VOL. 11, NO. 15, AUGUST 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A NOVEL TEMPLATE MATCHING IMPLEMENTATION OF OBJECT BASED IMAGE CLASSIFICATION BASED ON MULTIKERNEL FUSION SPARSE REPRESENTATION

Shivakumar G. S.[1], S. Natarajan[2] and K. Srikanta Murthy[3]
[1]Department of Computer Science and Engineering, Srinivas Institute of Technology, Mangalore, India
[2]Department of Information Science, PESIT, Bangalore, India
[3]Department of Computer Science and Engineering, PESSE, Bangalore, India
E-Mail: shivakumar_gs1@yahoo.co.in

ABSTRACT

This paper introduces and implements a novel object based image classification method on remote sensing images. The novelty introduced in this implementation is the application of a Multikernel Sparse Representation method on the object based image classification. The template-matching algorithm inspired from the object tracking implementation replaces the process of segmentation usually applied in object based image classification. The Multikernel fusion sparse representation based learning and prediction method is developed for remote sensing image classification. A particle filter framework for the sample template selection with the Multikernel Fusion Sparse Representation optimization technique is used to develop the image classification algorithm. The particle filter will act as the template-matching framework for our classification algorithm and the optimization of the observation model of this framework is carried out using the Multikernel Fusion Sparse Representation. Multikernel implementation has been proved to be more accurate than the feature extraction techniques since it extracts the internal intricacies of the image vector. The Kernels consume lesser memory space and lesser computational complexity compared to the traditional feature extracting methods. Multikernel Sparse representation has been proved to be more accurate and less computationally complex while implemented in other applications like the video object tracking. Affine transform based templates are extracted from the image which have to be trained and the kernel matrix is generated which is used for comparison with the templates extracted from the test images. Kernel Coordinate Descent (KCD) algorithm is used to find the similarity measure between the database kernel and the testing kernel. The weight values updated using the observation likelihood method that would indicate whether the test template matches with the database templates. The comparison is carried out with the multikernel method using the SVM classifier. The results that are observed are kappa coefficient and overall accuracy, which measure the classification accuracy, for images with higher and lower illumination and also the images are analysed for robustness to direction change and the classification performance for two different hyperspectral images.

Keywords: active learning, sparse representation, remote sensing, classification.

INTRODUCTION

Conditional Probability is the probability of occurrence of an event provided another event had occurred. Posterior Probability is the conditional probability, which considers the relevance between the events that occurred due to the conditional probability. Particle framework defined in the literature [18] is based on the posterior probability method on the video tracking implementation. Posterior probability defines the most certain occurrence of an event. The recent literature used the particle filter framework on image classification where the filtering, labeling and statistics are carried out using particle filter framework [17]. Sparse representation is castoff to symbolize the data under study in a comparable and expressive manner in a smaller memory space. SR-based algorithm has a drawback of high computation cost and less accuracy which is overcome by using kernel sparse representation (KSR)-based algorithm using single feature kernel and it also avoids introducing large number of trivial templates which will speed up compared to SR-based methods. Furthermore to overcome the weakness of single feature in object description, multikernel fusion method is proposed for multiple features integration. Furthermore to improve the method to have higher classification accuracy we utilize a prediction strategy called the adaptive multikernel fusion. Sparse representation would be a signal specific basis pursuit method that would be a more accurate representation for any non-linear signal. The deeper insight of sparse representation into the data structure would make it eligible for the classification algorithms of remote sensing. The sparse representation is the most compact representation by using the linear combination of the building blocks of the data. Remote sensing image classification using sparse implementation and active learning method is taken into account for implementation. The active learning method is the process where the data under learning has to be decided by the human intervention in order to have higher redundancy [1]. SVMs can be considered as techniques which use hypothesis space of linear separators in a high dimensional feature space, trained with a learning algorithm from optimization theory that makes a learning bias derived from statistical learning theory [2, 3]. Support vector machine is one of the important methods for use in the active learning method. The classification was binary but now the multiclass Support Vector Machine has been introduced for better classification.

This paper considers Sparseness as the measure of probability of evidence or posterior probability. The likelihood measurement employed in this paper considers the templates to be the events that are checked for the probability of evidence. The particle Filter framework is employed to develop the templates that serve to develop the implementation towards the likelihood function implementation. The Indian pines database is used and the implementation is carried out with MultiKernel Fusion based Sparse Learning. The method involves the MultiKernel Fusion for the feature extraction. The Sparse Learning methods would employ the likelihood method for assessing the class. As the MultiKernel Sparse Representation based implementation works on the Bayesian framework of Posterior Probability, we mention it as a sparse learning method. Kernel feature chosen for this sparse learning method are Histogram of Gradient (HOG) and Colour Histogram (CH). Matlab based implementation is carried out and the comparative Analysis with different images from the Indian pines database by finding the Kappa Coefficients for each database image is implemented. The same is furthered with and without tilting the images during the test process.

KERNEL METHODS

Kernels are instance-based learners usually used in Support Vector Machines (SVM). By using a limited number of samples the kernel could learn non-linear functions with lesser computational cost. Kernel operate in higher dimensional data by just finding the dot product between the trained samples in the feature space, but without computing the coordinate of that data in the space. Primary benefit of kernel is that it can be used in same algorithm with different kernel functions, and same kernel function with different algorithms.

Let us consider M to be a function in feature space. Selecting a function $M(\varphi, x_i) \rightarrow y_i$ where x_i is a set of input vectors and y_i is the desired output that represents an algorithm which is a two step process: firstly, we select a kernel function, and secondly, an algorithm that uses the kernel function and then learns the set of parameters φ. Different implicit features and distance relationships are specified by different kernel functions between the input vectors where as distance information is used by different algorithms from kernel function in order to learn the parameters φ which determines the function M.

Considering high dimensionality feature space M that maps the input vector to the desired output. We require the input vectors to be linearly separable which is done by kernel method. In order to separate the data linearly we require a method that remaps the input space into the feature space.

Mapping φ can be defined as

$$\varphi: x \rightarrow \varphi(x) \in F \quad (1)$$

By this mapping φ , dimensionality of input space is reduced and requires less number of training samples. Here *F* is the implicit mapping function.

Feature spaces with kernel methods are totally different. Initially, the feature space is not handcrafted but the kernel selected determines it. Different kernels specify different feature spaces implicitly. Secondly, the data is linearly separable since the feature space is of higher dimensionality. Depending on the easiness to linearly separate the data and the algorithm employed, the performance of the kernel method is known. The selection of the right kernel for a particular data distribution is essential which makes the data linearly separable.

Definition of a Kernel

A *kernel* is a function *k* that for all $x, y \in R^n$ satisfies

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle \quad (2)$$

Where φ is defined in (1) [5].

Naturally a kernel is a function that computes similarity between the two vectors. The value of kernel is higher for those two vectors when they are close to each other. Here a kernel is a mapping from $R^n \times R^n$ to R which is given by $(x, y) \mapsto k(x, y)$

Where $x, y \in R^n$ and $k(x, y) \in R$,R is the domain

An example of a kernel function of two vectors is the dot product:

$$k(x, y) = \langle x, y \rangle \quad (3)$$

Here in this case $\varphi(x) = x$. Therefore this kernel function computes distances in input space rather than feature space.

We consider the direct method to compute the kernel function [5, 6]. Taking an example of two-dimensional input space $X \subseteq R^n$ and a feature mapping

$$\varphi: x = (x_1, x_2) \mapsto \varphi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = R^3$$

For this feature mapping the kernel function is computed in the following way:

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle \quad (4)$$

$$= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (y_1^2, y_2^2, \sqrt{2}y_1y_2) \rangle$$

$$= x_1^2y_1^2, x_2^2y_2^2, 2x_1x_2y_1y_2 \quad (5)$$

Hence, we compute φ (x), then φ (y), and then their dot product. The drawback of this method is the number of calculations. The computation of φ (x) and φ (y) and their dot product requires a long time if φ is a mapping from a lower dimensional input space to a high dimensional feature space.

The Kernel trick

There is a different approach to computing the kernel function that speeds up computation time

considerably. Consider the same two-dimensional input space $x \subseteq R^2$ and feature mapping (4). Then according to (5) the kernel function is given by

$k(x,y) = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$

Note that

$x_1^2y_1^2 + y_2^2 + 2x_1x_2y_1y_2 = (x_1y_1, x_2y_2)^2 = \langle x,y\rangle^2$

and hence

$k(x,y) = \langle x,y\rangle^2$

is a kernel function for the feature mapping φ. The last computation is clearly much faster than computing the coordinates of each vector in feature space and then taking the dot product. The kernel trick allows us to avoid computing the coordinates in feature space of each input vector explicitly and it gives the result directly into the feature space [5, 6].

Also note that the feature space is not uniquely determined by the kernel function.

For

$\varphi: x = (x_1,x_2) \mapsto \varphi(x) = (x_1^2, x_2^2, x_1x_2, x_1x_2) \in F = R^4$

We have the same kernel function $k(x,y) = \langle x,y\rangle^2$.For simplicity we usually choose a particular kernel function and consider one of the feature mappings for that kernel function as our feature space.

The Kernel matrix

In this paper the learning is just creation of the kernel matrix as the kernels are instance based learners. To simplify data representation and also to better visualize kernels, we make use of the *kernel matrix* notation. Given a finite dataset $S= \{x_1, x_2 \ldots . x_n\}$, we represent all possible values of the kernel function for the dataset in the following kernel matrix notation:

$$K=\begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \text{ and}$$

$K_{ij} = k(x_i,x_j)$, where i, j=1…. n

Moreover, the algorithms use the kernel matrix to improve performance by avoiding the re-computation of the kernel function on the input data. This kernel can be seen:

$k(x_i,x_j) = \langle x_i,x_j\rangle$

This is the simplest kernel function and it can be used with algorithms to learn on linearly separable data where the feature space is the input space. The reason is that the feature mapping function $\varphi(x) = x$ is just the identity function.

Kernel fusion

The mapping functions or the feature spaces for this implementation are Histogram of Gradient $\mathbf{h_g}$ and Color Histogram $\mathbf{h_c}$.Each of features space for different templates created in the particle filter framework is given to kernel function to find the similarity among the training templates. We determine the relative similarity of the two samples considered using Bhattacharyya Coefficient [15]. After finding the Bhattacharya coefficients for each feature space among different templates kernel fusion is carried out to get one integrated representation. Bhattacharya coefficients for $\mathbf{h_g}$ feature space is denoted by k_g and that of $\mathbf{h_c}$ is denoted by k_c. These two coefficients are fused in order to get the fused kernel k.

Kernel sparse representation

In order to reduce more expensive computation we introduce kernel sparse representation, which would introduce a very fast and simple method of sparse representation. It implements a kernel trick on both the training samples (X) and the testing samples (y).

It introduces a function called φ(·) which would map a feature vector into the kernel space. φ(·) which satisfies $\varphi(x)^T\varphi(x)=1$ when $\|X\|_2^2 = 1$, which is the condition for convexity.

The KSR can be written as[1]

$$\hat{\beta} = \min_{\beta}\left(\frac{1}{2}\left\|\sum_{i=1}^{n}\beta_i\,\varphi(X_i) - \varphi(y)\right\|_2^2 + \lambda\|\beta\|_1\right) \quad (6)$$

Recently, the KSR model has been applied to image classification [6]. In kernel method we need to find inner production so the formula can be rewritten as following[1]

$$\hat{\beta} = \min_{\beta}\left(\frac{1}{2}\beta^T K\beta + K(\cdot,y)\beta + \lambda\|\beta\|_1\right) (7)$$

Where K is an n × n kernel matrix satisfying

$$K=\begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \quad (8)$$

and $K(i,y)=\boldsymbol{\varphi(X_i)^T\varphi(y)}$ is an n×1 vector.

The advantages of KSR on SR are as follows:

a) The function φ(·) can be regarded as a feature extraction function.KSR can introduce sophisticated feature which will be numb to occlusion and illumination variation.

b) Multiple features can be easily introduced in KSR by applying multikernel fusion. One branch among all multikernel fusion methods is the weighted multikernel fusion, in which a weighted summation is used to obtain a kernel[1]

$K=\sum_{i=1}^{n}\omega_i K_i$ (9)

Where K is the fused kernel;$K_i$is the kernel of *i*th feature andω_i is its corresponding weight, satisfying $\sum_{i=1}^{n}\omega_i = 1$ and $\omega i \geq 0$. Similarly, the kernel vector [1]

$\mathbf{K}(.,\mathbf{y}) = \sum_{i=1}^{n}\omega_i K_i(.,y)$ (10)

Adaptive Multikernel fusion methods focus on the calculation of ω_i. Making use of single feature it's difficult to build a robust image classification. For example considering the color feature alone will be robust for partial occlusion and deformation but sensitive to illumination variation while the edge feature is robust to illumination variation but fail in background clutter. So it's better to integrate multiple features and make them complementary. This can be achieved by KSR with multikernel fusion.

Optimization of Kernel sparse representation

Kernel Coordinate Descent (KCD)

By using KCD algorithm we can obtain coding vector β . KCD uses coordinate descent approach in [7] due to its simplicity and efficiency. Differentiating E(β) with respect to β_j and set it to 0 we get [1],

$$e(x_j) = \varphi(X_j)^T(\varphi(X_j) - \sum_{i=1,i\neq j}^{n} \beta_i \varphi(X_i))$$
$$= K(x_j, y) - \sum_{i=1,i\neq j}^{n} \beta_i K(x_j, y) \quad (11)$$

$\beta_j$is changed independently $\{\beta_i\}_{i=1,i\neq j}^{n}$
When calculating $\beta_j$we fix$\{\beta_i\}_{i=1,i\neq j}^{n}$, Hence β_j is calculated as

$$\beta_j = sgn\left(e(x_j)\right)[\,|e(x_j)| - \lambda]. \quad (12)$$

In summary, by making efficient use of KCD algorithm, we can update iteratively the coding vector β by (12). The initialization of β is obtained by kernel ridge regression [1]

$$\beta_{init} = (K + \gamma I)^{-1} K(\cdot, y) \quad (13)$$

Where γ is set to a small positive value. In practice, we set to $\gamma = 2\lambda$. During implementation, the iterative number maxIter is set to five, which is sufficient to obtain a sparse solution.

Particle filter framework

The particle filter is a Bayesian sequential importance sampling technique for determining the posterior distribution of state variables characterizing a dynamic system. It provides a convenient framework for estimating and propagating the posterior probability density function of state variables regardless of the underlying distribution. It consists of essentially two steps: prediction and update. We implement our tracking algorithm within the particle filter framework [7], [8].

Let $\alpha_t$be the state variable at time *t*, which is used to characterize the state of object, such as position, size, speed, and shape. Let $Z_t$be the observation at time *t*, while $Z_{1:t}$be all observations up to time *t*, namely,$Z_{1:t} = \{Z_1, Z_2, \ldots\ldots, Z_t\}$. Prediction is given by [9]

$$p(\alpha_t|Z_{1:t}) = \int p(\alpha_t|\alpha_{t-1}) p(\alpha_{t-1}|Z_{1:t-1}) d\alpha_{t-1} \quad (13)$$

And update

$$p(\alpha_t|Z_{1:t}) = \frac{p(Z_t|\alpha_t)p(\alpha_t|Z_{1:t-1})}{p(Z_t|Z_{1:t-1})} \quad (14)$$

In particle filter, $p(\alpha_t|Z_{1:t})$ is approximated by a set of N particles$\{\alpha_t^i\}_{i=1}^{N}$ with importance weights $\{w_t\}_{i=1}^{N}$, namely[1]

$$p(\alpha_t|Z_{1:t}) = \sum_{i=1}^{N} \delta(\alpha_t - \alpha_t^i) w_t^i$$

where$\delta(\cdot)$is the Dirac function. When using prior distribution as the importance sampling function, the weight of the *i*th particle is iteratively updated by [9]

$$w_t^i = w_{t-1}^i p(Z_t|\alpha_t^i) \quad (15)$$

wherew_t^i is the weight of the *i*th particle at previous time *t* -1 and$p(Z_t|\alpha_t^i)$is the observation likelihood of the *i*th particle. The weights $\{w_t^i\}_{i=1}^{N}$are normalized.

The classification process is governed by the observation model$p(Z_t|\alpha_t^i)$, where we estimate the likelihood of α_t^i observing Z_t and the dynamical model between the two states$p(\alpha_t|\alpha_{t-1})$.

A. Dynamical model

An affine image warping method is adopted for modeling the object motion in image sequence under the particle filter framework. Where this method consists of six parameters, they are $\{a_{11}, a_{12}, a_{21}, a_{22}\}$ are deformation parameters and $\{t_x, t_y\}$ are translation parameters. The candidate target region in the image is normalized to same size as object template via the object state in classification phase. Here α is the state variable which is represented as a vector above six parameters.

The dynamical model$p(\alpha_t|\alpha_{t-1})$ is modeled by a Gaussian distribution with the assumption that the six parameters are independent of each other, that is [9]

$$p(\alpha_t|\alpha_{t-1}) = \mathcal{N}(\alpha_t; \alpha_{t-1}, \sum) \quad (16)$$

where$\mathcal{N}(\cdot)$ is a Gaussian distribution with the mean α_{t-1} and the covariance $\sum$. The covariance matrix $\sum$ is a diagonal covariance matrix whose elements are the corresponding variances of affine parameters.

B. Observation model

The core of particle filter-based classification depends on the observation likelihood model $p(Z_t|\alpha_t^i)$. In actual fact, the SR based tracking method in [10] just improves it by using the SR. In this method [9], for the ith candidate object y, we get the optimized best-weighted kernel then KCD algorithm to obtain the coding vector β. Then the observation likelihood $p(Z_t|\alpha_t^i)$ is calculated as the residual [9]

$$p(Z_t|\alpha_t^i) = \exp\left(-\left\|\sum_{j=1}^{n}\beta c(x_j) - \varphi(y)\right\|_2^2 / \sigma^2\right) \quad (17)$$

Where σ is a constant.

In order to compute (14), we need to specify kernels, which correspond to the selected features. In this paper [9], spatial color histogram and spatial gradient histogram are adopted to represent the object. The color feature has gained more attention as it is a powerful alternative to characterize the appearance of object, especially if it can achieve robustness against deformation and partial occlusion.

For the ith candidate, we first obtain a rectangular region via its state parameter α_t^i. Then, rectangular region is divided into four subregions where in each subregion, we calculate a color subhistogram. The spatial color histogramh^c is obtained by connecting four subhistograms together. For the gradient feature, we first obtain image gradients by performing two filtering operations with kernels $[-1, 0, 1]$and$[-1, 0, 1]^T$. As with the spatial color histogram, we can obtain for subhistograms of oriented gradients [11]. The spatial gradient histogram h^g is obtained by connecting four subhistograms together. Gradient computation is clearly explained in the next section.

In classification algorithm, the classified portion state is determined by the particle having the maximum weight. The initial position of the object manually selected from the first frame and it is used as the training samples. The remaining templates are generated by perturbing a few pixel around corner points of the first template. Here balancing Classifying efficiency and computational complexity sets template number. The templates are updated by replacing a template that has the smallest representation coefficient of the current classified object for the changing target appearance.This will look as a continuous learning process as it is classifying. The particles are updated by (18).The state $\alpha_t^i = [a_{11}\, a_{12} a_{21}\, a_{22}\, t_x t_y]_t^1$ for ith particle at current time t is calculated as

$$\alpha_t^i = [a_{11}a_{12}a_{21}a_{22}t_x t_y]_t^1 = [a_{11}a_{12}a_{21}a_{22}t_x t_y]_{t-1}^1 + \varepsilon \quad (18)$$

Bhattacharyya coefficient

Based on the fact that the probability of classification error is directly related to the similarity of the two distributions, the choice of the similarity measure in [12] was such that it was supposed to maximize the Bayes error arising from the comparison of target and candidate. Bhattacharyya coefficient was chosen and its maximum searched for to estimate the target localization.

The Bhattacharyya coefficient is an approximate measurement of the amount of overlap between two statistical samples. The coefficient can be used to determine the relative similarity of the two samples being considered. Calculating the Bhattacharyya coefficient involves a rudimentary form of integration of the overlap of the two samples. The interval of the values of the two samples is split into a chosen number of partitions, and the number of members of each sample in each partition is used in the following formula,

$$Bhattacharyya = \sum_{i=1}^{n}\sqrt{\left(\sum a_i \cdot \sum b_i\right)}$$

where considering the samples a and b, n is the number of partitions, $\sum a_i$and$\sum b_i$, are the number of members of samples a and b in the i'th partition.

This formula hence is larger with each partition that has members from both sample, and larger with each partition that has a large overlap of the two sample's members within it. The choice of number of partitions depends on the number of members in each sample; too few partitions will lose accuracy by overestimating the overlap region, and too many partitions will lose accuracy by creating individual partitions with no members despite being in a surroundingly populated sample space.

The Bhattacharyya coefficient will be 0 if there is no overlap at all due to the multiplication by zero in every partition. This means the distance between fully separated samples will not be exposed by this coefficient alone.

Let $\mathbf{K_c}$ and $\mathbf{K_g}$be kernel matrices obtained based on the color and gradient features. Each value in kernel matrices $\mathbf{K_c}$ or$\mathbf{K_g}$is denoted as the similarity between two histograms. For example, each value in kernel matrix $\mathbf{K_c}$ is calculated by

$$\mathbf{K_c}(i,j) = \mathrm{BhaCoff}(h_c^i, h_c^j) \quad (19)$$

where$h_c^i$and $h_c^j$are two spatial-color histograms, and BhaCoff(.) is the coefficient function. The calculations of kernel vectors $\mathbf{K_c}(.,y)$and $\mathbf{K_g}(.,y)$are the same with the calculation of kernel matrices $\mathbf{K_c}$ and $\mathbf{K_g}$.

Proposed algorithm

Data: The kernel matrix K and the kernel vector K(·, **y**).
Result: The Classified class
Step 1: A training image is chosen from the Indian pines database.
Step 2: The portion of the image, which defines the classes from the original Indian Pines image by referring the ground truth image, is chosen according to how many classes are considered for classification.
The classes are considered from 5 to 10 numbers.
Step 3: Affine Transform of these portions of the images are derived which would provide an idea of ten different directional changes of the image portions defining classes.
Step 4: These image portions are split in to four equal parts. It is carried out applying image cropping. The image portions that are taken as the sample

for classification are split in to four equal parts along the column.

Step 5: The kernels used for fusion comprising Histogram of Gradient and the Color Histogram is found for each of these parts.

Step 6: These two kernels vectors (kernel vectors are the array of values depicting the values of the Histogram of Gradient and Color Histogram) are merged for the four portions in two separate vectors for each feature space.

Step 7: Bhattacharya coefficient is a measure of similarity between two vectors .In this implementation the color Histogram and the Histogram of Gradient is considered as the vectors that has to be compared.

Step 7: These merged vectors containing the kernel values found with the Bhattacharya coefficients between the color histogram vectors of the different templates portions and the histogram of gradient vectors of the different image portions are calculated.

Step8: These values of the coefficients are found with the interrelation to develop the K matrix
Initialize K and K(., **y**) by (9) and (10) ;

Step9: Initialize N particles $\{\alpha_1^i\}_{i=1}^N$ with the template *t* = 1;

Step 10: Initialize the weights of N particles $\left\{w_1^i = \frac{1}{N}\right\}_{i=1}^N$;

Step 11: for *t* = 2 tonumber of remote images

Step 12: for*i*= 0 to N do

Step 13: Predicting the state of the *i*th particle α_t^i by (17)

Step 14: Calculate *β* by Kernel Coordinate Descent
Initialize the coding vector $\beta_{init} = (K + \gamma I)^{-1} K(\cdot, y)$

Step 15: for *k* = 0 to maxIter
do

Step 16: for *j* = 0 to *n*
do

Step 17: Calculate *e*(**x***j***)** by (11);

Step 18: Update *βj by* (12);

Step 19: end

Step 20: end

Step 21: Calculate the observation likelihood by (17);

Step 22: Update the weight of the *i*th particle by (15);

Step 23: end

Step 24: Compute the index of maximum weight, i.e., maxI= max ($\left\{w_t^i = \frac{1}{N}\right\}_{i=1}^N$);

Step 25: Obtaining the classified template in the test image$\alpha_t = \alpha_t^{maxI}$;

Step 26: Iteration is continued until all the portions of the image are covered.

Step 27: Display the portion of the image that gives the maximum weight, which would give us the classified class.

RESULTS AND DISCUSSIONS

The Indian pines database images are taken in to consideration, which is hyperspectral in nature .In the database we have around 220 images .The first image or a randomly chosen image is learned sparsely by means of the proposed algorithm. The templates are chosen from the original image by comparing with the ground truth image. The ground truth image depicts the actual class present in the original image by means of different colors, each color depicting one class. As the object is chosen from the image as the template for learning this method comes under object-based classification. The following figures depict the different class extraction from the original image by taking ground truth as the reference image. These portions of the images are taken as the object or the template that would be trained sparsely. The Indian pines database images are resized to a new size and stored as the query image and the above procedure is applied on the query image using the template-matching algorithm that is discussed. The illumination change is taken care by the algorithm as the kernel and sparse representation techniques that are used are illumination independent. The implementation is carried out with two different situations that are developing the algorithm for database images with higher and lower illumination.

As implementation currently carried out is the object based image classification the whole process is working as the template-matching algorithm. The initial templates are taken and the templates are matched on different portions of the images from the start. During the illumination changes the algorithm is not needed to be changed. Thus the performance analysis of the proposed algorithm with the three considered situation is carried out. The overall accuracy, kappa coefficient and the execution time for different situations are calculated and tabulated. The system requirement of the implementation is as given below,

RAM :4Gb,64 bit OS,80GB hard disk, Matlab 2011b.

The overall accuracy in this implementation can be done by the use of position error of the matched template as this is a template matching method. The kappa coefficient is calculated from this error value in term of percentage for each class. The position error for each classes are calculated and the results obtained are given as below. The overall accuracy of the classification is calculated by knowing the number of pixels in the object which is classified correctly and that which is classified wrongly. The overall accuracy for each of the classes is calculated and the results are tabulated. By the use of overall accuracy the Kappa coefficients are calculated and the execution time for the current system requirement and the Matlab version is also tabulated.

The overall accuracy of classification from the implementation is calculated from the following formula,

$$OA = \frac{\sum_{i=1}^{k} n_{ii}}{|T|} \quad (20)$$

Where, is the number of pixels that the current class is correctly traced by the template matching and T is the total number of pixels that the current class is using in the database image? And k is the number of isolated places where the current class is available on the database image. The kappa coefficient is the measure of the classifier performance from the position error. The data set used in this implementation is a small segment of an Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) from the agriculture area in Indiana. There are 220 spectral channels with each spectral channel the image comprising 145 × 145 pixels. There are 220 spectral channels (spaced at about 10 nm) acquired in the 0.4–2.5 µm region.

The images from the dataset are chosen from the user and the templates are created from those images. The original Indian Pines image is as shown in Figure-1.

Figure-1. Indian pines original image.

The ground truth image provided for the corresponding original image dataset for the Indian Pines dataset is as shown in Figure-2.

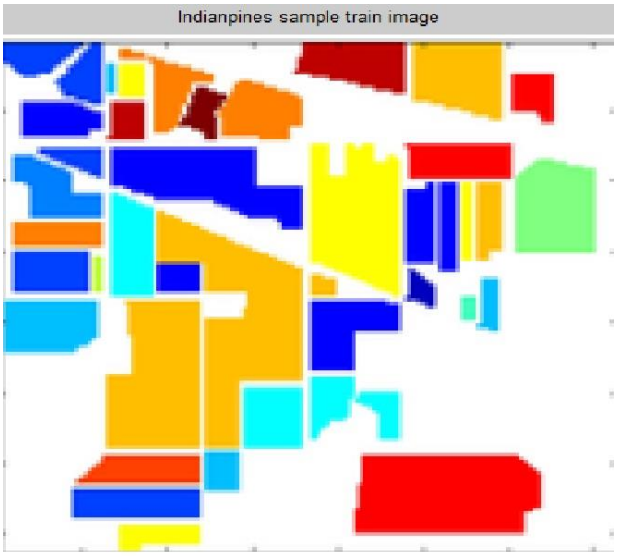

Figure-2. Ground truth image for training.

The original image with low illumination is given in Figure-3. The classified portions of the original image with 10 classes classified are shown in Figure-4. The classified portions of the original image with low illumination having 10 classes classified are shown in Figure-4. Figure-7 depicts the original image with 5 classes classified. The lower illumination original image with 5 classes classified.

Figure-3. Indian pines original image with low illumination.

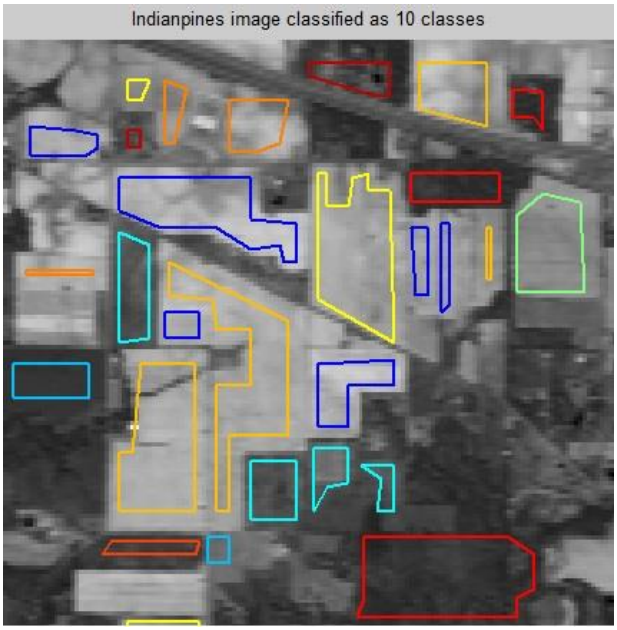

Figure-4. Indian pines original image with 10 classes classified.

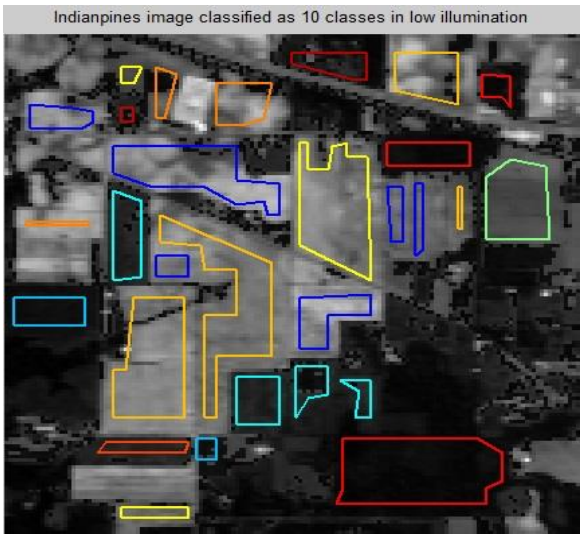

Figure-5. Indian pines original image with low illumination with 10 classes classified.

The results thus obtained from the proposed algorithm infer that the algorithm is highly robust as it could be observed that the images with lower illumination also will be able to provide a very good performance while classifying.

Figure-6. Salinasoriginal image.

The classified area on the images is approximate because the shape of the classified space is always rectangular. Thus in order to improve this a contour based segmentation algorithm can be implemented. As the multi kernel based method are implemented along with the sparse learning method the overall accuracy and the kappa coefficient, which are calculated in percentage, are in average larger than the SVM based method implemented in [16].Thus byincluding further kernels the accuracy can be improved.

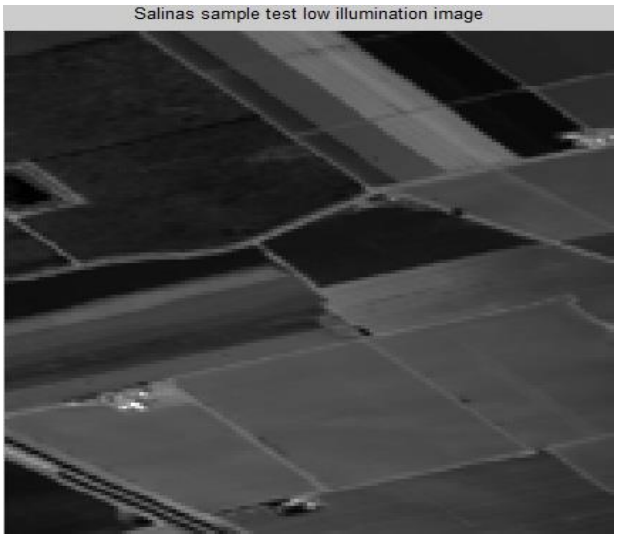

Figure-7. Salinas lower illumination original image.

The calculation of the kappa coefficient is as follows:

The classifier places the 826 pixels out of 21025 pixels in the class1; the reference from the image has 903 pixels out of 21025 pixels belonging to class1.kappa coefficient is defined as follows,

k=probability of correct classification-probability of chance agreement/(1-probability of chance agreement)

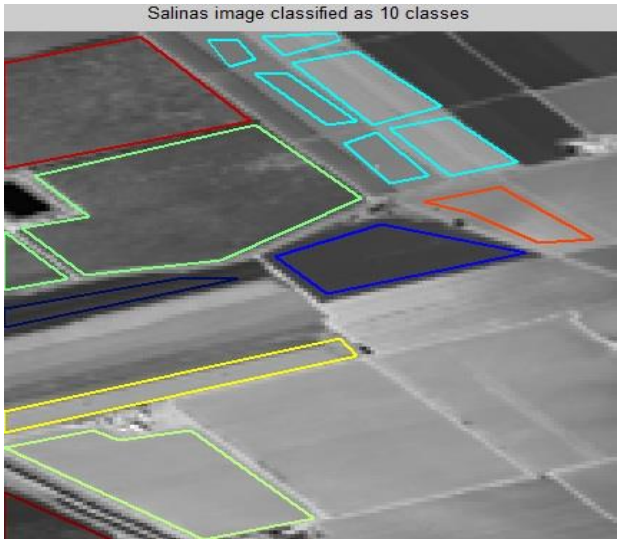

Figure-7. Salinasoriginal image with 10 classes classified.

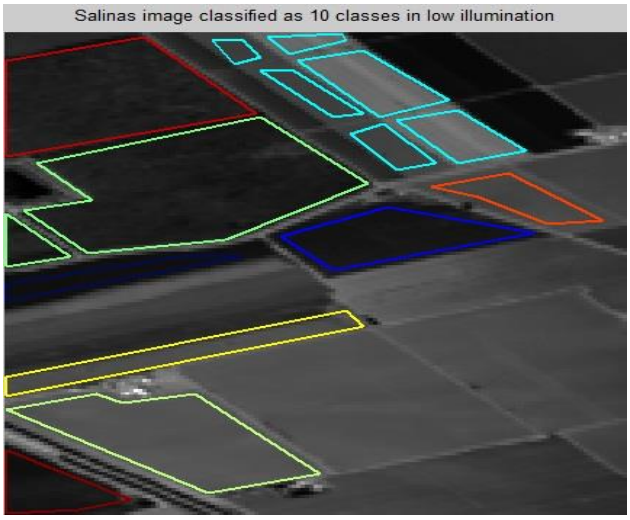

Figure-8. Salinaslow illumination with 10 classes classified.

Table-1. Classification efficiency parameter for two different situation of the database image.

	Indian Pines image			Salinas image		
Class number	**Overall accuracy in %**	**Kappa coefficient in %**	**Execution time in secs**	**Overall accuracy in %**	**Kappa coefficient in %**	**Execution time in secs**
Class1	82.214765	84.859533	3.119756	82.161235	83.920210	6.086553
Class2	92.397043	92.736826	4.449269	86.563307	87.841194	3.517823
Class3	92.134831	92.621535	4.340104	74.285714	79.521927	3.339309
Class4	87.514188	88.632689	7.244424	87.726358	89.016169	2.873072
Class5	95.815603	95.726131	5.677614	92.951542	93.409137	2.725683
Class6	74.047619	79.284329	4.878422	75.609756	80.369418	4.816973
Class7	78.977273	82.612784	2.798653	79.588015	82.955178	5.286839
Class8	79.816514	83.181882	3.873842	79.934211	83.017300	3.465873
Class9	86.363636	87.993659	3.015000	88.435374	89.595415	3.401889
Class10	88.607595	89.770792	1.817707	80.813953	83.893318	3.294631

CONCLUSIONS

A Multikernel Sparse Representation based image classification implementation was implemented and the results were tabulated. The results inferred that the accuracy of the MKSR based implementation are on an average better than the SVM implementations and on further improvement of the kernel features a very robust method with higher accuracy can be derived.Because of the robust nature of the algorithm used the implementation on images with lesser illumination also worked better as the HOG and CH both the kernel space were fused. The template matching if done using variable size template could improve the accuracy still further.

As a future enhancement, during the tilt and the size change the templates are tilted by the use of affine transform and during the size change the global location is found and then changing the size of the template carries out the local template matching.The image which are resized from the actual size in the database and the images that are tilted from the actual angle and also the images with different illumination are considered in order to find the accuracy and robustness of the algorithm thus used.

REFERENCES

[1] ShivakumarG.S,Dr.S.Natarajan and Dr.K.Srikanta Murthy. 2015. Sparse Representation in Active Learning Methods for Remote Sensing Image Classification -A Survey. International Journal of Applied Engineering Research (IJAER). 10(3): 6419-6430

[2] V. Vapnik. 1995. The natural of statistical learning theory. Springer, New York, USA.

[3] B. Scholkopf and A. Smola. 2002. Learning with kernels. MIT Press, Cabridge, MA.

[4] T. Mitchell. Machine Learning. McGraw-Hill, 1997.Shawe-Taylor, John and NelloCristianini. Kernel Methods for Pattern Analysis. Cambridge, UK: Cambridge University Press. 2004.

[5] Scholkopf Bernhard and Alexander J. Smola. 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. Cambridge, MA: The MIT Press.

[6] M. Isard and A. Blake. 1998. Condensation: Conditional density propagation for visual tracking.Int. J. Comput. Vision. 29(1): 5-28.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking.IEEE Trans. Signal Process. 50(2): 174-188.

[8] Lingfeng Wang, Hongping Yan, KeLv and Chunhong Pan. 2014. Visual Tracking via Kernel Sparse Representation with Multikernel Fusion.IEEE transactions on circuits and systems for video technology. 24(7).

[9] X. Mei and H. Ling. 2011. Robust Visual Tracking and Vehicle Classification via Sparse Representation. IEEETrans. Pattern Anal. Mach. Intell. 33(11): 2259-2272.

[10] Scholkopf Bernhard and Alexander J. Smola. 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. Cambridge, MA: The MIT Press.

[11] Comaniciu D.; Ramesh V.; Meer P. 2003. Kernel-based object tracking.Pattern Analysis and Machine Intelligence, IEEE Transactions on. 25(5): 564-577.

[12] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[13] http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes.

[14] Lingfeng Wang, Hongping Yan, KeLv, and Chunhong Pan. 2001. Visual Tracking via Kernel Sparse Representation withMultikernel Fusion. IEEE Transactions on Circuits and Systems for Video Technology. 24(7).

[15] Alberto Villa, Jon AtliBenediktsson, Jocelyn Chanussot, Christian Jutten. 2011. Hyperspectral Image Classification with Independent Component Discriminant Analysis. IEEE Transactions on Geoscience and Remote Sensing.

[16] Chu He *et al.* 2015. Particle Filter Sample Texton Feature for SAR Image Classification. IEEE Geoscience and Remote Sensing Letters. 12(5).

[17] Xue Mei Et Al. 2011. Robust Visual Tracking and Vehicle Classification via Sparse Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 33(11).