Fast Image Gradients Using Binary Feature Convolutions

Pierre-Luc St-Charles, Guillaume-Alexandre Bilodeau
LITIV lab., Dept. of Computer & Software Eng.
Polytechnique Montréal
Montréal, QC, Canada
{pierre-luc.st-charles,gabilodeau}@polymtl.ca

Robert Bergevin
LVSN - REPARTI
Université Laval
Québec City, QC, Canada
robert.bergevin@gel.ulaval.ca

Abstract

The recent increase in popularity of binary feature descriptors has opened the door to new lightweight computer vision applications. Most research efforts thus far have been dedicated to the introduction of new large-scale binary features, which are primarily used for keypoint description and matching. In this paper, we show that the side products of small-scale binary feature computations can efficiently filter images and estimate image gradients. The improved efficiency of low-level operations can be especially useful in time-constrained applications. Through our experiments, we show that efficient binary feature convolutions can be used to mimic various image processing operations, and even outperform Sobel gradient estimation in the edge detection problem, both in terms of speed and F-Measure.

1. Introduction

Binary feature descriptors have recently experienced a surge in popularity in computer vision tasks such as object matching, visual correspondence, and texture analysis. Their compact and discriminative nature allows them to perform just as well as (or better than) traditional handcrafted image feature descriptors like SIFT's [20] or SURF's [6], yet at a much lower computational cost [2, 5, 16]. Typically, binary features are created by comparing low-level image characteristics (intensities or gradients) based on a predetermined (handcrafted or learned) pattern. The boolean comparison results obtained are then concatenated into binary strings, and the "feature-space" distance between these descriptors can finally be defined as the Hamming distance between their binary strings.

Lately, research on feature descriptors has mostly focused on proposing new large-scale binary features for the keypoint description and matching problem [5, 17, 25]. These works show a clear trend towards data-driven pattern design for learning task-specific features. However, small-scale handcrafted binary patterns (e.g. Local Binary Patterns [24]) are still very popular in recognition tasks [7, 29–31] and in texture analysis and modeling [22, 27]. While large-scale and small-scale binary features differ considerably in their use, they both share the idea of comparing low-level image characteristics to describe shape context and local texture information. Small-scale binary descriptors are however often computed densely over entire images; therefore, given knowledge of their comparison pattern, they can be reused to accelerate other image processing operations.

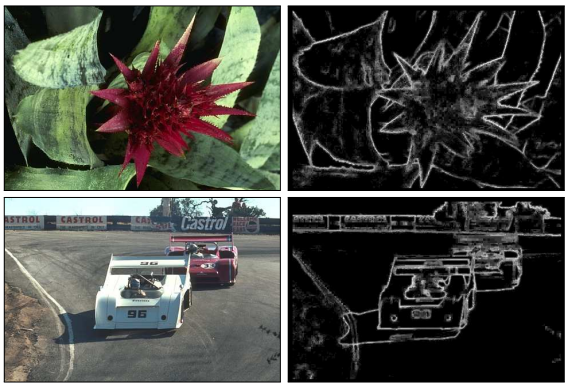


Figure 1: Gradient magnitude maps obtained by pooling the Hamming weights of 5x5, 16-bit binary feature descriptors across a 3-level image pyramid. The pyramid is generated using only pixels fetched via the feature's lookup pattern.

In this paper, we show that dense binary features can be used to efficiently approximate image processing operations that rely on convolutions, for instance the generation of an image pyramid, and local image gradient estimation. These operations are commonly used to solve computer vision problems, but to our knowledge, no previous work has studied how the side products of local binary descriptor computations can be used to approximate the low-pass and high-pass filtering steps they require. In most cases, applications which already need to compute dense binary descriptors can benefit from having *almost* costless filtered images at their disposition for further processing and analysis. Besides, depending on the type of local binary feature used, the "binary feature convolutions" we study may be several times faster than their traditional counterpart (even

when considering separable filters), and just as accurate. This opens the door to new possibilities for applications targeting low-power or hardware-limited mobile and embedded platforms. Examples of gradient magnitude maps obtained using a small-scale binary feature are presented in Figure 1.

In the following sections, we show that given the 5x5, 16-bit binary pattern displayed in Figure 2b, we can adequately mimic a traditional pyramid reduction based on Burt and Adelson's 5x5 kernel [9], and generate gradient maps that are visually similar to those obtained using 3x3 and 5x5 Sobel operators. Then, to demonstrate that binary feature convolutions are also comparable in terms of measurable performance to their classic counterparts in a computer vision problem, we set up an experiment based on Canny's edge detection method [11] using the BSDS500 boundary segmentation dataset [3]. We show that given identical non-maximum suppression and hysteresis thresholding steps, our proposed image processing operations based on binary feature convolutions provide faster and slightly improved results over Canny's traditional image processing operations. The source code used for our experiments is available online[1].

1.1. Binary Features

Small-scale binary descriptors were first used by Ojala et al. in [23] for texture analysis and classification. Local Binary Patterns (LBPs), in their 3x3, 8-bit form shown in Figure 2a, were demonstrated to be invariant to pixel intensity variations, and were directly used to compute texture histograms due to their small code size. The comparison operation used in the original LBP descriptor is simply a pixel intensity difference with respect to the pattern's central pixel; more specifically, the N-bit LBP descriptor of a pixel x is defined as

$$B(x) = \sum_{p=0}^{N-1} s(i_p\,,\, i_x) \cdot 2^p \tag{1}$$

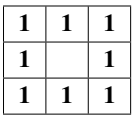
where i_p is the intensity of the p-th neighbor of x on the predetermined pattern, and

$$s(i_p\,,\, i_x) = \begin{cases} 1 & \text{if } i_p \geq i_x \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

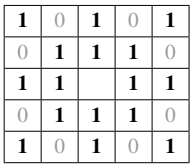
is the classic LBP comparison operator. LBPs were later generalized to any pixel neighborhood size in [24], and reused in various applications over the years [1, 14, 33].

Due to the simplistic nature of LBPs, numerous improvements have been proposed in the literature. Many authors have suggested alternative comparison operations [19, 28], have applied new comparison patterns in the spatio-temporal domain [32, 33], or have proposed to use supplemental gradient information along with LBPs [13, 18]. In this work, we rely on the Local Binary Similarity Pattern

[1] https://github.com/plstcharles/litiv



(a) LBP feature [23]



(b) LBSP feature [8]

Figure 2: Examples of small-scale binary feature lookup patterns where the central regions are used as references. Regions identified with ones are sampled and have their comparison result assigned to a descriptor bit, while regions with zeros are skipped.

(LBSP) descriptor (shown in Figure 2b) which was introduced in [8] for background texture modeling in video sequences. This descriptor compares intensity samples over 5x5 regions to generate binary strings with $N = 16$ bits; its comparison operator is defined by

$$s(i_p\,,\, i_x) = \begin{cases} 1 & \text{if } |i_p - i_x| \geq T_x \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $T_x > 0$ is a dynamic threshold for x, which we rediscuss in Section 3. We chose this descriptor for our experiments as its pattern can be used to mimic the discrete sampling of a 5x5 Gaussian kernel, and (3) is an ideal operator for the detection of strong intensity gradients. Moreover, its compact 16-bit size makes it an ideal candidate for a vectorized implementation using Single Instruction, Multiple Data (SIMD) instructions. In theory, any binary descriptor with similar characteristics could be used to obtain relatively good performance in our experiments — as we will rediscuss in Section 6, only minor changes would be required to compute the "convolutions" of another binary feature. Besides, note that (3) can be viewed as a simplification of the ternary comparison operator of [28] in which the sign bit is ignored.

As stated earlier, small-scale binary feature descriptors such as LBPs are better suited for local texture description than high-level keypoint matching. This latter problem is better tackled by large-scale binary features with invariance properties such as BRIEF [10], ORB [25], BRISK [16], FREAK [2], or LATCH [17], which typically compare several hundred samples in large pixel neighborhoods. For more details on the different types of binary features, the interested reader is referred to recent surveys [15, 22, 34]. In Sections 2 and 3 below, we now introduce the two image processing operations we aim to mimic using binary feature convolutions.

2. Pattern Lookup and Pyramid Generation

The computation of an LBP-like feature can be separated in two steps: first, N data samples (usually pixel intensities) are retrieved from the analyzed image according to the predetermined pattern (the "lookup" step). Then, these samples are tested using (1) and a comparison operator, e.g. (2) or (3), to generate the binary string (the "threshold"

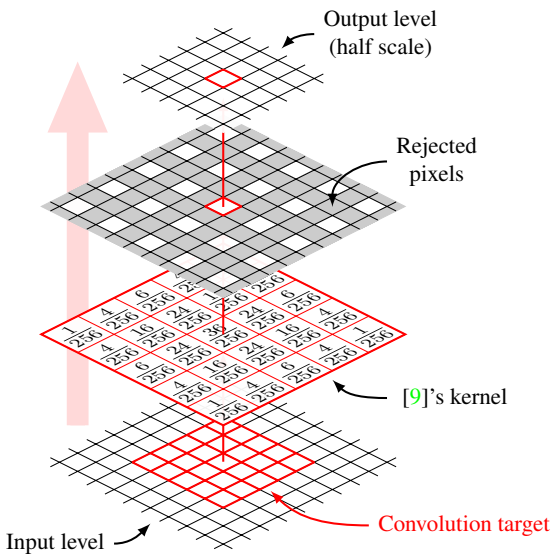


Figure 3: Illustration of the traditional pyramid reduction approach using the 5x5 Gaussian-like kernel of Burt and Adelson [9]. The lowest slice represents the input image, which is convolved and reduced to obtain the next pyramid image.

step). This separation is useful when parameter sweeps of the comparison operator are required (as the lookup step is constant), or when data vectorization is used. In this section, we focus on reusing the information gathered by the first step only.

Centrosymmetric binary lookup patterns such as the ones shown in Figure 2 can be viewed as approximations of Gaussian kernels where samples all have equal weights. Computing the average value of all samples during the lookup step is therefore similar to convolving this image using a binary low-pass filter. The cost of this operation is very low, as it only involves N additions and one division. For a separable $k \times k$ filter, this would demand $2k$ multiplications, $2k$ additions, and one division. Therefore, given the pattern of Figure 2b where $N = 16$, our approach is faster than using any similarly sized filter ($k = 5$). The convolved images we obtain can then be used for smoothing, noise suppression, or to create an image pyramid.

Image pyramids are often used in computer vision tasks where the multiscale analysis of image characteristics (e.g. textures or edges) is required. The *de facto* standard followed by most image processing libraries (including OpenCV and Matlab's) for image pyramid generation has been introduced by Burt and Adelson [9]: each pyramid level is computed by first blurring the previous level with a Gaussian kernel, and then downsampling the resulting image by keeping only alternating rows and columns. An iteration of this is illustrated in Figure 3, where the widely used Gaussian-like kernel of [9] is shown.

As stated before, averaging the N values sampled according to a binary feature pattern may serve as a rough alternative to Gaussian blurring. We compare in Figure 4

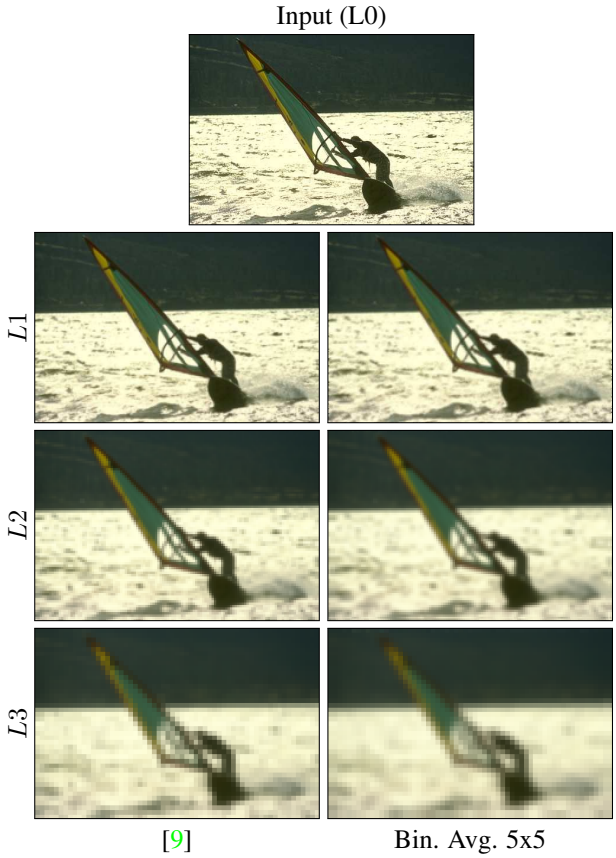


Figure 4: Visualizations of rescaled image pyramid levels obtained using [9]'s approach and using our proposed binary feature sample averaging technique.

individual image pyramid levels obtained by convolving an image using the 5x5 kernel of [9] as well as the 5x5, 16-bit binary feature pattern of Figure 2b. Each level is computed by resampling its predecessor directly, meaning errors can accumulate. Despite that, we can observe that the results are quite similar, even multiple levels down the pyramid. This shows that binary pattern sample averaging is a good approximation of Burt and Adelson's [9] approach, even though it relies on less information per pixel.

3. Thresholding and Gradient Estimation

We now discuss the second step in binary feature description, i.e. sampled value thresholding, and how it can be used to estimate image gradients. First, note that comparison operators based on absolute differences with respect to the central pixel such as (3) and the one introduced in [19] share an interesting property: since they produce non-zero bits for pixels with intensities significantly different from the central reference, the Hamming weight of their resulting binary string is strongest when the pattern is centered on a corner or an edge. This phenomenon has been exploited in the works of [8, 21], as detecting strong local gradients was helpful for both image classification and background modeling tasks.

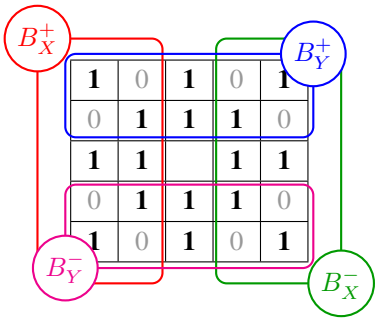


Figure 5: Illustration of which bits are used in the four reference binary strings (B_X^+, B_X^-, B_Y^+, and B_Y^-) of our 5x5 binary pattern. Bits which fall outside the highlighted areas for a given string are set to zero, meaning each reference string contains a total of six non-zero bits, and ten zero bits.

In our case, the comparison threshold T_x in (3) is responsible for distinguishing irrelevant noise from slight intensity variations. Using a value that directly scales with the central pixel intensity i_x would allow our binary descriptors to be fairly invariant to illumination changes, but it would also make them overly sensitive in dark regions, and insensitive in bright regions. In contrast, using a constant value would solve this sensitivity problem at the expense of the illumination invariance property. For our experiments, we determined empirically that gradients were best represented when both constant and scaling components were used simultaneously. As such, for the results presented in the following sections, we use

$$T_x = i_x \cdot T_s + T_c, \tag{4}$$

where we fix $T_s = 1/4$ and $T_c = 20$ for RGB and grayscale images.

Recreating accurate gradient orientation and magnitude maps using the information contained in precomputed binary descriptors with only $N = 16$ bits of information is quite challenging. Traditionally, gradients are estimated for the X and Y axes independently by convolving the analyzed image using Prewitt or Sobel operators. These are discrete differentiation operators that approximate the derivative functions used for continuous 2D signals. The two convolved images then obtained (noted G_X and G_Y) can be used to estimate pixel-wise gradient orientations via

$$\Theta(x) = \arctan2\Big(G_Y(x), G_X(x)\Big), \tag{5}$$

and gradient magnitudes via

$$G(x) = \sqrt{G_X(x)^2 + G_Y(x)^2}. \tag{6}$$

In our case, while we could apply differentiation operators on the values sampled in the lookup step (Section 2), directly using the binary descriptors allows for a much cheaper (yet seemingly accurate enough) solution.

In short, to determine the gradient magnitude $G(x)$ of a pixel x, we directly use the Hamming weight of its binary descriptor $B(x)$ computed via (1) instead of computing it via (6). For our 16-bit pattern, this means gradient magnitude values will be restricted to the $[0, 16]$ range. While this range is much smaller than the ones typically found with Prewitt or Sobel-based approaches, we will later show that it is actually quite sufficient for obtaining fairly accurate gradient maps. Next, given our knowledge of the descriptor's binary pattern, we can identify bit sets that are indicative of positive and negative gradients in the X and Y directions, and use them to form four reference binary strings. These strings are noted B_X^+, B_X^-, B_Y^+, and B_Y^-, and their bit sets are shown in Figure 5. Finally, $G_X(x)$ and $G_Y(x)$ can be approximated for x via

$$G_X(x) = \big(B(x) \cdot B_X^+\big) - \big(B(x) \cdot B_X^-\big), \tag{7a}$$
$$G_Y(x) = \big(B(x) \cdot B_Y^+\big) - \big(B(x) \cdot B_Y^-\big), \tag{7b}$$

where the dot product between two binary strings is used to compute a bitwise "and" as well as a Hamming weight. All four reference strings essentially act as binary-weighted operators, which can be viewed as Prewitt operators shaped like the lookup pattern. Their application in (7) can thus be seen as a convolution over a 2D binary signal. Note that using the 16-bit pattern of Figure 2b entails that $G_X(x)$ and $G_Y(x)$ can only take values in the $[-6, 6]$ range. Because of this, we cannot expect that $\Theta(x)$ values obtained via (5) will be very precise; however, they should be sufficient for applications that already bin gradient orientations into fewer than twelve sectors.

We show in Figure 6 examples of gradient maps which can be obtained with the binary feature convolution approach described so far, and compare them to the maps obtained via 5x5 Sobel convolutions. We can immediately observe that G_X and G_Y are not as accurate as their traditional counterparts, but our gradient magnitude maps are quite comparable, although seemingly noisier in some textured regions. This "noise" is due to the presence of fine-scale image structures which cause important localized gradient responses. This results in visual artifacts that can be considered irrelevant or harmful to some computer vision tasks. Larger convolution kernels or image preprocessing based on low-pass filtering are typically used to solve this problem. In our case, a comparable result can be achieved using the approximate image pyramid generation scheme we presented in Section 2: by min-pooling gradient map responses across all pyramid levels, fine-scale structures visible only in the lowest level can be easily suppressed. The advantage of min-pooling over cumulating gradient values is that it keeps the value ranges invariant to pyramid height, and it avoids edge smearing when upsampling via nearest-neighbor strategies. We show the new gradient maps obtained using only a two-level pyramid in Figure 7; we can now observe much lower noise levels for all maps, bringing them closer to Sobel's results.

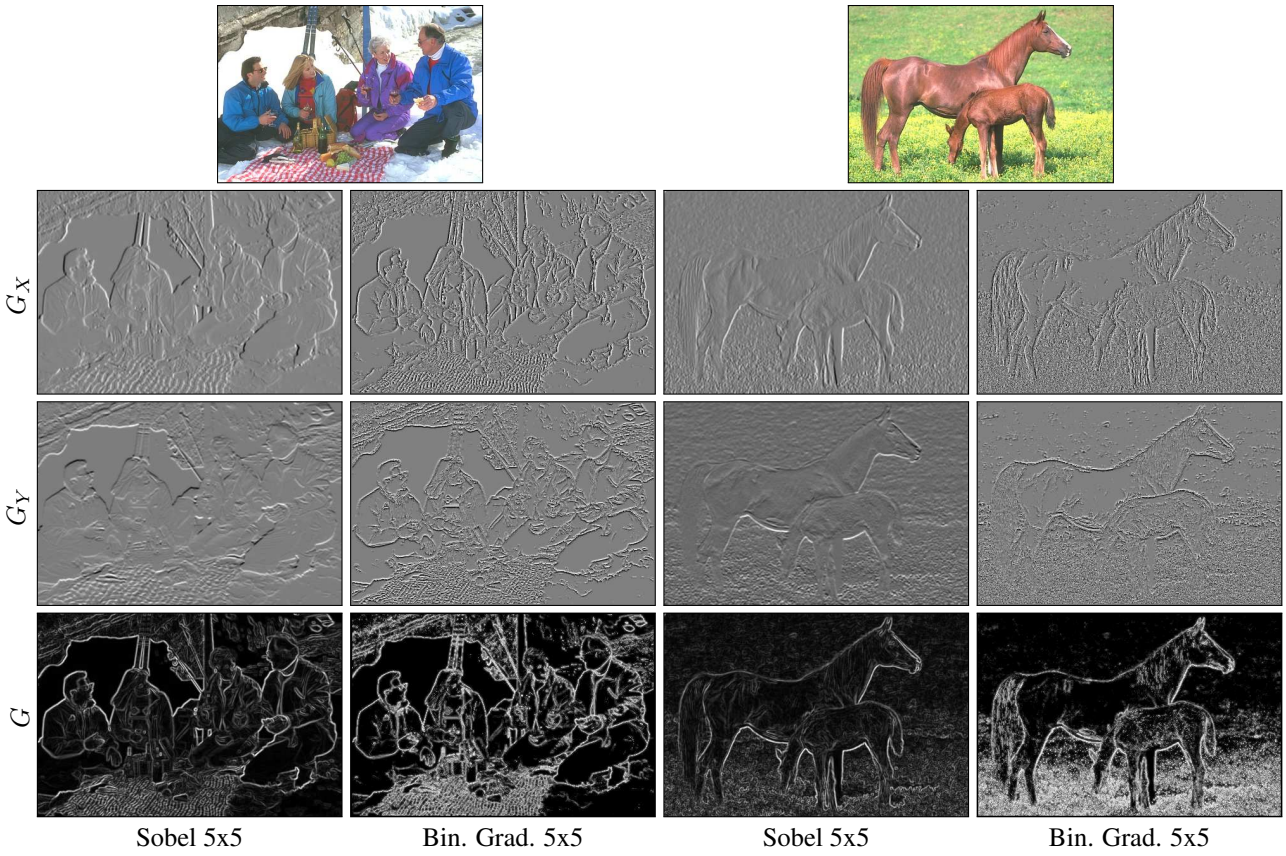


Figure 6: Visualizations of G_X, G_Y, and G maps obtained using 5x5 Sobel operators and 5x5 binary feature convolutions.

4. Edge Detection

To quantitatively assess the performance of binary feature convolutions and compare them to their traditional counterparts in a computer vision task, we opted to test our approach on the edge detection problem. We adapted an existing solution to use our proposed convolution approach for gradient estimation, and measure how well it performs. While many solutions have been recently proposed for edge detection [4, 12, 26], we picked the very well-known and formally defined algorithm introduced by Canny in [11] as our benchmark. Canny's method can be briefly summarized in three steps:

1. Estimate local gradients by convolving the analyzed image with a pair of differentiation operators;
2. Perform non-maximum suppression to localize and thin edges according to gradient orientations;
3. Assign and propagate edge labels through connected neighbors using hysteresis thresholding on gradient magnitudes.

Image gradient estimation (step 1) is performed using Sobel operators in most computer vision libraries. To test our binary feature convolution approach, we replaced this step with the approximate gradient-pyramid estimation described in the previous section. The other two steps were reimplemented based on the OpenCV source code of Canny's algorithm to make sure the new algorithm had similar complexity and constraints.

For our experiments, we used the latest version of the Berkeley Segmentation Data Set (BSDS500) for boundary detection [3]. This dataset contains 200 training, 100 validation, and 200 testing images, each of them provided with manually labeled edge maps. It comes with benchmarking tools that allow algorithms to be evaluated at different detection sensitivity settings, thus enabling precision-recall curves to be drawn (see Figure 8). Furthermore, the authors of [3] defined three standard measures for overall performance assessment, which we reuse here: the Average Precision (AP), the overall F-Measure for the Optimal Dataset Scale (ODS), and the overall F-Measure for the Optimal Image Scale (OIS). In the latter two measures, the "scale" refers to the edge detection sensitivity threshold (i.e. the sweep parameter).

For the modified algorithm, note that we determined the optimal parameter configuration of (4) using only the training and validation image sets of BSDS500, as required. Besides, Canny's high hysteresis threshold is used for the edge

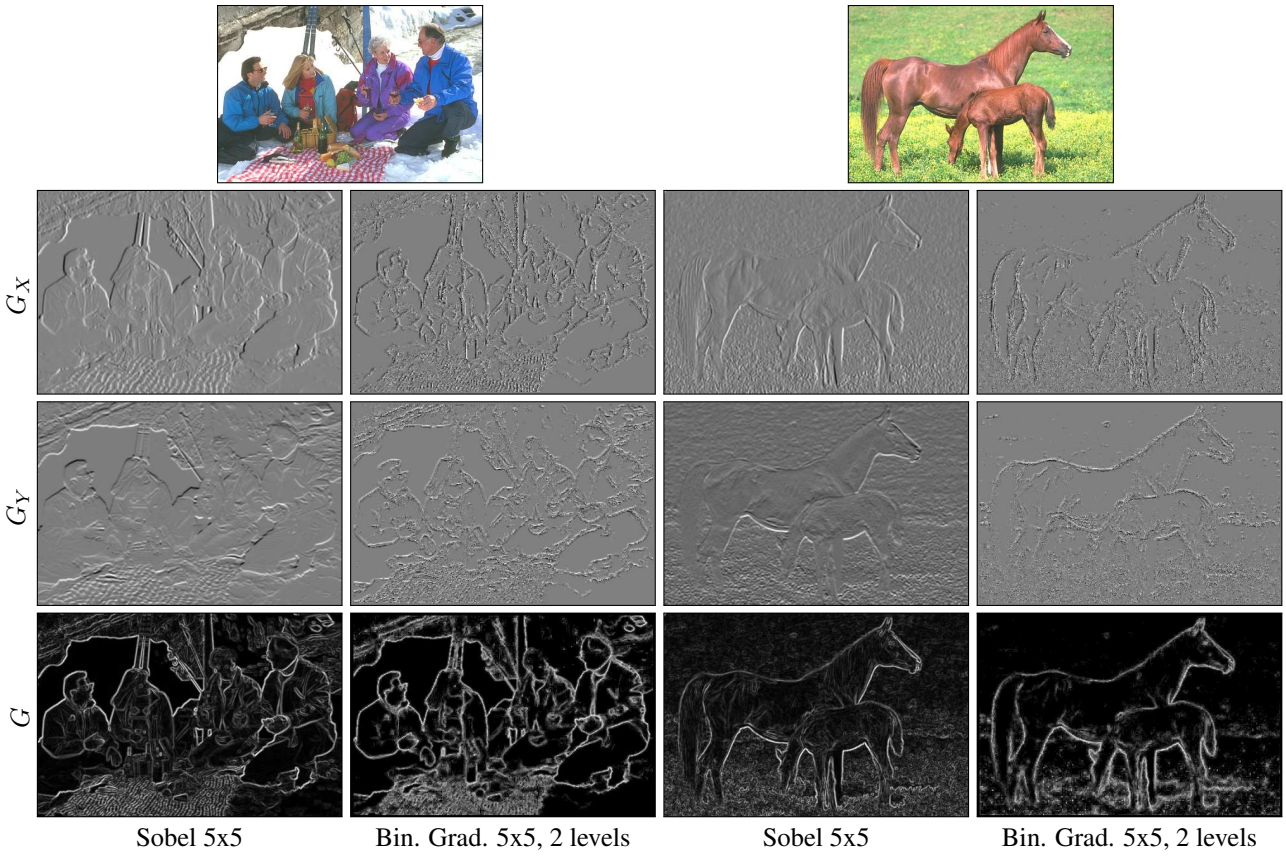


Figure 7: Visualizations of G_X, G_Y, and G maps obtained using 5x5 Sobel operators and 5x5 binary feature convolutions with two pyramid levels.

detection sensitivity sweep, and the low threshold is set as half of the high threshold. This last rule is similar to the one used in Matlab's default configuration of Canny's method. All results shown in this section are taken exclusively from the BSDS500 test set. Note also that Canny's method typically requires analyzed images to be blurred using a Gaussian kernel as a preprocessing step to improve edge detection precision. We avoided this step in the new version of the algorithm, but kept it for the Sobel version to recreate the traditional Canny implementation (with $\sigma = \sqrt{2}$). Finally, for RGB images, note that we used the maximum gradient across all three channels for the non-maximum suppression and hysteresis thresholding steps.

First, in Table 1, we can observe that our binary feature convolution-based gradient estimation approach (noted *Bin. Grad.*) offers the best ODS and OIS performances when used with at least two pyramid levels. When used without pyramiding, binary gradient estimation resulted in a substantial performance drop according to all measures compared to classic estimation via Sobel operators. This is not surprising, as we already noted in Section 3 that our naive pyramid-less approach was prone to noise caused by fine-scale image structures. Besides, using image pyramids for binary gradient estimation comes at a very low cost (we will discuss this further in Section 5), so it should not be seen as an unfair improvement over Canny's original method, which is already advantaged due to its preprocessing step.

Method	ODS	OIS	AP
Gradient with 3x3 Sobel	0.603	0.638	0.579
Gradient with 5x5 Sobel	0.613	0.648	0.590
Bin. Grad. 5x5 (no pyr.)	0.538	0.574	0.345
Bin. Grad. 5x5, 2 levels	0.618	0.648	0.550
Bin. Grad. 5x5, 3 levels	**0.638**	**0.664**	**0.616**
Human	0.80	0.80	-

Table 1: Overall performance of Canny edge detection on the BSDS500 dataset for the configurations listed in Figure 8; the best values are in bold red text, and the second best in red only. Human scores are also listed at the bottom.

As for the precision-recall curves shown in Figure 8, we can observe that our proposed gradient estimation approach with a 3-level pyramid outperforms Sobel's gradient estimation in the high precision regime due to proper gradient artifact suppression. The performance of binary gradient estimation when using a 2-level pyramid is slightly better than Sobel's (with both 3x3 and 5x5 operators). Besides, note that all the curves for our proposed configurations do not

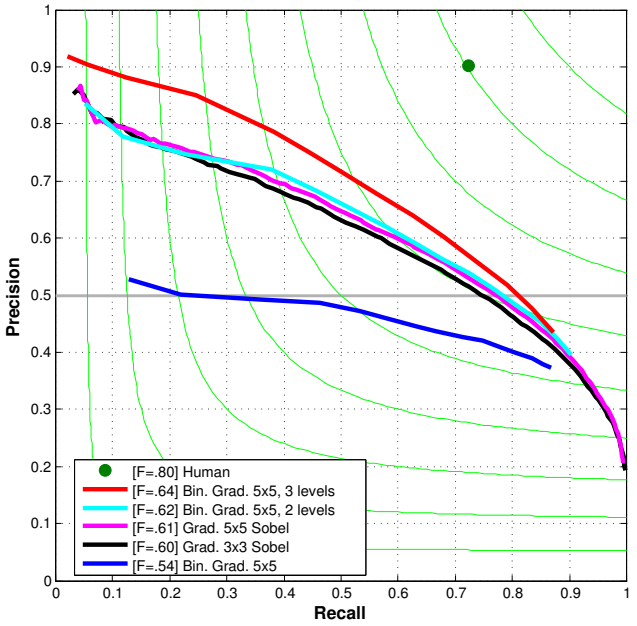


Figure 8: Results of Canny edge detection on the BSDS500 dataset using our binary gradient estimation approach (Bin. Grad.) with a 5x5 binary pattern and various pyramid levels, as well as using traditional 3x3 and 5x5 Sobel gradient estimation approaches.

stretch very far into the high recall regime; this is due to the presence of the absolute threshold component in (4), which prevents very small intensity variations from being captured by the binary descriptor. Finally, we show some examples of edge detection confidence maps overlapped with the groundtruth in Figure 9, where we can also observe fewer false positives for our approach.

5. Complexity Analysis

Approximating gradient maps by relying on binary features would not be appealing if it was slower than resorting to traditional gradient estimation means. For highly time-constrained applications that already require dense binary feature computations over large image regions, the potential speed gain can be very important. However, for applications that have no need for such features but still require gradient estimations, our approach could still be interesting due to its greater efficiency. We now roughly compare the complexity of our proposed approach (with a 5x5 binary pattern) to that of a traditional gradient estimation approach (using 5x5 Sobel operators) below.

First, in the traditional approach, consider that images are often preprocessed with a low-pass filter (e.g. the Gaussian blur used in Canny's method) before convolving them with differentiation operators in order to eliminate noise. The cost of this step depends on the filter's size, but let us consider Matlab's implementation of Canny's preprocessing method (which we used earlier) that relies on a 7x7 kernel. Since Gaussian kernels are separable, this first convolution results in 28 operations per pixel (14 multiplications and 14 additions). Next, the convolutions behind $G_X(x)$ and $G_Y(x)$ use two (also separable) 5x5 Sobel operators that result in 42 operations (10 multiplications, 10 additions, and one division each), for a total of 70 operations per pixel. Gradient magnitudes are then finally obtained via (6), which involves a square root computation. Besides, we ignore the cost of gradient orientations computations as our proposed approach also relies on (5).

On the other hand, gradient estimation via binary feature convolutions does not rely on preprocessing. The binary comparison threshold T_x computed via (4) can be obtained through a lookup table, so fetching it counts as a single operation. The computation of descriptors via (1) and (3) requires five operations per bit: two for the absolute difference, one for the comparison, one for shifting the boolean result, and one for placing it in $B(x)$. This results in 80 operations for our 16-bit pattern. The gradient magnitude value is then obtained using a single operation (Hamming weight) on this binary string. Finally, $G_X(x)$ and $G_Y(x)$ values can be obtained via (7) for five operations each (two bitwise ands, two Hamming weights computations, and one subtraction). This results in a total of 92 operations per pixel. While this total is greater than the one for the traditional approach, all operations used here are solely bitwise or based on integer arithmetics, so they will use far fewer CPU cycles than the floating point operations used in the traditional approach (the actual cycle counts is architecture-dependent). Besides, our binary gradient estimation approach also has other computational and architectural advantages: it requires less indexing and sampling operations on the analyzed image (the 5x5 pattern we used covers 17 pixels instead of 25), and it generates fewer cache misses (G_X and G_Y are essentially computed in a single pass).

Using pyramids for multiscale gradient estimation adds a few operations per pixel for min-pooling and upsampling via nearest-neighbor copy, and then multiplies the total operation count per pixel by the pyramid height. However, for each new pyramid level, only a quarter of the last level's pixels require new binary features to be computed and have their gradients estimated and pooled. Therefore, the total image-wide operation count (or, indirectly, the time cost) T_L for computing a L-level gradient pyramid can be approximated as a multiple of the first level's cost, noted T_0:

$$T_L = T_0 \cdot \sum_{l=0}^{L} \left(\frac{1}{4}\right)^l \tag{8}$$

This relation is a convergent series, where

$$\lim_{L\to\infty} \sum_{l=0}^{L} \left(\frac{1}{4}\right)^l = \frac{4}{3} \tag{9}$$

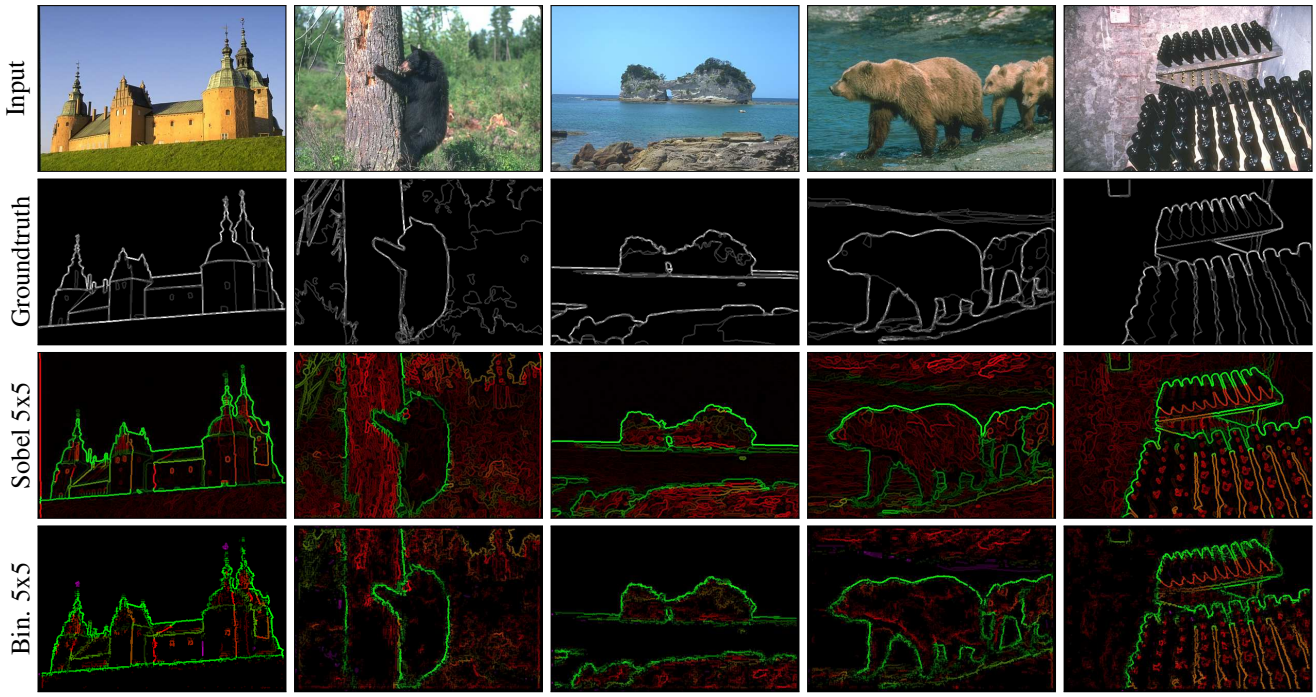


Figure 9: Visualizations of edge detection confidence maps roughly overlapped with the groundtruth. The results were obtained using Canny with 5x5 Sobel operators and 5x5 binary feature convolutions (with 3 pyramid levels). Edges are dilated using a 3x3 cross kernel to improve visibility. Green pixels indicate true positives, red pixels false positives, and magenta pixels false negatives. Brighter pixels indicate better detection confidence, or in other words, that the pixel was detected more often for different detection sensitivity thresholds. Note that for our approach, the detected edges are often jagged due to our rough gradient orientation approximation strategy.

meaning that no matter the pyramid height, the final gradient estimation cost will never exceed $\frac{4 \cdot T_0}{3}$. Therefore, our approach is still unlikely to be slower than the traditional gradient estimation approach, as it requires far fewer CPU cycles per pixel to compute the initial gradients (i.e. the first pyramid level).

In practice, for the implementations used in the experiments of Section 4, our binary gradient-based edge detector processed the BSDS500 images (with full detection threshold sweeps) on an Intel i7 mobile processor at 4.7 Hz without pyramid generation, and at 3.7 Hz with a 3-level pyramid. OpenCV's Canny implementation ran under the same conditions at 0.5 Hz, relying on highly optimized convolution subroutines. SSE SIMD instructions were disabled in both cases for a fair comparison.

6. Conclusion

We introduced a new strategy for gradient orientation and magnitude estimation based on binary features. While our "binary feature convolutions" are especially beneficial to time-constrained applications which already rely on such features, we expect that some applications will opt for this approach due its very low computational cost. Through our experiments, we showed that using a 5x5, 16-bit binary feature pattern results in image pyramids and gradient maps visually similar to those obtained via traditional means, and in slightly increased precision in edge detection when used in Canny's method. The efficiency of our approach could also be further improved by using the integer arithmetic SIMD instructions of modern hardware architectures.

Besides, our binary feature convolutions could theoretically be applied to a wide variety of binary feature patterns, whether small-scale or large-scale. Image pyramid generation only requires binary patterns to be (roughly) centrosymmetric, while gradient magnitude and orientation estimation can work with any similarity-based comparison operator, as long as oriented bit sets are provided with the pattern (e.g. like in Figure 5). Our approach could also be generalized to gradient estimation for 3D meshes and 4D medical imaging.

Acknowledgment

This work was supported in part by NSERC, FRQ-NT team grant No. 2014-PR-172083, and by REPARTI (Regroupement pour l'étude des environnements partagés intelligents répartis) FRQ-NT strategic cluster. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

References

[1] T. Ahonen, A. Hadid, and M. Pietikäinen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(12):2037–2041, 2006.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast retina keypoint. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 510–517, 2012.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

[4] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 328–335, 2014.

[5] V. Balntas, L. Tang, and K. Mikolajczyk. BOLD - binary online learned descriptor for efficient image matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2367–2375, 2015.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Comput. Vis. and Image Understanding*, 110(3):346 – 359, 2008.

[7] O. Bilaniuk, E. Fazl-Ersi, R. Laganiere, C. Xu, D. Laroche, and C. Moulder. Fast LBP face detection on low-power SIMD architectures. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, June 2014.

[8] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier. Change detection in feature space using local binary similarity patterns. In *Proc. Int. Conf. Comput. Robot Vis.*, pages 106–112, 2013.

[9] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Commun.*, 31(4):532–540, 1983.

[10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proc. European Conf. Comput. Vis.*, pages 778–792, 2010.

[11] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[12] P. Dollar and C. Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2015.

[13] Z. Guo, D. Zhang, and D. Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Trans. Image Process.*, 19(6):1657–1663, 2010.

[14] M. Heikkila and M. Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):657–662, 2006.

[15] J. Heinly, E. Dunn, and J.-M. Frahm. Comparative evaluation of binary features. In *Proc. European Conf. Comput. Vis.*, pages 759–773, 2012.

[16] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2548–2555, 2011.

[17] G. Levi and T. Hassner. LATCH: learned arrangements of three patch codes. *CoRR*, abs/1501.03719, 2015.

[18] S. Liao, M. Law, and A. Chung. Dominant local binary patterns for texture classification. *IEEE Trans. Image Process.*, 18(5):1107–1118, 2009.

[19] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1301–1306, 2010.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004.

[21] S. Manivannan, R. Wang, and E. Trucco. Extended gaussian-filtered local binary patterns for colonoscopy image classification. In *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, pages 184–189, Dec 2013.

[22] L. Nanni, A. Lumini, and S. Brahnam. Survey on LBP based texture descriptors for image classification. *Expert Systems with Applications*, 39(3):3634–3641, 2012.

[23] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognit.*, 29(1):51 – 59, 1996.

[24] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002.

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to sift or surf. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2564–2571, 2011.

[26] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3982–3991, 2015.

[27] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. SuBSENSE: A universal change detection method with local adaptive sensitivity. *IEEE Trans. Image Process.*, 24(1):359–373, 2015.

[28] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Trans. Image Process.*, 19(6):1635–1650, 2010.

[29] N. Werghi, C. Tortorici, S. Berretti, and A. Del Bimbo. Representing 3D texture on mesh manifolds for retrieval and recognition applications. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2521–2530, June 2015.

[30] J. Ylioinas, A. Hadid, Y. Guo, and M. Pietikäinen. Efficient image appearance description using dense sampling based local binary patterns. In *Proc. Asian Conf. Comput. Vis.*, pages 375–388, 2013.

[31] J. Zhang, K. Huang, Y. Yu, and T. Tan. Boosted local structured hog-lbp for object localization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1393–1400, June 2011.

[32] S. Zhang, H. Yao, and S. Liu. Dynamic background modeling and subtraction using spatio-temporal local binary patterns. In *Proc. IEEE Int. Conf. Image Process.*, pages 1556–1559, 2008.

[33] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):915–928, 2007.

[34] Y. Zhao. Theories and applications of LBP: A survey. In *Adv. Intelligent Comp. Theories and Appl. With Aspects of Artif. Intell.*, volume 6839, pages 112–120. Springer, 2012.