On-line Independent Support Vector Machines for Cognitive Systems

Francesco Orabona [a] Claudio Castellini [b]
Barbara Caputo [a c] Luo Jie [a c]
Giulio Sandini [d]

IDIAP–RR 07-63

November 2007

[a] IDIAP Research Institute, P.O.Box 592, 1920, Martigny, Switzerland
[b] University of Genova, viale F. Causa, 13, 16145 Genova, Italy
[c] Ecole Polytechnique Fdrale de Lausanne (EPFL), Switzerland
[d] Italian Institute of Technology, via Morego 30, 16163 Genova, Italy

IDIAP Research Report 07-63

On-line Independent Support Vector Machines for Cognitive Systems

Francesco Orabona Claudio Castellini Barbara Caputo Luo Jie
Giulio Sandini

November 2007

Abstract. Learning from experience and adapting to changing stimuli are fundamental capabilities for artificial cognitive systems. This calls for on-line learning methods able to achieve high accuracy while at the same time using limited computer power. Research on autonomous agents has been actively investigating these issues, mostly using probabilistic frameworks and within the context of navigation and learning by imitation. Still, recent results on robot localization have clearly pointed out the potential of discriminative classifiers for cognitive systems. In this paper we follow this approach and propose an on-line version of the Support Vector Machine (SVM) algorithm. Our method, that we call On-line Independent SVM, builds a solution on-line, achieving an excellent accuracy vs. compactness trade-off. In particular the size of the obtained solution is always bounded, implying a bounded testing time. At the same time, the algorithm converges to the optimal solution at each incremental step, as opposed to similar approaches where optimality is achieved in the limit of infinite number of training data. These statements are supported by experiments on standard benchmark databases as well as on two real-world applications, namely (a) place recognition by a mobile robot in an indoor environment, and (b) human grasping posture classification.

1 Introduction

On-line learning is the process by which a cognitive system learns continuously from experience, updating and enriching its internal models of the environment. This learning mechanism is the main reason why cognitive systems are capable of achieving a robust, yet flexible capability to react to novel stimuli. Realistic domains are highly dynamic and any autonomous system interacting with them, such as a robot, must be able to adapt to a number of changing parameters. For instance, *indoor visual place recognition* for robot localization suffers from the natural variability of environments in time (varying illumination conditions, objects moved around beause of daily use, rooms redecorated); *grasping by imitation* requires the ability to recognize a human subject's "style", and to adapt to different objects' shapes and affordances, and so forth.

Ideally, an on-line learning algorithm should satisfy four main requirements, namely:

1. *Adaptability.* The system must be robust to local changes in the environmental parameters. For instance, it should be able to recognize a grasping type even when performed by different subjects on unknown objects; it should be able to recognize an environment even after that some of the furniture has been relocated because of normal use, and so forth.

2. *Optimality.* Nevertheless, the ratio of correct recognition of the essential characteristics of the environment must not be affected. Autonomous systems acting in realistic settings, possibly shared by humans, must guarantee an optimal performance for each sensory channel, so to minimize mistakes.

3. *Limited resources.* All this must be done using limited computer power (CPU time, memory, etc.). Indeed, artificial cognitive systems are required to perform human-like tasks in every day scenarios. The complexity and richness of the stimuli to acquire and analyze, for each sensory channel, is in general very high. Decisions must be taken keeping into account all the available information, so to react and interact with the environment actively.

4. *Speed.* Lastly, the system must be able to adapt (training) and operate (testing) on-line, that is, quickly.

Some or all of the above issues have already been addressed in the past. Indeed, it has long been recognized in the autonomous system community that learning is a key feature for making cognitive systems capable of solving complex tasks in realistic environments [1]. Learning robots can operate robustly in unknown environments, can take advantage of knowledge from supervisors on-line and can compensate for changes by updating their internal representation about the environment and themselves.

In the field of robot navigation, on-line methods have been used for building topological maps and detect loop closure [2], to learn variability of environments due to illumination changes and natural dynamic of rooms [3], or for adaptive obstacle avoidance in dynamic indoor environments like corridors [4]. On-line methods have been applied both to indoor [2, 3] and outdoor navigation [5], mostly within a probabilistic framework [2, 5, 6]. With the notable exception of [3], all on-line approaches proposed so far have been tested on a very limited temporal domain, of few hours if not of few minutes. Thus, it is not clear if these methods are able to provide high accuracy, combined with controlled computing resources, in case of on-line learning across a time span of several months.

As far as grasping recognition is concerned, machine learning has never been applied on-line to the best of our knowledge, except in [7] where, however, *regression* is used to predict the grasping configuration. Batch approaches have been used to classify grasps, e.g., in [8] (using hidden Markov models), [9] (Gaussian mixtures) and [10] (neural networks). In [11], a comparison of classification systems for grasp recognition is presented, the main outcome of which being that (uncalibrated) human grasping data gathered from a CyberGlove can be used to an excellent extent to recognise the grasp type, although the performance can be heavily influenced by multiple user analysis. In all works examined, the emphasis is really on affordances [12] rather than on objects or hand configurations; in other words, one is generally interested in the types of grasps and what can be done using them.

Still, the use of discriminative classifiers like AdaBoost and Support Vector Machine has been gaining momentum in the last years, especially in the field of robot localization [13, 14]. How to use these methods in on-line settings is an open challenge: discriminative learning methods do have the ability of adapting to a changing environment (issue 1 above) via training. But they are normally used on batches of previously available data, which is unsatisfactory in our setting. On-line learning extensions of these algorithms have been proposed (e.g. [15]), but they either lack the capability of working with limited resources without losing too much accuracy (issues 2 and 3, e.g. [16]), or they suffer from unacceptably long training/testing times (issue 4, e.g., after training semplification [17]).

In this paper we apply Support Vector Machines (SVMs) to the problem of on-line learning for robotic systems, trying to address the above issues, all at a time. The initial choice of SVMs is mainly motivated by their assured optimality (no local minima in the cost functional to be minimized). It has also been shown that SVMs have a remarkable capacity of adaptation, since they have been successfully applied to such diverse fields as speech recognition, object classification and function approximation [18]. Lastly, the generalization power of SVMs is theoretically well founded, and the danger of data overfitting is sensibly smaller than with other approaches.

Still, SVMs require a long training time. An SVM can be up to 50 times slower than other specialized approaches with similar performances [19]. In fact, the time required by an SVM to train and predict is, in turn, cubic and linear in the number of support vectors [20]. As well, the number of support vectors found grows proportionally with respect to the number of samples [21]. This makes the approach unsuitable, in its current formulations, for on-line learning, where a potentially endless flow of data is acquired by the machine.

To solve this problem, and address issue 3 and 4 above with SVMs, we propose a new method called On-line Independent SVMs (OISVM), which builds a solution on-line, achieving an excellent accuracy vs. compactness trade-off in general, therefore keeping issue 2 in target. With our method the amount of memory required can be dramatically smaller than that of a SVM, while the accuracy of the obtained solution is almost the same. In particular the size of the solution, and the testing time are always bounded. This statement is supported by experiments on standard benchmark databases as well as on two real-world applications, namely (a) place recognition by a mobile robot in an indoor environment, and (b) human grasping posture classification.

Experiment (a) is concerned with a mobile robot moving around in an indoor environment subject to high variability due to human activities over long time spans. These variations include people appearing in different rooms during working time, objects such as cups moved or taken in/out of the drawers, pieces of furniture pushed around, and so forth. The robot is then required to localize its own position, despite all these changes.

In experiment (b) the system tries to classify the ways a number of human subjects grasp several different objects over a reasonably long time span, therefore being independent of the subjects' styles and the object's shapes.

Our experimental results fully confirm that OISVM achieve all of the objectives posed at the beginning: they produce very small models (issue 3), with a square complexity in training in the number of samples and with bounded testing time (issue 4); moreover, they keep optimal performance, or only slightly less than optimal (issue 2), and retain the full generalisation power of standard SVMs (issue 1).

The paper is organized as follows: Section 2 reviews some mathematical background and Section 3 describes OISVMs. Section 4 gives a detailed comparison with existing similar approaches; Section 5 describes the experiments and the results we obtained, and lastly Section 6 contains conclusions and the outline of future work.

2 Background Theory

This section contains the required mathematical background. We introduce SVMs both in the batch and on-line versions. The interested reader is referred to, e.g., [18, 22] for a comprehensive introduction to the subject.

2.1 Support Vector Machines

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 drawn from an unknown probability distribution. We want to find a function $f(\mathbf{x})$ such that $sgn(f(\mathbf{x}))$ best determines the category of any future sample $\mathbf{x}$ drawn from the same distribution. Assuming the data are linearly separable, we can seek a minimum norm *separating hyperplane* in $\mathbb{R}^m$, $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$. In this case, the hyperplane must respect the constraints

$y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1\geq 0$, for all $i=1,\ldots,l$ (from now on, this will be implicit whenever a subscript i appears free in a formula). In the most general case in which the data are not linearly separable, we can relax the problem introducing l slack variables ξ_i and rather require that $y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1+\xi_i\geq 0$, with $\xi_i\geq 0$. This new problem is then usually solved minimizing the following convex function:

$$\min_{\mathbf{w},b}\left(||\mathbf{w}||^2+C\sum_{i=1}^{l}\xi_i^p\right) \tag{1}$$

subject to the constraints

$$\begin{array}{rcl} y_i(\mathbf{w}\cdot\mathbf{x}_i+b) & \geq & 1-\xi_i \\ \xi_i & \geq & 0 \end{array} \tag{2}$$

where $C\in\mathbb{R}^+$ is an error penalty coefficient and p is usually 1 or 2 [18]. (1) and (2) can be compactly expressed in Lagrangian form by introducing l pairs of coefficients α_i,μ_i and then minimizing the objective function

$$\begin{array}{rcl} L_P & = & \frac{1}{2}||\mathbf{w}||^2-\sum_{i=1}^{l}\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1+\xi_i\right) \\ & & +C\sum_{i=1}^{l}\xi_i^p-\sum_{i=1}^{l}\mu_i\xi_i \end{array} \tag{3}$$

subject to the constraints that $\alpha_i,\mu_i\geq 0$. Using the Karush-Kuhn-Tucker (KKT) conditions [18], giving *necessary and sufficient* conditions for $\mathbf{w},b$ and α_i to be be a solution, we obtain, for $p=1$,

$$\frac{\partial L_P}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i=1}^{l}\alpha_i y_i\mathbf{x}_i = 0\Rightarrow\mathbf{w}=\sum_{i=1}^{l}\alpha_i y_i\mathbf{x}_i \tag{4}$$

$$\frac{\partial L_P}{\partial\xi_i}=C-\alpha_i-\mu_i = 0 \tag{5}$$

$$\frac{\partial L_P}{\partial b}=\sum_{i=1}^{l}\alpha_i y_i = 0 \tag{6}$$

$$\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1+\xi_i\right) = 0 \tag{7}$$

$$\xi_i(\alpha_i-C) = 0 \tag{8}$$

Whereas for $p=2$ condition (8) disappears and condition (5) becomes

$$\frac{\partial L_P}{\partial\xi_i}=2C\xi_i-\alpha_i=0 \tag{9}$$

Taking (4) into account, $f(\mathbf{x})$ can be expressed as

$$f(\mathbf{x})=\sum_{i=1}^{l}\alpha_i y_i\mathbf{x}\cdot\mathbf{x}_i+b \tag{10}$$

Note that, in the last Equation and in (3), the $\mathbf{x}$'s only appear in the form of inner products. In order to improve the discriminative power of SVMs then, the $\mathbf{x}_i$s are usually mapped to a highly, possibly infinite-dimensional space (the *feature space*) via a non-linear mapping $\Phi(\mathbf{x})$; the core of the SVM becomes then the so-called *kernel function* K such that $K_{ij}=K(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. This idea is called *kernel trick* and is standard in SVM literature; it avoids the necessity of explicitly knowing Φ. 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). Equation (10) then becomes

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{11}$$

After training, that is after the minimization of L_P, some of the α_is (actually most of them in many practical applications) are zero; those $\mathbf{x}_i$s for which this does not hold are called *support vectors*.

Using the KKT conditions in (3) we obtain that to train a SVM we must solve a quadratic programming problem (QP) with as many unknowns as training samples. State-of-the-art QP solvers decompose the problem into manageable subproblems or, in the limit, perform iterative pairwise optimization [23]. This approach is essentially batch, that is, all the training samples must be available from the start.

2.2 On-line Learning with SVMs

In on-line learning the training samples are available one at time and no a-priori knowledge of the full training set can be assumed. In general, an on-line learning framework works iteratively by refining a hypothesis h_i. In particular, at any point in time a new sample $\mathbf{x}_{l+1}$ is received, it is predicted using the current hypothesis h_l. The true label y_{l+1} is matched against its prediction, and a new hypothesis h_{l+1} is generated, taking into account the loss incurred in the prediction [22]. In other words, for any given sequence of samples/labels pairs, $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)$, a sequence of hypotheses $h_1, \cdots, h_l$ is generated, such that h_i depends only on h_{i-1} and $(\mathbf{x}_i, y_i)$.

In the case of kernel based algorithms such as SVMs, the representer theorem [24, 25] states that, under broad hypotheses, the solution to a problem such as (1) can be expressed as a linear combination of kernel functions evaluated on the training points. Hence for any l, h_l is a linear combination of kernel functions evaluated on $\mathbf{x}_1, \cdots, \mathbf{x}_l$. In a sense, h_l takes into account all samples up to $\mathbf{x}_l$, so there is no real difference in calculating the solution using all the samples received $\mathbf{x}_1, \cdots, \mathbf{x}_l$ or only using h_{l-1} and $(\mathbf{x}_l, y_l)$.

Note that, unlike in the batch setting, here we are also interested in the intermediate hypotheses $h_i, i = 1, \dots, l$. In fact the machine is used every time a new sample is acquired, for training and testing.

The standard training algorithms for SVMs are meant to be used in the batch setting. A first version of on-line SVMs is presented in [26], while the exact solution is presented in [15] and extended to regression in [27].

3 On-line Independent SVM

In general it is possible to obtain an alternative, equivalent, and possibly more compact representation of an SVM solution. This follows from the fact that the solution of an SVM problem is not unique if the kernel matrix K does not have full rank. This is equivalent to some of the support vectors being linearly dependent on the others *in the feature space*. In fact, as pointed out in [28], given a vector $\boldsymbol{\alpha}$ solution of (1) and (2), consider $\boldsymbol{\delta}$ that belongs to the null space of K, orthogonal to the vector all of whose components are 1 and satisfing $\sum_{i=1}^{l} \delta_i y_i = 0$. If $0 \leq \alpha_i + \delta_i \leq C$ then $\boldsymbol{\alpha} + \boldsymbol{\delta}$ is also a solution. This idea was exploited by Downs et al. [29], where they observed that if a support vector is dependent on the other support vectors in the feature space, i.e.

$$\exists \mathbf{x}_k : K(\mathbf{x}, \mathbf{x}_k) = \sum_{i=1, i\neq k}^{l} c_i K(\mathbf{x}, \mathbf{x}_i) \tag{12}$$

then the decision function (11) found after training can be written as

$$f(\mathbf{x}) = \sum_{i=1,i\neq k}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + \alpha_k y_k \sum_{i=1,i\neq k}^{l} c_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{13}$$

Hence it is possible to remove the dependent support vector $\mathbf{x}_k$, update the other coefficients, and obtain a new smaller representation of the decision function, without changing it in any way. Note that the new coefficients may not respect the KKT constraints.

It is possible to generalize this result showing that the space of possible solutions to an SVM problem is even larger than the one found by the standard formulation. In fact using the Representer Theorem [24, 30], (4) can be written as follows:

$$\mathbf{w} = \sum_{i=1}^{l} \beta_i \mathbf{x}_i \tag{14}$$

for a set of generic coefficients β_i. Substituting (14) in (3) and using the kernel trick, we get

$$\begin{aligned} L'_P \;\; = \;\; & \sum_{i,j}^{l} \left(\frac{1}{2}\beta_i - \alpha_i y_i \right) \beta_j K_{ij} - \sum_{i=1}^{l} \alpha_i (b y_i - 1 + \xi_i) \\ & + \sum_{i=1}^{l} C \xi_i^p - \sum_{i=1}^{l} \mu_i \xi_i \end{aligned} \tag{15}$$

Now, enforcing the KKT conditions on *this*, more general version of the problem, one obtains that

$$\frac{\partial L'_P}{\partial \beta_i} = \sum_{i=1}^{l} (\beta_i - \alpha_i y_i) K_{ij} = 0 \tag{16}$$

Clearly, in order for (16) to hold, the vector whose components are $\beta_i - \alpha_i y_i$ must be in the null space of K. Now if K has full rank, the null space only consists of the null vector, and therefore $\beta_i = \alpha_i y_i$ (this particular result already appears in [20]). Otherwise, there are infinite solutions to the SVM problem, and the β_is are not constrained at all: this agrees with Downs et al.'s method and generalizes it.

3.1 The proposed method

Given that we are interested in on-line training, a possible solution would be to simplify the solution after each update, that is after each time a new sample is acquired. This would be extremely time consuming, so we need a way to use independent SVs only. Hence, the main idea is to decouple the concept of "basis" vectors, used to build the classification function (11), from the samples used to find out the α_is. 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 only if it is linearly independent from those already present in the basis itself, in the feature space. 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 5 for a deeper discussion on this point).

Denoting the indexes of the vectors in the current basis, after l training samples, by $\mathcal{B}$, 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 $\mathcal{B}$; otherwise, leave $\mathcal{B}$ unchanged.

2. incrementally re-train the machine.

These are the core steps of our algorithm that we call On-line Independent Support Vector Machine (OISVM). We will describe in more details these two steps in the following sections.

In the following, the notations A_{IJ} and $\mathbf{v}_I$, $\mathbf{v}$ 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.

3.2 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 of basis vectors, 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 [31]. Let $d_j \in \mathbb{R}$; then let

$$\Delta = \min_{\mathbf{d}} \left\| \sum_{j \in \mathcal{B}} d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{l+1}) \right\|^2 \tag{17}$$

If $\Delta > 0$ then $\mathbf{x}_{l+1}$ is linearly independent with respect to the basis, and $l+1$ is added to $\mathcal{B}$. 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 to zero, OISVMs retain the exact accuracy of SVMs, in fact no approximation is involved. Note also that if the feature space has finite dimension n, then no more than n linearly independent vectors can be found, and $\mathcal{B}$ will never contain more than n vectors. While for infinite dimensional kernels, compact input domain and η greater than zero it is possible to demonstrate that the number of basis vectors will be finite for any training sequence [31]. Hence for both finite and infinite dimensional kernels we break the linear dependency between the number of SVs and the number of training samples [21].

We just need a way to efficiently calculate Δ. Expanding (17) we get

$$\begin{aligned}\Delta = \min_{\mathbf{d}} (& \sum_{i,j \in \mathcal{B}} d_j d_i \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_i) \\ & -2 \sum_{j \in \mathcal{B}} d_j \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_{l+1}) \\ & +\phi(\mathbf{x}_{l+1}) \cdot \phi(\mathbf{x}_{l+1}) \ )\end{aligned} \tag{18}$$

that is, applying the kernel trick,

$$\Delta = \min_{\mathbf{d}} \left(\mathbf{d}^T K_{\mathcal{B}\mathcal{B}} \mathbf{d} - 2\mathbf{d}^T \mathbf{k} + K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) \right) \tag{19}$$

where $k_i = K(\mathbf{x}_i, \mathbf{x}_{l+1})$ with $i \in \mathcal{B}$. Solving (19), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\tilde{\mathbf{d}} = K_{\mathcal{B}\mathcal{B}}^{-1} \mathbf{k} \tag{20}$$

and, by replacing (20) in (19) once,

$$\Delta = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \mathbf{k}^T \tilde{\mathbf{d}} \tag{21}$$

Note that $K_{\mathcal{B}\mathcal{B}}$ 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 $K_{\mathcal{B}\mathcal{B}}^{-1}$ then becomes

$$\begin{bmatrix} & & & 0 \\ & K_{\mathcal{BB}}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} \tilde{\mathbf{d}} \\ -1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{d}}^T & -1 \end{bmatrix} \tag{22}$$

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 [15]). Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{B}|^2)$, as one can easily see from (20).

It is also possible to view the linear independence in the functional space, that is to try to approximate the function $K(\mathbf{x}_{l+1}, .)$ with $\sum_{j\in\mathcal{B}} d_j K(\mathbf{x}_j, .)$, however the resulting equations are the same (cfr. section 10.2.1 in [32]).

With this method we are approximating the original kernel matrix K with another matrix $\widehat{K}$ [33]; the quality of the approximation depends on η. In fact it is possible to show that $trace(K - \widehat{K}) \leq \eta|\mathcal{B}| \leq \eta l$, where l is the number of samples acquired [31]. 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.

To gain other insights on this active sparsification method, consider the case in which a Gaussian kernel is used. Consider the expression (17) with $\mathcal{B} = \{i\}$, that is, as the i-th element is the only one in the base

$$\Delta_i = \min_{d_i} ||d_i \phi(\mathbf{x}_i) - \phi(\mathbf{x}_{l+1})||^2 \tag{23}$$

Obviously $\Delta_i \geq \Delta, \forall i \in \mathcal{B}$, so if $\Delta_i \leq \eta$ then we have that $\Delta \leq \eta$ and the sample $l+1$ will not be added to the basis set. Remembering (18)-(21), last equation can be expanded in

$$\Delta_i = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \frac{K(\mathbf{x}_{l+1}, \mathbf{x}_i)^2}{K(\mathbf{x}_i, \mathbf{x}_i)} \tag{24}$$

If we consider the case where the kernel is Gaussian we have that $K(\mathbf{x}, \mathbf{x}) = 1, \forall \mathbf{x}$ and we can write

$$\begin{aligned} \Delta_i \leq \eta \quad &\Leftrightarrow \quad 1 - K(\mathbf{x}_{l+1}, \mathbf{x}_i)^2 \leq \eta \\ &\Leftrightarrow \quad K(\mathbf{x}_{l+1}, \mathbf{x}_i) \geq \sqrt{1-\eta} \\ &\Leftrightarrow \quad \exp\left(-\gamma\, ||\mathbf{x}_{l+1} - \mathbf{x}_i||^2\right) \geq \sqrt{1-\eta} \\ &\Leftrightarrow \quad ||\mathbf{x}_{l+1} - \mathbf{x}_i||^2 \leq -\frac{1}{2\gamma} \log(1-\eta) \end{aligned} \tag{25}$$

Hence if at least one point $\mathbf{x}_i$ of the basis set is too near to the new point $\mathbf{x}_{l+1}$, it will be not added to the basis set. In other words when we use a Gaussian kernel, fixing a certain value of η implies imposing a minimum distance between the points selected as basis vectors.

3.3 Training the machine

The training method largely follows the modified Newton of Keerthi et al. [20, 34], that we have adapted for on-line training. The algorithm directly minimizes problem (1) as opposed to the standard way of minimizing its dual Lagrangian form, allowing to select explicitly the basis vectors to use. To apply the modified Newton we set $p = 2$ in (1) and transform it to an unconstrained minimization problem. Let $\mathcal{D} \subset \{1, \ldots, l\}$; then the unconstrained problem is

$$\min_{\boldsymbol{\beta}} \left(\frac{1}{2}\boldsymbol{\beta}^T K_{\mathcal{DD}} \boldsymbol{\beta} + \frac{1}{2} C \sum_{i=1}^{l} max\left(0, 1 - y_i K_{i\mathcal{D}} \boldsymbol{\beta}\right)^2 \right) \tag{26}$$

where $\boldsymbol{\beta}$ is the vector of the Lagrangian coefficients involved in $f(\mathbf{x})$, analogously to the α_is in the original formulation. For convenience the bias term has not been included, but the analysis presented in this section can be easily extended to include it. Then, we explicitly set $\mathcal{D} = \mathcal{B}$, assuring thus that the solution to the problem is unique, since $K_{\mathcal{BB}}$ is full rank by construction. Newton's method as modified by Keerthi et al. [20, 34] can then be used to solve (26) after each new sample. When the new sample $\mathbf{x}_{l+1}$ is received the method goes as follows:

1. use the current value of $\boldsymbol{\beta}$ as starting vector;
2. let $o_{l+1} = K_{l+1,\mathcal{B}}\boldsymbol{\beta}$, if $1 - y_{l+1}o_{l+1} \geq 0$ stop: the current solution is already optimal;
3. let $\mathcal{I} = \{i : 1 - y_i o_i > 0\}$ where $o_i = K_{i,\mathcal{B}}\boldsymbol{\beta}$ is the output of the i-th training sample;
4. update $\boldsymbol{\beta}$ with a Newton step: $\boldsymbol{\beta} - \gamma \mathbf{P}^{-1}\mathbf{g} \rightarrow \boldsymbol{\beta}$ where $\mathbf{P} = K_{\mathcal{BB}} + CK_{\mathcal{BI}}K_{\mathcal{BI}}^T$ and $\mathbf{g} = K_{\mathcal{BB}}\boldsymbol{\beta} - CK_{\mathcal{BI}}(\mathbf{y}_{\mathcal{I}} - \mathbf{o}_{\mathcal{I}})$;
5. let $\mathcal{I}^{new} = \{i : 1 - y_i o_i > 0\}$ where o_i are ricalculated using new $\boldsymbol{\beta}$. If $\mathcal{I}^{new}$ is equal to $\mathcal{I}$ stop; otherwise $\mathcal{I}^{new} \rightarrow \mathcal{I}$ and go to step 4.

In Step 4 above, we have set γ to one, without experiencing any convergence problem. With this choice the update of $\boldsymbol{\beta}$ is $C\mathbf{P}^{-1}K_{\mathcal{BI}}\mathbf{y}_{\mathcal{I}} \rightarrow \boldsymbol{\beta}^{new}$. In order to speed up the algorithm, we maintain an updated Cholesky decomposition of $\mathbf{P}$ and a vector with the product $K_{\mathcal{BI}}\mathbf{y}_{\mathcal{I}}$: every time a sample enters or exits from the set $\mathcal{I}$ these two quantities are updated. It turns out that the algorithm converges in very few iterations, usually 0 to 2; the time complexity of the re-training step is $O(|\mathcal{B}|l)$, as well as its space complexity. So the time complexity for training l training points is $O(l^2)$, because, as said in Section 3.2, after a certain number of samples the size of $\mathcal{B}$ will stop growing. Hence, keeping $\mathcal{B}$ small will speed up the training time as well as the testing time.

3.4 Multiclass extension

OISVM can be easily extended to multiclass, in particular the ONE-vs-ALL framework can be used. N different machines are trained, one for each class, to discriminate between one class versus all the others. The ONE-vs-ALL has been demostrated to have equal performances to the ONE-vs-ONE multiclass extension [35], while having in average more support vectors [36]. This is not a problem in our setting because, as said above, OISVM breaks the linear dependency between number of training samples and number of support vectors. In fact, remembering that the sparsification procedure is unsupervised, that is doesn't use the labels y_i, the kernel matrix $K_{\mathcal{BB}}$ is the same for each trained machine. In this way the linear indipendence check must be done only once for each sample, cutting computational costs.

In the following all the experiments on multiclass databases will be done using this methodology.

4 Comparison with Existing Approaches

Comparing our approach to similar ones in relevant literature, several constraints present in the problem must be taken into account. On-line learning rules out all methods relying on the knowledge of the complete training set; this includes, e.g., methods for low rank approximation of the kernel matrix based on Incomplete Cholesky Factorization [33, 37, 38]. For the same reason, any after-training simplification procedure cannot be used (e.g. [17, 29], chapter 18.3 in [32]). In fact this idea is useful in reducing the testing time, but it is unfeasible in an on-line setting, since the simplification should be performed every time a new sample is acquired.

Other methods to heuristically select a subset of the support vectors have been proposed, e.g., in [20, 39]. In [40], instead, a method to directly build a “vocabulary” of vectors is proposed, but the formulation is not convex and the SVM feature of having a unique solution is lost. Besides this, these methods require the knowledge of the full training set too, and, again, are not suited for on-line learning.

On the other hand, a solution with *bounded complexity* must be produced. As a matter of fact, the number of support vectors retained by an SVM is proportional to the number of training samples [21], and the testing time is in turn proportional to the number of support vectors; so in an on-line setting the machine will eventually become unfeasibly slow. Obviously, while bounding the complexity of the solution, one also wants to have the best possible solution.

To bound the number of operations during testing, the complexity of the predictor must be somehow bounded a priori. Training in the primal with a linear kernel can be a way to have a small solution, but it cannot be used with infinite dimensional kernels. On the other hand several on-line kernel-based methods have been proposed to bound complexity of predictor [16, 31, 41–43]. For all of them the price to pay is a loss in prediction performance. Moreover in many of them (e.g. [16, 41–43]) only an approximate solution is found after each update, and the algorithm will slowly converge to the true solution using a gradient-descent-like procedure.

On the other hand, OISVM always finds the best possible solution, given the subset of vectors selected as basis vectors to build it. This is due to the fact that all the received training samples are stored, like in the on-line methods in [15, 27]. This can be a problem if we attempt to move towards life-long learning [1], but it could be solved using, for example, some kind of forgetting strategy, like the ones proposed in [42]. Alternatively, one could 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.

Even if the sparsification method presented in this paper is also used in [31, 41], we use the original loss function of the SVM, more suited for classification tasks. In fact the maximization of the margin and the concept of the margin itself can be taken into account during training only using this loss function [18]. In fact in both the papers cited above the loss function used is the squared error, more suited for regression problems, implicitly assuming an additive gaussian noise on the output values. On the other hand the exact solution to on-line SVM learning in [15] cannot be used to reduce the number of support vectors.

A different method has been proposed by Liu et al. [44] and rediscovered by Collobert et al. [45]: they have used a non-convex formulation of the learning problem where training errors are no longer support vectors, thus dramatically reducing the growth rate of the support vectors with the training samples. However, in the paper it is not clear if the number of support vectors reaches a limit or if it will grow indefenitely, even if less than with standard SVM.

5 Experimental Evaluation

In this section we report the experimental evaluation of OISVMs. We first test the method on a set of databases commonly used in the machine learning community (section 5.1); we then apply it to more realistic scenarios, of wider interest to the community of cognitive systems. The first is about place recognition, where the aim is to update the model to handle variations in an indoor environment (section 5.2). In the second we show how our method classifies different types of human grasps, incrementally updating the model with the information coming from the observation of different subjects (section 5.3). These applications are representative of the needs of an autonomous agent as discussed in the introduction.

OISVMs have been implemented in Matlab and tested against LIBSVM v2.82 [46]. For the sake of comparison, LIBSVM has been also modified as suggested by the Authors in order to set $p = 2$ in (1); this modified version is called LIBSVM-2 in the following. The LIBSVM software library was also extended to various families of kernels, and to the fixed-partition incremental SVM [26] one approximate incremental extension of SVM[1]. In the case of finite-dimensional kernels, we only show the performance of LIBSVM-2 against OISVMs with η at machine precision, since the solution found by OISVM is exactly equivalent; in the case of infinite-dimensional kernels, we show curves for various values of η.

5.1 Standard Benchmarks

Fig. 1 shows the above mentioned comparison on two standard benchmark databases, available at `http://www.csie.ntu.edu.t`
For each benchmark, data are obtained by running 10 random $75\%/25\%$ train/test runs.

Consider Fig. 1, left pane, Diabetes dataset. When all samples have been loaded, LIBSVM-2 has about 427 SVs, and LIBSVM about 290. The kernel used is a homogeneous polynomial with degree 3 and the benchmark has 8 features, therefore the dimension of the feature space is $\binom{10}{3} = 120$ (see, e.g., [28]); as expected, OISVM stops acquiring new SVs when there are exactly 120, although it loads a few more than the other approaches

[1]We have not considered the algorithm in [15] for the comparison being its solution equal to the one found by LIBSVM.

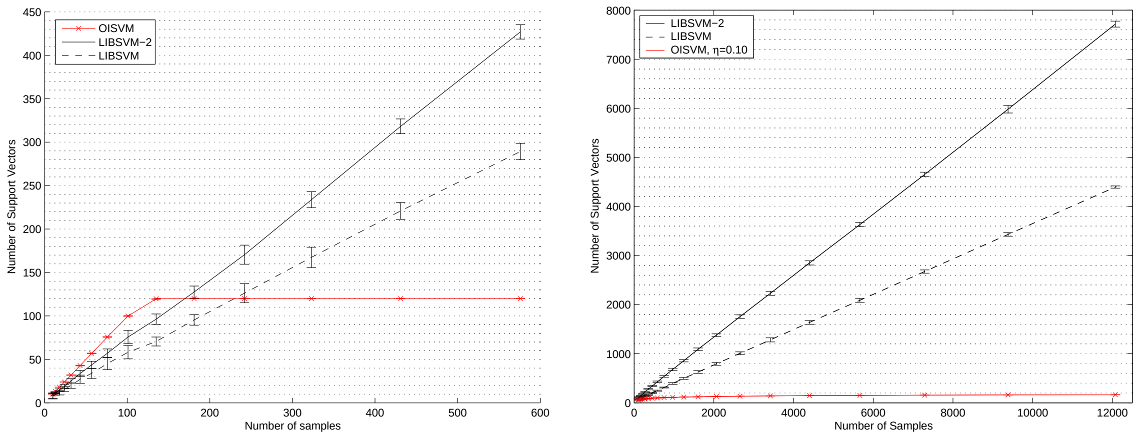

Figure 1: Comparison of OISVM and LIBSVM on the *Diabetes* (left pane) and *Adult7* (right pane) benchmarks. *Diabetes* is solved using a polynomial kernel with degree 3, while *Adult7* is solved using a Gaussian kernel.

Table 1: Comparison of OISVM, RSVM, LIBSVM and LIBSVM-2 on 10 standard benchmarks, solved using a Gaussian kernel. For each benchmark, we report the classification rate and the number of SVs. The value of η has been chosen in order not to loose more than 0.5% accuracy respect to LIBSVM-2.

Dataset	OISVM	RSVM	LIBSVM-2	LIBSVM
Banana	89.54 ± 0.41 (18.90 ± 1.29)	88.81 ± 0.63	89.75 ± 0.30 (173.60 ± 21.20)	89.77 ± 0.28 (121.10 ± 9.01)
Breast	73.51 ± 4.21 (36.00 ± 2.67)	71.95 ± 4.79	74.03 ± 4.15 (199.70 ± 0.67)	74.55 ± 4.16 (122.60 ± 4.95)
Diabetis	76.83 ± 2.13 (8.60 ± 0.52)	75.77 ± 2.85	76.83 ± 2.21 (417.00 ± 4.00)	76.83 ± 1.77 (283.30 ± 8.17)
Flare	66.35 ± 1.17 (10.40 ± 0.70)	63.77 ± 4.05	66.35 ± 1.23 (631.10 ± 4.20)	67.15 ± 1.63 (555.70 ± 8.59)
German	76.20 ± 1.93 (52.60 ± 2.46)	75.43 ± 2.20	76.70 ± 1.79 (600.70 ± 11.89)	76.37 ± 2.60 (384.70 ± 7.01)
Heart	84.90 ± 1.91 (14.30 ± 0.67)	84.30 ± 2.95	85.00 ± 1.83 (161.60 ± 2.63)	85.00 ± 2.87 (85.00 ± 3.27)
Ringnorm	98.03 ± 0.27 (87.60 ± 6.01)	98.60 ± 0.10	98.57 ± 0.10 (377.00 ± 4.06)	98.50 ± 0.10 (213.00 ± 4.74)
Titanic	77.84 ± 0.66 (11.90 ± 1.37)	74.81 ± 3.24	77.60 ± 1.63 (146.00 ± 5.27)	77.28 ± 0.35 (86.80 ± 6.91)
Twonorm	97.14 ± 0.34 (60.60 ± 5.44)	97.18 ± 0.32	97.44 ± 0.26 (400.00 ± 0.00)	97.65 ± 0.09 (299.90 ± 5.74)
Waveform	89.67 ± 0.47 (78.20 ± 3.12)	89.66 ± 0.65	90.10 ± 0.36 (324.60 ± 9.43)	89.62 ± 0.58 (220.30 ± 10.49)

before reaching the limit. The accuracy (not displayed) is exactly the same. Again, note that, after having acquired 120 SVs, OISVM will never acquire any more, while keeping the same accuracy, whereas LIBSVM and LIBSVM-2 will, as theoretically proved in [21].

Consider now Fig. 1, right pane, Adult7 dataset. The kernel used is Gaussian and its dimension is infinite. The benchmark is large and complex (16100 samples, 123 features); nevertheless, with $\eta = 0.1$, at the end OISVM has about 2% of the SVs used by LIBSVM-2 and less than 4% with respect to LIBSVM. The accuracy is essentially the same as that of LIBSVM-2 (namely, $0.069\% \pm 0.068$ on average worse).

Lastly, consider Table 1, which shows the very same data in compact form for 10 more standard databases. In each column we show the mean recognition rate on 10 train/test splits taken from [47], and the number of support vectors in parenthesis. We have compared our method to the batch method LIBSVM-2 and LIBSVM. Even if, as said above, the OISVM formulation is analogous to LIBSVM-2, using the square of the slack variables in (1), we have considered also the more standard formulation with the norm-1 of the slacks because it is known to be more sparse. Cross validation was used to find the best parameters for each dataset, separately for the norm-1 and norm-2 formulation, while for OISVM we used the same parameters of the norm-2. OISVM attains a number of SVs which is about 3 to slightly more than 60 times less than LIBSVM-2, whereas the accuracy is not worse than 0.5%. Moreover it is always sparser than the norm-1 formulation.

For a complete comparison we have used also a variation of the RSVM method [39], the same used also in [20], in which for each run we have randomly selected a number of support vectors equal to the one selected

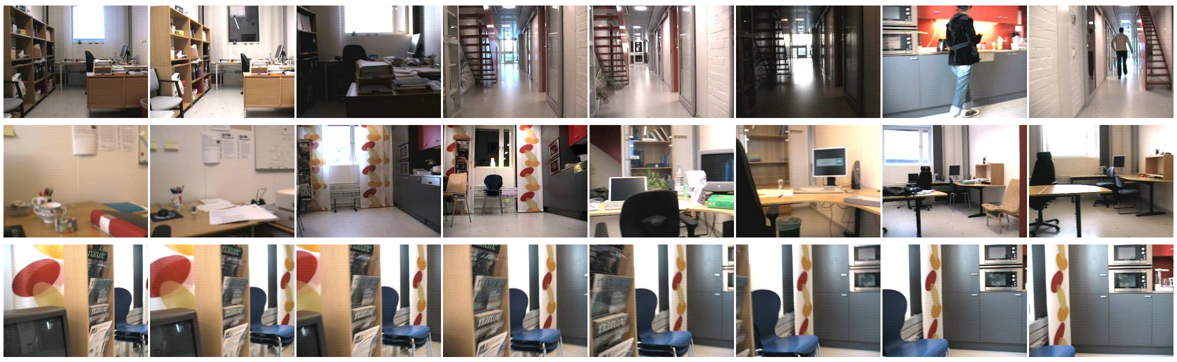
Figure 2: Sample images illustrating the variations captured in the IDOL2 database. Images in the top row show the variability introduced by changes in illumination for two rooms (first six images) as well as people appearing in the environment. The middle row shows the influence of people's everyday 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.

by OISVM. We have used a Wilcoxon signed-ranks test to compare the performances of the two methods, as suggested in [48], obtaining a p-value less than 0.02. This confirms that our strategy to select support vectors is better than the random sampling, suggested by many authors, e.g. [49]. As a plus our strategy assures that, as said before, the growth of support vectors will always eventually stop.

5.2 Robot Navigation

We performed a second series of experiments, namely place recognition in an indoor environment, to evaluate our algorithm. We considered a realistic scenario where the algorithm had to incrementally update the model, so to adapt to the variations in an indoor environment due to human activities over long time spans. These variations include people appearing in different rooms during working time, objects such as cups moved or taken in/out of the drawers, pieces of furniture pushed around, and so forth.

Experiments were conducted on the IDOL2 database (Image Database for rObot Localization 2, [50]), which contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms. The acquisition was performed within an indoor laboratory environment consisting of five rooms of different functionality. The sequences were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. Thus, these data capture natural variability that occurs in real-world environments because of both natural changes in the illumination and human activity. Fig. 2 shows some sample images from the database, illustrating the difficulties 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, we refer the readers to [50].

The evaluation was performed using composed receptive field histograms (CRFH) [51] as global image features and SIFT [52] for extracting local features. In the experiments, we consider both the *exponential* χ^2 kernel for SVM (when using CRFH), and the *matching kernel* [53] (when using SIFT). Note that the kernel in [53] is not always positive semidefinite [54], 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.

We benchmarked OISVM against the approximate incremental SVM extensions of fixed-partition technique [26] and against the standard batch algorithm. Note that for the standard SVM the training is not on-line.

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 [55], 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, the bottom row of Fig. 2 shows some sample images acquired within a time span of 1.6 sec. This setup allows us to examine how the algorithms perform on data with fewer variations. 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 the exponential χ^2 kernel and for the matching kernel; here we report the average results, plus/minus one standard deviation. Fig. 3, left, shows the recognition rates of the exponential χ^2 kernel (top) and matching kernel (bottom) experiments obtained at each step using OISVM, LIBSVM and the approximate incremental algorithm. Fig. 3, 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 achieve 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 3, 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 no 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 [54]; moreover, in the induced space of the matching kernel, there seems to be a high probability that pairs of training points be (almost) orthogonal to each other (note 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 [31].

5.3 Learning by Imitation

Lastly, we realised a grasping classification experiment involving human subjects, in order to check how well our approach works in identifying essential components of object grasping, by reducing the number of support vectors of models trained on classification of grasping postures. The general idea is that of obtaining a statistically relevant collection of grasping postures (independent of the subject, and of the object grasped) and then trying to identify the posture. This would give us an indication of how well the *affordances* [12] associated with several objects can be mechanically understood.

5.3.1 Materials and methods

Eight subjects were involved in the experiment, 5 men and 3 women, all able-bodied, aged between 23 and 36. They were given no prior knowledge on the aim and scope of the experiment. Each subject would sit confortably in front of a workspace large about one squared meter and wear a 22-sensors Immersion CyberGlove [56] on the right hand, and a Force Resistor Sensor (FSR) on the thumb. Figure 4 (upper row) shows the devices, as worn by a subject.

The CyberGlove returns 22 8-bit numbers linearly related to the angles of the subject's hand joints; the sensors are embedded in the glove in order for them to be adherent to the subject's skin. The resolution of the sensors is on average about 0.5 degree [56], but the noise associated with the sensors has been experimentally determined to be 1.1 on average and 3 at the maximum [57]. The sensors describe the position of the three

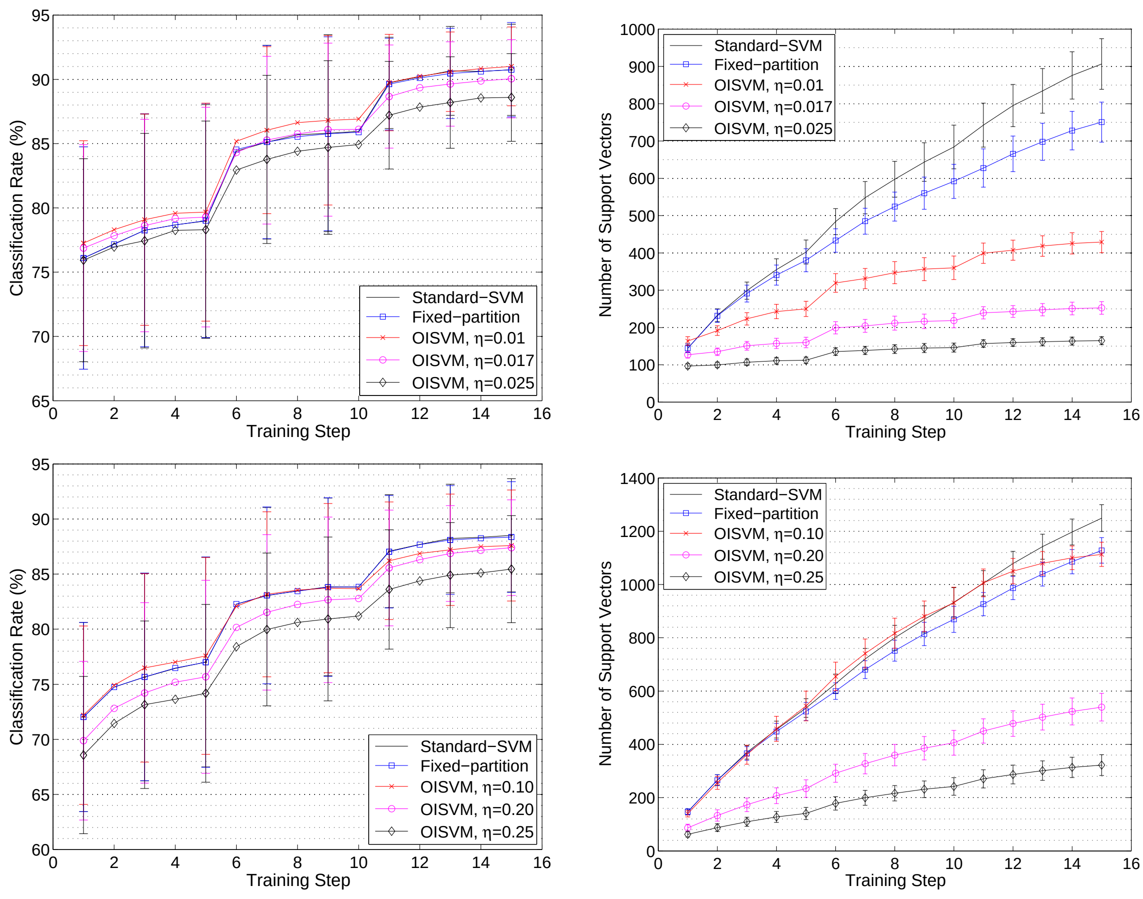

Figure 3: Expermental results on the IDOL2 database, using OISVM with three different values of η, the fixed-partition and the standard SVM. Top: χ^2 kernel, Bottom: matching kernel.

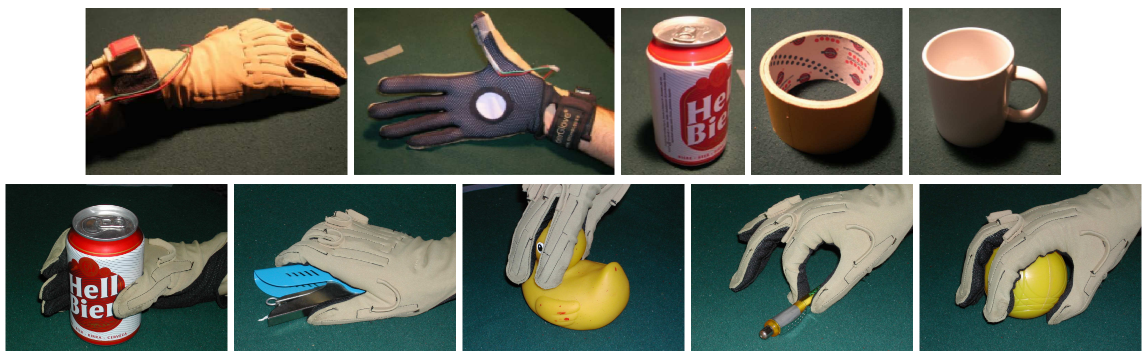

Figure 4: (above) The devices and some of the objects used for the experiment, left to right: the CyberGlove, the Force Resistor Sensor attached to the subject's thumb, the beer can, the duct tape roll and the mug. (below) The 5 grasp types to be recognised, left to right: power large, power flat, tripodal precision, thumb/index precision and spherical precision.

phalanxes of each finger (for the thumb, rotation and two phalanxes), the four finger-to-finger abductions, the palm arch, the wrist pitch and the wrist yaw.

The FSR returns a 32-bit number inversely related to the pressure applied to the surface of the sensor. We used it as an on-off indicator of when the subject was actually holding an object. Toghether, the CyberGlove and FSR would give us a precise idea of what the subject's hand posture was when grasping. All data were collected, synchronised, and saved in real time at a frequency of 50Hz.

We then collected a number of objects encountered in everyday life, and split them among 5 pairs of sets, each set containing objects of various size, colour, shape and affordances [12]. The idea is that each pair of sets would be associated with a particular type of grasp, as identified by [58]. The grasp types and associated pairs were:

1. *power large grasp:* a beer can, a bottle and a box (set 1); an anatomical hand model, a wooden toy dragon and a mug (set 2);

2. *power flat grasp:* a hammer and a long Lego block (set 1); a stapler, a screwdriver and and a TV remote control (set 2);

3. *tripodal precision grip:* a small and a large ball, a rubber duck and a beer can (set 1); a marker, a ballpoint pen and a ball (set 2);

4. *thumb/index precision grip:* a knife, a duct tape roll, a short Lego block and a rubber duck (set 1); a marker and a ballpoint pen (set 2);

5. *spherical precision grip:* a small and a large ball and a rubber duck (set 1); a fluffy toy airplane, a ball and a mug (set 2).

Figure 4 shows three of the objects used in the experiment and five examples of grasp types. Note that each object may appear in more than just one set: in fact, the sets are organised by grasp type (not by object). For instance, a rubber duck can be grasped either via a tripodal precision grip, a thumb/index precision grip and/or a spherical precision grip.

The experiment consisted of two phases. During the first phase, for each pair, the first set of objects was put on the workspace, and the subject was asked to choose one of the objects, grasp it with the right hand, lift it and then put it back on the workspace. We explicitly asked the subject to grasp *using the grasp type associated to the objects on the workspace*. We collected 60 grasps for each set of objects, resulting in 300 grasps per subject, divided by grasp type. During the second phase, we repeated the same procedure but on a different day and using the second set of each pair. Therefore, we would obtain analogous results to the first phase, but allowing the subject to grasp different objects with the same grasp type, and leaving a long time in between, in order for the subject not to get used to a particular grasp type. Each phase was completed by the subjects in 17 to 18 minutes.

In the end, this would allow us to gather 120 grasps per grasp type and subject; this means 960 grasps per grasp type (for a total of 4800 grasps, since we had 5 grasp types), well distributed across various subjects, times of the day, fatigue conditions and objects.

The actual grasps, which would build the SVM training set, were detected as follows: we found each interval of time during which the FSR would signal contact, and then we took the CyberGlove sensors values in the middle of the interval. We assumed that the hand posture would not sensibly change during the grasping act, that is, while the subject was lifting the object. Spurious grasp detections were removed from the sample set, resulting in a total of 4512 samples.

Lastly, we set up five categories, each corresponding to a grasp type, and set the input space to $\mathbb{R}^{22}$, that is, one dimension per each CyberGlove sensor.

5.3.2 Data Analysis

The optimal hyperparameters C and σ were found via grid search and cross-validation, as is customary. The machines were trained on data gathered during the first phase and tested on data gathered during the second phase, and then vice-versa. This way, the system was always tested on data related to objects different from

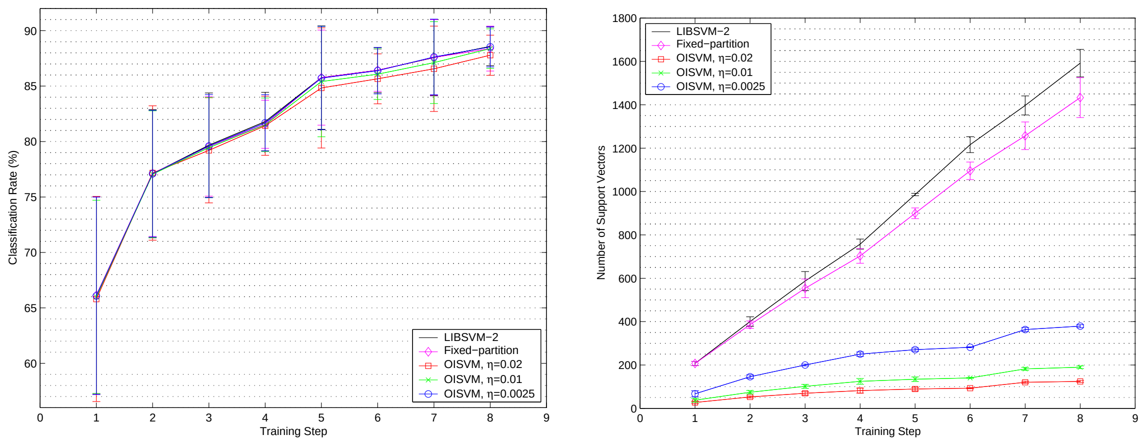

Figure 5: Classification rate (left) and average number of support vectors (left) for the grasping classification experiment. OISVM uses a Gaussian kernel and three different values of η.

those seen during training, in order to remove the possibility to learn the exact configuration of the hand rather than the type of grasp.

As in the place recognition scenario, we have compared the batch method LISVM-2 with the fixed-partition techique and OISVM. Each training step corresponds to a different user, and it also is a partition of the fixed-partition method. A Gaussian kernel and three different values of η were used for OISVM. The results are shown in Figure 5.

Consider the Figure, left pane: it is apparent that the classification rate is basically the same, uniformly and for all approaches tested, except for the curve with $\eta = 0.02$. The right pane shows that, in agreement with the previous experiments, LIBSVM-2 and the fixed partition method gather a number of SVs which grows proportionally with the training set. On the other hand, OISVM with various values of η uniformly show a dramatically smaller number of SVs, getting to as few as 124 SVs in the case for $\eta = 0.02$, losing only 0.8% of accuracy compared to LIBSVM-2.

6 Conclusions

Artificial cognitive systems cannot operate in realistic situations without a continuous learning algorithm, so to take advantage of experience and update accordinginly their internal representations. Large margin classifiers have been used successfully in several cognitive systems applications, such as grasping classification and topological localization. We focused on the Support Vector Machine algorithm, and in this paper we presented a new method to reduce the number of support vectors needed by a Support Vector Machine in an online setting, called OISVMs (Online Independent Support Vector Machines). The method avoids using in the solution those support vectors which are linearly dependent of previous ones in the feature space — in other words, the kernel matrix is always kept at full rank. The optimization problem is solved via an incremental algorithm which benefits of the small size of the kernel matrix.

We tested the method both on a standard set of benchmark databases and on two real-world case studies, namely: (a) place recognition in an indoor environment and (b) learning by imitation, i.e., human grasping classification. Both real-world experiments have been carefully crafted in order to take into account changing conditions in the environment: robot images were acquired under different weather conditions and across a time span of several months; grasping was done across a time span of several days, and on several different objects.

The performance of OISVMs depends on a parameter, η, which can be used to trade a better/worse accuracy for more/fewer support vectors. The experimental results, carried out for various values of η, show that OISVMs allow for an excellent trade off between accuracy and number of support vectors. Furthermore, our analysis clearly shows that OISVMs improve on SVMs, proving to be highly adaptive to environmental changes (Introduction, requirement 1) and accurate (Introduction, requirement 2), by keeping the number of support vectors extremely small, thus minimizing the need for storage and making testing and training significantly faster (Introduction, requirements 3 and 4).

This work can be extended in many ways. While OISVMs are able to perform continuous learning on data collected on a span of time of up to several months, it is still not possible to use the algorithm in a life-long learning scenario. To achieve this goal we plan to extend the method so to include a forgetting mechanism. Another important issue is the multi-modality: artificial cognitive systems typically acquire inputs from different modalities that must be combined together during learning so to build a rich internal model. Thus, we plan to extend OISVMs so to work on multimodal data streams. Finally, we considered so far a purely supervised learning framework, but while it is reasonable to have a privileged teacher for some time, in general the system will have to act autonomously. Thus, OISVMs should be extended to the semi-supervised framework. Future work will address these issues.

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] S. Thrun, “A lifelong learning perspective for mobile robot control,” in *Intelligent Robots and Systems*, V. Graefe, Ed. Elsevier, 1995.

[2] A. Tapus and R. Siegwart, “A cognitive modeling of space using fingerprints of places for mobile robot navigation,” in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[3] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt, “Incremental learning for place recognition in dynamic environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS07)*, San Diego, California, October 2007.

[4] S. Zeng and J. Weng, “Obstacle avoidance through incremental learning with attention selection,” in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

[5] M. Artač, M. Jogan, and A. Leonardis, “Mobile robot localization using an incremental eigenspace model,” in *Proceedings ICRA’02*, 2002.

[6] E. Brunskill and N. Roy, “SLAM using incremental probabilistic PCA and dimensionality reduction,” in *Proceedings ICRA’05*, 2005.

[7] C. Castellini, F. Orabona, G. Metta, and G. Sandini, “Internal models of reaching and grasping,” *Advanced Robotics*, 2007, special issue on Learning by Imitation. To appear.

[8] S. Ekvall and D. Kragić, “Grasp recognition for programming by demonstration,” in *Proceedings of IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2005.

[9] C. de Granville, J. Southerland, and A. H. Fagg, “Learning grasp affordances through human demonstration,” in *Proceedings of the International Conference on Development and Learning (ICDL’06)*, 2006.

[10] H. Friedrich, V. Grossmann, M. Ehrenmann, O. Rogalla, R.-D. Zöllner, and R. Dillmann, “Towards cognitive elementary operators: grasp classification using neural network classifiers,” in *Proceedings of the IASTED International Conference on Intelligent Systems and Control, Santa Barbara (CA), USA*, oct 1999, pp. 121–126.

[11] G. Heumer, H. B. Amor, M. Weber, and B. Jung, “Grasp recognition with uncalibrated data gloves - a comparison of classification methods,” *IEEE Virtual Reality*, pp. 19–26, 2007.

[12] J. J. Gibson, “The theory of affordances,” in *Perceiving, acting and knowing: toward and ecological psychology*, R. Shaw and J. Bransford, Eds. Erlbaum, 1977, pp. 67–82.

[13] O. Martínez Mozos, C. Stachniss, and W. Burgard, “Supervised learning of places from range data using adaboost,” in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA’05)*, Barcelona, Spain, 2005, pp. 1742–1747.

[14] A. Pronobis, B. Caputo, P. Jensfelt, and H. I. Christensen, “A discriminative approach to robust visual place recognition,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS06)*, Beijing, China, October 2006.

[15] G. Cauwenberghs and T. Poggio, “Incremental and decremental support vector machine learning,” in *Advances in Neural Information Processing Systems*, 2000, pp. 409–415.

[16] J. Kivinen, A. Smola, and R. Williamson, “Online learning with kernels,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[17] D. Nguyen and T. Ho, “An efficient method for simplifying support vector machines,” in *ICML ’05: Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA: ACM Press, 2005, pp. 617–624.

[18] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[19] C. J. C. Burges and B. Schölkopf, “Improving the accuracy and speed of support vector machines,” in *Advances in Neural Information Processing Systems*, 1996, pp. 375–381.

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

[21] I. Steinwart, “Sparseness of support vector machines,” *Journal of Machine Learning Research*, vol. 4, pp. 1071–1105, 2003.

[22] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms (Adaptive Computation and Machine Learning)*. The MIT Press, December 2001.

[23] J. C. Platt, “Fast training of support vector machines using sequential minimal optimization,” in *Advances in kernel methods – support vector learning*, B. Schlkopf, C. Burges, and A. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[24] D. Cox and F. O’Sullivan, “Asymptotic analysis of penalized likelihood and related estimators,” *Ann. Statist.*, vol. 18, pp. 1676–1695, 1990.

[25] B. Schölkopf, R. Herbrich, and A. J. Smola, “A generalized representer theorem,” in *COLT ’01/EuroCOLT ’01: Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*. London, UK: Springer-Verlag, 2001, pp. 416–426.

[26] N. Syed, H. Liu, and K. Sung, “Incremental learning with support vector machines,” in *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99)*, 1999.

[27] J. Ma, J. Theiler, and S. Perkins, “Accurate on-line support vector regression,” *Neural Computation*, vol. 15, no. 11, pp. 2683–2703, 2003.

[28] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Knowledge Discovery and Data Mining*, vol. 2, no. 2, 1998.

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

[30] G. Kimeldorf and G. Wahba, “A correspondence between bayesian estimation of stochastic processes and smoothing by splines,” *Ann. Math. Statist.*, vol. 41, pp. 495–502, 1970.

[31] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least squares algorithm,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, 2004.

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

[33] F. R. Bach and M. I. Jordan, “Predictive low-rank decomposition for kernel methods,” in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

[34] S. S. Keerthi and D. DeCoste, “A modified finite Newton method for fast solution of large scale linear SVMs,” *Journal of Machine Learning Research*, vol. 6, pp. 341–361, 2005.

[35] R. Rifkin and A. Klautau, “In defense of one-vs-all classification,” *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, 2004.

[36] C.-W. Hsu and C.-J. Lin, “A comparison of methods for multiclass support vector machines,” *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, 2002.

[37] S. Fine and K. Scheinberg, “Efficient SVM training using low-rank kernel representations,” *The Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2002.

[38] G. Baudat and F. Anouar, “Feature vector selection and projection using kernels,” *Neurocomputing*, vol. 55, no. 1–2, pp. 21–38, 2003.

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

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

[41] L. Csató and M. Opper, “Sparse representation for gaussian process models,” *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[42] J. Weston, A. Bordes, and L. Bottou, “Online (and offline) on an even tighter budget,” in *Proceedings of AISTATS 2005*, R. G. Cowell and Z. Ghahramani, Eds. Society for Artificial Intelligence and Statistics, 2005, pp. 413–420.

[43] L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli, “Implicit online learning with kernels,” in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007.

[44] Y. Liu, X. Shen, and H. Doss, “Multicategory ψ-learning and support vector machine: Computational tools,” *Journal of Computational & Graphical Statistics*, vol. 14, no. 1, pp. 219–236, 2005.

[45] R. Collobert, F. Sinz, J. Weston, and L. Bottou, “Trading convexity for scalability,” in *ICML ’06: Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA: ACM Press, 2006, pp. 201–208.

[46] 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.

[47] G. Rätsch, “Benchmark repository,” Intelligent Data Analysis Group, Fraunhofer-FIRST, Tech. Rep., 2005, available at http://ida.first.fraunhofer.de/~raetsch.

[48] J. Demar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, January 2006.

[49] R. Rifkin, G. Yeo, and T. Poggio, “Regularized least-squares classification,” in *Advances in Learning Theory: Methods, Models and Applications*, ser. NATO Science Series III: Computer and Systems Sciences, J. A. K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, Eds. VIOS Press, 2003, pp. 131–154.

[50] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt, “The KTH-IDOL2 database,” KTH, CAS/CVAP, Tech. Rep. 304, 2006, available at http://cogvis.nada.kth.se/IDOL2.

[51] O. Linde and T. Lindeberg, “Object recognition using composed receptive field histograms of higher dimensionality,” in *Proceedings ICPR’04*, 2004.

[52] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, vol. 2. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–1157.

[53] C. Wallraven, B. Caputo, and A. Graf., “Recognition with local features: the kernel recipe,” in *Proceedings of ICCV’03*, 2003.

[54] S. Boughorbel, J.-P. Tarel, and F. Fleuret, “Non-mercer kernels for SVM object recognition,” in *Proceedings of British Machine Vision Conference (BMVC’04)*, London, England, 2004, pp. 137–146.

[55] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt, “Incremental learning for place recognition in dynamic environments,” accepted for *IROS’07*, to appear, 2007.

[56] *CyberGlove Reference Manual*, Virtual Technologies, Inc., 2175 Park Blvd., Palo Alto (CA), USA, August 1998.

[57] G. D. Kessler, L. F. Hodges, and N. Walker, “Evaluation of the cyberglove as a whole-hand input device,” *ACM Trans. Comput.-Hum. Interact.*, vol. 2, no. 4, pp. 263–283, 1995.

[58] M. R. Cutkosky, “On grasp choice, grasp models and the design of hands for manufacturing tasks,” *IEEE Trans. on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.