Global and Efficient Self-Similarity for Object Classification and Detection

Thomas Deselaers and Vittorio Ferrari
Computer Vision Laboratory, ETH Zurich, Zurich, Switzerland
lastname@vision.ee.ethz.ch

Abstract

Self-similarity is an attractive image property which has recently found its way into object recognition in the form of local *self-similarity descriptors [5, 6, 14, 18, 23, 27] In this paper we explore* global *self-similarity (GSS) and its advantages over local self-similarity (LSS). We make three contributions: (a) we propose computationally efficient algorithms to extract GSS descriptors for classification. These capture the spatial arrangements of self-similarities within the entire image; (b) we show how to use these descriptors efficiently for detection in a sliding-window framework and in a branch-and-bound framework; (c) we experimentally demonstrate on Pascal VOC 2007 and on ETHZ Shape Classes that GSS outperforms LSS for both classification and detection, and that GSS descriptors are complementary to conventional descriptors such as gradients or color.*

1. Introduction

Good image descriptors are the basis for many successful methods in computer vision. Shechtman *et al*. [23] first proposed a descriptor based on *local self-similarities* (LSS). Compared to conventional image descriptors, LSS is indirect: instead of measuring features such as gradients or color of a pixel, it measures how similar they are to the pixel's neighbors. The LSS descriptor captures the internal geometric layout of local regions and can be compared across images which appear substantially different at the pixel level.

This descriptor has been quickly adopted in the object detection and classification community [5, 6, 14, 18, 27]. While the original work [23] matches ensembles [4] of these descriptors, most later works use it as yet another feature in the *bag-of-visual-words* framework. This makes it easy to use LSS descriptors in the machine-learning frameworks which proved to work well for conventional local descriptors [6, 14, 18, 27].

In this paper we demonstrate that self-similarity can and should be used globally rather than locally to capture long-range similarities and their spatial arrangements. Fig. 1 shows two selected patches and their global self-similarity (GSS), as the patch correlation with the entire image. Contiguous (patch 1) and repeating (patch 2) structures can be well recognized. Patch 2 shows that GSS can capture long-range similarities within an image. The indirection characteristic of self-similarity results in similar patterns in the GSS images, although the original images appear very different. The spirit of self-similarity is that images are similar if the way patterns repeat within them is similar and not because they have similar colors or textures.

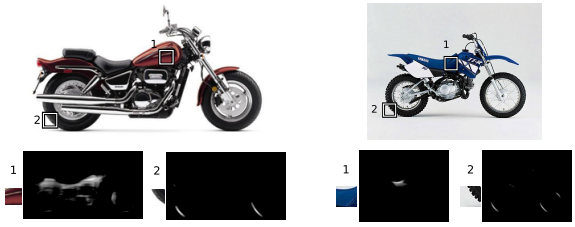

Fig. 1: **Global self-similarity:** self-similarity of two image patches with their respective images.

To fully exploit self-similarity we propose to consider it globally rather than locally. One drawback of GSS is that it is very expensive to compute if done directly (sec. 3.1). We first review existing works using (local) self-similarity (sec. 2), and how it is applied for object classification and detection. Then, we analyze GSS (sec. 3) and propose a computationally efficient method to obtain it (sec. 3.2), and to store it using very little memory. Next, we propose two descriptors based on GSS: *bag-of-correlation-surfaces* (sec. 4.1) and *self-similarity hypercubes* (sec. 4.2). To the best of our knowledge, we are the first to propose a GSS descriptor. Finally, we show how to use self-similarity hypercubes efficiently for object detection in the sliding-window framework (sec. 5.1) and in the branch-and-bound framework [17] (sec. 5.2). We analyze the computationally complexity of all descriptors and algorithms we present.

In sections 6 and 7 we experimentally evaluate several variants of our GSS descriptors and compare them to LSS for classification and for detection. Moreover, for classification we combine GSS and LSS descriptors with conventional cues such as histograms of oriented gradients (HOG) [8], GIST [22], and bag-of-visual-words [25]. The experiments reveal that: (i) GSS outperforms LSS; (ii) our efficient variants of GSS outperform the direct one, in addition to being computationally much more efficient; (iii) self-similarity descriptors are truly complementary to conventional descriptors, as their combination perform better than either alone. The notation used throughout this pa-

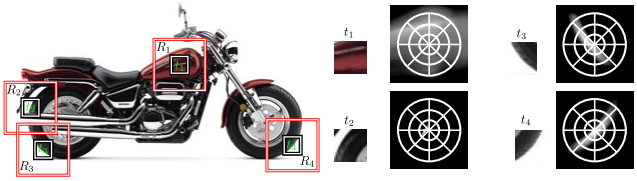

Fig. 2: **Local self-similarity descriptors.** The correlation surfaces of four patches t_p are computed and quantized using a log-polar grid.

per is summarized in tab. 1. Source code for the global self-similarity descriptors is available at http://www.vision.ee.ethz.ch/~{}calvin.

2. Local Self-Similarity (LSS)

LSS as proposed by Shechtman *et al.* [23] captures self-similarities within relatively small (40×40 pixel) regions (sec. 2.1). LSS has been used for object recognition and detection as yet another local descriptor in bag-of-visual-words frameworks [6, 14, 18, 27] (sec. 2.3) or in nearest-neighbor classifiers [5].

Junejo *et al.* [15] perform human action recognition in video using temporal self-similarities. They exploit that periodic motion (such as walking) results in periodic patterns easy to detect from temporal self-similarity. The self-similarity is computed from the distance of tracked local features and other cues such as HOG.

Another idea related to self-similarity is symmetry. Stark *et al.* [26] proposed a shape-based model for object recognition. To train it from very few samples they propose to transfer knowledge from known classes. They report that *local symmetries* are good features to transfer.

2.1. Original Local Self-similarity Descriptor [23]

The LSS descriptor $\mathcal{L}_p$ for pixel p measures the similarity of a small patch t_p around it with the larger surrounding region R_p [23] (fig. 2). It is computed as follows:
(1) Determine the $N \times N$ correlation surface $\mathcal{C}_p$ of the $w \times w$ patch t_p with the surrounding $N \times N$ region R_p. Both R_p and t_p are centered on p. $\mathcal{C}_p(x)$ is the correlation of t_p with a patch t_x centered on x:

$$\mathcal{C}_p(x) = \exp\left(-\frac{\mathrm{SSD}(t_p, t_x)}{\sigma}\right) \quad (1)$$

(2) Discretize the correlation surface $\mathcal{C}_p$ on a log-polar grid and store the maximal value of $\mathcal{C}_p$ within each grid bin:

$$\mathcal{L}_p(\rho, d) = \max_{x \in \mathrm{BIN}(\rho,d)} \{\mathcal{C}_p(x)\} \quad (2)$$

Tab. 1: Notation used throughout this paper.

symbol	description	symbol	description
p	pixel	$\mathcal{C}_p$	correlation surface for p
t_p	$w \times w$ patch around p	$\mathcal{M}$	prototype assignment map
R_p	$N \times N$ region around p	$\mathcal{B}$	BOCS descriptor
I	image	$\mathcal{H}$	$D_1 \times D_2 \times D_1 \times D_2$ SSH
$\mathcal{L}_p$	LSS descriptor of pixel p	Θ	codebook of k patch prototypes θ
$\mathcal{S}$	exact GSS tensor	Λ	codebook of correlation
$\tilde{\mathcal{S}}$	approximate GSS tensor		surface prototypes λ

Typically a few hundred $\mathcal{L}_p$ are extracted either at interest points or at position on a regular grid.

Shechtman and Irani [23] use these descriptors with an ensemble matching method [4] for recognition and retrieval.

2.2. Efficient Convolution using the FFT.

The cost to compute the LSS descriptor for pixel p is dominated by the computation of the correlation surface $\mathcal{C}_p$. This takes $N^2 w^2$ operations, as t_p must be correlated to N^2 patches t_x. Although not mentioned in [23], an easy speedup is to compute convolutions using the Fast Fourier Transform (FFT), resulting in $3N^2 \log N^2 + N^2$ operations ($N^2 \log N^2$ is the cost of one FFT; we have to perform three: FFT of R_p and t_p, and inverse FFT of the result. N^2 is the cost for pixelwise multiplication in the spectral domain). However here the speedup is marginal, as $N > w$.

2.3. Bag of local self-similarities (BOLSS)

Ensemble matching [4] allows to use the LSS descriptors for object detection and retrieval [23] but it cannot easily be integrated into existing frameworks and is computationally expensive. Most other object recognition frameworks require descriptors for an image, rather than a pixel. To use LSS descriptors in their own frameworks, various authors have used the *bag-of-visual-words* (BOW) approach leading to *bag-of-local-self-similarities* (BOLSS) [6, 14, 18, 27].

In the BOW approach [25, 28] an image is described as a collection of regions. Each region is described by its local appearance and the spatial relations between regions are ignored.

The region appearance space is vector quantized into a codebook of visual words, and the set of region descriptors for an image is represented as a histogram over visual words (one bin per word). For object categorization, typically 500-2000 words are used.

To create the BOLSS of an image, we follow [27]: (1) extract $\mathcal{L}_p$ on a regular 5×5 pixel grid (with $N = 40, w = 5$, 3 radial bins for d and 10 angular bins for ρ) [1]; (2) assign each $\mathcal{L}_p$ to one of the 300 visual words in the codebook.

This representation can easily be used in various classifiers such as support vector machines (SVMs) and it can be used for detection using sliding-windows [27] or efficient subwindow search (ESS) [17].

Given the visual word codebook, the effort to create the BOLSS for an $H \times W$ image is to extract $H/5 \times W/5$ descriptors and assigning them to visual words. This results in $H/5 \cdot W/5 \cdot 3N^2 \log N^2 + N^2$ operations for the extraction and $H/5 \cdot W/5 \cdot 3 \cdot 10 \cdot k$ operations for the assignment.

3. Global Self-Similarity Tensor $\mathcal{S}_I$ (GSS)

In the following we describe the GSS tensor $\mathcal{S}_I$ for an image I, which can be computed by directly extending LSS (sec. 3.1). We also propose an efficient approximation to GSS which is much faster to compute and uses far less

[1] we use the code from http://www.robots.ox.ac.uk/~{}vgg/software/Self-Similarity/

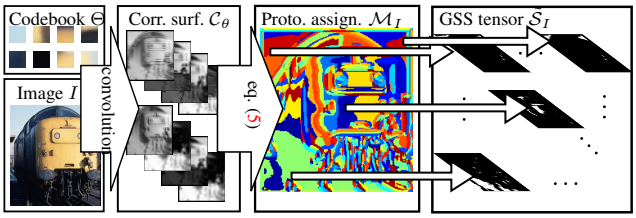

Fig. 3: **Computing the prototype assignment map $\mathcal{M}_I$.** The image is convolved with prototypes $\theta \in \Theta$; for each pixel p the prototype θ with maximal correlation is chosen (eq. (5)). The GSS tensor $\tilde{\mathcal{S}}_I$ is obtained from $\mathcal{M}_I$ using lookups.

memory (sec. 3.2). The GSS tensor extracted in this section forms the basis for GSS descriptors for object classification (sec. 4) and detection (sec. 5).

3.1. Direct global self-similarity

To compute the GSS tensor $\mathcal{S}_I$ for image I, we correlate t_p for each pixel $p \in I$ with the entire $H \times W$ image resulting in $H \times W$ different correlation surfaces $\mathcal{C}_p$ (as in eq. (1) but with $R_p = I$). $\mathcal{S}_I$ is a 4D tensor collecting the $\mathcal{C}_p$'s

$$\mathcal{S}_I(p, p') = \mathcal{C}_p(p') \qquad \forall p, p' \in I \tag{3}$$

For every pair of pixels $p = (x, y)$ and $p' = (x', y')$, $\mathcal{S}_I(p, p')$ is the correlation of a $w \times w$ patch t_p centered on p with patch $t_{p'}$ centered on p'. For a symmetric patch similarity measure (like SSD) $\mathcal{S}_I(p, p') = \mathcal{S}_I(p', p)$. However, any measure can be used.

The size of $\mathcal{S}_I$ is quadratic in the size of the input image I: $H \times W \times H \times W$. Thus, $\mathcal{S}_I$ is very large even for a moderately sized image.

To compute $\mathcal{S}_I$ we correlate the $w \times w$ patch t_p centered on p with the entire image for every pixel p. Doing this directly requires $H^2W^2w^2$ operations. Using the FFT (sec. 2.2) would result in speedups of factor 2-20 (depending on the choice of w and the image size). Example correlation surfaces $\mathcal{C}_p$ for four pixels are shown in fig. 1.

Note that the LSS descriptor can also be represented within this tensor, bringing a unified view on self-similarity.

3.2. Efficient Global Self-Similarity

In this section we present an efficient method to approximate the computation of $\mathcal{S}_I$ and show how the resulting $\tilde{\mathcal{S}}_I$ can be stored in a fraction of the memory.

To create $\tilde{\mathcal{S}}_I$ we quantize the patches t_p according to a codebook Θ of prototype patches θ (sec. 3.3). Then, we define that two patches t_p and $t_{p'}$ are similar if they are assigned to the same prototype θ. This is a valid assumption since similarity is transitive: if two patches t_p and $t_{p'}$ are both similar to a prototype θ, then they are also similar to each other ($t_p \simeq \theta \wedge t_{p'} \simeq \theta \Rightarrow t_p \simeq t_{p'}$). Further assuming that patches are only similar if they are assigned to the same prototype θ ($t_p \simeq \theta \wedge t_{p'} \simeq \theta \Leftrightarrow t_p \simeq t_{p'}$) effectively quantizes the correlation surfaces $\mathcal{C}_p$ to binary. In most cases, this is a good approximation due to the exponential in the similarity measure (eq. (1)), analogously to maximum approximation in exponential models [2].

Fig. 4: **Patch prototype codebooks.** From top to bottom: three images; the first prototypes from the generic DCT codebook; some prototypes out of 2000 in a codebook created from 20 images; some prototypes out of 100 in each image-specific codebook.

Given a codebook Θ of prototypes θ we compute which pixels p are assigned to which patch prototype θ, leading to $k = |\Theta|$ binary correlation surfaces $\mathcal{C}_\theta$. Since each pixel is assigned to exactly one θ, the $\mathcal{C}_\theta$'s are disjoint and can be stored in a single $H \times W$ *prototype assignment map* $\mathcal{M}_I$ (fig. 3). The GSS tensor $\tilde{\mathcal{S}}_I$ is derived from $\mathcal{M}_I$ using simple lookups as

$$\tilde{\mathcal{S}}_I(p, p') = \delta(\mathcal{M}_I(p) = \mathcal{M}_I(p')) \tag{4}$$

where $\delta(x)$ is the Kronecker delta: $\delta(x) = 1$ iff x is true, $\delta(x) = 0$ otherwise.

$\mathcal{M}_I$ has several advantages over the direct $\mathcal{S}_I$: (i) it can be stored in a fraction of the memory (HW instead of H^2W^2); (ii) it can be computed in a fraction of the time (see below); (iii) it leads to better classification accuracy (sec. 6).

When computing $\mathcal{M}_I$, accuracy is traded for speed. With a larger codebook Θ computation is slower but the resulting prototype assignment map is more finely quantized. Finally, note that although our prototype assignment maps $\mathcal{M}_I$ are similar to texton-maps used in patch-based object classification methods [24], the resulting descriptors differ profoundly as they encode similarities between regions within an image rather than their actual appearance.

Efficient Computation of $\mathcal{M}_I$. Given an image I and a patch prototype codebook Θ with $k = |\Theta|$ prototypes θ, we need to correlate each θ with the entire image resulting in k correlation surfaces $\mathcal{C}_\theta$ and requiring $kHWw^2$ operations, if done directly. $\mathcal{M}_I$ is obtained from the $\mathcal{C}_\theta$'s as

$$\mathcal{M}_I(p) = \arg\max_{\theta \in \Theta} \{\mathcal{C}_\theta(p)\} \tag{5}$$

Compared to direct GSS (sec. 3.1), the speedup brought by our method is factor HW/k. Additionally, FFT could also be used here leading to further speedups as for direct GSS.

3.3. Patch Prototype Codebooks Θ

We propose three variants (fig. 4) of patch prototype codebooks suited to compute $\mathcal{M}_I$:

Generic prototypes Θ_{DCT}. The discrete cosine transform (DCT) is a standard technique in signal processing aiming at representing signals as a linear combinations of basis func-

tions [1]. By discarding some of the coefficients, signals can be compressed. The DCT can be computed very efficiently.

Here, we choose Θ to be a subset of the basis functions of a DCT for $w \times w$ prototypes (i.e. we discard the DC component and high frequencies). We apply the DCT for each color-channel independently and determine the prototype θ_p^{DCT} with the highest response for each pixel p. $\mathcal{M}_I(p)$ is the composition of the prototypes of the three channels. The prototypes $\theta \in \Theta_{\text{DCT}}$ can immediately be applied to any image. With specialized DCT implementations, $\mathcal{M}_I$ can be computed in $3HWw^2 \log w^2$ operations: for each pixel p and color channel, the DCT of a patch is computed. Fig. 4 shows the first few prototypes in a DCT codebook.

Dataset-specific prototypes Θ_{DB}. Here we learn patch prototypes specific to an image dataset. As in the creation of codebooks for object classification [16, 21], we create dataset-specific prototypes $\theta \in \Theta_{\text{DB}}$ by randomly sampling patches from the images and then quantizing them into k prototypes θ with k-means. Θ_{DB} is computed once beforehand and does not have to be taken into account to compute $\mathcal{M}_I$. Here, the choice of k trades representation accuracy for speed. For large variations in the image set, a large k is necessary for a good representation. Fig. 4 show several prototypes from a dataset-specific codebook. They exhibit a wide variety of colors and patterns.

Image-specific prototypes Θ_I. The GSS tensors $\tilde{\mathcal{S}}_I$ do not contain the codebooks Θ. Following the philosophy of self-similarity, and as opposed to the BOW framework, $\tilde{\mathcal{S}}_I$ only encodes whether two pixels in an image are assigned to the same prototype θ, but not to which prototype. As a consequence, we can use a different codebook for each image, enabling image-specific patch prototypes $\theta \in \Theta_I$.

To create Θ_I specific to image I, we proceed analogously as when creating Θ_{DB}. Instead of jointly clustering randomly sampled patches from a set of images, we only sample patches from I and cluster them independently of other images. Since variation within a single image is smaller than within a set of images, $|\Theta_I| << |\Theta_{\text{DB}}|$ allows for comparable representation quality. To create a prototype assignment map $\mathcal{M}_I$ now we have to perform two steps:

(1) Creation of the codebook Θ_I: cluster n randomly sampled patches from I to form k prototypes. This requires k distance computations per patch in each of the L k-means iterations, giving a total of w^2nkL operations.

(2) Compute patch-assignment maps $\mathcal{M}_I$: assign the most similar prototype $\theta \in \theta_I$ to each pixel (eq. (5)). As $k = |\Theta_I| < |\Theta_{\text{DB}}|$, the number of convolutions is lower than when using Θ_{DB}.

Fig. 4-bottom shows several prototypes from Θ_I's corresponding to the images in fig. 4-top. Compared to the dataset-specific prototypes in the same figure, Θ_I are clearly better suited to represent their respective images.

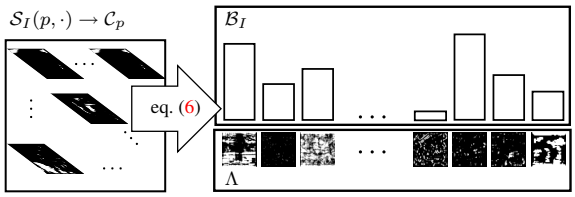

Fig. 5: **Bag of Correlation Surfaces.** Correlation surfaces $\mathcal{C}_p = \mathcal{S}_I(p,\cdot)$ are vector-quantized according to the correlation surface codebook Λ and counted (eq. (6)) to obtain $\mathcal{B}_I$.

4. Descriptors based on global self-similarity

While the previous section describes methods to efficiently and compactly capture the GSS tensor $\mathcal{S}_I$, it does not provide means of using these for recognizing and detecting objects. $\mathcal{S}_I$ is a 4D tensor depending on the size of the image I. It cannot be used directly as most machine learning frameworks require fixed-size descriptors.

In the following we propose two fixed-size global image descriptors based on the GSS tensor. They can be obtained either from the direct GSS tensor $\mathcal{S}_I$ or from the approximate $\tilde{\mathcal{S}}_I$. To ease notation we always write $\mathcal{S}_I$ below.

4.1. Bag of Correlation Surfaces $\mathcal{B}_I$ (BOCS)

A *bag-of-correlation-surfaces* (BOCS) descriptor is based on the same principles as BOW descriptors: we vector quantize the correlation surfaces $\mathcal{C}_p$ from an image I and represent them as a histogram.

Let $\mathcal{S}_I(p,\cdot) = \mathcal{C}_p$ be the correlation surface for pixel p. We learn a correlation surface codebook Λ from a set of images by scaling their correlation surfaces $\mathcal{C}_p$ to a common size ($m \times m$) and clustering to k prototypes $\lambda \in \Lambda$.

Given Λ, the BOCS $\mathcal{B}_I$ is derived from $\mathcal{S}_I$ by (1) assigning each correlation surface to its most similar prototype λ, and (2) counting how many surfaces are assigned to each prototype (fig. 5):

$$\mathcal{B}_I(\lambda) = \sum_{p \in I} \delta\left(\lambda = \arg\min_{\lambda' \in \Lambda} \{\text{SSD}(\lambda', \mathcal{S}_I(p,\cdot))\}\right) \quad (6)$$

In a BOCS, the ordering of the $\mathcal{C}_p$'s is lost. However, the individual $\mathcal{C}_p$'s still convey spatial structure, as each is the correlation surface for a pixel.

The computational complexity to obtain $\mathcal{B}_I$ from $\mathcal{S}_I$ is $HW|\Lambda|m^2$: each of the $H \times W$ correlation surfaces $\mathcal{C}_p$ is compared to each of the prototypes $\lambda \in \Lambda$.

4.2. Self-similarity Hypercubes $\mathcal{H}_I$ (SSH)

Self-similarity hypercubes (SSH) aim at preserving the entire spatial structure of the GSS tensor $\mathcal{S}_I$, while (i) making it easier to handle by reducing its size to (ii) a fixed size that can be processed in standard machine-learning frameworks, and (iii) enabling the efficient extraction of SSH for many windows in an image (sec. 5).

An SSH $\mathcal{H}_I$ is a 4D tensor of size $(D_1 \times D_2 \times D_1 \times D_2)$ and can be considered as $\mathcal{S}_I$ scaled down to a fixed size. To obtain $\mathcal{H}_I$, a regular $D_1 \times D_2$ grid is projected onto the image. $\mathcal{H}_I(J, J')$ describes how similar two grid cells

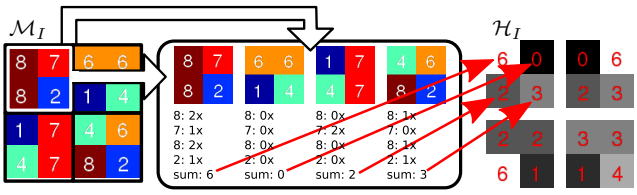

Fig. 6: **Self-similarity hypercubes.** Constructing a 2^4 SSH $\mathcal{H}_I$ from a 4x4 prototype assignment map $\mathcal{M}_I$ according to eq. (8).

$J = (i, j)$ and $J' = (k, l)$ are.

$$\mathcal{H}_I(J, J') = \sum_{p \in J} \sum_{p' \in J'} \mathcal{S}_I(p, p') \tag{7}$$

where the summations run over all pixels p inside J, and p' inside J'.

If $\mathcal{S}_I$ is represented by a prototype assignment map $\mathcal{M}_I$, we can compute the SSH descriptor directly from it, without forming $\mathcal{S}_I$ and therefore saving memory (fig. 6). In this case, we count how many pixels in J were assigned to the same template as pixels in J':

$$\mathcal{H}_I(J, J') = \sum_{p \in J} \sum_{p' \in J'} \delta(\mathcal{M}_I(p) = \mathcal{M}_I(p')) \tag{8}$$

Computing SSH require H^2W^2 operations in both cases.

5. Efficient Extraction of SSHs for Detection

Many recent object localization schemes are based on sliding-windows: localization is reduced to subsequently evaluating a classifier for many (possibly all) windows in the image. The score of the classifier indicates whether the object is present in the window. Since even moderately sized images already contain an enormous number of windows, only very efficient methods can be applied in this framework.

Lampert *et al.* [17] presented a method based on branch-and-bound, called efficient-subwindow search (ESS), which enables to efficiently search all possible windows in an image I, for certain classifiers. In the following we show how to efficiently compute SSH descriptors $\mathcal{H}_{I'}$ for an arbitrary window $I' \in I$ (sec. 5.1) and how this can also be used for branch-and-bound search, analogously to ESS (sec. 5.2).

5.1. Efficient Extraction of SSHs from subwindows

For sliding-window detection we need to efficiently extract SSHs $\mathcal{H}_{I'}$ for arbitrary windows I' in an image I. A simple way to extract $\mathcal{H}_{I'}$ for one window I' from the GSS tensor $\mathcal{S}_I$ is to crop it to cover only I' and then compute $\mathcal{H}_{I'}$ from it using (7), which requires $(H' \cdot W')^2$ operations (where $H' \times W'$ is the size of I').

When extracting $\mathcal{H}_{I'}$ for many windows, a significant speedup can be achieved using an integral self-similarity $\mathcal{S}_I^\Sigma$, built analogously to integral images [7] in 4D:

$$\mathcal{S}_I^\Sigma(p_{xy}, p'_{x'y'}) = \sum_{i=1}^{x} \sum_{j=1}^{y} \sum_{k=1}^{x'} \sum_{l=1}^{y'} \mathcal{S}_I(p_{ij}, p_{kl}) \tag{9}$$

This structure can be built linearly ($4H^2W^2$) in the size $(H \times W)^2$ of $\mathcal{S}_I$. Then, computing $\mathcal{H}_{I'}$ for an arbitrary window I' takes $D_1^2 \cdot D_2^2 \cdot 16$ lookups. For each of the $D_1^2 \cdot D_2^2$ entries of $\mathcal{H}_{I'}$ corresponding to grid cells J and J', 16 lookups are necessary to compute the sum over the corresponding part of $\mathcal{S}_I$. Thus, the cost to compute $\mathcal{H}_{I'}$ for a window I' now is $16(D_1 \cdot D_2)^2$, compared to $(H' \cdot W')^2$ before. As typically D_1, D_2 are around 10 and H', W' around 100, the cost is reduced by about 100 times.

For classification we use a linear SVM. The corresponding score function is

$$f(\mathcal{H}_{I'}) = \beta + \sum_{J,J'} \theta(J, J') \mathcal{H}_{I'}(J, J') \tag{10}$$

where β is the bias and θ is the SVM hyperplane (formatted to the dimensionality of $\mathcal{H}_{I'}$). Computing $f(\mathcal{H}_{I'})$ requires $D_1^2 \cdot D_2^2$ operations.

5.2. Efficient Subwindow Search on SSHs

To apply branch-and-bound techniques such as ESS [17] we need to define an upper-bound on the score of a (contiguous) set of windows $\mathcal{R}$. This upper-bound must deliver the score of I' if $\mathcal{R}$ only contains a single window I'. Rewriting eq. (10) as

$$f(\mathcal{H}_{I'}) = \beta + \sum_{(J,J'):\theta(J,J')>0} \theta(J, J') \mathcal{H}_{I'}(J, J') + \sum_{(J,J'):\theta(J,J')<0} \theta(J, J') \mathcal{H}_{I'}(J, J')$$

enables formulating an upper-bound $U(f, \mathcal{R}) \geq \max_{I' \in \mathcal{R}} f(\mathcal{H}_{I'})$:

$$U(f, \mathcal{R}) = \beta + \sum_{(J,J'):\theta(J,J')>0} \theta(J, J') \mathcal{H}^+_{\mathcal{R}}(J, J') + \sum_{(J,J'):\theta(J,J')<0} \theta(J, J') \mathcal{H}^-_{\mathcal{R}}(J, J')$$

where $\mathcal{H}^+_{\mathcal{R}}(J, J')$ is an upper-bound on $\mathcal{H}_{I'}(J, J')$ for all $I' \in \mathcal{R}$:

$$\mathcal{H}^+_{\mathcal{R}}(J, J') \geq \max_{I' \in \mathcal{R}} \mathcal{H}_{I'}(J, J')$$

and analogously for the lower-bound $\mathcal{H}^-_{\mathcal{R}}(J, J')$:

$$\mathcal{H}^-_{\mathcal{R}}(J, J') \leq \min_{I' \in \mathcal{R}} \mathcal{H}_{I'}(J, J')$$

We obtain $\mathcal{H}^+_{\mathcal{R}}(J, J')$ and $\mathcal{H}^-_{\mathcal{R}}(J, J')$ from $\mathcal{S}_I^\Sigma$. Analog to [17], $\mathcal{H}^+_{\mathcal{R}}(J, J')$ is the sum over the part of $\mathcal{S}_I$ corresponding to the union of all possible grid cells J, J' in $\mathcal{R}$. $\mathcal{H}^-_{\mathcal{R}}(J, J')$ is defined analogously, but over the intersection of those cells.

6. Experimental Evaluation: Classification

We evaluate the classification performance on object subimages cropped out of the Pascal VOC 2007 dataset [9] (Pascal07 from now on) according to their annotation bounding-box. We use all objects that are not marked as truncated or difficult. For training we also discard objects with unspecified viewpoint resulting in a total of 9608 object subimages (training on 1688+1653 from the train+val Pascal07 sets, testing on 6267 of test).

In secs 6.1 to 6.4 we experiment on a subset consisting of 629 subimages from six classes (airplane, boat, bus, motorbike, sheep, train), each restricted to one viewpoint. We use this subset to (a) set the parameters of all methods; (b)

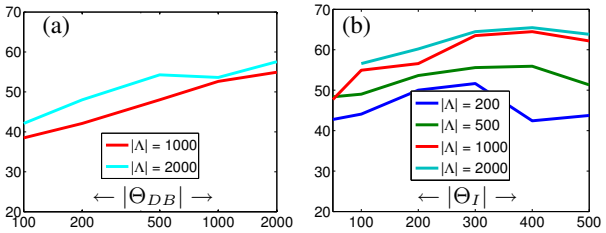

Fig. 7: Classification accuracy [%] *vs.* size of the patch prototype codebook (a) Θ_{DB} (b) Θ_I using BOCS with correlation surface codebooks Λ of different sizes and linear SVMs for classification.

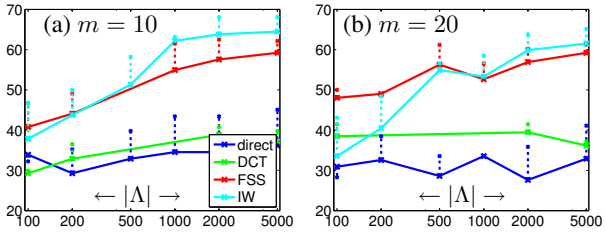

Fig. 8: Classification accuracy [%] of BOCS descriptors *vs.* size of the correlation surface codebook Λ using the different types of GSS tensors. Size m of the correlation surfaces (a) 10 (b) 20. Dashed vertical lines show the improvements using IK SVMs.

evaluate BOCS and SSH alone (secs. 6.1, 6.2); (c) compare them to LSS and direct GSS (sec. 6.3); (d) combine the SS descriptors with conventional descriptors (sec. 6.4). In sec. 6.5 we confirm our findings by evaluating the best working setups on the full Pascal07 set (9608 subimages).

6.1. Bag of correlation surfaces $\mathcal{B}_I$

A BOCS can be computed from a direct GSS tensor $\mathcal{S}_I$ or from an approximate GSS tensor $\tilde{\mathcal{S}}_I$ (using any of $\Theta_{\text{DCT}}, \Theta_{\text{DB}}, \Theta_I$). First we determine the parameters to extract GSS tensors, then those of BOCS.

Direct GSS tensor $\mathcal{S}_I$. To obtain $\mathcal{S}_I$, the patch similarity measure and the size of the patches w have to be chosen. We compared sum of squared distances (SSD) and normalized cross-correlation (NCC) and found SSD to perform better. It seems that the additional brightness invariance of NCC is not necessary within *one* image and only hurts discriminative power. We found the descriptors to be quite robust w.r.t. the patch-size w. In all experiment we set w as 1/20 of the image size.

Efficient GSS tensor $\tilde{\mathcal{S}}_I$. For each codebook type, the patch-size w and the number $|\Theta|$ of prototypes $\theta \in \Theta$ have to be determined:

Generic codebooks Θ_{DCT}. We evaluated $|\Theta_{\text{DCT}}| = 5^3$ and 10^3 (5 and 10 prototypes per color channel) and found the larger codebooks to perform slightly better.

Dataset-specific codebooks Θ_{DB}. Fig. 7(a) shows classification accuracy *vs.* $|\Theta_{\text{DB}}|$ for two different BOCS. The accuracy saturates at about $|\Theta_{\text{DB}}| = 2000$ regardless of the size of Λ. We expect larger Θ to be preferable for larger datasets (sec. 3.3).

Image-specific codebooks Θ_I. Fig. 7(b) shows classification accuracy *vs.* $|\Theta_I|$ for different BOCS. The best results are obtained at $|\Theta_I| \approx 400$ regardless of the size of Λ

Parameters of BOCS ($|\Lambda|$, m). Fig. 8 shows the impact of the size of the correlation surface codebooks Λ on the classification accuracy for (a) $m = 10$ and (b) $m = 20$ for the different GSS tensors. The approximate GSS tensors $\tilde{\mathcal{S}}_I$ outperform the direct $\mathcal{S}_I$. $\tilde{\mathcal{S}}_I$ obtained with image-specific patch prototypes Θ_I performs best, followed by dataset-specific Θ_{DB}. Small correlation surfaces ($m = 10$, fig. 8(a)) perform better than large ($m = 20$, fig. 8(b)). Using an intersection kernel (IK) SVM instead of a linear one, the accuracy is consistently improved (dashed vertical lines).

Tab. 2: Classification accuracy [%] for SSH with $D_1 = D_2 = 10$.

	SVM	
GSS tensor	linear	IK
direct	45.7	36.5
Θ_{DB}	49.3	60.9
θ_I^{500}	52.6	**67.8**

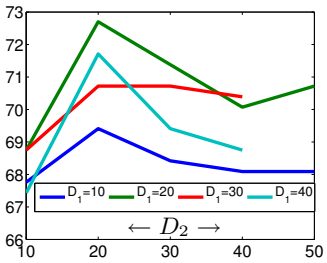

Fig. 9: Classification accuracy [%] *vs.* D_2 for SSH with different D_1 (using IK SVMs).

6.2. Self-similarity Hypercubes $\mathcal{H}_I$

SSHs $\mathcal{M}_I$ can also be obtained from the four variants of the GSS tensor $\mathcal{S}_I$ and $\tilde{\mathcal{S}}_I$. For the GSS tensor we use the parameters that worked best for BOCS, and investigate here the impact of the sizes D_1, D_2 of SSH on classification accuracy. Tab. 2 compares the performance for SSHs ($D_1 = D_2 = 10$) for the different types of GSS tensors. Again, the image-specific codebooks perform best, followed by the dataset-specific codebooks. Also here, using an IK SVM brings a significant boost of accuracy. Fig. 9 shows the impact of the size (D_1, D_2) of SSH on the classification accuracy (using Θ_I with $|\Theta_I| = 400$). Clearly $D_1 = D_2 = 20$ performs best, leading to a $20^4 = 160\,000$ dimensional descriptor.

Normalization of SSH. The SSH $\mathcal{H}_I(J, \cdot)$ roughly corresponds to downscaled correlation surfaces and can be interpreted as an (unnormalized) histograms of how many pixels in the respective other grid cells J' are similar (*i.e.* assigned to the same patch prototype) as pixels in a cell J. Applying histogram normalization to each of these $D_1 \times D_2$ correlation surfaces increases classification accuracy from 72.7% to 76.0%. (using Θ_I, $|\Theta_I| = 500$ and $D_1 = D_2 = 20$).

6.3. Comparison to LSS and GSS

The column "alone" in tab. 3 shows classification accuracy for the different self-similarity descriptors used alone. SSH performs best, BOCS comes second. Both clearly outperform BOLSS. For both SSH and BOCS, image-wise codebooks perform best, database-wise second best. For SSH it is important to have sufficiently large descriptors ($D_1 = D_2 = 20$). Normalizing SSH visibly raises classification accuracy. These results confirm that self-similarity is

Tab. 3: Classification accuracy [%] on the Pascal07 subset using conventional descriptors, LSS, GSS, and combinations of these.

		alone		BOW		HOG		GIST	
GSS descriptor		lin.	IK	lin.	IK	lin.	IK	lin.	IK
none		–	–	66.8	67.4	70.4	74.0	78.6	81.9
BOLSS		61.2	63.4	73.4	71.7	70.4	74.0	79.6	**84.5**
BOCS with $\|\Lambda\| = 2000, m = 10$									
direct $\mathcal{S}_I$		34.5	43.4	61.5	68.1	69.4	75.0	71.7	83.2
$\tilde{\mathcal{S}}_I$ with Θ_{DB}		57.6	62.5	68.8	72.4	70.4	75.0	79.0	84.5
$\tilde{\mathcal{S}}_I$ with Θ_I		**64.5**	**67.1**	**74.3**	**76.0**	**73.7**	**77.0**	**75.7**	**84.5**
SSH	D_1, D_2								
$\tilde{\mathcal{S}}_I$ with Θ_{DB}	10	49.3	60.9	67.8	73.7	70.4	76.6	77.0	82.9
	20	52.0	60.2	68.1	74.0	70.4	74.0	78.6	82.9
$\tilde{\mathcal{S}}_I$ with Θ_I	10	52.6	67.8	70.1	74.0	70.4	77.0	78.3	82.9
	20	58.9	72.7	69.7	77.0	70.4	77.3	79.3	84.2
+norm.	20	**79.0**	**76.0**	**78.0**	**81.3**	**79.9**	**79.9**	**84.2**	**85.5**

more powerful as a *global* rather than as a *local* descriptor.

6.4. Combination with conventional descriptors

To investigate the GSS descriptors further, we combine them with several conventional descriptors:

GIST [22] is based on localized histograms of gradient orientations. It captures the rough spatial arrangement of image structures, and has been shown to work well for describing the overall appearance of an image.

Bag of visual words (BOW) are standard for many recognition tasks [3, 13, 14, 16–18, 25, 28]. We use SURF descriptors [3, 17] and quantize them into 2000 words with k-means. A window is described by a BOW of SURF.

Histograms of oriented gradients (HOG) also are an established descriptor for object class recognition [8, 10].

Tab. 3 shows classification accuracy for the conventional descriptors alone (row "none") and combined with LSS, BOCS, and SSH. For combination we train a separate SVM for each descriptor and combine their scores in a weighted sum. The weight is determined on the validation set.

While all self-similarity descriptors raise the accuracy in combination with conventional cues, SSH achieves the largest improvement. These results demonstrate that GSS descriptors are truly complementary to conventional descriptors.

6.5. Full Pascal07 dataset

To confirm our findings, we evaluate the best GSS descriptor (SSH with $D_1 = D_2 = 20$, Θ_I, $|\Theta_I| = 500$, normalized) and BOLSS on the full Pascal07 set. We train a separate classifier per viewpoint (left, right, front, back) For testing, we use all objects in the test set (including those with unspecified viewpoint) and measure average classification accuracy over the 20 classes (*i.e.* no need to predict viewpoint). As tab. 4 shows, SSH clearly outperforms BOLSS. In combination with GIST, both BOLSS and SSH improve over GIST alone, but SSH moderately outperforms BOLSS. These results reinforce the findings from before: (a) GSS performs better than LSS, (b) self-similarity is complementary to conventional descriptors.

Tab. 4: Classification accuracy [%] on the full Pascal07 set using GSS, LSS and combinations of with GIST.

	alone		GIST	
descriptor	linear	IKSVM	linear	IKSVM
BOLSS	25.0	31.9	52.9	57.5
SSH	44.0	45.7	**55.1**	**59.4**

6.6. Runtimes for computing descriptors

Computing the GSS tensor of a 200x200 pixels image using our efficient method of sec. 3.2 with an image-specific codebook of 200 patch prototypes takes 81s. Instead, extracting the GSS tensor directly would take 5512s (sec. 3.1). Deriving the BOCS and SSH descriptors from the GSS tensor takes a negligible time. So, our methods bring about a 70-fold speedup. For reference, computing a GIST descriptor takes 0.4s and BOLSS 0.7s.

7. Experimental Evaluation: Detection

For detection, we compare the performance of BOLSS and SSH on the ETHZ Shape Classes dataset [12] (ETHZ-SC from now on). This dataset contains a total of 255 images from five classes (apple logos, bottles, giraffes, mugs, and swans). We follow the setup of [11, 20] and train one detector per class using the first half of the positive training images and the same number of negative training images (in equal shares taken from all other classes). We consider a detection correct if the detection window intersection-over-union with the ground-truth window is > 0.5 (Pascal criterion).

Training. We train detectors for BOLSS and SSH using a training algorithm inspired by [10]:

(1) Initialization: Extract descriptors from annotated bounding-boxes of positive training images and five random bounding-boxes per negative training image to train an initial linear SVM. Also compute the average aspect-ratio of the ground-truth bounding-boxes.

(2) Apply the detector in sliding-window mode (step-size 16 with fixed aspect ratio) on each training image and then run greedy non-maxima suppression [10]. All wrong detections (according to the Pascal criterion) from all images are collected as *hard negative samples*.

(3) Retrain the SVM by adding those hard negative samples.

(4) Repeat steps (2)-(3) 40 times (or until convergence).

Detection. As in [11, 20], the test set for a class consists of *all* ETHZ-SC images not used to train that class (including images of other classes). To detect multiple objects per image, we use a sliding-window detector, as the branch-and-bound framework does not easily support this [17, 19][2].

Fig. 10 and tab. 5 show detection performance for SSH ($|\Theta_I| = 500, D_1 = D_2 = 20$) and BOLSS. Performance is measured in terms of detection rate and the average rate of false-positives per image (FPPI). Overall, SSH greatly

[2]We evaluated ESS and found it to be about 30x faster than sliding windows when searching the same space of windows.

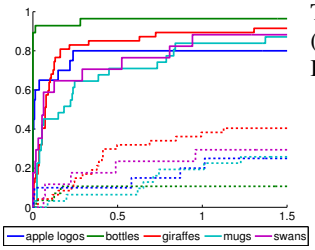

Fig. 10: Detection rate *vs*. FPPI on ETHZ-SC for SSH (solid) and BOLSS (dashed).

Tab. 5: Det. rate at (a) FPPI=0.3 (b) FPPI=0.4 on ETHZ-SC using BOLSS and SSH.

	BOLSS		SSH	
category	(a)	(b)	(a)	(b)
apple logos	10.0	10.0	**80.0**	**80.0**
bottles	10.7	10.7	**96.4**	**96.4**
giraffes	17.0	23.4	**83.0**	**85.1**
mugs	6.5	6.5	**64.5**	**67.7**
swans	17.6	17.6	**70.6**	**70.6**
average	12.4	13.6	**78.9**	**80.0**

outperforms BOLSS. The large gap in detection performance can be explained by three reasons: (i) Linear SVMs on BOW features are known to localize the object in images containing it [17], but are not powerful enough to discriminate false positives from images not containing it. Therefore, more powerful discriminative models such as IK SVMs are necessary [27]. (ii) Spatial arrangements are important to recognize classes from the ETHZ-SC dataset, but BOLSS largely discards them. (iii) The dimensionality of the descriptor (160 000 for SSH *vs*. 300 for BOLSS) plays an important role in training linear SVMs. Since here the number of negative training samples is huge, this gives an additional advantage to the SSH descriptor.

Note how there are recent methods performing even better than SSH on this dataset, *e.g*. [20] on average obtains a detection rate of 91.9%/93.2% at FPPI 0.3/0.4 using a discriminative max-margin Hough transformation. However, our aim is not to outperform the state-of-the-art but to demonstrate that (i) it is possible to use GSS for detection efficiently and (ii) GSS performs better than LSS.

Runtimes for detection. Computing the GSS tensor for an entire image takes about 80s. It is done only once and reused for all classes. After this, detecting one class using SSH with $D_1 = D_2 = 20$ takes approximately 4min. With $D_1 = D_2 = 10$, this speeds up to 30s with only a minor loss of performance (avg. 78.0/79.4 instead of 78.9/80.0). For comparison, computing direct GSS separately for the same number of windows in an image (about 25000) would take about 4 years. So, our methods made object detection with GSS descriptors possible.

8. Conclusion

We explored GSS, discussed its advantages over LSS, propose descriptors based on it, and shown how to use them for classification and detection. In detail, (a) we proposed an efficient method to extract GSS from images; (b) we developed efficient image descriptors based on GSS. These capture self-similarities and their spatial arrangement within an entire image, as opposed to previous local self-similarity descriptors; (c) we have shown how to use these descriptors efficiently for detection in the sliding-window framework and in the branch-and-bound framework; (d) we experimentally demonstrated on Pascal VOC 2007 and on ETHZ Shape Classes dataset that our GSS descriptors outperform LSS for both classification and detection, and that they are complementary to conventional descriptors such as BOW, HOG, and GIST.

Thanks to Christoph Lampert for helpful hints on ESS and to SNSF for supporting this research.

References

[1] N. Ahmed, T. Natarajan, and K. Rao. Discrete cosine transform. *IEEE Trans. Computers*, pages 90–93, 1974.
[2] O. Barndorff-Nielsen and P. Jupp. Approximating exponential models. *Annals of the Inst. of Stat. Math.*, 41(2):247–267, 1988.
[3] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. SURF: Speeded up robust features. *CVIU*, 110(3):346–359, 2008.
[4] O. Boiman and M. Irani. Detecting irregularities in images and in video. *IJCV*, 74(1):17–31, 2007.
[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.
[6] K. Chatfield, J. Philbin, and A. Zisserman. Efficient retrieval of deformable shape classes using local self-similarities. In *NORDIA Workshop at ICCV 2009*, 2009.
[7] F. Crow. Summed-area tables for texture mapping. In *SIGGRAPH*, 1984.
[8] N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *CVPR*, 2005.
[9] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007.
[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2009. in press.
[11] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1):36–51, 2008.
[12] V. Ferrari, T. Tuytelaars, and L. van Gool. Object detection by contour segment networks. In *ECCV*, 2006.
[13] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009.
[14] E. Hörster and R. Lienhart. Deep networks for image retrieval on large-scale databases. In *ACM Multimedia*, 2008.
[15] I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. Cross-view action recognition from temporal self-similarities. In *ECCV*, 2008.
[16] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.
[17] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 2009. in press.
[18] C. H. Lampert, H. Nickisch, S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
[19] A. Lehmann, B. Leibe, and L. van Gool. Feature-centric efficient subwindow search. In *ICCV*, 2009.
[20] S. Maji and J. Malik. Object detection using a max-margin hough tranform. In *CVPR*, 2009.
[21] F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests. In *NIPS*, 2006.
[22] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.
[23] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007.
[24] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context. *IJCV*, 81(1):2–23, 2009.
[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
[26] M. Stark, M. Goesele, and B. Schiele. A shape-based object class model for knowledge transfer. In *ICCV*, 2009.
[27] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.
[28] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73(2):213–238, 2007.