^{S_1 \times S_2 \times \cdots \times S_n}$, we define a short-hand notation for an indexed sum

$$\nu(S'_1, S'_2, \ldots, S'_n) = \sum_{i_1 \in S'_1} \sum_{i_2 \in S'_2} \cdots \sum_{i_n \in S'_n} \nu_{i_1, i_2, \ldots, i_n}.$$

Local Polytope. The MRF labeling problem can be expressed as an integer linear program using the sets $\mathcal{P}$ and $\mathcal{N}$. As usual in the literature, we use binary variables $\nu \in \{0,1\}^{\mathcal{P} \times \mathcal{L}}$, where $\nu_{i,\ell} = 1$ ($i \in \mathcal{P}$, $\ell \in \mathcal{L}$) if and only if $f(i) = \ell$. Similarly, $\mu \in \{0,1\}^{\mathcal{N} \times \mathcal{L} \times \mathcal{L}}$ represents edges: For $\{i,j\} \in \mathcal{N}$ and ℓ_1, $\ell_2 \in \mathcal{L}$, $\mu_{\{i,j\},\ell_1,\ell_2} = 1$ if and only if $f(i) = \ell_1$ and $f(j) = \ell_2$. The equations of the so-called *local polytope* $\mathbb{P}$ describe the set of feasible labelings by coupling

Figure A.4: Depth map resulting from optimization of the Planesweep_1280_1022_96 dataset for (a) GCO and (c) our solver. (b) shows the difference image (a gray value of 128 corresponds to a difference of 0).

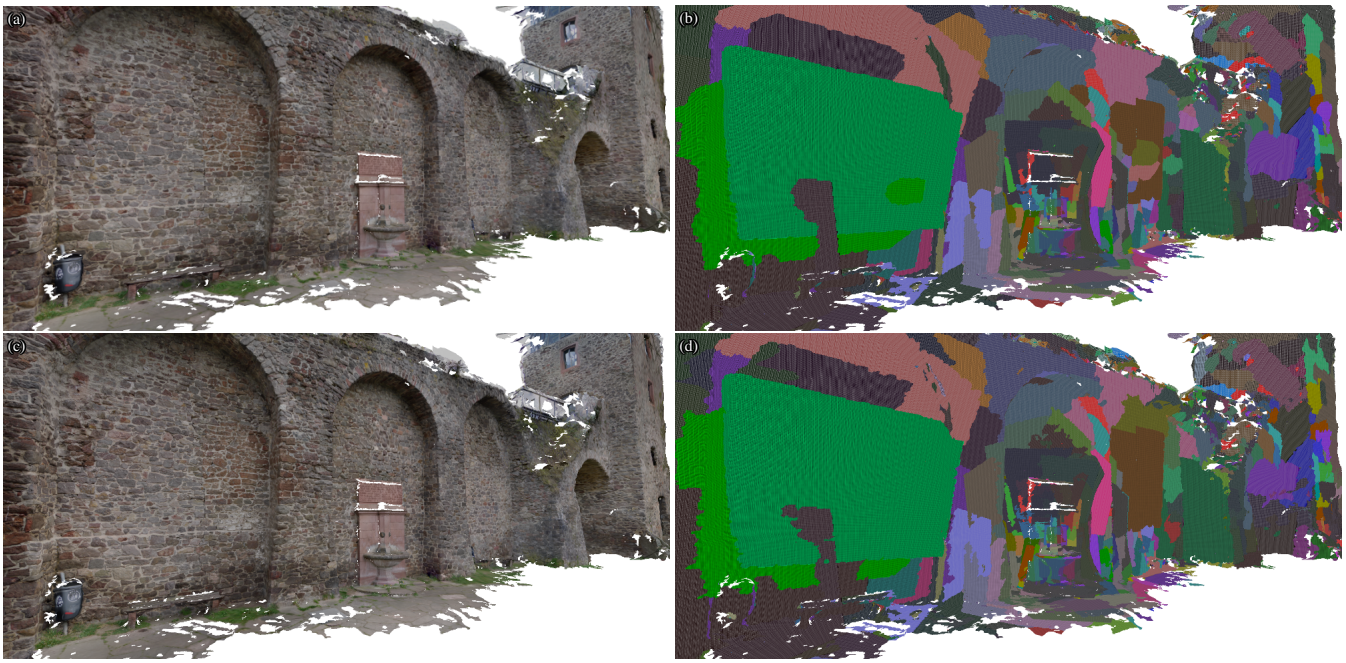

Figure A.5: Citywall-100 texturing results for (a) GCO and (c) our solver and labeling visualization for (b) GCO and (d) our solver. Each color corresponds to a specific view from which the region is textured. Both results are largely similar with only minor differences.

node and edge variables:

$$\begin{aligned} \nu(\{i\},\mathcal{L}) &= 1 && \forall i \in \mathcal{P} \\ \mu(\{i,j\},\mathcal{L},\mathcal{L}) &= 1 && \forall \{i,j\} \in \mathcal{N} \\ \mu(\{i,j\},\{\ell_1\},\mathcal{L}) &= \nu_{i,\ell_1} && \forall \{i,j\} \in \mathcal{N}, \forall \ell_1 \in \mathcal{L} \\ \mu(\{i,j\},\mathcal{L},\{\ell_2\}) &= \nu_{j,\ell_2} && \forall \{i,j\} \in \mathcal{N}, \forall \ell_2 \in \mathcal{L}. \end{aligned}$$

Objective Function. Objective (1) (without label costs) translates directly into the new domain of (ν,μ) variables:

$$E(\nu,\mu) = \sum_{i\in\mathcal{P}}\sum_{\ell\in\mathcal{L}} \nu_{i,\ell} D_i(\ell) + \sum_{\{i,j\}\in\mathcal{N}}\sum_{\ell_1\in\mathcal{L}}\sum_{\ell_2\in\mathcal{L}} \mu_{\{i,j\},\ell_1,\ell_2} V_{\{i,j\}}(\ell_1,\ell_2) + \sum_{\ell\in\mathcal{L}} \delta_\ell M_\ell.$$

B.2. Effectiveness of Larger Coordinate Sets

We start with proving the fact that

Lemma 1 In BCD, choosing a coordinate set $\mathcal{C}_2$ over $\mathcal{C}_1$, where $\mathcal{C}_1 \subseteq \mathcal{C}_2$, leads to a potentially higher decrease in energy, *i.e.* for their respective resulting assignments $\hat{f}_1, \hat{f}_2$ the relation $E(\hat{f}_2) \leq E(\hat{f}_1)$ holds.

Proof Let $\mathcal{C}$ be a non-empty coordinate set. Without loss of generality, for an edge $\{i,j\} \in \mathcal{N}_\Delta$ we can assume that $i \in \mathcal{C}$ and $j \in \mathcal{F}$ by corresponding changes in the ordering of edges considered as dependencies. In the space of the local polytope $\mathbb{P}(\mathcal{C})$, the sets $\mathcal{C}$ and $\mathcal{F}$ partition the variables into *free* (coordinates) and *fixed* variables. Accordingly, the description of the local polytope given an

assignment f changes to

$$\begin{aligned}
\nu(\{i\},\mathcal{L}) &= 1 && \forall i \in \mathcal{C} \\
\nu_{i,f(i)} &= 1 && \forall i \in \mathcal{F} \\
\mu(\{i,j\},\mathcal{L},\mathcal{L}) &= 1 && \forall \{i,j\} \in \mathcal{N}_\mathcal{C} \\
\mu(\{i,j\},\{\ell_1\},\mathcal{L}) &= \nu_{i,\ell_1} && \forall \{i,j\} \in \mathcal{N}_\mathcal{C},\ \forall \ell_1 \in \mathcal{L} \\
\mu(\{i,j\},\mathcal{L},\{\ell_2\}) &= \nu_{j,\ell_2} && \forall \{i,j\} \in \mathcal{N}_\mathcal{C},\ \forall \ell_2 \in \mathcal{L} \\
\mu(\{i,j\},\{f(i)\},\mathcal{L}) &= 1 && \forall \{i,j\} \in \mathcal{N}_\Delta \\
\mu(\{i,j\},\mathcal{L}\setminus\{f(i)\},\mathcal{L}) &= 0 && \forall \{i,j\} \in \mathcal{N}_\Delta \\
\mu_{\{i,j\},f(i),f(j)} &= 1 && \forall \{i,j\} \in \mathcal{N}_\mathcal{F}.
\end{aligned}$$

Removing constant terms yields the objective of the subproblem, see Equation (2):

$$E_\mathcal{C}(g) = \sum_{i\in\mathcal{C}} D_i(g(i)) + \sum_{\{i,j\}\in\mathcal{N}_\mathcal{C}} V_{\{i,j\}}(g(i),g(j)) + \sum_{\{i,j\}\in\mathcal{N}_\Delta} V_{\{i,j\}}(g(i),f_\mathcal{F}(j)).$$

Now let $\mathcal{C}_1,\mathcal{C}_2$ be as introduced above; both shall respect the partition requirement. The corresponding polytopes are $\mathbb{P}(\mathcal{C}_1)$ and $\mathbb{P}(\mathcal{C}_2)$. We show that $\mathbb{P}(\mathcal{C}_1) \subseteq \mathbb{P}(\mathcal{C}_2)$ (note that both polytopes live in the space of the original variables (ν,μ), only some variables are fixed to either 0 or 1):
Let $\nu_{i,\ell}$ for $i \in \mathcal{P}, \ell \in \mathcal{L}$ be a free variable in $\mathbb{P}(\mathcal{C}_1)$. Since $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $\nu_{i,\ell}$ is also free in $\mathbb{P}(\mathcal{C}_2)$. For μ-variables $\mu_{\{i,j\},\ell_1,\ell_2}$ $(\ell_1,\ell_2 \in \mathcal{L})$ we use a case distinction:

- $\{i,j\} \in \mathcal{N}_{\mathcal{C}_1}$: Since all covered links are included in the coordinate sets, $\mathcal{N}_{\mathcal{C}_1} \subseteq \mathcal{N}_{\mathcal{C}_2}$. Thus, $\mu_{\{i,j\},\ell_1,\ell_2}$ is also free in $\mathbb{P}(\mathcal{C}_2)$.
- $\{i,j\} \in \mathcal{N}_{\Delta_1}$: If $j \in \mathcal{F}_2$, then by definition $\{i,j\} \in \mathcal{N}_{\Delta_2}$ and $\mu_{\{i,j\},\ell_1,\ell_2}$ is free in $\mathbb{P}(\mathcal{C}_2)$. Similarly, this holds if $i \in \mathcal{C}_2$, since then $\{i,j\} \in \mathcal{N}_{\mathcal{C}_2}$ (dependency becomes a link). This shows that $\mathcal{N}_{\Delta_1} \subseteq \mathcal{N}_{\Delta_2} \cup \mathcal{N}_{\mathcal{C}_2}$.

Thus, every variable that is free in $\mathbb{P}(\mathcal{C}_1)$ is also free in $\mathbb{P}(\mathcal{C}_2)$. For optimal solutions g_1^*, g_2^* of the subproblems with respect to Equation (2), this clearly implies

$$E_{\mathcal{C}_1}(g_1^*) \geq E_{\mathcal{C}_2}(g_2^*). \tag{B.1}$$

For the claim, we now turn to the newly constructed solutions. To this end, we define $\hat{f}_{\mathcal{C}_1}$ and $\hat{f}_{\mathcal{C}_2}$ to be the assignments obtained by replacing the labels in $\mathcal{C}_1$ and $\mathcal{C}_2$, by their counterparts of the optimal solutions g_1^* and g_2^*, respectively. With little partitioning, we notice that Objective (1) can be written as

$$E(\hat{f}_1) = E_{\mathcal{C}_1}(g_1^*) + \underbrace{\sum_{i\in\mathcal{F}_1} D_i(\hat{f}_1(i))}_{a_1} + \underbrace{\sum_{\{i,j\}\in\mathcal{N}_{\mathcal{F}_1}} V_{\{i,j\}}(\hat{f}_1(i),\hat{f}_1(j))}_{a_2}$$

and

$$E(\hat{f}_2) = E_{\mathcal{C}_2}(g_2^*) + \underbrace{\sum_{i\in\mathcal{F}_2} D_i(\hat{f}_2(i))}_{b_1} + \underbrace{\sum_{\{i,j\}\in\mathcal{N}_{\mathcal{F}_2}} V_{\{i,j\}}(\hat{f}_2(i),\hat{f}_2(j))}_{b_2}.$$

With $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $\mathcal{N}_{\mathcal{C}_1} \subseteq \mathcal{N}_{\mathcal{C}_2}$ and $\hat{f}_1(i) = \hat{f}_2(i)\ \forall i \in \mathcal{C}_2$, we clearly have $b_1 \leq a_1$ and $b_2 \leq a_2$. Together with Inequality (B.1), this yields the claim. □

B.3. Monotonicity of BCD

Following the proof above, we now give a short proof sketch for BCD's monotonicity, *i.e.*

Lemma 2 For any coordinate set $\mathcal{C}_2$ and assignment f, BCD results in an assignment $\hat{f}$ such that $E(\hat{f}) \leq E(f)$.

Proof Use a coordinate set $\mathcal{C}_1 = \mathcal{P}$ and thus $E_{\mathcal{C}_1}(g_1^*) = E(f)$ in the proof above. □

We assumed the partitioning requirement to be satisfied above. The next section deals with the consequences of not obeying it.

B.3.1. Relaxing the Partitioning Requirement

Whenever the partitioning requirement is not satisfied, monotonicity cannot be guaranteed. Considering $E_\mathcal{C}$ in the proof above, we notice that the difference to E arises from edges included in if $\mathcal{N}$, but not in $\mathcal{N}_F$, $\mathcal{N}_\mathcal{C}$, and $\mathcal{N}_\Delta$. Thus, there is a complicating set $\mathcal{N}_{\tilde{\Delta}}$ missing to fulfill the partitioning requirement. In terms of two coordinate sets as above, between $\mathcal{N}_{\tilde{\Delta}_1}$ and $\mathcal{N}_{\tilde{\Delta}_2}$, there is no clear relation and thus no guarantee for monotonicity concerning the original objective function.

If we relax the requirement such that not all links are actually included in $\mathcal{N}_\mathcal{C}$ are used, *i.e.*,

$$\mathcal{N}_\mathcal{C} \subseteq \mathcal{N}[\mathcal{C}],$$

our notation also covers Veksler's spanning trees [Vek05] and even loopy belief propagation. (Loopy belief propagation can be expressed by optimizing on a set of coordinates whose free variables cover the original MRF's variables. In each step, we fix all variables but one free variable, which then receives messages from all neighboring nodes. Before the labeling can be updated, all such fixings must be used to update the marginals – the BCD steps are thus executed simultaneously before updating the current assignment.)

With this definition $\mathcal{N}_F$, $\mathcal{N}_\Delta$, and $\mathcal{N}_\mathcal{C}$ are not a partition of $\mathcal{N}$. Therefore Objective (2) is not directly comparable to Objective (1) anymore, *i.e.*, $E_{\mathcal{C}_1}(g_1) \leq E_{\mathcal{C}_2}(g_2)$ does not imply $E(\hat{f}_1) \leq E(\hat{f}_2)$. In this case, the only possible statement applies to a changed energy function

$$\tilde{E}_\mathcal{C}(\hat{f}) = E_\mathcal{C}(g) + \sum_{i\in\mathcal{F}} D_i(\hat{f}(i)) + \sum_{\{i,j\}\in\mathcal{N}_\mathcal{F}} V_{\{i,j\}}(\hat{f}(i),\hat{f}(j)),$$

where g equals $\hat{f}$ on $\mathcal{C}$ and using arguments analogous to the proof above.

The difference between $\tilde{E}_\mathcal{C}$ and E basically results from the edges dropped by removing the partition constraint:

$$E(\hat{f}) = \tilde{E}_\mathcal{C}(\hat{f}) + \sum_{\{i,j\}\in\mathcal{N}_{\tilde{\Delta}}} V_{\{i,j\}}(\hat{f}(i),\hat{f}(j)), \tag{B.2}$$

where $\mathcal{N}_{\tilde{\Delta}}$ partitions $\mathcal{N}$ with $\mathcal{N}_\mathcal{C}$, $\mathcal{N}_\mathcal{F}$, and $\mathcal{N}_\Delta$.

Inserting augmented solutions $\hat{f}_1$ (from g_1^*) and $\hat{f}_2$ (from g_2^*) into Equation (B.2) yields

$$E(\hat{f}_1) = \tilde{E}_{\mathcal{C}_1}(\hat{f}_1) + \underbrace{\sum_{\{i,j\}\in\mathcal{N}_{\tilde{\Delta}_1}} V_{\{i,j\}}(\hat{f}_1(i),\hat{f}_1(j))}_{t_1}$$

as well as

$$E(\hat{f}_2) = \tilde{E}_{\mathcal{C}_2}(\hat{f}_2) + \underbrace{\sum_{\{i,j\} \in \mathcal{N}_{\tilde{\Delta}_2}} V_{\{i,j\}}(\hat{f}_2(i), \hat{f}_2(j))}_{t_2}.$$

While $\tilde{E}_{\mathcal{C}_2}(\hat{f}_2) \leq \tilde{E}_{\mathcal{C}_1}(\hat{f}_1)$ is easy to see, $t_2 \leq t_1$ cannot be proved since it was not included in the optimization process. Hence, no guarantees on the original objective can be given. This is why loopy belief propagation or Veksler's trees do not offer a monotonicity guarantee.

B.4. Label Cost Approximation Bounds

Lastly, we prove Proposition 1:

Proof The function $\tilde{E}$ drops nonnegative terms from E. Thus, we have $\tilde{E}(f) \leq E(f)$. Since $\tilde{f}$ is optimal for Equation (5), $\tilde{E}(\tilde{f}) \leq \tilde{E}(f)$. Moreover, f is optimal for Equation (1) and therefore $E(f) \leq E(\tilde{f})$. Thus, the first two inequalities follow. By setting all δ_ℓ in $\tilde{f}$ to 1, the last inequality holds. □

References

[CGA15] CGAL PROJECT: *User and Reference Manual*. 2015. 1, 2

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *TOG* (2009). 1

[DEdW09] DEMANGE M., EKIM T., DE WERRA D.: A tutorial on the use of graph coloring for some problems in robotics. *European Journal of Operational Research* (2009). 1

[Gra16] GRAPHICS, CAPTURE AND MASSIVELY PARALLEL COMPUTING GROUP: MVE datasets. www.gcc.tu-darmstadt.de/home/proj/mve/, 2016. Accessed: 2016-03-14. 1

[ivr] iVRML viewer examples. www.vakuumverpackt.de/ivv/#examples. Accessed: 2016-03-14. 1

[KAH*15] KAPPES J. H., ANDRES B., HAMPRECHT F. A., SCHNÖRR C., NOWOZIN S., BATRA D., KIM S., KAUSLER B. X., KRÖGER T., LELLMANN J., KOMODAKIS N., SAVCHYNSKYY B., ROTHER C.: A comparative study of modern inference techniques for structured discrete energy minimization problems. *IJCV* (2015). 1, 2, 3

[MYW05] MELTZER T., YANOVER C., WEISS Y.: Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV* (2005). 3, 6

[SP07] SCHARSTEIN D., PAL C.: Learning conditional random fields for stereo. In *CVPR* (2007). 1

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer* (2008). 1

[Sta] STANFORD COMPUTER GRAPHICS LABORATORY: The Stanford 3D scanning repository. http://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2016-03-14. 1

[Vek05] VEKSLER O.: Stereo correspondence by dynamic programming on a tree. In *CVPR* (2005). 8

[WMG14] WAECHTER M., MOEHRLE N., GOESELE M.: Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV* (2014). 1, 2