

# Cascade of Box (CABOX) Filters for Optimal Scale Space Approximation

Victor Fragoso, Matthew Turk  
University of California Santa Barbara  
{vfragoso,mturk}@cs.ucsb.edu

Gaurav Srivastava, Abhishek Nagar, Zhu Li, Kyungmo Park  
Samsung Research America, Richardson, TX  
{gaurav001.s,a.nagar,zhu1.li,kyungmo.park}@samsung.com

## Abstract

Local image features, such as blobs and corners, have proven to be very useful for several computer vision applications. However, for enabling applications such as visual search and augmented reality with near-realtime latency, blob detection can be quite computationally expensive due to numerous convolution operations. In this paper, we present a sparse convex formulation to determine a minimal set of box filters for fast yet robust approximation to the Gaussian kernels used for blob detection. We call our feature detector as CABOX (CAscade of BOX) detector. Although box approximations to a filter have been studied in the literature, previous approaches suffer from one or more of the following problems: 1) ad hoc box filter design, 2) non-elegant trade-off between filter reconstruction quality and speed and, 3) limited experimental evaluation considering very small datasets. This paper, on the other hand, contributes: 1) an elegant optimization approach to determine an optimal sparse set of box filters, and 2) a comprehensive experimental evaluation including a large scale image matching experiment with about 16 K matching and 170 K non-matching image pairs. Our experimental results show a substantial overlap (89%) between the features detected with our proposed method and the popular Difference-of-Gaussian (DoG) approach. And yet CABOX is 44% faster. Moreover, the large scale experiment shows that CABOX closely reproduces DoG’s performance in an end-to-end feature detection and matching pipeline.

## 1. Introduction

Local image features are at the heart of several applications in computer vision such as augmented reality [4], structure from motion [6], and image search and retrieval [11]. Various feature detectors identifying local image patterns like corners and blobs, have been proposed [3, 5, 10, 14] and successfully used in these appli-

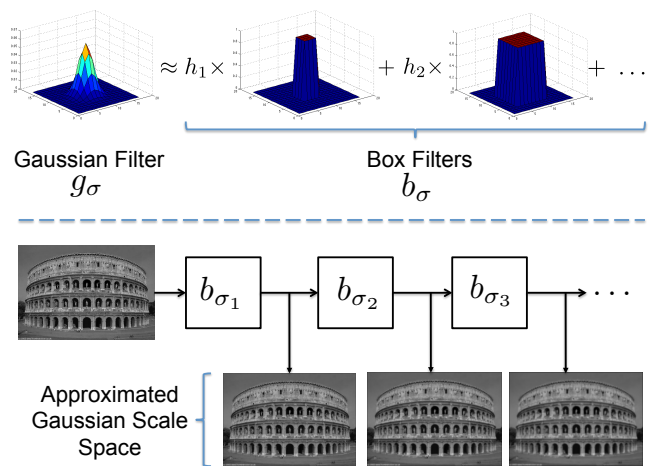


Figure 1: (Top) CABOX approximation to a Gaussian filter. (Bottom) The box filters  $b_\sigma$  used to compute Gaussian Scale Space.

cations. The widely used SIFT [10] combines blob feature detector and gradient histogram descriptor to generate robust keypoint representation for an image.

Blobs have proven to be effective in terms of repeatability across several imaging conditions as well as rotation and scale covariance [3, 7, 10]. Blob detection is typically performed using a multi-scale scheme whereby a series of Laplacian-of-Gaussian (LoG) filters at different scales detect several features of different sizes. Due to the computational overhead of LoG, Lowe [10] proposed to approximate the LoG filter by Difference-of-Gaussian (DoG), which can be implemented efficiently due to the separability property of the Gaussians. In the recent years, speed optimizations to the local feature detection have been proposed through the use of box filters (e.g. CenSurE [3], SURF [5], fast approximated SIFT [8]).

These approaches, although efficient, lack optimality in the set of box filters used for approximating the LoG or Gaussian filters. The size, height and number of box fil-

ters are usually hand-crafted. There is no principled elegant way of determining a minimal set of box filters that result in a desirable trade-off between the filter reconstruction error and the speed of operation. Moreover, the experimental evaluations are usually limited to small image datasets. One most commonly used dataset in such evaluations is Oxford dataset [2] which contains only 48 images. From a practical standpoint, it is imperative to evaluate new detector schemes on large scale datasets for real-world applications such as visual search and image retrieval. A performance evaluation on an end-to-end system is necessary where: 1) first, keypoints are detected using a new method, 2) then descriptors are computed for these keypoints, 3) local and global representations are created for an image using these descriptors, and 4) finally, two images are matched using such local and global descriptors. The MPEG (Moving Picture Experts Group) is currently working on creating the CDVS (Compact Descriptors for Visual Search) standard [11] that aims at standardizing different aspects of this pipeline. They have created a large benchmark dataset [1] of about 16K matching image pairs, 170K non-matching image pairs, and 1 million distracter images to evaluate different algorithms for feature detection, description, global descriptor aggregation and image matching and retrieval.

In order to address the above mentioned issues, this paper proposes an elegant optimization approach to determine an optimal set of box filters for approximation. The proposed sparse convex formulation finds fewest number of box filters while still maintaining good quality filter reconstruction. We investigate two different box filter dictionaries for sparse approximation and show that for Gaussian kernel approximation, the dictionary with concentric box filters is more suitable for obtaining fewer box filters and maintaining a low filter reconstruction error. In Fig. 1 we illustrate the approximation of a Gaussian kernel as well as the Gaussian scale space computation. Another unique contribution is the large scale extensive evaluation of the detector performance on the CDVS dataset. The experimental results suggest that CABOX can approximate the Gaussian scale space accurately while reducing the blob detection time by 44 % compared to the popular DoG approach. Moreover, our pairwise matching experiment shows that CABOX detects features that can reproduce SIFT's performance in the end-to-end image matching pipeline.

## 2. Related work

Bay et al. [5] proposed the SURF feature detector based on an approximation of second order Gaussian derivatives using box filters. In contrast to other multi-scale detectors (e.g. SIFT [10]) where the image is downsampled, they upscaled the approximated filter, leveraging the fast convolution of box filters using integral images. They used 3-4 boxes for the approximation and the box filter heights are hand-selected with no justification provided. They used a

dataset of 216 images to demonstrate the practical applicability of the detector for object recognition experiment.

Agrawal et al. [3] introduced CenSurE, a fast feature detector that approximates LoG filter using bi-level center-surround filters. They proposed different approximated versions including octagonal filter for more accuracy and box filters for higher speed. The box filter approximation used 2 boxes, which amounts to using only one box to approximate a Gaussian. This certainly is a non-optimal approximation and the filter heights require choosing an initial parameter whose details are not provided as to how it is chosen.

The closest work to ours is by Grabner et al. [8], where the LoG filtering operation is replaced by a Difference-of-Mean (DoM) filter which approximates a Gaussian with a single box filter. Their evaluations are conducted on only 17 images. Another similar work is by Pires et al. [13] who proposed an iterative but quite elaborated method to find box filter approximation for a given kernel. Hussein et al. [9] describe an interesting approach to non-uniform filter approximation using kernel integral images.

In contrast, CABOX selects multiple box filters whose heights are determined by solving a convex optimization problem with sparsity constraint on the number of box filters. This not only helps in providing a good approximation but also using fewest box filters.

## 3. CABOX

In this section we first describe the optimization problem whose solution determines which box filters to use for approximating a given filter (e.g. Gaussian or LoG). Subsequently, we describe the approximation of the Gaussian scale space using box filters for blob detection.

### 3.1. Filter approximation

We propose to approximate a given 2D-filter  $\mathbf{g}$  as a linear combination of several box filters  $\mathbf{b}_i$ . Thus, we need to find the coefficients to minimize some approximation error. We solve the following constrained optimization problem:

$$\begin{aligned} \underset{\mathbf{h}}{\text{minimize}} \quad & \frac{1}{2} \|\mathbf{g} - \mathbf{B}\mathbf{h}\|_2^2 + \lambda \|\mathbf{h}\|_1 \\ \text{subject to} \quad & \mathbf{1}^T \mathbf{B}\mathbf{h} = \alpha \end{aligned} \quad (1)$$

where  $\mathbf{g} \in \mathbb{R}^d$  is the 2D-filter reshaped as a column vector;  $\mathbf{h} \in \mathbb{R}^k$  is the coefficient vector;  $\mathbf{B} \in \mathbb{R}^{d \times k}$  is a matrix whose columns hold the box filters reshaped as a column vectors and which we call dictionary; and  $\alpha$  is a scalar that dictates the normalization of the filter. The equality constraint is useful to normalize the resultant filter, for example, when approximating a LoG filter, the entries must sum up to zero and then  $\alpha = 0$ . For approximating Gaussian kernel, we set  $\alpha = 1$ .

Problem (1) without the equality constrained is known as the L1 regularized least-squares and can be solved via LASSO [15]. Adding the equality constrain, which is an

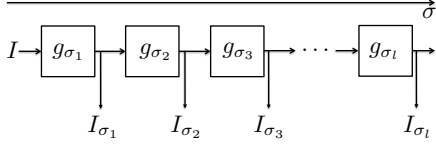


Figure 2: Gaussian scale space computation.

affine function, does not affect the convexity property of the problem and can be solved numerically. It is well known that LASSO tends to provide sparse solutions, which means that only few box filters are used for the approximation. This property is beneficial for our goal as the computational complexity depends on the number of box filters to use.

The choice of dictionary influences the solution considerably, as we must consider the trade-off between approximation accuracy and the number of box filters, so that the solution is computationally appealing. In Sec. 4.1 we discuss the effects of the dictionary and the trade-off.

### 3.2. Gaussian Scale Space approximation

We describe in this section the integration of our filter approximation method described in Sec. 3.1 in the construction of the Gaussian scale space used for detecting blobs.

The Gaussian scale space is constructed by blurring the input image with a series of Gaussian filters. In Fig. 2 we illustrate the scale space construction: the input image is convolved with the first filter in the series, subsequently, the output is again convolved with another filter, and the process is repeated  $l$  times. Once this process is completed, which corresponds to the creation of an octave, the final image is downsampled and passed again to the scale space construction process. The reader is referred to [10, 16] for a more detailed explanation of the scale space construction.

To reduce computational complexity of the required 2D convolutions, separability of the Gaussian filters can be used; this reduces the additions and multiplications for a  $M \times N$  kernel from  $\mathcal{O}(MN)$  to  $\mathcal{O}(M + N)$  per pixel. To reduce the complexity even further, we can approximate the Gaussian kernel using  $k$  box filters, which requires 4 additions and 1 multiplication per box filter per pixel using an integral image. Therefore, our approximation has to use as few box filters as possible to be computationally attractive.

We approximate the bank of Gaussian filters used to construct a single octave offline by solving the optimization problem (1) and setting  $\alpha = 1$ . We build the dictionary,  $B$ , with box filters that are “concentric”; see Fig. 4a for an illustration. This dictionary allows us to have a small number of boxes while keeping the approximation error small. A larger dictionary provides a more accurate approximation but it requires many more box filters (see Sec. 4.1). Subsequently, we take the difference between the blurred images and detect blobs as proposed by Lowe [10] in SIFT.

## 4. Experiments

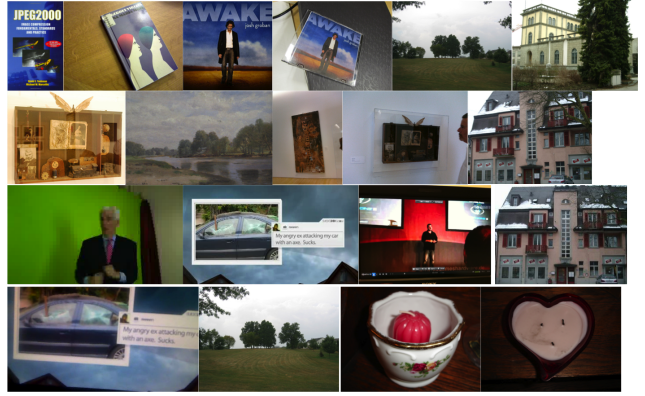
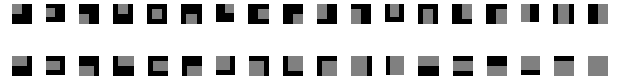


Figure 3: Sample images from the CDVS dataset.



(a) Concentric squares dictionary visualization.



(b) Extended dictionary visualization.

Figure 4: Dictionary visualizations. Fig. 4a: a concentric squares dictionary for approximating a  $13 \times 13$  filter. Fig. 4b: an extended dictionary for approximating a  $4 \times 4$  filter.

We present four major experiments in this section : (1) filter approximation evaluation (Sec. 4.1); (2) Gaussian scale space approximation evaluation (Sec. 4.2); (3) feature detection: a comparison between DoG and CABOX (Sec. 4.3); and (4) large scale pairwise image matching experiment (Sec. 4.4).

**Datasets.** We used two different datasets for our experiments. The first contains three images depicting buildings with rich textures and images of different sizes (see Table 2). The second is the CDVS dataset [1] (see Fig. 3 for a small sample), which is provided by multiple universities and companies participating in the MPEG CDVS standardization. This dataset comprises of different image categories, including paintings, buildings, commonly used objects, {CD, DVD, book} covers, video frames and text documents. It contains 30,256 images and has a total of about 186 K labeled image pairs for benchmarking (16 K matching pairs and 170 K non-matching pairs).

### 4.1. Filter approximation evaluation

This evaluation shows that our approach can approximate the bank of Gaussian filters used to construct the Gaussian scale space in VLFeat [16] with minimal error. We tested two different dictionaries: 1) concentric squares (Fig. 4a); and 2) an “extended” dictionary containing rectangles of different sizes and positions (see Fig. 4b). To build the dictionary, we first calculate the kernel size using the

Table 1: Gaussian filter approximations using concentric squares dictionary and an extended dictionary. The influence of the used dictionary determines not only the quality of the approximation but also the number of boxes required.

$\sigma$	Concentric Squares		Extended	
	Num. of Boxes	Residual	Num. of Boxes	Residual
1.249000	3	0.0554	43	0.0068
1.226270	3	0.0578	21	0.0067
1.545008	4	0.0358	42	0.0146
1.946588	6	0.0248	36	0.0087
2.452547	5	0.0192	16	0.0169
3.090016	8	0.0142	18	0.0163

formula:  $s = 2\lceil 4\sigma \rceil + 1$  (see [16]).




**Concentric squares dictionary construction.** Given the size of the Gaussian kernel  $s$ , we first build a squared box filter of size  $s_1 = 3$ , which is the smallest box filter of odd size. Subsequently, we keep adding squared box filters of size  $s_i = s_{i-1} + 2 \forall i > 1$  to the dictionary until  $s_i = s$ . In Fig. 4a we illustrate a concentric squares dictionary construction for a filter of size  $s = 13$ . This dictionary grows in a linear manner as a function of  $s$ , which helps in maintaining a low computational cost, but potentially increases the approximation error due to fewer box filters to reconstruct the Gaussian kernel.

**Extended dictionary construction.** Given the size of the Gaussian kernel  $s$ , we first build all the possible rectangles, including squares with minimal size  $s_i = 2$  and which fit within the area of the Gaussian kernel. Subsequently, we increase the minimal size by one, i.e.  $s_{i+1} = s_i + 1$ , and build all the possible rectangles in the same manner. In Fig. 4b we show an example of the extended dictionary construction for a kernel size of  $s = 4$ . This dictionary grows in an exponential manner as a function of  $s$ , which means that more boxes can be used to approximate the Gaussian kernel and lower the error but it increases the computational cost.

The results of the approximations of all the Gaussian kernels used in VLFeat are shown in Table 1. Expectedly, the extended dictionary provides a better approximation, as indicated by the residual  $\|\mathbf{g}_\sigma - B\mathbf{h}\|_2$ , but requires more box filters. On the other hand, the concentric squares dictionary causes higher residual while requiring fewer box filters. The remarkable point to note is that the residuals produced by the two different dictionaries are quite small and still the number of box filters with concentric squares dictionary is an order of magnitude smaller than those resulting with the extended dictionary. Therefore we chose the concentric squares dictionary for further experiments.

Fig. 5 shows a qualitative assessment of the approximations produced by using the different dictionaries. Top row shows the original Gaussian filters, middle row shows the approximation with extended dictionary and the bottom row

Table 2: The Root Mean Square Error (RMSE) of our approximation method across all octaves and scales is minimal, considering that the pixel intensities are in the range of 0 to 1.

	Colosseum	Oxford Bldg.	Zürich
			
RMSE	0.0671	0.0387	0.0794
Octaves	5	6	5
Scales	6	6	6

shows the approximation with the concentric squares dictionary. The extended dictionary produces approximations closely resembling original Gaussian filters while the concentric squares dictionary produces pyramid-like approximations.

## 4.2. Gaussian Scale Space evaluation

We present in this section, an assessment of the quality of the scale space constructed by the box filter approximations, as well as the estimation of the computational savings. We implemented CABOX as a smoothing function using our box filter approximations and an integral image in VLFeat. This function was invoked from the code that builds the scale space replacing the original VLFeat blurring function. We compared the constructed scale space against the scale space produced by the Gaussian filters at every scale and per octave.

The comparison of these two scale spaces is in terms of the Root Mean Squared Error (RMSE) between corresponding images at every level of an octave and across all octaves. Finally, all the RMSEs were averaged. The results are shown in Table 2.

The average RMSE shows a small error in the construction of the scale space using Gaussian filters and our approximations, where the pixel intensities were in the range  $[0, 1]$ . Thus, we can conclude that our proposed approximation is very close to the one built with Gaussian filters.

**Timings** To evaluate the scale space construction time, we used 19 images (Fig. 3) chosen at random from the CDVS dataset. These images have several dimensions and depict various objects and natural scenes. For every image, we measured the time involved in the construction of the scale space per octave, including the computation of the required integral images. Fig. 6 presents these time measurements. From the figures and data we conclude that CABOX on average reduced the scale space construction time in approximately 44 %. The average time savings for the first, second, third, fourth, and fifth octaves were approximately 22 %, 32 %, 54 %, 52 %, and 59 %, respectively. This confirms that CABOX brings computational benefits.

## 4.3. Feature detection evaluation

We present an evaluation of blob detection, specifically a comparison of the features detected by VLFeat and



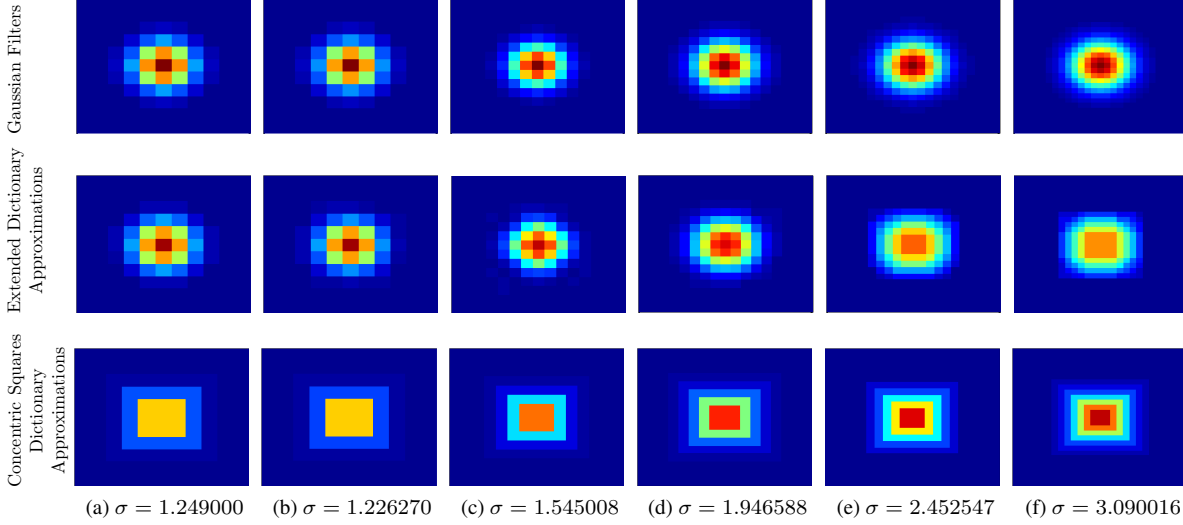


Figure 5: Gaussian filters and their box approximations computed with two different dictionaries. (Top row) Original Gaussian filters, (Middle row) approximation with the extended dictionary, (Bottom row) approximation with the concentric squares dictionary.

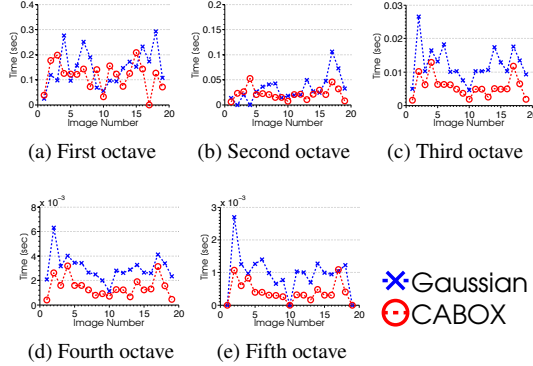


Figure 6: Gaussian scale-space construction times for several octaves for 19 different images. CABOX on average tends to be faster. In particular, the higher the octave, the faster CABOX is.

CABOX. More precisely, we are interested in evaluating which features are detected by both methods (considering location and scale) and how many new features are introduced due to Gaussian approximation errors.

For this experiment, we used the same three images as for evaluating the scale space construction (Sec. 4.2). We ran both detectors (VLFeat and CABOX) with a peak threshold of 0.04 and an edge threshold of 10 and stored the location and scale of every feature detected. To measure the number of features detected by both, we searched in the set of features detected by CABOX for the nearest neighbor of every feature detected by VLFeat, and considered them as the “same” feature when the location difference  $< 5$  pixels and the scale difference  $< 2^{1.5}$ . 89 % of those features detected with CABOX are also detected by the DoG approach (VLFeat), while introducing on average 11 % new features. We present in Fig. 7 a visualization

of VLFeat features (green) and CABOX features (red); the center of the circle represents the location while the radius represents the scale. This experiment shows that CABOX detects a substantial amount of blobs detected by the DoG, while introducing a small fraction of new features.

#### 4.4. Pairwise image matching experiment

We also present the application of CABOX in an end-to-end pairwise image matching system. The system determines whether two given images are related or not i.e. when two images depict the same object or scene then both images are considered a ‘match’ otherwise this pair is a ‘non-match’. We used the CDVS dataset [1] for this experiment. We ran the system using both feature detectors i.e. CABOX and VLFeat DoG detector, for comparing their matching performance. Descriptors for both types of detected points were generated using the same SIFT implementation.

We followed the pairwise matching procedure as described by the Test Model document of the ongoing MPEG CDVS standard [11], which produces a compact binary descriptor for an image. This compact descriptor comprises of: 1) a global descriptor based on the Fisher vector [12]; 2) several local SIFT descriptors that are quantized and compressed. For the pairwise matching task, given the CDVS descriptors for two images, first the binary descriptors are decoded. Then the global descriptors are compared using a weighted Hamming distance and a matching score is computed. Subsequently, the compressed SIFT descriptors are matched using an L1 norm distance and an estimate of the number of correct feature matches is calculated. Finally, the system predicts that the image pair is a ‘match’ or a ‘non-match’ based on the matching score obtained from the global descriptor comparison and the estimated number of SIFT descriptor matches.

We show the true positive rates (tpr) for this experiment

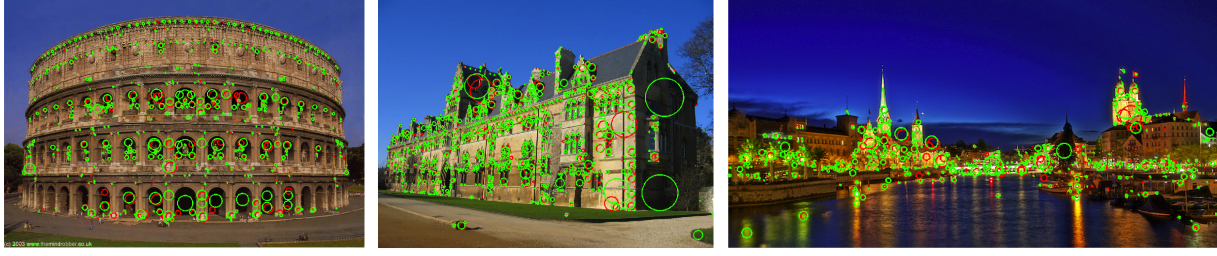


Figure 7: Blob detection using DoG implemented by VLFeat (green circles) and CABOX (red circles). Of those features detected with CABOX, 89 % are also detected by VLFeat while the remaining 11 % correspond to new features.

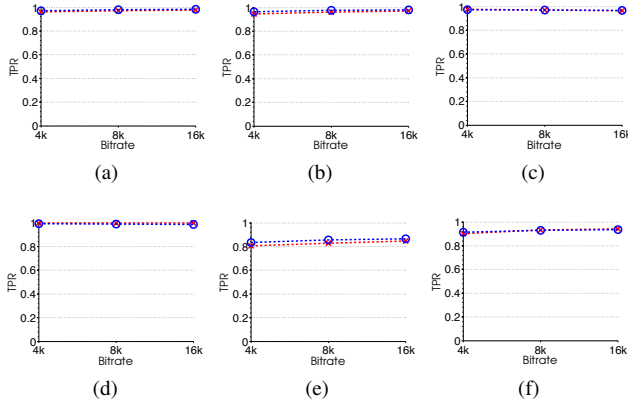


Figure 8: True positive rates (tpr) as a function of CDVS descriptor size. False positive rate is approximately 1 %. CABOX performs very competitively to DoG. Datasets: **8a** Mixed text and graphics; **8b** Mixed text and graphics (VGA + heavy JPEG compression); **8c** Paintings; **8d** Video frames; **8e** Buildings and landmarks; and **8f** Common objects.

in Fig. 8, where a true positive means that the system correctly predicted a 'match'. The decision parameters for the experiment were chosen in such a manner that fixes the false positive rate  $< 1\%$ . CABOX (red curve) achieves high a tpr across all the datasets and performs competitively in comparison with the reference DoG method (blue curve). However, CABOX computed approximately 44 % faster due to the fast scale space construction.

## 5. Conclusions and Future Work

We have introduced CABOX for approximating the Gaussian scale space. The box filters are computed by solving a constrained regularized least-squares problem, providing a good approximation of the scale space and use very few box filters to reduce computational complexity.

Our experiments showed that CABOX can reduce the scale space construction time by approximately 44% while maintaining a low average mean square error. Moreover, our pairwise image matching experiment demonstrated that CABOX performs competitively in comparison with the DoG feature detector. We believe that CABOX can be used for applications needing a fast yet robust feature detector.

We plan for future work a parallel implementation of CABOX for mobile devices, as the convolution of every box filter can be computed independently, thus speeding up the scale space construction.

## References

- [1] MPEG CDVS Benchmark Dataset. <http://pacific.tilab.com/Dataset-20120210/>. **1, 4, 4.4**
- [2] Oxford Region Covariance Detector Dataset. <http://www.robots.ox.ac.uk/~vgg/research/affine/>. **1**
- [3] M. Agrawal, K. Konolige, and M. R. Blas. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In *Proc. European Conf. on Computer Vision*, 2008. **1, 2**
- [4] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *Proc. IEEE Int'l. Symposium on Mixed and Augmented Reality*, 2009. **1**
- [5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (surf). *CVIU*, 110(3):346–359, June 2008. **1, 2**
- [6] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a cloudless day. In *ECCV*, 2010. **1**
- [7] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking. *IJCV*, 94(3):335–360, Sept. 2011. **1**
- [8] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Proc. Asian Conf. on Computer Vision*, 2006. **1, 2**
- [9] M. Hussein, F. Porikli, and L. Davis. Kernel integral images: A framework for fast non-uniform filtering. In *CVPR 2008*. **2**
- [10] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60:91–110, 2004. **1, 2, 3.2, 3.2**
- [11] D. Pau, G. Cordara, M. Bober, S. Paschalakis, K. Iwamoto, G. Francini, V. Chandrasekhar, and G. Takacs. White paper on compact descriptors for visual search, 2013. <http://mpeg.chiariglione.org/standards/mpeg-7/compact-descriptors-visual-search>. **1, 4.4**
- [12] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. **4.4**
- [13] B. R. Pires, K. Singh, and J. Moura. Approximating image filters with box filters. In *ICIP*, 2011. **2**
- [14] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *ECCV*, May 2006. **1**
- [15] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. **3.1**
- [16] A. Vedaldi and B. Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In *Proc. of the Int'l. Conf. on Multimedia*, 2010. **3.2, 4.1**