GPU-ACCELERATED ONE-CLASS SVM FOR EXPLORATION OF REMOTE SENSING DATA

Fabien Giannesini, Bertrand Le Saux

Onera - The French Aerospace Lab
F-91761 Palaiseau, France
bertrand.le_saux@onera.fr

ABSTRACT

We present a machine-learning based method for the exploration of remote sensing data. Our framework mixes an intuitive interface and a one-class support-vector machine to look for rare patterns in satellite images. It benefits from a fast implementation on the Graphics Process Unit that allows reasonable times for system-user interactions. We validate our approach with ground-truth experiments and demonstrate the method on real-world datasets. We achieve faster computations when compared with sequential implementations of the same methods (up to 80 times faster for feature extraction) and with other classification methods (such as local distribution comparison).

Index Terms— Remote sensing, Machine learning, Support vector machines, Image classification, Parallel programming

1. INTRODUCTION

As the number and the resolution of satellite images increase, the work of image analysts becomes more and more tedious. A large image (greater than 100 MPixels) cannot be entirely visualized at full resolution and must be inspected area by area. They need tools to accelerate this process and reduce the search area to the most promising regions.

Designing filters and classifiers that detect previously defined objects is a standard manner to solving this problem. Among these approaches, the One-Class Support Vector Machine (OC-SVM) recently became a popular way to build such detectors [1, 2, 3] since it only requires examples of the searched class, unlike the standard SVM that also requires negative examples.

Now, with today's level of detail, it becomes difficult to maintain a wide range of filters for all the possible objects that interest the various users. This can be linked to the page zero problem in information retrieval [4]: how does one start his exploration of an unknown dataset ? Multi-class classification is a possible answer [5], but it does not take into account what the user has in mind. An interactive framework better suits this purpose: for example in [6], the user defines his query by giving positive and negative examples in the image.

Our approach is also interactive, but intead of looking for positive examples (that are often scarce: camps in deserted areas, man-made structures in the countryside, boats in coastal waters, etc.), we propose to interactively define what the user is not looking for. OC-SVM is then used for anomaly detection, unlike previous systems. Online processing is made possible by a novel implementation using the computational power of Graphics Process Units (GPUs) to accelerate both feature extraction and classification.

In the rest of the paper, the feature extraction and classification parts of our approach are described in section 2. We show experimental results on real data in section 3 and give concluding remarks in section 4.

2. ONE-CLASS SVM EXPLORATION

2.1. Fast Feature Extraction

First the system computes features that characterize the local appearance of the image: phase, energy and orientation. Local orientation encodes the geometric information of the signal, whereas energy represents the information about the local luminance and contrast. Phase can be used to differentiate between diverse local structures independently of light or scale change.

These features are low-level primitives that can be extracted using convolution-based operations with spatial filters, like Gabor or Gaussian derivatives. They have been used in many remote sensing tasks such as edge detection or image segmentation [7]. Diaz showed that Gaussian derivatives, which are based on complex steering filters, are particularly suitable for a parallel implementation [8]. A steering filter is a particular class of filters such that a filter of arbitrary orientation can be synthesized as a linear combination of a set of basis filters [9], i.e. for a complex filtering at orientation θ:

$$\begin{aligned}(I * h_\theta) &= c_\theta(x,y) + j \cdot s_\theta(x,y) \\ &= \sum_\phi k_\phi(\theta) G_\phi(x,y) + j \cdot \sum_\phi l_\phi(\theta) H_\phi(x,y)\end{aligned} \quad (1)$$

where $G_\phi(x,y)$ denotes gaussian derivative filters and $H_\phi(x,y)$ the Hilbert transforms of gaussian derivatives. Thus, we compute filter outputs for $\theta = k * \pi/N, 0 \leq k \leq N, (N = 8)$ and

get the following feature values:

$$\theta_0 = \frac{1}{2}\arg\Big(\sum_\theta \frac{4}{3}\sqrt{s_\theta^2 + c_\theta^2}\exp(2\theta)\Big) \quad (2)$$

$$E_0 = \frac{1}{N}\sum_\theta s_\theta^2 + c_\theta^2 \quad (3)$$

$$\psi_0 = \arctan(s/c) \quad (4)$$

$$\text{with :}$$

$$s = \sum_\theta s_\theta \,\mathrm{sign}(\cos(\theta - \theta_0))\cos^2(\theta - \theta_0)$$

$$c = \sum_\theta c_\theta \cos^2(\theta - \theta_0)$$

Feature extraction can be hugely accelerated by a parallel processing of the image pixels. Diaz'tricks for parallelizing the computations are also usable on GPU: we avoid square-root computations by replacing amplitude by energy and avoid cross-dependencies by approximating the formulas. The features are finally implemented using:

$$\theta_0 = \frac{1}{2}\arg\Big(\sum_\theta \frac{4}{3}(s_\theta^2 + c_\theta^2)\exp(2\theta)\Big)$$

$$E_0 = \frac{1}{N}\sum_\theta s_\theta^2 + c_\theta^2$$

$$\psi_0 = \arctan\left(\frac{s}{c}\right) \text{ with } s = \sum_\theta c_\theta \text{ and } c = \sum_\theta c_\theta$$

2.2. One-class SVM Detection

Given an image and its local descriptors, our system processes online classification as follows. The user defines a region of low-interest by selecting areas using our GIS-like system ParadisSAT. The system then uses the features computed on these areas as training data for OC-SVM and learns the distribution model of the appearance of the image in the given area. The resulting classifier is then applied to the whole image, and regions that do not follow this distribution can be considered as anomalous, so are highlighted for exploration by a human image analyst.

One-Class SVM [10] is an adaptation of the standard two-class Support-Vector Machine (SVM) to the one-class classification problem. It has notably been used for document classification and retrieval [11]. In a nutshell, it uses all given samples for forming the first training class and treats the origin as the only member of the second class. Then, after transforming the features with a kernel, usual SVM techniques are employed.

Using the LIBSVM implementation [12] of OC-SVM, we identified the time-consuming functions of the algorithm. Scalar-product computations, though a fast operation *per se*, become the main bottleneck when frequently executed. This is primarily true during the classification of the (large) whole image, while the relatively small areas used for training ensure reasonable training times. We propose a fast classification step using the GPU.

Let $\Phi : X \to H$ be the kernel map which transforms the feature samples in a high-dimension space, and introduce the kernel function $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. The classification of an unknown pixel x is then performed using the prediction rule defined by the support vectors x_j:

$$\forall x,\ f(x) = \mathrm{sign}\Big(\sum_j (w_j \cdot k(x, x_j)) - \rho\Big) \quad (5)$$

where the weights w_j and the bias ρ result from the SVM optimization. The trick for acceleration is to invert the sum over the pixels and the sum over the support vectors, of which there are far fewer. Thus, we compute kernel products with one support vector in parallel for all pixels, then sum up the results to obtain the classification map.

3. EXPERIMENTS AND RESULTS

3.1. Acceleration of computation times on large images

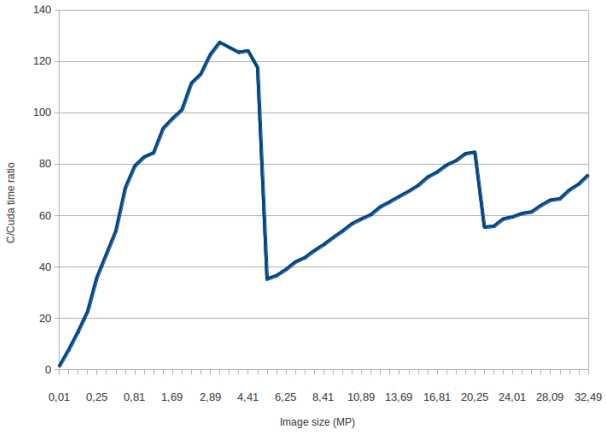
Fig. 1 shows the ratio between computation times on the standard sequential processor and after parallelization on GPU. Times are dramatically reduced by using the GPU implementation: by a factor 80 on average for feature extraction, and by a factor 6 for SVM classification, depending on the image size. For a 25-MPixel image, feature generation takes 4s and classification takes around 3min on GPU, while 4min and 20min respectively on CPU.

The various plateaux on the graphs emphasize a last bottleneck: the maximum number of pixels that can be loaded simultaneously on the GPU memory of size $w_{GPU} * h_{GPU}$. If the image size $w_{im} * h_{im}$ exceeds this limit, the image has to be divided in $((w_{im} \% w_{GPU} + 1) * (h_{im} \% h_{GPU} + 1) + 1)$ areas that have to be loaded sequentially.

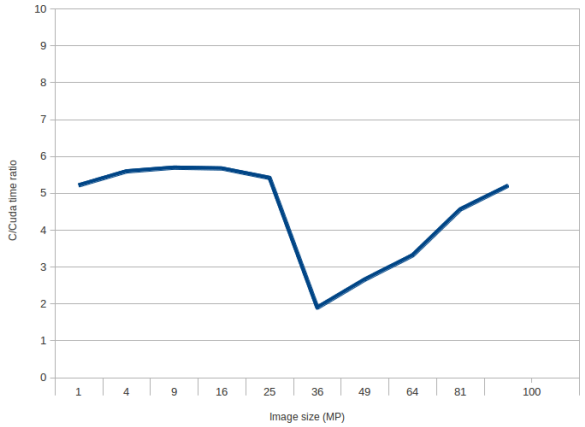
3.2. Comparison of classification approaches.

For comparison's sake, we also implemented on GPU a standard classification method that compares the local appearance distribution of the image to the user-defined area appearance. More precisely, our baseline is as follows:

- Image features are quantized into a few homogeneous clusters by the *k-means* algorithm.
- Distribution of features in the user-selected area $\mathcal{A}$ is estimated by a histogram $P_\mathcal{A}$ computed with respect to the adapted quantization.
- The whole image is classified by:
  - moving a sliding window $\mathcal{W}$ over it;
  - estimating the local quantized histogram $P_\mathcal{W}$;



(a) Feature Extraction



(b) One-Class SVM

Fig. 1. Ratios of computational times using either standard CPU implementation in C or GPU implementation in CUDA for (a) feature extraction and (b) classification using OC-SVM.

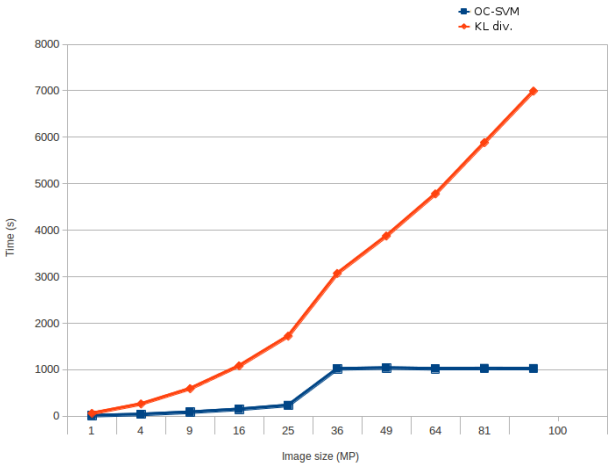


Fig. 2. Time comparison between OC-SVM and standard Kullback-Leibler divergence between local appearance distributions (KL div.).

– comparing it to the user area using the Kullback-Leibler divergence:

$$D_{KL}(P_{\mathcal{A}} \| P_{\mathcal{W}}) = \sum_i P_{\mathcal{A}}(i) \ln \left(\frac{P_{\mathcal{A}}(i)}{P_{\mathcal{W}}(i)} \right)$$

Fig. 2 shows GPU-computation times of both methods for various image sizes: OC-SVM is 3 to 8 times faster than the appearance-distribution comparison. Moreover, the plateau over 36 MPixels shows it is not limited by the sliding-window approach and fully benefits from GPU-parallelization.

3.3. Detection results

True urban areas	False non-urban areas	False urban areas
77.62%	7.76%	14.62%

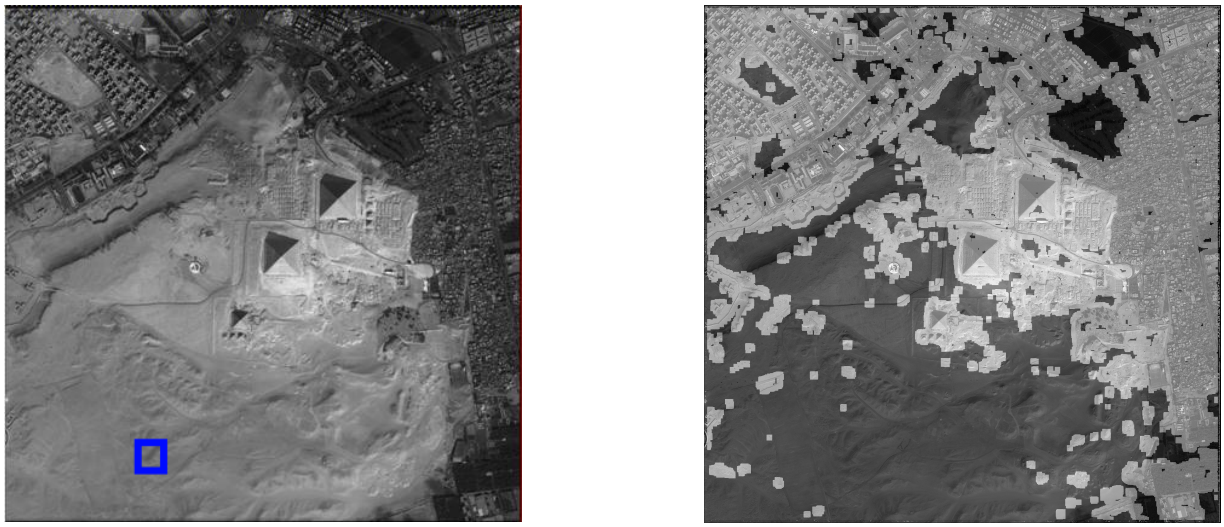
Table 1. Classification rates for urban area detection.

Fig. 3 shows an urban area detection scenario on a $4,456 * 4,465$ QuickBird image. Even with a small training area, the system is able to learn statistics on the whole image and retrieve the urban areas. In particular the public gardens and non-built-up areas inside the town are correctly classified as non-structured areas. We manually annotated ground-truth on this image. Table 1 shows our approach obtains a good classification rate of near 80% with few missed areas.

4. CONCLUSION

In order to provide image analysts with tools that help them explore remote-sensing data, human-centered mechanisms are required. With this in mind, we presented a framework that uses a One-Class Support-Vector Machine for learning and classifying image areas. Unlike previous approaches, it does not apply classifiers trained offline to detect specific objects of interest. It learns online what is not interesting, and focuses the attention on the potentially interesting areas. Fast interaction times are obtained by GPU-processing.

Our approach is promising since it makes the most of OC-SVM (that can be tuned for avoiding missed detections) for parsing large volumes of data, and relies on human aptitudes for a fine interpretation.



(a) Original (b) Urban-area detection

Fig. 3. Urban-area detection process. (a) The original QuickBird image and the selected area of non-interest (blue square). (b) Detection results (white mask).

5. AKNOWLEDGEMENTS

The authors thank S. Herbin for helpful ideas and conversations during this work.

6. REFERENCES

[1] Q. Guo, G. Dua, Y. Liu, and D. Liu, "Integrating object-based classification with one-class support vector machines in mapping a specific land class from high spatial resolution images," in *Proc. of XXXVII ISPRS Congress*, Beijing, China, 2008, vol. 27, pp. 1159–1163.

[2] J. Munoz-Mari, F. Bovolo, L. Gomez-Chova, L. Bruzzone, and G. Camps-Valls, "Semisupervised one-class support vector machines for classification of remote sensing data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 8, pp. 3188–3197, August 2010.

[3] X. Song, G. Fan, and M. Rao, "Svm-based data editing for enhanced one-class classification of remotely sensed imagery," *IEEE Geosci. Remote Sens. Letters*, vol. 5, no. 2, pp. 189–193–, April 2008.

[4] B. Le Saux and N. Boujemaa, "Unsupervised robust clustering for image database categorization," in *IEEE-IAPR International Conference on Pattern Recognition*, Quebec, Canada, august 2002.

[5] M. Molinier, J. Laaksonen, and T. Häme, "Detecting man-made structures and changes in satellite images with a content-based information retrieval system built on self-organizing maps," *IEEE Trans. on Geosci. Remote Sens.*, vol. 45, no. 4, pp. 861–874, April 2007.

[6] M. Schröder, H. Rehrauer, K. Seidel, and M. Datcu, "Interactive learning and probabilistic retrieval in remote sensing image archives," *IEEE Trans. Geosci. Remote Sens.*, vol. 38, no. 5, pp. 2288–2298, May 2000.

[7] A. Baraldi and F. Parmigiani, "Segmentation of sar images by means of gabor filters working at different spatial resolutions," in *Proceedings of International Geoscience And Remote Sensing Symposium*, Lincoln, Nebraska, 1996.

[8] J. Diaz, E. Ros, S. Mota, and R. Carrillo, "Local image phase, energy and orientation extraction using fgpas," *Int. J. of Electronics*, 2008.

[9] W. Freeman and E. Adelson, "The design and use of steerable filters," *IEEE Trans. on Pattern Analysis and Mach. Int.*, vol. 13, no. 9, pp. 891–906, 1991.

[10] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[11] L. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of Machine Learning Research*, vol. 2, pp. 139–154, 2002.

[12] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Trans. on Intell. Syst. and Technol.*, vol. 2, pp. 1–27, 2011.