Indoor Place Recognition using Online Independent Support Vector Machines

Francesco Orabona, Claudio Castellini
LIRA-Lab, University of Genova
Genova, Italy
{bremen,drwho}@liralab.it

Barbara Caputo, Jie Luo
IDIAP Research Institute / EPFL
Martigny, Switzerland
{bcaputo,jluo}@idiap.ch

Giulio Sandini
Italian Institute of Technology
Genova, Italy
giulio.sandini@iit.it

Abstract

In the framework of indoor mobile robotics, *place recognition* is a challenging task, where it is crucial that self-localization be enforced precisely, notwithstanding the changing conditions of illumination, objects being shifted around and/or people affecting the appearance of the scene. In this scenario online learning seems the main way out, thanks to the possibility of adapting to changes in a smart and flexible way. Nevertheless, standard machine learning approaches usually suffer when confronted with massive amounts of data and when asked to work online. Online learning requires a high training and testing speed, all the more in place recognition, where a continuous flow of data comes from one or more cameras. In this paper we follow the Support Vector Machines-based approach of Pronobis et al. [26], proposing an improvement that we call Online Independent Support Vector Machines. This technique exploits linear independence in the image feature space to incrementally keep the size of the learning machine remarkably small while retaining the accuracy of a standard machine. Since the training and testing time crucially depend on the size of the machine, this solves the above stated problems. Our experimental results prove the effectiveness of the approach.

1 Introduction

Place recognition is an open and highly challenging problem in computer vision, especially when applied to mobile robotics in indoor environments. Simply stated, the problem is that of determining what room of a house or office a mobile robot is in, based upon what the robot sees through one or more cameras. The problem is made very difficult by at least three factors: (a) the input space is huge, since we deal with images, usually at a reasonable resolution and in colour; (b) images of the same place can be quite different as illumination conditions change and moving obstacles get in the way; and (c), recognition must be done on-line in real time, as the robot is moving around. The topic is widely researched, but incremental learning approaches have been so far mostly used for constructing the geometrical map, or the environment representation, online [6, 1]. Robustness to illumination changes, and more generally to realistic visual variations in time, has been addressed in [26], where it was shown that a pure learning approach can be very effective for tackling the first two issues: indeed it was demonstrated that an approach based upon Support Vector Machines (SVM, see, e.g., [4]) *in batch mode* could achieve a remarkable robustness to illumination changes and variability due to the normal use of the environments. At the same time the work elicited the problem of the growth of the testing time when a bigger training set was used, to have better recognition performances. In fact, as far as the third issue is concerned, it is well known that both the training and testing time of an SVM crucially depend on the number of samples considered [16]; as well, the number of Support Vectors (SVs) found, which determine the complexity of the solution to the problem, grows proportionally with respect to the number of samples [28]. This makes the approach unsuitable, at least so far, for on-line learning, where a potentially endless flow of data is acquired by the machine. SVMs can be up to 50 times slower than other specialized approaches with similar performances [8]. Several exact and approximate approaches have been proposed so far for simplifying the SVM decision function: see for instance [12], based upon linear independence of the SVs in the feature space performed after training, and other after-training simplification methods (e.g. chapter 18.3 in [27] and [23]). The exact solution to online SVM learning was given by Cauwenberghs and Poggio in 2000 [9], but their algorithm cannot be used to reduce the number of SVs. In [29] and [25] approximate incremental versions of the SVM are proposed, that also achieve a reduction of the number of SVs with small degradation of their performances.

In this paper we propose an improvement to SVMs that we call Online Independent Support Vector Machines (OISVMs). OISVMs incrementally select "basis vectors" that are used to build the solution of the SVM training problem, based upon *linear independence in the feature space*: vectors which are linearly dependent on already stored ones are rejected, and a smart, incremental minimization algorithm is employed to find the new minimum of the cost function. This keeps the number of SVs much smaller than usual, reducing the complexity of the solution and therefore both the training and testing time. Unsupervised rank reduction methods have been proposed [3] as well as supervised ones [2] that achieve the same goals, but no application of these ideas appears so far, to the best of our knowledge, in online settings. This is particularly important since in an online setting the size of a SVM would grow indefinitely, and so would the testing time. Our experiments instead indicate that the number of basis vectors of OISVMs does not grow linearly with the training set, but reaches a limit and then stops growing. This result is theoretically confirmed, e.g., in [14], even in the case the feature space is infinite-dimensional.

Such an approach is actually what is needed to tackle the problem of place recognition in mobile robotics. To support this claim, we show a set of experimental results obtained by comparing SVMs and OISVMs on a real-world place recognition problem in an indoor environment. Data images are acquired continuously by two robot platforms under different weather conditions and across a time span of several months. Our results show that our method achieves a speed-up of $3.5 - 2.3$ times with respect to the time required by the standard SVM, respectively with χ^2 kernel and matching kernel [30], while retaining essentially the same accuracy.

The paper is structured as follows: after an overview of background mathematics proper to SVMs, Section 3 describes OISVMs. Section 4 shows the experimental results and lastly, in Section 5, conclusions are drawn and future work is outlined.

2 Background Mathematics

Due to space limitations, this is a very quick account of SVMs — the interested reader is referred to [7] for a tutorial, and to [11] for a comprehensive introduction to the subject. Assume $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$, is a set of samples and labels drawn from an unknown probability distribution; we want to find a function $f(\mathbf{x})$ such that $sign(f(\mathbf{x}))$ best determines the category of any future sample $\mathbf{x}$. In the most general setting,

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{1}$$

where $b \in \mathbb{R}$ and $K(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2)$, the *kernel function*, evaluates inner products between images of the samples through a non-linear mapping Φ. The α_is are Lagrangian coefficients obtained by solving (the dual Lagrangian form of) the problem

$$\begin{array}{rl} \min\limits_{\mathbf{w},b} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l} \xi_i^p \\ \text{subject to} & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{array} \tag{2}$$

where $\mathbf{w}$ defines a separating hyperplane in the *feature space*, i.e., the space where Φ lives, whereas $\xi_i \in \mathbb{R}$ are slack variables, $C \in \mathbb{R}^+$ is an error penalty coefficient and p is usually 1 or 2. In practice, most of the α_i are found to be zero after training; the vectors with an associated α_i different from zero are called *support vectors*. Notice that, from (1), the testing time of a new point is proportional to the number of SVs, hence reducing the number of SVs implies reducing the testing time.

In the following, the term *kernel dimension* will refer, as is customary, to the dimension of the feature space. The kernel dimension is related to the generalization power of the machine, and it depends on the choice of the kernel itself. Widely used kernels include the *polynomial* one (finite-dimensional) and the *Gaussian* one (infinite-dimensional).

3 Online Independent Support Vector Machines

Let the *kernel matrix* K be defined such that $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, with $i, j = 1, \ldots, l$. The possibility to obtain a more compact representation of $f(\mathbf{x})$ follows from the fact that the solution to a SVM problem (that is, the α_is) is not unique if K does not have full rank [7], which is equivalent to some of the SVs being linearly dependent on some others *in the feature space* (this is the core of Downs et al.'s [12] original idea). In an online setting, to apply Downs et al.'s idea, or any other post-training method to reduce the number of SVs, means to simplify the solution each time a new sample is acquired, which is obviously infeasible. We need a way to use independent SVs only, that is to decouple the concept of "basis" vectors, used to build the classification function (1), from the samples used to evaluate the ξ_i in (2). If the selected basis vectors span the same subspace as the whole sample set, the solution found will be equivalent — that is, we will not lose any precision.

We hereby propose, after having received a new training sample, to incrementally add it to the basis if it is linearly independent in the feature space from those already present in the basis itself. The solution found is *the same* as in the classical SVM formulation; therefore, no approximation whatsoever is involved, unless one gives it up in order to obtain even fewer support vectors (see Section 4 for a deeper discussion on this point).

Denoting the indexes of the vectors in the current basis, after l training samples, by $\mathscr{B}$, and the new sample under judgment by $\mathbf{x}_{l+1}$, the algorithm can then be summed up as follows:

1. check whether $\mathbf{x}_{l+1}$ is linearly independent from the basis in the feature space; if it is, add it to $\mathscr{B}$; otherwise, leave $\mathscr{B}$ unchanged.
2. incrementally re-train the machine.

Hence the testing time for a new point will be $O(|\mathscr{B}|)$, as opposed to $O(l)$ in the standard approach; therefore, keeping $\mathscr{B}$ small will improve the testing time without losing any precision whatsoever.

In the following, the notations A_{IJ} and $\mathbf{v}_I$, where A is a matrix, $\mathbf{v}$ is a vector and $I, J \subset \mathbb{N}$ denote in turn the sub-matrix and the sub-vector obtained from A and $\mathbf{v}$ by taking the indexes in I and J.

3.1 Linear independence

In general, checking whether a matrix has full rank is done via some decomposition, or by looking at the eigenvalues of the matrix; but here we want to check whether a *single* vector is linearly independent from a matrix which is already known to be full-rank. Inspired by the definition of linear independence, we check how well the vector can be approximated by a linear combination of the vectors in the set [13]. Let $d_j \in \mathbb{R}$; then let

$$\Delta = \min_{\mathbf{d}} \left\| \sum_{j \in \mathscr{B}} d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{l+1}) \right\|^2 \tag{3}$$

If $\Delta > 0$ then $\mathbf{x}_{l+1}$ is linearly independent with respect to the basis, and $\{l+1\}$ is added to $\mathscr{B}$. In practice, we check whether $\Delta \leq \eta$ where $\eta > 0$ is a tolerance factor, and expect that larger values of η lead to worse accuracy, but also to smaller bases. As a matter of fact, if η is set at machine precision, OISVMs retain the exact accuracy of SVMs. Notice also that if the feature space has finite dimension n, then no more than n linearly independent vectors can be found, and $\mathscr{B}$ will never contain more than n vectors.

Expanding equation (3) we get

$$\Delta = \min_{\mathbf{d}} \left(\sum_{i,j \in \mathscr{B}} d_j d_i \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_i) - 2 \sum_{j \in \mathscr{B}} d_j \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_{l+1}) + \phi(\mathbf{x}_{l+1}) \cdot \phi(\mathbf{x}_{l+1}) \right) \tag{4}$$

that is, applying the kernel trick,

$$\Delta = \min_{\mathbf{d}} \left(\mathbf{d}^T K_{\mathscr{B}\mathscr{B}} \mathbf{d} - 2\mathbf{d}^T \mathbf{k} + K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) \right) \tag{5}$$

where $k_i = K(\mathbf{x}_i, \mathbf{x}_{l+1})$ with $i \in \mathscr{B}$. Solving (5), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\tilde{\mathbf{d}} = K_{\mathscr{B}\mathscr{B}}^{-1}\mathbf{k} \tag{6}$$

and, by replacing (6) in (5) once,

$$\Delta = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \mathbf{k}^T\tilde{\mathbf{d}} \tag{7}$$

Note that $K_{\mathscr{B}\mathscr{B}}$ can be safely inverted since, by incremental construction, it is full-rank. An efficient way to do it, exploiting the incremental nature of the approach, is that of updating it recursively: after the addition of a new sample, the new $K_{\mathscr{B}\mathscr{B}}^{-1}$ then becomes

$$\begin{bmatrix} & & & 0 \\ & K_{\mathscr{B}\mathscr{B}}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\Delta}\begin{bmatrix} \tilde{\mathbf{d}} \\ -1 \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{d}}^T & -1 \end{bmatrix} \tag{8}$$

where $\tilde{\mathbf{d}}$ and Δ are already evaluated during the test (this method matches the one used in Cauwenberghs and Poggio's incremental algorithm [9]). Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathscr{B}|^2)$, as one can easily see from Equation (6).

With this method we are approximating the original kernel matrix K with another matrix $\widehat{K}$ [2]; the quality of the approximation depends on η. In fact it is possible to show that $trace(K - \widehat{K}) \leq \eta|\mathscr{B}| \leq \eta l$, where l is the number of samples acquired [14]. If we consider a normalized kernel, that is a kernel for which $K(x,x)$ is always equal to 1, we can write $trace(K - \widehat{K})/trace(K) \leq \eta$. On the other hand a bigger η means of course a smaller number of SVs, hence it controls the trade-off between accuracy and speed of the OISVM.

3.2 Training the machine

The training method largely follows Keerthi et al. [17, 16], that we have adapted for online training. The algorithm directly minimizes problem (2) as opposed to the standard way of minimizing its dual Lagrangian form, allowing to select explicitly the basis vectors to use. We set $p = 2$ in (2) and transform it to an unconstrained problem. Let $\mathscr{D} \subset \{1, \ldots, l\}$; then the unconstrained problem is

$$\min_{\beta}\left(\frac{1}{2}\beta^T K_{\mathscr{D}\mathscr{D}}\beta + \frac{1}{2}C\sum_{i=1}^{l} max(0, 1 - y_i K_{i\mathscr{D}}\beta)^2\right) \tag{9}$$

where β is the vector of the Lagrangian coefficients involved in $f(\mathbf{x})$, analogously to the α_is in the original formulation. If we set $\mathscr{D} = \mathscr{B}$, then the solution to the problem is unique since $K_{\mathscr{B}\mathscr{B}}$ is full rank by construction. Newton's method as modified by Keerthi et al. [17, 16] can then be used to solve (9) after each new sample. When the new sample $\mathbf{x}_{l+1}$ is received the method goes as follows:

1. let $\mathscr{I} = \{i : 1 - y_i o_i > 0\}$ where $o_i = K_{i\mathscr{B}}\beta$ and β is the vector of optimal coefficients with l training samples; if $\mathscr{I}$ has not changed, stop.

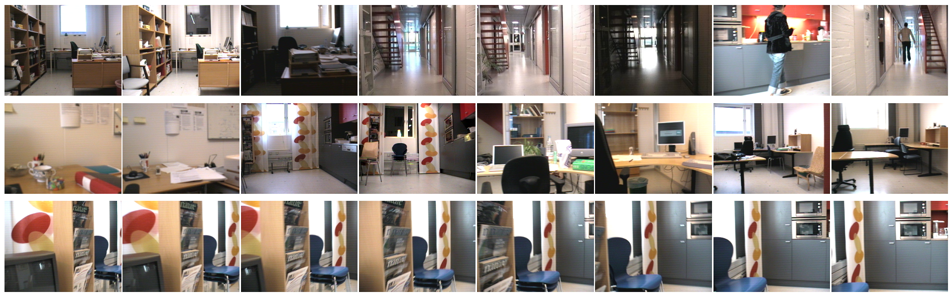


Figure 1: Sample images illustrating the variations in the IDOL2 database. Images in the top row show the variability introduced by changes in illumination as well as people appearing in the environment. The middle row shows the influence of people's everyday activity (first four images) as well as larger variations which happened over a time span of 6 months. Finally, the bottom row illustrates the changes in viewpoint observed for a series of images acquired one after another in 1.6 seconds.

2. otherwise, let the new β be $\beta - \gamma \mathbf{P}^{-1}\mathbf{g}$, where $\mathbf{P} = K_{\mathscr{B}\mathscr{B}} + CK_{\mathscr{B}\mathscr{I}}K_{\mathscr{B}\mathscr{I}}^T$ and $\mathbf{g} = K_{\mathscr{B}\mathscr{B}}\beta - CK_{\mathscr{B}\mathscr{I}}(\mathbf{y}_{\mathscr{I}} - \mathbf{o}_{\mathscr{I}})$.

3. go back to Step 1.

In Step 2 above, γ is set to one. In order to speed up the algorithm, we maintain an updated Cholesky decomposition of $\mathbf{P}$. It turns out that the algorithm converges in very few iterations, usually 0 to 2; the time complexity of the re-training step is $O(|\mathscr{B}|l)$, as well as its space complexity; hence, keeping $\mathscr{B}$ small will speed up the training time as well as the testing time.

4 Place Recognition via OISVMs

In this section we report the experimental evaluation of OISVMs on the place recognition scenario, where the aim is to update the model to handle variations in an indoor environment due to human activities over long time spans.

Experiments were conducted on the IDOL2 database (Image Database for rObot Localization 2, [21]), which contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms, while moving in an indoor laboratory environment consisting of five different rooms. The sequences were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. Thus, this data capture natural variability that occurs in real-world environments because of both natural changes in the illumination and human activities. Fig. 1 shows some sample images from the database, illustrating the difficulty of the task. The image sequences in the database are divided as follows: for each robot platform and for each type of illumination conditions, there were four sequences recorded. Of these four sequences, the first two were acquired six months before the last two. This means that, for each robot and for every illumination condition, there are always two sequences acquired under similar conditions, and two sequences acquired under very different conditions.

This makes the database suitable for different kinds of evaluation on the adaptability of an incremental algorithm. For further details about the database see [21].

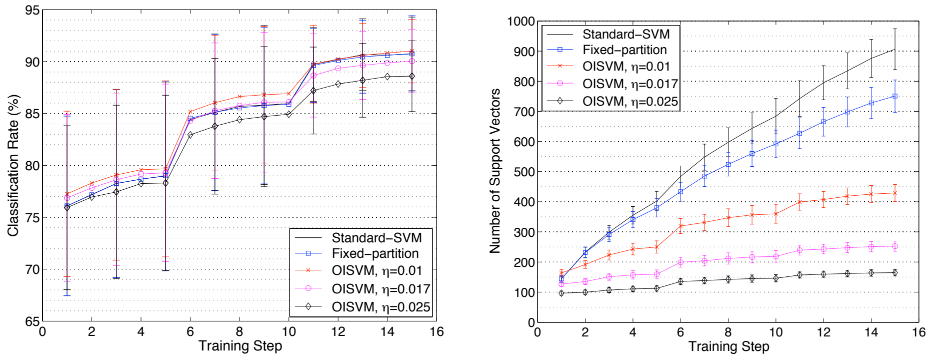
The evaluation was performed using Composed Receptive Field Histograms (CRFH) [19] as global image features and SIFT descriptors [20] of local features computed using a Harris-Laplace detector [15]. In the experiments, we consider both exponential χ^2 kernel for SVM (when use CRFH), and local kernels [30] (SIFT). Note the kernel in [30] is not always positive semidefinite [5], so this is also a test on non-Mercer kernels that have proved useful for visual recognition. The kernels used are infinite-dimensional, so for both kernels we run the OISVM using different values of η.

OISVMs have been implemented in Matlab and tested against LIBSVM v2.82 [10]. The software library has been extended to various families of kernels, and to the fixed-partition incremental SVM [29], an approximate incremental extension of SVM. In this way we can do a straightforward comparison between exact and approximate methods on this task. Notice that for the standard SVM the training is not online.

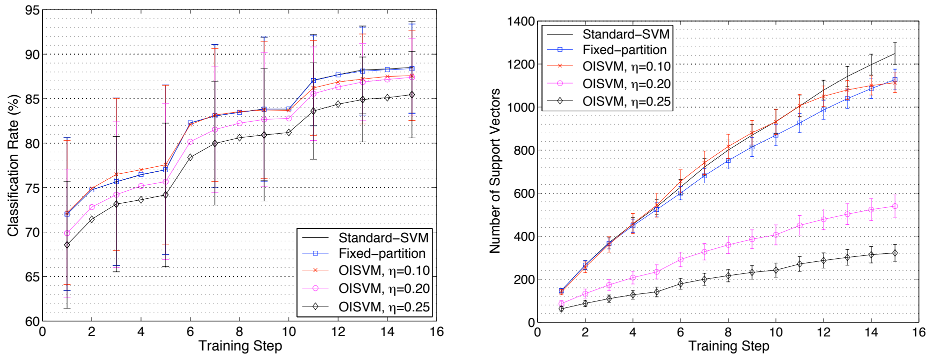
As in the experimental setup of [22], the algorithm was trained incrementally on three sequences from IDOL2, acquired under similar illumination conditions with the same robot platform; the fourth sequence was used for testing. In order to test the various properties of interest of the incremental algorithms, we need a reasonable number of incremental steps. Thus, every sequence was split into 5 subsequences, so that each subset contained one of the five images acquired by the robot every second (image sequences were acquired at a rate of 5fps). Since during acquisition the camera's viewpoint continuously changes [22], the subsequences could be considered as recorded separately in a static environment but for varying pose. This setup allows us to examine how the algorithms perform on data with less variations. In order to get a feeling of the variations of the frame images in a sequence, bottom row of Fig. 1 shows some sample images acquired within a time span of 1.6 sec. As a result, training on each sequence was performed in 5 steps, using one subsequence at a time, resulting in 15 steps in total. Overall, we considered 36 different permutations of training and test sequences for both the exponential χ^2 and matching kernels; here we report average results with standard deviation. Fig. 2, left, shows the recognition rates of the exponential χ^2 kernel (top) and matching kernel (bottom) experiments obtained at each step using OISVM, the fixed-partition algorithm and the standard SVM. Fig. 2, right, reports the number of support vectors stored in the model at each step of the incremental procedure, for both kernel types.

We see that, performance-wise, all methods achieves statistically comparable results; this is true for both kernel types. As far the machine size is concerned, the OISVM algorithm shows a considerable advantage with respect to the fixed-partition method. In the case of the exponential χ^2 kernel this advantage is truly impressive (Fig 2, top right): for $\eta = 0.017$ and 0.025 the size at the final incremental step is $34\%/22\%$ of that of the fixed-partition method and $28\%/18\%$ of that of the standard batch method. Even more important, OISVM, for these two values of η, has found a plateau in memory, while for other methods the trend seems to be of a growth proportional to the number of training data. Note that the choice of the parameter η is crucial for achieving an optimal trade-off between compactness of the solution and optimal performance.

It is very interesting to note that, in the case of the matching kernel, the memory reduction for OISVM is less pronounced, and there is not a clear plateau in memory growth by any of the algorithms. This behavior might be due to several factors: to begin with, the matching kernel is not a Mercer kernel [5], which might affect the algorithm.



(a) Number of support vectors and classification rate obtained at each incremental step using χ^2 kernel.



(b) Number of support vectors and classification rate obtained at each incremental step using local kernel.

Figure 2: Average results obtained for experiment performed on the IDOL2 database, using OISVM with three different values of η, the fixed-partition and the standard SVM.

Also, the algorithm does not reach a plateau in the SVs growth because, in the induced space of the matching kernel, there seems to be a high probability that pair of training points are orthogonal, or almost orthogonal, to each other (notice that, as the kernel is not a Mercer one, the geometric interpretation might not be valid). Anyway, given enough training points, the machine will always reach a maximum size and will stop growing [14]. Other tests on a set of standard databases commonly used in the machine learning community, as well as more details about OISVM can be found in [24].

It is worth noting that, even if the solution is kept small and the number of support vectors will be finite in any case, all the received training samples must be stored. This can be a problem in an online setting, but it could be solved using, for example, some kind of forgetting strategy. Another strategy can be the use out-of-core storage of the data (i.e., storage on the hard disk) in order to be able to deal with big training sets.

5 Discussion and conclusions

In this paper we have shown a promising improvement to Support Vector Machines, that we call Online Independent Support Vector Machines (OISVM). OISVMs can effectively solve the problem of place recognition by a mobile robot, at least in the experiment we have shown. OISVMs were tested on the IDOL2 image database, which consists of image sequences acquired by two robot platforms under different weather conditions and across a time span of several months. OISVMs avoid using in the solution those support vectors which are linearly dependent of previous ones in the feature space. The optimization problem is solved via an incremental algorithm which benefits of the small number of the basis vectors.

As far as we know, this method is different from all analogous procedures presented so far in literature (e.g., [12, 23, 18, 31, 16]) since it is *not* an after-training simplification and it assumes *no knowledge whatsoever* of the full training set beforehand. Moreover in case of finite-dimensional kernel and $\eta = 0$, the solution is exactly the same of the standard formulation because no approximation is used.

Our experimental results show that in the case of infinite-dimensional kernels, the number of support vectors is dramatically reduced at the price of a negligible degradation in the accuracy. In fact in the case of χ^2 kernel, we get as few as 3.5 times less SVs with respect to the batch formulation and 3 times less with respect to the fixed-partition method, while retaining essentially the same accuracy. In the case of the local kernel, the speed up are respectively 2.3 and 2.1.

Since the training and testing time depend polynomially on the number of support vectors, reducing them brings an obvious speed up. A careful study of the relationship between η and the degradation in performance is being carried on; in fact, according to [14], imposing a value of η strictly larger than zero will eventually result in a *finite* number of basis vectors, *even in the case the feature space is infinite-dimensional*. Further research about finding a precise relationship between η and this number will allow us to precisely dimension the machine depending on the required precision.

Acknowledgments

This work was supported by EU projects RobotCub (IST-2004-004370), CONTACT (NEST-5010), NEURObotics (FP6-IST-001917) and DIRAC (FP6-0027787).

References

[1] M. Artač, M. Jogan, and A. Leonardis. Mobile robot localization using an incremental eigenspace model. In *Proceedings of ICRA'02*, 2002.

[2] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of ICML'05*, 2005.

[3] G. Baudat and Fatiha Anouar. Feature vector selection and projection using kernels. *Neurocomputing*, 55(1-2):21–38, 2003.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of COLT'92*, pages 144–152. ACM press, 1992.

[5] S. Boughorbel, J.-P. Tarel, and F. Fleuret. Non-mercer kernels for svm object recognition. In *Proceedings of BMVC'04*, pages 137–146, London, England, 2004.

[6] E. Brunskill and N. Roy. Slam using incremental probabilistic pca and dimensionality reduction. In *Proceedings of ICRA'05*, 2005.

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.

[8] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In *Proceedings of NIPS'96*, pages 375–381, 1996.

[9] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proceedings of NIPS'00*, pages 409–415, 2000.

[10] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods).* CUP, 2000.

[12] T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

[13] Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *Proceedings ECML'02*, 2002.

[14] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, 52(8), 2004.

[15] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of AVS'88*, 1988.

[16] S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 8:1–22, 2006.

[17] S. S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[18] Y. J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the SIAM International Conference on Data Mining*, 2001.

[19] O. Linde and T. Lindeberg. Object recognition using composed receptive field histograms of higher dimensionality. In *Proceedings of ICPR'04*, 2004.

[20] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of ICCV'99*, 1999.

[21] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. The KTH-IDOL2 database. Technical Report 304, KTH, CAS/CVAP, 2006. Available at `http://cogvis.nada.kth.se/IDOL2/`.

[22] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Accepted in IROS'07*, 2007.

[23] D. D. Nguyen and T. B. Ho. An efficient method for simplifying support vector machines. In *Proceedings of ICML'05*, pages 617–624, New York, NY, USA, 2005. ACM Press.

[24] F. Orabona. *Learning and Adptation in Computer Vision*. PhD thesis, University of Genoa, 2007.

[25] A. Pronobis and B. Caputo. The more you learn, the less you store: memory-controlled incremental support vector machines. In *Proceedings of ICVW'06*, Gratz, 2006.

[26] A. Pronobis, B. Caputo, P. Jensfelt, and H. I. Christensen. A discriminative approach to robust visual place recognition. In *Proceedings of IROS'06*, 2006.

[27] A. Smola and B. Schölkopf. *Learning with Kernels*. MIT press, Cambridge, MA, USA, 2002.

[28] I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003.

[29] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at IJCAI'99*, 1999.

[30] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proceedings of ICCV'03*, 2003.

[31] M. Wu, B. Schölkopf, and G. Bakir. A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624, 04 2006.