Performance analysis on Image Super-resolution Using Kernel Resolution Synthesis Based on SVR

Jacob Thomas, Dr. S. Mohan, Jesna Anver
Department of Information Technology, Karunya University, Coimbatore, India, Department of computer science and Engineering, Saintgits College of Engineering, M G University, Kerala, India
jacobmarythomas@gmail.com
jesna_anver@yahoo.com
Professor and Head, Department of computer science and Engineering, Dr. N G P Institute of Technology, Anna University, Coimbatore, India
s.mohan77@gmail.com

***Abstract*—Image super-resolution is the process by which additional information is incorporated to enhance a low resolution image thereby producing a high resolution image. In the simplest case super-resolution of a single image is a process of obtaining high-resolution image with more number of pixels with more resolving power. Therefore the super-resolved image should demonstrate an improvement in the perceived detail content compared to that of the low-resolution images. This will typically involve restoration of the high-frequency content. Along with the original information inherent within the low-resolution image, the additional information may come in several forms: a group of several shifted versions of the low resolution image, a collection of optimally estimated filters selected for specific image content i.e., a relationship representing a training set that contains low and high resolution image pairs. The common image interpolation functions, like the cubic spline for example, often gives blur edges and image details, and that such analytical methods cannot reproduce details in textured regions well. The new training-based super-resolution approach using the support vector regression, which maintains all the main features that characterize the maximal margin algorithm used for classification to regression analysis, such as duality, sparseness, kernel and convexity, produces better results.**

***Keywords*— Interpolation, Training set, Super resolution, SVR.**

I. INTRODUCTION

This paper should first serve a self-contained introduction to Support Vector regression and it attempts to give an overview of recent developments in the field of super resolution. The Support vector algorithm is a nonlinear generalization of the Generalized Portrait algorithm developed in Russia in the sixties. The Support vector machine (SVM) was largely developed at AT&T Bell Laboratories by Vapnik and co-workers. Support Vector Machines are a new class of learning machines. A learning machine here is represented by a family of functions f(x, a) where x represents input and a represents adjustable parameters. The basic idea of SVM is to use linear model to implement nonlinear class boundaries through some nonlinear mapping of the input vector into the high dimensional feature space. A linear model constructed in the new space can represent a nonlinear decision boundary in the original space. In the new space, an optimal separating hyper-plane is constructed. Thus SVM is known as the algorithm that finds a special kind of linear model, the maximum margin hyper-plane. The maximum margin hyper-plane gives the maximum separation between the decision classes. The training examples that are closest to the maximum margin hyper-plane are called support vectors.

II. SUPPORT VECTOR MACHINES : SVM

Classifying data is a common need in machine learning. It is given data points each of which belongs to one of two classes; it is possible to determine which class a new data point will be in. In the case of support vector machines, view a data point as a dimensional vector (a list of p numbers), and to know whether it is possible to separate them with a $p - 1$-dimensional hyper plane. This is called a linear classifier. Then achieve maximum separation (margin) between the two classes. The hyper is such a way that the distance from the hyper plane to the nearest data point is maximized. That is to say that the nearest distance between a point in one separated hyper plane and a point in the other separated hyper plane is maximized. Such a hyper plane is called as the maximum-margin hyper plane and such a linear classifier is known as a maximum margin classifier. An SVM will construct a separating hyper plane which maximizes the "margin" between the two data sets. Data points of different classes separated by a hyper plane and the two bounding planes passing through support vectors are shown in fig.1.

To calculate the margin, first construct two parallel hyper planes, one on each side of the separating one, which are "pushed up against" the two data sets. A good separation is achieved by the hyper plane that has the largest distance to the neighboring data points of both classes i.e., the larger the margin or distance between these parallel hyper planes, the

better the generalization error of the classifier will be. This hyperplane can be expressed as:

$$Y=W_0+W_1X_1+W_2X_2+W_3X_3 \quad (1)$$

Where Y is the outcome Xi are the attribute values, and there are four weights 'Wi' to be learned by the learning algorithm. In the above equation, the weights 'Wi' are parameters that determine the hyper-plane. The maximum margin hyper-plane can be represented as the following equation in terms of the support vectors:

$$y=b+\text{sigma } (atytK(x(t).x)) \quad (2)$$

The function K(x(i).x) is defined as the kernel function. There are different kernels for generating the inner products to construct machines with different types of nonlinear decision surfaces in the input space.

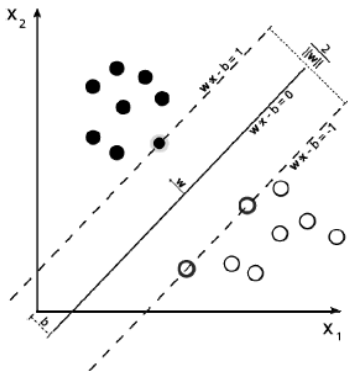


Fig. 1. Maximum-margin hyper plane and margins for a SVM trained with samples from two classes. Samples on the margin are called the support vectors.

In many applications, some input points may not be exactly assigned to one of above said two classes. Some are more important to be fully assigned to one class so that SVM can separate these points more correctly. Some data points corrupted by noises are less meaningful and the machine should better to discard them. SVM lacks this kind of ability. Traditional neural network approaches have suffered difficulties with generalization, producing models that can over fit the data. SVMs were developed to solve the classification problem, but recently they have been extended to the domain of regression problems. In the literature the terminology for SVMs can be slightly confusing. In this paper the term SVM will refer to both classification and regression methods, and the terms Support Vector Classification (SVC) and Support Vector Regression (SVR) will be used for specification.

III. SUPPORT VECTOR REGRESSION

The Support Vector method can be applied to regression, which maintains all the main features that characterize the maximal margin algorithm used for classification to regression analysis, such as duality, sparseness, kernel and convexity. Support vector machines have been concerned with classification, or about the problem of dividing the elements into different types. After the realization of SVM's which are used to classify, other machines where devised to resolve the problem of regression, where the solution generated is a real number.

The SVR uses the same principles as the SVM for classification, in the case of regression, a margin of tolerance ε is set in approximation to the SVM which would have already requested from the problem. Support vector regression is used to provide a relationship between known (low-resolution) and unknown (high-resolution) information by using a function model with estimated parameters. The support vector machines applied to regression helps to learn the relationship with the use of functional rather than a single function. SVM is mainly used for classification and a Hyper plane is used to get maximum separation between classes. While doing so, the SVM algorithm tries to achieve maximum separation between the classes as shown in fig. 2.

Separating the classes with a large margin minimizes a bound on the expected generalization error. A 'minimum generalization error', means that when new examples (data points with unknown class values) arrive for classification, the chance of making an error in the prediction (of the class which it belongs) based on the learned classifier (hyper plane) should be minimum. Intuitively, such a classifier is one which achieves maximum separation-margin between the classes. Fig.2. given below illustrates the concept of 'maximum margin'. The two planes parallel to the classifier and which pass through one or more points in the data set are called 'bounding planes'. The distance between these bounding planes is called the 'margin' and SVM 'learning', means, finding a central hyper plane which maximizes this margin.

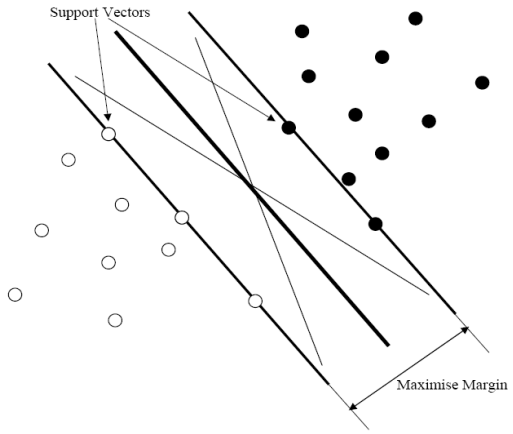


Fig. 2. Hyper plane with support vectors

There may be another situation wherein the points are clustered such that the two classes are not linearly separable, that is, if one tries for a linear classifier, it may have to tolerate a large training error. In such cases, one prefers non-linear mapping of data into some higher dimensional space called 'feature space', F, where it is linearly separable.

IV. IMAGE SUPER-RESOLUTION USING KERNEL RESOLUTION SYNTHESIS

Classification-based super-resolution algorithms have been shown to provide good generalization. In these works, rather than a static filter for all possible image content, content is classified (i.e., edges, texture, smooth portions, etc.), and then the appropriate filters are applied accordingly. Hence, another means of simplification is to partition the domain by data similarity to reduce problem complexity i.e., by partitioning the problem into smaller ones, the individual problems are more easily approximated than the original one. The kernel resolution synthesis is an extension to this concept by using the resolution synthesis framework. The idea is to segment image content with classification and then provide function estimation via SVR on individual segments.

A) Kernel Resolution Synthesis

The work in [1] uses a stochastic approach to super-resolution by determining the conditional expectation of high-resolution pixels given low-resolution $D\times D$ patches and training data. That is, when $y=f(x)$, where x and y are defined, and $f(x)$ is approximated.

$$g(x)=E[y\,|\,x,\Omega]. \tag{3}$$

The $g(x)$ equates the idea of an all-encompassing function to solving several smaller functions $g_j(x)$ that sharpen, smooth, etc., depending on when it is necessary to do so. By dividing the domain into several classes and based on the classes it can be decided to use which $g_j(x)$. In [1], classes are divided by image content whose distribution is modeled as a Gaussian mixture. Thus, the expression for the expected value for classified training data is the weighted average of possible reconstructed values based on various image features. Denoting the random variable J as the class number of input x, then the expectation of high resolution image given the low resolution image is given by,

$$E[y\,|\,x]=\sum_j E[y\,|\,x,J=j]\,P(J=j\,|\,x) \tag{4}$$

Where $$P(J=j\,|\,x)=\frac{P(x\,|\,J=j)\,P(J=j)}{P(x)}$$

$$=\frac{P(x\,|\,J=j)\,P(J=j)}{\sum_j P(x\,|\,J=j)\,P(J=j)} \tag{5}$$

and the likelihoods $P(x\,|\,J=j)$ are determined by expectation maximization. By approximating the class conditional expectation with a, regression device, i.e., $E[y\,|\,x,J=j]=g_j(x)$, the solution becomes,

$$E[y\,|\,x]=\sum_j g_j(x)\,P(J=j\,|\,x) \tag{6}$$

B) The Expectation maximization Algorithm

In statistical methods, pixel labels are described according to the probability value, which are determined based on the intensity distribution of the image. Statistical methods are mainly divided in to two categories “Parametric” and “Non-Parametric”. In non-parametric methods, no prior assumption is made about the functional form of the distribution but a large number of correctly labeled training points are required in advance. In parametric method, approaches rely on an explicit functional form of the intensity density function. In case of images the functional form is represented by the distribution of pixel values in a specific form. In probability model, one of the techniques is model-based clustering; this method assumes that a finite mixture of underlying probability distributions generated the data. Among all distribution Gaussian is best one as many image histograms follow this probability distribution. So the Gaussian mixture model is a powerful tool for many applications. In addition, selecting a “good” clustering method and determining the “correct” number of clusters are reduced to model selection problems in the probability framework. Multi-band image segmentation method is proposed on the basis of the maximum likelihood (ML) estimation for the clustering problem.

$$P_{i,k}=\frac{a_k\,\phi(x_i\,|\,\mu_k^p,\sigma_k^{2^p})}{\sum_{k=1}^{K} a_k^p\,\phi(x_i\,|\,\mu_k^p,\sigma_k^{2^p})} \tag{7}$$

In order to calculate the above expression, prior probabilities are required which are assumed as arbitrary values a_k and x represents the pixel intensity. Mathematically, $P_{K,i}$ (posterior probability) represents: (Prior probability * likelihood) / total probability. Each Gaussian function has mean and variance which are represented with μ and σ^2. K represents the required number of clusters for an image. The posterior probability $P_{K,i}$ has to be maximized, for which many methods are available one among them is EM algorithm which takes less time than other methods like gradient method. After maximization the log likelihood has to be found. This log likelihood gives a measure of similarity between the expected values and true values of Gaussian

distribution. The log likelihood, $L(x|\theta)$, is obtained by applying log to posterior probability function.

$$L(x|\theta) = \log\left(P_{i,k}\right)$$

$$L\left(x|\theta\right) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} a_k \phi\left(x_i|\mu_k,\sigma_k^2\right)\right) \quad (8)$$

By using Maximum Likelihood from equation (8), estimate the best suitable model from the mixture of models. Hence EM algorithm can do this. EM algorithm has two steps, E step and M step. E step specifies the estimation of the posterior probabilities P (Observation, Clustercenters). M-step maximizes the cluster center likelihood. There are two cases where EM algorithm is indispensable. The first case is when the data indeed has missing values, due to the limitations of the observation process. The second case is for optimizing the parameters of Gaussian distribution. EM is more commonly used in the computational pattern recognition. The function $L(\theta|\theta_n)$ is upper-bounded by the likelihood function $L(\theta)$. The functions are equal at $\theta = \theta_n$. The EM algorithm chooses θ_{n+1} as the value of θ for which $L(\theta|\theta_n)$ is a maximum**.**

C) *EM algorithm*

1. Initialize mean, variance and prior probability

2. **E-step:** Compute the posterior probabilities for all *i=1.....n, k=1....K*

Where $$\phi\left(x|\mu_k,\sigma^2{}_k\right) = \frac{1}{\sqrt{2\pi\sigma^2{}_k}} - \exp\left(\frac{-(x-\mu_k)}{2\sigma^2{}_k}\right) \quad (9)$$

μ and σ^2 are mean and variance. K represents the number of clusters in the image. $\phi(x|\mu_k,\sigma^2{}_k)$ is the posterior probability function. $\theta = \{\theta_1,\theta_2,.....,\theta_k\}$ Contain mean, variance and probability of each Gaussian. $a = \{a_1,a_2,.....,a_n\}$ is a vector of mixture probabilities such that $a_k \geq 0$ for k=1…..K. The sum of all the probabilities is equal to 1. $\sum_{k=1}^{K} a_k = 1$

3**. M-step**: maximize the mean, variance and prior probabilities of Gaussian model.

$$a_k^{(p+1)} = \frac{\sum_{i=1}^{n} P_{i,k}}{n},\ \mu_k^{(p+1)} = \frac{\sum_{i=1}^{n} P_{i,k} x_i}{\sum_{i=1}^{n} P_{i,k}},$$

$$\sigma^2{}_k^{(p+1)} = \frac{\sum_{i=1}^{n} P_{i,k}\left(x_i - \mu_k^{(p+1)}\right)\left(x_i - \mu_k^{(p+1)}\right)}{\sum_{i=1}^{n} P_{i,k}}. \quad (10)$$

4. Repeat step 2 and step 3 until it converges. The EM algorithm flow chart is shown in fig.3. The parameters are estimated by Expectation-Maximization (EM) algorithm [2], [3]. This is a procedure for iteratively maximizing likelihoods in situations where there are unobserved quantities and estimation would be simple if these were known.

D) *EM Algorithm Flow chart*

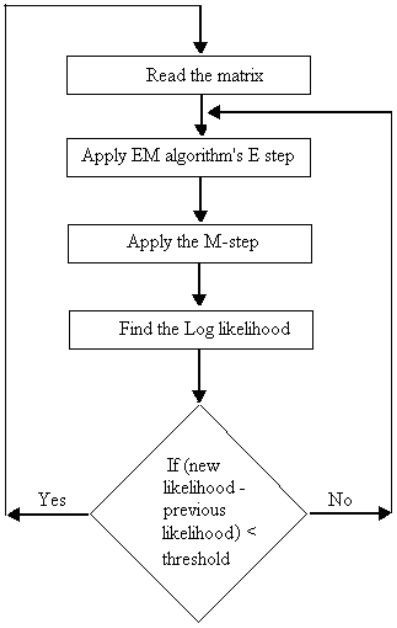


Fig.3. The EM algorithm flow chart

In [3], the regression for g_j is done linearly by $g_j = A_j x + b_j$ .In [10] it is said to use different SVR regressors for each segment (g_j), which is not a simple substitution in the implementation because it is required to alter the optimization problem slightly. In resolution synthesis after forming the Gaussian PDF, each A_j and b_j parameter is generated from the each point in the data set by weighting the point contribution by its posterior probability. In kernel

resolution synthesis, this is not so easily done, as the parameters in kernel resolution synthesis arises.

$$\min_{w_j, b_j, \xi^{+/-}_{(i,j)}} \quad (\frac{1}{2}\|w_j\| + C\sum_{i=1}^{N}(\xi^{+}_{(i,j)} + \xi^{-}_{(i,j)}))$$

subject to

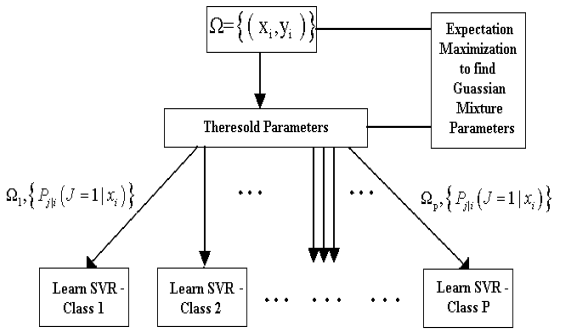
$$(\langle w_j, \phi(x_{(i,j)})\rangle + b_j) - y_{(i,j)} \le \in + \xi^{+}_{(i,j)}$$

$$y_{(i,j)} - (\langle w_j, \phi(x_{(i,j)})\rangle + b_j) \le \in + \xi^{-}_{(i,j)}$$

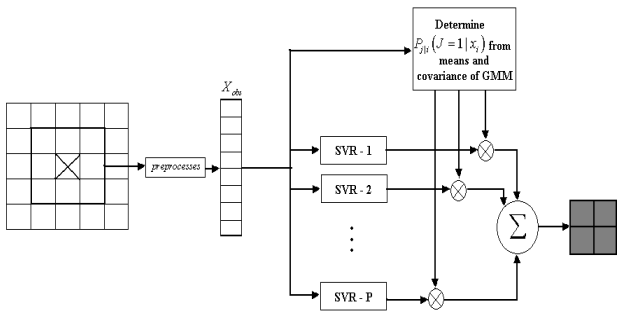
$$\xi^{-}_{(i,j)}, \xi^{+}_{(i,j)} \ge 0 \qquad (11)$$

Dual Problem

$$\max_{\alpha^+, \alpha^-} \quad -\frac{1}{2}\sum_{i,k}\left\{\left(\alpha^{+}_{(i,j)} - \alpha^{-}_{(i,j)}\right)\left(\alpha^{+}_{(k,j)} - \alpha^{-}_{(k,j)}\right)K_j(x_k, x_i)\right\}$$
$$-\in \sum_i\left(\alpha^{+}_{(i,j)} + \alpha^{-}_{(i,j)}\right) + \sum_i y_i\left(\alpha^{+}_{(i,j)} - \alpha^{-}_{(i,j)}\right)$$



(a)



(b)

Fig.4. Kernel resolution synthesis algorithm. a. Training algorithm; b. Testing algorithm.

To weight the training points by their importance is analogous to the effect of the C variable on the solution hyper-plane. The C variable is actually a cost parameter whose value comes out of cross -validation. In the dual problem, the larger the cost parameter, the more the $\alpha^{+/-}_{(i,j)}$ values can deviate for an exact regression, in effect granting freedom to closely fit the training data in exchange for flatness in the objective function. Therefore, for the pair (x_i, y_i) in Ω, if the limit is more then $\alpha^{+/-}_{(i,j)}$, less the effect of the i^{th} point on the solution hyper-plane. In terms of the primal problem, the value of C does this by scaling the slack variables $\xi^{-}_{(i,j)}$ and $\xi^{+}_{(i,j)}$, essentially restricting the quantity of points deviating from the solution hyper-plane and by how much these points deviate. In kernel resolution synthesis it scales each slack variable for every point in the training set by how much it is believed it to be relevant to the current class j. This can be done by pre-multiplying all $\xi^{-/+}_{(i,j)}$ with the corresponding posterior probability of that particular class, $P_{i,j}$, where j, again is the class, and i refers to the particular data point in the training set. So, for the j^{th} regression, the primal optimization problem is described by

$$\min_{w_j, b_j, \xi^{+/-}_j} \quad \frac{1}{2}\|w_j\|^2 + C.P_{j|i}(J = j \mid x_i)^T\left(\xi^{+}_j + \xi^{-}_j\right)$$

subject to

$$(\langle w_j, \phi(x_i)\rangle + b_j) - y_i \le \in + \xi^{+}_j$$

$$y_i - (\langle w_j, \phi(x_i)\rangle + b_j) \le \in + \xi^{-}_j$$

$$\xi^{-}_j, \xi^{+}_j \ge 0. \qquad (12)$$

where $P_{j|i}(J = j \mid x_i)$ denotes a vector containing the posterior probabilities of classes. Hence, more slack for the variables can be allowed that are less important (i.e., have smaller posterior probabilities). This solution has the potential to consume extensive computation both in CPU cycles and memory, and so a simplification of the problem would be to consider per class those points which meet a certain criterion with respect to their respective posterior probabilities. This has been implemented by partitioning Ω into $\{\Omega_j\}$ and considering the i^{th} point for class j only if its posterior probability exceeds a certain threshold, η_j. The final

algorithm is depicted in Fig .4, it shows both the training algorithm as well as the testing algorithm. The threshold in Fig.4 is cross validated. A user can choose the threshold in many ways. One dynamic threshold choice would be to include a point in class j if its posterior probability is the largest among all other classes. The result is an exclusive point to class pairing that, if equitably divided by EM, contains N/K points per class on average, where K is the number of classes. This type of classification is not generally suggested. However, in terms of evaluating savings in memory, it serves as a nice point of reference because the extent of the savings in static thresholding is comparable to exclusive thresholding and provides a good approximation of order.

V. Result Analysis

The output of the variable whose function is known is predicted, using non-linear SVM Regression and analysed the difference between the actual and learned values. The performance analysis based on images of different resolutions, using kernel resolution synthesis is shown in Table 1.

Table I: Analysis based on Kernel Resolution Synthesis.

Serial no:	Input image size	Output image size	Elapsed time in seconds
1	16×16	32×32	0.72
2	16×16	64×64	0.92
3	64×64	128×128	136.67
4	64×64	256×256	142.63
5	128×128	256×256	248.08
6	128×128	512×512	257.758

The graph corresponding to the Table I is shown in fig. 5.

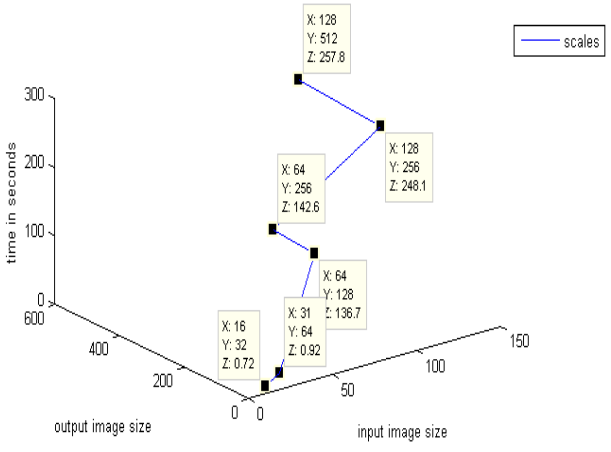


Fig.5. Graphical representation in 3D

VI. Conclusion

Support vector regression is the natural extension of large margin kernel methods, which is used for classification to regression analysis. It retains all the properties like duality, sparseness, kernel and convexity. The difference with support vector machine is that, support vector regression formulation introduces the concept of loss function which ignores errors occurring within a distance of the true value. The kernel resolution synthesis has more generalization .The kernel resolution synthesis algorithm is simulated in Matlab. Only the Gaussian Kernel is used here. All training and reconstructed images were selected from standard TIF and GIF files. Before training of the data, the image is classified using the expectation maximization algorithm. The number of classes assumed is 75. Many experiments were conducted to classify the image properly; at first the pixels were classified by considering to which class it has larger probability. Even though this method seems logically possible, the results obtained were not satisfactory. The second acceptable approach was that, to which all classes the pixels have a minimum threshold. This method allows a pixel to be in more than one class. In the learning part the high resolution samples are taken using the imresize function in Matlab.

References

[1] Steve R. Gunn, "Support Vector Machines for Classification and Regression", Faculty of engineering , science and Mathematics School of electronics and computer Science, 1998.

[2] K. I. Kim, M. Franz, and B. Scholkopf, "Kernel hebbian algorithm for single-frame super-resolution," Statist. Learn. Comput. Vis., pp.135–149, 2004.

[3] C. B. Atkins and C. Bouman, "Classification based methods in optimal image interpolation," Ph.D. thesis, Purdue Univ., West Lafayette, IN, 1998.

[4] Karl S. N, and Truong Q. Nguyen "Image Superresolution Using Support Vector Regression", IEEE Transaction on image processing, VOL. 16, NO. 6, JUNE 2007.

[5] Karl S. Ni, Sanjeev Kumar, Nuno Vasconcelos, Truong Q. Nguyen "Single image superresolution based on support vector regression", ECE Dept, UCSD, La Jolla, CA 92093-0407.

[6] K. R. Rao and P. Yip, Discrete Cosine Transforms: Algorithms, Advantages, Applications, Academic Press, Inc., 1990.

[7] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming", J. Mach. Learn. Res., vol. 5, pp. 27–72, 2004.

[8] S. Qiu and T. Lane, "Multiple kernel learning for support vector regression", Tech. Rep., Univ. New Mexico, Albuquerque, 2005.

[9] K. Ni, S. Kumar, and T. Q. Nguyen, "Learning the kernel matrix for superresolution", presented at the IEEE Conf. Multimedia Signal Processing, Oct. 2006.

[10] L. Tuncel, "On the Slater condition for the SDP relaxations of nonconvex sets", Oper. Res. Lett., vol. 29, pp. 181–186, 2001.

[11] F. M. Candocia and J. C. Principe, "Super-resolution of images based on local correlations", IEEE Trans. Neural Netw., vol. 10, no. 2, p. 372, Mar. 1999.

[12] A.Dasgupta ,A.E Rafter , "Detecting features in spatial point processes with clutter via model-based clustering", Journal of the American Statistical Association (1998).

[13] A. P. Dempster, N. M. Laird and D. B. Bubin, "Maximum Likelihood from Incomplete data via EM algorithm", J. Royal Statistical Society, ser. B, (1977).