Low Compute and Fully Parallel Computer Vision with HashMatch

Sean Ryan Fanello[1*] Julien Valentin[1*] Adarsh Kowdle[1] Christoph Rhemann[1]
Vladimir Tankovich[1] Carlo Ciliberto[2] Philip Davidson[1] Shahram Izadi[1]

perceptiveIO[1] University College London[2]

Abstract

Numerous computer vision problems such as stereo depth estimation, object-class segmentation and foreground/background segmentation can be formulated as per-pixel image labeling tasks. Given one or many images as input, the desired output of these methods is usually a spatially smooth assignment of labels. The large amount of such computer vision problems has lead to significant research efforts, with the state of art moving from CRF-based approaches to deep CNNs and more recently, hybrids of the two. Although these approaches have significantly advanced the state of the art, the vast majority has solely focused on improving quantitative results and are not designed for low-compute scenarios. In this paper, we present a new general framework for a variety of computer vision labeling tasks, called HashMatch. Our approach is designed to be both fully parallel, i.e. each pixel is independently processed, and low-compute, with a model complexity an order of magnitude less than existing CNN and CRF-based approaches. We evaluate HashMatch extensively on several problems such as disparity estimation, image retrieval, feature approximation and background subtraction, for which HashMatch achieves high computational efficiency while producing high quality results.

1. Introduction

Since the groundbreaking work of Krizhevsky et al. [29], deep learning is now the method of choice for a variety of computer vision problems. Although significant efforts have been undertaken to improve the performance of CNNs on various labeling tasks, these models are still far from being computationally efficient. Most of the work on efficient deep learning focuses on trying to compress deep models without losing precision and accuracy [38, 20, 11]. For example in [38], the authors train a deep architecture making use of binary weights for both the input and the filters. In [20], the authors try to remove redundant connections and force multiple neurons to share the same quantized weights. Others like [22] attempt at designing more compact layers that use a reduced number of parameters. Similar to [38], methods proposed in [11, 12] try to binarize the full network. However, these solutions still require many computational layers that involve multiple convolutions to infer per-pixel labels. Although there is an improved efficiency from the computational perspective, they suffer from accessing image patches stored in memory multiple times, and hence these algorithms are both memory and computationally bound.

Prior to the deep learning era, Conditional Random Fields (CRFs) were used as one of the major tools for image labeling problems. In their 'simplest' form, CRFs are composed of a pairwise term, encouraging structural coherence in the solution and a unary term, which is responsible for modeling the compatibility between each data point/pixel and a pre-defined set of labels. Machine learning is commonly used to predict this compatibility function. It is accepted that the more sophisticated the unary potential, the better the results obtained after solving the CRF. As a consequence, practitioners tend to use expensive feature representations, e.g. HOG [13], SIFT [30] or even deep-learning intermediate representations, followed by advanced classifiers such as kernel-SVM [42]. Once the unary potential has been estimated for each pixel, the actual CRF inference can be performed. Unfortunately, solving multi-label CRFs is a NP hard problem [7]. The high demand for fast and accurate solvers resulted in significant research efforts in this space, each offering different trade-offs in terms of compute and closeness to the posterior captured by the CRF. It is worth noting that it is possible to compute unary potentials and solve the CRF at interactive rates (e.g. [10, 48]) but these approaches still have some sequential steps and have large model complexity.

In this paper, we propose to bridge this gap in the literature by introducing HashMatch, a generic and extremely low-compute framework that has been designed from the ground-up for parallel, i.e. pixel independent, processing.

*Authors equally contributed to this work.

As demonstrated in this paper, the efficiency and parallel nature of our approach allows us to achieve compelling results on a variety of computer vision problems, at speeds never before demonstrated. For example, estimating disparity or segmentation masks on 1.3 megapixel images at 1000 fps on high-end GPUs (e.g. NVIDIA Titan X) and at 200fps on VGA images on mobile architectures (e.g. Tegra TX1) while also providing compelling results on multiple computer vision tasks. Our main technical contributions are two-fold. First we propose a *binary* embedding for classification, regression and nearest neighbor tasks that is trained with sparsity and anti-sparsity constraints. This allows evaluating a robust unary potential with only a few operations per pixel. Second, we present a new inference scheme that fully operates in parallel with complexity that is not a function of the size of the solution space. These technical contributions are formulated in a mathematical framework whose objective function is directly applicable to many different computer vision problems.

2. Related Work

Binary Representations. The task of finding binary and compact representations has been exhaustively studied in the literature. This is usually known as *hashing* and the problem is generally formulated as:

$$\mathbf{b} = h(\mathbf{x}) \tag{1}$$

with $\mathbf{x} \in R^n$, $\mathbf{b}$ a binary code in $\{0,1\}^k$, and h the hashing function. h can be a linear projection, a spherical function, a kernel, a neural network, a non-parametric function, etc. Here we focus on the family of linear hash functions of the form $h(\mathbf{x}) = sign(\mathbf{x}\mathbf{W})$, with $\mathbf{W} \in R^{n \times k}$, where $sign(x)$ returns 1 if $x \geq 0$ and 0 otherwise. The most popular data independent approach to generate those hash function is called Locality-Sensitive Hashing (LSH) [23, 9]. These usually make use of random gaussian projections to generate the hyperplanes $\mathbf{W}$. Despite their simplicity, these hashing schemes perform reasonably well in practice. For an extensive review of LSH, we refer the reader to [51].

Data dependent hashing schemes have also been proposed [19, 37, 8, 54, 21, 43, 56]. These methods are usually unsupervised and they try to design an objective function that preserves the similarities of the input signal in the new binary space. Iterative Quantization (ITQ) [19] obtains low dimensional codes by applying PCA to the data and then find the optimal rotation that makes the codes as close as possible in the binary space. [8] casts the hashing problem to an auto-encoder formulation, however part of the optimization resorts to enumerating all the possible solutions, leading to very slow training procedures. In [37], authors exploit sparsity to have a runtime that is cost independent of the original signal dimensionality. However an ℓ_2-normalization of the signal is first performed, therefore the overall running cost still depends on the input dimension. More recently, [52] uses an ensemble of fast decision trees to model hash functions, however the method requires a non-trivial aggregation step, adding compute.

Our work is very different from the ones in the literature. Whereas most of the approaches focus on nearest neighbor tasks, our framework is very flexible and can be used on tasks ranging from classification to signal reconstruction. Different from others [8, 19], we heavily exploit sparsity at runtime and remove normalization steps such as in [37].

Inference in Graphical Models. Estimating the Maximum A Posteriori (MAP) of a multi-label Conditional Random Field (CRF) is a well known NP complete problem [7]. Given the successful use of CRFs for advancing the state of the art, performing (approximate) inference over those models has received much attention. The major difference among all the solvers is whether they are deterministic or stochastic. Stochastic methods include the family of Markov Chain Monte-Carlo methods, which have exponential convergence rate in the worst case but can provide exact results. This family of methods is regarded as too compute intensive for real-time applications that make use of 'large' CRFs. Besides stochastic techniques resides a wide array of deterministic methods. Successful techniques include Move-Making algorithms [7], Belief Propagation [17], Iterated Conditional Modes [3], Tree Reweighted Message Passing [27], Quadratic Pseudo-Boolean Optimization [41] and Mean-Field [28]. Each of these techniques comes with various trade-offs in terms of quality of the approximation and speed.

It is worth noting that when the data term is strong and high-speed inference is required (e.g. depth estimation at VGA resolution), global optimization of the posterior is usually dropped in favor of local optimization [40, 4, 32].

Thanks to our learned binary representation we can provide a very strong data term. This low entropy data term allows us to design and use a new parallel global inference method which reaches high quality solutions for 1.3 Megapixel images in less than one millisecond on GPUs.

3. The HashMatch Framework

Our framework is based on a pairwise-CRF that can be expressed using the following probabilistic factorization:

$$P(Y|D) = \frac{1}{Z(D)} e^{-E(Y|D)} \tag{2}$$

$$E(Y|D) = \sum_i \psi_u(l_i) + \sum_i \sum_{j \in \mathcal{N}_i} \psi_p(l_i, l_j), \tag{3}$$

The data term $\psi_u(l_i)$ models how likely a node in the graph (usually a pixel) belongs to a particular class l_i (e.g. 'foreground'). The exact implementation of $\psi_u(l_i)$ depends on the task at hand. For instance, for finding the nearest neighbor between image patches, labels l_i correspond to vectors (u, v) which define the displacements in the image directions. Then

$$\psi_u(l_i) = |h(\mathbf{x}_i) - h(\mathbf{x}_{i+l_i})| \tag{4}$$

measures the compatibility of two image patches $\mathbf{x}$ centered at 2D pixel location i and $i + l_i$. The function $h(\mathbf{x}) = \text{sign}(\mathbf{x}\mathbf{W})$ is a binary feature, which allows us to compute $\psi_u(l_i)$ highly efficiently via the Hamming distance in (4). For any other classification or regression problem we define ψ_u as

$$\psi_u(l_i) = -\log(g(l_i, h(\mathbf{x}_i))) \tag{5}$$

where g is a learned classifier or regressor that evaluates the likelihood of label l_i given the binary code $h(\mathbf{x}_i)$ of an image patch $\mathbf{x}_i$. The smoothness cost is defined as $\psi_p(x_i = l_i, x_j = l_j) = \max(\tau, |l_i - l_j|)$ and it encourages that neighboring pixels i and j are assigned to similar labels, where τ is a truncation threshold.

Our key contribution is a novel method to compute $h(\mathbf{x})$ and $g(\cdot)$ that captures the essential information in the data and it is detailed in the remainder of the section.

3.1. HashCodes Learning

We now detail our proposed approach to train a function h that maps a signal $\mathbf{x} \in \mathbb{R}^n$ to a binary space $\mathbf{b} = h(\mathbf{x}) = \text{sign}(\mathbf{x}\mathbf{W}) \in \{0, 1\}^k$. This binary representation $\mathbf{b}$, is then used to learn a function $g(l, \mathbf{b})$ that performs any given task $\mathbf{y} \in \mathbb{R}^d$. It is important to note that $\mathbf{y}$ can correspond to tasks as diverse as multi-label classification and structured regression. In particular, one can define $\mathbf{y} = \mathbf{x}$ for nearest-neighbor search. In order to keep the computational cost as low as possible, we consider a linear model for each entry y_l, i.e. $y_l = g(l, \mathbf{b}) = \mathbf{b}^\top \mathbf{z}_l$.

More formally, we learn a set of hyperplanes $\mathbf{W} \in \mathbb{R}^{n\times k}$ and a task function $\mathbf{Z} \in \mathbb{R}^{k\times d}$ that minimizes a loss $\mathcal{L}$:

$$\min_{\mathbf{W},\mathbf{Z}} \ \mathcal{L}(\ \text{sign}(\mathbf{X}\mathbf{W})\ \mathbf{Z}, \mathbf{Y}\ ) + \Gamma(\mathbf{W}) + \Omega(\mathbf{Z}) \tag{6}$$

where $\mathbf{X} \in \mathbb{R}^{m\times n}$ and $\mathbf{Y} \in \mathbb{R}^{m\times d}$ are matrices whose i-th row corresponds respectively to $\mathbf{x}_i$ and $\mathbf{y}_i$. The terms $\Gamma(\mathbf{W})$ and $\Omega(\mathbf{Z})$ are suitable *regularizers* encouraging desired structures on the two predictors.

The model $\mathbf{y} = \mathbf{Z}^\top \text{sign}(\mathbf{W}^\top \mathbf{x})$, can be interpreted as a neural network with one hidden layer and with the operator sign$(\cdot)$ as non-linearity (in contrast to a sigmoid or ReLu [34]). In particular, when $\mathbf{Y} = \mathbf{X}$, the model becomes similar to an *autoencoder* with internal binary representation (e.g. [8]). However, the optimization of Eq. (6) cannot be performed using first-order methods (e.g. backpropagation) because sign$(\mathbf{X}\mathbf{W})$ is a piece-wise constant function (and therefore the subgradient with respect to $\mathbf{W}$ is zero almost everywhere). We circumvent this issue by decoupling the task $\mathbf{y}$ from the binary mapping $h(\mathbf{x})$. To do so we introduce an additional variable $\mathbf{B} = \text{sign}(\mathbf{X}\mathbf{W})$ and then relax the equality constraint by means of a *dissimilarity* measure $\mathcal{D}(\mathbf{X}\mathbf{W}, \mathbf{B})$ that will be minimized. This leads to the problem

$$\begin{aligned} \min_{\mathbf{W},\mathbf{Z},\mathbf{B}} \quad & \mathcal{L}(\mathbf{B}\mathbf{Z}, \mathbf{Y}) + \Gamma(\mathbf{W}) + \Omega(\mathbf{Z}) + \gamma\, \mathcal{D}(\mathbf{X}\mathbf{W}, \mathbf{B}) \\ \text{s.t.} \quad & \|\mathbf{B}\|_\infty \leq \mu \end{aligned} \tag{7}$$

where $\|\mathbf{B}\|_\infty = \max_{ij}|\mathbf{B}_{ij}|$ denotes the ℓ_∞ (or max) norm of $\mathbf{B}$ and $\mu > 0$ a scalar hyperparameter. The constraint $\|\mathbf{B}\|_\infty \leq \mu$ is introduced to encourage so-called *anti-sparse* solutions, such that the minimizer $\mathbf{B}_*$ of Eq. (7) would have all entries $\mathbf{B}_{*ij} = \pm\mu$. The concept of anti-sparsity was originally introduced in the signal processing literature where it was observed that imposing constraints based on the max-norm would induce 'binary' solutions. We refer the reader to [33, 18, 25, 53, 44] for an in-depth discussion on the anti-sparse properties of max-norm regularization (and constraints). The idea of introducing a variable $\mathbf{B}$ with binary entries is akin to the one proposed in [8]. However in such work the authors imposed the constraint $\mathbf{B} \in \{-1, 1\}$ in the optimization, leading to an NP-hard problem. On the other hand, the max-norm constraint ball is a convex set, and in the following we discuss an efficient optimization algorithm to find $\mathbf{B}$ in practice.

Interestingly, when $\mathbf{B}$ is a binary matrix with entries equal to $\pm\mu$, the problem of learning a linear predictor $\mathbf{W}$ such that $\frac{1}{\mu}\mathbf{B} \sim \text{sign}(\mathbf{X}\mathbf{W})$ corresponds to a standard multi-label (or multi-task) binary classification problem, with each column of $\mathbf{B}$ representing a different binary task. Therefore a natural choice for the dissimilarity measure $\mathcal{D}(\mathbf{X}\mathbf{W}, \mathbf{B})$ is a loss function used for classification problems such as the logistic, hinge, least-squares etc.

3.2. Optimization

The optimization problem described by Eq. (7) is not jointly convex in $\mathbf{W}, \mathbf{Z}, \mathbf{B}$ but, for convex loss functions $\mathcal{L}, \mathcal{D}$ and regularizers Γ, Ω the objective functional is convex separately in each variable. A natural strategy to address this problem is therefore to perform either alternated minimization [47, 39] or block coordinate descent [5]. Below we detail how we propose to perform the minimization of Eq. (7) by independently optimizing $\mathbf{W}, \mathbf{B}$ and $\mathbf{Z}$.

Optimizing $\mathbf{W}$. We propose to use the regularizer $\Gamma(\mathbf{W}) = \lambda|\mathbf{W}|_1$ in order to induce sparse solutions. In particular, in

our experiments λ is chosen so that the corresponding solution $\mathbf{W}_*$ has at most $s << n$ non-zero entries in each column. Indeed, this allows to compute a code $h(\mathbf{x}) = \text{sign}(\mathbf{W}_*^\top \mathbf{x})$ in $O(sk)$ operations rather than $O(nk)$. When the dissimilarity measure $\mathcal{D}$ is smooth, one can employ a standard proximal forward-backward splitting method to find the best $\mathbf{W}$ for fixed $\mathbf{B}$ and $\mathbf{Z}$. The algorithm consists of producing a sequence of updates defined by

$$\mathbf{W}_{t+1} = \text{Prox}_{\sigma_1 \lambda |\cdot|_1}(\mathbf{W}_t - \sigma_1 \mathbf{X}^\top \gamma \nabla \mathcal{D}(\mathbf{X}\mathbf{W}_t, \mathbf{B})) \quad (8)$$

where $\text{Prox}_{\sigma_1 \lambda |\cdot|_1}$ denotes the proximal operator of $\sigma_1 \lambda |\cdot|_1$ (see [2]). For the case of the ℓ_1 norm, the proximal operator is well-known to correspond to the entry-wise *soft-thresholding* [2]: for each scalar w, the soft thresholding is such that $\text{Prox}_{\sigma_1 \lambda |\cdot|_1}(w) = 0$ if $|w| \leq \sigma_1 \lambda$ and $\text{Prox}_{\sigma_1 \lambda |\cdot|_1}(w) = w - sign(w)\sigma_1 \lambda$ otherwise. For a suitable choice of step size σ_1 (by either line search or depending on the Lipschitz constant of the gradient of $\mathcal{D}$), iterating Eq. (8) is guaranteed to converge to the solution $\mathbf{W}_*$ with the value of the objective functional decreasing at a rate of $O(1/t)$ [2]. Following [57] we also use an early stopping criterion to fix the desired number of variables with the highest absolute values for each column of $\mathbf{W}$.

Optimizing B. If $\mathcal{L}$ is smooth, one can again adopt the proximal forward-backward splitting approach to minimize Eq. (7) w.r.t. $\mathbf{B}$ (for fixed $\mathbf{W}$ and $\mathbf{Z}$) obtaining the updates

$$\begin{aligned} \widetilde{\mathbf{B}}_{t+1} &= \mathbf{B}_t - \sigma_2 \left(\nabla \mathcal{L}(\mathbf{B}_t \mathbf{Z}, \mathbf{Y}) \mathbf{Z}^\top + \gamma \nabla \mathcal{D}(\mathbf{X}\mathbf{W}, \mathbf{B}_t) \right) \\ \mathbf{B}_{t+1} &= \text{Prox}_{\{\|\cdot\|_\infty \leq \mu\}} (\widetilde{\mathbf{B}}_{t+1}) \end{aligned} \quad (9)$$

To compute the proximal operator, we make use of the Moreau decomposition, stating that for any function ϕ, $\text{Prox}_{\phi}(\mathbf{B}) = \mathbf{B} - \text{Prox}_{\phi^*}(\mathbf{B})$, with ϕ^* denoting the Fenchel's conjugate of ϕ (defined as $\phi^*(\mathbf{B}) = \sup_{\mathbf{C} \in \mathbb{R}^{m \times k}} \text{tr}(\mathbf{B}^\top \mathbf{C}) - \phi(\mathbf{C})$). In our case, ϕ can be interpreted as the indicator function of the max-norm ball of radius μ (namely the function that is zero when $\|\mathbf{B}\|_\infty \leq \mu$ and $+\infty$ otherwise). It is straightforward to show that the corresponding Fenchel's conjugate is $\phi^*(\mathbf{B}) = \mu |\mathbf{B}|_1$. As a consequence we have

$$\mathbf{B}_{t+1} = \widetilde{\mathbf{B}}_{t+1} - \text{Prox}_{\mu |\cdot|_1}(\widetilde{\mathbf{B}}_{t+1}) \quad (10)$$

with $\text{Prox}_{\mu |\cdot|_1}$ the entry-wise soft-thresholding operator introduced for the optimization of $\mathbf{W}$. We again obtain a convergence rate in the order of $O(1/t)$ [2].

Optimizing Z. We consider the Frobenius norm regularizer $\Omega(\mathbf{Z}) = \eta \|\mathbf{Z}\|^2$ to avoid overfitting. This problem can be solved by standard gradient descent updates

$$\mathbf{Z}_{t+1} = \mathbf{Z}_t - \sigma_3 \left(\mathbf{B}^\top \nabla \mathcal{L}(\mathbf{B}\mathbf{Z}, \mathbf{Y}) + \eta \mathbf{Z}_t \right) \quad (11)$$

which is known to converge for a suitable choice of step η. Convergence rates of $O(1/t)$ are guaranteed also in this case [2]. Faster rates can be achieved adding further hypotheses on the conditioning of the loss $\mathcal{L}$ [6]. Moreover, if we consider $\mathcal{L}(\mathbf{B}\mathbf{Z}, \mathbf{Y}) = \|\mathbf{B}\mathbf{Z} - \mathbf{Y}\|^2$, we can compute the the solution to the problem in closed form as $\mathbf{Z}_* = (\mathbf{B}^\top \mathbf{B} + \eta I)^{-1} \mathbf{B}^\top \mathbf{Y}$, with I the $k \times k$ identity matrix.

Convergence Rates of Block Coordinate Descent. Block coordinate descent methods consists of iterating across the steps at Eq. (8,9,11) by optimizing over one variable at the time while keeping the other two variables fixed. In general, it is challenging to prove convergence of the iterations to a stationary point (e.g. local minima or saddles) let alone prove rates on how fast such convergence can be guaranteed. However, for the choice of loss functions and regularizers considered in this work, the proposed approach belongs to the family of *Proximal Alternating Linearized Minimization (PALM)* optimization methods [5] whose convergence properties have been recently studied. As a corollary to Theorem 1 and Remark 6 in [5] we have the following

Theorem 1 (Convergence of PALM)**.** *With the notation of Eq. (7), let $\mathcal{L}$ and $\mathcal{D}$ satisfy the hypotheses in [5] (Thm.1), in particular, they are differentiable, have Lipschitz continuous gradient with associated Lipschitz constants $L_{\mathcal{L}}$ and $L_{\mathcal{D}}$ respectively. Consider the iterative sequence of $(\mathbf{W}_t, \mathbf{B}_t, \mathbf{Z}_t)_{t=0}^{\infty}$ obtained by updating each variable iteratively according to respectively Eq. (8,9,11) with step size respectively $\sigma_1 \leq (\gamma L_{\mathcal{D}} \|\mathbf{X}\|_{op})^{-1}$, $\sigma_2 \leq (\eta L_{\mathcal{L}} + \gamma L_{\mathcal{D}})^{-1}$, $\sigma_3 \leq (\mu m L_{\mathcal{L}} + \eta)^{-1}$ (with m defined in Thm 3.1 of [5], $\|\mathbf{X}\|_{op}$ denoting the operator norm of X, namely the maximum singular value of $\mathbf{X}$). Then there exists a stationary point $\mathbf{W}_*, \mathbf{B}_*, \mathbf{Z}_*$ for the functional at Eq. (7) such that*

$$\|(\mathbf{W}_t, \mathbf{B}_t, \mathbf{Z}_t) - (\mathbf{W}_*, \mathbf{B}_*, \mathbf{Z}_*)\| = O(1/t) \quad (12)$$

The above theorem states that for specific choices of descent steps $\sigma_1, \sigma_2, \sigma_3$, one can expect convergence to a stationary point at a sublinear rate of the order of $O(1/t)$.

Choice of $\mathcal{L}$ and $\mathcal{D}$. The relaxed problem described in Eq. (7) and the optimization approach described apply to any choice of convex smooth function $\mathcal{L}$ and dissimilarity measure $\mathcal{D}$. In the experiments reported in this work we adopted the least squares loss function $\mathcal{L}(\mathbf{B}\mathbf{Z}, \mathbf{Y}) = \|\mathbf{B}\mathbf{Z} - \mathbf{Y}\|^2$ and $\mathcal{D}(\mathbf{W}\mathbf{X}, \mathbf{B}) = \|\mathbf{W}\mathbf{X} - \mathbf{B}\|^2$ for which it is easy to recover the Lipschitz constant of the gradient to derive the descent step sizes $\sigma_1, \sigma_2, \sigma_3$ in Thm.1. This choice was also motivated by the fact that least-squares is the standard loss function for regression and reconstruction problems (hence a natural choice for $\mathcal{L}$) but also often used in classification settings [55] (hence a viable choice for the dissimilarity $\mathcal{D}$). In the supplementary material we report the pseudocode for

the optimization strategy described in for this choice of loss and dissimilarity.

3.3. Parallel Inference

Infering the posterior probability, and a fortiori the MAP (Maximum a Posteriori) of $P(Y|D)$, is in general very hard as it requires solving for a very complex series of integrals over all the variables $x \in X$. To approximate this complex distribution, we resort to using variational approximation. More precisely, we are aiming at finding a distribution Q that is a 'close' approximation of P within the class of distributions that can be factorized as a product of independent marginals, i.e.

$$Q(Y) = \prod_i Q(Y_i) \tag{13}$$

This approximation is computationally attractive but is very likely to lose a lot of information about the original distribution $P(Y|D)$. Nevertheless, the result of MAP and MPM (Maximum Posterior Marginal) inference will be quite similar when the entropy of $P(Y|D)$ is low. Broadly speaking, in the case of the pairwise CRF we described previously, good approximations of $P(Y|D)$ are obtained when the unary potentials are 'peaky' (i.e. low entropy). The quality of the approximation between $Q(Y)$ and $P(Y|D)$ is usually measured using $KL(Q(Y)||P(Y|D))$, where KL is the Kullback-Leibler divergence. Taking the fixed point solution of the Kullback-Leibler divergence [26], we obtain the following update for the label l_i in the marginal of random variable x_i:

$$Q_i^t(l_i) = \frac{1}{B_i} e^{-M_i(l_i)} \tag{14}$$

$$M_i(l_i) = \psi_u(l_i) + \sum_{j \in \mathcal{N}_i} \sum_{l_j \in \mathcal{L}} Q_j^{t-1} \psi_p(l_i, l_j) \tag{15}$$

$$B_i = \sum_{l_i \in \mathcal{L}} e^{-M_i(l_i)} \tag{16}$$

The underlying coordinate ascent procedure results in a better approximation of P by Q for each iteration, but also guarantees convergence. Note that popular techniques like Belief Propagation are not guaranteed to converge when performing inference over graphs. At this stage, it is crucial to note that the complexity of evaluating the updated $Q^t(Y)$ is $O(|Y||\mathcal{L}|(|\mathcal{N}||\mathcal{L}| + 1))$. The quadratic complexity on $\mathcal{L}$ is not of a practical concern for high-speed inference when $\mathcal{L}$ is small. Nevertheless, this makes the (computationally attractive) mean-field framework too slow when this number grows large. As mentioned before, the MPM and MAP solutions are similar in the pairwise CRF (Eq. (3)) when the entropy of the unary is low. We go one step further in the approximation and explicitly assume that Q also has low entropy and approximate it with a Dirac δ function. This corresponds to setting $Q_i = \delta(l_i - \arg\max_{l_j} Q_j)$. We can now rewrite (15) as follows

$$M_i(l_i) = \psi_u(l_i) + \sum_{j \in \mathcal{N}_i} \psi_p(l_i, \arg\max_{l_j} Q_j) \tag{17}$$

Since Q_i^t (14) now follows a Dirac δ function, computing the normalization function B_i (16) is not required anymore. The compute complexity of updating $Q^t(Y)$ is now $O(|Y||\mathcal{N}|(1+|\mathcal{N}|))$. Roughly speaking, this corresponds to a reduction of complexity along the lines of $O(|\mathcal{L}|^2/|\mathcal{N}|)$.

Note that $|\mathcal{N}|$ is small for many problems (e.g. stereo depth estimation), and that in most these problems $|\mathcal{L}| > |\mathcal{N}|$ (e.g. $|\mathcal{L}|$ is in the hundreds and $|\mathcal{N}| = 4$ when estimating disparities).

3.4. Computational Analysis

The following is a computational analysis of the proposed method for (discrete) disparity estimation. We assume an input image with $|Y|$ pixels, and $\mathcal{L}$ possible labels. Typical values for disparity estimation are $|Y| = 1280 \times 1024$ and $\mathcal{L} = 512$. The hyperplanes $\mathbf{W}$ are trained to have maximum 4 non-zero elements for each code. Each pixel i is associated with a patch P_i of size $|P_i| = 11 \times 11$. Taking the sign of the dot product between P_i and $\mathbf{W}$ remaps P_i into $k = 32$ binary codes. The computation of the binary codes $\mathbf{b}$ is independent of the window size $|P|$ and it involves 4 multiplications and sums for each hyperplane. This corresponds to a per-pixel complexity of $O(4\,k)$ to compute a hash code. We initialize the data term ψ_i by evaluating only a very limited set of 32 random label hypotheses. The distances are computed using the Hamming distance in the new space and this can be efficiently implemented in O(1) making use of the *__popc()* function that is implemented in most of the latest GPU architectures. The initialization step then has a per-pixel complexity $O(4\,k)$. Regarding the inference, we only use the immediate $\mathcal{N} = 3 \times 3$ neighbors of each pixel in the pairwise potential. The marginal of each pixel can be updated in parallel without waiting for sequential propagation steps like in [4, 16]. In practice, we use 4 steps of the proposed inference, resulting in a per-pixel complexity of $O(4\,|\mathcal{N}|(1 + |\mathcal{N}|))$. Note that in contrast to other approaches, the proposed algorithm is independent of the number of labels $\mathcal{L}$ and the patch size $|P_i|$. The relatively low computational complexity and fully parallel nature of all the components of the proposed method make it particularly suitable for high-speed applications on low compute devices. We tested the HashMatch framework on an NVIDIA Titan X GPU, with an overall running time of $890\mu s$ per frame. We also implemented the algorithm on a NVIDIA Tegra TX1 with an overall running time of $5ms$,

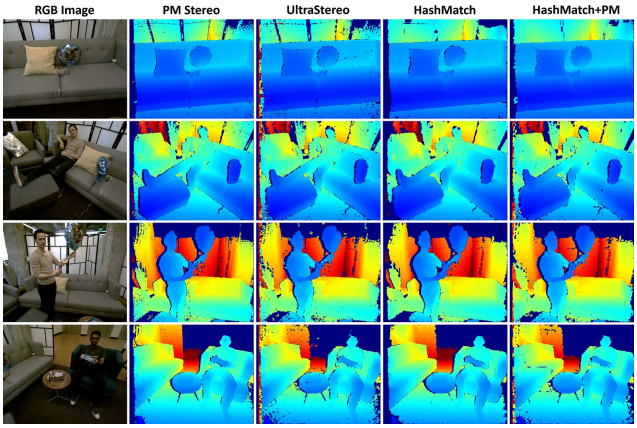

Figure 1. Qualitative comparisons of depth maps generated with state of the art methods. HashMatch shows the most complete results in complex scenes.

which opens up the possibility of high speed applications on mobile platforms.

4. Results

We evaluate the HashMatch framework on a diverse set of computer vision tasks. For each problem, we explicitly describe the form of the unary potential that is used. We first show how our method can handle continuous labeling problems such as disparity estimation. Further, we evaluate the proposed hashing scheme on retrieval and feature approximation tasks. Finally, we assess the quality of the proposed inference for background subtraction. Note that for the tasks where the proposed inference is used, the number of iterations is constant and set to 4.

4.1. Depth Estimation

In this section, we focus on depth estimation from stereo images under active illumination [15, 16]. For our experiments, we use a hardware setup similar to [16], i.e. two IR cameras in a stereo configuration as well as a Kinect V1 DOE. When the IR images I_L and I_R are calibrated and rectified, each pixel $p = (u, v)$ in I_L has a corresponding pixel $q = (u + l, v)$ in I_R that lies on the same scanline v. We apply HashMatch to retrieve the continuous disparity $l \in \mathrm{R}$. The disparity is then remapped in the depth domain via $Z = \frac{bf}{l}$, where b is the baseline of the system, f the focal length of the camera and l is the inferred disparity.

In this section, the data term $\psi_u(l_i)$ is computed according to Eq. (4). We train the hyperplanes $\mathbf{W}$ by acquiring 10000 images and extracting 11×11 image patches $\mathbf{x}$. We set the task $\mathbf{y} = \mathbf{x}$. We use $k = 32$ hyperplanes $\mathbf{W}$ with maximum 4 non-zero elements for each column.

We compare HashMatch with state of the art methods, including both qualitative and quantitative evaluations. We set the exposure time of the cameras very low ($2ms$) in order to perform evaluations on real data captured at $500Hz$.

Figure 2. Example of high quality depthmap and point clouds generated with HashMatch. Notice the level of details and the absence of quantization and outliers.

This data has a significatively lower signal to noise ratio (SNR) compared to data captured at $30Hz$. In Fig. 1 we show qualitative results generated with HashMatch, PatchMatch Stereo [4] and UltraStereo (US) [16]. Notice how the baseline methods suffer from the relatively low SNR whereas HashMatch is able to predict complete and smooth depthmaps. We also generate results using our unary term and the PatchMatch inference [1] (HashMatch+PM in Fig. 1). The proposed parallel inference produces results very close to those generated with the PatchMatch inference, but is $2X$ faster compared with the very optimized implementation described in [16]. In Fig. 2 we show a high quality depthmap and pointcloud generated with HashMatch, notice the level of details that are captured by our method.

To quantitatively assess the proposed framework, we follow the procedure presented in [15] and acquired images of a flat wall at multiple known distances, varying between 500 and 3500 cm. For each set distance, 100 frames are recorded from which we estimate the average depth bias (defined as the average error), and depth jitter (defined as the standard deviation). We compare with Kinect V1, RealSense R200, PatchMatch Stereo [4], HyperDepth [15] and UltraStereo [16] as baseline methods; results are reported in Fig. 3. HashMatch outperforms most of the competitors and is on par with more involved methods such as [15].

Finally, we compare the quality of our hashing scheme with other state of the art methods such as: ITQ [19], SBE [37], CBE [56], Binary Autoencoders (BA) [8] and UltraStereo [16]. We used the synthetic dataset from [16] with perfect groundtruth disparities and perform the (discrete) nearest neighbor search over the pixel disparities. We defined the accuracy as the percentage of pixels for which the estimated disparity is below 1 pixel; results are reported in

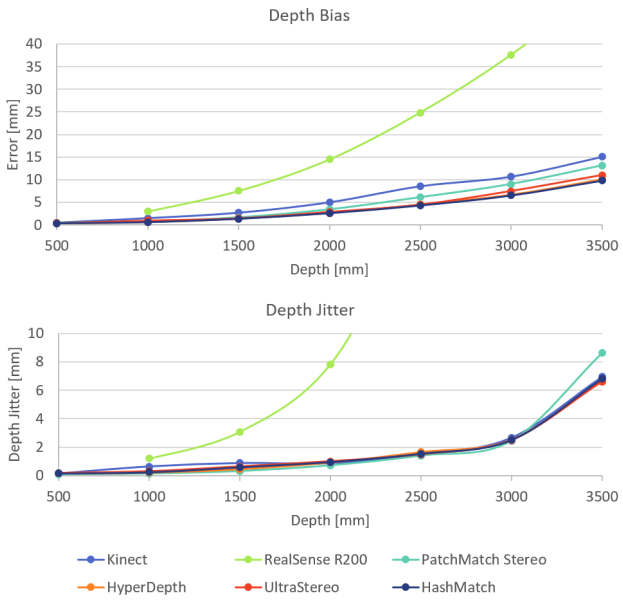

Figure 3. Quantitative comparisons with state of the art methods. HashMatch achieves the lowest error using the lowest compute.

Tab. 1. Although our HashMatch descriptor uses only 4 non-zero elements per hyperplane, the results are on par with dense hashing schemes such as BA. We also outperform other sparse hashing schemes such as CBE and SBE, proving the quality of the proposed representation. It is interesting to note that besides providing smooth results, using the proposed inference raises the precision reported in Tab. 1 from 77% to 96%.

HashMatch	ITQ	SBE	CBE	LSH	BA	US
77%	76%	68%	70%	58%	**77%**	73%

Table 1. Hashing Schemes Comparisons. We compare our method with other popular hashing schemes and binary representations: ITQ [19], SBE [37], CBE [56], Binary Autoencoders [8] and UltraStereo (US) [16]. HashMatch uses only 4 non zero elements and is on par with dense methods like BA.

4.2. Nearest Neighbor Retrieval

In this section, we evaluate HashMatch on a nearest neighbor retrieval task. We assume we have a two set of feature vectors, query and base. For each sample in the query set, the goal is to find the nearest neighbor in the base set. This is a typical case for correspondence search problems between two or more image. The goal of this section is to evaluate only our hashing scheme and compare it with other state of the art methods, therefore no parallel inference is required. The data term $\psi_u(l_i)$ is computed according to Eq. (4). Notice that for this experiment the smoothness term is not needed, thus we dropped it. We use the GIST1M dataset [24], which is composed of three disjoint sets: train, query and base. Each descriptor $\mathbf{x}$ has 960 dimensions, and we minimize Eq. (6) setting the task $\mathbf{Y} = \mathbf{X}$. We trained data

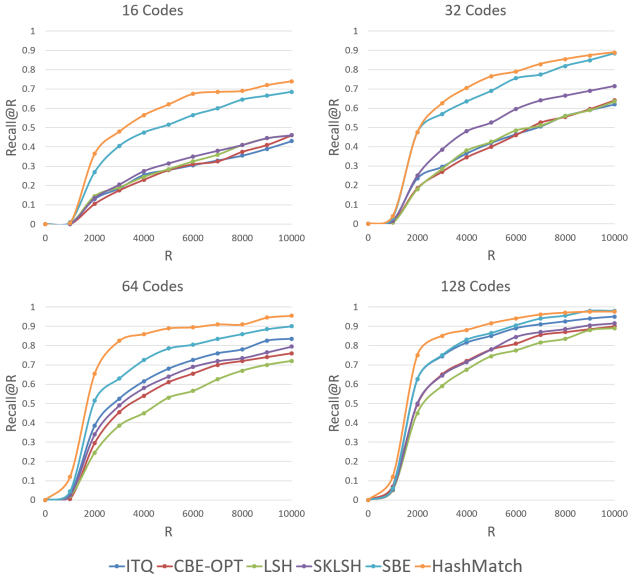

Figure 4. Recall@R curves on the GIST1M dataset.

dependent hashing schemes using the training set and test on the other sets. We compute the Recall@R, defined as the recall for R retrievals.

We compare our method with the following state of the art hashing techniques: ITQ [19], SBE [37], CBE [56], LSH [23], and SKLSH [36]. For each method we trained $16, 32, 64, 128$ codes, perform the binary embedding according to Eq. (1) and compute the nearest neighbor search using the Hamming distance. We do not perform any data preprocessing such as normalization steps or augmentation. For HashMatch and SBE we set the sparsity parameter such that we have 10% of non-zero elements.

We report the Recall@R curves in Fig. 4. For small number of codes, HashMatch greatly outperforms all the competitors, including dense ones such as ITQ. When larger codes are used (i.e. 128), this gap reduces but HashMatch still provides for the largest area under curve (AUC).

4.3. Feature Approximation

In this experiment we consider the problem of approximating complex feature descriptors given a set of 11×11 image patches $\mathbf{x}$. For this particular application we consider SIFT descriptors [31] $\mathbf{s} \in \mathrm{R}^{128}$. The goal is to minimize Eq. (6) where the task $\mathbf{Y} = \mathbf{S}$ is the set of SIFT descriptors. In other words, we apply HashMatch to a regression problem, where the target continuous function is computed from handcrafted features. In general we could apply the same framework to learn more sophisticated descriptors.

We consider the EPFL wide-baseline stereo data [46], where we trained HashMatch on those sequences with no groundtruth available. Training data is generated by extracting SIFT descriptors in an unsupervised way. At test time we detect corners and compute SIFT and HashMatch

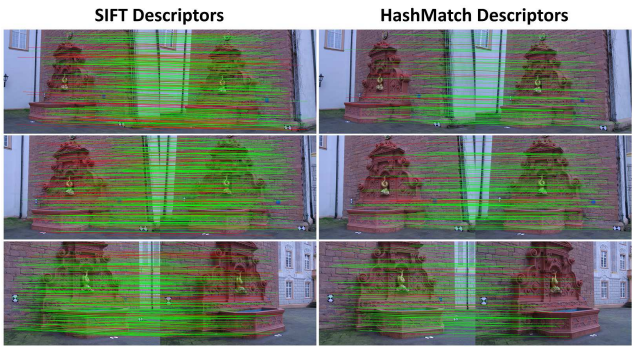

Figure 5. Qualitative Experiments for feature approximation (see Sec. 4.3).

descriptors, respectively. We match the descriptors based on the closest ℓ_2 distance and we filter outliers by imposing the epipolar constraint between the two images. In Fig. 5 we report qualitative results: in green we depict correct matches, in red those matches with distances greater than 1 cm in the provided groundtruth. SIFT achieves an average end-point error of 0.8 cm, whereas HashMatch gets very close with 1.2 cm. If we consider as inliers the percentage of retrieved matches with error $<$ 1cm, HashMatch reports an overall accuracy of 90%, whereas SIFT retrieves 81% corrected matches. On average, SIFT is able to retrieve 350 good matches per image, HashMatch 150. While the number of matches are fewer in case of HashMatch we get more reliable matches. In practice, the order of feature matches we obtain from HashMatch would cater to most applications that use SIFT, and with much less compute making HashMatch very powerful in such scenarios.

4.4. Background Subtraction

In this last experiment, we evaluate the proposed parallel inference on a background segmentation task with static cameras. Typical applications are surveillances and people tracking scenarios. We consider natural (indoor) rooms. Assuming a clean background shot is available (RGB and depth), we want to detect any new object entering the scene.

The goal of this application is to evaluate the proposed inference and compare it with well established approaches like belief propagation (BP) [45], tree-reweighted message passing (TRW) [50, 27], mean field [49] and patch match [1]. Assuming that RGB and depth information is available, we use a unary potential of the form $\psi_u(l_i) = \psi_{rgb}(l_i) + \psi_{depth}(l_i)$. The two terms are simple differences in the HSV space and a logistic function in the depth domain as defined in [35] and [14]. Note that the unary potential is shared amongst all the baselines methods and that the pairwise term is the standard Potts model.

We collected 100 frames from different subjects and objects, and manually labeled the foreground from the images to obtain the ground truth. We report the average energy obtained by the proposed inference and the baselines in Table 4.4. It is interesting to note that although much less computationally demanding, the proposed inference achieves final energies that are very similar to that obtained by the baselines. Some qualitative results are shown in Fig. 6.

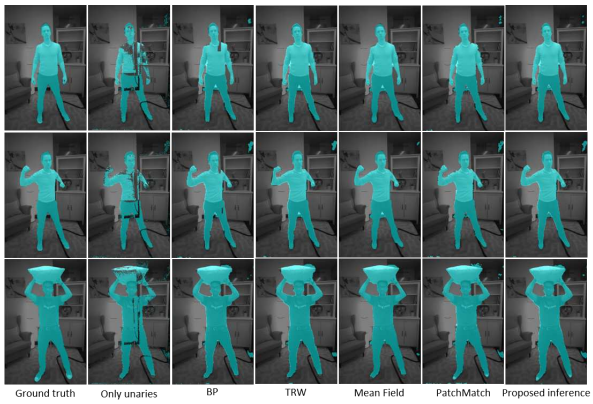

Figure 6. Qualitative results of background subtraction. The foreground segmentation corresponds to the cyan region, which is overlaid on the gray-scale version of the color image. Note that we achieve results that are comparable with more computationally demanding approaches, but orders of magnitude faster.

Inference algorithm	Average energy
Only Unaries	$5.8 * 10^5$
BP [45]	$2.9 * 10^5$
TRW [50, 27]	$\mathbf{2.85 * 10^5}$
Mean Field [49]	$2.9 * 10^5$
PatchMatch [1]	$3.0 * 10^5$
Proposed inference	$2.9 * 10^5$

Table 2. Quantitative evaluation of the proposed inference on a background subtraction task. We compare our method against established inference techniques. We report the average energy obtained by each approach. Note how all the propagation approaches significantly reduce the energy obtained by the initial solution (first row). Also note that although the proposed inference is less computationally demanding than the baselines, it reaches very comparable energy levels.

5. Conclusion

In this paper we presented HashMatch, an efficient framework tailored for parallel compute architectures. Through extensive experiments on a diverse set of computer vision tasks, we demonstrated that although HashMatch operates at extreme speeds, it makes little compromise in precision compared to more compute intensive approaches. All these characteristics make HashMatch an appealing framework for low compute mobile platforms and for products required to operate at very high speeds.

Acknowledgements

We thank the entire perceptiveIO team for continuous feedback and support regarding this work.

References

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM SIGGRAPH and Transaction On Graphics*, 2009. 6, 8

[2] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011. 4

[3] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986. 2

[4] M. Bleyer, C. Rhemann, and C. Rother. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *BMVC*, 2011. 2, 5, 6

[5] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 2014. 3, 4

[6] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. 4

[7] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001. 1, 2

[8] M. . Carreira-Perpin and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015. 2, 3, 6, 7

[9] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM symposium on Theory of computing*, pages 380–388. ACM, 2002. 2

[10] M.-M. Cheng, V. A. Prisacariu, S. Zheng, P. H. Torr, and C. Rother. Densecut: Densely connected crfs for realtime grabcut. In *Computer Graphics Forum*, volume 34, pages 193–201. Wiley Online Library, 2015. 1

[11] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015. 1

[12] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 1

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 1

[14] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. Orts Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi. Fusion4d: Real-time performance capture of challenging scenes. 2016. 8

[15] S. R. Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. O. Escolano, D. Kim, and S. Izadi. Hyperdepth: Learning depth from structured light without matching. In *CVPR*, volume 2, page 7, 2016. 6

[16] S. R. Fanello, J. Valentin, C. Rhemann, A. Kowdle, V. Tankovich, and S. Izadi. Ultrastereo: Efficient learning-based matching for active stereo systems. In *CVPR*, 2017. 5, 6, 7

[17] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70(1):41–54, 2006. 2

[18] J. J. Fuchs. Spread representations. In *ASILOMAR*, 2011. 3

[19] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *PAMI*, 35(12):2916–2929, 2013. 2, 6, 7

[20] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 1

[21] J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer. Compact hashing with joint optimization of search accuracy and time. In *CVPR*, 2011. 2

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 1

[23] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM symposium on Theory of computing*, 1998. 2, 7

[24] H. Jegou, M. Douze, and C. Schmid. Searching with quantization: approximate nearest neighbor search using short codes and distance estimators. In *INRIA Technical report*, 2009. 7

[25] H. Jégou, T. Furon, and J. J. Fuchs. Anti-sparse coding for approximate nearest neighbor search. In *ICASSP*, 2012. 3

[26] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 5

[27] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006. 2, 8

[28] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011. 2

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[30] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 1

[31] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 7

[32] J. Lu, H. Yang, D. Min, and M. Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *CVPR*, 2013. 2

[33] Y. Lyubarskii and R. Vershynin. Uncertainty principles and vector quantization. *IEEE Transactions on Information Theory*, 2010. 3

[34] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. 2012. 3

[35] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754. ACM, 2016. 8

[36] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009. 7

[37] M. Rastegari, C. Keskin, P. Kohli, and S. Izadi. Computationally bounded retrieval. In *CVPR*, 2015. 2, 6, 7

[38] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 1

[39] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013. 3

[40] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR*, 2011. 2

[41] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 2

[42] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001. 1

[43] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *PAMI*, 34(1):66–78, 2012. 2

[44] C. Studer, T. Goldstein, W. Yin, and R. G. Baraniuk. Democratic representations. *CoRR*, 2014. 3

[45] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *ICCV*, 2003. 8

[46] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *PAMI*, 2010. 7

[47] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001. 3

[48] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)*, 2015. 1

[49] V. Vineet, J. Warrell, and P. H. S. Torr. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. 2012. 8

[50] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *IEEE Transactions on Information Theory*, 2002. 8

[51] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014. 2

[52] S. Wang, S. R. Fanello, C. Rhemann, S. Izadi, and P. Kohli. The global patch collider. *CVPR*, 2016. 2

[53] S. Wang and N. Rahnavard. Binary compressive sensing via sum of l1-norm and l(infinity)-norm regularization. In *MILCOM*, 2013. 3

[54] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009. 2

[55] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 2007. 4

[56] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014. 2, 6, 7

[57] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. In *Journal of the Royal Statistical Society*, 2005. 4