Parallel Cut Pursuit For Minimization of the Graph Total Variation

Raguet Hugo, Loic Landrieu

Parallel Cut Pursuit For Minimization of the Graph Total Variation

Raguet Hugo [1] **Landrieu Loic** [2]

Abstract

We present a parallel version of the cut-pursuit algorithm for minimizing functionals involving the graph total variation. We show that the decomposition of the iterate into constant connected components, which is at the center of this method, allows for the seamless parallelization of the otherwise costly graph-cut based refinement stage. We demonstrate experimentally the efficiency of our method in a wide variety of settings, from simple denoising on huge graphs to more complex inverse problems with nondifferentiable penalties. We argue that our approach combines the efficiency of graph-cuts based optimizers with the versatility and ease of parallelization of traditional proximal splitting methods.

1. Introduction

In 2017, (Landrieu & Obozinski, 2017) introduced the cut-pursuit algorithm, a working-set algorithm for minimizing functionals involving the total variation structured by a graph $G = (V, E, w)$, $w \in \mathbb{R}_+^E$ being edge weights:

$$F : x \in \Omega^V \mapsto f(x) + \sum_{(u,v)\in E} w_{(u,v)} \left\| x_u - x_v \right\| \ . \quad (1)$$

where $x = (x_v)_{v\in V} \in \Omega^V$ is the variable of interest and Ω is some base space, typically $\mathbb{R}$ or $\mathbb{R}^n$, $n \in \mathbb{N}^*$. The core idea of this approach is to exploit the *coarseness* of the solution, *i.e.* the fact that it can be decomposed into a small number of constant connected graph components of the graph compared to the total number of vertices. This method was further refined by Raguet & Landrieu (2018), allowing for the efficient regularization of a larger class of functionals. In particular, they dropped the convexity requirements in the optimality analysis, and extended the range of function f to extended directionally differentiable functions with a possibly nondifferentiable part which is separable along the graph G.

The cut-pursuit algorithm starts by associating all vertices to the same constant connected component and then repeats the two following steps: *reduction* and *refinement*. In the reduction step, the problem is solved under the constraint that all vertices in a same component have the same value. On many problems, when the number of constant connected components is smaller than the number of vertices in the graph, the reduced problem can be solved more efficiently than the original one, as it has fewer variables. In the refinement step, new "degrees of freedom" are added by splitting each constant connected component into smaller ones, based on a subproblem involving the directional derivatives of the functionals and which can be solved by graph cuts. This process is illustrated in Figure 1.

This scheme provably converges to a critical point in a finite number of steps; ensuring global optimality in the convex case. When the number of constant connected components of the solution is actually small, only a few iterations are needed for convergence. Succinctly, this approach allows to solve a wide range problems regularized by the graph total variation, in a little more than a few graph cuts.

In contrast, classical first-order proximal algorithms require many iterations, each of them updating all variables. Still, they attract considerable attention, notably because they can be preconditioned and easily parallelized; see for instance (Raguet & Landrieu, 2015; Möllenhoff et al., 2018; Kumar et al., 2015; Padilla et al., 2017; Barbero & Sra, 2017). On the other hand, the efficient use of graph cuts is reminiscent of the method of (Chambolle & Darbon, 2009), which uses a parametric maximum flow formulation. However, the latter can only be used to compute the *proximity operator* of the graph total variation (also called "graph total variation denoising", or "graph fused LASSO signal approximation"), that is f restrained to a sum of square differences. Not only can our method handle a much more general class of problems, but now that it can be parallelized, it combines the advantages of both proximal and graph cuts-based methods.

[1]INSA Centre Val-de-Loire Université de Tours, LIFAT, France [2]Univ. Paris-Est, IGN-ENSG, LaSTIG, STRUDEL, Saint-Mandé, France. Correspondence to: Raguet Hugo <hugo.raguet@lilo.org>.

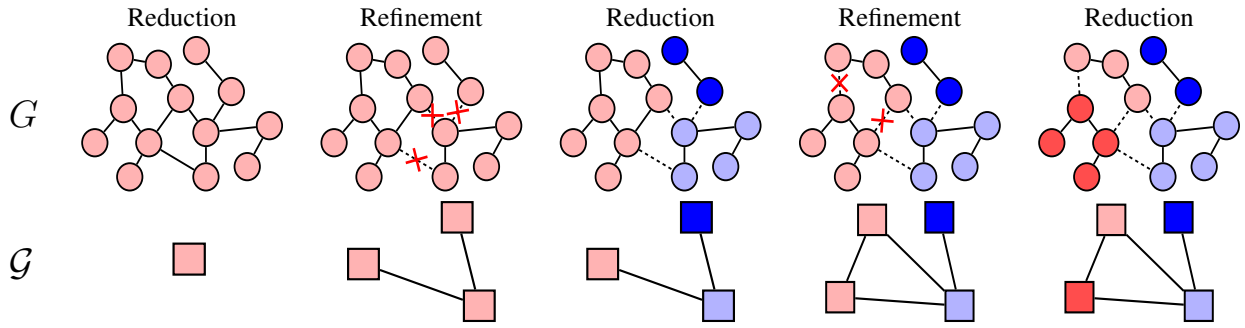


Figure 1. Illustration of the cut pursuit algorithm. The vertices of a graph G are initially combined into a single component. This graph is recursively split into constant connected components. At each iteration, a reduced graph $\mathcal{G}$ encoding the adjacency between constant components is computed. The values associated with each constant components can be computed using the smaller reduced graph. The vertex colors represent their values, from *low* in blue to *high* in red.

2. Method

2.1. Reduced Problem

We consider $\mathcal{V} = \{U_1, \dots, U_{|\mathcal{V}|}\}$ a partition of the vertices V of graph G. We define the *reduced problem* as minimizing the objective functional F under the constraint that the variable is constant over each component of the partition:

$$F^{(\mathcal{V})} : \xi \in \Omega^{\mathcal{V}} \mapsto F(\sum_{U \in \mathcal{V}} \xi_U \otimes 1_U) \,,$$

where $\xi_U \otimes 1_U \in \Omega^V$ is a notation for $(\xi_U \otimes 1_U)_v = \xi_U$ if $v \in U$, 0 otherwise. In many problems, $\xi \mapsto f(\sum_{U \in \mathcal{V}} \xi_U \otimes 1_U)$ has regularity and structure similar to f. But since the reduced problem can be rewritten as a problem of only $|\mathcal{V}|$ variables, $F^{(\mathcal{V})}$ is easier to solve than F, typically with preconditioned first-order proximal splitting algorithms.

2.2. Refinement

The goal of the refinement step in the cut pursuit algorithm is to split the current partition $\mathcal{V}$ such that the solution of the next reduced problem can decrease the objective functional F as much as possible. This is done by considering first-order information through finding a steepest directional derivative of F at the current iterate $x = \sum_{U \in \mathcal{V}} \xi_U \otimes 1_U$, where ξ is the solution of the last reduced problem:

$$d^{(x)} \in \operatorname*{arg\,min}_{d \in \Omega^V} F'(x, d) \,.$$

Interestingly, to make the problem more tractable, the space of directions to consider can be restricted to a *finite set* D. Landrieu & Obozinski (2017) show that if f is differentiable and $\Omega = \mathbb{R}$, $D = \{-1, 1\}^V$ is sufficient to retain optimality at convergence. Raguet & Landrieu (2018) extend this result to functions f with a nondifferentiable part which is separable along the graph G, with $D = \{-1, 0, 1\}^V$. Furthermore, they provide heuristic direction sets $D^{(x)} = \times_{v \in V} D_v^{(x)}$ for multidimensional Ω.

The search for a steepest descent direction restricted to a finite set D defines a new partition by computing the cross partition between the current partition and the maximal constant connected components of $d^{(x)}$. This problem is a combinatorial optimization problem involving unary and binary terms:

$$F'(x, d) = \sum_{v \in V} \delta(x, d_v) + \sum_{(u,v) \in E_=^{(x)}} w_{u,v} \, \|d_u - d_v\| \,, \quad (2)$$

where $E_=^{(x)} = \{(u, v) \in E \mid x_u = x_v\}$. In the monodimensional case, this problem is equivalent to finding a *minimum cut* in a convenient *flow graph*; in the multidimensional case, it can be solved approximately *via* a series of graph cuts.

2.3. Parallelization

In practice, the refinement step is often the computational bottleneck of the cut-pursuit algorithm. While there exists very efficient graph cut solvers, they require a lot of memory and are not well-suited to parallelization.

Now, since the only binary terms in (2) are for edges *within* the constant components of x, the steepest descent problem is separable along the components of $\mathcal{V}$:

$$F'(x, d) = \sum_{U \in \mathcal{V}} F'_U(x_U, d_U) \,, \quad \text{with for all } U \in \mathcal{V},$$

$$F'_U(x_U, d_U) = \sum_{v \in U} F'(x_v, d_v) + \sum_{(u,v) \in E_=^{(x)} \cap U^2} w_{u,v} \|d_u - d_v\| \,,$$

where we note $d_U = \{d_v\}_{v \in U}$. Consequently, this problem can be decomposed into finding the steepest descent direction $d_U^{(x)} \in D_U^{(x)} = \times_{v \in U} D_v^{(x)}$ in each component U independently. This allows us to perform the graph cuts in parallel with only a slight adaptation of *augmenting path* graph cut methods such as the one of Boykov and Kolmogorov (2004) (Boykov & Kolmogorov, 2004), and with no supplementary memory cost.

2.4. Balancing the Parallel Workload Distribution

The above modifications already allows us to significantly speed up the cut-pursuit algorithm. However, thread utilization can still be improved. Indeed, when performing the split step in parallel along components, each component is assigned to a single thread, thus the computation is at least as long as the time required for splitting the hardest one, typically the largest. If the partition is unbalanced, this can lead to most threads being idle while the last one finishes. Consider in particular the very first iteration, in which the partition only has one component.

A naive approach would be to cap the maximal number of vertices in each component of $\mathcal{V}$ according to the problem size and the number of available threads; coherence with the graph structure could be enforced by greedily constructing each component with breadth-first search, until the component is exhausted, or until the cap is reached. However, such arbitrary *balancing components* might not be beneficial for the reduced problem: in nonconvex settings, they might lead to bad local minima, and even in convex settings, they might not be consistent with the structure of the main problem and both slow down and reduce the accuracy of the solution.

For this reason, we advocate to decompose large components *only for the split step*; then modifying the augmenting path algorithm accordingly to perform the graph cuts in parallel along these components. In a nutshell, this corresponds to isolating the balancing components by setting their border edge capacities to 0 instead of $w_{u,v}\,\|d_u - d_v\|$ as they would be normally set to in the steepest direction problem. Unfortunately, the corresponding minimum cuts now solve only approximately the original problem.

As edges separating the balancing components are ignored by these parallel graph cuts, the question remains as how to define the new partition of a large component from the steepest directions found for its balancing components. Observe that all vertices of a component U share the same set of candidate directions $D_U^{(x)}$, even in the multidimensional setting where it might change from one component to another. Consequently, the balancing components partitioning a given large component also share the same direction set. We can can thus refine U from $d_U^{(x)}$ regardless of the presence of balancing components.

Note that this is not the only possible strategy. Depending on the problem at hand, it might still be relevant to apply the naive approach, or to compensate the discarded edge capacities by adjusting unary terms. However, the strategy presented here provided the most consistent improvement across our numerical experiments (alternative not shown).

We stress that solving only approximately the steepest direction problem invalidates the theoretical guarantees of the monodimensional setting, and might degrade the approximation further in the multidimensional setting. Although our experiments display this effect, they also show that such solutions remain satisfying at usual precision levels.

Algorithm 1 Parallel Cut Pursuit (PCP)

Initialize: $\mathcal{V} = \{V\}$
repeat
  - - - - - *reduced problem* - - - - -
  Find: $\xi^{(\mathcal{V})}$ stationary point of $F^{(\mathcal{V})}$
  $x \leftarrow \sum_{U\in\mathcal{V}} \xi_U^{(\mathcal{V})} \otimes 1_U$
  - - - - - *parallel refinement* - - - - -
  for all $U \in \mathcal{V}$ **do in parallel**
    Find: $d_U^{(x)} \in D_U^{(x)}$ minimizing $F'_U(x, d_U)$
    $\mathcal{U} \leftarrow$ max. constant connected components of $d_U^{(x)}$
    $\mathcal{V} \leftarrow \mathcal{V}\backslash\{U\} \cup \mathcal{U}$
  end for
until $\mathcal{V}$ does not change

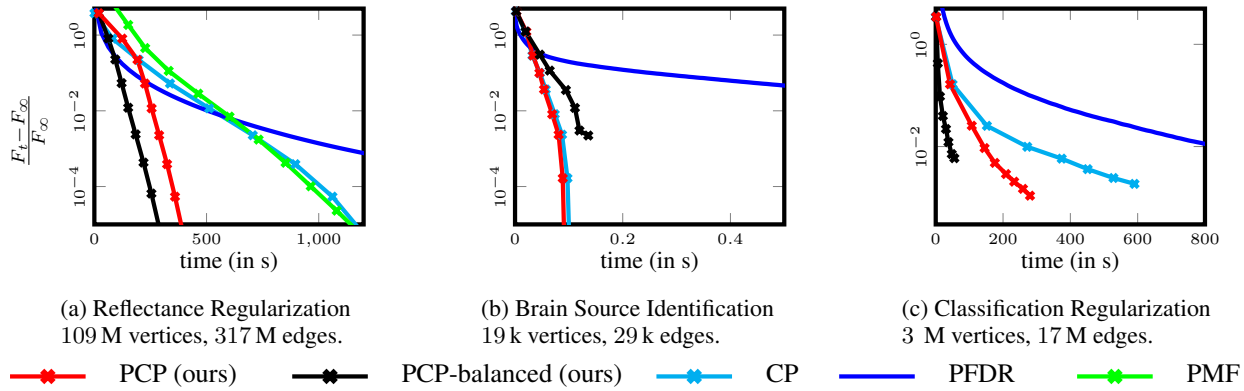
Let us finally note that there is still room for many improvements. For instance, finding rationales replacing the greedy breadth-first construction of the balancing components, or even randomizing it along iterations, might alleviate further the suboptimality. More importantly, in-depth study of the graph cuts complexity might suggest better parallel scheduling than a balancing only based on the number of involved vertices. This exploration is however beyond the scope of the present work, and left for future research.

3. Numerical Experiments

In this section, we show that our algorithm outperforms not only the highly specialized parametric maximum flow approach for the simpler problem of the proximity operator of the graph total variation, but also flexible preconditioned proximal algorithms on ill-conditioned problems involving several nondifferentiable regularizers beyond the total variation. All experiences are run on a 14-cores, 28-threads i9-7940X CPU 3.10GHz with 64 GB RAM.

Throughout this section, we compare our approach to the following state-of-the-art algorithms:

- **PMF:** the parametric max flow-based algorithm of Chambolle & Darbon (2009) for the proximity operator of the graph total variation;
- **PFDR:** the preconditioned Forward-Douglas–Rachford splitting algorithm of Raguet (2018), with all proximal steps parallelized;
- **CP:** the cut pursuit algorithm of Raguet & Landrieu (2018), using PFDR to solve the reduced problem;
- **PCP:** our proposed parallelization of CP, without balancing the parallel workload distribution;



(a) Reflectance Regularization
109 M vertices, 317 M edges.

(b) Brain Source Identification
19 k vertices, 29 k edges.

(c) Classification Regularization
3 M vertices, 17 M edges.

Figure 2. Objective functional against the running time of the algorithms; optimal values are estimated by longest, high-precision runs.

- **PCP-balanced:** PCP with balancing.

Point Cloud Reflectance Regularization: We consider the problem of spatially regularizing LiDAR reflectance on a large point cloud acquired with a mobile scanning vehicle as described by Paparoditis et al. (2012). The problem is defined as a graph-total variation denoising problem $f : x \mapsto \|x - y\|^2$ with y the observed noisy reflectance. G is the 5-nearest-neighbors graph of the point cloud ($|V| = 109\,412\,178$ and $|E| = 317\,096\,212$).

Brain Source Identification in Electroencephalography We consider the inverse problem of brain source identification in Electroencephalography. The brain of a patient is mapped to a triangular mesh with adjacency structure $G = (V, E)$ with $|V| = 19\,626$ and $|E| = 29\,439$. A set of $N = 91$ electrodes records the brain activity $y \in \mathbb{R}^N$ of the patient and the goal is to retrieve the activity on the mesh G. The relationship between the electrodes output and the brain activity is given by the *lead-field* operator $\phi : \mathbb{R}^V \mapsto \mathbb{R}^N$. To model the regularity, sparsity and positivity of brain signals, we chose

$$f : x \mapsto \frac{1}{2} \|y - \phi x\|^2 + \sum_{v \in V} \left(\lambda_v |x_v| + \iota_{\mathbb{R}_+}(x_v) \right) ,$$

with λ_v the parameters of the weighted LASSO regularization and $\iota_{\mathbb{R}_+}$ the set indicator function of $\mathbb{R}_+$.

Point Cloud Classification Regularization We consider the problem of spatially regularizing a noisy semantic labeling of a point cloud with class set K (Hackel et al., 2017). G is the 10-nearest-neighbors graph of the point cloud, ($|V| = 3\,000\,111$ and $|E| = 17\,206\,938$). A noisy classification $y \in \mathbb{R}^{V \times K}$ is obtained from a random forest classifier trained on handcrafted geometric features as described by Guinard & Landrieu (2017). Noting ι_{Δ_K} the convex indicator of the standard simplex in dimension $|K|$, and $\mathrm{KL}(r, s) = \sum_{k \in K} r_k \log(r_k / s_k)$ the *Kullback–Leibler divergence*, we choose

$$f : x \mapsto \mathrm{KL}\left(\beta u + (1-\beta) y_v, \beta u + (1-\beta) x_v\right) + \sum_{v \in V} \iota_{\Delta_K}(x_v),$$

with $u = (1/|K|)_{k \in K}$ the uniform discrete distribution and β a smoothing constant (taken as 10^{-1}).

We report performances in Figure 2. All cut-pursuit approaches significantly outperform PFDR. Our parallelization scheme further increases this gap by a large margin for the large-scale point cloud regularization problems, even outperforming the specialized PMF. In contrast, parallelization brings virtually no gain for the medium-scale brain source identification problem. Then, balancing the parallel workload distribution of the split step is beneficial in terms of speed, in particular for the classification regularization problem, in which one of the components (the road in the middle of the scene) comprises half of the vertices. However, it can be seen that the balanced version stops at a suboptimal solution, which is nonetheless close to solutions found by the others (data not shown). Finally for the brain source identification problem, our balancing method is detrimental both in terms of speed and of accuracy. Our interpretation is that it misses the important sparsity structure of the problem and the strong correlation between the variables introduced by the lead-field operator.

Conclusion

We introduce the first fully parallel graph-cut based approach for graph total variation minimization. It combines the advantages of traditional first-order proximal splitting algorithms and specialized graph cut-based methods. It significantly outperforms the state-of-the-art on many of the currently used total variation-regularization formulations. We provide an implementation of our method, in C++ parallelized with OpenMP, with interfaces for GNU Octave, Matlab and Python, at one of the authors GitHub repository `/1a7r0ch3/parallel-cut-pursuit`.

References

Barbero, A. and Sra, S. Modular proximal optimization for multidimensional total-variation regularization. *The Journal of Machine Learning Research (JMLR)*, 2017.

Boykov, Y. and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1124–1137, 2004.

Chambolle, A. and Darbon, J. On total variation minimization and surface evolution using parametric maximum flows. *International journal of computer vision*, 84(3):288, 2009.

Guinard, S. and Landrieu, L. Weakly supervised segmentation-aided classification of urban scenes from 3d lidar point clouds. In *ISPRS Workshop*, 2017.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pp. 91–98, 2017.

Kumar, K., Barbero, A., Jegelka, S., Sra, S., and Bach, F. Convex optimization for parallel energy minimization. *arXiv preprint arXiv:1503.01563*, 2015.

Landrieu, L. and Obozinski, G. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4):1724–1766, 2017.

Möllenhoff, T., Ye, Z., Wu, T., and Cremers, D. Combinatorial preconditioners for proximal algorithms on graphs. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Padilla, O. H. M., Sharpnack, J., Scott, J. G., and Tibshirani, R. J. The dfs fused lasso: Linear-time denoising over general graphs. *The Journal of Machine Learning Research (JMLR)*, 18:176–1, 2017.

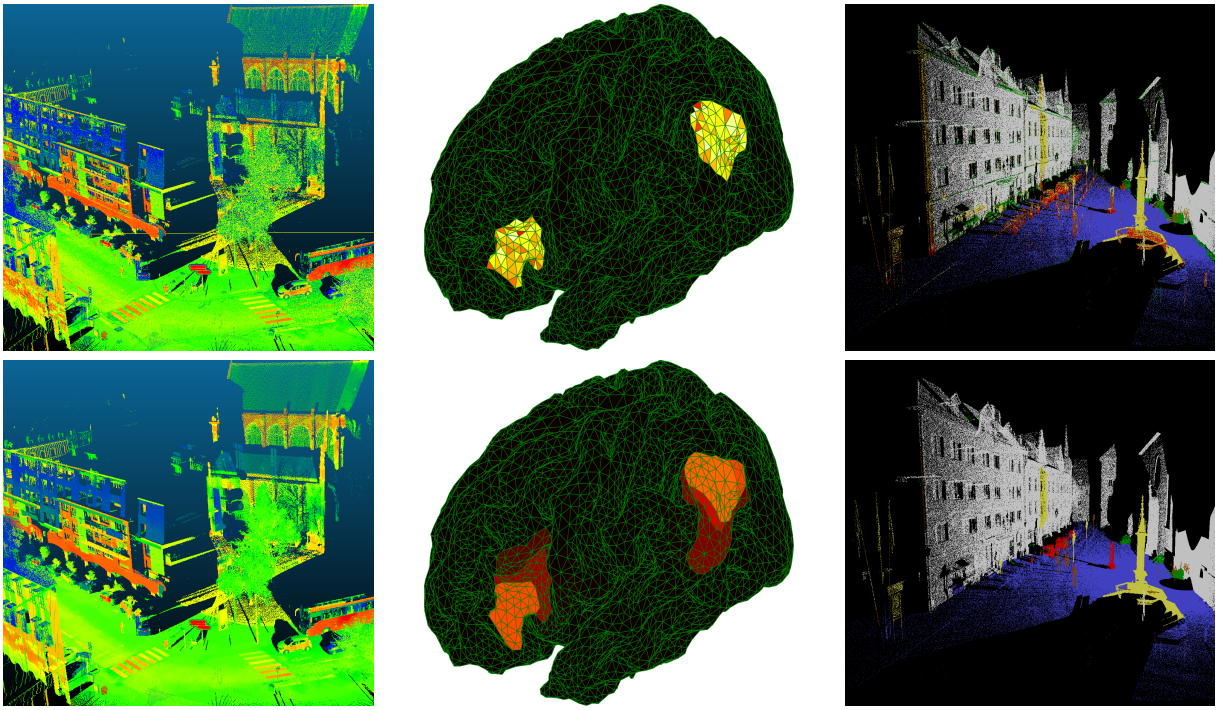
Paparoditis, N., Papelard, J.-P., Cannelle, B., Devaux, A., Soheilian, B., David, N., and Houzay, E. Stereopolis ii: A multi-purpose and multi-sensor 3d mobile mapping system for street visualisation and 3d metrology. *Revue française de photogrammétrie et de télédétection*, 200(1):69–79, 2012.

Raguet, H. A note on the forward-douglas–rachford splitting for monotone inclusion and convex optimization. *Optimization Letters*, pp. 1–24, 2018.

Raguet, H. and Landrieu, L. Preconditioning of a generalized forward-backward splitting and application to optimization on graphs. *SIAM Journal on Imaging Sciences*, 8(4):2706–2739, 2015.

Raguet, H. and Landrieu, L. Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. In *International Conference on Machine Learning (ICML)*, 2018.

APPENDIX



(a) Reflectance regularization (b) Brain source identification (c) Classification regularization

Figure 3. Illustration of the solutions given by the cut-pursuit algorithms; note that there is no discernable differences between the different versions. (a) noisy and regularized point cloud reflectance (detail of a much larger scene); (c) synthetic and recovered ground truth brain activity; (b) noisy prediction and regularized point cloud classification. Images best seen on a monitor.