Image objects detection based on boosting neural network

10th Symposium on Neural Network Applications in Electrical Engineering, NEUREL-2010
Faculty of Electrical Engineering, University of Belgrade, Serbia, September 23-25, 2010

Image Objects Detection Based on Boosting Neural Network

Ningqing Liang, Hans Hegt, *Senior Member, IEEE,* and Valeri M. Mladenov, *Senior Member, IEEE*

***Abstract*—This paper discusses the problem of object area detection of video frames. The goal is to design a pixel accurate detector for grass, which could be used for object adaptive video enhancement. A boosting neural network is used for creating such a detector. The resulted detector uses both textural features and color features of the frames.**

***Index Terms*—boosting neural network, feature, grass**

I. Introduction

THE demands on high quality TV programs increases rapidly. Video enhancement technology is essential to enable low quality videos gaining good watching experience when played on High-End Displays. Object adaptive video enhancement is the new technology under development. In order to achieve the best quality, adaptive enhancement applies most suitable enhancement operations to specific objects in the frames according to the characteristics of the target object. For example, the best noise reduction techniques for a smooth sky area and a wavy sea area in the same frame are different. Before the suitable enhancement can be executed, a decent object segmentation of the frame is a must.

This project aims at creating an object detection method for making excellent object segmentation. The example object used in this research is football-field grass since there's substantial demand for playing high-definition football matches on TVs. The problem is then turned into a classification problem of automatically labeling the grass part and non-grass part of the input video frames. Several approaches have been tried for this problem, either textural model based or textural and color probability model based [1] [2] [3]. Nevertheless, all approaches using complex models which result in high computation demand. A boosting neural network might be a solution to this and it is proved that a boosting method works well with neural networks [4]. Meanwhile, image features based approaches are with lower computation complexity. As a result, an image feature based method using a boosting neural network is used in this paper. It only requires heavy computation during training and runs quickly when applied to real frames.

N. Liang and J. A. Hegt are with the Department of Electrical Engineering, Eindhoven University of Technology, Den Dolech, 5600MB, the Netherlands (e-mail: n.liang@student.tue.nl). , (e-mail: j.a.hegt@tue.nl).

V. M. Mladenov is with both the Department Theory of Electrical Engineering, Technical University of Sofia, 8 Kl. Ohridski Str., Sofia-1000, Bulgaria and the Department of Electrical Engineering, Eindhoven University of Technology, Den Dolech, 5600MB, the Netherlands (e-mail: valerim@tu-sofia.bg).

II. Preparing the Data

A. Data Obtaining

All video frames used in this project are obtained from high definition videos of recorded football matches. The frames are images defined in 8-bit YUV color space with the dimension of 1024*768. The YUV space is widely used in TV signal broadcasting, in which the Y channel contains the textural information of the frame and the UV channels contain the color information. For each pixel, there are component values of Y, U and V, which make the whole frame a 3D matrix of size 1024*768*3 and 8 bits means every entry in the matrix is within 0~2^8-1, which is 0~255.

B. Image Processing

In order to prepare the data for training and evaluation of a detector, we have to first label the frames. That is manually making a classification result of every frame. Here, we label the pixels of frames into three categories, pixels of the object, not of the object and skipped labeling pixels. This process has been done with a labeling tool from Philips. What we get are labeled frames with different colors indicating the category numbers of the pixels: green for pixels of the object, red for pixels that are not of the objects and orange for skipped labeling pixels. Usually, the skipped pixels lie on the border of object and non-object areas, shown in Fig. 1.

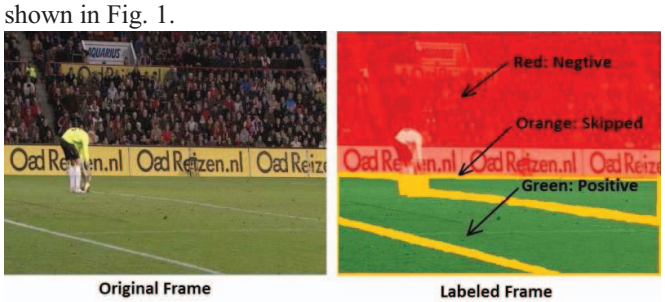


Fig.1. Illustration of the labeling results

During processing, the category information is translated back to labels from the color of the pixels. Thus after preprocessing, every frame has a corresponding labeling matrix telling the category number of all pixels of the frame. In this research project, we started from the frames with their labeled copies.

978-1-4244-8820-9/10/$26.00 ©2010 IEEE

III. Feature Extraction

A. Textural Features

Traditional feature based image classification methods mainly focus on textural features because the textural characteristics contain most of the detail information of the original image. We designed three groups of textural features which are widely used for image classification, the dynamical range, standard deviation and variance, all of which are valued on a block of pixels. A sample block forming is depicted in Fig. 2.

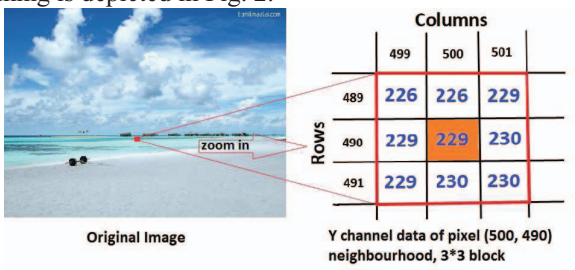


Fig.2. 3*3 block pixel data on Y channel for feature generating centered on (500, 490).

The 3*3 block is the neighborhood of the center pixel. The entries are the Y channel luminance values. Let block33 stand for the set of the 9 values, the feature dynamical range is calculated as:

Feature DR3*3 = max(block33) - min(block33) = 230 - 226 = 4

Max(block33) returns the max value in block33 and min(block33) returns the min. The result is the feature value of the highlighted red center pixel of the block.

TABLE I
LIST OF ALL THE USED TEXTURAL FEATURES

Block\feature	Standard Deviation	Variance	Dynamical Range
49*1		✓	✓
31*1		✓	✓
1*31		✓	✓
17*1		✓	✓
1*17		✓	✓
13*13	✓	✓	✓
11*11	✓	✓	✓
9*9	✓	✓	✓
7*7	✓	✓	✓
5*5	✓	✓	✓
3*3	✓	✓	✓

✓ -the feature is used.

With various defined sizes of blocks, three groups of textural features are designed.

B. UV Color Plane Features

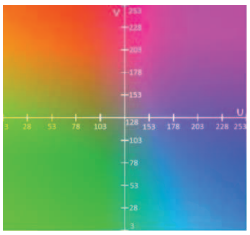


Fig.3. 8-bit UV color plane, composed by U and V axis, valued in [0, 255]

Fig. 3 shows a clear division of color in the UV plane which is a subspace of YUV color space. Pink, yellow and red, green, and blue occupy the four quadrants of the plane. This good division can be turned into good features. However, every pixel of a frame, which corresponds to a unique point in the UV plane, has a 2-D value while every feature could only have a single value. A projection from a 2-D plane to a 1-D line has been used to solve this problem. The 2-D plane is called the projected area and the 1-D line is called the target line. The target line is defined always going through the origin to ease calculation. A general definition of the projection used is shown in the left part of Fig. 4.

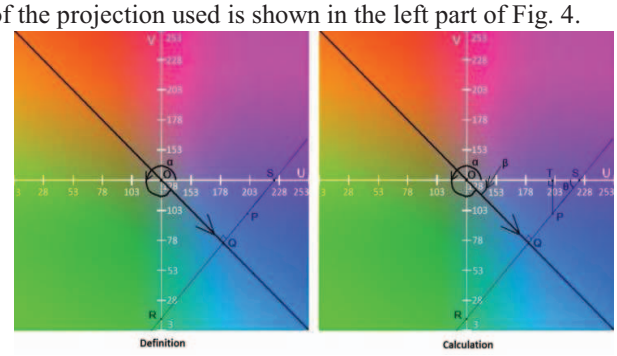


Fig.4. Illustration of UV plane projection features

In the figure, α is the angle of the target line with the right U axis, O is the origin, P is the original (U, V) point and Q is the projected point on the target line. The line PQ is called the projection line and it intersects with V axis at R and U axis at S. The length OQ is a 1-D value which could be used as a feature value. However, since obtaining the value of OS is much easier than OQ and the length of OS is uniquely corresponds to the length of OQ, so finally the value used is the length of OS. The calculation is depicted in the right part of Fig. 4. It is:

$$OS = OT + TS = U + (-V)\cot(\theta) = U + (-V)\tan(\beta) = U - (-V)\tan(\alpha) = U + V\tan(\alpha).$$

Different features can be created by varying α, but as a result of a few experiments, 60 degrees which is the optimal of a few tested values is selected to create the feature Grass_Pd.

The detection speed is also an important factor. Therefore we have created the feature Grass_Prj which is a computational reduced version of Grass_Pd by taking α=45 because then we can have |tan (α)|=1.

C. RGB Color Space Features

The RGB color space is also a three-component space like the YUV space, consisting of red, green and blue. However, all the three components contain both textural and color information. Therefore, a feature that is defined on any component of the RGB space would take textural details of the object into account. Furthermore, since the grass is green, the green component value is expected to be a good characteristic for classification. The original frames are in YUV format and there is a standard conversion from YUV to RGB format.

1) The Basic RGB Features

The values of RGB channels could well express objects that have similar color to the channel colors. Therefore, the green channel value could be used as the first RGB feature of the green grass (Here, G is the values of the green channel of the pixels):

$$\text{Feature G\_RGB\_1} = \text{G}$$

The perceived color of a pixel in an RGB image is closest to one of R, G and B which has the maximum value. This color is defined as the main color and the main color of grass is green. A second feature G_RGB_2 has been made on the "maximum" principle of the main color. It is a binary feature valued on if the green channel has the maximum value:

$$\text{Feature G\_RGB\_2} = \begin{cases} 1, & if\ \max(R,G,B) = G \\ 0, & else \end{cases}$$

When a pixel has values of the three channels close to each other, it is like in grayscale. These pixels also have value "1" on the second feature. It is not the ideal result. As the difference between the main color channel and the other two are much smaller for these pixels than for objects that have a clear main color, we define the difference as:

$$\text{Difference} = \frac{2*\text{main\_color} - \text{sum\_of\_rest\_two}}{2*\text{main\_color}}$$

It is normalized. Then we created G_RGB_3 which should have better performance than G_RGB_2.

$$\text{Feature G\_RGB\_3} = \text{G\_RGB\_2} * \text{Difference\_of\_green}$$

And:

$$\text{Difference\_of\_green} = \frac{2 * \text{G} - (\text{R} + \text{B})}{2 * G}$$

.

2) An Advanced RGB Feature

After a few experiments performed with the basic RGB features, the results imply a low detection rate on some frames that contain other objects which have similar color with the target object. Thus better RGB features are needed.

The binary feature G_RGB_2 has value "1" on all green pixels. However, G_RGB_1 has different values for green grass and a green flag. A combination of the two might be able to first find out green objects and then distinguish grass from other green objects. In addition, there are values of the main color that the target object could never have. Such as, 220 of green is too light for grass. As a result, feature Green_Mix has been developed on all the ideas:

$$\text{Feature Green\_Mix} = \text{Green\_Threshold\_Clip}\,(\text{G}) * \text{G\_RGB\_2}$$

The threshold in this feature, Green_Mix: G > 160, is used to filter out some obviously negative samples.

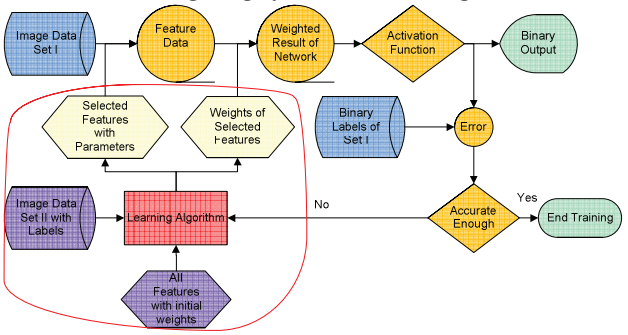
$$\text{Green\_Threshold\_Clip}\,(R) = \begin{cases} 0, & if\ G > 160 \\ G, & else \end{cases}$$

IV. THE BOOSTING NEURAL NETWORK

In chapter III, a lot of features have been created for this classification problem. However, a boosting neural network is used to train the detecting process and only select a few best features to use.

A. The Structure

The network is a double weighting boosting neural network. It is built on the basis of AdaBoost [8] and adapts the structure of an error correcting neural network. The inputs of the network are image pixel values in YUV space (Image Data Set I) with corresponding pixel labels (Binary Labels of Set I) of classification. When the inputs are loaded, the data goes into two stages of the weighting process before meeting the activation function, and when it comes out of the activation function, the output would be the binary classification result of each pixel. It will be evaluated by comparing to input prior labels to get the classification error and a decision is made on the error whether further training is needed. If the accuracy doesn't meet the requirement, the learning algorithm runs a new iteration to achieve a more accurate result by smartly tuning the weights of the double weighting system. Section B gives the details.



Fig,5. The structure of the Network: the blue components are the inputs of the network, the oranges components are the main functioning flow and the green components are the outputs of the network while the remaining components, closed by the red arcs, form the training part of the network: the purple ones are the inputs of the learning algorithm which is in red and the yellow ones are the double weighting outputs of the training process.

B. The Learning Algorithm

y As can be seen from Fig. 5, the learning process has different data input (Image Data Set II with Labels) from the main network. This technique is to avoid the problem of data over fitting. Other than the data input, there are also inputs of initial weights which are the designed features and initial weights of features (universally distributed on every pixel) for training.

The learning algorithm is also divided into two processes: the first optimizes every feature and selects the best feature based on known feature weights while the second finds out the optimal weight of the selected feature. Starting from a library of known features and their initial weights, the two processes iteratively decrease the error to an acceptable value.

Actually, the two optimized processes are called weak classifiers and strong classifiers respectively, which are widely used in machine learning. A weak classifier is weakly correlated with the true classification while a strong classifier has a strong correlation with the true classification. For example, in our case, the output of a weak classifier would probably tell if a pixel is green; and the output of a strong classifier would tell if a pixel is grass. The boosting neural network used here is one of the algorithms for composing a strong classifier from a weighted linear combination of several weak classifiers. In this paper, suppose we have totally n pixels and N features, then a weak classifier (the m-th weak classifier) could be defined on a feature as:

$$h_m(x_i, f_{im}, T_m, p_m) = \begin{cases} 1 & if\ p_m * f_{im} < p_m * T_m \\ 0 & otherwise \end{cases}$$

TABLE II
DETAILED ALGORITHM OF THE BOOSTING NEURAL NETWORK

The data inputs of the learning algorithm are Image Data Set II which are values of n pixels (written as x_i's) and their classification labels (written as y_i's), formed as (x_1, y_1), (x_2, y_2)... (x_n, y_n), where y_i = 1 for positive samples and 0 for negative samples.

The main parameters used are:

Pd, the minimum detection rate,

Pfa, the maximum false alarm rate.

Initialization:

1. Set the iteration NO. t = 0, Initial Pfa_0 = 1, Pd_0 = 1.

2. Initialize pixel weights $w_{t,i}$ = 1, for i = 1, 2...n. These are the initial values for calculating the feature weights.

Iteratively Learning:

While Pfa_t > Pfa { t ← t+1 (increasing the iteration NO.), do the following steps:

1. Normalize the pixel weights, $w_{t,i} = w_{t,i} \Big/ \sum_{i=1}^{n} w_{t,i}$.

2. Select the best weak classifier (optimizing first level weighting).

(1) Define the weighted error of the m-th weak classifier:

$$\varepsilon_{t,m} = \sum_{i=1}^{n} (w_i * | h_{t,m}(x_i, f_{im}, T_{t,m}, p_{t,m}) - y_i |)$$

(2) For the m-th feature (m = 1, 2…N), find a best parity and threshold pair of $p_{t,m0}$ *and* $T_{t,m0}$ that divides the pixels into two classes and achieves the smallest weighted classification error $\varepsilon_{t,m0}$ of the feature.

(3) In the whole feature set, find feature k(t) which has the minimum error with $p_{t,k(t)}$ *and* $T_{t,k(t)}$. Form a weak classifier $h_{t,k(t)}(x_i) = h_{t,k(t)}(x_i, f_{ik(t)}, T_{t,k(t)}, p_{t,k(t)})$ with this feature, and $(\varepsilon_t)_{\min} = \varepsilon_{t,k(t)}$.

3. Update the pixel weights and feature weights.

(1) Pixel weights: $w_{t+1,i} = w_{t,i} * \beta_t^{1-E_i}$, Where E_i=0 if sample x_i is correctly classified, otherwise E_i=1, and $\beta_t = \dfrac{(\varepsilon_t)_{\min}}{1-(\varepsilon_t)_{\min}}$.

(2) Feature weight: $\alpha_t = \log \dfrac{1}{\beta_t}$

4. Form the temporary strong classifier.

$$C_t(x) = \begin{cases} 1 & if \sum_{j=1}^{t} \alpha_j * h_{j,k(j)}(x) \geq \frac{1}{2} \sum_{j=1}^{t} \alpha_j \\ 0 & otherwise \end{cases}$$

5. Apply the strong classifier to the validation data set and update the false alarm rate Pfa_t and the detection rate Pd_t using Ct(x).

}

When the iteration is over, the network has finished training and the obtained strong classifier is the temporary strong classifier in the last iteration. It has first level weights $h_{j,k(j)}(x)$ and second level weights α_j, j = 1, 2…t.

Here, i = 1, 2…n, m = 1, 2…N, x_i is a sample pixel, f_{im} is the m-th feature value of x_i, T_m is the classification threshold and p_m is the parity which indicates the direction of the inequality. The binary classification result depends on whether f_{im} is larger or smaller than T_m and p_m can be set to -1 to reverse the direction of the inequality.

The algorithm for forming a strong classifier [5] (the outputs of the learning algorithm which consist of two level weights) from the best weak classifiers is explained in Table II.

C. The Activation Function

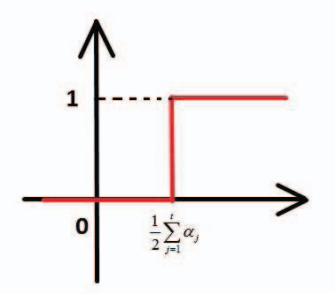


Fig.6. The nonlinear activation function: a binary limiter

The activation function of this boosting neural network is illustrated in Fig. 6. When all the weights are applied, the weighted values of network output can be obtained. These values are pushed into the activation function to get the final output of the network.

The definition of the nonlinear activation function is:

$$C_t(x) = \begin{cases} 1 & if \sum_{j=1}^{t} \alpha_j * h_{j,k(j)}(x) \geq \frac{1}{2} \sum_{j=1}^{t} \alpha_j \\ 0 & otherwise \end{cases}$$

V. EXPERIMENTS AND RESULTS

A. The Experiments

In the experiment, 10 frames which have football field grass have been used for training. Since the training process is too computation expensive with Matlab programming, only less than 1% randomly selected pixels of each frame is used. The final error for training is set as 0.01 = 1%.

B. Results

TABLE III
ACCURACY AND PERFORMANCE

ID	Percent of Pixels Used (%)	Final Error (%)	#Weak Classifiers Used	Detecting Time/frame (seconds)
1	0.01	1.04	4	9.8
2	0.02	0.91	10	7.9
3	0.05	1.02	6	8.6
4	0.1	1.03	4	7.7
Average	0.045	1.00	6	8.5

Several experiments are performed with different percentages of total pixels for training, on a HP Compaq 8510w with T9300 CPU + 4G Memory+ 7200r Disk + Windows 7 32bit + Matlab 2007b. The results are listed in Table III.

Table III shows that the trained detector uses about 6

weak classifiers and has an approximate error rate of 1%. It can detect a frame within almost 8~10 seconds. An example of the training process is depicted in Table IV.

An example of a resulting frame is shown in Fig. 7. It has

TABLE IV
SAMPLE TRAINING WITH 0.05 PERCENT OF PIXELS USED

ID Weak Classifiers	Name of Weak classifiers	Detection Error (%)
1	Green_Mix	1.0425
2	Variance 1*31	1.0425
3	G_RGB_3	1.0425
4	Variance 49*1	1.0707
5	Grass_Pd	1.0707
6	Variance 49*1	0.9298

made good segmentation of grass and non-grass area. However, there are still a few false positive areas caused by objects which are of the same color as grass. This implies that further improvements could start from solving this problem.

TABLE V
EXPLANATIONS OF COLORS AND SYMBOLS USED IN FIG. 7

		Labeled		
		Yes	No	Skipped
Classified	Yes	TP: Green	FP: Red	Skipped Yes: Pink
	No	FN: Purple	TN: Gray	Skipped No: Blue

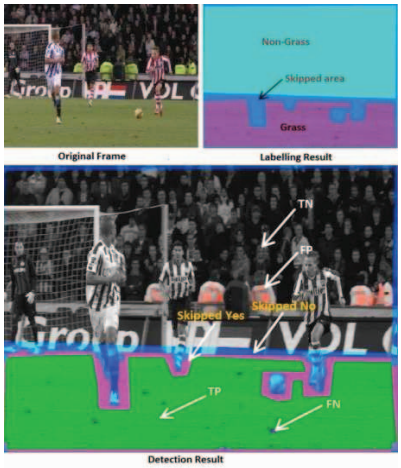


Fig.7. The detection result: colors and symbols are explained in Table V.

VI. CONCLUSION AND DISCUSSION

A. Performance and Limitation

Using a boosting neural network, a detector has been created for image object detection problem. The detector achieved a detection accuracy of about 99% and takes about 8~10 seconds for a frame of 1024*768. The performance is limited by the training condition in our situation since the Matlab scripts require high resources. A version in C-language may greatly speed up both the training and detection. However, there is not much room for the accuracy to be improved and the improving ability for this single layer boosting neural network method is limited as we inferred from the result of the experiments.

In conclusion, the detection based on a boosting neural network is accurate and fast, but is still not perfect for implementation and real-time detection.

B. Further Development

A first try of improvement could be made on translating this detector into a C-version. With a lower requirement of resources, the percentage of pixels could be increased and the final error could be decreased during the training process. This is expected to bring an improvement.

Secondly, Fig. 7 shows a problem of detecting objects that are of the same color as the target object. This implies that there may be undiscovered better features which will overcome the problem. For those small areas of mis-detected pixels, a post processing area size filter could be employed to overcome this problem, i.e. if an area of detected object is smaller than the size requirement, it will be re-classified as non-object.

Furthermore, [6] suggests a block averaging scheme which is a filter for data preprocess that may improve the performance. In [7], a multi-layer boosting algorithm is used. It speeds up the detection and increases the accuracy from single layer detection. This also seems a promising approach for improvement.

umber

REFERENCES

[1] B. Zafarifar and P. H. N. de With, "Grass Field Detection for TV Picture Quality Enhancement," in *Proc. IEEE Int. Conf Consumer Electronics,* Las Vegas, USA, 2008, pp. 329–330.

[2] B. Zafarifar and P. H. N. de With, "Blue Sky Detection for Content-based Television Picture Quality Enhancement," in *Proc. IEEE Int. Conf Consumer Electronics,* Las Vegas, USA, 2007, pp. 437–438.

[3] S. Herman and J. Janssen, "Automatic segmentation-based grass detection for real-time video," European Patent EP 1 374 170, January 2004.

[4] H. Schwenk and Y. Benggio, "Boosting Neural Networks," in *Neural Computation*, vol.12, no.8, Cambridge, MA, USA: MIT Press, 2000, pp.1869-1887.

[5] Y. Freund and R.E. Schapire, "A Short Introduction to Boosting," in *Transl. J. Journal of Japanese Society for Artificial Intelligence,* vol. 14, no. 5, Sept. 1999, pp.771-780.

[6] S. Herman and E. Bellers, "Image segmentation based on block averaging," United States Patent US 2006/0072842 A1, April 2006.

[7] P. Viola and M. Jones, "Robust Real-time Object Detection," in *2nd Int. Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling,* Vancouver, Canada, July 13, 2001.

[8] Yoav Freund, Robert E. Schapire, A Short Introduction to Boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.