VLFeat - An open and portable library of computer vision algorithms

Andrea Vedaldi
Department of Engineering Science
Oxford University
Oxford, UK
vedaldi@robots.ox.ac.uk

Brian Fulkerson
Computer Science Department
University of California at Los Angeles
Los Angeles, CA, USA
bfulkers@cs.ucla.edu

ABSTRACT

VLFeat is an open and portable library of computer vision algorithms. It aims at facilitating fast prototyping and reproducible research for computer vision scientists and students. It includes rigorous implementations of common building blocks such as feature detectors, feature extractors, (hierarchical) k-means clustering, randomized kd-tree matching, and super-pixelization. The source code and interfaces are fully documented. The library integrates directly with Matlab, a popular language for computer vision research.

Categories and Subject Descriptors

D.0 [**Software**]: General; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding

General Terms

Algorithm, design, experimentation

Keywords

Computer vision, object recognition, image classification, visual features

1. INTRODUCTION

Current computer vision research builds on established vision and machine learning algorithms such as feature extraction, clustering, and learning. Most of these foundational algorithms are relatively new and experimental, and when an implementation is available, it is often only in binary form and only for a few specific platforms. The lack of source code availability poses operative restrictions that hamper the adoption of new algorithms; most importantly, it makes difficult to *learn* the algorithm details and to *modify* them to implement novel research ideas. This does not favour open, reproducible, and efficient research and undermines the educational value of these closed-source implementations.

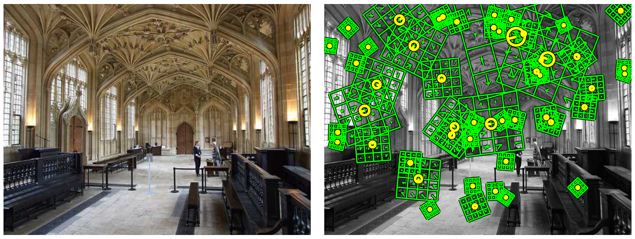

Figure 1: SIFT. Left: an input image. Right: examples of detected SIFT features along with their descriptors.

VLFeat [14] bundles high-quality implementations of common computer vision algorithms in an open, flexible, and portable package. The intended users are computer vision researchers and students. As such, the library strives for convenience of use, openness, and rigor. The library includes optimized implementations (e.g. SSE2 based computation of kernel matrices) for algorithms that are likely to be bottlenecks in computer vision research. VLFeat source code and algorithms are fully documented and their usage is exemplified by numerous example programs. VLFeat has no external software dependencies (beyond the C runtime), simplifying compilation and encouraging users to study and modify the code.

VLFeat is usually accessed through its Matlab[1] interfaces, although a partial command line interface is available, as well as third party Python bindings [12]. The interfaces access algorithms implemented in a core library, written in C. The choice of the C language favours portability and binary compatibility. It also makes the code simple to understand for the large base of programmers familiar with C and its variants.

Section 2 introduces some of VLFeat's functionalities, Section 3 shows how to obtain and run VLFeat, Section 4 illustrates an application to semantic image categorization, Section 5 motivates VLFeat design, and Section 6 summarizes the material and suggests future directions.

2. ALGORITHMS

This section introduces a selection of algorithms implemented in VLFeat.

SIFT feature detector and descriptor (Fig. 1). The Scale Invariant Feature Transform (SIFT) [8, 9] is probably

MM'10, October 25–29, 2010, Firenze, Italy.

[1] Experimental support for GNU Octave is included.

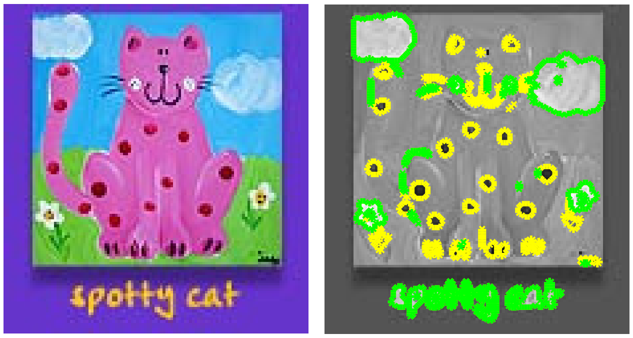

Figure 2: MSER. Left: original image. Right: detected MSER regions (green: positive level sets, yellow: negative level sets).

the most popular feature used in computer vision. SIFT detects salient image regions (keypoints) and extracts discriminative yet compact descriptions of their appearance (descriptors). SIFT keypoints are invariant to viewpoint changes that induce translation, rotation, and rescaling of the image. Keypoints from multiple views of the same scene can be put in correspondence by comparing their descriptors. This may be used as a basis for a three-dimensional reconstruction of the scene. Alternatively, keypoints with discretized descriptors can be used as *visual words* [13] as an intermediate image characterization. Histogram of visual words can then be used by a classifier to map images to abstract visual classes (e.g. car, cow, horse).

Despite its popularity, the original SIFT implementation is available only in binary format [8]. The VLFEAT implementation is output equivalent.

Dense SIFT. Dense SIFT is a fast algorithm for the computation of a dense set of SIFT descriptors. Some of the best performing image descriptors for object categorization use these descriptors (see Section 4).

MSER feature detector (Fig. 2). Maximally Stable Extremal Regions (MSER) [10] is a robust and fast feature detector. In contrast to SIFT keypoints, MSERs are invariant to full affine transformations, which makes them suitable to track flat surface elements through arbitrary viewpoint changes. The VLFEAT MSER implementation supports data of arbitrary dimension, extending MSERs to video sequences and volumetric data.

Randomized kd-trees (Fig. 3). A kd-tree is a hierarchical structure for fast (approximate) nearest neighbor computation and similarity queries. VLFEAT implements the randomized kd-tree variant introduced by FLANN [11].

K-Means (Fig. 4). K-means is a standard clustering algorithm often used on large sets of feature descriptors for the computation of dictionaries of visual words. VLFEAT implements the standard Lloyd [7] k-means algorithm and a variant from Elkan [2]. For large datasets, the latter implementation can be an order of magnitude faster. VLFEAT also includes a hierarchical version, which can be used to efficiently create very large dictionaries.

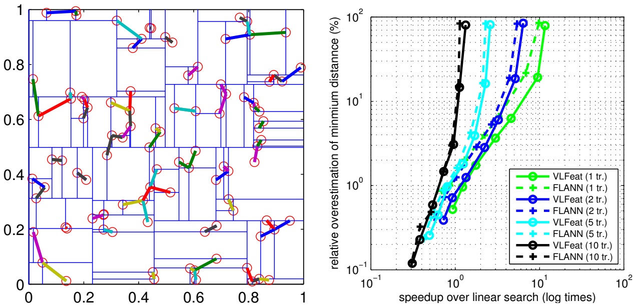

Figure 3: Randomized kd-tree forest. Left: a kd-tree used to find nearest-neighbors relations in a set of 2D points. Right: speedup in the approximated NN matching of two sets of SIFT features (comparison between FLANN [11] and VLFEAT implementations).

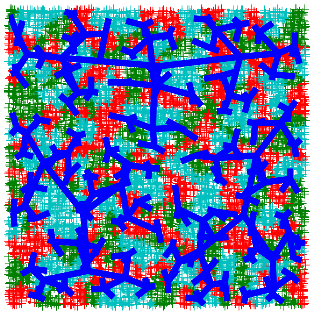

Figure 4: Hierarchical k-means tree computed from of a dataset of 2-D points uniformly distributed on a square. k-means can be used to construct large yet efficient vocabularies of visual words.

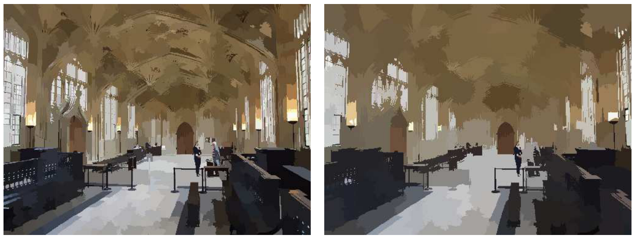

Figure 5: Quick shift superpixels at two different scales. Superpixels over-segment an image into visually coherent parts and can be used to simplify further processing.

Fast distances and similarity computation. Often, algorithms need to calculate the mutual similarity of large sets of vectors. For instance, repeated evaluation of a similarity measure is the basis for the evaluation of non-linear support vector machines. VLFEAT includes SSE2 optimized routines to do this, with 2x or 4x speedup over trivial implementations.

Quick shift superpixels (Fig. 5). Quick shift [15] is a mode seeking algorithm like mean shift that can be used to partition an image into a set of superpixels. Unlike mean shift, quick shift does not iteratively shift each point towards a local mode; instead, it forms a tree of links to the nearest neighbor which increases an estimate of the density.

Agglomerative information bottleneck (AIB) (Fig. 6).

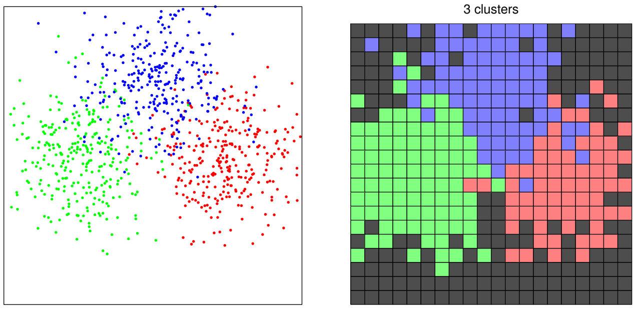

Figure 6: Agglomerative Information Bottleneck clusters discrete data while preserving the mutual information between the data and its labels.

Informative visual vocabularies tend to be significantly large, including thousands of visual words. Compression techniques can be used to reduce the dimensionality of such vocabularies while preserving the descriptive power of the visual words. The Agglomerative Information Bottleneck (AIB) algorithm is one such technique. It works by iteratively merging pair of visual words (or more generally, any discrete alphabet) while preserving the mutual information between words and the image visual class.

3. GETTING STARTED

Installation. The VLFeat website [14] includes both source code and binary packages. Binaries are available for Linux, Mac OS X, and Windows in 32 and 64 bit variants. Installation in a Matlab environment is as simple as downloading the binary package and including VLFeat to Matlab's path by adding the command `vl_setup` to Matlab's startup file.

Compiling from source. VLFeat has no external dependencies. On GNU/Linux and Mac OS X typing `make` is usually enough (this assumes that GCC is installed; compiling the Matlab interface requires the Matlab `mex` command to be in the path). For Windows, a `nmake` makefile is included for compilation with Microsoft Visual C++. Detailed compilation instructions can be found in the VLFeat website [14].

Demos. The command `vl_demo` runs a suite of demos that demonstrate the usage of the VLFeat commands (the figures in this paper have been generated using this code). For instance, extracting SIFT features from the bundled image `a.jpg` image is as simple as:

```
im = imread(fullfile(vl_root, ’data’, ’a.jpg’)) ;
im = im2single(rgb2gray(im)) ;
[frames, descriptors] = vl_sift(im) ;
```

The resulting keypoint frames and descriptors can then be visualized by

```
image(im) ; hold on ;
vl_plotsiftdescriptor(descriptors, frames) ;
```

Documentation. The Matlab commands are self documented (e.g. `help vl_sift`); an HTML version of the Matlab command documentation is available from the VLFeat website. Detailed information about the algorithms and API is embedded in the source code of the VLFeat core library (in Doxygen format) and a copy is available on the website as well. Beyond detailing the VLFeat C interface, this documentation includes a full technical description of the implemented algorithms.

4. EXAMPLE: IMAGE CATEGORIZATION

Image categorization is the problem of assigning an image to one of a number of visual categories. VLFeat can be used to obtain state-of-the-art image descriptors in the Caltech-101 [4] classification benchmark. The full code of this example, a single Matlab file, can be downloaded from [14]. This combines the following building blocks:

Feature extraction. VLFeat `vl_dsift` is used to efficiently compute a dense set of multi-scale SIFT descriptors from a given input image (PHOW descriptors [1]). While `vl_sift` can also be used for this purpose, `vl_dsift` is an order of magnitude faster.

Vocabulary learning. VLFeat `vl_kmeans` is then used to cluster a few hundred thousands visual descriptors into a vocabulary of 10^3 visual words. `vl_kmeans` implements the Elkan algorithm, which is for this task several times faster than the standard Lloyd k-means implementation, and orders of magnitude faster than the Matlab native k-means.

Spatial histograms. A *spatial histogram* [6] characterizes the joint distribution of appearance and location of the visual words in an image. VLFeat `vl_kdtreequery` can be used to map visual descriptors to visual words efficiently by using a KD-Tree and VLFeat `vl_binsearch` and `vl_binsum` can be used to quickly accumulate the visual words into a spatial histogram.

Training of a non-linear SVM. The spatial histograms are used as image descriptors and fed to a linear SVM classifier. Linear SVMs are very fast to train [3], but also limited to use an inner product to compare descriptors. Much better results can be obtained by pre-transforming the data through `vl_homkermap`, which computes an explicit feature map that "emulates" a non linear χ^2-kernel as a linear one [16]. Then, training with 15 images for each of the 101 Caltech-101 categories can be done by using `vl_pegasos`, an implementation of stochasting gradient SVM training.

Results. The computation and quantization of the dense SIFT features and testing of the SVM requires under a quarter of a second for each image. Training the SVM requires less than a minute. In terms of accuracy Fig. 7 shows that the PHOW features, efficiently computed using VLFeat, are a state-of-the-art image descriptor (better classification results can be obtained by combining a variety of different descriptors [5]).

5. DESIGN

Granularity. VLFeat avoids implementing specific applications (e.g. there is no face detector); instead it focuses in cimplementing algorithms (feature detectors, clustering, etc.) that can be combined in order to produce full applications.

Openness. VLFeat is distributed under the GNU GPL license. The source code is fully documented in Doxygen format. The library is written in as simple a manner as possible to provide a good trade-off between speed and un-

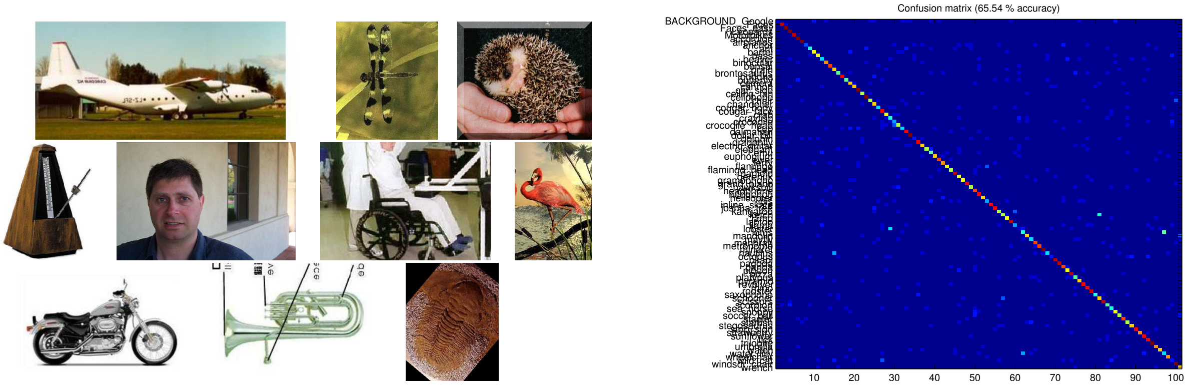

Figure 7: Caltech-101 image categorization example. Left: some images from the Caltech-101 benchmark. Right: confusion matrix obtained by using a χ^2-SVM on top of 4×4 spatial histograms and dense SIFT features, computed by using VLFeat. This is one of the best image descriptors available for this type of task.

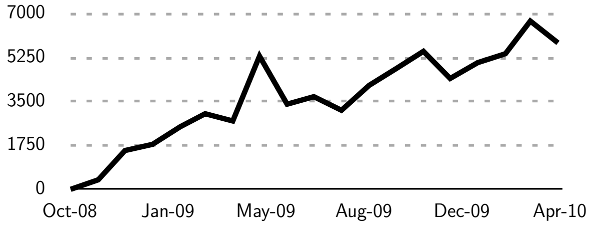

Figure 8: VLFeat impact. Number of visits to vlfeat.org per month since October 2008.

derstandability. This encourages users to study and possibly modify VLFeat source code.

Portability. A goal of VLFeat is to provide binaries that can be readily used in Linux, Mac OS X, and Windows. Portability is important in a research environment, where different computational platforms are likely to be in use, perhaps concurrently. Portability is enhanced by the fact that VLFeat is written in C and has essentially no external dependencies. It is compatible with GNU/Linux (GNU GCC), Windows (Visual C), and Mac OS X (GNU GCC/Xcode). The project also provides pre-compiled binaries for most architectures.

Speed. VLFeat's main focus is not speed, but certain portions of the library have been optimized for speed when this was likely to constitute a bottleneck for research. Example include feature extraction, kd-tree matching, and vector comparisons. Some functions, such as image convolution and distance calculations, are SSE enhanced.

6. CONCLUSIONS

We have introduced VLFeat, a library of computer vision algorithms for fast prototyping in computer vision research. VLFeat encapsulates powerful computer vision algorithms in a simple and portable package that encourages code understanding and customization. VLFeat is already popular in the computer vision community (Fig. 8). In the future, we would like to increase the number of contributors to the project.

Acknowledgments. We are grateful for financial support from ERC grant VisRec no. 228180 and ONR MURI N00014-07-1-0182.

7. REFERENCES

[1] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *Proc. ICCV*, 2007.

[2] C. Elkan. Using the triangle inequality to accelerate k-means. In *Proc. ICML*, 2003.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.

[4] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proc. ICCV*, 2003.

[5] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, 2009.

[6] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[7] S. Lloyd. Least square quantization in PCM. *IEEE Trans. on Information Theory*, 28(2), 1982.

[8] D. Lowe. Implementation of the scale invariant feature transform.
`http://www.cs.ubc.ca/~lowe/keypoints/`, 2007.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2(60):91–110, 2004.

[10] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, 2002.

[11] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. In *Proc. VISAPP*, 2009.

[12] M. Rousson. A VLFeat Python wrapper.
`http://github.com/mmmikael/vlfeat`, 2009.

[13] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.

[14] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms.
`http://www.vlfeat.org/`, 2008.

[15] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proc. ECCV*, 2008.

[16] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proc. CVPR*, 2010.