A Learning Approach based on Support Vector Regression for Single-Image Super-Resolution

Jinho Park
dkszmffps@gmail.com (Number ID: 2013126531)

Introduction

Super-resolution (SR) is a process of generating a high-resolution image from one or several low-resolution versions, and it has been an active research topic in image processing and computer vision. Conventional SR methods are based on registration and alignment of multiple low-resolution images of the same scene in sub-pixel accuracy. These methods can be regarded as an inverse problem, which recovers the high-resolution image as a linear operation of its low-resolution versions.

In recent years, much attention has been drawn to single-image SR methods. Ni et al. [1] recently proposed a SR method using support vector regression (SVR) to fit low-resolution image patches in spatial or DCT domains. Due to excellent generalization of support-vector based methods, no assumption on the data such as the distribution of different image patch categories is needed.

Originally applied to signal recovery, compressed sensing [2] is adopted to SR problems by Yang et al. [3], who assumed that the patches from high resolution images can be a sparse representation with respect to an over-complete dictionary composed of signal-atoms. They have shown that, under mild conditions, the sparse representation of high-resolution images can be recovered from the low-resolution image patches. However, they used a set of selected image patches for training, and implied that their method only applies to images with similar statistical nature. Considering image regions with both high and low spatial frequencies, I do not limit to the case of natural images. My approach suggests the learning of sparse image representations for image patches from low resolution images, and I use SVR to learn/predict the associated pixel labels.

1. Single-Image Super-Resolution

Figure 1. shows the flow chart of proposed method. It consists of three steps: image patch categorization, image sparse representation, and SVR for SR image.

![Flow Chart](image.png)

Figure 1. The flow chart of proposed method.

1.1 Image Patch Categorization

In the training stage of proposed method, collect high and low-resolution image pairs.

For a low-resolution image, I first use bicubic interpolation to synthesize its high-resolution version. I extract all 5 x 5 patches from this synthesized image. To determine whether each patch corresponds to regions with high or low spatial frequencies, I detect edge on the original low-resolution image to locate pixels on the boundaries or...
corners. Next, if the center of an extracted patch from this synthesized image is a part of image edges or corners, that image patch will be categorized as the set of regions with high spatial frequencies; otherwise, it belongs to the set of those with low spatial frequencies.

In my work, I use sobel filter to detect edge the image for determination for the above image patch sets (see Figure 2.)

![Figure 2. Image patch categorization, Left: the original image, Middle: The edge detection result, Right: High-resolution image.](image)

1.2 Sparse Representation of Image Patches

Instead of working directly with the image patches sampled from low resolution images, I learn compact representations $D_h$ and $D_l$ for patches with high and low spatial frequencies, respectively.

In the training stage, I apply the sparse coding tool developed by [4] to learn the dictionaries $D_h$ and $D_l$. I determine the associated sparse coefficient vectors $a_h$ and $a_l$, which minimize the reconstruction error with a small number of non-zero coefficients. Considering high spatial-frequency patches for example, the sparse coding problem can be formulated as

$$
\min \|a_h\| \text{ s.t. } \|D_h a_h - y_h\|_2^2 \leq \epsilon,
$$

where $y_h$ is the training image patch, $D_h$ is an over-complete dictionary to be determined, and $a_h$ is the sparse coefficient vector. A small and positive $\epsilon$ takes into account the possibility of noise present in image data. Equivalently, I solve the optimization problem below

$$
\min \frac{1}{2} \|D_h a_h - y_h\|_2^2 + \lambda \|a_h\|_1,
$$

where the Lagrange multiplier $\lambda$ balances the sparsity of $a_h$ and the $l_2$-norm reconstruction error. Similar remarks apply to the learning of $D_l$ for low spatial-frequency patches.

Once the above process is complete, I use the sparse coefficients $a_h$ and $a_l$ as the features for our SVR models, which learns the mapping functions between these input features and the associated pixel labels in high-resolution images.
1. 3 Support Vector Regression

1. 3. 1 SVR Learning

Support vector regression (SVR) [5] is the extension of support vector machine. Using kernel tricks, the task of SVR is to use nonlinear functions to linearly estimate the output function in high-dimensional feature space. Similar to SVMs, the generalization ability makes the SVR very powerful in predicting unknown outputs.

In training, my SVR solves the following problem

\[
\min_{w,h,\xi,\xi^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)
\]

\[\text{s.t.} \quad y_i - \left( w^T \phi(a_i) + b \right) \leq \varepsilon + \xi_i, \]

\[\left( w^T \phi(a_i) + b \right) - y_i \leq \varepsilon + \xi_i^*, \]

\[\xi_i, \xi_i^* \leq 0, i = 1,..., n \]

I note that \( y \) is the associated pixel label (at the same location as the center of the patch considered) in the high-resolution image, \( n \) is the number of training instances, \( \phi(a_i) \) is the sparse image patch representation in the transformed space, and \( w \) represents the nonlinear mapping function to be learned. \( C \) is the tradeoff between the generalization and the upper and lower training errors \( \xi_i \) and \( \xi_i^* \), subject to a threshold \( \varepsilon \). I note that Gaussian kernels are used in all our SVRs, and their parameters are selected via cross validation.

1. 3. 2 SVR Prediction

After the SVR models for high and low spatial frequency patches are learned, I use them to predict the high-resolution image of a given low-resolution test input. As the progress shown in Figure 1, I first synthesize the high-resolution version of the test input using bicubic interpolation, and categorize all image patches accordingly (as discussed in Sect. 2.1). Based on the categorization results, I use the training dictionary \( D_h \) or \( D_l \) to calculate the corresponding sparse coefficient vector \( \alpha \) for each image patch. Finally, I update the pixel values in the synthesized image using the previously learned SVRs in sparse representation domain and obtain the final SR image.

2. Result

(a) Ground truth  (b) Bicubic interpolation  (c) Proposed method
Figure 3. Example high-resolution images. Note that the face regions are scaled for detailed comparisons. Figure 3. shows the result of proposed method. The original image (a) is down scaled by bicubic interpolation (the magnification factor is 0.5), and then magnify each method, bicubic interpolation (b) and proposed method (c). The proposed method result show better performance than bicubic interpolation.

3. Reference


Object Recognition by Association Using Separate Distance Functions for Each Exemplar

Hejin Cheong
Department of Image
Chung-Ang University
Seoul, Korea
Hjcheong87@gmail.com

Abstract
For object recognition, generally, a novel object is explained solely in terms of a small set of exemplar objects to which it is visually similar. Inspired by this method, I propose a recognition method which learns separate distance functions for each exemplar. However, the distances are interpretable on an absolute scale and can be thresholded to detect the presence of an object. I use the distance functions to detect in novel images by association for each exemplar. The exemplars can be represented as image regions and the learned distances capture the relative importance of shape, color, and texture features for that region. The experiment results show the algorithm on real-world outdoor scenes from the LabelMe dataset.

1 Introduction
Object recognition is one of the important problems in computer vision. Usually, object recognition is assumed to mean object naming. However, since our language does not have a name for every possible object. Moreover, two visually similar objects that have been arbitrary assigned different labels (e.g. “boat” and “ship”).

For solving this problem, I propose the object recognition method not as object naming, but as object association, as shown figure 1. Recently, with the appearance of such large image collections, several systems have shown that simple k-nearest-neighbor (kNN) approaches can often perform surprisingly well [1], [2]. However, all these methods match the image as a whole, which effectively limits them to operating on the coarse scene level.

Figure 1: Recognition by association
In this project, I propose a recognition method which learns separate distance functions for each exemplar. To match individual objects within scenes, I partition the image into each object and use the distance functions to detect in novel images by association for each exemplar. And, I propose a largely data-driven approach which weakly uses the object labels to automatically learn for each exemplar a distance function and which subset of exemplars are similar to exemplar. The exemplars can be represented as image regions and the learned distances capture the relative importance of shape, color, and texture features for that region.

The main contributions of this paper are following. The first point is data association. In this setting, a novel object is defined solely in terms of a small set of exemplars to which it is similar. At the recognition stage, there is no mention of labels, categories or classes. This data-driven definition requires better ways of object matching. Another point is improvement nearest neighbor performance by learning interpretable per-exemplar distances. I learn individual distance metrics for each exemplar. But unlike theirs, the distances are interpretable on an absolute scale, and can be thresholded to perform detection. In addition to learning a distance, I also determine for each exemplar the subset of other exemplars that are similar to it. It can be to capture visual relationships within the dataset that were not reflected in the labels.

This paper is organized as follow. In Section 2, algorithm for learning associations between objects is discussed. Section 3 summarizes experimental results of the proposed method for finding data associations in novel images, Section 4 concludes the paper.

2 Learning Object Similarity

This section presents dataset which is used in training, segment features, and learning distance functions for recognition by association. As opposed to kNN methods, the proposed distance function can associate each potential input and a variable number of exemplars.

2.1 Dataset

A reasonably general class of images requires handling a large number of different objects that occur in everyday life. Therefore, the choice of the right training data is of the utmost importance. Of all the currently available datasets, the only one containing a large number of real-world scenes, with a wide variety of everyday objects that are not only labeled but also segmented, is the LabelMe dataset [3]. LabelMe is an ongoing online image annotating collaboration involving many labelers. I use a subset of LabelMe which consists of over 5000 images. The dataset which is used for training is composed mainly of images which have car, building, and tree.

Table 1: The Region-based features used to represent objects.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>Centered Mask</td>
<td>32x32</td>
</tr>
<tr>
<td></td>
<td>BB Extent</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Pixel Area</td>
<td>1</td>
</tr>
<tr>
<td>Texture</td>
<td>Right Boundary Tex-Hist</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Top Boundary Tex-Hist</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Left Boundary Tex-Hist</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Bottom Boundary Tex-Hist</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Interior Tex-Hist</td>
<td>100</td>
</tr>
<tr>
<td>Color</td>
<td>Mean Color</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Color std</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Color Histogram</td>
<td>3x11</td>
</tr>
</tbody>
</table>
2.2 Segment Features

I represent exemplars as segments. As shown table 1, each object segment is characterized with $N_f = 11$ different features (3 shape, 5 texture, 3 color features). Elementary distances are defined for each of these features to be simply the $L_2$ norm between the feature representations. To extract shape features, I compute the centered object mask in a canonical $32 \times 32$ frame, the size of the region, and the size of region's bounding box. To extract texture features, I compute normalized texton histograms [4] in the interior of the object, and, separately, along the boundaries of the object. For color, I compute the mean RGB-value, its standard deviation, and a color histogram.

2.3 Learning Distance Functions

The proposed distance functions are positive linear combinations of elementary distances. Similarly to [5], I focus on linear decision boundaries and pose each distance function learning problem as a SVM convex optimization problem.

Each exemplar has its own distance function. I denote exemplar $e$'s distance function as $D_e$. I define $d_{ez}$ to be the $N_f + 1$ dimensional positive distance vector between $e$ and input $z$ (the $j$-th component of $d_{ez}$ is just the $L_2$ distance between the $j$-th feature vectors of $e$ and $z$).

Each distance function is calculated by weight vector $w_e$ and takes the form:

$$D_e(z) = w_e \cdot d_{ez}$$  \hspace{1cm} (1)

In addition to $w_e$, each exemplar is associated with a binary vector $\alpha_e$. The length of $\alpha_e$ is equal to the number of exemplars with the same label as $e$. The non-zero elements of $\alpha_e$ are precisely the exemplars that should be similar to $e$. I learn $w_e$ and $\alpha_e$ simultaneously while keeping each exemplar's learning problem independent of the other distance functions. The learning problem is formulated as follows:

$$\{w^*, \alpha^*\} = \arg \min_{w, \alpha} \sum_{i \in \mathcal{C}} \alpha_i L(-w \cdot d_i) + \sum_{i \in \mathcal{C}} L(w \cdot d_i)$$  \hspace{1cm} (2)

where $w \geq 0$, $\alpha_j \in \{0,1\}$ and $\sum_j \alpha_j = K$. $L(\cdot)$ is any positive loss function, and $\mathcal{C}_e$ is the set of all exemplars with the same label as $e$. Without the $\alpha$ parameter and with no constraint on $w$, this is just the primal form that many convex statistical learning techniques (such as SVMs) can be cast in. Since the presence of the binary $\alpha$'s renders the problem non-convex, I proceed iteratively estimating $\alpha$ given $w$ and estimating $w$ given $\alpha$.

During the iteration, the value of the objective function is not increased and thus I can find a local minimum. I start with an initial distance function $w^0$ and proceed as follows:

$$\alpha^k = \arg \min_{\alpha} \sum_{i \in \mathcal{C}} \alpha_i L(-w \cdot d_i)$$  \hspace{1cm} (3)

$$w^{k+1} = \arg \min_{w, \alpha} \sum_{i \in \mathcal{C}} L(-w \cdot d_i) + \sum_{i \in \mathcal{C}} L(w \cdot d_i)$$  \hspace{1cm} (4)

Given $w^k$, I minimize equation 3 by setting $\alpha_i$ equal to 1 for the $K$ smallest values of $L(-w \cdot d_i)$, and 0 elsewhere. Given $\alpha^k$, the problem of solving equation 4 is just the classical convex statistical learning problem. This procedure converges when $\alpha^{k+1} = \alpha^k$.

After learning the distance functions, I apply each exemplar's distance function to all of the other exemplars and consider the support set $\text{Supp}(D_e)$ as $z \in \text{Supp}(D_e) \iff D_e(z) < 1$. In practice the resulting support sets wildly vary in size. For exemplars from generic classes such as "car" where we expect many cars to be rather similar, $|\text{Supp}(D_e)| \gg K$. I remove away the exemplars with an empty support set.

Figure 2 shows a result of classification by distance function in synthetic data. The training dataset is generated with 2 features, 6 labels, and 200 data in each label. The new input dataset has 900 data and the each data has evenly spacing features.
Several learned distance functions and the top elements in their support sets are shown and compared to the neighbors given a simple texton-histogram distance in Figure 2. The learned distance functions are doing a good job at combining elementary distances to measure similarity.

Figure 3: Data association in the training set.
3 Experimental Results

In order to determine if the distance functions are overfitting, we consider a segment-labeling task utilizing over 1000 objects extracted from a held-out subset of LabelMe. Thus, I use a test set of 159 outdoor images all coming from one specific subfolder in LabelMe. This testing subset contains a total of 1146 objects and 18 labels. I use a street picture as input image and the image is divided manually into each object. As shown figure 4, the proposed algorithm can successfully find associations.

![Figure 4: Data Association in the test set.](image)

4 Conclusion

This paper proposed a recognition method which learns separate distance functions for each exemplar. Based on the principle of data association, I associated a segment extracted from a novel image with visually similar exemplars. I showed that the integral component of such exemplar-based systems is the learning of exemplar-specific distance functions. The proposed algorithm can successfully find associations.
References


Camera Parameter Analysis Method for Manipulated Image Detection

Seokhwa Jeong

Abstract

Digital images are easy to be manipulated and edited by various image processing algorithms and editing software. Therefore, a detection algorithm of image manipulation and editing is needed to authenticate important images. This project, presents an improved manipulation image detection algorithm by analyzing image feature and the exchangeable image file format (EXIF) header in the JPEG images. I first estimate a modified weights values using learning and optimization of the training set from various camera brands. I also detect manipulated image using EXIF features and modified weights values. Consequently, the proposed manipulation detection method can detect manipulated image by analyzing reliable features.

1. Introduction

Advanced digital image processing algorithms and image editing software make it possible to produce indistinguishable forgeries. As a result, a digital image can no longer be trusted as legal evidence. In order to ensure authenticity of the digital image, a reliable manipulation detection method in needed. In this project, judges whether that is manipulated or not with metadata acquired from images. It extract EXIF information and statistical noise features with the various training images according to camera’s features and saves the information as a weight coefficient by using correlation of each feature. Also, to distinguish the manipulated things from others, it assumes the value which makes minimum errors of the weight coefficients and correlation of each feature by extracting EXIF header information and the features of statistical noise in the image. Consequently, if the error value is bigger than the critical value, it judges the image is manipulated.
2. Methodology

2.1 Extraction of image noise residue

Digital images contain unique sensor noise in the entire image. However, the sensor noise is modified during the manipulation process. Therefore, I estimate noise residue using denoising filter. Knowing that different denoising filter is effective in removing different types of image noises. I choose a set of denoising filters including averaging filter, Gaussian filter, median filter and Wiener filter for extracting different types of residuals. I statistically extract a set of image noise features based on four types of denoising filters. Since the noise estimation can be adversely affected by the heterogenous image content, especially in the sharp transitional area, I compute noise features based on the nonsharp image area.

2.2 Extraction of EXIF header features

The EXIF is a standard that specifies the formats for images, sound, and ancillary tags used by digital cameras, scanners, and other systems. Currently, the EXIF supports digital image in TIFF and JPEG formats for diverse camera brands and models. The EXIF header parameters such as ISO speed rating, exposure time, and F-number are commonly recorded into the JPEG image as shown in Fig 1.

![Fig. 1. A photo with the corresponding EXIF header information; (a) a JPEG image and (b) the corresponding EXIF header parameters.](image)

<table>
<thead>
<tr>
<th>EXIF header parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Number</td>
</tr>
<tr>
<td>5.6</td>
</tr>
</tbody>
</table>

Camera settings expressed in the log scale are used to represent the EXIF header. The EXIF feature, $y_f, f = 1, 2, 3,$ is extracted for a given forgery image.
Aperture value \[ y_1 = \log_2(N^2) \]
Shutterspeed value \[ y_2 = \log_2(1/t) \]
ISO speed rating value \[ y_3 = \log_2(I/3.125) \]

where \( N \) is the F-number, \( t \) is the exposure time, and \( I \) is the ISO speed rating value.

Object removal and copy-and-move are also popular factors of the forgery. However, the forgery images do not change the EXIF header file. So an improved detection algorithm should utilize this information.

Fig. 2. The forgery image where people are copied from Fig. 1.

### 2.3 Modeling the EXIF-Image correlation

I correlate each EXIF header feature with the image statistical noise features. Extracted noise feature vectors from learning image are represented \( Z_p = \{ Z_{pi} \mid i = 1, \ldots, C \} \) and EXIF feature vectors are \( y_p = \{ y_{pj} \mid j = 1, \ldots, J \} \). Here, \( Z_p \) and \( y_p \) form a pair of image feature vector and EXIF feature vector for the p-th image. \( C = 24 \) and \( J = 3 \) are the dimensions of the image and the dimensions of the image and the EXIF feature vectors respectively.

\[
X_p = \{ z_{p1}, \ldots, z_{pc}, z_{p1}z_{p2}, z_{p1}z_{p3}, \ldots, z_{p(c-1)}z_{pc}, \ldots, z_{p1}^2, \ldots, z_{pc}^2 \}
\]  

\( X_p \) is vector of selected \( Z_p \) features from eq (2) denotes new feature vector, which includes both the first-order and the second-order polynomials of noise features. The second-order polynomials are added to cater for nonlinearity of the underlying correlation. After processing all training images, a set of compact features is obtained. Suppose \( y_{pj} \) is the j-th EXIF feature of the p-th image from the intact image.

Next, weight vector is represented eq (3).

\[
W_j = \{ w_{ij} \mid i = 1, \ldots, S_j \}
\]
Where, $S_j$ is represent number of $X_p$ vector.

Error value is computed by difference EXIF feature vector and value of multiplying the noise feature vector with weight.

$$e_j = y_j - X_j w_j$$  \hspace{1cm} (4)

Consequently, EXIF correlation value that minimizes the error through modeling weights can be obtained.

$$\min(||e_j||^2) \Rightarrow w_j = (X_j^T X_j)^{-1} X_j^T Y_j$$  \hspace{1cm} (5)

### 2.4 Detection of image manipulation

In the above section, weight value is computed through EXIF correlation modeling using statistical noise feature vector and EXIF feature vector in learning image. In order to judge whether manipulated image, forged image is performed EXIF correlation modeling process as learning image. Next, compute statistical noise feature $X_j$ and EXIF feature $Y_j$ and represent EXIF feature $X_j W_j$ through weight vector. The estimation error, eq (6), between the genuine EXIF header feature, and the estimated header feature is computed.

$$E_j = |Y_j - X_j W_j|$$  \hspace{1cm} (6)

### 3. Experimental results

In this section, I demonstrate the performance of the proposed algorithm for various forgery JPEG images. To evaluate this method, it generates learning data as 455 images for various brands such as Samsung, Sony, etc. These photos are captured in the automatic mode with different camera settings. Also the proposed method is evaluate manipulation detect recognition rate using various JPEG images. And the estimation error value between the genuine EXIF header feature and the estimated header feature is evaluated as recognition rate of detecting image manipulation.

Fig. 3(a) shows an original JPEG image with camera parameters (f-number: 13, exposure time: 1/25 sec, and ISO: 100). Fig. 3(b) shows the forgery version with the copy-and-move manipulation.
Fig. 3. Experimental JPEG image.

Table. 1. Error value of intact JPEG image and manipulated image.

<table>
<thead>
<tr>
<th>EXIF</th>
<th>Error value of Intact image</th>
<th>Error value of Manipulated Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error value of Intact image</td>
<td>Error value of Manipulated Image</td>
</tr>
<tr>
<td>F-number</td>
<td>Exposure time</td>
<td>ISO speed rating</td>
</tr>
<tr>
<td>Remove of</td>
<td>9.11E-07</td>
<td>1.69E-06</td>
</tr>
<tr>
<td>object</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Conclusion

In this paper, presents a manipulation image detection algorithm by analyzing image feature and the exchangeable image file format (EXIF) header in the JPEG images. The model correlates the image statistical noise features with the EXIF header features. And I estimate weight values using learning data of training set and linear regression algorithm from various camera brands. These weights allow estimating the EXIF feature parameters from the noise features. Consequently, the proposed manipulation detection method can detect manipulated image by analyzing statistical noise features.

5. Reference

Human Detection Using HOG-LBP features

Gwanghyun Jo
Adaptive Filtering and Machine Learning

Abstract

We proposed human detection algorithm by combining Histograms of Oriented Gradients (HOG) and Local Binary Pattern (LBP) as the feature set. Proposed algorithm consists of three steps: i) Feature extraction by using HOG, ii) Feature extraction by using LBP, iii) Combination of HOG and LBP features and classification by using linear SVM. The proposed human detection algorithm can be applied in the wide areas of surveillance, robotics, smart vehicles, and automotive safety. Our method has been trained and tested on INRIA datasets.

1. Introduction

Human detection has very important applications in surveillance, robotics, smart vehicles, and automotive safety. However, human detection in images is challenging task owing to variations in appearance, body shape, pose, clothing, illumination and background clutter.

Dalal et al/1] proposed human detection method which extracts the Histograms of Oriented Gradients (HOG) as the feature vectors and employs the Support Vector Machine (SVM) as the classifier. HOG has been widely accepted as one of the best features to capture the edge or local shape information. It has shown great success in object detection and recognition.
However, HOG performs poorly when the background is clustered with noisy edges. LBP is complementary in this aspect. It can filter out noises using the concept of uniform pattern [2]. LBP is an exceptional texture descriptor and has been used in various applications and has achieved very good results in face recognition.

Therefore, we propose a more powerful human detector by combining the HOG and LBP feature. And then, the linear SVM is used to train and test the human classifier using the combined feature vector.

2. Human Detection Using HOG-LBP feature

The proposed human detection procedure based on the HOG-LBP feature is shown in Fig. 1.

![Fig. 1: The proposed human detection algorithm using HOG-LBP features](image)

A. Histograms of Oriented Gradients

The Feature Extraction (HOG) Block extract the Histograms of Oriented Gradients feature vectors. HOG is an excellent descriptor for capturing the edge direction or the distribution of local intensity gradients of objects. It has been applied successfully to detect the whole body of human.

Firstly, for each detection window we compute its gradient magnitude using 1-D masks, i.e., \([-1 0 1\]). Secondly, we divide the gradient magnitude of the image into non-overlapping blocks. The gradient magnitude of each pixel in the cell is voted into 9 bins according to the orientation of
the pixel's gradient. The nine orientation bins are evenly spaced over 0-180 ("unsigned" gradient).

Thirdly, each block is represented by a 36-D feature vector that is normalized by the L2-norm to reduce the influence of the local variation in illumination and foreground-background contrast. Finally, the feature vectors of the blocks are concatenated into the feature vector.

B. Local Binary Patterns

The second block in Fig. 1 extract the Local Binary Patterns feature vectors. LBP is an excellent texture descriptor for its invariance to gray-scale and rotation. Therefore, the combination of the shape information and the texture information can describe the human body better and thus enhance the detection performance. The LBP patterns we used is \(LBP_{8,1}^{u} \), where the subscript denotes that 8 points with radius 1 are sampled for each pixel, and the superscript stands for using only uniform patterns. A binary pattern is called uniform pattern if the binary pattern contains at most two bitwise transitions from 0 to 1 or vice versa when the bit pattern is considered circular. We then use the L2-norm to normalize the histograms of the blocks. Finally, the LBP feature vectors are concatenated into the final LBP feature.

C. Classification

We extract the HOG feature and the uniform LBP feature and then combine both as our training feature set. Hard examples (false positives) are necessarily added to the negative set to re-train and then obtain the final detector. Then the feature set is fed to a linear SVM classifier to get a preliminary detector.

3. Experimental Results
A. Dataset

We implement our algorithm on the INIRA dataset, which contains 2416 96*160 positive samples and 1218 negative images for training. The positive samples are cropped to standard size of 64*128 pixels in our experiments. The negative examples are randomly sampled from the 1218 negative images. The INRIA dataset also contains 1132 positive examples and 453 negative images for testing. Each of these positive examples includes at least one person and is variable in clothing, illumination and background. Some of the positive training samples are shown in Fig. 2.

Fig. 2: Some of the positive samples from training data set

B. Training and Evaluation

We use the linear SVM to train and classify on the INRIA dataset. To quantify the detection performance, we use DetectionError Tradeoff (DET) curves, and plots of miss rate versus false positives per window (FPPW). Lower miss rate means better detection performance on the same FPPW.

In the experiment, we contrast the classification performance between LBP detector, HOG detector and HOG-LBP detector. The experiment results show in Fig. 3. We achieve 92% detection rate at $10^{-1}$ FPPW for the HOG feature and 79% detection rate at $10^{-3}$ FPPW for the LBP feature. For the HOG-LBP feature, the detection rate is 97%, which performs best. Therefore, the
combination of HOG and LBP feature can compensate for their inadequacy and improve the detection rate significantly.

![Performance Comparison](image.png)

Fig. 3: The performance comparison of and HOG, LBP and HOG-LBP on INRIA dataset

### 4. Conclusion and Future Work

We propose a human detector which uses HOG-LBP as the feature set. Compared to the state-of-the-art algorithms, our detector is more distinctive leading to significant improvements in detection accuracy.

Although linear SVM detector is reasonably efficient. There is still room for optimization and to further speed up detectionsit would be useful to develop a coarse-to-fine. Also, the resulted HOG-LBP feature contains important information on how to separate human from other objects, yet redundant information may also be included in the feature. So we will applied to learn a new feature from the HOG-LBP feature using Adaboost [4].
Reference


Abstract

This project presents particle sampling-based fast background subtraction for object detection. The proposed method is designed to quickly focus foreground region in a probabilistic way. For the estimation of the attentional region, a foreground probability map is generated by spatial, temporal, frequency properties of foregrounds. By using this foreground probability map as weight of importance sampling, particle with a high weight makes adaptive square kernel to cover all object region for detection.

1 Introduction

1.1 Problems of the conventional method

Background subtraction is popular method which aims to segment moving object from a relatively stationary backgrounds. Recently pixel-based probabilistic model methods gained lots of interests and have shown good results. There have been many improvements in detection performance for these methods under various situations, but the computational time still takes too much time. Computation time reduction issue is getting more important in a systematic view, because the background subtraction is generally considered as a low level image processing task, which needs to be done with little computation, and video sizes are getting bigger.

To reduce computation time of background subtraction methods, several approaches have been studied. The first type of approach is based on optimizing algorithms. Although the Gaussian mixture model (GMM) scheme proposed by Stauffer and Grimson[1] works well for various environments, it suffers from slow learning rates and heavy computational load for each frame.

The second type of approach is using parallel computation. Multi-core processors in a parallel form, using the OpenMP system are applied for speed-up. Also Graphical Processing Units (GPUs) are used to achieve real-time performance with computationally heavy algorithms. They could successfully achieve speed-up, but special hardware resources are required.

A selective sampling based speed-up method is the third type of approach. Kim et al.[4] presented a sampling mask designing method which can be readily applied to many existing object detection algorithms. This algorithm compactly designed grid pattern masks to detect small objects, but these grid patterns still cause redundant operations. In this project, a new method of the third type of approach is proposed, it can be utilized together with the other two approaches. This project aim to find an active attentional sampling solution which can be
generally applied to most conventional background subtraction methods. A foreground probability map based on spatial, temporal, and frequency properties of the foreground region is designed. Using previous foreground detection result, the foreground probability map is updated. By combining with conventional background subtraction methods, this method makes these methods even be able to handle full HD videos.

In particular most pixels from surveillance video are background region, and foreground region takes very small portion (2~10%) in both spatially and temporally. The proportions of foreground regions are very small. Hence, if background subtraction can be focused on foreground area, necessary calculation would be reduced significantly. In this project, attentional region in a current frame is considered as foreground region by spatial property detected in a previous frame.

1.2 Overview

Proposed method approaches stochastic-search using a particle filter. The three property (spatial, temporal, frequency) of foreground pixel is used as probability of being the foreground.

To cover all foreground region, adaptive kernel, \( w \times w \) square shape by the weight, is adopted at each particle point, as shown below picture. \( w \) is kernel size.

Object region is determined by these kernels. And object region is used for initial condition of object tracking.

Fig 1. Flowchart of proposed method

2 Estimation Probability Map

The background subtraction task finds a sequence of detection masks \( \{D^1,\ldots,D^T\} \) using a sequence of video frames \( \{I^1,\ldots,I^T\} \).

Each video image \( I^T \) and detection mask \( D^T \) are composed of \( N \) pixels \( \{I^T(1),\ldots,I^T(N)\} \) and \( \{D^T(1),\ldots,D^T(N)\} \) respectively. All the masks are binary masks. The detection mask at pixel \( n \) shall be denoted with the symbol \( D(n): D(n) = 0 \) if pixel \( n \) belongs to the background and \( D(n) = 1 \) if it belongs to the foreground.

2.1 Estimation of Foreground Properties

Estimation models are proposed to measure the spatial, temporal, and frequency properties of each pixel. The three property measures are referred to as \( \{M_T, M_S, M_F\} \). The temporal property measure \( M_T \) is estimated by the recent history of detection results. The spatial property \( M_S \) is estimated by the number of foreground pixels around each pixel. The frequency property \( M_F \) is estimated by the ratio of detection result flipping over a period of time. All estimation models are updated by a running average method, with learning rates \( \alpha_T, \alpha_S, \) and \( \alpha_F \) (all learning rates are between 0 and 1). The estimation models for the measures of the properties are given in the following.
(1) **Spatial property** ($M_S$)

$M_S$: Detection results of nearby pixels are used to measure the spatial coherency of each pixel $n$.

$$M_S'(n) = (1 - \alpha_S)M_S'^{-1}(n) + \alpha_S s'(n),$$

$$s'(n) = \frac{1}{w^2} \sum_{i\in\mathcal{N}(n)} D'(i),$$

where $\mathcal{N}(n)$ denotes a spatial neighborhood around pixel $n$ ($w \times w$ square region centered at $n$). $M_S'(n)$ closer to 1 means high probability of being a part of the foreground.

(2) **Temporal property** ($M_T$)

$M_T$: At each location $n$, a recent history of detection mask results at that location are averaged to estimate the property.

$$M_T'(n) = (1 - \alpha_T)M_T'^{-1}(n) + \alpha_T D'(n),$$

As the value of $M_T'(n)$ comes close to 1, the possibility of foreground appearance at the pixel is high.

(3) **Temporal property** ($M_F$)

$M_F$: If detection results have been changed twice during previous three frames, it is considered as a clue of dynamic scene.

$$M_F'(n) = (1 - \alpha_F)M_F'^{-1}(n) + \alpha_F f'(n),$$

$$f'(n) = \begin{cases} 255 & (D'^{-2} = D'^{-1} = D') \\ 0 & \text{otherwise.} \end{cases}$$

where $f'(n)$ denotes a frequently changing property at $n$. Unlike the other measures, the pixel $n$ has a high probability of being a foreground, as the value $M_F'(n)$ is close to 0.

### 2.2 Foreground Probability Map: $P_{FG}$

By estimating the three foreground properties, three measurements, $M_T$, $M_S$, and $M_F$, every measurement has a value between 0 and 255. So the foreground probability is defined for a pixel $n$ at frame $t$ as

$$P_{FG}^t(n) = M_T^t(n) \times M_S^t(n) \times (255 - M_F^t(n)).$$

The foreground probability map $P_{FG}^t(n)$ is a composition of $\{P_{FG}^t(n)\}_{n=1}^{N}$. Fig 2 shows estimation result of probability map.
3 Salience region detection by particle filter

3.1 Particle Sampling using $P_{FG}$

$P_{FG}$ is used to find salience region as log-likelihood of particle filter. Particle filtering model should be considered in state space. The state of particles can be represented as $s = [x, y, v_x, v_y]^T$, and the state transition model with constant acceleration can also be defined as

$$
x_k = x_{k-1} + v_x^{k-1} \Delta k,
\dot{y}_k = y_{k-1} + v_y^{k-1} \Delta k,
\dot{v}_x^k = v_x^{k-1},
\dot{v}_y^k = v_y^{k-1},
$$

(5)

where $(v_x^k, v_y^k)$ represents the velocity of the particle. The discrete version of the state transition equations with $\Delta k = 1$ can be expressed as

$$
s_k^- = As_k^+ + w_k,
$$

(6)

where

$$
A = \begin{bmatrix} 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } w_k = \begin{bmatrix} N(0, Q_{pos}) \\
N(0, Q_{pos}) \\
N(0, Q_{vel}) \\
N(0, Q_{vel}) \end{bmatrix}.
$$

(7)

The measurement $m_k$ can be represented in the predicted state $s_k^-$ with $P_{FG}$, and the likelihood is calculated as

$$
L = -\log(2\pi R) - \frac{\|m_D\|^2}{2\pi},
$$

(8)

where $R$ represents the standard deviation of the $P_{FG}$ estimation noise, $\|m_D\| = m_0 - m_k$, where $m_0$ represents the target $P_{FG}$. We then compute the cumulative sum using regularized log-likelihood as

$$
c_i = \sum_{m=1}^{i} \frac{L_i}{L_{total}}, \text{ for } i = 1, \ldots, N,
$$

(9)

where $N$ is the total number of PPs, $L_{total}$ represents the total sum of $L$. Then, real random numbers $u_i$ that uniformly distribute in [0,1] is generated. Then, resampling is performed to obtain a posteriori particles $s_{k,i}^+$ based on $L_i$. For $i = 1, \ldots, N$, we find a positive integer $j$ such that $c_{j-1} < u_i$ and $c_j \geq u_i$, and then update the current state as $s_{k,i}^- \leftarrow s_{k,j}^+$.

3.2 Adaptive Kernel Generation

Particles are resampled to around object by $P_{FG}$. However, resampled particles are hard to cover whole object region. It is therefore necessary to fill the space between sparse points in

---

Fig 2. Estimation result of probability map; (a) input frame, (b) spatial property, (c) temporal property, (d) Frequency property, (e) probability map.
the foreground region. Adaptive size kernel is proposed to construct a full foreground region. Kernel size of \( w^t(i) \), is defined as follow

\[
 w^t(i) = \text{round}\left( L_R(i) \times \sqrt{\frac{k}{N/N_s}} \right) 
\]  

(10)

where \( L_R(i) \) is regularized value of log likelihood of \( i \)-th sample in [0, 255], \( N \) is total number of the image, \( N_s \) is the number of the particle (usually, 1 ~ 3% of total pixel is used), and \( k \) is the kernel expanding constant. Fig 3 shows resampled particles and adaptive size kernel generation results.

Adaptive size kernel is defined as salience region. Therefore background subtraction is applied only this salience region, and GMM [1] is applied.

4 Experimental results

Proposed method is evaluated in full-HD and 720×576 resolution sequence. The results are compared to the Gaussian Mixture Model background subtraction [1]. I implemented proposed algorithm in Matlab 2013 with Intel core i7 processor.

Experimental results of performance of proposed method are shown in Figure 4 and Figure 5.
Fig 5. $P_{FG}$ and GMM results in 720×576 sequence (PETS 2009); $P_{FG}$ result of (a) the 700th, (b) 720th, (c) 740th, and (d) 760th image, GMM result of (e) the 700th, (f) 720th, (g) 740th, and (h) 760th image.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Average time [s]</th>
<th>Resolution</th>
<th>Average time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM [1]</td>
<td>1920×1080</td>
<td>720×576</td>
<td>9.71</td>
</tr>
<tr>
<td>Proposed method</td>
<td>49.24</td>
<td>12.59</td>
<td>3.88</td>
</tr>
</tbody>
</table>

Table 1. Comparison result of computational time with conventional method

Table 1 shows Comparison result of computational time with conventional GMM. Though computational time is depend on object ratio in the image, it was efficient in general image.

**Conclusion**

In this project, salience region detection method for speed-up of background subtraction is proposed. And Experimental results show that this method reduces computational time efficiently. Also it can be applied such as HD and UHD surveillance system. In the C++ environment, I will be work real time.

**References**


A Performance-Comparison Study Between SVM And Neural Network On Face Recognition On Same Database

Cesar Niyomugabo
Department of Imaging Science and Art
Graduate School of Advanced Imaging Science, Multimedia and Film
Chung-Ang University
Seoul, Republic of Korea
niyombofils@gmail.com

Abstract—This paper presents a project about a comparative investigation on recognition performance between the SVM and Neural Network machine learning algorithms. AdaBoost method is used for faces extraction from images and a single database is constructed with these faces. Experiment results show that both algorithms have same strength based on measured accuracy of each of those algorithms.

Keywords—Performance-comparison, SVM, NN, face recognition, same database.

I. INTRODUCTION

Face recognition is a technology behind an artificial system that can scan a person's face and match it against a library of known faces. The technology may be used for biometric identification where an automatic identification of a person from a digital image or a video frame from a video source is needed. So far, different recognition systems have been implemented; however still on a given dataset different developed algorithm performances are not same [1]. In this regard this project aims to investigate which performs best between SVM and NN on same dataset of faces. Accuracy is taken as metric for our performance computation.

The remaining part of this document is organized as follows: Section II talks about face detection. Section III is devoted to face recognition. Section IV discuss about experiment and results. Section V draws the conclusion of project.

II. FACE DETECTION

Face detection is about determining whether or not there are any faces in a given image and estimate the location and size of any detected faces. In this project, face detection based on haar-like features and Adaboost is called in use [2]. Adaboost trained classifier is used to confirm face presence in a given image.

Viola and Jones came up with a face detection method based on a huge set of haar-like features which are computed in scaled analysis windows. The rectangular Haar-like features are sensitive to edges, bars and other similar structures in the image and they are computed using an efficient method based on the integral image concept. After computation of a huge number of features for each analysis, the AdaBoost algorithm is used for combining a small number of these features to form an effective classifier.

As example, for an analysis window of size 24x24, there are around 160 000 features, far more than the number of pixels. A variant of AdaBoost is used both to select the best features and to train the final classifier [3].

III. FACE RECOGNITION

After face presence confirmed in the image, then recognition is our next step, both machine learning algorithms, SVM (Non-linear SVM) and Neural Network (Backpropagation) are called to match detected faces against faces already in the database, (performance will be based on how algorithms recognize true class of each image. One part of the database is used for training and another one used for testing). Here is how they work:

A. SVM learning algorithm

Support Vector Machines (SVMs) are useful techniques for data classification. A classification task usually involves separating data into training and testing sets. Each element in the training set contains one target value (the class labels) and several attributes” (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes [4]. Following is the concept definition in case of a binary classification:-

Given a set of Data points D,

\[ D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}^{n}_{i=1} \] (1)

Where, \( x_i \) is a point in p-dimensional vector.
\( y_i \) is the corresponding class label.
We search for \( \omega \in \mathbb{R}^n \) and bias \( b \), forming the Hyperplane H:
\[
\omega^T x + b = 0
\]  
(2)
That separates both classes so that:
\[
\begin{align*}
\omega^T x + b &= 1, & y = 1 \\
\omega^T x + b &= -1, & y = -1
\end{align*}
\]  
(3)
The optimization problem that needs to be solved is:
\[
\min \frac{1}{2} \omega^T \omega
\]  
(4)
Subject to: 
\[
\begin{align*}
\omega^T x + b &\geq -1, & y = 1 \\
\omega^T x + b &\leq -1, & y = -1
\end{align*}
\]
Notice: Nonlinear SVM is used in our work.

### B. Neural Network machine learning algorithm
An Artificial Neural Network is a biological inspired computational model. Inputs multiplied by weights result in activation and form the output of a network. In this project we will use Backpropagation as kind of NN; this one goes like follows [5]:

1. Initialize weights with random values
2. Present the input vector to the network
3. Evaluate the output of the network after a forward propagation of the signal
4. Calculate \( \delta_j = (y_j - \sigma_j) \)

\( \sigma_j \) is the target output of neuron \( j \) and \( y_j \) is the actual output:
\[
y_j = g(\Sigma_i \sigma_i x_i) = (1 + e^{-x_i})^{-1}
\]  
(6)
(When the activation function is of a sigmoid type).
5. For all other neurons (from the first to the last layer) calculate
\[
\delta_j = \sum_i \sigma_i \cdot g'(x) \delta_i
\]  
(7)
\( g'(x) = y_j (1 - y_j) \)
(8)
6. Update weights with
\[
\omega_j(t+1) = \omega_j(t) - \eta y_j (1 - y_j) \delta_j
\]  
(9)
\( \eta \) is the learning rate.
7. Termination Criteria. Go to Step 2 for a fixed number of iterations or an error.

The network error is defined as:
\[
E = \frac{1}{2} \sum_{j=1}^{w} (d_j - y_j)^2
\]  
(10)

### IV. Experiment Result

#### A. Database:
We used a part of Georgia Tech face database; two distinct subjects (persons) (15 images per subject) are selected. Images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images are taken against a complex background with the subjects in an upright, frontal position (with tolerance for some side movement). 10 images for each subject are used for training, rest used for testing.

#### B. Software used:
For implementation and experimental test, OpenCV and C++ are used for the development of this project’s application.

#### C. Accuracy evaluation:
The performance between SVM and NN are evaluated based on Accuracy computed as follows [5]:
\[
\text{Accuracy} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}
\]  
(11)
True_positive and false_positive stand for a number of well and wrong classified images respectively, by each of above said algorithms.
Bellow a flowchart of the work is displayed:

![Flowchart](image)

Figure 1. Shows the flowchart of the work, the two learning algorithms are applied on same data set, results are displayed in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>SVM</td>
<td>NN</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The table1. Shows the experiment results which explain that the two methods performances are equal on used database (all faces were recognized as required).
V. CONCLUSION

This work presents a comparative study on recognition performance between the SVM and Neural Network machine learning algorithms. AdaBoost method was called for faces registration from images, a single database is constructed based on these faces. Experiment results show that both algorithms have same strength based on measured accuracy of each.

ACKNOWLEDGMENT

We would like to thank our Professor Monson H. Hayes for the useful course; “Adaptive Filter and Machine Learning” and for every help he provided for us in this work.

REFERENCES

[4] Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin “A Practical Guide to Support Vector Classification” National Taiwan University, Taipei 106, Taiwan.