

Improvements in Blind Image Restoration

Utkarsh Srivastava
Sachin Grover



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Improvements in Blind Image Restoration

Thesis submitted in

May 2010

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

Utkarsh Srivastava

(Roll 10606065)

Sachin Grover

(Roll 10606059)

under the supervision of

Prof. Banshidhar Majhi



Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Banshidhar Majhi
Professor & Head

May 7, 2010

Certificate

This is to certify that the work in the thesis entitled *Improvements in Blind Image Restoration* by *Utkarsh Srivastava* and *Sachin Grover*, bearing roll number 10606065 and 10606059 respectively, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

B. Majhi

Acknowledgment

This dissertation, though an individual work, has benefited in various ways from several people. Whilst it would be simple to name them all, it would not be easy to thank them enough.

The enthusiastic guidance and support of *Prof. Banshidhar Majhi* inspired us to stretch beyond our limits. His profound insight has guided our thinking to improve the final product. Our solemnest gratefulness to him.

Our sincere thanks to *Ratnakar Dash* for their continuous encouragement and invaluable advice.

Special thanks to *Prof Pankaj Kumar Sa*, without whom, the work in latex would not have been this simple.

Many thanks to our comrades and fellow research colleagues. It gives us a sense of happiness to be with you all.

Utkarsh Srivastava Sachin Grover

Abstract

We present a revisal of blind image deconvolution technique for the restoration of linearly degraded images, without the explicit knowledge of either original image or the psf- the point spread function. Even the scenes which consist of finite support object over a uniformly black, white or grey background, this technique works fine. Occurrence includes certain types of medical imaging, astronomical imaging, and (1-D) gamma ray spectra processing. The only information that is required are the nonnegativity of the true image and the support size of the original object.

The restoration procedure involves recursive filtering of the blurred image to minimize a convex cost function. The new approach is experimentally shown to be more reliable and to have faster convergence than existing nonparametric finite support blind deconvolution methods, for situations in which the exact object support is known.

This thesis covers the basic implementation of NAS-RIF method, using steepest descent, followed by implementation of swarm optimization technique- ACO, to optimize the results.

Keywords: Image restoration, PSF, out-of-focus blur, motion blur, blind image deconvolution.

Contents

Certificate	ii
Acknowledgement	iii
Abstract	iv
List of Figures	vii
List of Tables	viii
1 Introduction	1
2 Optimization algorithms	5
2.1 General Descent Algorithms	6
2.1.1 Steepest Descent Algorithm	7
2.1.2 Conjugate Gradient Algorithm	7
3 Swarm Intelligent Techniques	9
3.1 Ant Colony Optimization	10
3.1.1 Double Bridge Experiment	11
3.1.2 Algorithm	12
3.1.3 ACO variants	14
3.2 Particle Swarm Optimization	15
3.2.1 Algorithm	16
3.2.2 PSO variants	17
3.3 Results and Implementation	18

3.3.1	Parameters	19
4	NAS-RIF and its variants	21
4.1	Problem Formulation	22
4.1.1	NAS-RIF Algorithm	22
4.2	NAS-RIF using steepest descent	24
4.2.1	Results For Steepest Descent	25
4.3	NAS-RIF using conjugate gradient	25
4.3.1	Results For Conjugate Gradient	28
5	Proposed NAS-RIF Algorithm using ACO	30
5.1	Algorithm	31
5.2	Implementation and Results for Proposed NAS-RIF using ACO . .	33
5.2.1	<i>Out of Focus</i> Blur Images	33
6	Conclusion	36
	Bibliography	38

List of Figures

1.1	Model for image degradation and restoration process	2
3.1	Same length Setup for Double Bridge Experiment	11
3.2	Different length setup for Double Bridge Experiment	12
4.1	NASRIF flowchart	23
4.2	The concept of True Support	23
4.3	Original, Blurred and Restored Image for NAS-RIF using steepest Descent for <i>motion blur</i>	25
4.4	Original, Blurred and Restored Image for NAS-RIF using steepest Descent for <i>out of focus blur</i>	25
4.5	NAS-RIF using Steepest Descent	27
4.6	Original, Blurred and Restored Image for NAS-RIF using Conjugate Gradient for <i>out of focus blur</i>	28
4.7	NAS-RIF using Conjugate Gradient Method	29
5.1	Original, Blurred and Restored Image for proposed Ant Colony Optimized NAS-RIF for <i>out of focus blur</i>	33
5.2	ACO working on the blurred Image 5.1 used for <i>NAS-RIF using ACO</i>	33
5.3	NAS-RIF using Ant Colony Method	34
5.4	ACO working on the blurred Image 5.3(b) used for NAS-RIF . . .	35

List of Tables

3.1 Average Results for Continuous Optimization	20
---	----

Chapter 1

Introduction

The standard degradation model in most of images can be modeled as the following [1] [2] -

$$g(x, y) = h(x, y) * f(x, y) + n(x, y) \quad (1.1)$$

where,

(x, y) discrete pixel coordinate of the image

$g(x, y)$ image which has been blurred

$f(x, y)$ original image

$h(x, y)$ the point spread function (PSF)

$n(x, y)$ additive noise

* 2D convolution operator

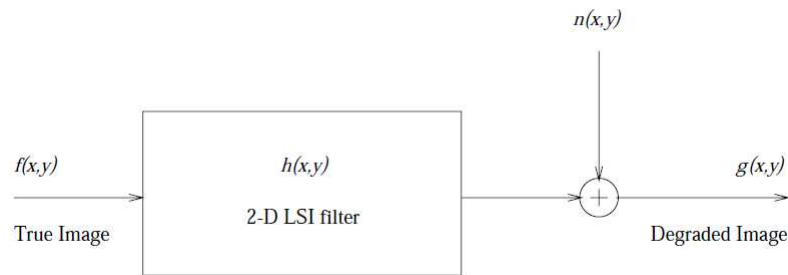


Figure 1.1: Model for image degradation and restoration process. [1]

In the above model, the degraded image $g(x, y)$, the original image $f(x, y)$, and the noise $n(x, y)$ are linearly combined, and as a result the problem of retrieving the image from degraded image is called as *linear image restoration problem*. The currently available image restoration algorithms approach that the point spread function is available *a priori* and they try to regain the original image by trying to revert the PSF [3], by changing the information about the noise, original image and PSF.

However, in real life, very little or no information is available about the original image, and the point spread function is almost always unknown. Hence very little can one approach about the real image. In such cases, the current existing linear image restoration algorithms are not applicable as they require some prior

information about the PSF as well as the image. The method of concurrently assaying the point spread function (or either the inverse of it) and reconstructing the original image which is unknown is called as *Blind Image Restoration*. In the cases where the noise $n(x, y)$ is disregarded, the process is distinctively called as *blind image deconvolution*.

The reasons to act as the compelling factor to use blind image restoration are many. In most of the situations, the ability to measure the amount of degradation using either fine tuning or online identification methods is limited; and apart from that, it may be dangerous, costly or tangibly impossible to acquire *a priori* information about the image which is supposed to be captured. For eg, in space imaging and sensor remote sensing, inconstancy in the value of Point spread function are difficult to distinguish, and it is almost impossible to statistically portrait the original image. As a result of which, blind image deconvolution techniques may be required for post processing.

In usual procedure, some *a priori* fact about the original image is required to restore the image profitably. The real deciding factor is to design an algorithm that display the most suitable compromise among reliability, portability, computational complexity and robustness to noise for a particular application [4] [5].

This thesis contributes by first describing development and implementation of a new blind image deconvolution technique, for restoring the images which have been degraded linearly, or as per the given model 1.1. Specific information about either the real image or the PSF is not necessary. The technique studied requires that the images that are to be restored should be taken against white, grey or a black background. Also that the object in consideration is within the bounds of the image, that is, the support for the true image is defined. The only information needed for restoring the real image is the non negativity of the original image and the true support size, or the edge size of the original object in consideration. The restoration method comprises of repeatedly filtering of the degraded image to minimize a convex cost operation. The algorithm for the studied image deconvolution technique is discussed in subsequent pages, with various optimization techniques implemented.

Another contribution of this thesis is to describe the designing of the foretold algorithm with swarm optimization technique used for optimization. An exhaustive study of the various techniques under swarm optimization is described, followed with the embedding of the ACO technique as optimization strategy for the blind deconvolution technique.

Despite the fact that this thesis deals with 2 dimensional signals, the work proposed is equally applicable on 1 dimensional signals as well, for eg, γ ray spectra processing.

Chapter 2

Optimization algorithms

A lot of techniques are available to minimize a convex cost function. Since the suggested cost function is a non linear function, it is very difficult to statistically determine the function which can be used as a minimizer for the given function. Normally, the various methods available are different from one another as they make different assumptions about the initial conditions, and work differently subsequently. Also are different are the inference of the structure for the descent required, the requirement of the gradient, and argosy requirements.

2.1 General Descent Algorithms

Such an optimization algorithm is required which makes an effectual usage of the accessible facts. And since we have the knowledge of the cost function, and hereby the corresponding gradient of the cost function, hence, the family of *descent algorithms* [1] is considered.

In methods such as the descent algorithms, a catenation $\{u_k\}$ is build such that [1]

$$u_k \rightarrow \bar{u} \text{ where } J(\bar{u}) \leq J(u) \forall u \in R \quad (2.1)$$

or at least [1]

$$\nabla J(u_k) \rightarrow 0 \text{ when } k \rightarrow \infty \quad (2.2)$$

In this method, the cost function J is compelled to decrease at every step, or iteration. And a minimization algorithm is said to converge only iff equation 2.2 holds true.

In the given algorithm for a general descent, the following are the parameters. J depicts the cost function given, u_k is the projection of the result at the kth loop, for the line search algorithm to find the u_{k+1} , the direction is given by d_k , and the current step size is given by t_k . The budget $J(u)$ and $\nabla J(u)$ exist and are known to be decipherable.

Algorithm 1 General Descent Algorithm [1]

```
Set an Initial guess
if  $f(x) \geq \delta$  then
    Determine the direction of descent
    Perform Line Search
    Update solution estimate
end if
```

We describe two different types of descent algorithms, which have been investigated: the steepest descent algorithm and the conjugate gradient algorithm.

2.1.1 Steepest Descent Algorithm

One of the most widely and popularly used method is the method of steepest descent [6] [7]. The prime factor of its popularity being that it is undismaying and very easy to use. In this method, the value of the cost function is calculated based upon the current abscissa, and the direction is decided by computing the first order differential of the cost function. Now, to traverse along the steepest negative direction, so as to reach the Global minima, the algorithm then assumes a step size, which remains constant through out the working of the algorithm.

Although this algorithm is fairly simple and easy to implement, it suffers problems like slow convergence to the solution, and inability to dynamically change the step size. As a result, it at times may under reach the solution, or overshoot the solution. The initial guess of the step size T_k is very important. The following is the algorithm used for steepest descent. In this, the new gradient of the line is always perpendicular to the direction of the gradient traversed a step before that, and hence may not reach the global minimum.

2.1.2 Conjugate Gradient Algorithm

This method is a much more efficient method than the steepest descent algorithm, and it assumes that the gradient of the cost function is calculable at every step, and

Algorithm 2 Steepest Descent Algorithm

define the permissible δ value near the minima of the function $f(x)$
 find a random starting point $x^{(0)}$
 Fix a step size α to minimize $f(x)$
if $f(x) \geq \delta$ **then**
 Calculate the gradient of function $f(x)$ at $x^{(0)}$
 Let the search direction be $-\nabla f(x)$
 Update $x^{(k+1)} = x^{(k)} + \alpha * d^{(k)}$ {where $d^{(k)}$ is the direction of the fall of the gradient}
end if

that this information can be used to improve the search for the global minimum. The Conjugate gradient method [8] [9] uses the conjugate gradients for traversing downhill, in place of the gradient of the cost function. As a result, the solution is quick and reached in comparatively fewer iterations.

The following is the algorithm for the conjugate gradient algorithm:

Algorithm 3 Conjugate Gradient Algorithm

define the permissible δ value near the minima of the function $f(x)$
 find a random starting point $x^{(0)}$
if $f(x) \geq \delta$ **then**
 Calculate the gradient of function $f(x)$ at $x^{(0)}$
 Let the search direction be $-\nabla f(x)$
 Find an appropriate step size α such that $f(x + \alpha^{(k)} * d^{(k)}) < f(x + \alpha * d^{(k)})$
 {where $d^{(k)}$ is the direction of the fall of the gradient}
 Update $x^{(k+1)} = x^{(k)} + \alpha * d^{(k)}$
end if

Chapter 3

Swarm Intelligent Techniques

Swarm Intelligent Techniques [10, 11] consists of two keywords. First one is Swarm which means a group or a herd of organisms moving together, towards a common goal, second one being Intelligent which refers to the intelligence i.e. the functioning of a herd working together to reach that common goal, or how it reacts to a change in conditions etc. These techniques are biologically inspired techniques, after studying the functioning various organisms like ants, particle swarms, bird flocks, and even the evolution of human population etc. Each of these gives rise to a different technique, which are heavily used these days in day to day optimization problems. Our study of these techniques include implementing them for optimizing continuous functions:

- Ant Colony Optimization (ACO): based on the study of Ant colonies looking for food
- Particle Swarm Optimization (PSO): based on the study of Bird flocks searching for food

Now, we look into these Algorithms in depth.

3.1 Ant Colony Optimization

Ant Colony Optimization [12, 13] also known as ACO, is a population based metaheuristic which is being used extensively now a days to solve complex optimization problem. It has evolved after studying the foraging behavior of some species of ants. While looking for food ants deposit some amount of a chemical called pheromone to mark some favorable path. The higher the amount of pheromone on a path more is the probability of an ant to take that path. This gradual approach of foraging is harnessed to solve many optimization problems which have huge solution set and are computationally classified as NP hard or NP complete like Travelling Salesperson Problem [14].

In early 19th century, French entomologist Pierre-Paul Grassé [15] observed that some species of ants and termites react to the stimuli created by one of

them only. He called this phenomenon to be *Stigmergy* [16] to describe this type of communication among individuals where the "Workers are stimulated by the performance they have achieved". This phenomenon forms the basis of Ant Colony Optimization algorithm.

3.1.1 Double Bridge Experiment

This was the first experiment performed to test the pheromone production of Argentine ants. In this, a setup was made consisting of two paths for ants to travel to reach the food source. The ants were left to find the optimized path. The lengths of both the paths were varied, and interesting results were seen [12].

When the length of the paths were same, at the start of the two paths, for the first time the paths were taken randomly by ants. When more number of ants started taking one side of the bridge that got more preference because of high concentrations of pheromone deposited on it. It was found, by repeating the whole experiment again and again that both of the bridges were taken with equal probability.

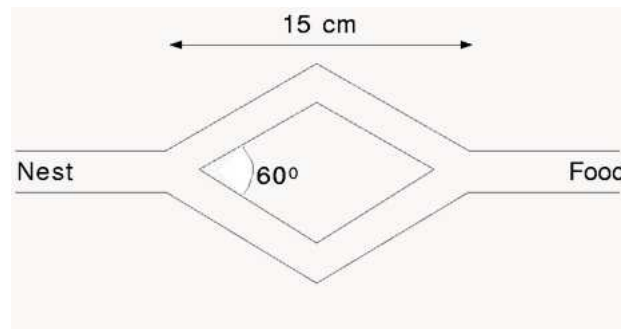


Figure 3.1: Same length Setup for Double Bridge Experiment

When the length of the paths were significantly different, for the first time at the intersection the bridges were again chosen randomly. But the ant going from the shorter bridge did reach the food source before the ants going from the longer bridge. So the pheromone concentration increased at a faster rate on the smaller path than on the longer one. Hence the shorter route was preferred by

more number of ants the next time. After sometime it was observed that all the ants took the shorter path towards the food source.

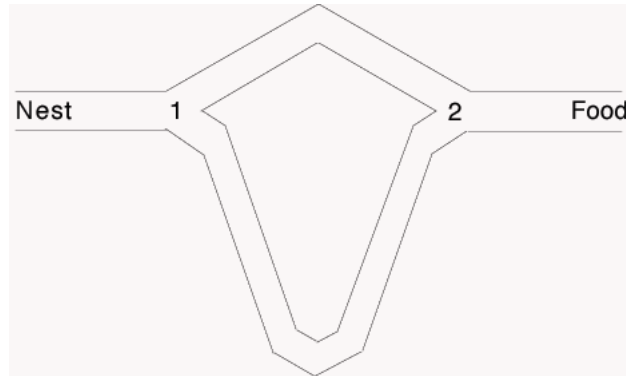


Figure 3.2: Different length setup for Double Bridge Experiment

3.1.2 Algorithm

The construction of solutions to optimization problems using Ant Colony Optimization by traversing the fully connected construction graph. The set of all possible solution components are defined using artificial ants, then they are left to choose and traverse their paths randomly over that fully connected graph, and pheromone values are updated with every iteration.

The basic algorithm consists of [17]:

Algorithm 4 Ant Colony Optimization

Require: numberOfAnts, Graph

- 1: Set Parameters and initial pheromone values, and rate of evaporation
 - 2: **while** Optimal solution is not reached **do**
 - 3: Construct Ant Solution
 - 4: Demon Action
 - 5: Update Pheromone
 - 6: **end while**
-

First, we set the parameters and initial pheromone deposition and evaporation rate. The steps after this are repeated again and again until an optimal solution

is reached. These steps are as follows:

Construct Ant Solution

We start by construction a partially empty solution state $s^p = \phi$, and then this partial solution is extended by adding a solution component from the feasible value. Finding a solution can be regarded as searching a path on a graph $G_c(V, E)$, which are defined during the construction phase of the solution set. The path finding is done probabilistically depending on the pheromone values that are there on all the paths. This probability function varies in all the variants of Ant Colony System.

Demon Action

This is an optional step. Now we add some problem specific actions that may be required. In this we implement some centralized action which can not be done by a single ant, which are completely problem specific.

Pheromone Update

This is the most important step in the algorithm. The aim of this step is to increase the pheromone values of the good solution and to decrease the values of bad ones. This is achieved by:

1. Decreasing all the pheromone values by a certain factor.
2. Increasing the good values by conjugate of that factor, depending on the fitness value calculated.

We use a general pheromone upgradation equation:

$$\tau_{ij}^k = (1 - \rho)\tau_{ij}^{k-1} + \rho \sum_s F(s), \quad (3.1)$$

where,

- τ_{ij}^k is the pheromone value associated with a path,
- ρ is a parameter called evaporation rate,

- $F(s)$ is the fitness value of each solution seen

Pheromone evaporation is the factor which defines the rate of forgetting the bad solutions and as per the requirement of the problem can be taken to high values, which might be infallible biologically.

3.1.3 ACO variants

There are a lot variations of ACO that have been published in the literature for specific problem. Like for continuous optimization algorithms like CACO (Continuous Ant Colony Optimization) [18]. Here are some of the earliest Ant Colony variants:

Ant System

This was the first ACO algorithm in literature [13], in this the pheromone values are updated by all the ants that have completed a tour.

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (3.2)$$

where,

- m are the number of ants,
- τ_{ij} and ρ are the pheromone value and the evaporation rate respectively,
- $\Delta\tau_{ij}^k$ is the quantity of pheromone laid on the edge (i, j) by the k -th ant,

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k} & \text{if ant } k \text{ uses the edge}(i, j)\text{in its tour,} \\ 0 & \text{otherwise,} \end{cases} \quad (3.3)$$

where L_k is the tour length of k -th ant.

At each vertex Ant System makes a probabilistic decision by using:

$$p(C_{ij}|s_k^p) = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{c \in N(s_k^p)} \tau_{ij}^\alpha \cdot \eta_{ij}^\beta} & \text{if } j \in N(s_k^p), \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

where,

- $N(s_p^k)$ are the solutions that do not belong to the partial solution s_p^k of ant k ,
- η_{ij} is the heuristic information and $\eta_{ij} = \frac{1}{d_{ij}}$ where d_{ij} is the length component
- α and β are the parameters that control the pheromone and the heuristic information.

Max-Min Ant System

This is another variant which was proposed in the literature [19], and it was applied to the Travelling Salesperson Problem. It differs from Ant System as only the best ant add the pheromone trail, i.e. if the best path is taken the pheromone trail is added, as per the following equation:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij}^{best} \quad (3.5)$$

where, $\Delta\tau_{ij}^{best} = \frac{1}{L_{best}}$ and L_{best} is the length of the tour of the best ant and it is taken to be 0 otherwise. This is the usual idea i.e. taken for applying it to the continuous functions.

3.2 Particle Swarm Optimization

Particle Swarm Optimization [20] also known as PSO, is a population based stochastic which is being used to solve continuous as well as discrete optimization problems. It was inspired by the social behavior of bird flocking or fish schooling. In this the each individual called as particle moves in the search space changing its velocity and direction the rules of changing are taken randomly or are problem specific. Then the particle checks for the food at the new position that it has reached.

In PSO technique an individual is represented as a single particle in n -dimensional space and the solution to the problem is encoded as the coordinates

of the particle in space. PSO has been applied to many problems with tremendous success [21] [22].

3.2.1 Algorithm

Algorithm 5 PSO

Require: particleList, GBEST, iterations

```
1: for  $i = 0$  to iterations do
2:   for every particle in particleList do
3:     calculate fitness
4:     if fitness of particle > personal best fitness of particle then
5:       personal best = new fitness
6:     end if
7:     if fitness of particle > global best fitness of particle then
8:       global fitness = new fitness
9:     end if
10:  end for
11:  for every particle in particleList do
12:    update velocity
13:    update position
14:  end for
15: end for
```

The particles in the swarm are made to fly in the solution space until they arrive at the solution to the problem, i.e, the value of individual dimensions represent values of the variables of the objective function. The solution is represented by a fitness function which is usually the objective function that needs to be maximized or minimized. The result of evaluation of the fitness function for a particular particle is called the fitness of the particle. Particles communicate with each other and their search through the solution space is guided by the fitness of the global best particle, and their own fitness. The particle i is a structure consisting of 4

variables namely $(X_i, V_i, PBEST_i, Fitness_i)$ which are defined as:

- $X_i = \{x_1, x_2, x_3, \dots, x_n\}$ represents the position vector of the particle in n -dimensional space.
- $V_i = \{v_1, v_2, v_3, \dots, v_n\}$ represents the velocity vector of the particle in n -dimensional space.
- $PBEST_i = \{x_1, x_2, x_3, \dots, x_n\}$ represents the position of the particle in which it had the best fitness value.
- $Fitness_i$ represents the fitness of the particle i .

The PSO algorithm also maintains a GBEST value, which is the position of current best fitness value amongst all the particles in the swarm. The PSO algorithm is given in *Algorithm 5*. At each successive iteration, the particle's velocity and position vectors are updated according to Equations (3.6) and (3.7) respectively.

$$V_i^t = wV_i^{t-1} + c_1(PBEST_i - X_i) + c_2(GBEST_i - X_i) \quad (3.6)$$

$$X_i^t = X_i^{t-1} + V_i^t \quad (3.7)$$

V_i^t represents the velocity of particle i at iteration t and similarly X_i is the position of particle i at iteration t . w is a linearly decreasing *constraining* factor which limits the increase in the velocity of the particle and keeps the particle within the bounds of the solution space. c_1 and c_2 are referred to as the cognitive and the social factors respectively. They govern the amount of influence that *PBEST* or *GBEST* have on the particle's next position.

3.2.2 PSO variants

Since, the proposal of Particle Swarm Optimization, a lot of variants have been proposed. Most of them change the way of velocity is updated for a particle. Some of the main variants are listed below:

Deiscrete PSO

The basic PSO is designed to search in the continuous domain, but there are a number of variants that run in discrete spaces. In this algorithm the velocity of the particle is taken to be continuous but the position is taken to be discrete and are updated using:

$$x_{ij}^{t+1} = \begin{cases} 1 & \text{if } r < \text{sig}(v_{ij}^{t+1}), \\ 0 & \text{otherwise} \end{cases}, \quad (3.8)$$

where,

- x_{ij} is the j^{th} component of the particle,
- r is the uniformly distributed random number in the interval $(0, 1]$,
- v_{ij}^{t+1} is the velocity function of the j^{th} particle, and
- $\text{sig}(x) = \frac{1}{1+\exp^{-x}}$

Fully Informed PSO

In basic PSO algorithm the particles is attracted towards its best neighbor [23], in this algorithm particle uses the information provided by all its neighbors to update its velocity. Their can be weights attached to the information so that the results can be improved.

3.3 Results and Implementation

Ant Colony Optimization Algorithm and Particle Swarm Optimization Algorithm have been used successfully on various optimization problems in literature. We have implemented these algorithms for optimization of standard multim minima optimization functions.

These functions are:

- Griewangk Function

$$f1 = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \frac{x_i}{\sqrt{i}} + 1$$

- Rastrigin Function

$$f2 = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos 2\pi x_i)$$

- Rosenbrock Function

$$f3 = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_{i-1})^2)$$

All of them have various local minimas, but global minima is present only at $(0, \dots, 0)$.

3.3.1 Parameters

For, ACO:

- The number of Ants were taken to be 30,
- The pheromone updation rate was taken to be 0.05,
- the Evaporation rate was taken to be 0.95,

For, PSO:

- The number of particles were taken to be 30,
- The personal best constant was taken to be 2 and global best constant was also taken to be 2.

The stop criteria was taken to be number of Functional evaluations, which was 30,000. The average value was found over 10 independent experiments. The table shows the following results:

Table 3.1: Average Results for Continuous Optimization

function	ACO	PSO
f1	3.57e+0	1.29e+0
f2	4.93e+0	2.20e+1
f3	2.11e+1	8.98e+0

Chapter 4

NAS-RIF and its variants

4.1 Problem Formulation

The purpose of the doing blind deconvolution of an image is to get some approximation of the actual, or the original image. In the studied algorithm, the following assumptions are pre conceived: [2] [24]:

1. The degradation is modeled by the equation given in Introduction 1.1
2. The object under consideration is enclosed entirely within the image bounds
3. The image is composed of the background grey, white or black.
4. The image is completely non negative.
5. The true support of the image is given *a priori*.
6. The original image and the point spread function both are irreducible, which means that either cannot be expressed as the 2 dimensional convolution of 2 or more images.
7. The point spread function and its inverse both are absolutely summable.

4.1.1 NAS-RIF Algorithm

In the Nonnegativity And Support Constraints Recursive Inverse Filtering Algorithm also known as NAS-RIF algorithm [1] [2], the degraded image $g(x, y)$ is used as input to a FIR filter $u(x, y)$. The output of the filter represents the approximation of the original image, denoted by $\hat{f}(x, y)$. The following describes the cost function used for the deconvolution. It has three parts:

1. first disciplines the negative pixels inside the support region.
2. second penalizes the pixels in the non support region which differ in value from the background value L_b .
3. third component is active in case the background value of the true image L_b is black.

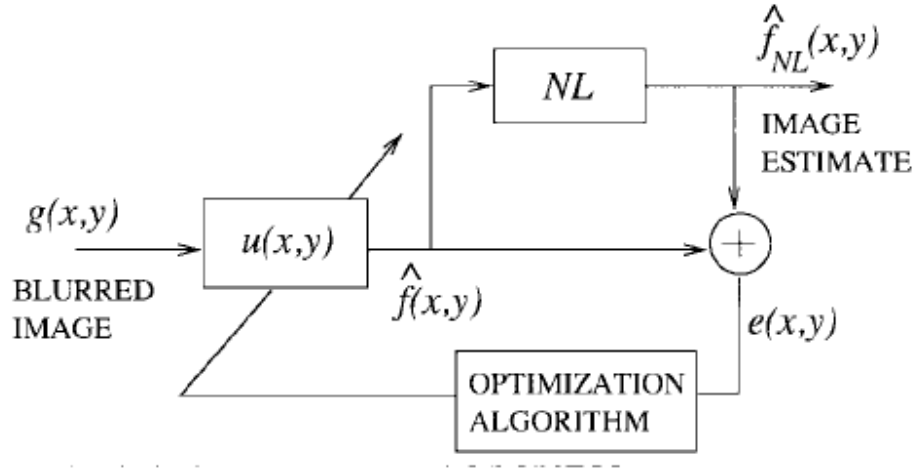


Figure 4.1: NASRIF flowchart. [2]

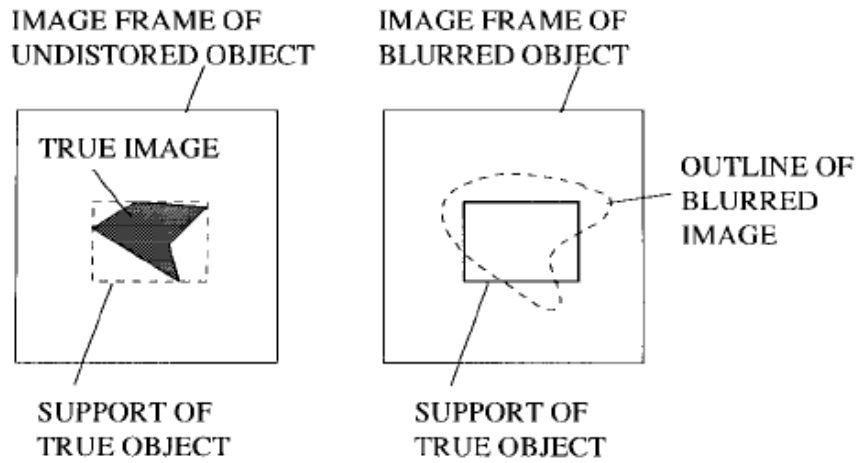


Figure 4.2: The concept of True Support [2]

The following is the equation for calculating the cost function:

$$\begin{aligned}
 J = & \sum_{(x,y) \in D_{sup}} \hat{f}(x,y) \left[\frac{1 - \text{sgn}(\hat{f}(x,y))}{2} \right] + \sum_{(x,y) \in \overline{D}_{sup}} [\hat{f}(x,y) - L_B]^2 + \\
 & \gamma \left[\sum_{\forall(x,y)} u(x,y) - 1 \right]^2 \quad (4.1)
 \end{aligned}$$

Definitions

- u_k is the vector filter coefficient $u(x, y)$ of dimension $N_{xu} \times N_{yu}$ in lexicographical order at the k th iteration,
- $u_k(x, y)$ is the filter $u(x, y)$ at the k th iteration,
- $J(u_k)$ is the cost function of Equation 4.1 at parameter setting u_k ,
- $\nabla J u_k$ is the $N_{xu} N_{yu} \times 1$ gradient vector of J at u_k ,
- $[M]_{x,y}$ denotes the x th row and y th column of matrix M ,
- $\langle f, g \rangle$ represents the inner product of functions f and g ,
- $\|f\|$ is the Euclidean norm,
- D_{sup} is the set of pixels inside the region of support,
- $\overline{D_{sup}}$ is the set of pixels outside the region of support.

4.2 NAS-RIF using steepest descent

This algorithm makes use of the the steepest descent algorithm [1] described in the section 2.1.1. The Step size is initially chosen such as to minimize the cost function, and the gradient is computed everytime in each iteration.

The filter $u_k(x, y)$ is updated using the step size and the gradient.

Algorithm 6 NAS-RIF algorithm using steepest descent

Set initial conditions, $u_k = [0, 1, 0]$, tolerance $\delta > 0$ and step size t

if $J(u_k) \geq \delta$ **then**

$$f_k(x, y) = u_k(x, y) * g(x, y)$$

calculate ∇J

calculate $J(u_k)$

$$u_{k+1} = u_k + t * d_k$$

end if

4.2.1 Results For Steepest Descent

Results for NAS-RIF using Steepest descent on *Motion Blur* images

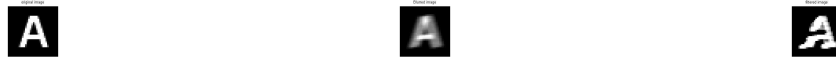


Figure 4.3: Original, Blurred and Restored Image for NAS-RIF using steepest Descent for *motion blur*

The method of NAS-RIF was implemented on a self generated sample image of the letter A. The image was degraded using motion blur filter with length of PSF as 7 and the direction as 10° , and then subsequently restored using optimization of Steepest Descent.

Results for NAS-RIF using Steepest descent on *Out of Focus* Blur images

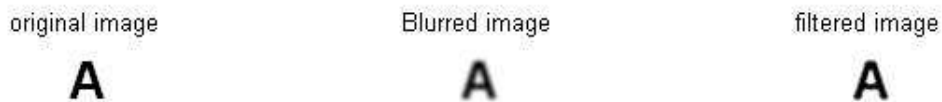


Figure 4.4: Original, Blurred and Restored Image for NAS-RIF using steepest Descent for *out of focus blur*

4.3 NAS-RIF using conjugate gradient

This implementation of the NAS-RIF [1] [2] employs the conjugate gradient method of optimization. The conjugate gradient method, being an iterative

method, is used mainly where the system is so large that the direct method takes up too much time.

In the following algorithm, the information about the gradient is used to update the step size using the conjugate directions computed from previous gradient.

Algorithm 7 NAS-RIF algorithm using Conjugate Gradient

Set initial conditions, $u_k = [0, 1, 0]$, tolerance $\delta > 0$

if $J(u_k) \geq \delta$ **then**

$$f_k(x, y) = u_k(x, y) * g(x, y)$$

calculate ∇J

calculate $J(u_k)$

if $k=0$ **then**

$$d_k = -\nabla J$$

else

$$\beta_{k-1} = \frac{\langle \nabla J(u_k) - \nabla J(u_{k-1}), \nabla J(u_k) \rangle}{\|\nabla J(u_{k-1})\|^2}$$

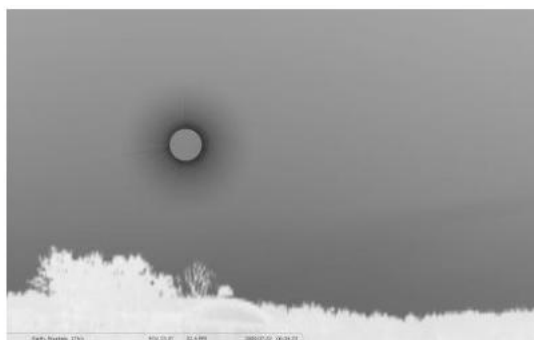
$$d_k = -\nabla J(u_k) + \beta_{k-1} d_{k-1}$$

end if

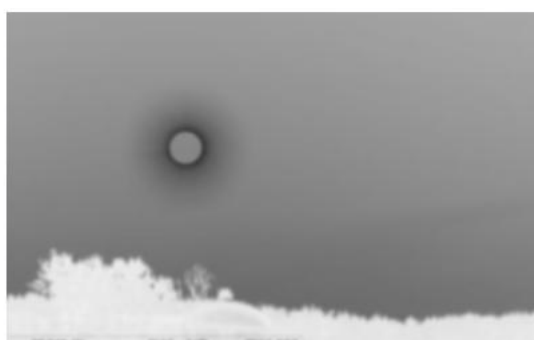
Perform line minimization i.e $J(u_k + t_k d_k) \leq J(u_k + t d_k)$

$$u_{k+1} = u_k + t_k * d_k$$

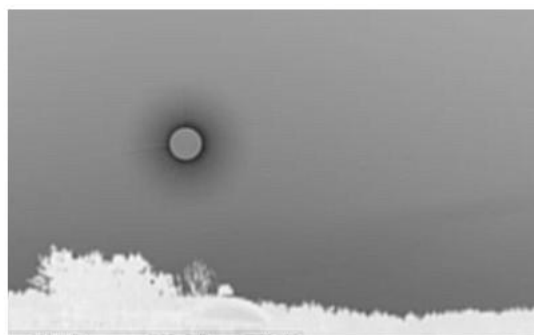
end if



(a) Original Image



(b) Blurred Image



(c) Restored Image

Figure 4.5: NAS-RIF using Steepest Descent

4.3.1 Results For Conjugate Gradient

Results for NAS-RIF using Conjugate Gradient on *Out of Focus* Blur images



Figure 4.6: Original, Blurred and Restored Image for NAS-RIF using Conjugate Gradient for *out of focus blur*

Figure 4.6 show the result, when Conjugate Gradient method was applied to out of focus blur. The results were achieved in 6 iterations which was pretty less compared to the steepest descent method.

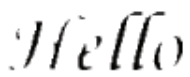
Figure 4.7 shows the result when NAS-RIF using Conjugate Gradient Method was applied to motion blur, again in this case the results were achieved in 7 iterations. The results can be further improved using morphological methods.



(a) Original Image



(b) Blurred Image



(c) Restored Image

Figure 4.7: NAS-RIF using Conjugate Gradient Method

Chapter 5

Proposed NAS-RIF Algorithm using ACO

In this chapter, we propose a different approach for optimizing Blind Image Restoration using NAS-RIF algorithm, as described in [2] [1], by using Swarm Intelligent Technique to find the global minimum in the cost function of NAS-RIF as per the Equation 4.1. NAS-RIF algorithm using steepest descent, as explained in Section 4.1 has a constant step size, which iteratively takes it closer to the optimum value. In this new algorithm we have calculated this step size using the pheromone updation value, as per the MAX-MIN Ant System, as explained in Section 3.1.3.

5.1 Algorithm

The MAX-MIN ant system is used, first twenty random instances of step size are generated and each ant carries that value. Corresponding to each step size calculate the cost function, then pheromone value corresponding to the minimum cost value to the pheromone value of the same ant in the previous iteration are subtracted and the change is added to every ant. Hence, all the ant moves towards the highest gradient step and the results are found. The complete algorithm is as follows:

Algorithm 8 NAS-RIF algorithm using Ant Colony Optimization

Set initial conditions, $u_k = [0, 1, 0]$, tolerance $\delta > 0$ and number of ants noa
 give randomly generated value to every ant {each ant holds a different value of
 the step size}

if $J(u_k) \geq \delta$ **then**

for every ant **do**

$f_k(x, y) = u_k(x, y) * g(x, y)$

calculate ∇J

$J(i) =$ calculate $J(u_k)$ {for every ant we get a corresponding J value}

end for

minimum = $\min J(u_k)$

find corresponding ant value {that is the corresponding to the minimum J
 value}

$phero_{k+1} = phero_k(1 - \rho) + \rho * \Delta phero_{minimum}$ {pheromone upgradation
 equation}

$u_{k+1} = u_k + phero * d_k$ {multiplying the new pheromone value to find next
 iteration u_k }

end if

5.2 Implementation and Results for Proposed NAS-RIF using ACO

5.2.1 *Out of Focus* Blur Images

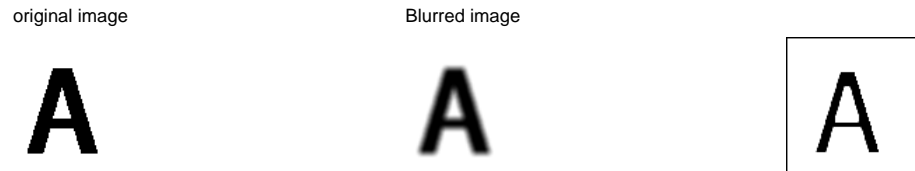


Figure 5.1: Original, Blurred and Restored Image for proposed Ant Colony Optimized NAS-RIF for *out of focus blur*

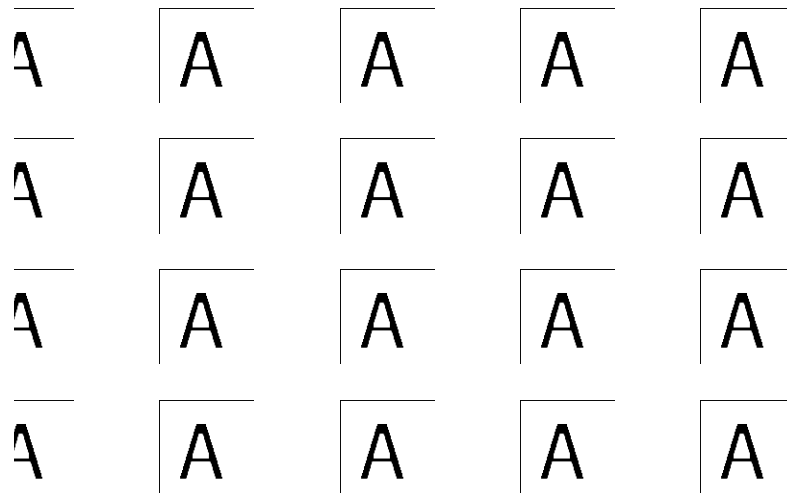


Figure 5.2: ACO working on the blurred Image 5.1 used for *NAS-RIF using ACO*

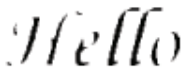
Figure 5.1 shows the original, blurred and restored image when NAS-RIF using Ant Colony Optimization was applied to out of focus blur. Figure 5.2 shows all the ants i.e. the intermediate result which were achieved by the new approach.



(a) Original Image



(b) Blurred Image



(c) Restored Image

Figure 5.3: NAS-RIF using Ant Colony Method

Figure 5.3 shows the results of NAS-RIF where optimization of the cost function was done using Ant Colony Optimization Method. The first Figure 5.3(a) shows the original figure on which the blurring filter is applied to give the Figure 5.3(b). During the course of Ant Colony Optimization Figure 5.4 shows the working of all the ants trying to deblur the image and Figure 5.3(c) shows the image corresponding to the least value of the cost function from all the ants.

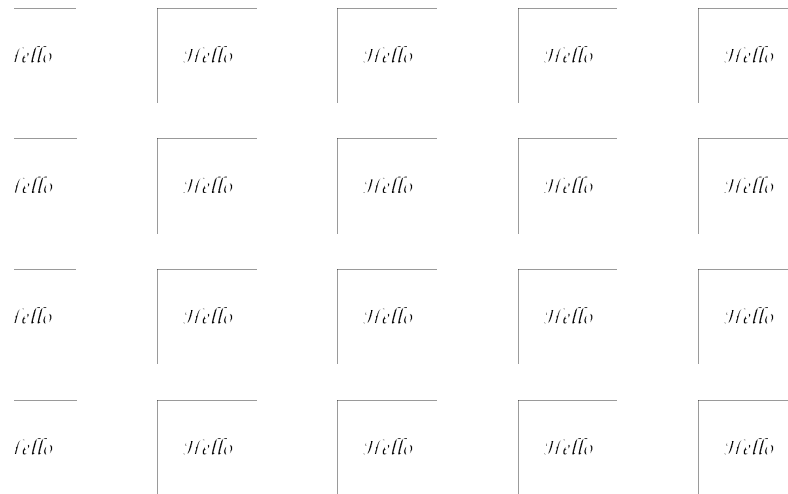


Figure 5.4: ACO working on the blurred Image 5.3(b) used for NAS-RIF

Chapter 6

Conclusion

NAR-RIF performed unsatisfactorily against motion blur, as is evident from the results obtained. But after a lot of trial and testing, it was found that NAS-RIF provides better results against *Out-Of-Focus* blur. The following were the testing conditions:

1. The background of the image was taken to be white: This helped remove the possibility of total black output.
2. The support size of the image was taken to be the smallest size of the rectangle which could encompass the true image.
3. The output was tested with filter sizes of both 3×3 as well as 5×5 : The difference was that the number of iterations to reach the goal state reduced exponentially with the increase in filter size.

The steepest descent algorithm as well as conjugate algorithm were implemented and studied for NAS-RIF. The steepest descent algorithm, when used for optimization, converged much slowly, and took many iterations before stopping. The conjugate gradient algorithm, when used for optimizing NAS-RIF, converged to result in much less iterations.

A novel implementation of NAS-RIF was proposed, using Ant Colony Optimization Algorithm. The novel approach was run for both *out of focus* blur and *motion blur*. The results that were obtained, were comparable to the results of NAS-RIF using steepest descent and conjugate gradient.

Future works include the following:

1. Improvement of optimization algorithm for NAS-RIF.
2. Implementation of a novel support finding algorithm to be used in case when the true support of the image is unknown.

Bibliography

- [1] Deepa Kundur. Blind Deconvolution of Still Images using Recursive Inverse Filtering. *University of Toronto, CS Dept.*, 22(1):23 – 33, January 1995.
- [2] Deepa Kundur. A Novel blind Deconvolution Scheme for Image Restoration using Recursive Inverse Filtering. *IEEE*, 46(2):375 – 390, February 1998.
- [3] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*. Pearson, 2nd edition.
- [4] WANG Ting BI Xiao-jun. Adaptive Blind Image Restoration Algorithm of Degraded Image. *IEEE Congress on Image and Signal Processing*, 1(5):536 – 541, September 2008.
- [5] Deepa Kundur and Dimitrios Hatzinakos. Blind Image Deconvolution. *IEEE Signal Processing Magazine*, pages 43 – 65, May 1996.
- [6] Method of Steepest Descent. http://en.wikipedia.org/wiki/Method_of_steepest_descent.
- [7] Lectures on steepest descent. http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-084JSpring2004/E0FFFC12-1D5D-4BA7-8A0B-3E9F6DC12A6E/0/lec5_steep_desce.pdf. Open Courseware MIT.
- [8] Non-Linear Conjugate Gradient Method. http://en.wikipedia.org/wiki/Nonlinear_conjugate_gradient_method.
- [9] Conjugate Gradient Method. http://en.wikipedia.org/wiki/Conjugate_gradient_method.
- [10] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, 1999.
- [11] James Kennedy, Russell C. Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann, April 2001.
- [12] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization-artificial ants as a computational intelligence technique. In *IEEE Computational Intelligence Magazine*, volume 1, pages 28–39, 2006.

-
- [13] Marco Dorigo, Vittorio Maniezzo, and Alberto Coloni. Ant system: Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-*, 26(1):29–41, February 1996.
- [14] L.M. Gambardella and M. Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis and S. Russell, editors, *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, pages 252–260. Morgan Kaufmann, 1995.
- [15] P. P. Grasse. *Les insectes dans leur univers*, 1946.
- [16] P. P. Grasse. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis* et *cubitermes* sp. la theorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.
- [17] Marco Dorigo. Ant colony optimization. http://www.scholarpedia.org/article/Ant_colony_optimization.
- [18] Lachlan Kuhn. *Ant Colony Optimization for Continuous Spaces*. PhD thesis, The University of Queensland, October 2002.
- [19] Thomas Stutzle and Holger Hoos. The max-min ant system and local search for the travelling salesman problem. In T. Bck, Z. Michalewicz, and X. Yao, editors, *IEEE International Conference on Evolutionary Computation*, pages 309–314, 1997.
- [20] J. Kennedy and R. ; Eberhart. Particle swarm optimization. In *Neural Networks, IEEE International Conference*, volume 4, pages 1942–1948, 1995.
- [21] Wenzhong Guo, Guolong Chen, Xiang Feng, and Lun Yu. Solving multi-criteria minimum spanning tree problem with discrete particle swarm optimization. In *Third International Conference on Natural Computation*, pages 471–478, 2007.
- [22] Fernanda Hembeker, Heitor S. Lopes, and Walter Godoy Jr. Particle swarm optimization for the multidimensional knapsack problem. In *Adaptive and Natural Computing Algorithms*, volume 4431/2007, pages 358–365. Springer, 2007.
- [23] Marco Dorigo, Marco A. Montes de Oca, and Andries Engelbrecht. Particle swarm optimization. http://www.scholarpedia.org/article/Particle_swarm_optimization.
- [24] Deepa Kundur. Blind Deconvolution of Still Images using Recursive Inverse Filtering. *University of Toronto, CS Dept.*, 22(1):23 – 33, January 1995.
- [25] Mansour Jamzad Mohsen Ebrahimi Moghaddam and Hamid Reza Mahini. Motion Blur Identification in Noisy Images Using Feed-Forward Back Propagation Neural Network. *Springer-Verlag*, page 369 – 376, 2006.