Multifeature Based Visual Tracking for Surveillance

P.Kiruthika[a]* and R.Kalaiprasath[b]

[a]*MPhil Scholar, Selvamm Arts & Science College, Department of Computer Science, Namakkal, Tamilnadu, India*

[b]*Research Scholar, Department of CSE, Bharath University, Asst. Professor, Aksheyaa College of Engg., Chennai.*

Abstract

This project attempts a robust vehicle tracking technique considering all the problems that would otherwise disturb the vehicle tracking. Like the variation in the pose of the vehicle, change in illumination in the video, occlusion as of any material coming in between the line of focus and the camera. There were various methods that were proposed in the literature before but sparse representation has been a more superior method. In this paper we build up an algorithm to optimize the weight updation involved in multi-kernel fusion on visual tracking system. This makes this algorithm to be adaptive and optimized. Particle Swarm Optimization (PSO) algorithm is introduced in order populate the weight value and update it in every iteration in order to attain the objective function which will provide us the sparsest realization of the feature from the frame. The previous work had put light on the multi-kernel fusion alone but we improve that to be a multi-objective optimization using Particle Swarm Optimization algorithm. The thought generated must be tested with the software tools like Mat lab to validate the efficiency.

Index Terms*- Sparse representation, Kernel sparse representation, Particle swarm optimization, Multifeature fusion, Particle filter framework.*

I.INTRODUCTION

Visual tracking is an essential task within the field of computer vision. The abundance of high-end computers, the existence of high quality video cameras, and the ever-increasing need for automated video analysis have generated a great deal of interest in visual tracking algorithms. Normally , the use of visual tracking is pertinent in the tasks of movement-based recognition, surveillance, video indexing, human–computer interaction and vehicle navigation, etc.For a visual tracking to be helpful in real-world scenarios, it must be planned to handle and overcome cases where the target's appearance changes from one frame to another. Even though much improvement has been made in recent years, it is still a tricky problem to develop a robust algorithm for complex and dynamic scenes due to large appearance change caused by varying illumination, camera movement, occlusions, background clutter, pose variation and shape deformation .

Recently, sparse representation has been effectively applied to visual tracking [2, 3]. In this case, the trackerrepresents eachtarget candidate as a sparse linear combination of dictionary templates that can be dynamically updated to maintain an up-to-date target appearance model. This representation has been shown to be robust against partial occlusions, which leads to improved tracking accuracy. However, sparse coding based trackers perform computationally expensive l1 minimization at each frame. In a particle filter framework, computational cost grows sequentially with the number of particles sampled.

In [4], the multi-task tracking is proposed by Zhang, B. Ghanem, S. Liu, and N. Ahuja. In this method, same particle filter is used as multi-task sparse problem, here lp,q mixed norm is used which is used to regularize both the sparsity and particle representation. Before particles were delt independently but in this particles are interdependently used and mining that interdependence would improve the performance of the tracking. L1 tracker is also present here if p=q=1 is the case. Accelerated Proximal Gradient (APG) method yields the sequence of close forms updates. Heavy occlusion drawn illumination changes and large pose variation are solved here.

The traditional SR-based tracking method suffered from huge computational cost. In order to reduce this Bao et al. [5] adopted an accelerated proximal gradient approach. In this method there will be template created and sparse approximation is applied over a template set which lead to L1 tracker. To develop a robust and faster L1 tracker along with L1 norm minimization, L1 norm regularization was also implemented at trivial templates. The L1 norm minimization was accelerated using the accelerated proximal gradient approach.

---

* Corresponding author. Tel.: +91 9789702674.

E-mail address: vickysivam86@gmail.com

In order to overcome computational cost and tracking accuracy in visual tracking, kernel Sparse representation is proposed in [1] by Lingfeng Wang, Hongping Yan, KeLv, and Chunhong Pan. The Sparse Representation tracker does not use sophisticated features to represent tracking objects. So kernel sparse representation (KSR) is presented by introducing the kernel method into Sparse Representation. Additionally making use of multikernel fusion method multiplesophisticated object features are considered during the tracking process.

In this paper, we propose a novel weight optimization on multi-kernel fusion. Here, we develop an algorithm to optimize the weight updationinvolved in multi-kernel fusion on visual tracking system. This makes this algorithm to be adaptive and optimized. Particle Swarm Optimization (PSO) algorithm is introduced in order populate the weight value and update it in every iteration in order to attain the objective function which will provide us the sparsest realization of the feature from the frame. The previous work had put light on the multi-kernel fusion alone but we improve that to be a multiobjective optimization using Particle Swarm Optimization algorithm.

The main advantages of using Particle swarm algorithm are: Faster convergence, less parameters to tune, easier searching in very large problem spaces, easy to perform, efficient in global search.

II.SPARSE REPRESENTATION

Sparse representation is the problem of determining a sparse multi-dimensional vector that satisfies a linear system of equation given high-dimensional observed data and a design matrix. Sparse approximation techniques are widely used in applications such as image processing. In numerical analysis, a sparse matrix is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, then the matrix is considered dense. The fraction of zero elements (non-zero elements) in a matrix is called the sparsity (density).

Let **X** = $[x_1, x_2, \ldots\ldots, x_n] \in R^{d\times n}$ be *n* training samples, where *d* is the feature length. In SR, the testing sample **y** $\in R^{d\times 1}$ is sparsely represented on **X** via l_1 minimization [1]

$$\hat{\beta} = \min_{\beta} E(\beta) = \min_{\beta}\left(\frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1\right) \qquad (1)$$

whereλ is a slack variable that balances the reconstruction error and the sparseness of the coding vector β. The coding vector β is an $R^{n\times 1}$vector.

L1-minimization refers to finding the minimum L1-norm solution to an underdetermined linear system y=βx. Under certain conditions as described in compressive sensing theory, the minimum L1-norm solution is also the sparsest solution.

In the SR-based tracker [2], training samples are the templates obtained in the initial frame, and the testing sample is a candidate obtained in the current frame. To describe occlusion,Mei and Ling [2] introduce a set of trivial templates **O** defined as

O = [**I**; −**I**] $\in R^{d\times 2d}$ (2)

Where**I** $\in R^{d\times d}$is an identity matrix. Each column of the matrix **I** represents a positive occlusion template, while each column of the matrix −**I** represents a negative one. Hence, in [2], the samples for SR are $\hat{X}$= [**X**;**O**] $\in R^{d\times(n+2d)}$and the SR problem is performed on $\hat{X}$, namely [1]

$$\hat{\beta} = \min_{\beta} E(\beta) = \min_{\beta}\left(\frac{1}{2}\|y - \hat{X}\beta\|_2^2 + \lambda\|\beta\|_1\right) \qquad (3)$$

The size of the training sample decides the computational complexity of the calculations, i.e., the column number of X in (1). In practice, the training sample number *n* is greatly smaller than the feature length *d*.

III.KERNEL SPARSE REPRESENTTION

In order to reduce more expensive computation we bring in 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],

$$\hat{\beta} = \min_{\beta}\left(\frac{1}{2}\left\|\sum_{i=1}^{n}\beta_i\,\varphi(X_i) - \varphi(y)\right\|_2^2 + \lambda\|\beta\|_1\right) \qquad (4)$$

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]

$$\hat{\beta} = \min_{\beta}\left(\frac{1}{2}\beta^T K\beta + K(\cdot, y)\beta + \lambda\|\beta\|_1\right) \qquad (5)$$

Where K is an n×n kernel matrix satisfying

$$K=\begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \quad (6)$$

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

The advantages of KSR on SR are as follows

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

(2)Multiple features can be introduced into KSR by applying multikernel fusion.

One among multikernel fusion methods is the weighted multikernel fusion, where weighted summation is used to obtain a kernel

$K=\sum_{i=1}^{n}\omega_i K_i$ (7)

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

$\mathbf{K}(.,y)=\sum_{i=1}^{n}\omega_i K_i(.,y)$ (8)

Adaptive Multikernel fusion methods focus on the calculation of ω_i. This is calculated using PSO algorithm to get best value of ω_i.

Particle swarm optimization (PSO)is a stochastic optimization approach which maintains a swarm of candidate solutions, referred to as particles [7, 8]. Particles are ‘‘flown’’ through hyper-dimensional search space, with each particle being attracted towards the best solution found by the particles neighborhood and the best solution found by the particle. Swarm intelligence provides a usefulparadigm for implementing adaptive systems.

PSO is used to solve the optimization problems. PSO is initialized with a group of random particles and searches for the optimized by updating. Each particle weight in every iteration is updated by following two "best" values. The first one in the group is called pbest and another tracked by the particle swarm optimizer in the population is called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values [14], the particle updates its velocity and positions with following equation (9) and (10).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[ ] - present[]) (9)

present[ ] = present[ ] + v[ ] (10)

v[ ] is the particle velocity, present[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors, usually c1 = c2 = 2.

The pseudo code of the procedure is as follows[14]

```
For each particle
   Initialize particle
END

Do
   For each particle
      Calculate weight value
      If the weight value is better than the best weight value (pBest) in history
         set current value as the new pBest
   End

   Choose the particle with the best weight value of all the particles as the gBest
   For each particle
Calculate particle velocity according equation (9)
Update particle position according equation (10)
   End
While maximum iterations or minimum error criteria is not attained
```

By making use of PSO we can achieve the best weight value of the particle which would make the system adaptive and optimized.

Further to optimize the KSR in equation (5) we need to use the kernel coordinate descent algorithm. KCD uses coordinate descent approach in [9] due to its simplicity and efficiency. Differentiate E(β) with respect to β_j and set it to 0 we get [1],

$$e(x_j) = \varphi(X_j)^T \left(\varphi(X_j) - \sum_{i=1, i\neq j}^{n} \beta_i \varphi(X_i) \right)$$

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

β_j is changed independently $\{\beta_i\}_{i=1, i\neq j}^{n}$

When calculating β_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.

IV.PARTICLE FILTER-BASED TRACKING ALGORITHM

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 [10], [11].

Let α_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[1]

$$p(\alpha_t | Z_{1:t}) = \int p(\alpha_t | \alpha_{t-1}) p(\alpha_{t-1} | Z_{1:t-1}) d\alpha_{t-1} \quad (14)$$

And update

$$p(\alpha_t | Z_{1:t}) = \frac{p(Z_t | \alpha_t) p(\alpha_t | Z_{1:t-1})}{p(Z_t | Z_{1:t-1})} \quad (15)$$

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[1]

$$w_t^i = w_{t-1}^i p(Z_t|\alpha_t^i) \quad (16)$$

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.

An affine image warping method is adopted for modeling the object motion in video 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 visual tracking. 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[1]

$$p(\alpha_t|\alpha_{t-1}) = \mathcal{N}(\alpha_t; \alpha_{t-1}, \sum) \quad (17)$$

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.

The core of particle filter-based tracking depends on the observation likelihood model $p(Z_t|\alpha_t^i)$. In actual fact, the SR based tracking method in [2] just improves it by using the SR. In this method [1], for the ith candidate object y, we first perform the PSO algorithm to 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 [1]

$$p(Z_t|\alpha_t^i) = \exp\left(-\left\|\sum_{j=1}^{n} \beta\varphi(x_j) - \varphi(y)\right\|_2^2 / \sigma^2\right) \quad (18)$$

where σ is a constant.

In order to compute (18), we need to specify kernels, which correspond to the selected features. In this paper [1], spatial color histogram and spatial gradient histogram are adopted to represent the object. The color feature has gained more attention as it is apowerful alternative to characterize the appearance of object, especially if it can achieve robustness against deformation and partial occlusion.

Algorithm 1: KSR-Based Tracking Algorithm

Data: The kernel matrix Kand the kernel vector K(·, **y**).
Result: The coding vector β.
Step 1:InitializeK and K(., **y**) by (7) and (8) ;
Step 2: Initialize N particles $\{\alpha_1^i\}_{i=1}^N$ at time $t = 1$;
Step 3: Initialize the weights of N particles $\left\{w_1^i = \frac{1}{N}\right\}_{i=1}^N$;
Step 4: **for** $t = 2$ **to** videoLength**do**
Step 5: **for** $i = 0$ **to** N **do**
Step 6:Predicting the state of the *i*th particle α_t^i by (19)
Step 7: Calculateβ by Kernel Coordinate Descent

Initialize the coding vector

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

Step 8: **for** *k* = 0 **to** maxIter**do**
Step 9: **for** *j* = 0 **to** *n* **do**
Step 10: Calculate*e*(**x***j***)**by (11);
Step 11: Update*βj by* (12);
Step 12: end
Step 13: end
Step 14:Calculate the observation likelihood by (18) ;
Step 15:Update the weight of the *i*th particle by (16) ;
Step 16:end
Step 17:Compute the index of maximum weight, i.e., maxI= max ($\left\{w_t^i = \frac{1}{N}\right\}_{i=1}^{N}$);
Step 18: Obtaining the tracking state $\alpha_t = \alpha_t^{maxI}$;
Step 19:Resampling (sampling with replacement) according to the weights $\{w_1^i | i = 1, ., N\}$
Step 20: end

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 histogram h^c is obtained by connecting four subhistograms together. For the gradient feature, we first obtain image gradients by performing two filtering operations with kernels $\left[-\frac{1}{2} \quad 0 \quad \frac{1}{2}\right]$ and $\left[-\frac{1}{2} \quad 0 \quad \frac{1}{2}\right]^T$. As with the spatial color histogram, we can obtain for subhistograms of oriented gradients [12]. The spatial gradient histogram h^g is obtained by connecting four subhistograms together.

In tracking algorithm, the object 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 perturbating a few pixel around corner points of the first template. Here template number is set by balancing tracking efficiency and computational complexity. The templates are updated by replacing a template that has the smallest representation coefficient of the current tracked objectfor the changing target appearance. The particles are updated by (17).The state $\alpha_t^i = [a_{11}\, a_{12} a_{21}\, a_{22}\, t_x t_y]_t^i$ for ith particle at current time t is calculated as

$$\alpha_t^i = [a_{11}\, a_{12} a_{21}\, a_{22}\, t_x t_y]_t^i = [a_{11} a_{12} a_{21}\, a_{22}\, t_x t_y]_{t-1}^i + \varepsilon \qquad (19)$$

V. RESULTS AND DISCUSSION:

The Matlab based simulation was carried out to examine the validity of this method. A car video from the website [13] referred was used in our implementation. And the video that was taken had the occlusions like the shadow and tree shadows on the car. The result when multiple objects being in the same frame is given in the Fig1.

Fig1. Tracking the Object while multiple Object on the same frame.

The figure 2 depicts that the algorithm tracking the object while the shadow occlusion is available on the frame.

The algorithm tracks the car passing under the bridge which is fully covered by shadow of bridge shown in fig.3 considering the shadow occlusion.

For running this program we used the PC with the following specifications

Intel i5, 3.3GHz, 8GB RAM, 32 bit operating System Windows 7 with Matlab 2011b.

The execution time of the simulation took around 101.407737 seconds. The video with 600 frames were taken and the frame rate was taken as 17 frames /sec.

The proposed algorithm could reduce the execution time but maintaining the accuracy the same. The multichannel method has helped the execution of the tracking more accurate even if there are occlusions like shadow on the car and near the car.

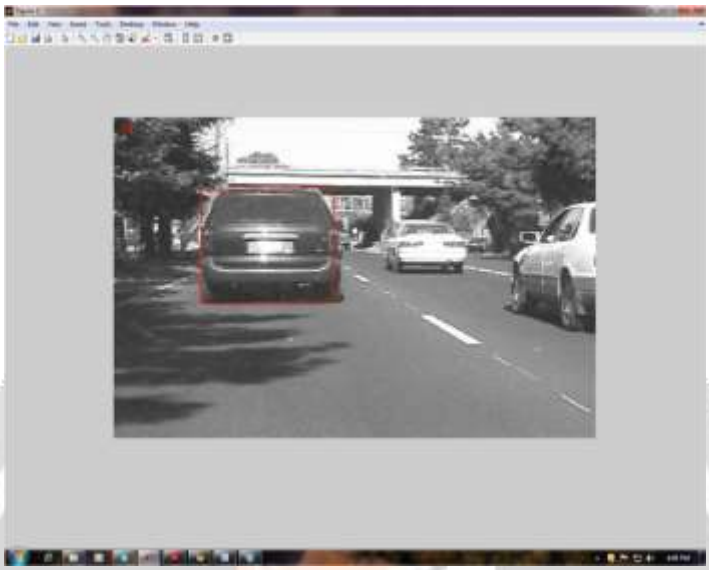

Fig2. Tracking with shadow occlusion

Fig.3. Tracking with full shadow on the Object

VI.CONCLUSION

A novel technique of optimizing the weight updation of the multikernel fusion is introduced. Although computational complexity is higher PSO would prove itself to be attaining a optimized output with reduced execution time. Matlab simulation would prove that the optimization of the weight updation would provide us a sparser kernel matrix. The PSO method used for weight optimization in the algorithm will help in implementing the algorithm faster. The execution time of the method was found to be faster than the earlier method but maintaining the accuracy the same. The tracking was found consistent with the occlusions like the shadow of the tree on the car, shadow of the car itself and the shadow of a bridge over the car.

Acknowledgment

I am very grateful and would like to thank my guide Asst. Prof. Vijaya Saratha for his advice and continued support. I would like to thank my friend for the thoughtful and mind stimulating discussion we had, which prompted us to think beyond the obvious.

References

[1] Lingfeng Wang, Hongping Yan, KeLv, and Chunhong Pan, "Visual Tracking via Kernel Sparse Representation with Multikernel Fusion "*IEEE transactions on circuits and systems for video technology,* VOL. 24, NO. 7, JULY 2014.

[2] Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. Kulikowski. Robust and fast collaborative tracking with two stage sparse optimization. In ECCV, 2010.

[3] X. Mei and H. Ling. Robust Visual Tracking and Vehicle Classification via Sparse Representation. TPAMI, 33(11):2259–2272, 2011.

[4] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proc. IEEE Conf. Comput. Vision PatternRecognit.*, Jun. 2012, pp. 2042–2049.

[5] Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2012, pp. 1830–1837.

[6] S. Gao, I. W.-H. Tsang and L.-T. Chia, "Sparse representation with kernels," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 423–434, Feb. 2013.

[7] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEEInternational Joint Conference on Neural Networks, IEEE Press, 1995, pp. 1942–1948.

[8] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings ofthe Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan,1995, pp. 39–43.

[9] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Stat. Softw.*, vol. 44, no. 1, pp. 1–22, 2010.

[10] M. Isard and A. Blake, "Condensation: Conditional density propagation for visual tracking," *Int. J. Comput. Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[11] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity based collaborative model," in *Proc. IEEE Conf. Comput. Vision PatternRecognit.*, Jun. 2012, pp. 1838–1845.

[12] N. Dalal and B. Triggs,"Histograms of oriented gradients for human detection,"in*Proc. IEEE Conf. Comput. Vision Pattern Recognit.*,Jun. 2005, pp. 886-893.

[13] www.ist.temple.edu.