Paper

Speed up in Computation of HMAX Features for Object Localization

Takuya Minagawa†, Hideo Saito (member)†

Abstract While HMAX features have been proved to have excellent performance in image categorization tasks, the computational cost of recognition is expensive. If we aim to apply the HMAX features to object localization tasks, in which the categorization tasks are repeatedly performed by sliding windows, their processing time increases enormously. In this paper, we propose a method for speed up in computation of object localization based on HMAX features. We found that the HMAX features cause specific redundancies in the sliding window approach. The speed up is achieved by eliminating the redundancies in our method. The results from experiments using the University of Illinois-Urbana-Champaign (UIUC) car dataset and the face detection dataset benchmark (FDDB) indicate that this modification improved processing speeds significantly with insignificant reductions in precision.

Key words: object recognition, object localization, HMAX, bags-of-features, feature descriptor, speed up.

1. Introduction

Understanding the content of an images is a key technology in the analysis of multimedia content. Image categorization and object localization tasks have been studied extensively for that purpose in the last decade: the image categorization tasks identify “what” there are in images, and the object localization tasks search “where” target objects are in images. Many types of image feature descriptors have been proposed for both tasks: global features called Gist, the bags-of-features (BoF), and other BoF-like descriptors for categorization tasks, and Haar-like features, histogram of oriented gradients (HOG), and others for object localization tasks.

The features generated by HMAX model are also one of the efficient descriptors for image categorization tasks. HMAX model was originally proposed by Riesenhuber and Poggio inspired by the behavior of simple and complex cells in a visual cortex studied by Hubel and Wiesel. The HMAX model were expanded to recognize real world-image by Serre et al. and proved to have competitive performance in image classification tasks. They have also been used for object localization tasks using sliding windows and have demonstrated excellent performance. However, the computational cost of HMAX is so high that object localization with sliding windows takes vast amounts of time.

The main contribution of this research was to improve the processing speed of the HMAX model in object localization tasks. Some expansions of the HMAX model have succeeded in reducing its processing speed in categorization tasks. Of course, the speed-up of categorization tasks also improves the speed of localization tasks; however, some redundancies still remain in the sliding window approach. The redundancies are derived from the natures of HMAX computation: convolutions, max-pooling, and multi-scale features. Our approach focused on eliminating these “HMAX specific” redundancies of the sliding window approach. The sliding window approach estimates existence of target objects at each position in images. On the other hand, our method searches the similar image regions to the shapes that the object has. This idea enabled to narrow the search area from coarse to fine and avoid duplicate processes in overlapping regions between sliding windows and in overlapping scales.

This paper is organized as follow. Section 2 explains HMAX model, conventional speed-up approaches of HMAX, and the redundancies of HMAX in the sliding window approach. A description of our improvements to HMAX for object localization is given in Section 3. Section 4 then describes evaluations of our proposal, where we focus on comparing our method to a sliding window approach with HMAX and clarify the impact of our improvements. We discuss limitations with our approach and present some ideas on improvements in...
Section 5. Finally, Section 6 summarizes and draws conclusions on our approach.

2. Related Work

2.1 HMAX Model

The HMAX model\(^1\)\(^2\) has a hierarchical structure of four-layers (S1, C1, S2, and C2). An input image is processed from the S1 to the C2 layer and an output of each layer is an input to the next layer. Finally, an output of the C2 layer is a feature vector for classification; we call this C2 vector HMAX features in this paper.

1) S1 layer

An input image is processed at a unit on the S1 layer by a 2D Gabor filter with orientation \(\theta_o\), wavelength \(\lambda_s\), and effective width \(\sigma_s\):

\[
F(x, y, s, o) = \exp\left(-\frac{x^2 + y^2}{2\sigma_s^2}\right) \times \cos\left(\frac{2\pi x x_0}{\lambda_s}\right) 
\]

\[
x_0 = x \cos \theta_o + y \sin \theta_o, y_0 = -x \sin \theta_o + y \cos \theta_o
\]

The arguments \(s\) and \(o\) correspond to an index of a scale band and an orientation. If 16 scale bands and four orientations (\(\theta \in \theta^1, 45^\circ, 90^\circ, 135^\circ\)) are used then 64 S1 values, which correspond to one pixel, are generated.

2) C1 layer

The C1 layer adds invariance of scale and position to signals from the S1 layer by max pooling as:

\[
r_{c1}(x_{c1}) = \max_{x_{s1} \in R(x_{c1})} r_{s1}(x_{s1})
\]

where \(r_{s1}(x_{s1})\) is a response value of the S1 unit at position \(x_{s1}\) and \(r_{c1}(x_{c1})\) is that of the C1 unit at position \(x_{c1}\). Position vector \(x\) consists of coordinates \((x, y)\) and scale \(s\). Each processing unit on the C1 layer passes the maximum signal value from the S1 layer in region \(R(x_{c1})\) that is defined by the \(N_{c1} \times N_{c1}\) area in neighboring \(\Delta S_{c1}\) scale bands at each orientation. Max-pooling areas \(N_{c1}\) are overlapped with ratio \(\Delta N_{c1}\).

3) S2 layer

An S2 unit reacts to a signal similar to the pre-defined feature patches that are randomly selected regions of C1 output from training images. These \(N_p\) patches are defined as a feature dictionary here. The response of the S2 unit is calculated from the distance between an input and each patch by using the following radial basis function (RBF):

\[
r_{s2}^n = \exp(-\beta ||X - P_n||^2)
\]

where \(X\) is part of a signal generated by the C1 layer, \(P_n\) is the \(n_{th}\) feature patch, and \(\beta\) is the sharpness of reaction. \(X\) and \(P_n\) are vectors that have \(N_{s2} \times N_{s2} \times D\) elements when \(D\) is the number of orientations \(o\) and \(N_{s2}\) is the patch size.

The S2 layer is the most time consuming process because RBF must be computed for all patches at each position and scale. This work and the previous work approached this problem by reducing the number of patches to be calculated\(^{19}\)\(^\)\(^2\)\(^1\)\(^3\)\(^\)\(^2\).\(^1\)

4) C2 layer

The C2 layer integrates the S2 outputs of all positions and scales by taking the maximum one. Therefore the response of the C2 layer is an \(N_p\)-dimensional vector in which each element represents the signal most similar to each feature patch over all scales and positions. \(N_p\) is the number of feature patches.

Finally, this vector is used for training and discrimination by a machine learning algorithm to recognize the object category.

As seen above, the HMAX model convolves shape filters at each scale and position to obtain similarities to each shape, which takes long time to calculate. As we reported\(^2\), it takes more than 10 sec to process a quarter video graphics array (QVGA) size image with 1000 feature patches.

2.2 Speed Up Approaches

There are four types of approaches to the HMAX model that reduce processing time, i.e., make a feature descriptor faster, eliminate redundant features, restrict regions to calculate, and parallel its processing on hardware.

The integral techniques are popular approaches to make descriptors faster such as the integral images\(^5\)\(^9\)\(^3\), the integral histograms\(^9\)\(^3\) and the integral channel features\(^9\)\(^3\). Previous work have approximated Gabor filters on the S1 layer as haar-like features that can be calculated quickly using the integral images\(^9\)\(^3\)\(^2\). Chikkerur and Poggio also surveyed other approximation approaches to S1 and S2 feature descriptors such as 1-D separable filters, row rank 2-D matrix with singular value decomposition (SVD), and filter reduction with principal component analysis (PCA)\(^9\)\(^3\).

A popular method of reducing the number of features is to use Adaboost to select important features\(^9\)\(^1\)\(^2\)\(^9\)\(^9\)\(^2\)\(^8\)\(^9\)\(^1\).\(^2\)\(^6\)\(^9\)\(^8\)\(^9\)\(^9\)\(^6\)\(^9\)\(^8\). Huang et al. applied Adaboost to select important S2 feature patches of the HMAX model for recognition\(^9\)\(^8\)\(^9\)\(^8\). Minagawa and Saito\(^8\)\(^9\) adopted a clustering approach to reduce the number of features.

Key point detectors have been used to find salient regions and to restrict the number of points to extract image feature descriptors\(^7\)\(^8\)\(^9\). The same kind of idea
could be applied to the HMAX model, viz., to extract the points of high gradients or contrasts to compute features\(^{19,20}\). This approach has been proved to have great benefits in speeding up computation. Although Chikkerur and Poggio tried random sampling of computation points at the S2 layer, this did not work well for large images\(^{23}\).

Another idea for speeding up computation is to use a parallel processing architecture. Mutch et al. implemented a cortical network simulator (CNS) that was a framework on graphics processing unit (GPU) for a cortex like model that included HMAX\(^{22}\).

2.3 Object Localization Using HMAX

The procedures of the sliding window approach with HMAX follow the steps below:

1. Set windows over all positions of an input image.
2. Trim the image regions in the windows.
3. Process the trimmed images through the S1 to the C2 layer.
4. Predict existences of the object by a machine learning algorithm using the C2 features.

In multi-scale object localization tasks, the input image is resized larger or smaller and processed with the same sliding window size in the same steps above. Notice in step 3 that input images are decomposed to multi-scale information at the S1 layer, and all scale information are gathered into a single scale at the C2 layer. The conventional HMAX based localization approaches have taken this approach\(^{17-19}\).

The methods of speeding up HMAX described in the previous section are mainly for image categorization tasks. Of course they can also contribute to localization tasks; however, the sliding window approach using HMAX still has three redundancies as seen in Fig.1 (see red words). First, the overlapping areas between sliding windows are computed by duplicate processes through the S1 to the C2 layer. Second, as HMAX computes an image by multi-size convolution filter, the process may be duplicated for multi-scale localization. Finally, max-pooling at the C2 layer of HMAX only takes the maximum similarity value to each patch over all scales and ignores the rest, which means information on patch size is discarded. Unlike the categorization task, a localization task should make use of scale information to recognize an object’s size. Thus the max-pooling over scales is unnecessary for localization tasks. These redundancies are derived from natures of the HMAX computation which have convolutions, multi-scales, and max-pooling.

Eliminating overlapping areas and scales in Fig.1 contributes to speed-up directly by avoiding duplicate processing. We took a simple approach to eliminating the overlapping areas by filtering a whole image region at one time. The overlapping scales are also omitted by sharing processes of between small filters in larger objects and large filters in smaller objects. We also addressed “ignoring shape sizes” in Fig.1 by avoiding the max-pooling over scales in the C2 layer. It does not directly contribute to speed-up because the computing cost of the max-pooling is not expensive. However, the information on shape sizes can be used to estimate a target’s position and size in a coarse-to-fine manner to reduce processing times.

Although the “Branch-and-bound” technique\(^{29}\) is also the major method of eliminating the calculation area without sliding windows, our approaches are suitable for eliminating the redundancies in the HMAX. In addition, the “filtering a whole image region” and the “coarse-to-fine” techniques are also suitable for similarity based features like the HMAX. For instance, the HOG\(^{10}\) is also similarity based features which compute distances between filters and appearances. The deformable part model\(^{30}\) calculates the HOG features and scores of a support vector machine (SVM) classifier in a whole image without sliding windows, and Pedersoli et al. took advantage of these scores for coarse-to-fine object localization\(^{31}\).

Brosch and Neumann also applied a coarse-to-fine approach to reduce the processing region for their combined features of HMAX and HOG\(^{32}\); the method computed the likelihood of an object’s presence at each sliding window on a coarse scale and then estimated the positions at local peaks of the likelihood on a fine scale.
Thus, their coarse-to-fine approach still had redundancies over positions and scales: the overlapping areas in sliding windows and the duplicate filtering between coarse and fine scales in multi-scale object localization tasks (see Fig.1).

This paper explains how to speed up the HMAX model by avoiding these redundancies in the sliding window approach. Because this paper focuses on localization specific problems with HMAX, our approach is able to be expanded to other HMAX based models referred to in this section\(^\text{18–20(23)}\).

3. Proposed Method

The main idea in our approach to eliminate the redundancies in HMAX for localization tasks is that the image region of a target object should have high degrees of similarity to the shapes that the object has. The idea was inspired by the work of Chikkerur et al.\(^\text{33}\) who created a saliency map to simulate the top down attention of humans by making use of shapes in images. Therefore, our method defines a model of a target as a group of shapes. These shapes are searched from large to small in an image. In other words, smaller shapes are not searched where larger shapes have not been found. This idea of searching similar parts is in contrast to the idea of the sliding window approach, which is to check an object’s existence at each position. Consequently, our model not only avoids overlap redundancies in sliding windows but also restricts the processing area for small shapes.

Fig.2 overviews our approach to detect single scale objects. The left of the Fig.2 represents the flow to search large shapes, and the right represents that for small shapes. The confidence map, \(I_{\text{prob}}\), expresses the probability of a target existing at each position, which is computed from the similarities between the appearance of the image and shapes. The candidate area, \(M\), is calculated by thresholding \(I_{\text{prob}}\). Smaller shapes are only searched above candidate area \(M\). Compared with the sliding window approach (Fig.1), our method: (1) filters a whole image but not each region to avoid duplicate computation in overlapping areas, (2) computes max-pooling in restricted area of single scale but not in whole region of all scales, and (3) narrows the processing regions in fine scales.

3.1 Detection

(1) Object model

The object must be defined as a combination of feature patches \(P_n\) (see section 2.1(3)) at each scale band to search an object with its shape; these patch groups of an object are defined as an object model in this paper. Fig.3 illustrates an example of an object model that has two scale bands. \(P_{G1}\) is a set of feature patches \(P_n\). The shapes of a target are searched from large to small in an image; if similar shapes of large size are not found, then the search for smaller ones is canceled. However, similar areas to feature patches are searched through an image pyramid from low to high resolution in actual implementation while the sizes of patches are fixed, for the purpose of faster processing. Because patches become relatively larger in smaller images, the lower row in Fig.3 represents the search of larger shapes more than the higher row does. The bottom and top in Fig.3 correspond to flows on the left and right of Fig.2.

The four main differences between the basic HMAX model described in Section 2.1 and the object model defined in this section are summarized as follows:

- The basic model creates multi-scale information with multi-scale Gabor filters at S1; our model creates it with image pyramids.
- The basic model processes each scale band simultaneously; our model processes it from a coarse to a fine resolution.
At the first step, similarities between image age pyramids of each position are estimated by the classifier that represents the probability of a target existing at with Eq.(3). At the second step, confidence map of object model requires: Algorithm 1 an object model

Require: an input image \( I \)

Require: an object model \( \Omega \) that has \( l \) scale bands: the \( i_{th} \) scale band has the following properties
- Group of feature patches: \( P_{gi} = [P_{i,1}, \ldots P_{i,N_i}] \)
- Classifier: \( C_i \)
- Threshold: \( T_i \)

Create image pyramids: \( I \rightarrow I_1 \ldots I_l \)

Computing Mask: \( M \leftarrow 1 \)

for \( i = 1 \) to \( l \) do

Resize the mask \( M \) to the same size of \( I_i \)

for all position \( a \) where \( 1 \) to \( M \) do

\( I_i \rightarrow I_{C1} \) on the S1 layer by Eq.(1).
\( I_{C1} \rightarrow I_{C2} \) on the C1 layer by Eq.(2).
\( I_{C1} \rightarrow I_{S2} \) on the S2 layer with \( P_{gi} \), by Eq.(3).
\( I_{S2} \rightarrow I_{C2} \) on the C2 layer by Eq.(6).
\( I_{C2} \rightarrow I_{prob} \) by Eq.(4),(5) of the classifier \( C_i \).

Binarize \( I_{prob} \) by threshold \( T_i \) then update \( M \).

end for

end for

De-noise \( I_{prob} \) by smoothing

Compute peak position \( X_{obj} \) of \( I_{prob} \) by neighbourhood suppression.

return \( X_{obj} \)

- The basic model applies the same feature patches to all scale bands; our model applies different ones to each.
- The basic model takes the maximum response of each patch over all scale bands and categorizes it with a classifier; our model has a classifier at each scale for coarse-to-fine processing.

As a result, the object model can also be assumed to have a basic HMAX model, which only has a single-scale band at the S2 layer, at each scale band.

(2) Flow of localization

The pseudo-code to detect single-scale objects is in Algorithm 1.

Candidate regions \( M \) of the target object are narrowed down from coarse to fine, as shown in Fig.2.

First, an object model that has already been trained (see Section 3.2) must be prepared. Here, object model \( \Omega \) is assumed to have \( l \) scale bands. The \( i_{th} \) scale band of object model \( \Omega \) has group of feature patches \( P_{gi} \), classifier \( C_i \), and threshold \( T_i \) as properties. Then, image pyramids of \( l \) scales, \( I_1 \ldots I_l \), are created from input image \( I \), which are ordered from small to large. The image pyramids are processed from the \( I_{s2} \) to the \( i_{th} \) scale band in masked region \( M \), which is narrowed down at each scale.

Mask \( M \) is updated at the \( i_{th} \) scale in three steps.

At the first step, similarities between image \( I_i \) of pyramids and all patches \( P_{ij} \) in group \( P_{gi} \) are computed with Eq.(3). At the second step, confidence map \( I_{prob} \) that represents the probability of a target existing at each position is estimated by the classifier \( C_i \). Finally, candidate region \( M \) is updated by binarizing \( I_{prob} \) with threshold \( T_i \). Only regions over the threshold are used for processing at the next \( i+1 \) scale band. If \( i \) is equal to final scale band \( l \), the peaks of confidence map \( I_{prob} \) are extracted by neighborhood suppression\(^{\text{34}} \). Finally, these peaks are assumed to be the positions of target objects.

HMAX features are extracted from resized input image \( I_i \) in the similarity calculation step. The whole image area is processed through the S1 layer to the C2 layer on mask \( M \) to avoid redundancies in overlapping sliding windows. \( I_{S1}, I_{C1}, I_{S2}, \) and \( I_{C2} \) in Algorithm 1 correspond to the outputs of the S1, C1, S2, and C2 layer. The similarities to each patch are computed at the S2 layer and, as described in Minagawa and Saito\(^{\text{20}} \), this is the most time consuming process. Only important patches in a feature dictionary, which has randomly gathered \( N_p \) patches (see Section 2.1), are used to reduce the time for computation. These important patches for recognition are able to be selected with machine learning methods\(^{\text{19}} \). Real Adaboost\(^{\text{35}} \) was adopted in our implementation for two reasons: first, the number of feature patches could be arranged via its number of iterations in training, and second it returned continuous values that were later transferred to probabilistic values. These important patches are grouped as \( P_{gi} \) at each scale band.

The combination of real Adaboost and a logistic sigmoid function is defined as classifier \( C_i \) that is used to compute confidence map \( I_{prob} \) from HMAX output \( I_{c2} \). Eqs.(4) and (5) respectively explain real Adaboost and the logistic sigmoid function.

\[
H(I_{C2}(x, y)) = \sum_{j=1}^{N_i} h_j(I_{C2}(x, y)) \quad (4)
\]

\[
I_{prob}(x, y) = \frac{1}{1 + \exp(-\frac{\alpha}{N_i} H(I_{C2}(x, y)))} \quad (5)
\]

The strong classifier, \( H \), of Adaboost generally returns binary value -1 or 1 for classification tasks by using a sign function; however, the function has been eliminated here to return continuous values from \(-\infty \) to \( \infty \). Here, \( h_j \) is a weak classifier that has been trained with the similarity values of certain feature patch \( P_{i,j} \). The \( N_i \) is the number of feature patches selected in this scale band. The returned value from the strong classifier in Eq.(5) turns to a range from zero to one due to the logistic sigmoid function. The \( \alpha \) is the amplifier to strong classifier \( H \), which we set to 2.0.
(3) Dilate filter for max-pooling
Our approach computes a whole image from the S1 to the S2 layer to avoid duplicate processing in overlapping regions between sliding windows; however, the same procedure cannot be applied into the C2 layer. The C2 layer takes the maximum similarity value of each S2 feature patch over all positions and scales, hence the information of the positions and scales of the object is lost. Thus, the size of max-pooling must be restricted by the object size $W_i$ to sustain the information. This max-pooling process is equal to a dilate filter in morphological operation, which is written as:

$$I_{C2}^j(x, y) = \max_{(x', y') \in R(W'_i)} I_{S2}^j(x + x', y + y')$$

where $R(W'_i)$ is an object region whose center is (0, 0) and width is $W'_i$, which is the object size on the S2 layer at the $i_{th}$ scale. $j$ is the ID of a feature patch.

Fig. 4 illustrates why max pooling is represented by a dilate filter. The ‘+’ at the top of Fig. 4 is the position where feature patch A takes the maximum similarity value in the object region of ‘c’, and ‘-’ is where feature patch B takes the maximum value; therefore, the C2 output at position ‘c’ is a vector that includes the values of A at ‘+’ and of B at ‘-’ as its elements. The bottom left and the bottom right in Fig. 4 correspond to the S2 outputs computed from feature patches A and B (see Eq.(3)). We can see that the values at position ‘c’ are equal to those at the ‘+’ of A and at the ‘-’ of B by processing each S2 output with the dilate filter (Eq.(6)).

The target position on the input image is sustained by this dilate filter at the C2 layer; however, the positions of feature patches in the object region have still been lost. Some expansions to the HMAX model have divided max-pooling regions on the C2 layer to take into consideration the positions of shapes for categorization tasks. Mutch and Lowe\(^{18}\) and Minagawa and Saito\(^{20}\) revealed that considering the positions of features in an object area improved accuracy. They divided an image into $W_{c2} \times H_{c2}$ areas on the C2 layer, and then calculated the maximum value of similarity to each patch in each area. Therefore, the number of feature vector dimensions was expanded from $N_p$, which is the number of feature patches in a dictionary, to $N_p \times W_{c2} \times H_{c2}$; then, important feature patches and their positions were also selected by real Adaboost.

“Divided” max-pooling can also be implemented as dilate filter as well as Eq.(6). Fig. 5 outlines an example of a max-pooling region that has been placed at the top left in a $2 \times 2$ divided object region with overlap ratio $\Delta N_{c2}$; the overlap ratio must be $0 \leq \Delta N_{c2} < 1$. The ‘c’ is the center of the object region and ‘ˆc’ is the center of the max-pooling region. The $cx$ and $cy$ are the coordinates of c from ‘ˆc’. Then, the max-pooling can be assumed to be a dilate filter with size $W''_i$ and anchor point $(cx, cy)$, which is expressed as:

$$I_{C2}^j(x, y) = \max_{(x', y') \in R(W''_i)} I_{S2}^j(x + x' - cx, y + y' - cy)$$

Size $W''_i$ is calculated as:

$$W''_i = \frac{W'_i}{W_{c2} + (1 - W_{c2})\Delta N_{c2}}$$

(4) Multi-scale localization
Algorithm 1 only explains object detection of fixed size, but it is easy to expand our method to multiple sizes. Fig. 6 illustrates the detection of two object sizes. An object model has a parameter, $R_m$, which defines
the shape size ratio in scale bands (see Section 3.2). The object in Fig.6(a) is searched at every scale ratio $R_m$, which is equal to that of the object model; the processing of the second scale of the image pyramid is shared between two sizes of the object. This shared computation between multi-scale objects eliminates the redundancies over scales in Fig.1. The calculation cost does not increase linearly for this reason. Fig.6(b) illustrates a case where the object is searched at every scale ratio, which is smaller than ratio $R_m$ of the object model. An image pyramid is created in this example by the ratio $\sqrt{R_m}$. When a ratio of image pyramid is set to $\sqrt{R_m}, \sqrt{R_m}, ..., a$ scale ratio of the target size to be searched becomes increasingly smaller in the same way.

3.2 Training

This section explains how an object model was built. We wanted to design an object model to reject background regions at lower resolution stages because rejected regions on lower scales are never processed in higher ones. Smaller images are also less expensive to compute. This approach is similar to that for the cascaded classifier\(^8\) and the training method is thereby also similar in three respects. First, training is started from earlier to later stages and each stage has a classifier and a threshold. Second, the threshold of each stage is determined to satisfy the minimum rate for acceptance detection. Third, only negative training samples that are not classified correctly by a trained classifier of this stage are used for training at the next stage.

A pseudo code is given in Algorithm 2.

First, image pyramids of training samples that consist of positives $P$ and negatives $N$ must be created with the number of scale $l$ and the ratio of image pyramid $R_m$. The image pyramids of all training samples must be transformed to the same sizes $W_1, ..., W_l$ at each scale. Then, the classifiers are trained from the smaller $W_1$ to the larger $W_l$ images. The classifier at each scale is trained to satisfy the minimum rate for acceptance detection, $d$. Rate $d$ should be high enough (e.g., 0.99) because the entire detection rate of this model turns out to be around $d^l$.

HMAX features $D_{C_2}$ at scale $i$ are computed from training samples of size $W_i$. Classifier $C_i$ is trained from $D_{C_2}$ by real Adaboost. The group of patch features $P_{g_i}$ is simultaneously selected by boosting. The training images are evaluated with Eqs.(4) and (5) with this classifier $C_i$ to generate the confidence value of each sample. Threshold $T_i$ is determined to satisfy the target detection rate $d$ using the confidence values. Exceptionally, threshold $T_i$ of the last scale, $l$, is always 0.5. Then, only negative samples over threshold $T_i - \epsilon$ are used for training at the next stage, $i + 1$. $\epsilon$ is a value to select negative samples near the boundary. If there are few false positive samples, then the number of negative samples over minimum negative sample ratio $f$ are set to $N_i$.

Finally, the parameters $P_{g_i}, C_i$, and $T_i$ are set to the object model, $\Omega_i$.

Fig.7 presents an example of an object model of
car. The $P_g_1$ and $P_g_2$ are groups of important feature patches, which were selected to discriminate between positive and negative samples by real Adaboost at each scale. Real Adaboost combines these important patches with weak classifiers to estimate likelihood of a target existing.

4. Experiments

The main focus of the experiments was to clarify what effect our improvements had on the sliding window approach of HMAX. First, the performance and processing time for these methods had to be measured with two datasets: 1) the University of Illinois-Urbana-Champaign (UIUC) car dataset and 2) the face detection dataset benchmark (FDDB). Next, the impact of each improvements on our method was evaluated using the UIUC car dataset.

The parameters for HMAX in the experiments followed the empirical environment of Serre et al.\(^3\), i.e., four orientations of the Gabor filter, two scale bands of the S1 layer integrated into one scale band of the C1 layer by max-pooling, and four sizes for the feature patches, $P_n$. We created a dictionary that had 1000 feature patches for each size; therefore, there were a total of 4000 patches in the dictionary. These feature patches were randomly sampled from the Caltech-256 dataset.\(^7\) A total five feature dictionaries were generated in the same way in our experiments.

The criteria for correct detection were the same as the Pascal visual object classes (VOC)\(^1\); the overlap ratio, $a_o$, between the predicted bounding box, $B_p$, and the ground truth bounding box, $B_{gt}$, had to be over 0.5 according to:

$$a_o = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$ (9)

These experiments were carried out in the following environment:

- Machine: Dell Dimension C521
- CPU: AMD Athlon 64 2.20GHz
- RAM: 960 MBytes
- Programming Language: C/C++ with OpenCV\(^8\)

4.1 Evaluation of Performance and Speed

(1) Parameters

We measured the performance and processing time of our method and the sliding window approach of HMAX using the UIUC car dataset and FDDB. The parameters for our coarse-to-fine model were set for each dataset, as summarized in Table 1. The meanings of the nine parameters in the table are as follows:

- $R_m$: The scale ratio of the image pyramid of the object model
- $l$: The number of image pyramid
- $d$: The minimum acceptance detection ratio at each scale for training
- $f$: The minimum negative sample ratio at each scale for training
- $N_t$: The number of iterations to train Adaboost at each scale band
- $W_l \times H_l$: The training sample size
- $N_{s2}$: The sizes of feature patches on the S2 layer
- $W_{c2} \times H_{c2}$: The number of divided area on the C2 layer
- $R_d$: The scale ratio of the image pyramid for detection

(2) Sliding window approach

The sliding window approach samples an image region every four pixels horizontally and vertically, and then the sampled image is resized to the same size as the training samples. This sampling interval was established by considering the C1 max-pooling size and its overlap ratio. Finally, an HMAX feature vector is extracted from the captured image and then scored by real Adaboost and a logistic sigmoid function. Our implementation of the sliding window approach does also eliminated the computation time for HMAX by using only feature patches selected by Adaboost, which is the same as that in our coarse-to-fine method.

The parameters for the sliding window approach were set to those listed in Table 1 as much as possible to enable comparison with our coarse-to-fine method. For instance, training sample size $W_l \times H_l$, the number of C2 layers divided by $W_{c2} \times H_{c2}$, and the scale ratio of the image pyramid to detect $R_d$ were the same for the coarse-to-fine and the sliding window approaches. The number of training iterations for the sliding window was $N_t \times l$, which was equal to the sum of the iterations over all scale bands of the object model.

(3) UIUC car dataset

Our approach in this experiment was compared with the sliding window approach using the UIUC car dataset.
The UIUC car dataset consists of small (100x40) training images of cars and backgrounds, and larger test images in which there is at least one car to be found. There were 550 positive samples and 500 negative samples. There were two sets of test images: a single-scale set in which the cars to be detected were roughly the same size (100x40 pixels) as those in the training images, and a multi-scale set. Because of the size of the training images, only two sizes of feature patches {4x4, 8x8} that had 1000 patches for each were used for this dataset.

We examined the detection task three times with different feature dictionaries. There are examples of our results in Fig.8. The results at the bottom right in Fig.8(a) seem to have shifted slightly from the correct car position. This could happen because of max-pooling especially at the C2 layer. As we did not divide the object region for C2 max-pooling (see Section 3.1(3)) in this experiment, small shift of shape was ignored with dilate filter.

Fig.9 plots the recall-precision curves for both methods. These curves were drawn by changing the threshold of the output value represented in Eq.(5). Only the confidence map, I_{prob}, from the last scale band in the coarse-to-fine approach was used to draw the curve by thresholding. We found that these two curves almost fit each other in each test set. In fact, the recalls at equal-error rates (recall = precision) corresponded to 0.775 and 0.79 for the single-scale data and 0.43 and 0.42 for the multi-scale data for the sliding window and coarse-to-fine approaches (See Table 2). Mutch and Lowe in Table 2 achieved very high score with the improved version of HMAX with sliding windows. As previously mentioned, our coarse-to-fine approach can be expanded to these modified HMAX models.

Brosch and Neumann modified the Mutch and Lowe's HMAX and combined it with HOG features, where they employed the coarse-to-fine approach to also reduce the processing area on the fine scale. The results from their method that only used HMAX features have been listed in Table 2 for comparison. As their main interest was not to reduce computational cost, the actual processing time was not mentioned; however, they claimed they decreased the processing time to about 15% on the fine scale (not for the whole process). In contrast, our approach decreased the time for the whole detection process to about 0.33% (calculated from the “UIUC car (multi-scale)” in Table 3) because our approach not only eliminated the processing region on the fine scale but also redundancies over positions and scales. That is, our approach processed whole images from the S1 to the C2 layer to get rid of overlapping regions in sliding windows, as described in Section 3.1(2). Furthermore, some processes on coarse and fine scales were shared in multi-scale localization tasks, as described in Section 3.1(4).

Table 3 lists the average processing times for both data (single- and multi-scale). The numbers in paren-
theses “()” in Table 3 are the standard deviations. These results indicate that our method reduced the computational cost of sliding windows significantly without reducing accuracy. We achieved 1.5% better recall in $2.36 \times 10^{-3}$ time in the single-scale test and 1.0% less recall in $1.31 \times 10^{-5}$ time in the multi-scale test. Note that as feature vectors were only computed from the selected patches by Adaboost, the normal cost of feature calculations increased. We ran the full computation of HMAX feature vector in the single scale tests and it took 167.38 sec per image. That means our approach reduced the processing time to about a total of less than $10^{-3}$ times.

(4) FDDDB

Our coarse-to-fine and the sliding window methods were also tested with FDDDB\textsuperscript{36} to compare them with other categories. The FDDDB has image sets for face detection tasks, which have 2845 images with a total of 5171 faces. There are some examples of the faces detected with our approach in Fig.10. The relative size of the detected window to the face in the top left image in Fig.10 seems larger than that in the other results. That is because the size ratio of the object to search was large, which was around 1.41; hence, the optimal size was omitted. Nevertheless, the max pooling operations in the C1 and the C2 layer succeeded in covering the gaps.

We have followed the “EXP-1” experimental protocol in this experiment, which was a 10-fold cross-validation of these face images. The negative training samples were randomly created from the Caltech-256 dataset\textsuperscript{37}. The results obtained from localization are plotted as receiver operating curves (ROC) in Fig.11. The discrete score is the probability of a face existing where the overlap ratio, $a_o$, in Eq.(9) is over 0.5. The continuous score is just defined as $a_o$. The details of these evaluation protocols are given in Jain and Learned-Miller\textsuperscript{36}. The results from Viola & Jones\textsuperscript{8} face detector implemented in OpenCV\textsuperscript{38} are also plotted in Fig.11 as a benchmark, even though the parameters, such as detection scale-factors, are not the same as those in our implementation.

The difference in the true-positive rate between the sliding window and coarse-to-fine approaches is only about 0.05 on the discrete score and about 0.03 on the continuous score; however, the processing time for our approach is 255 times faster than that for the sliding window approach as can be seen in Table 3.

These results indicate that our approach could reduce the processing speed of HMAX enormously with a small reduction in accuracy not only in car detection but also in face detection tasks.

4.2 Evaluation of modifications

The main objective of this section is to explain our analysis of what impact our modifications had on performance in our approach. We investigated three points of view: the first was the number of iterations in training, the second was the “coarse-to-fine” approach, and the third was division of the C2 area, which is explained by Eq.(7) and Fig.5.

(1) Iterations for training

The number of iterations of real Adaboost affected the number of feature patches because each iteration selected one patch from $N_p$ patches in the feature dictionary. The number of feature patches had an influence on accuracy and processing speed.

We plotted the relation between the number of it-
Fig. 10  Examples of face detection.

Fig. 11  ROC curves for FDDB.

(a) ROC curves based on discrete scores
(b) ROC curves based on continuous scores

The number of iterations. The recalls were satiated at around 50 - 100 iterations. We selected “100” iterations by considering the trade-off between them.

The process time at 20 iterations is inexplicably higher than that at 50 iterations in the graph. The reason for this is that the first scale band of the object model trained for 20 iterations has less ability to discriminate; thereby, this model rejected a smaller area at an early stage and caused a slight increase in time.

(2) Fine vs coarse-to-fine

The fine object model that only had a single-scale band was trained to compare it with the coarse-to-fine object model which had two scale bands to study the effects of the coarse-to-fine approach. In other words, the fine model is the case of $l = 1$ in Algorithm 1, and the coarse-to-fine model is the case of $l = 2$ in this experiment. The fine model was trained with 200 iterations of Adaboost to fit the total number of iterations to the coarse-to-fine model. The recall and process times are summarized in Table 4 and 5.

The coarse-to-fine model did not seem to improve its processing speed while its recall was inferior to that of the fine model in the single scale data test (UIUC car).
There are two reasons for this behavior. First, half the number of features in the coarse-to-fine approach were trained at lower resolution because all the features of fine model were trained at higher resolution. In addition, the coarse-to-fine model eliminated the low confidence area at an earlier stage; therefore, its ability to discriminate was reduced. Second, the total number of features was not always equal to the number of iterations because real Adaboost often chose duplicated features. The fine model in this case chose more duplicated features than the coarse-to-fine model did since it had to select all features on the same scale. As the coarse-to-fine model had more than the fine model for this reason, the process time could not be shorter.

The coarse-to-fine model, on the other hand, succeeded in reducing the process time by about 10% while maintaining accuracy in the multi-scale test (UIUC car). The localization tasks of multi-scale dataset were better than those in the single-scale dataset because a small reduction in the process time would have accumulated. Similarly, the process time for the coarse-to-fine approach in the FDDB test that had multi-scale faces was 59% of that for the fine approach with only a 2% decline in recall.

Comparing the fine object model with the sliding window model (see Tables 3 and 4) reveals that removing overlapping regions between sliding windows had a great effect on computing cost.

(3) Dividing max-pooling area in C2 layer

It had been reported that information on feature patch positions improved the accuracy of classification tasks\(^{(18-20)}\), as stated in Section 3.1. The previous work achieved this simply by dividing the max-pooling area in the C2 layer. We introduced this “dividing” approach and implemented it with our method of localization by using a dilation filter (see Eq.(7)).

We wanted to confirm whether the dividing approach worked for localization tasks. The second scale band of the object model was horizontally divided into two regions in this experiment. The overlap ratio, \(\Delta N_{c2}\), between two regions was set to zero. The divided model and the non-divided models were compared.

The recall-precision curves for single- and multi-scale tests are in Fig.13. Performance has obviously been improved in the multi-scale test but not in that in the single-scale test. The position information of features indicated that it improved performance in more difficult tasks.

5. Discussion

The current implementation of our method did not take into account state-of-the-art performance. However, we focused on redundancies in sliding window approach of the “basic” HMAX model so that it could be implemented in other “improved” HMAX models\(^{(18-20)}\)\(^{(22-23)}\) that have been reported to achieve better performance and faster speed than the basic-model. As these improved models mainly focused on categorization tasks, our work might not conflict with these methods.

The current implementation of our method was re-

| Table 4 | Average process times per image of fine and coarse-to-fine models (sec). |
|---------|-----------------|-----------------|------------------|
|         | Single-scale    | Multi-scale     | FDDB             |
| Fine    | 0.19 (0.20)     | 0.95 (0.72)     | 5.13 (0.82)      |
| Coarse-to-Fine | 0.19 (0.18) | 0.85 (0.61)     | 3.05 (0.50)      |

| Table 5 | Recalls for fine and coarse-to-fine models. |
|---------|-----------------|-----------------|-----------------|
|         | Single-scale    | Multi-scale     | FDDB             |
| Fine    | 0.84            | 0.42            | 0.49             |
| Coarse-to-Fine | 0.80    | 0.42            | 0.47             |

Fig. 13 Recall-precision curves for divided and non-divided C2 areas.
stricted to the localization of single categories; however, we plan to expand this model to multi-category localization tasks. Since objects can be explained as groups of feature patches in our model, common features in categories can be shared to be searched. Because of this, the process time to detect multi-classes is not expected to linearly increase. We also expect the method proposed by Dean et al. to be applied into the multi-category localization of HMAX, which can select highly similar patches to image appearances rapidly with locally sensitive hashing\textsuperscript{(39)}. In addition, the implementation of multi-category detection can also resolve other limitations. The aspect ratio of training samples must be the same in the current method because it determines the filter size of C2 dilation. The architecture for multi-class detection can treat the same objects with different aspect ratios as different objects. Of course, as the feature patches of these objects are the same, computational costs may increase slightly.

Another idea to attain improvements is to avoid image pyramids for faster processing. For example, Haar-like features are known to calculate multi-scale features without creating image pyramids. Dollar et al.\textsuperscript{(40)} and Benenson and Mathias\textsuperscript{(31)} also reported that other scales of HOG features could also be approximated without image pyramids. It is worth considering whether the same kind of idea can be applied to the Gabor filter in the S1 layer or to patch similarities calculations in the S2 layer.

6. Conclusion

We proposed a method that improves the process time of HMAX features for object localization tasks. The previous localization approaches using HMAX simply made use of a sliding windows. We focused on improving three redundancies that were specific to the HMAX model in the sliding window approach; the first was overlapping regions in sliding windows, the second was duplicated filters in multi-scale localization tasks, and the third was omission of an object size by max-pooling in the C2 layer. The main idea in our approach to eliminate the redundancies was that the image region of a target object should have high degrees of similarity to the shapes that the object has. This idea was implemented by three main changes from sliding window approach. First, the whole image region is processed at one time to eliminate overlapping areas in sliding windows. Second, processing of small shapes in larger objects and large shapes in smaller objects are shared to eliminate overlapping scales. Finally, an image region that has high degree of similarity to parts of the target object is searched from coarse-to-fine area.

The experiments proved that these ideas reduced the processing time enormously with negligible reduction in precision. In addition, we evaluated the effect of various parameters and functions such as the number of training iterations, coarse-to-fine approach, and division of the C2 layer, and hence confirmed their optimal settings or benefits.

References
8) P.Viola and M.Jones: “Rapid Object Detection using a Boosted Cascade of Simple Features”, IEEE Conference on Computer Vision and Pattern Recognition, pp.511-518 (Jun 2001)
18) J.Mutch and D.G.Lowe: “Object class recognition and localiza-