VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

MULTI ORIENTATION PERFORMANCE OF FEATURE EXTRACTION FOR HUMAN HEAD RECOGNITION

Panca Mudjirahardjo, Rahmadwati, Nanang Sulistiyanto and R. Arief Setyawan
Department of Electrical Engineering, Faculty of Engineering, Universitas Brawijaya, Jl. MT. Haryono, Malang, Indonesia
E-Mail: panca,rahma}@ub.ac.id

ABSTRACT

The main component for head recognition is a feature extraction. One of them as our novel method is histogram of transition. In this paper we evaluate multi orientation performance of this feature for human head detection. The input images are head and shoulder image with angle of 315º, 330º, 345º, 15º, 30º and 45º. We use SVM classifier to recognize the input image as a head or non head, which is trained by using normal orientation (0º) images. For comparison, we compare the recognition rate with the existing method of feature extraction, i.e. Histogram of Oriented Gradient (HOG) and Linear Binary Pattern (LBP). The experimental results show our feature more robust than the existing feature.

Keywords: histogram of transition, head recognition, multi orientation performance.

INTRODUCTION

Head detection and recognition have been an important research in the last few years. Many applications use this research, such as robotics, automated room monitoring, people counting, person tracking, etc. Many new methods are introduced in this field, to improve the computation time and the recognition rate. One of them is the method based on feature extraction.

Feature extraction plays an important role in head recognition. It transforms an original image into a specific vector to be fed into a classifier. An original image cannot be further processed directly. Raw information in an original image does not represent a specific pattern and a machine cannot understand that information.

In an image, there are foreground and background patterns. In a simple image, foreground and background pattern can be separated clearly. In a complex image, however, foreground and background pattern cannot be separated clearly. There are many texture patterns both on foreground and background. Sometimes, both foreground and background contain similar texture and color on them. This is a difficult task in a head detection and recognition system. The system has to recognize a foreground pattern as a head or a non-head. Correct choice of a foreground extraction method will increase the recognition rate.

A feature is assumed to be able to distinguish a foreground and a background pattern. All of features distinguish a foreground pattern over the background from the edge pattern of the foreground, since a foreground has a specific edge pattern over the background.

Currently the most commonly used feature extraction methods are Histogram of Oriented Gradients (HOG)[1][2] and Linear Binary Pattern (LBP) [3]. The new feature extraction is a histogram of transition as our novel method [5][6]. This feature is relied on a background extraction. The simple method to extract a foreground is by using a difference function. Where we label some pixels as foreground pixels, then we calculate all pixel intensity with respect to the foreground pixels. If the difference result is less than or equal to the threshold, then the pixel is consider as foreground, otherwise is as background.

The overview of this experiment is shown in Figure-1. The structure of this paper is as follows. Section 2 explains the feature extraction methods. Experimental results are shown in section 3. Finally the paper is concluded in section 4.

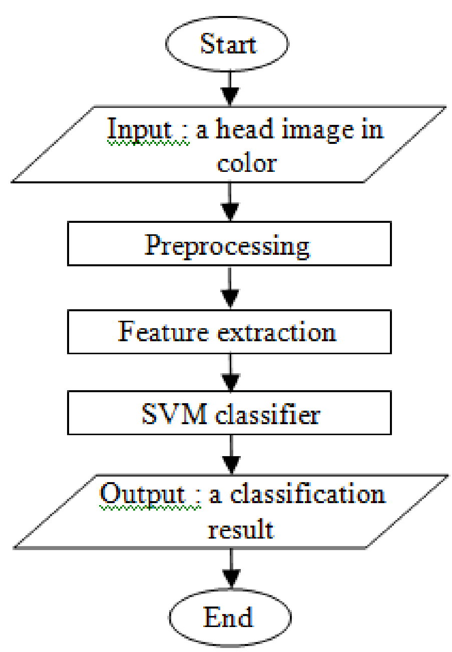

Figure-1. Overview of the experiment.

ARPN Journal of Engineering and Applied Sciences

FEATURE EXTRACTION

Histogram of oriented gradients [1, 2, 11]

This feature is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ("*cells*"), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, *etc.*, it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("*blocks*") and using the results to normalize all of the cells in the block. This will refer to the normalized descriptor blocks as *Histogram of Oriented Gradient (HOG)* descriptors. The overview of HOG feature extraction is depicted in Figure-2.

We use a simple 1-D [-1, 0, 1] mask to compute the image gradients and no pre-smoothing of the image is applied, both in horizontal and vertical directions. The gradient in horizontal direction, I_x, and the gradient in vertical direction, I_y, are defined as follows,

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} I(x-1,y) & I(x,y) & I(x+1,y) \\ I(x,y-1) & I(x,y) & I(x,y+1) \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad (1)$$

$$|I| = \sqrt{I_x^2 + I_y^2} \qquad (2)$$

$$\theta = \tan^{-1}\frac{I_y}{I_x} \qquad (3)$$

where I (x,y) is the intensity of a grayscale image at location x-th column and y-th row. $|I|$ is a gradient module magnitude and θ is a gradient module orientation.

At the next step, a clustering process based on the gradient module orientation is done. In practice, the gradient image is split into 8 image bins, each one representing an orientation with a certain range within 0^o to 360^o. For every pixel, the corresponding module of the gradient is stored in the appropriate orientation image bin. The orientation image bin is shown in Figure-3.

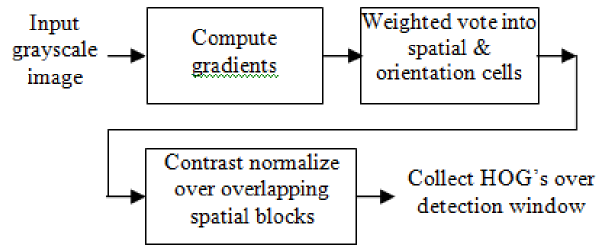

Figure-2. The overview of HOG feature extraction.

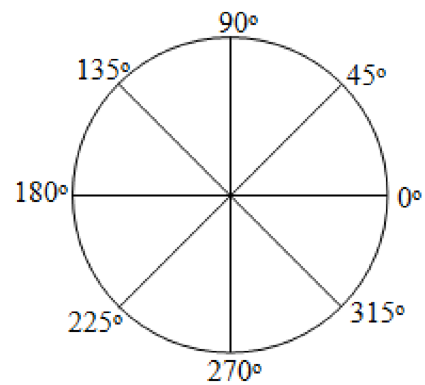

Figure-3. The orientation image bin.

At this point, we calculate the histogram of oriented gradient in the square-cells. The cell size is 5×5 pixels. Each cell is calculated directly from the gradient image and they are not overlapping.

Then we perform contrast normalization over overlapping spatial blocks. Block size is 15×15 pixels or 3×3cells. The overlapped pixels are 5 pixels. We use normalization as follows;

$$L2-norm: \boldsymbol{v} = \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + \varepsilon^2}} \qquad (4)$$

Here ε is a small constant larger than zero.

Local binary pattern [3, 11]

The local binary pattern (LBP) is a non-parametric operator which describes the local spatial structure of an image. Ojala *et al*. [4] first introduced this operator and showed its high discriminative power for texture classification. At a given pixel position (x,y), LBP is defined as an ordered set of binary comparisons of pixel intensities between the center pixel and its eight surrounding pixels, as shown in Figure-4. The decimal form of the resulting 8-bit word (LBP code) can be expressed as follows;

VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

$$LBP(x,y)=\sum_{n=0}^{7}s(i_n-i_c)2^n \quad (5)$$

where i_c corresponds to the grey value of the center pixel (x,y), i_n to the grey values of the 8 surrounding pixels, and function $s(x)$ is defined as

$$s(x)=\begin{cases}1 & \text{if} \quad x\geq 0\\ 0 & \text{if} \quad x<0\end{cases} \quad (6)$$

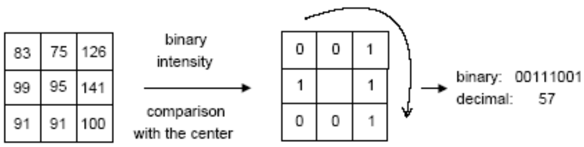

Figure-4. The LBP operator.

By definition, the LBP operator is unaffected by any monotonic gray-scale transformation which preserves the pixel intensity order in a local neighborhood. Note that each bit of the LBP code has the same significance level and that two successive bit values may have a totally different meaning. Actually, The LBP code may be interpreted as a kernel structure index.

An image is usually divided into small regions. For each region, a cumulative histogram of LBP codes, computed at each pixel location within the region, is used as a feature vector.

Histogram of transition [5, 6, 11]

Another feature is the novel method, a histogram of transition [5, 6]. As the first step to create a histogram of transition, we calculate a transition feature. A transition feature is to compute the location and the number of transitions from background to foreground along horizontal and vertical lines. So, this transition feature relies on foreground extraction. Figure-5 shows the overview of creating a histogram of transition.

In this paper, we consider to use a simple foreground extraction. A simple algorithm to extract foreground is that we determine some reference pixel coordinates as foreground [5, 6]. Then we compare another pixel's intensity (I_x) to the reference pixel's intensity (I_R). Since we extract foreground in an RGB image, we have pixel's intensity in red, green and blue. We determine a pixel at coordinate (x,y) as foreground or background by equation (7),

$$I(x,y)=\begin{cases}\text{foreground} & \delta(I_x,I_R)<th\\ \text{background} & \text{otherwise}\end{cases} \quad (7)$$

where $\delta(.)$ is a distance function. In this paper, we use Euclidean distance.

After we get the foregrounds, then we extract the feature of these foregrounds. Our feature refers to [7] [8]. Transition feature has been used successfully in handwritten characters recognition, but it hasn't been used in head detection yet. Due to a simple calculation to create a feature vector, we apply the idea to head recognition. We do some modifications on it to be able to be used for head recognition.

The idea is to compute the location and the number of transitions from background to foreground along horizontal and vertical lines. This transition calculation is performed from right to left, left to right, top to bottom, and bottom to top. Since a constant dimensional feature is required as input to the SVM classifier, an encoding scheme is developed.

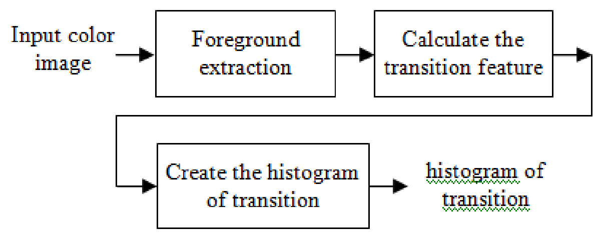

Figure-5. The overview of histogram of transition feature extraction.

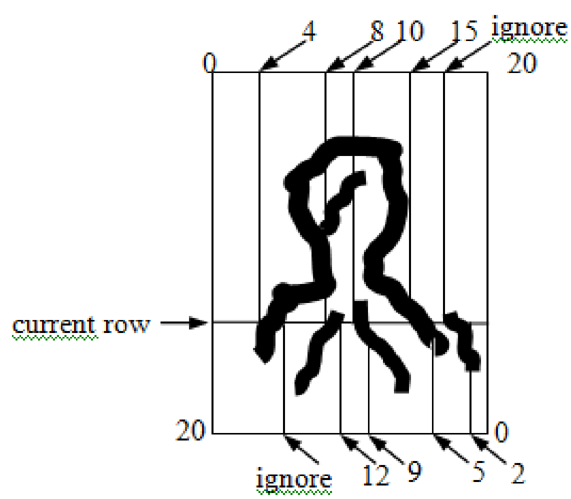

Figure-6. The first stage of transition feature extraction shown for transitions from the left and from the right on one row of the image, with M = 4.

In the first stage of feature extraction, the transition in each direction is calculated. Each transition is represented as a location of changing from background to foreground in the direction under consideration. This transition is computed in the increasing order, differed from [8] in decreasing order. For example, when calculating the location of transitions from left-to-right, a transition close to the left edge would have a low value and a transition far from the left edge would have a high value as illustrated in Figure-6.

For each line, we determine the maximum number of transitions, M. If there are more than M transitions in a line, only the first M are counted and the rest are ignored. M is set to 4 in the experiment. If there are less than M transitions on a line, then the "nonexistent" transitions are assigned as a value of 0.

More precisely, by a line we mean a row or a column of the head image. Let h be the height of the image and w be the width of the image. We assign exactly M values to each line, say $t_1,t_2,\ldots,t_M$. We assume that there are n transitions on a line located at (x_i,y_i) for $i = 1,2, \ldots n$. The algorithm for calculating the transition feature can be represented as follows: It doesn't require normalization as in [8]:

```
for i = 1 to min (n,M)
    if the line is a row then
        t_i = x_i;
    else
        t_i = y_i;
    end if;
end for;
if n < M then
    For i = n+1 to M
    t_i = 0;
    end for;
end if;
```

The transitions are resampled to a 4-point sequence for each direction and assembled into a feature vector. The four transitions for each row (column) are represented as two-dimensional (2-D) array, $t = [t_{ij}]$ for $i = 1,\ldots,h(w)$ and $j = 1,\ldots,4$.

The second stage is generating a histogram of transition. It is different from [8] where they calculated local averaging on the columns of t. Histogram of transition shows how often the location of the transition occurs at each transition. An example of generating a histogram of transition for transition left-to-right is shown in Figure-7.

This histogram of transition creates a feature vector to be fed into the input of a SVM classifier [8].

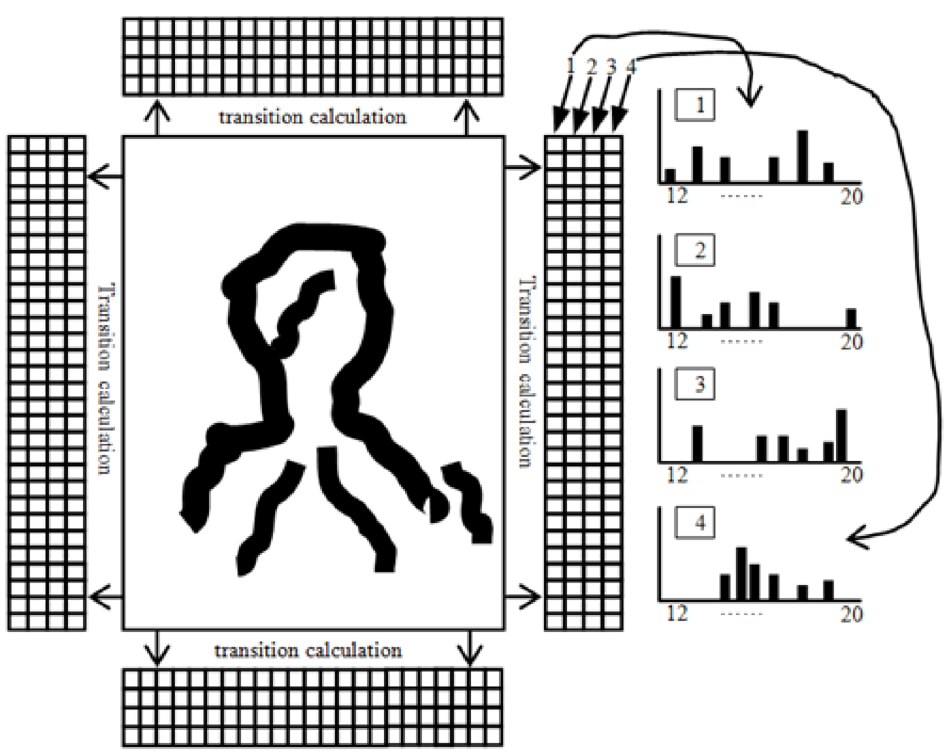

Figure-7. The second stage of transition feature calculation consisting of generating a histogram of transition.

The HOG feature contains gradient information of a pixel among its neighbor. Thus they give a high magnitude at the edge. On the other hand, the LBP feature gives a binary pattern with a pixel among its neighbor. The histogram of transition feature looks like the HOG feature: It gives the edge position from right, left, top and bottom side. In contrast to the HOG feature, the calculation of the histogram of transition feature is simpler.

EXPERIMENTAL RESULTS

The experimental environment is as follows: Operating system is Windows 7 professional; the processor is Intel® core™ i7 CPU 870 @2.93GHz and the used software is Microsoft Visual Studio 2010.

For robust detection, we use backgrounds and negative samples at outdoor scenery. We use INRIA data [1] [10] for training and testing images. For training, we use image size 20×30 pixels, positive sample of 2,000 images, negative sample of 4, 500 images. For testing, we use image size of 20×30 pixels. Positive sample of 100 images, negative sample of 300 images are employed. The image orientation are 315º, 330º, 345º, 15º, 30º and 45º. Figure-8 shows some positive and negative samples of normal orientation, and Figure-9 shows the input image with multi orientation.

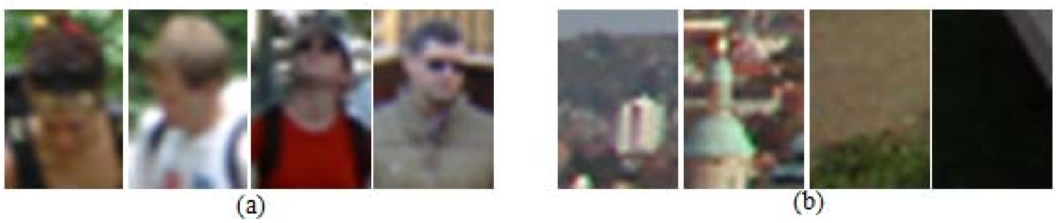

Figure-8. Samples for SVM training with normal orientation: (a) positive samples, (b) negative samples.

Figure-9. Input image with multi orientation, from left to the right, 315º, 330 º, 345 º, 15 º, 30 º and 45 º respectively.

In this research, we use 9 bins to cluster the gradient image [11]. One is for value I_x and I_y equals zero, the other 8 bins for representing the orientation from 0º to 360º. For image size 20×30, cell size 5×5, and block size 15×15 with overlapped pixels of 5 pixels, the number of feature dimensions equals

the_number_of_bins × *the_number_of_cells_in_a_block* × *the_number_of_blocks*, which amounts to 648 dimensions.
In using the LBP operator, we divide an image into four non-overlapping regions [11]. The number of feature dimensions equals

the_number_of_region × *max_number_of_decimal _value*, which amounts to 1020 dimensions.

For the image preprocessing to extract transition feature, we extract the foreground using a difference function Equation (9). First, we determine 5 and 32 reference pixel coordinates as foreground. These coordinates are fixed for all the training and the test data. The coordinates should represent position of head and shoulder. For 5 reference pixel coordinates are (10,8), (10, 15), (10, 22), (5, 22) and (15, 22) as in [5, 6]. For 32 reference pixel coordinates are (10, *i*) and (*j*, 26), where *i* = 6,7,…,26 and *j* = 15,16,…,25. Then, we check all pixels' intensity to the five reference pixel's intensity with Euclidean distance, by Equation (9). If a pixel's intensity distance to the one or more of five reference pixel's intensity is less than a threshold, the pixel should be a foreground pixel, otherwise a background pixel [11].

In this research, the maximum number of transition, M, is 4. The number of feature dimension, 2×((*M*×*width_of_image*)+ (*M*×*height_of_image*)), are 400 dimensions.

The recognition rate of positive and negative sample for Histogram of Transition. From that chart, the HSV color image yields the recognition rate better than the RGB one, both for positive and negative image. So, for the

ARPN Journal of Engineering and Applied Sciences

next experiment, we use the HSV one for the input image and 32 reference pixel coordinates.

The result of head recognition for normal orientation is summarized in Table-1, and for multi orientation is summarized in Table-2.

Table-1. The result of head recognition for normal orientation.

Feature	**The number of array**	**Recognition rate** (%)		**Execution time** (ms)
		Positive	**Negative**	
HOG	648	84	98.3	0.353
LBP	1020	95	80.0	0.261
Histogram of transition (32)	**400**	**92**	**99.7**	**0.087**

Table-2. The result of head recognition with multi orientation.

Orientasi	**FHOG**			**LBP**			**Histogram of transition (5)**		
	Positive	**Negative**	**Total**	**Positive**	**Negative**	**Total**	**Positive**	**Negative**	**Total**
315 ^{o}	45	92.33	80.5	43	98	84.25	77	94.33	**90**
330 ^{o}	56	98.33	87.75	59	94.67	85.75	79	95.67	**91.5**
345 ^{o}	57	99.33	88.75	69	94.67	88.25	88	98	**95.5**
15 ^{o}	63	99.67	90.5	88	87	87.25	91	99.33	**97.25**
30 ^{o}	49	99.67	87	48	95.67	83.75	76	95.67	90.75
45^o	44	98	84.5	56	94	84.5	80	95.67	**91.75**

Orientasi	**Histogram of transition** (32)		
	Positive	**Negative**	**Total**
315 ^{o}	42	100	85.5
330 ^{o}	54	100	88.5
345 ^{o}	84	99	95.25
15 ^{o}	86	99.33	96
30 ^{o}	76	98.67	**93**
45^o	55	98.67	87.75

CONCLUSIONS

In this paper, we evaluate the performance of multi orientation of input image, then we extract the feature and perform a comparison of the existing image features extraction methods using a static image. The existing features are HOG and LBP, and the proposed feature is a histogram of transition.

In design, the proposed feature is robust for multi orientation image and has the acceptable recognition rate compared to the existing features.

The proposed feature has the number of array less than the existing features, and the computation of feature transition is simpler than the existing features. These conditions give the computation of the proposed feature faster than the computation of existing features. This performance shows that the proposed feature can be used for real time application.

As future work, we are going to conduct experiments to improve foreground extraction.

REFERENCES

[1] N. Dalal, B. Triggs. 2005. Histograms of oriented gradients for human detection. Proceeding of Computer Vision and Pattern Recognition. 1: 886-893.

ARPN Journal of Engineering and Applied Sciences

[2] M. Perdersoli, J. Gonzalez, B. Chakraborty, J. Villanueva. 2007. Boosting histograms of oriented gradients for human detection. In: Proc. 2nd Computer Vision: Advances in Research and Development (CVCRD). pp. 1-6.

[3] G. Heusch, Y. Rodriguez, S. Marcel. 2006. Local binary patterns as an image preprocessing for face authentication. Proceeding of Automatic face and Gesture Recognition (FGR 2006). pp. 9-14.

[4] T. Ojala, M. Pietik¨ainen, D. Harwood. 1996. A comparative study of texture measures with classification based on featured distributions. Pattern Recognition. 29(1): 51-59.

[5] P. Mudjirahardjo, J.K. Tan, H. Kim, S. Ishikawa. 2014. Head detection and tracking for an intelligent room. Proc. SICE Annual Conference. pp. 353-358.

[6] P. Mudjirahardjo, J.K. Tan, S. Ishikawa. 2015. A study on human motion detection - Toward abnormal motion identification. Ph.D. Thesis, Kyushu Institute of Technology, Japan, pp. 35-55.

[7] P. Mudjirahardjo. 2001. Penerapan jaringan perambatan balik untuk pengenalan kode pos tulisan tangan. (The implementation of back-propagation network for recognition of handwritten post code), Master Thesis, Universitas Gadjah Mada, pp. 1-100.

[8] P.D. Gader, M. Mohamed, J.H. Chiang. 1997. Handwritten word recognition with character and inter-character neural networks. IEEE Trans. On Systems, Man, and Cybernetics - Part B: Cybernetics. 27(1): 158-164.

[9] C.W. Hsu, C.C. Chang, C.J. Lin. 2010. A practical guide to support vector classification. Proceeding of IEEE Intelligent Vehicles Symposium. 1: 1-6.

[10] http://pascal.inrialpes.fr/data/human/

[11] P. Mudjirahardjo, J.K. Tan, H. Kim, S. Ishikawa. 2015. Comparison of feature extraction methods for head recognition. To be appeared in Proc. 17th International Electronics Symposium (IES).