BIMP: A Real-Time Biological Model of Multi-Scale Keypoint Detection in V1

Kasim Terzić[a,*], J.M.F. Rodrigues[a], J.M.H. du Buf[a]

[a]*Vision Lab (LARSyS), FCT, University of the Algarve, Gambelas Campus, 8000 Faro, Portugal*

Abstract

We present an improved, biologically inspired and multiscale keypoint operator. Models of single- and double-stopped hypercomplex cells in area V1 of the mammalian visual cortex are used to detect stable points of high complexity at multiple scales. Keypoints represent line and edge crossings, junctions and terminations at fine scales, and blobs at coarse scales. They are detected by applying first and second derivatives to responses of complex cells in combination with two inhibition schemes to suppress responses along lines and edges. A number of optimisations make our new algorithm much faster than previous biologically-inspired models, achieving real-time performance on modern GPUs and competitive speeds on CPUs. In this paper we show that the keypoints exhibit state-of-the-art repeatability in standardised benchmarks, often yielding best-in-class performance. This makes them interesting both in biological models and as a useful detector in practice. We also show that keypoints can be used as a data selection step, significantly reducing the complexity in state-of-the-art object categorisation.

Keywords: Computer Vision, Gabor Filter, V1, Keypoint, Categorization

1. Introduction

Accurate detection of stable interest points is a central task in many object detection and recognition approaches, and an important part of early human visual processing. While many computer vision algorithms have been motivated by insights gained from biological vision, including image processing with Gabor wavelets and current work on deep hierarchies, existing biologically plausible keypoint detection algorithms are limited to a single scale [1], or are computationally too complex to run in real time on a CPU [2]. Furthermore, no comparative benchmarking of biological keypoint models is available in the literature. In this paper, we present an optimised keypoint extraction algorithm based on existing models of end-stopped cells in the mammalian striate cortex and evaluate its performance.

Early processing in the area V1 of the mammalian visual cortex has been extensively studied in the literature. The image signal from retina enters V1 via the Lateral Geniculate Nucleus (LGN) and is then processed by layers of so-called simple cells, complex cells and hypercomplex (or end-stopped) cells. Simple cells, often modelled using oriented Gabor filters, respond to lines and edges. Complex cells provide more position-invariant responses to both. End-stopped cells respond to line terminations (single-stopped cells), as well as to corners and blobs (double-stopped cells). Earlier work has shown that models of these cells can act as a general-purpose keypoint detector, but they require convolutions with large filter kernels, making them prohibitively slow for most applications in computer vision and cognitive robotics.

The main contributions of this paper are (i) a new and optimised algorithm which is fast enough to run on a CPU and which runs in real time on GPUs due to its parallel nature; and (ii) extensive benchmarking of the algorithm, showing state-of-the-art performance compared to best available algorithms, and setting several records in terms of repeatability and precision. To the best of our knowledge, this is the first extensive comparison of a biological model with the state of the art in computer vision. We have released the CPU and GPU implementations of our detector as Free Software, so others can use them for real-world applications.

*Corresponding author at: CINTAL, University of the Algarve, Gambelas Campus, 8000 Faro, Portugal. Phone: +351962846514

Email addresses: kterzic@ualg.pt (Kasim Terzić), jrodrig@ualg.pt (J.M.F. Rodrigues), dubuf@ualg.pt (J.M.H. du Buf)

URL: w3.ualg.pt/~kterzic (Kasim Terzić), w3.ualg.pt/~jrodrig (J.M.F. Rodrigues), w3.ualg.pt/~dubuf (J.M.H. du Buf)

1.1. Related Work

There exist a number of approaches for detecting interest points in images which are stable under a wide range of transformations, including scaling, translation and rotation. Early work on corner detection used structure tensors [3, 4], which have recently been extended to provide scale invariance [5]. Other computational approaches include Difference of Gaussians [6] and the Determinant of Hessian [7]. Meaningful blobs have been detected using region-based methods [8, 9] and by other affine-invariant region detectors. Additional interest point and region detectors are described in [10].

Biologically inspired approaches to keypoint detection attempt to model early processing stages of the mammalian visual cortex (area V1), consisting of layers of cells. So-called simple cells are modelled using a bank of bandpass filters, usually Gabor wavelets. Beyond this, responses of complex and end-stopped cells are often represented implicitly, as a spatial combination of simple-cell responses [11, 12]. However, there have also been efforts to model complex and end-stopped cells directly, in order to obtain an explicit representation of keypoints corresponding to strong activations of complex cells [1, 13], but there is no comparative benchmarking of such models against state-of-the-art keypoint detection in computer vision.

Our model follows the early single-scale model of Heitger *et al.* [14], which consists of single and double end-stopped cells and two inhibition schemes. Several extensions have been proposed [2, 15], capable of detecting keypoints at multiple scales and adapting NCRF inhibition [16] to keypoints. Our new model is inspired on the one by Rodrigues and du Buf [2], which is too slow for practical use, taking hours on large images. In this paper, we expand on our earlier work presented in [17].

We completely reformulate and re-implement the algorithm. Instead of modelling individual cells, as in [2], we model populations of cells as activation maps, obtained by parallel filtering operations which can be efficiently evaluated on modern CPUs and GPUs. We use a Gaussian pyramid combined with sub-pixel localisation and show that this step significantly improves repeatability compared to [2]. We also introduce a scale selection method which reduces the redundancy of detected keypoints. These changes result in a significant improvement in both speed and accuracy, such that biologically-inspired keypoints are now suitable for real-time applications. We benchmark our improved approach on standard datasets, showing that it improves on both [2] and the state of the art in computer vision.

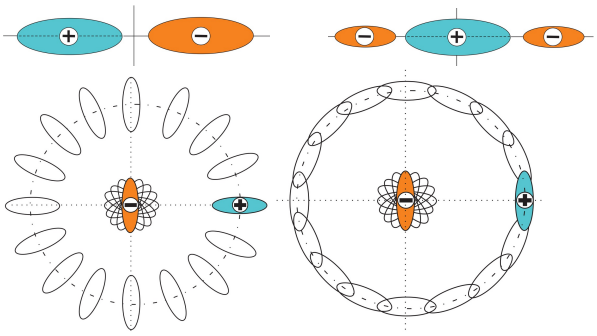

Figure 1: End-stopped cells and inhibition schemes. Top row shows the receptive fields of single (left) and double (right) end-stopped cells. Bottom row shows the two inhibition schemes: radial (left) and tangential (right). The tangential component is in the same orientation, the radial component is in the orthogonal orientation.

2. An Optimised Computational Model of V1

Our approach is based on area V1 of the mammalian visual cortex, with layers of specialised cells responding to increasingly complex patterns. At the highest level, responses of single and double end-stopped cells are used to detect stable events (corners, blobs and terminators) at all scales.

Basically, the keypoint model works as follows. Simple and complex cells respond to lines and edges. Assume that there is a corner formed by a vertical and a horizontal edge, and the goal is to detect only the corner position. Complex cells tuned to the edge orientations will produce a maximum response at the edge positions. Cells tuned to other orientations will also respond at the edges, but less. Now, single and double end-stopped cells are modelled by first and second derivatives of the responses of complex cells, in the same orientations as those of the complex cells; see Fig. 1 (top).

This implies that first derivatives (single-stopped cells) will produce responses astride the edges: on both sides but zero in the middle. Second derivatives (double-stopped cells) will also produce responses, but these are maximum at the edge positions and they decrease on both sides. Hence, when all responses are summed over all orientations, there will be a peak at the corner, where all cells respond, but also significant responses (derivatives) at and astride the two edges. Responses along edges are a common problem in keypoint detection. For example, Difference of Gaussian blob detection used by the SIFT algorithm produces strong responses along edges, just like complex cells in our model, which results in poorly localised features. SIFT uses the ratio of eigenvalues of the Hessian matrix to discard keypoints along edges. In our model, we ap-

ply two inhibition schemes to the responses of complex cells to suppress such responses when applying end-stopped cells. Tangential inhibition serves to suppress all responses astride the edges. Radial inhibition suppresses responses at the edges but, because of the orthogonal kernels, not at the corner. For a detailed explanation of the inhibition schemes used in our algorithm, we refer to Fig. 10 in [14].

2.1. Multi-scale Filter Kernels as V1 Model

The multi-scale extension of the Heitger *et al.* model [2] applies the same derivation and inhibition schemes. Obviously, at coarser scales the sizes of all cell models are bigger, and this makes the multi-scale model so expensive in terms of computations.

In our new model, each layer of cells is modelled as a linear filtering operation, where the kernel corresponds to a typical weight profile of a particular type of cell. Unlike the original computational approach [2], this formulation allows for easy parallel implementation on GPUs and consistent use of filtering in the frequency domain. As is common, we define simple cells using complex Gabor filters

$$g_{\lambda,\sigma,\theta}(x,y) = \exp\left(-\frac{\tilde{x}^2+\gamma\tilde{y}^2}{2\sigma^2}\right)\exp\left(i\frac{2\pi\tilde{x}}{\lambda}\right), \quad (1)$$

with $\tilde{x} = x\cos\theta + y\sin\theta$, $\tilde{y} = y\cos\theta - x\sin\theta$ and $\gamma = 0.5$. λ is the wavelength (in pixels) and σ the receptive field size (in pixels), which are related by $\sigma/\lambda = 0.56$. θ determines the filter orientation (typically 8 orientations are used). Simple cell responses R are obtained by convolving the image I with the complex Gabor filter, and complex cells C are defined as the moduli of the simple cell responses:

$$R_{\lambda,\theta} = I * g_{\lambda,\theta}\ ; \qquad C_{\lambda,\theta} = \left|R_{\lambda,\theta}\right|. \quad (2)$$

Simple cells respond to line and edge stimuli, complex cells respond to both and exhibit more spatial invariance. All other cells are defined by employing combinations of Gaussian filter kernels. Let $G(\hat{\sigma})$ be a 2D Gaussian function with standard deviation $\hat{\sigma}$ centered at the origin, and $G(x,y,\hat{\sigma})$ its equivalent centered at x,y. Let $ds = 0.6\lambda\sin\theta$ and $dc = 0.6\lambda\cos\theta$ be offsets from the kernel centre. Then kernels representing single- and double-stopped cells are defined by

$$k^S_{\lambda,\theta} = G(ds,-dc,\hat{\sigma}) - G(-ds,dc,\hat{\sigma}), \quad (3)$$

$$k^D_{\lambda,\theta} = G(\hat{\sigma}) - \frac{1}{2}G(-2ds,2dc,\hat{\sigma}) - \frac{1}{2}G(2ds,-2dc,\hat{\sigma}). \quad (4)$$

The parameters used in this step were carefully selected in order to obtain best results. $\hat{\sigma}$ is used to control the amount of smoothing performed at this step, which is useful for reducing the effect of noise, and is typically set to $\sigma/2$. When $\hat{\sigma}$ approaches zero, the kernels become a combination of Dirac functions. This can be a useful optimisation at the expense of noise sensitivity, so we use this in our CPU-based implementation. End-stopped cell response maps are then computed by convolutions

$$S_{\lambda,\theta} = C_{\lambda,\theta} * k^S_{\lambda,\theta}\ ; \qquad D_{\lambda,\theta} = C_{\lambda,\theta} * k^D_{\lambda,\theta}. \quad (5)$$

In order to suppress responses along lines and edges, tangential and radial inhibition are used, as in [2]. Each one is modelled as a layer of inhibition cells represented by the two kernels

$$k^{IT}_{\lambda,\theta} = -2G(\hat{\sigma}) + G(dc,ds,\hat{\sigma}) + G(-dc,-ds,\hat{\sigma}), \quad (6)$$

$$k^{IR}_{\lambda,\theta} = G(dc/2,ds/2,\hat{\sigma}) + G(-dc/2,-ds/2,\hat{\sigma}). \quad (7)$$

Inhibition cell response maps are obtained by convolving the responses of complex cells with these kernels:

$$I^T_{\lambda,\theta} = C_{\lambda,\theta} * k^{IT}_{\lambda,\theta}\ ; \qquad I^R_{\lambda,\theta} = C_{\lambda,\theta} * 2G(\hat{\sigma}) - C_{\lambda,\theta+\pi/2} * A k^{IR}_{\lambda,\theta} \quad (8)$$

where A determines the inhibition strength, usually set between 4 and 16. Note that radial inhibition uses two response maps of complex cells, the second one being orthogonal to the main orientation θ. Figure 2 illustrates all kernels used to obtain the keypoint response maps.

Finally, the keypoint response maps are calculated by applying the inhibition maps, i.e.,

$$K^S_\lambda = \sum_{\theta=0}^{\pi} S_{\lambda,\theta} - \left(\sum_{\theta=0}^{2\pi} I^T_{\lambda,\theta} + \sum_{\theta=0}^{2\pi} I^R_{\lambda,\theta}\right), \quad (9)$$

$$K^D_\lambda = \sum_{\theta=0}^{\pi} D_{\lambda,\theta} - \left(\sum_{\theta=0}^{2\pi} I^T_{\lambda,\theta} + \sum_{\theta=0}^{2\pi} I^R_{\lambda,\theta}\right). \quad (10)$$

Keypoint locations are obtained by detecting local maxima of the inhibited responses. The first sum in Eqns 9 and 10 combines the responses of end-stopped cells, resulting in strong responses at keypoint locations. The two sums inside the brackets represent the inhibition terms, removing responses along and astride lines and edges as described in Section 2, thus ensuring that only well-localised keypoints are kept. We stress that all parameters, both distances and kernel sizes, have been carefully optimised in order to obtain localised peaks at singularities while suppressing all other responses.

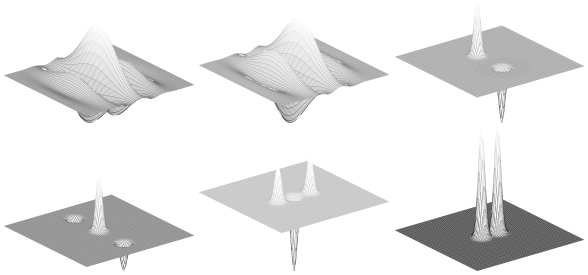

Figure 2: Filter kernels. Top row: even simple cell, odd simple cell, single-stopped cell. Bottom row: Double-stopped cell, tangential inhibition cell, radial inhibition cell.

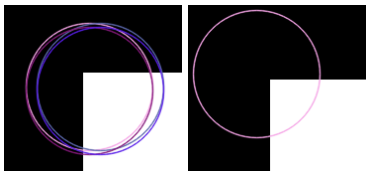

Figure 3: A comparison between [2] (left) and our improved algorithm (right) at scale $\lambda = 128$. The number of spurious keypoints is significantly reduced in general, leading to better repeatability. The loss of precision caused by 32× subsampling is partially compensated by subpixel localisation in the right image.

2.2. Efficient Computation of V1 Responses

Convolutions in Eqns 2–8 are very expensive in the case of large kernels, so filtering should ideally be done in the frequency domain. However, since large filter kernels have a narrow frequency response, careful zero-padding is needed which, again, slows down the filtering. Another problem with the original approach [2] is that steps in Eqns 3 to 8 perform a very sparse sampling of the area surrounding the responses of end-stopped cells when parameter $\hat{\sigma}$ is small, so the resulting activation peaks become increasingly jagged for larger kernels, and this leads to many local maxima. As a result, many spurious keypoints are found at coarser scales, as can be seen in Fig. 3 (left).

Instead of operating on the original image, we create a Gaussian scale-space by convolving the original image with a Gaussian kernel and subsampling it repeatedly. If $I = I_0$ is the original image, and I_s is the image at pyramid level s, then we replace Eqn 2 by

$$R_{\lambda',\sigma',\theta} = I_s * g_{\lambda',\sigma',\theta} \,, \tag{11}$$

where $\lambda' = \lambda/2^s$ and $\sigma' = \sigma/2^s$. Level s is chosen such that λ' always falls within the range $[4, 8)$, as this provides the most reliable results in practice. The algorithm is performed as follows: (i) λ is initialised with the smallest scale $\lambda = \lambda_0$, (ii) the image is convolved with a Gaussian kernel and resampled so that λ' is inside $[4, 8)$, (iii) keypoint detection and merging are applied to the resampled image I_s, and finally (iv) the keypoint coordinates are transformed back into the original image. Then the process is repeated at the next coarser scale.

This process is still biologically plausible – coarse-scale end-stopped cells with large dendritic fields effectively perform a form of subsampling. The new formulation fixes several problems: the resulting peaks are much smoother, leading to fewer spurious keypoints at coarse scales (Fig. 3), and λ' is kept in a range where convolution is fast. Furthermore, since kernel sizes are kept relatively small, we can safely perform filtering in the frequency domain.

2.3. Improved Localisation

An undesired side-effect of employing a Gaussian pyramid is the incurred loss of precision. Using the approach described in the previous section, a filter with wavelength $\lambda = 32$ pixels is replaced by a filter with $\lambda = 4$ pixels on an image 1/8th of the size in each direction. This results in a localisation uncertainty of 8 pixels, which is insufficient for many applications. Since cell responses after inhibition resemble smooth peaks in the keypoint maps, we fit separate 1D parabolas in x and y directions around local maxima of the cell response maps K^S_λ and K^D_λ, and use parabola peaks as x and y coordinates. While this step is not biologically motivated, it is only used at the very end to extract the keypoint locations and to compensate for the error introduced by the numerical optimisations.

2.4. Automatic Scale Selection

Our visual system extracts a huge number of keypoints at many scales, capturing very fine details of scene objects, but leading to redundancy. Only part of this information is needed for many tasks such as object categorisation, but existing biological models have not yet addressed selecting a representative scale for each keypoint. Since our algorithm detects stable keypoints at several scales, we use double-stopped cell responses for scale selection.

We compare the response of each keypoint K at scale λ_i to the corresponding keypoints at neighbouring scales: K^- at λ_{i-1} and K^+ at λ_{i+1}. The corresponding keypoints must be at roughly the same spot: we apply a tolerance of $\lambda_i/4$ pixels. We look for local maxima, so if $T(K)$ is the double-stopped cell response at keypoint K, K is only kept if

$$\Big(T(K^-) < T(K)\Big) \wedge \Big(T(K^+) < T(K)\Big) \tag{12}$$

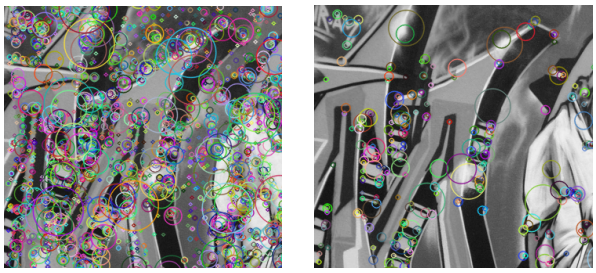

Figure 4: Effect of scale selection illustrated on a section of a real world image (Graffiti from [10]). Left: all keypoints extracted at all scales. Right: 1000 strongest keypoints after scale selection. Both fine-scale (sharp corners) and coarse-scale (curve segments and blobs) keypoints with strong responses are preserved.

is true. This yields keypoints whose end-stopped cell responses are a local maximum in both location and scale, resulting in fewer but still meaningful keypoints. As shown in [2], another possibility is to analyse the congruency of keypoints in scale space. This is also biologically plausible, but more difficult because keypoints tend to drift away when going from fine to coarse scales, and smaller keypoints merge into larger ones, forming complex tree structures which are difficult to process using neuronal models. For this reason, and because we want a real-time method, we apply the simple scheme based on Eqn 12, which only examines neighbouring scales. This results in much faster processing time suitable for real-time performance. The number of keypoints can be further reduced by only selecting the subset with strongest responses. Figure 4 shows the result of scale selection on a real image.

2.5. GPU-Accelerated Implementation

Due to their inherently parallel design, GPUs are often used for speeding up parallel computations. Our algorithm is computationally simple, but it generates many layers of information, in which each cell needs to examine its neighbourhood. This leads to complex memory access patterns, where many megabytes of data have to be stored in global GPU memory.

We have a CUDA-based and an OpenCL-based version of the GPU keypoint algorithm. They use several threads to minimise the time which the CPU and GPU spend waiting for each other. The first thread reads and pre-processes images from a camera and pushes them onto a stack. The second thread reads the last image and clears the stack, then transfers the image to the GPU, applies the kernels, and stores the activation maps in a second stack. The third thread reads the last image from the second stack and runs the final maxima localisation step on the CPU, and finally displays the results.

Figure 5: Example images from the repeatability benchmark. Top row: Leuven, Trees and Wall. Bottom row: Graffiti, Bark and Boat.

Our GPU implementation running on a GeForce GTX 560 Ti is about 10 times faster than the CPU version running on a quad-core Intel i5 760 processor with a 2.8 GHz clock speed, making it fast enough for real-time scenarios.[1]

3. Keypoint Evaluation

In this section, we compare the repeatability and speed of our algorithm to the state of the art in keypoint detection.

3.1. Repeatability

The algorithm was tested by applying the well-known repeatability benchmark by Mikolajczyk *et al.* [10]. This benchmark consists of images of planar scenes (Fig. 5). For each scene there is a reference image, and five additional images which show the same scene after undergoing a transformation, such as rotation, scaling, blur, or perspective. The homography between the first image and the remaining five images is known, so detections in the first image can be projected into the remaining images. A detection is "repeated" if there is a region in the second image which overlaps with the projected region with less than 40% error. For this purpose, we defined meaningful regions as circles centered around each keypoint with diameter λ (the detection scale).

We compared our approach to a number of state-of-the-art scale invariant detectors: SIFT [6], SURF [7] and SFOP [5], as well as two affine-invariant detectors: Hessian affine [10] and MSER [8]. SIFT, SURF and SFOP are scale-invariant interest point detectors and compete directly with our algorithm. SIFT and SURF in particular are known for their reliability and versatility. Hessian affine and MSER are included since they represent

[1] A video demo is available at `w3.ualg.pt/~kterzic/videos.html`

the state of the art in region detection and are best-in-class on many tasks in this benchmark. We used the original Matlab/C++ benchmark code provided by the authors. We also used author-supplied implementations of the Hessian affine, SFOP and MSER detectors, applying the default parameters. For SIFT and SURF we relied on widely used OpenCV implementations. We used our CPU-based implementation for testing, but our GPU implementation produces comparable results. Our implementation based on OpenCV is available as open source so others can reproduce our results. [2]

For convenience, we refer to our algorithm as "BIMP" (Biologically Inspired Multiscale keyPoints). Unless stated otherwise, we used seven scales, $\lambda \in \{8, 8\sqrt{2}, 16, 16\sqrt{2}, 32, 32\sqrt{2}, 64\}$, eight orientations θ equally spaced on $[0, \pi)$, and no scale selection.

Figure 6 shows a selection of results. It can be seen that our detector performs very well, showing best-in-class performance in many cases. It significantly outperforms the state of the art with respect to illumination changes (the Leuven image set), and also beats the state of the art with small affine transformations (Wall) and blur (Trees). As expected, it fails with strong affine distortions (Wall), but still outperforms the other non-affine invariant detectors on this set.

The scales used by our algorithm were spaced half an octave apart, which is more than the maximum acceptable scale error in this benchmark. This way, scales are sampled densely enough to match a keypoint from the first image of a set to a corresponding keypoint in the remaining images at a proper scale, but sparsely enough to prevent many possible matches for any given keypoint. So despite not using a scale selection mechanism in this step, our algorithm does not have an unfair advantage.

In order to measure localisation precision, we have also plotted repeatability as a function of overlap error, as suggested in [5]. The results can be seen in Fig. 7: the repeatability between the first and the third images of the Boat and Leuven sequences, respectively. Our detector achieves the best results on the illumination benchmark (Leuven) and is highly competitive on the rotation/zoom benchmark (Boat).

3.2. Speed

Performance is a common problem of biological approaches due to many convolutions. We have both a CPU-based and a GPU-based implementation of our algorithm. Table 1 shows the CPU time our test computer (quad-core Intel i5) needed to process the first image of

[2] Code available at `w3.ualg.pt/~kterzic/software.html`

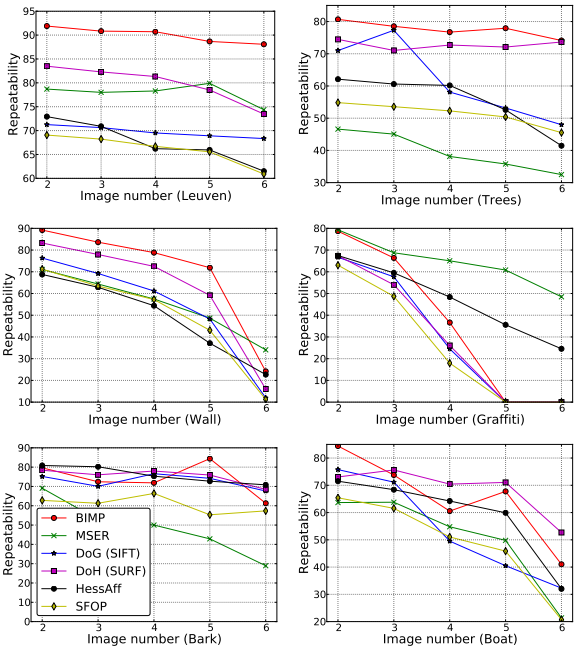

Figure 6: Repeatability on standard image sets compared to state-of-the-art detectors. Keypoints detected in the first image of a set are compared to those in images 2-6. Top row: illumination change (left) and blur (right). Our detector performs exceptionally well on these. Middle row: perspective change. Our algorithm performs well with small affine distortions, but fails with large ones, like all detectors lacking affine invariance. Bottom row: rotation and scaling. Our detector is competitive with the state of the art.

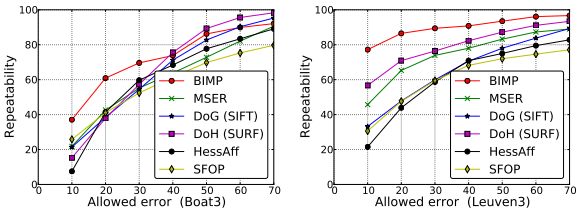

Figure 7: Repeatability as a function of maximum allowed overlap error. Our algorithm is one of the best performing detectors.

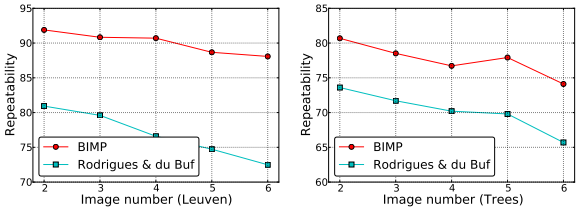

Figure 8: Repeatability compared to Rodrigues and du Buf [2] on two image sets. Due to the reduction of spurious keypoints at coarse scales, our new algorithm significantly outperforms the original algorithm while requiring only a fraction of the time.

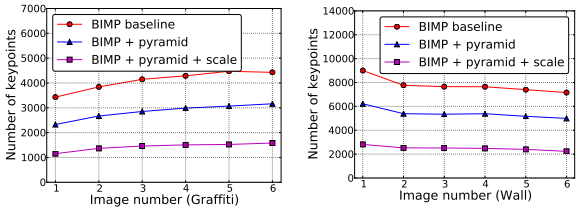

Figure 9: Keypoint reduction due to different stages of our algorithm. Starting with a baseline re-implementation of the original algorithm from [2], we add pyramidal processing and scale selection. Each step significantly reduces the number of detected points, performing more stringent feature selection.

the Graffiti set, averaged over ten trials. Table 2 shows a comparison of our GPU-accelerated implementation with the state of the art. It can be seen that, although our CPU implementation is still slower than the fastest computational approaches, it is significantly faster than the original implementation [2] and several modern algorithms such as IBR, EBR and SFOP. The improvement over the original biological method [2] is particularly striking. As expected, our GPU implementation is faster than any CPU-based algorithm, achieving 50 frames per second at a resolution of 600x400 pixels on a GeForce GTX 560 Ti. Since part of the time is spent converting between formats and transfering data over the PCI bus, this can be optimised even further. One of the reasons for the competitive performance is the algorithm's parallel nature, which makes good use of modern multicore CPUs and GPUs. The performance shows that biological algorithms can be feasible for many computer vision tasks even when they run on a CPU, especially given the precision shown in this section, and that real-time vision based on cortical models is already feasible due to GPU processing.

3.3. Comparison to the Baseline Algorithm

Our new algorithm is based on multiscale keypoints of Rodrigues and du Buf [2]. We compared the speed and reliability of our algorithm with the original algorithm from [2], using code provided by the authors.

Table 1 shows the run-time on a medium-sized image. It can be seen that our implementation outperforms the original implementation by four orders of magnitude, which is a very large improvement. Whereas the original biological approach is not suitable for most applications, our new algorithm is comparable to the state of the art in terms of speed. Most of this increase in speed comes from filtering in the frequency domain and using a Gaussian pyramid. Additional improvement is due to software optimisations. Using our GPU implementation, we gain another order of magnitude (see Table 2).

We also compared the effect that our modifications have on keypoint repeatability on two sets of images from the repeatability dataset (Fig. 8). Our algorithm significantly improves on the original implementation, owing to the reduction of spurious keypoints at coarse scales, as illustrated in Fig. 3. Finally, we evaluated the effect of using a Gaussian pyramid and scale selection on the number of detected keypoints (Fig. 9). The first step reduces spurious keypoints at coarse scales, the second step reduces redundancy caused by similar keypoints at neighbouring scales.

4. Keypoint-based Object Categorisation

It is believed that end-stopped cells in V1 play an important role in visual attention, by identifying local regions with large complexity. In contrast, most object recognition methods from computer vision sample invariant descriptors on a global, dense and periodic grid, which yields large and redundant feature vectors and which leads to long learning and/or classification times. Interest points can reduce the amount of data used for recognition, but they have not yet been able to offer competitive results. Below, we apply the state-of-the-art *Local Naive Bayes Nearest Neighbour* classifier [19]. We show that by using our keypoints for feature selection, we can achieve a performance similar to the grid-based approach while using only a fraction of the number of descriptors.

4.1. Categorisation Approach

We detect keypoints at standard scales and extract a SIFT descriptor at each keypoint location,[3] with its diameter equal to the keypoint wavelength λ (see Fig. 10). Essentially this means dense sampling around points of high complexity and sparse or no sampling elsewhere. Most standard categorisation methods sample overlapping descriptors on a dense grid, spaced only 2-3 pixels apart. This method can capture all important information in the image, but results in a very large number of descriptors, and hence in slower learning and recognition. In this section, we show that by sampling only at keypoint locations, we can capture most of the important information while using only a small number of descriptors.

[3] Although SIFT descriptors are not very biologically founded, they are widely used for categorisation because of their performance. In this paper, we only evaluate the benefit of using V1 keypoints for data selection and thus use the same descriptors as used by competing methods. This enables a fair comparison.

Table 1: Runtimes on the first image from the Graffiti set, 800×640 pixels (lower is better). We compared BIMP against the original biological approach by Rodrigues and du Buf [2], EBR [9], SFOP [5], IBR [9], Hessian Affine [10], SIFT [6], SURF [7] and MSER [8].

detector	Rodrigues-du Buf	EBR	SFOP	IBR	HessAff	SIFT	**BIMP**	SURF	MSER
time (s)	3200	48.7	9.14	3.96	1.01	0.50	**0.32**	0.18	0.13

Table 2: Runtimes on the first image from the Graffiti set using GPU-accelerated detectors. We tested against GPU-based implementations of SURF [7] and ORB [18]. GPU times were obtained using the OpenCV implementation and exclude GPU transfer times.

detector	G-SURF	**G-BIMP**	G-ORB
time (s)	0.036	**0.028**	0.024

Figure 10: Comparison of keypoint-based data selection and a grid-based sampling approach. Centre: image from Caltech 101. Left: keypoints sampled on a regular grid. Right: our keypoints. In real applications, multiple scales and much denser sampling are used, with descriptors sampled at 2 or 3 pixel intervals. In this simplified figure, we only show one scale and coarser sampling for clarity.

After extracting descriptors at keypoint locations, we apply the local NBNN algorithm of [19], which we briefly summarise here for completeness. Each descriptor is augmented by its location in the image (we use the image centre as the origin) scaled by a factor $\alpha = 1.6$. Factor α is a part of the NBNN algorithm and determines the relative importance of keypoint appearance (represented by a SIFT descriptor) and its location in the image. The conditional probability of a descriptor d_i given a class C is approximated by using r nearest neighbours. We use the Manhattan distance.

$$P(d_i|C) \approx \frac{1}{L_C} \sum_{j=1}^{r} K(d_i - d_j^C), \tag{13}$$

where L_C is the total number of descriptors associated with class C in the training set, and K is the Parzen kernel, i.e., a Gaussian. Classification is done by applying a sum of log-odds:

$$C = \underset{C}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \log \frac{P(d_i|C)}{P(d_i|\bar{C})} + \log \frac{P(C)}{P(\bar{C})} \right], \tag{14}$$

where $\bar{C}$ is the set of all $N-1$ classes other than C. $P(d_i|\bar{C})$ is approximated by a single sample, the next nearest neighbour $r+1$. A set of K-D trees is constructed and searched in parallel for efficient nearest neighbour lookups.

4.2. Categorisation Complexity

The complexity of the local NBNN classification algorithm is $O(cN_D \log(N_C N_T N_D))$, where N_C is the number of classes, N_T the number of training images per class, N_D the average number of descriptors per image, and c the number of times a K-D tree should be traversed. In our case, c, N_C and N_T are the same, but a decrease in N_D, for example by using our keypoints, results in a more-than-linear speed improvement. Lowering N_D from 2000 to 200 results in a speed-up of more than 12× on Caltech 101, and reduces memory requirements by a factor of 10.

4.3. Categorisation Performance

We tested the performance on two popular object categorisation datasets: Caltech 101 [20] and Caltech 256 [21] following standard evaluation procedures. We performed 10 random splits of the data into disjoint training and testing sets (using 15 and 30 training images per class) and report the mean classification rate and standard deviation. All images were scaled to a uniform size (long side scaled to $l = 300$px, preserving the aspect ratio). SIFT descriptors were extracted from the image at the locations provided by our method and on a dense grid for comparison. When comparing with the results from the literature, we report the average number of features per image used by competing methods. Since we are interested in feature selection and reducing the number of features needed for categorisation, we report classification rates as a function of the number of used features. This is achieved primarily by varying the number of scales. We are motivated by biological vision: coarse scale information propagates faster and leads to quicker (but less accurate) recognition.

4.3.1. Caltech 101

This dataset contains natural images from 101 different categories, with about 40 and 800 images per category. Figure 11 shows the results. Our own keypoints consistently outperform grid results, especially

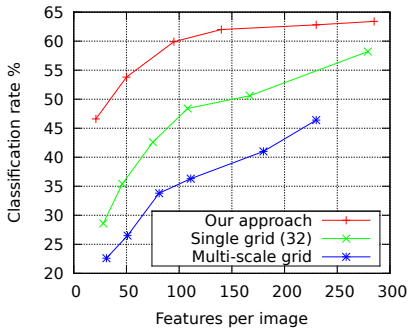

Figure 11: Classification rates on Caltech 101 as a function of the number of used features in the case of 15 training images. The multi-scale grid using 4 scales as in [19] does poorly with few features. Single-scale grid with a fixed feature diameter of 32 pixels does better due to a higher density of features. Our keypoint method performs consistently well and degrades gracefully.

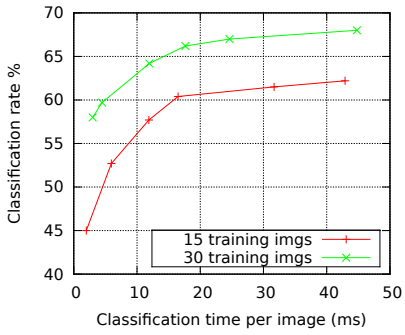

Figure 12: Trade-off between classification time (in milliseconds) and accuracy on the Caltech 101 dataset. Slower recognition increases accuracy, but very good results can already be obtained at 50 frames per second (20 ms). As a comparison, grid-based local NBNN needs about a second per image for the same performance, classic NBNN about 10 seconds [19]. LLC needs about 250ms per image and a long learning stage.

when using a smaller number of features. Good performance with few features indicates that our keypoints consistently capture the most relevant information in the image. As we increase the number of keypoints, our results asymptotically approach the best published results with a dense grid using an NBNN method. We note that in our experiments, the use of SIFT and SURF keypoints failed to match the grid-based approach. Table 3 shows a comparison with state-of-the-art categorisation methods, all using the same SIFT descriptors. Our method can maintain state-of-the-art performance even when using only a fraction of the features as used by the other methods. Our best results were obtained by using the default parameters which were used in all previous experiments: 8 orientations and 7 scales logarithmically spaced between $\lambda = 8$ and $\lambda = 64$. We emphasise that a dense grid provides a superset of our features and that we cannot expect to outperform these more expensive methods. However, our results demonstrate that a good feature selection can achieve comparable results at a fraction of the cost as shown in Fig. 12 (minor differences are due to implementation details). The best reported local NBNN results using a very dense grid [19] were slightly better than the ones reported here (66.1% and 71.9% using 15 and 30 training images, respectively) but the authors did not report the exact sampling method. Still, we come very close to matching these results, using only 371 features per image. Since we are using the NBNN-based classifier from [19], only comparisons with other NBNN methods make sense. We expect that combining our feature selection process with other classifiers such as LLC [22] and ScSPM [23] will enable a similar reduction in necessary features for those algorithms.

Table 3: Comparison to the state of the art on Caltech 101 using SIFT descriptors. We only list results where the average number of features per image was reported by the authors. Numbers marked with (*) were calculated based on the reported sampling method (spacing and number of scales) and an average image size of 300x250 pixels. Note that [19] applies a contrast-based data selection step. We report two results for our method: one with coarse scales only and one with all 7 scales, showing that our algorithms yields competitive results even with one tenth of the points used by grid methods.

	# feat.	15 imgs	30 imgs
NBNN-based methods			
NBNN kernel [24]	2000	61.3±0.2	69.6±0.9
NBNN [24]	2000	62.7±0.5	65.5±1
BIMP+LNBNN	**200**	**64.3±0.5**	**69.7±0.5**
Local NBNN [19]	1639	65	
BIMP+LNBNN	**371**	**65.7±1.0**	**72±1**
Non-NBNN methods			
Gehler *et al.* [25]	3000*	54.5±0.9	63.8±1
SPM [26]	1172*	56.4	64.6±0.8
LLC [22]	3516*	65.4	73.4
ScSPM [23]	1400*	67±0.5	73.2±0.5
NBNN+phow [24]	2000	69.2±0.9	75.2±1.2

4.3.2. Caltech 256

The Caltech 256 dataset is a considerably more challenging benchmark compared to Caltech 101. It features a much larger number of classes, posing a more difficult problem. Since we use the local NBNN algorithm for classification [19], the increase in computational complexity is only logarithmic with respect to the number of classes. The objects in this benchmark are not always centered in the image and there is a larger variation in

Table 4: Comparison to the state of the art on Caltech 256 using SIFT descriptors. Literature results are summarised from [19], which used 2000 features per image or more. As in Table 3, we report two results for our method, one with coarse scales only, and one using all scales.

	# feat.	15 imgs	30 imgs
SPM [26]		27.3±2.6	33.1±0.5
NBNN [27]		30.5	37
ScSPM [23]		33.2±0.8	39.5±0.4
Local NBNN [19]		33.5±0.9	40.1±0.1
BIMP+LNBNN	**340**	**31.4±0.4**	**37.0±0.8**
BIMP+LNBNN	**575**	**33.5±0.5**	**39.3±0.6**

pose. In addition, the objects occupy smaller parts of the images than in Caltech 101, which means that larger parts of the images represent the background.

Table 4 shows our results compared to several state-of-the-art methods. We compare only against methods based on SIFT features and do not include far more complex ensemble classifiers. Unfortunately, most publications do not report the exact number of features used for Caltech 256, but they typically use the same multi-scale grid as used for Caltech 101, with at least 2000 features per image. It can be seen that we can once again almost match the state of the art with a very modest number of features.

5. Conclusion

We presented a biologically-inspired keypoint operator inspired by the Rodrigues and du Buf model [2]. The new algorithm expands on the original in several important ways and is considerably faster, with a performance on a modern CPU approaching that of other computational algorithms. The results on the standard repeatability benchmark from Mikolajczyk *et al.* [10] show that our detector significantly outperforms the state of the art in keypoint detection with respect to illumination changes (Leuven) and textured blur (Trees), and is best-in-class in many other scenarios, particularly rotation (Bark) and small affine transformations (Wall). The lack of affine invariance makes it a poor match in the case of strong perspective distortion, but it still outperforms other non-affine invariant detectors in this scenario.

V1 is believed to play an important role in early attention, so we also tested the ability of our keypoints to capture important information in an image. Our experiments on combining keypoint extraction with SIFT descriptors and Naive Bayes Nearest Neighbour classifier showed that it is possible to achieve state-of-the-art categorisation performance on the Caltech 101 and Caltech 256 benchmarks, but using only a fraction of the data as used by leading methods – significantly reducing memory requirements and runtime.

What makes our detector interesting is its biological background, as it explicitly models V1 cortical cells. This means that our algorithm not only gives excellent and reliable multi-scale keypoints, but can also be used as a basis for neuronal models of vision. We believe that the combination of state-of-the-art results and biological plausibility makes this work interesting to both computer vision and biological vision communities.

In the future, we are planning to complement keypoints with biologically inspired region information, like colour and texture, leading towards fully biological object recognition. We are also working on extending our keypoint work to include affine invariance.

Acknowledgements. This work was supported by the EU under the FP-7 grant ICT-2009.2.1-270247 *NeuralDynamics* and the FCT under the grant PEst-OE/EEI/LA0009/2011 .

References

[1] T. Hansen, H. Neumann, Neural mechanisms for the robust representation of junctions, Neural Computation 16 (2004) 1013–1037.

[2] J. Rodrigues, J. du Buf, Multi-scale keypoints in V1 and beyond: Object segregation, scale selection, saliency maps and face detection, BioSystems 86 (2006) 75–90.

[3] C. Harris, M. Stephens, A combined corner and edge detector, in: The Fourth Alvey Vision Conference, 1988, pp. 147–151.

[4] J. Shi, C. Tomasi, Good features to track, in: CVPR, 1994, pp. 593 – 600.

[5] W. Förstner, T. Dickscheid, F. Schindler, Detecting interpretable and accurate scale-invariant keypoints, in: ICCV, Kyoto, Japan, 2009.

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

[7] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Computer Vision and Image Understanding 110 (3) (2008) 346–359. doi:http://dx.doi.org/10.1016/j.cviu.2007.09.014.

[8] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, in: BMVC, 2002, pp. 384–393.

[9] T. Tuytelaars, L. J. Van Gool, Matching widely separated views based on affine invariant regions, International Journal of Computer Vision 59 (1) (2004) 61–85.

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Van Gool, A comparison of affine region detectors, IJCV 65 (2005) 2005.

[11] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Object recognition with cortex-like mechanisms, IEEE T-PAMI 29 (3) (2007) 411–426.

[12] S. Fidler, A. Leonardis, Towards scalable representations of object categories: Learning a hierarchy of parts, in: CVPR, Minneapolis, 2007.

[13] E. Barth, C. Zetzsche, G. Krieger, Endstopped operators based on iterated nonlinear center-surround inhibition, in: Human Vision and Electronic Imaging III, SPIE, Vol. 3299, 1998, pp. 67–78.

[14] F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, O. Kuebler, Simulation of neural contour mechanisms: from simple to end-stopped cells, Vision Res. 32 (5) (1992) 963–981.

[15] R. Würtz, T. Lourens, Corner detection in color images by multiscale combination of end-stopped cortical cells, Image Vision Comput. 18 (6–7) (2000) 531541.

[16] C. Grigorescu, N. Petkov, M. Westenberg, Contour detection based on nonclassical receptive field inhibition, IEEE Trans. Im. Proc. 12 (7) (2003) 729–739.

[17] K. Terzić, J. Rodrigues, J. du Buf, Real-time object recognition based on cortical multi-scale keypoints, in: J. M. Sanches, L. Micó, J. S. Cardoso (Eds.), IbPRIA 2013, LNCS, Vol. 7887, Springer, Funchal, Madeira, Portugal, 2013, pp. 314–321.

[18] E. Rublee, V. Rabaud, K. Konolige, G. R. Bradski, ORB: An efficient alternative to SIFT or SURF, in: ICCV, Barcelona, 2011, pp. 2564–2571.

[19] S. McCann, D. G. Lowe, Local naive bayes nearest neighbor for image classification, in: CVPR, Providence, 2012, pp. 3650–3656.

[20] R. F. L. Fei-Fei, P. Perona, Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories, in: CVPR Workshop on Generative-Model Based Vision, Washington DC, 2004.

[21] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset, Tech. Rep. 7694, California Institute of Technology (2007).
URL `http://authors.library.caltech.edu/7694`

[22] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: CVPR, San Francisco, 2010.

[23] J. Yang, K. Yu, Y. Gong, T. S. Huang, Linear spatial pyramid matching using sparse coding for image classification., in: CVPR, IEEE, 2009, pp. 1794–1801.

[24] T. Tuytelaars, M. Fritz, K. Saenko, T. Darrell, The NBNN kernel, in: ICCV, Barcelona, 2011.

[25] P. V. Gehler, S. Nowozin, On feature combination for multiclass object classification, in: ICCV, 2009, pp. 221–228.

[26] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: CVPR, New York, 2006, pp. 2169–2178.

[27] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: CVPR, Anchorage, 2008.