International Journal of Humanoid Robotics
© World Scientific Publishing Company

RECOGNITION OF DAILY LIFE ACTION AND ITS PERFORMANCE ADJUSTMENT BASED ON SUPPORT VECTOR LEARNING

TAKETOSHI MORI
MASAMICHI SHIMOSAKA
TATSUYA HARADA
TOMOMASA SATO
Graduate School of Information Science and Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, JAPAN 113–8656
{*tmori, simosaka, harada, tomo*}*@ics.t.u-tokyo.ac.jp*

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

This paper presents a recognition method for human daily life action. The system deals with actions related to regular human activity such as walking or lying down. The main features of the proposed method are: 1) simultaneous recognition, 2) expressing unclarity in human recognition, 3) defining similarity between two motions by utilizing kernel functions derived from expressions of action based on human knowledge, 4) robust learning capability based on support vector machine. Comparison with neural networks optimized by back propagation algorithm and decision trees generated by C4.5 proves that the accuracy of recognition in the proposed method is superior to the others. Recognizing daily life action robustly is expected to ensure smooth communication between humans and robots and to enhance support functionality in intelligent systems.

Keywords: Action Recognition; Learning; Support Vector Machine; Motion Capture File.

1. Introduction

In the future robots, including humanoids, are expected to support humans in everyday tasks and activities in the future. To provide assistance in various applications, it will be important for robots to communicate effectively with humans. Recognizing and understanding human actions has the potential to contribute to this communication ability along with many other applications, such as human computer interaction and search engines for multi media databases.

In most research works, the target actions are sign gestures. For example, Starner et al. made a system that recognizes American sign language[1] and Wilson et al.[2] implemented a system that recognizes user's gestures. Regular actions, such as walking and sitting will be important as the target actions to recognize in support systems for everyday life. We have designed and implemented a recognition system for such regular actions[3]. The details of these features are described in section 2. Because this system has lacked learning ability, in other words, the performance of the system must be tuned by hand, there have been problems with extensibility and versatility of the system. The main contribution of this paper is to solve these problems through the design and implement of a recognition algorithm whose performance can be tuned by a learning process using sample motion data.

2. Recognition of Human Daily Life Action

2.1. *Primitive characteristics in our recognition method*

The proposed recognition system in this paper contains the characteristics of our former recognition system[3] described below.

2.1.1. *Simultaneous recognition*

It is preferable that a recognition system for human daily life action be able to output multiple action names simultaneously as the recognition result. For example, humans can readily recognize the two actions involved when observing someone waving his or her hand while walking.

In order for the output of a recognition system to be closer to human recognition, we should design the system to have this ability. Specifically, a recognition system should be designed to be able to output multiple action names at the same time. We designed our system to have this ability of outputting multiple action names simultaneously.

2.1.2. *Unclarity in recognition*

Humans can not always could not decide, with an absolute certainty, on whether some actions really occur or not when watching someone acting. For instance, the decision as to whether a human is lying or not made by a human may have a degree of uncertainty when observing someone getting up from lying to sitting. In order for the output of a recognition system to be closer to human recognition, the system should be designed to be able to output “fuzzy” recognition results. We designed our system so it can output not only decisive but also unclear results.

2.1.3. *Utilizing expressions of actions hand-written by human*

Feature selection and extraction is considered as one of the most important elements in action recognition, because the specific features of each action such as the position and the pose of the body region vary widely. For example, the forward motion of hips could be one of the features of walking, while head direction is considered as an irrelevant and unnecessary feature of sitting.

Humans can intuitively extract specific important features of each action. In other words, humans can easily express an action by interpreting the motion or the pose of body parts. Thus we pay attention to these expressions. Consequently, our system uses only relevant features of the target action derived from the expressions.

2.2. *Criterion of Performance*

The performance of an action recognition system should be evaluated along with many other pattern recognition applications such as OCR, face recognition and speech recognition. The main criterion of the evaluation is the accuracy of recognition results. The correctness of recognition result is basically based on human judgment.

While humans can intuitively recognize human motion adequately and assign several action names, there are several way of evaluating the accuracy of recognition results. Because of the variety of actions and the time segmentation problem, it is hard to evaluate properly the accuracy of a recognition result.

Thus, in order for us to evaluate the performance of a recognition system, the correctness of the recognition result by humans is defined as follows. On observing someone acting when the person is performing several actions simultaneously, reference data of recognition result is labeled in synchronized with the input motion for all the candidate action names. Thus, as the performance evaluation of the system, the performance evaluation for each candidate action name is executed separately. In other words, the correctness of recognition results is generated by humans from moment to moment with someone's motion as decision results whether one specific action occurs or not.

The recognition algorithm we implemented in the former recognition system[3] fundamentally uses fuzzy logic, which is extended to handle time series problems. This method has the advantage that the system can be implemented intuitively. However, the ability of learning process is an essential factor in the action recognition system for the following two reasons. The first reason is that there is difficulty in tuning the performance by hand. The second is that there is a variety of action names we have to target.

3. Recognition System Implementation

3.1. *Input and output*

While many other recognition systems[4] use a sequential series of images as the input to the systems, our system uses a motion capture file that contains human posture, which is measured by some optical or mechanical motion capturing system. This enables the system to obtain information on motion easily and robustly, and we can concentrate on designing the recognition process itself.

The output of our system contains some action names in synchronized with a frame of the input, so that the problem of the segmentation of the time direction is not treated in this paper.

The format of the file is BVH[5], a de-facto standard motion file format by Biovision Corporation. BVH files contain the structure of a human as a linked joint model(figure) and the motion of the figure per frame. Fig. 1 depicts an example of the time-series (i.e. sequence) of input and output of our recognition system.

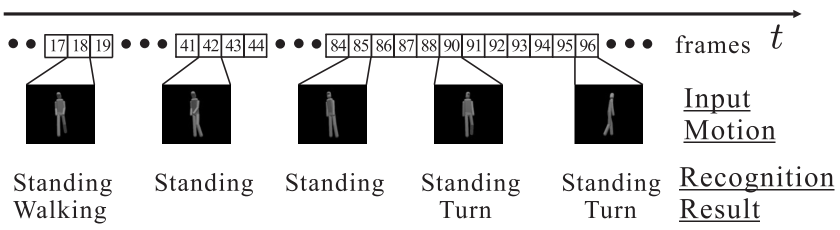


Fig. 1. This figure shows a time-series of the input and output of our recognition system.

The figure used in the proposed system is shown in Fig. 2. The hips are defined as the root joint of the figure and has 6 degrees of freedom (DOF). As for the torso, neck, legs and arms, each joint has 3 DOF. Therefore, the figure has total of 36 DOF. BVH files can be generated from the data captured by a magnetic motion capturing system (MotionStar, Ascension Technology) in which an actor wears magnetic sensors fitted to the corresponding joints. The body motion is measured every 33 msec.

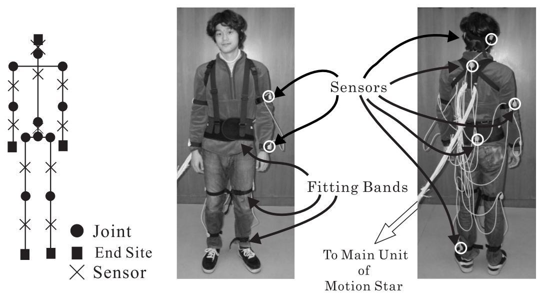


Fig. 2. Human as linked joint model and an actor with magnetic sensors are shown

3.2. *Recognition system configuration*

Fig. 3 shows the processing flow and the structure of our system. In order to realize the simultaneous recognition, the system contains multiple recognition processes, each of which is assigned to the recognition of one action. This primitive recognition process for one action is called an “Action Element Recognizer (AER)”.

The process of each AER runs in parallel with the other AERs. For example, AERs recognizing such actions as walking, sitting, and other actions run in parallel.

The system collects the results of all the processes, and outputs the results of each recognition process per frame. New action can be recognized simply by adding that recognition process to the system.

As noted above, an AER recognizes one assigned action. In practice, an AER recognizing walking discriminates whether someone is walking or not walking. Thus, the result of one AER process is independent of the others, and the total performance of the system relies on the performance of each AER.

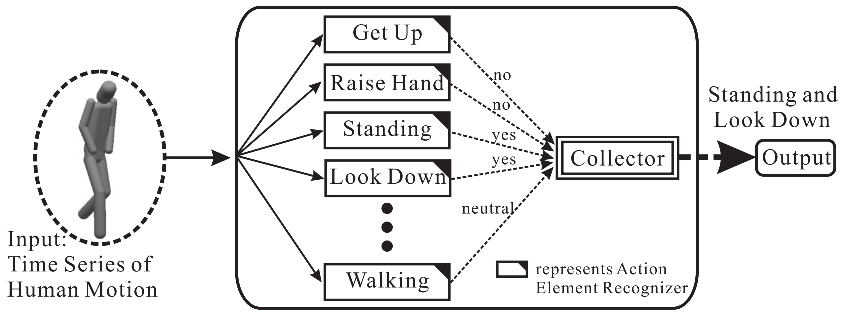


Fig. 3. Configuration of recognition system

3.3. *AER configuration*

A time series of human motion is the input of each AER. An AER outputs the recognition result whether the assigned action occurs or not per frame in synchronized with the input motion. An AER in our former system outputs a matching value with range 0 to 1 by utilizing fuzzy logic, so that the output of the AER gives the system the unclarity associated with humans. But this causes difficulties in preparing referential recognition results of humans.

Hence, in the newly proposed AER, the output consists of multiple classes that represent not only decisive but also inexplicit result, so that humans can easily make the databases which contain humans recognition results. Specifically, the number of the categories is three. One is the category named “yes” which represents that the assigned action clearly occurs. Another is named “no” which represents that the opposite meaning of “yes”. The last one represents the unclarity of recognition result called “neutral”. Thus, the proposed AER is a multi-class (three classes) classifier.

In general, the framework of the multi-class classifier is summarized as following three types. One of the methods consists of “one vs. one” classifier. This requires three binary classifiers. Another one is built as “one vs. the other” classifier. This method also requires three binary classifiers. The last one is categorized into K nearest neighbors method.

In spite of the above technique, the proposed AER requires two binary classifiers and outputs the integrated result of two binary values. Specifically, the one binary classifier judges whether "yes" or non-"yes", the other judges whether "no" or non-"no". The reason that we have adopted this composition is that the "neutral" category rarely happens for some actions. In other words, humans can explicitly discriminate motion as sitting down when watching someone standing then sitting.

3.3.1. *Binary classifier in AER*

In brief, each of two binary classifiers in an AER recognizes input human movement by utilizing some templates that represent a time series of human motions acquired in advance. In other words, the binary classifier utilizes a template matching technique. Specifically, the output is calculated based on the similarities between the input motion and the templates with humans reference.

The system utilizes the expressions of action described by humans, in order that the AER can take full advantage of the excellent power of feature selection and extraction by humans. More specifically, the system selects the input motion and transforms it into adequate features for AER.

The expressions of action hand-written by humans are divided into three categories in terms of the manner of conversion from the input motion to the features. The three categories are listed as follows.

- **Status**
  For example, the phrase "*the position of the head is high*" is categorized into this type. In this type of expression, the input motion is normalized by some reference value, such as body height. Finally, this normalized value is converted to the input feature with range from 0 to 1 by some scaling factors. This process makes the system easy to deal with variant types of the input motion, such as angle, velocity and height.
- **Transition**
  This represents the transition of the status of some motion with time. For example, the phrase "*the height of the head goes down*" is categorized into this. In this type, after the input motion is converted to the normalized value in the same manner as **Status**, the converted value for a certain period is selected as the input feature. For example, 13 samples of height of head, which are sampled from a certain time during 2 seconds at 6 Hertz, are selected as the input feature.
- **Iteration**
  This represents the repetition of some event that occurs in input motion. For example, the phrase "*each foot contacting on the ground*" is categorized into this way. This process converts input motion to the input feature by

the frequency domain in order to strengthen the iteration of motion.

In general, a recognition system changes the recognition algorithm by considering the difference in the treatment of the input motion in terms of time series. The several categories of expressions are generated by the difference in the treatment of the input motion in terms of time series. Thus, there is a the problem with how the way of recognition differs with the type of the expressions arises. Specifically, the different types of the expressions cause the corresponding template matching method. In the worst case, the way of the learning process is different because of the types of AER.

In order to reduce and eliminate such differences caused by several types of expressions and to unify the method of recognition and learning in all the AERs, the binary classifier in the proposed AER utilizes the kernel technique[6]. The kernel serves as the function that mathematically calculates similarity between one sample and another sample. This has drawn attention in the field of statistical machine learning community. We set the binary classifier in the proposed AER as the kernel classifier, and the kernel technique enables the binary classifier to deal with the three types of expressions uniformly.

3.3.2. *Binary classifier as kernel classifier*

From here, a kernel classifier is introduced as the binary classifier in the AER. We denote by $\boldsymbol{x}$ the time series of input motion. $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ are the input-output pairs in total l frames, where $\boldsymbol{x}_i$ represents i th frame sample motion and its corresponding reference binary (e.g. "yes" or non-"yes") signal by y_i. We can write by α_i the coefficients whose value is proportional to the importance of the templates. The similarity value between the input motion and one template is represented by Kernel $K(\boldsymbol{x}_i, \boldsymbol{x})$. By utilizing these notations, the mapping between the input and the output of the binary classifier in the AER f can be written as

$$f(\boldsymbol{x}) = \mathrm{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \tag{1}$$

The parameter b depicts the bias value and the function $\mathrm{sgn}(\cdot)$ is a step function where the relation of input-output is as follows.

$$\mathrm{sgn}(t) = \begin{cases} +1 \text{ if } t > 0 \\ -1 \text{ if otherwise} \end{cases} \tag{2}$$

3.3.3. *Deriving kernels: combination of kernel values per expression*

The proposed method derives the kernel value in the classifier as the products of all the kernel values corresponding to the similarity in each expression. Specifically, the kernel value that corresponds to the similarities in j th expression $K_j(\cdot,\cdot)$ can be written as

$$K_j(\varphi_j(\boldsymbol{x}_i^{(j)}), \varphi_j(\boldsymbol{x}^{(j)})), \tag{3}$$

where $\boldsymbol{x}^{(j)}$ denotes the selected input motion in the j th expression, and $\varphi_j(\cdot)$ represents the converter from the selected input motion to the input feature.

When the numbers of the expressions in the target action is d, the kernel value in the target action $K(\cdot,\cdot)$ can be written as

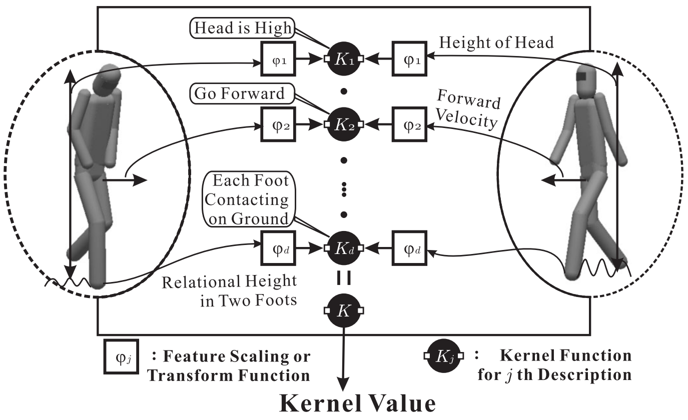
$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \prod_{j=1}^{d} K_j(\varphi_j(\boldsymbol{x}_i^{(j)}), \varphi_j(\boldsymbol{x}^{(j)})). \tag{4}$$

3.3.4. *Example: deriving kernel in walking*

The derivation process of the kernel value in walking is as follows. The relations between expressions, selected motions, and the type of expressions are listed in the Table 1. The image of the derivation process is depicted in the Fig. 4.

Table 1. Example: information for recognizing walking

Expressions	Selected Motions	Type of Expressions
Position of Head is High	The height of the head	Status
Moving forward	Forward velocity	Status
Each foot contacting ground	Relative height between right and left legs	Iteration


Head is High
Height of Head
Go Forward
Forward Velocity
Each Foot Contacting on Ground
Relational Height in Two Foots
φ1
φ2
φd
K1
K2
Kd
K
φj : Feature Scaling or Transform Function
Kj : Kernel Function for j th Description
Kernel Value
Fig. 4. Example: Derivation process of the kernel value in the binary classifier of the walking AER.

4. Learning Process of AER

4.1. *General outline of learning process*

The learning process in the AER tunes the coefficients (α_i, b) of the binary classifiers from the training data, that contain the input-output pairs. The things we have to pay attention to in the learning process are listed as follows.

- Generalization ability
- Incremental (Online) learning ability
- Model selection in terms of kernel
- Computational cost of learning and recognition process

Even though there are several types of learning processes for the kernel classifier proposed in the statistical machine learning community, such as Gaussian process[7], relevance vector machine[8] and Support Vector Machine (SVM)[9], SVM is utilized as the basic learning scheme of our proposed method. This is because no kernel classifiers except SVM can optimize their performance incrementally by utilizing the same criterion as the batch process. In the following sections, the detailed explanations of the learning process are described. Firstly, the batch SVM learning process is introduced as the basic scheme of our learning process. Secondly the online learning algorithm based on SVM is described. Finally, the kernel parameters optimization technique which serves as the part of the model selection problems is explained.

4.2. *Learning scheme based on SVM*

4.2.1. *Batch learning algorithm*

If some adequate motion samples with humans' judgment which contain "yes", "neutral" and "no" can be acquired, the batch learning process can be executed to optimize the weighting and the bias parameters. Specifically, this algorithm optimizes the weighting parameters α_i and the bias parameter b from the motion samples with humans judgment $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ and the kernels given previously. This utilizes the criterion so called "margin" criterion which represents the degree of separation between one category and the other. The detailed derivation of this algorithm is described by Vapnik[10].

The optimization algorithm in this learning process is derived from the quadratic programming problem of weighting parameters $\alpha_i,\ i = 1, \ldots, l$, and can be written as

$$\begin{aligned}
&\text{minimize :} \quad W(\boldsymbol{\alpha}) \equiv \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{l}\alpha_i + b\sum_{i=1}^{l}\alpha_i y_i \\
&\text{subject to :} \quad \begin{cases} \displaystyle\sum_{i=1}^{l}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C,\ \forall i = 1, \ldots, l \end{cases},
\end{aligned} \tag{5}$$

where the constant positive number C penalizes the training error in the non-separable case.

This QP problem makes the solution have a number of weighting parameters equal to zero. Since there is a weighting parameter α_i associated each of the motion templates $\boldsymbol{x}_i$, only the templates corresponding to non-zero α_i (the "support vectors") will influence the output of the classifier.

4.2.2. *Online learning algorithm*

The online learning algorithm relies on the same scheme as the batch learning of SVM mentioned above. The online learning algorithm we use can be applied, only if optimized binary kernel classifiers are given in advance and the additional motion with human judgment is input. This algorithm was originally proposed by Causwenberghs et al.[11]. It basically uses the Karush-Kuhn-Tucker (KKT) condition in the batch learning of SVM QP problem in order to avoid the over-fitting problems which often occur in many other online learning algorithms. The optimal condition used in this algorithm is denoted as follows, which is derived by KKT optimal condition.

We define by $g(\cdot)$ the derived function of the performance function W in the Eq. (5)

$$g(\boldsymbol{x}_i) \equiv \frac{\partial W}{\partial \alpha_i} \tag{6}$$

$$= y_i f(\boldsymbol{x}_i) - 1 \tag{7}$$

Then, the relation between the weighting parameter α_i and the function $g(\boldsymbol{x}_i)$ should be written as

$$g(\boldsymbol{x}_i) = \begin{cases} > 0 \text{ if } \alpha_i = 0 \\ = 0 \text{ if } 0 < \alpha_i < C \\ < 0 \text{ if } \alpha_i = C \end{cases} \tag{8}$$

If we consider that the optimized binary classifier trained with data $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ is given, we denote by α_i^* the optimal weighting parameters. Next, the new labeled sample denoted by $\{\boldsymbol{x}_c, y_c\}$ is added to the classifier, then the value of derived function $g(\boldsymbol{x}_c)$ is calculated as the first step of the online learning. If the value of $g(\boldsymbol{x}_c)$ is equal to or less than zero, the corresponding weighting parameter should be more than zero, which is derived from Eq. (8). In other words, the additional motion should be one of the support vectors.

In the case that the additional motion must be a support vector, the iterative process is performed as follows. The iterative process increases the value of the coefficients α_c in incremental steps to keep the Eq. (8) in the previous training data, until additional training data satisfies the Eq. (8). The change of the other coefficients by α_c growth is derived by equilibrium status based on KKT condition. The detailed information for deriving the equilibrium relations is described by Causwenberghs et al.[11].

4.2.3. *Kernel parameters optimization*

It is a fundamental premise that the kernel types and their parameters are priori given in the learning process of any kernel classifiers, and the performance is surely dependent on the kernel types and their parameters. SVM performs better than other classical learning algorithms in terms of the performance for many applications. However, the performance fails to provide high accuracy when some poor type of kernel and its parameters are set. In this paper, the kernel parameters optimization scheme is adopted as the part of the solution for these kind of model selection problems.

A generalization error of a SVM can be written as the function of the parameter $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$, the kernel parameters, by $T(\boldsymbol{\theta})$. Finally, we denote by $\boldsymbol{\theta}^*$ the optimized kernel parameters as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} T(\boldsymbol{\theta}) \tag{9}$$

In order to realize the kernel optimization scheme easily and robustly, the following list is a summary of what needs to be considered.

- Utilizing an effective searching algorithm in the space of kernel parameters.
- Using an accurate estimator of the performance of SVM computed with a small amount of calculation

In this paper, the gradient descent algorithm is utilized as the effective search engine. Thus, the general outline of the kernel optimization algorithm is listed as

```
(1) Initialize $\boldsymbol{\theta}$ with some values.
(2) Using a batch SVM algorithm, find the optimal coefficients.
(3) Update the parameter $\boldsymbol{\theta}$ such that $T(\boldsymbol{\theta})$ is minimized.
    This can be written as
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \frac{\partial T(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}.$   (10)
(4) Go to step 2 or stop when the minimum of $T$ appears
    in balance. We can write the terminal condition
    of this step as
        $\left| \frac{\partial T(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right| < \epsilon,$   (11)
    where $\epsilon$ is some positive constant number.
```

There are several good estimators for the performance of SVM. In this research, we adopt the "Span" technique proposed by Chapelle et al.[12]. This is because the gradient descent technique requires the derived function for kernel parameters and coefficients, and we can explicitly write it if the estimator is based on span bound. The property of the span bound technique has a close relationship with the *Leave-One-Out* cross validation error estimation, whose computational cost is too high but has excellent quality accuracy of estimation.

The estimator based on the span bound is defined as

$$T = \frac{1}{l}\sum_{p=1}^{l} \Psi(\alpha_p S_p^2 - 1), \tag{12}$$

where Ψ is sigmoid function which is denoted as

$$\Psi(t) \equiv \frac{1}{1+\exp(-At+B)},\ A > 0,\ B \geq 0. \tag{13}$$

The parameters A and B are derived with the Platt's process[13]. The variable S_p represents the span bound of training data corresponding to p th support vector and is defined as

$$\Lambda_p = \left\{ \sum_{i \neq p, \alpha_p > 0} \lambda_i \phi(\boldsymbol{x}_i), \sum_{i \neq p} \lambda_i = 1 \right\}, \tag{14}$$

$$S_p^2 = d^2(\boldsymbol{x_p}, \Lambda_p) = \min_{\boldsymbol{x} \in \Lambda_p} (\boldsymbol{x}_p - \boldsymbol{x})^2, \tag{15}$$

where Λ_p represents the linear combination of all the support vectors except the p th support vector training data. Thus, we explain S_p holds the distance between Λ_p and $\boldsymbol{x}_p$. In practice, the computation of S_p is easy enough as

$$S_p^2 = \frac{1}{(\tilde{K}_{sv}^{-1})_{pp}}, \tag{16}$$

where $\tilde{K}_{sv}$ denotes

$$\tilde{K}_{sv} = \begin{pmatrix} K_{sv} & \mathbf{1} \\ \mathbf{1}^t & 0 \end{pmatrix}. \tag{17}$$

We define by K_{sv} the matrix containing the kernel values corresponding to all the support vectors.

5. Performance Evaluation Experiments

We present some experiments for evaluating the performance of the proposed recognition system optimized by the learning process described in the previous section. The evaluation criterion of all the experiments we performed is based on the average frame-wise accuracy of AERs.

The experiments we conducted can be divided into three categories:

- the evaluation of the performance acquired by the batch SVM learning process and comparison with the performance acquired by other classical batch learning algorithms,
- the evaluation of the properties of the SVM based online learning algorithm,
- the evaluation of the properties of the kernel parameter optimization algorithm.

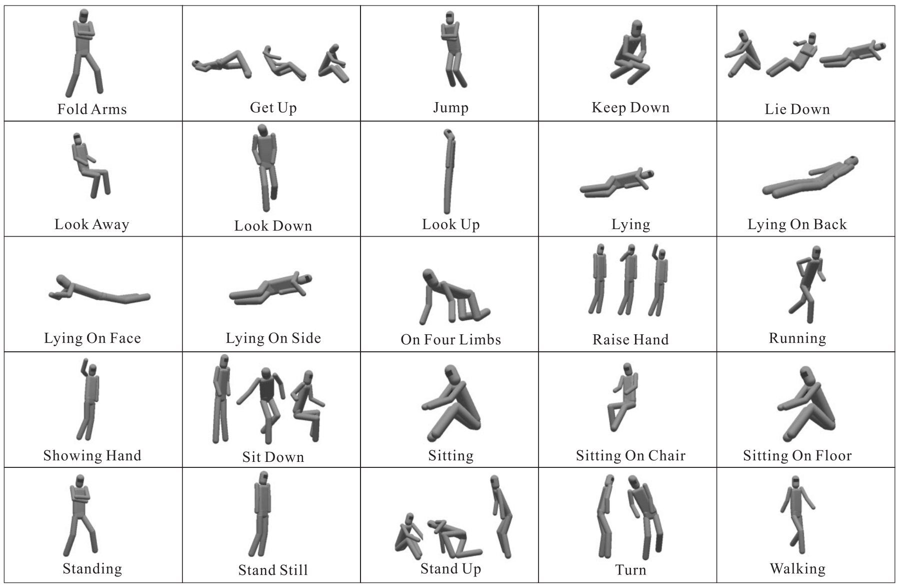


Fig. 5. The 25 daily life actions and their names are listed as the targets in the experiments

5.1. *Target actions and motion data*

For the experiments, we selected 25 actions as the target actions in the experiments which occur indoors and can be measured by the motion capturing system, such as lying and getting up. Fig. 5 depicts all the names and snapshots of the target actions. In each experiment, we have measured motion data by a magnetic motion capturing system. All the motion data we have used in the experiments is incorporated in ICS Action Database[14]. Each motion data in this database contains a BVH formatted motion capture file and its reference files per each target action. One reference file contains judgment by humans for the assigned action per frame by three degrees (“yes”, “neutral” and “no”) .

Specifically, the motion data used in the experiments consists of 5 collections of BVH files. One set contains 25 BVH files and its reference files. In other word, 125 BVH files and 3,125 reference files are used in our experiments. The average span of one BVH file is almost 3 sec. and the total length of all BVH files is 12,126 frames. The actor in all the BVH files is the same male person. Fig. 6 depicts a thumbnail of one BVH file when the actor stands up. The thumbnails are selected per 10 frames, i.e. 0.33 sec.

5.2. *Evaluation of performance by batch learning*

We evaluated the performance of the system optimized by the batch SVM learning algorithm. The evaluation is done by calculating the average accuracy per frame

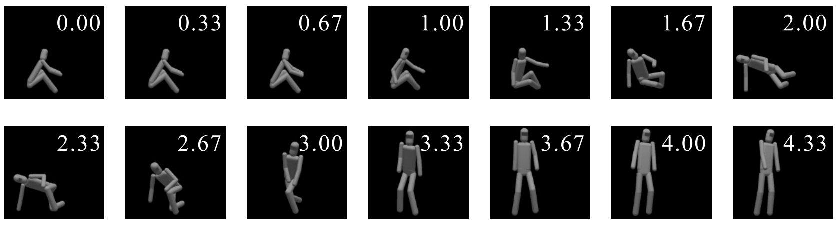


Fig. 6. A BVH file from ICS Action Database where a male stands up from sitting. The thumbnails are selected per 0.33 sec. from the file. The figures shown in the right-top side of each thumbnail indicate seconds from the starting time.

of all AERs when the test motion data are input to the system. We define the accuracy ratio as the percentage of the frames agreeable with the reference and the output. We select one collection of BVH as the training data and four collections as the evaluation data in turn. Then we calculate the average performance from five accuracy ratios.

Here, the prior parameters of SVM utilized in this experiment are described. We utilize radial basis function in all the kernels in all the AERs. We denote by $\sigma > 0$ the kernel parameter, which is the same value in all kernels. If we define by f_d the total number of the features dimension, then the kernel parameter σ is set to $1.5\sqrt{f_d}$ and the penalizing constant number C is $100\sqrt{f_d}$.

Besides the evaluation of the performance of the proposed system, the comparison with multi layered neural networks optimized by back propagation algorithm (BP) and decision trees automatically generated by C4.5[15] is conducted. All the features utilized in these classifiers are same as the features which enters directly to the kernels of the proposed system.

The neural networks configuration in the experiments contains three layers of neurons. The type of the neurons in the input and the output layer is basically a linear function, while the type in the hidden layer is a sigmoid function. The number of the neurons in the hidden layer is empirically set at $2f_d$ to perform better. As the implementation of C4.5 algorithm, we use the software by Quinlan[16].

The average performance per AER optimized by the batch SVM, BP and C4.5 are shown in Fig. 7. The numbers on the horizontal axes of Fig. 7 depicts the ID of the target action name written in Table 2. By the batch SVM, the mean accuracy of all the target actions per frame reaches 97.2%, meanwhile the accuracy reaches only 96.0% in the neural networks by BP and 96.1% in the C4.5 decision trees. Table 2 also shows average and variance of the performance per target action name. This shows that the proposed method has better performance than the other method in the most target action names(21/25).

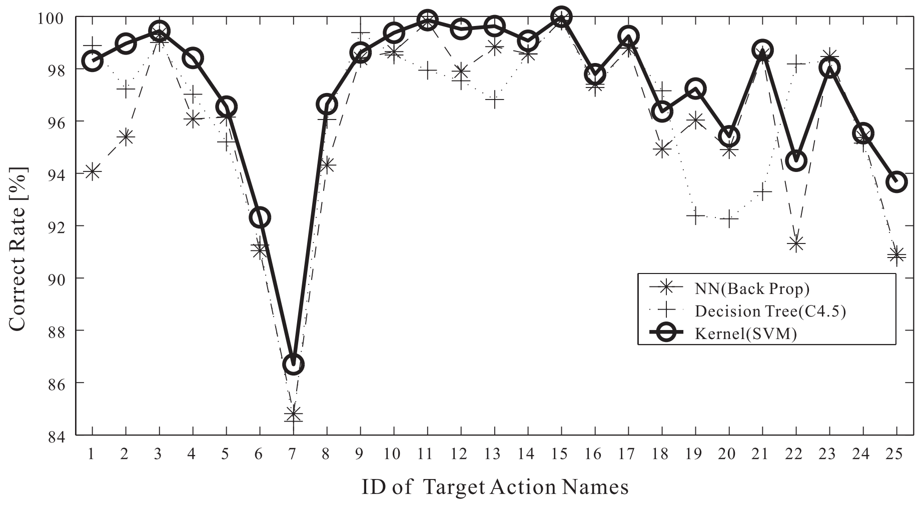


Fig. 7. This shows the average accuracy ratio per AER in each recognition method.

Table 2. This table shows the mean of the accuracy rate and its standard deviation for each learning algorithm (SVM, BP, C4.5) per target action name

Target Action	Kernel Machine(SVM)	NN(Back Prop)	Decision Tree(C4.5)
1. Fold Arms	98.3 ± 2.0 [%]	94.1 ± 5.3 [%]	**98.9 ± 1.6 [%]**
2. Get Up	**99.0 ± 0.6 [%]**	95.4 ± 1.3 [%]	97.2 ± 1.3 [%]
3. Jump	**99.4 ± 0.2 [%]**	99.3 ± 0.0 [%]	99.0 ± 0.6 [%]
4. Keep Down	**98.4 ± 1.4 [%]**	96.1 ± 2.3 [%]	97.0 ± 1.2 [%]
5. Lie Down	**96.5 ± 1.7 [%]**	96.2 ± 0.8 [%]	95.2 ± 2.7 [%]
6. Look Away	**92.3 ± 1.4 [%]**	91.1 ± 2.1 [%]	91.3 ± 2.1 [%]
7. Look Down	**86.7 ± 2.2 [%]**	84.8 ± 2.2 [%]	84.5 ± 1.6 [%]
8. Look Up	**96.6 ± 1.0 [%]**	94.3 ± 2.0 [%]	96.1 ± 2.6 [%]
9. Lying	98.6 ± 0.5 [%]	98.4 ± 0.5 [%]	**99.4 ± 1.3 [%]**
10. Lying On Back	**99.4 ± 0.8 [%]**	98.7 ± 1.9 [%]	98.5 ± 1.9 [%]
11. Lying On Face	**99.8 ± 0.1 [%]**	**99.8 ± 0.1 [%]**	97.9 ± 1.8 [%]
12. Lying On Side	**99.5 ± 0.3 [%]**	97.9 ± 1.8 [%]	97.5 ± 2.8 [%]
13. On Four Limbs	**99.6 ± 0.1 [%]**	98.8 ± 1.0 [%]	96.8 ± 2.5 [%]
14. Raise Hand	**99.1 ± 0.1 [%]**	98.6 ± 0.2 [%]	98.6 ± 0.4 [%]
15. Running	**100.0 ± 0.0 [%]**	99.8 ± 0.0 [%]	99.8 ± 0.2 [%]
16. Showing Hand	**99.8 ± 0.4 [%]**	97.6 ± 0.8 [%]	97.3 ± 1.3 [%]
17. Sit Down	**99.2 ± 0.1 [%]**	98.8 ± 0.1 [%]	98.8 ± 0.1 [%]
18. Sitting	96.4 ± 0.8 [%]	94.9 ± 1.6 [%]	**97.2 ± 4.5 [%]**
19. Sitting On Chair	**97.2 ± 1.2 [%]**	96.0 ± 1.6 [%]	92.4 ± 1.8 [%]
20. Sitting On Floor	**95.4 ± 1.9 [%]**	94.9 ± 2.1 [%]	92.3 ± 3.3 [%]
21. Standing	**98.7 ± 0.5 [%]**	98.5 ± 0.4 [%]	98.3 ± 0.8 [%]
22. Stand Still	**94.5 ± 2.4 [%]**	91.3 ± 2.5 [%]	93.3 ± 2.0 [%]
23. Stand Up	98.0 ± 0.1 [%]	**98.5 ± 0.2 [%]**	98.2 ± 0.5 [%]
24. Turn	**95.5 ± 0.6 [%]**	95.4 ± 0.7 [%]	95.2 ± 1.1 [%]
25. Walking	**93.7 ± 1.4 [%]**	90.9 ± 0.5 [%]	90.8 ± 2.0 [%]

5.3. *Evaluating properties of online learning*

In this experiment, we evaluated the property of the online learning algorithm mentioned above. We compared the performance and some parameters of the recognition system acquired by the batch SVM with those by the online SVM. Specifically, we compared the performance, the number of the support vectors and the sum of all the coefficients of the classifier, which can be written as $\sum_{i=1}^{l} \alpha_i$. The reason why we evaluated the sum of the weighting parameters is that it represents the relations of the margin between two classes. The two types of AERs to be compared are described as follows. The former is acquired only by the batch learning algorithm from two sets of BVH files. The latter is acquired by the online SVM with one training data set, after the batch learning process with one collection of training data.

The experimental result shows that the average difference in the performance between two classifiers is 0.1%. The number of support vectors by the incremental SVM is 1.13 times those of the batch SVM. Thus, we consider this online learning algorithm has the similar properties as the batch learning algorithm.

In fact, the number of the target actions we can actually evaluated in this experiment was 19. This is because the online learning process is broken by the numerical unsteadiness of the iterative steps for the other actions (i.e. six of them) such as turn and look away. Specifically, the unsteadiness occurs at the inversion of the matrix where each component represent the product of the kernel value and the two labels. We will implement a recursive updating of the inversion of the matrix which will avoid this unsteadiness, instead of calculating the inversion directly.

5.4. *Evaluating property of kernel parameters optimization*

An experiment to evaluate the property of our kernel parameters optimization was conducted. Specifically, the performance and the changes in the number of the support vectors are the criteria for the evaluation. In this experiment, we selected 8 actions such as folding arms and lying to satisfy all the conditions listed as follows.

- Actions whose accuracy cannot reach 99%.
- Actions where the types of the expressions consists only of “Status”.
- Actions whose classifier is NOT burdened by the improper reference signal and feature selection from the expressions.

This experiment was conducted as follows. The kernel parameters optimization executes with one collection of the motion set, which is exactly the same as the training set in the experiment of the batch learning. Next, the performance evaluation was performed with 4 collections of the motion data. Then we compared the number of the support vectors in these classifiers with that in the classifiers acquired by the experiment in the batch SVM.

The experimental results show that the performance increases only 0.4%, while the number of the support vectors shows a dramatic 34.2% decrease.

6. Conclusion

We designed and implemented a recognition system for daily human life action. The design principles of our former system were the simultaneous recognition, expressing the unclear result in recognition, and utilizing expressions of action by humans knowledge in order to achieve a recognition result close to that of a human. The proposed system in this paper improves on the above features. The modification of the recognition process to incorporate learning ability to the system is the main feature reported in this paper.

Motion capture files are utilized as the time series of input motion of the proposed system. In order for the system to output multiple recognition results simultaneously, it consists of the recognition processes (AER), as many as the number of the target actions , with AERs running in parallel.

The AER which contains two binary classifiers, is designed to output multi-class decision results of one action to realize the unclarity in recognition. The binary classifier utilizes voting framework as a linear combination of similarities between input motions and the sample motion data acquired in advance. By utilizing a kernel technique to calculate similarity, the recognition and learning process can be unified to reduce the differences caused by the multi types of expressions of action by humans.

The learning scheme we adopted is based on Support Vector Machine, which can execute with the training data (motion and humans judgment). In this paper, the extended learning algorithms of SVM, which can be computed in the online situation, and kernel optimization as a part of model selection of kernel classifiers, are applied to the recognition system.

The experimental results show that the performance of our system is superior to other classical classifiers such as the multi layered neural network optimized by a back propagation algorithm, and the decision tree generated by a C4.5 algorithm.

We also show that the property of the online learning is similar to batch SVM algorithm. The result of the evaluation of the property of kernel parameters optimization showed that it effectively reduces the number of Support Vectors.

In the future we intend to make an algorithm to integrate the online learning algorithm and the kernel optimization algorithm. This is because that the parallel execution of two algorithms has the potential to be a more user friendly automatic optimization process and guarantee the optimality of its performance.

We also plan to design a new kernel function which can more effectively incorporate the expressions of action and to develop a new learning algorithm incorporating prior knowledge of action. The expansion of target actions and the construction of a knowledge database of action are also important.

References

1. T. Starner and A. Pentland, Visual Recognition of American Sign Language Using Hidden Markov Models, in *International Workshop on Automatic Face and Gesture*

Recognition (FGR) (IEEE Press, Zürich, Switzerland, 1995), pp. 189–194.
2. A. Wilson and A. Bobick, Parametric Hidden Markov Models for Gesture Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(9), 884–900(1999).
3. T. Mori, K. Tsujioka and T. Sato, Human-Like Action Recognition System on Whole Body Motion-captured File, in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, Hawaii, USA, 2001), pp. 2066–2073.
4. C. Cédras and M. Shah, Motion-Based Recognition: A Survey, *Image and Vision Computing* **13**(2), 129–155(1995).
5. BVH File Format, http://www.biovision.com/bvh.html.
6. B. Schölkopf and A. Smola, *Learning with Kernels* (MIT Press, Cambridge, Massachusetts, 2002).
7. D. Mackay, Gaussian Processes: A Replacement for Supervised Neural Networks?, *Lecture notes for a tutorial at NIPS 1997.*
8. M. Tipping, Sparse Bayesian Learning and the Relevance Vector Machine, *Journal of Machine Learning Research* **1**, 211–244(2001).
9. C. Cortes and V. Vapnik, Support-Vector Networks, *Machine Learning* **20**(3), 273–297(1995).
10. V. Vapnik, *The Nature of Statistical Learning Theory (Statistics for Engineering and Information Science)* (Springer Verlag, 1995).
11. G. Cauwenberghs and T. Poggio, Incremental and Decremental Support Vector Machine Learning, in Todd Leen, Tom Dietterich and Volker Tresp(eds.), *Advances in Neural Information Processing 13* (MIT Press, Cambridge, Massachusetts, 2001), pp. 409–415.
12. O. Chapelle and V. Vapnik, Model Selection for Support Vector Machine, in S. Solla and T. Leen and K. Müller(eds.), *Advances in Neural Information Processing Systems 12* (MIT Press, Cambridge, Massachusetts, 2000), pp. 230–236.
13. J. Platt, Probabilities for SV Machines, in A. Smola, P. Bartlett, B. Scholkopf and D. Schuurmans(eds.), *Advances in Large Margin Classifiers* (MIT Press, Cambridge, Massachusetts, 1999), pp. 61–74.
14. ICS Action Database, http://www.ics.t.u-tokyo.ac.jp/action/.
15. J. Quinlan, Induction of Decision Trees, *Machine Learning* **1**(1), 81–106(1986).
16. C4.5 Release 8, http://www.cse.unsw.edu.au/ quinlan/c4.5r8.tar.gz.

Taketoshi Mori received his M.E. and Ph.D. degrees in Information Engineering from the University of Tokyo, in 1992 and 1995, respectively. From 1995 to 1998, he was a Research Associate at Research Center for Advanced Science and Technology of the University of Tokyo. From 1998, he was a Lecturer at the university. Now, he is an Associate Professor at the university. His research interests include recognition, robot vision, image processing, and pervasive computing. Ph.D. Mori is an active member of the IEEE Robotics and Automation Society, the IEEE Computer Society, the Robotics Society of Japan, the Japan Society of Mechanical Engineers, the Institute of Electronics, Information and Communication Engineers, and Japanese Society for Artificial Intelligence.

Masamichi Shimosaka received his B.E. and Master of Information Science and Technology from the University of Tokyo, in 2001 and 2003, respectively. Now, he is a Ph.D. candidate at Graduate School of Information Science and Technology, the University of Tokyo. From 2004, he is also a Research Fellow of the Japan Society for the Promotion Science. His research interests include machine learning, perception, and kinematics, dynamics, and robot vision. He is a student member of the IEEE and the Robotics Society of Japan.

Tatsuya Harada received his M.E. and Ph.D. degrees in Mechanical Engineering from the University of Tokyo, in 1998 and 2001, respectively. From 2000 to 2001, he was a Research Fellow of the Japan Society for the Promotion Science. From 2001, he is a Research Associate at Graduate School of Information Science and Technology, the University of Tokyo. His research interests include human motion observation, modeling, perception, sensors, and embedded systems. Ph.D. Harada is an active member of the IEEE, the Robotics Society of Japan, and the Society of Instrument and Control Engineers.

Tomomasa Sato received the B.S., M.S. and Ph.D. degrees in mechanical engineering from the University of Tokyo, Japan, in 1971, 1973 and 1976 respectively. Since 1976, he has been with the Electrotechnical Laboratory (ETL) of the Ministry of Industrial Science and Technology. In 1991, he moved to the Research Center for Advanced Science and Technology (RCAST) of the University of Tokyo. From 1998, he is currently a Professor of the Department of Mechano-Informatics of the university. His current research interests include intelligent machine, human symbiosis robot and environmental type robot systems. Ph.D. Sato is an active member of the IEEE Robotics and Automation Society, the Japan Society of Mechanical Engineers, the Society of Instrument and Control Engineers, and the Robotics Society of Japan.