ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

SOLVING IMAGE PROCESSING PROBLEMS BY USING NONSTANDARD REGULARIZATION

M.Sc. THESIS

Tolga ACAR, B.Sc.

Date of Submission : June 13^{th}, 1994
Date of Approval : June 29^{th}, 1994

Examination Committee

Supervisor : Assoc.Prof.Dr. Muhittin GÖKMEN
Member : Prof.Dr. Erdal PANAYIRCI
Member : Prof.Dr. Ahmet H. KAYRAN

June 1994

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

STANDART OLMAYAN DÜZGÜNLEŞTİRME KULLANARAK GÖRÜNTÜ İŞLEME PROBLEMLERİNİN ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Müh. Tolga ACAR

Tezin Enstitüye Verildiği Tarih : 13 Haziran 1994
Tezin Savunulduğu Tarih : 29 Haziran 1994

Tez Danışmanı : Doç.Dr. Muhittin GÖKMEN
Diğer Jüri Üyeleri : Prof.Dr. Erdal PANAYIRCI
: Prof.Dr. Ahmet H. KAYRAN

Haziran 1994

PREFACE

This master thesis has grown out of some of my studies in the area of image processing and computer vision. Some of these studies are based on term projects of image processing and computer vision courses given in the post graduate level by Assoc. Prof. Muhittin Gökmen, for example the second chapter. Each chapter, in fact, is an expanded form of a conference paper even though some of them has not been published yet.

There should be only one person who caused all works in this thesis to born and bred and supported me to do things that couldn't have been placed on this thesis. The person who changed my mind in the direction of being a research assistant, who taught me how to write papers, how to present a speech and present a poster session, how to organize works and people, who taught me the first steps about what a workstation is and how to use it efficiently, and finally who gave me the feeling of what being a scientific person is, if I could. Apart from being my advisor and my working couple, Mr. Muhittin Gökmen, in fact, is the co-author of all these works presented in this thesis. Without him, there would be nothing. Here I thank him.

I am grateful to the research assistants Osman Aliefendioğlu and D. Turgay Altılar who have shared their time and knowledge with me in discussing some or other part of my work, even though the subjects are not necessarily in their major study areas. I have my thanks to my parents for supporting and encouraging me to complete my MS degree, and to my daughter for listening the keystrokes while I write the thesis during late night hours.

Last but not least, I would like to thank Özlem for her endless patience and understanding while I spent untold hours sticking onto the chair in front of the computer and being somewhat late for anything.

June, 1994

Tolga Acar
B.Sc. Control and Computer Eng.

CONTENTS

List of Figures

List of Tables

ABSTRACT

This thesis is mainly concerned with the surface reconstruction, edge detection, edge integration, restoration, image coding and compression problems of image processing and computer vision. In this thesis, a unified approach based on regularization theory has been applied in solving these problems.

Some of the edge detection and surface reconstruction methods based on non-standard regularization by using weak membrane modeling has been investigated and also compared to a convolution based method. In addition to edge detection and surface reconstruction, non-standard regularization has been expanded to solve problems in image coding and restoration Then a new multiscale edge integration method is developed where edges are obtained by another multiscale regularization based edge detection method called DORS. The energy functional related with membrane is modified to reach a predefined effect such as deblurring or surface reconstruction from sparse data, or the edges are used to obtain a form of coding and compression, or multiple reconstructed surfaces are used for edge detection.

In the first chapter, two different approaches to edge detection are qualitatively and quantitatively compared and thus possible relations with the other edge detection algorithms are tried to be figured out. The methods under consideration are adaptive smoothing which accomplishes detection of edges by nonlinear filtering and weak membrane modeling which is a non-standard regularization method. In the second chapter, restoration of an image blurred with a known blurring function is considered. By modifying the energy functional of weak membrane model, it is aimed to obtain a clear and noise free image by eliminating the blurring effect caused by misfocusing and the noise added onto this blurred image.

In the third chapter, a regularized approach to image coding is explained. In this approach, both coding part and the decoding part are based on the regularized solutions of the weak membrane modeling. In the fourth chapter, a multiscale edge representation using difference of regularized solutions and a multiscale edge integration scheme using weighted accumulation are presented.

ÖZET

STANDART OLMAYAN DÜZGÜNLEŞTİRME KULLANARAK GÖRÜNTÜ İŞLEME PROBLEMLERİNİN ÇÖZÜMÜ

Kelime olarak görü (*vision*) sözlük anlamında, "*görme yeteneği*" [1] olarak tanımlanmaktadır. İnsanların görme sistemi bu yeteneğe sahiptir ve ek olarak çevredeki pek çok karmaşık nesneyi anlamamıza ve düzenlememize olanak tanımaktadır. Bilgisayarla görmenin de, insanların dış dünyayı nasıl gördüğünü ve anladığını açıklayan insan görmesiyle çok yakından ilgisi vardır. Bu alanda çalışan araştırmacılar, en güçlü sistemleri kullanarak insan görme sisteminin benzetişimine çalışmakta, farklı yöntemlerle problemlere çözüm aramaktadır, ancak henüz hiç biri tam bir başarıya ulaşamamıştır. Bunun nedenleri arasında, bilgisayarların insan görme sistemine göre henüz ilkel olması, bilgisayarla görmedeki gelişmelerin henüz olgun bir aşamaya gelmemesi, insan görme sisteminin çok fazla karmaşık olması yer almaktadır. Neden her ne olursa olsun, bilgisayarla görme alanındaki araştırma çalışmaları diğer alanlardakilere göre uç noktada, işlem gücü gerektiren, matematik olarak karmaşık, yüksek teknoloji gerektiren ve çok geniş bilgisayar olanakları isteyen yapıda olmuştur.

Üç boyutlu dünyanın iki boyuttaki izdüşümleri, çevremizdeki üç boyutlu dünyanın karmaşıklığı hakkında bize az miktarda bilgi vermektedir. Herhangi bir görü sistemi, gerçek üç boyutlu bir nesnenin bir veya birden fazla ardışıl resim çerçevesini kullanarak, o nesnenin öz niteliklerini çıkartabilmelidir. Tanıma söz konusu olduğunda bu nitelikler, nesne sınırlarını bularak veya bütün bir görüntüyü bölütlendirmeyle daha küçük parçalara ayırarak, her nesneyi diğerinden ayırmada kullanılabilmektedir.

Bölge (*region*), bütün beneklerin yanındakilerle bitişik olduğu bağlantılı benekler kümesi olarak tanımlanmaktadır. Görüntü bölütlendirme (*segmentation*)

[1] "Longman Dictionary of Contemporary English," Longman Group Ltd. 1978

ise, sayısal bir resmin anlamlı ve birbiriyle örtüşmeyecek biçimde bölgelere parçalanması anlamına gelmektedir. Bölütlendirme, temel olarak iki değişik yaklaşımla gerçeklenmektedir. Bölge temelli yaklaşımda, belli nesnelere veya bölgelere aynı benek değerleri atanmaktadır. Ayrıt temelli yaklaşımda ise, bölgeler arasındaki sınırların bulunmasına çalışılmaktadır.

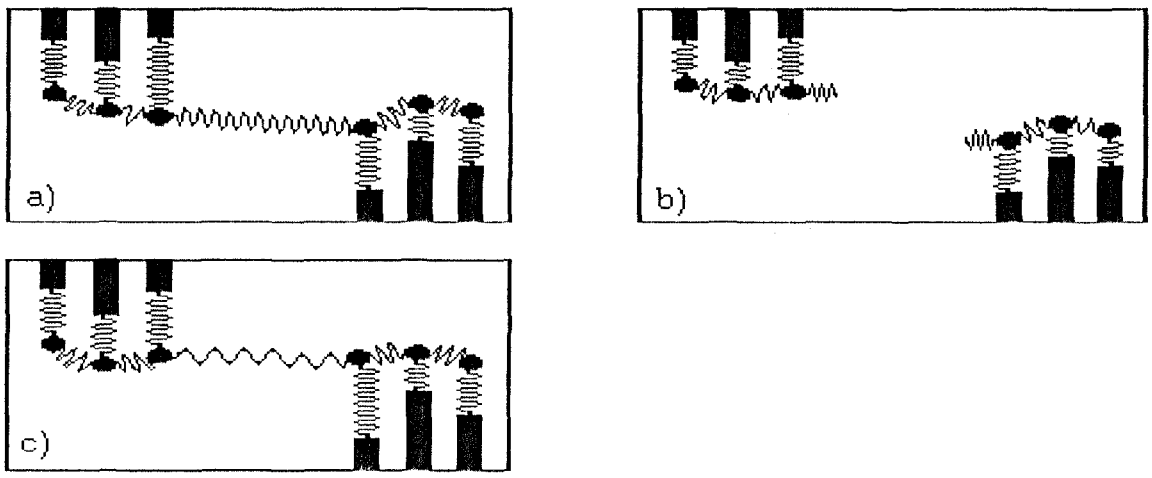
Ayrıt, yüzey yapısındaki değişiklik, aydınlatma veya görünen yüzeylerin izleyiciden uzaklığı gibi nedenlerden ortaya çıkan, nesnelerin fiziksel değişikliklerinin görüntüye de yansımasından oluşan sınırlar veya hatlar olarak tanımlanmaktadır. Nesne tanıma gibi üst düzey işlemlerin başarımı, ayrıtların tam ve doğruluğuna bağlı olduğu için, ayrıt saptama bilgisayarla görmenin esas araştırma konularından birisini oluşturmaktadır. Nesnelerin fiziksel yapılarından kaynaklanan ve görüntüye yoğunluk (*intensity*) değişimleri olarak yansıyan ayrıtların saptanmasının amacı, seyrek olmasının yanı sıra nesneler hakkında tam ve anlamlı tanımlar elde etmektir. Buna bağlı olarak ileriki bölümlerde, ayrıt saptamada göz önüne alınması gereken kriterler sunulmakta, iyi bir ayrıt saptayıcıdan beklenen özellikler verilmektedir.

Ayrıt saptama yöntemleri, evriştirmeye (*convolution*) ve enerji azaltılmasına dayanan algoritmalar olarak da ikiye ayrılmaktadır [2]. Bu tez içinde her iki yönteme de dayanan ayrıt saptayıcılar ele alınmıştır. İkinci bölümde ele alınan iki ayrıt saptayıcıdan birisi olan uyarlanır düzleme [1] katlamaya dayanmaktadır. Düzenleme teorisine dayanan zar modellemesi ve bunun yırtılabilir [2] hali ise enerji indirgenmesiyle gerçeklenmektedir. Ayrıt saptamanın üzerine inşa edilen kodlamada enerji en aza indirgenmesi, seyrek veri olması durumunda gerçeklenmektedir. Resimdeki süreksizlik noktalarından faydalanan bulanıklık giderme yönteminde ise en aza indirilecek enerji fonksiyoneli içinde evriştirme içeren bir fonksiyon yer almaktadır.

Düzgünleştirme teorisi içinde yer alan zar modeli, bu çalışma içinde ayrıt saptama, yüzey kurma, kodlama, netleştirme, yeni bir çok ölçekli gösterilim elde etme amacıyla kullanılmıştır. Bu tezde, düzgünleştirme teorisine dayanan tümleşik bir yaklaşım, bu problemlerin çözümünde kullanılmıştır. Özellikle yırtılabilir zarın kullanılmasıyla gerçekleştirilen ayrıt saptamadan yola çıkılarak geliştirilen kodlama ve netleştirme gibi diğer yöntemler, düzgünleştirme teorisinin kullanım alanını genişletmektedir. Böylece görüntü işleme alanındaki pek çok probleme aynı teorinin kullanılmasıyla çözüm getirilmektedir.

Burada incelenen ve geliştirilen yöntemlerde yırtılabilir zar modeli kullanılmıştır. Bu modele ilişkin enerji fonksiyoneli dışbükeydir. Yani fonksiyonelin birden fazla en az durumu vardır, ancak bunlardan ancak bir tanesi bütünsel en

[2]Ayrıt saptamadan önce yapılan düzleme işlemleri de, saptayıcıların önemli bir özelliğidir. Örneğin iki ayrıt saptama yönteminin karşılaştırılmasının yapıldığı bölümde (sayfa 7) resim düzleme yöntemleri, alçak geçiren süzgeçlerin kullanılması, gürültülü resme yerel olarak eğriler uydurma ve bütünsel eğri uydurma olmak üzere üç ana bölüme ayrılmaktadır.



Şekil 1. Dışbükeylik. (a) ve (b) durumları kararlı durumlardır. Ancak ortadaki (c) durumu diğerlerine göre daha yüksek enerjiye sahiptir.

azdır, diğerleri yerel en az durumlardır. Sistem bu tür yerel en az durumlarından birisine takıldığında, dışarıdan yeterince büyük bir etkinin gelmesiyle diğer bir en az duruma geçebilmektedir. Bu durum Şekil 1 ile de gösterilebilir. Bu durumda yay, deformasyon veya kırılma noktasını geçecek şekilde uzatılmıştır. Kırılma noktasını geçen yay da kararlı olmasına rağmen yapısal olarak ilk haldeki yaydan farklı bir yapıdadır ve her iki hal farklı en az durumlarını temsil etmektedir. Buradaki yayın da birden fazla en az durumu vardır.

Bu nedenle, yırtılabilir zar modellemesinde elde edilen enerji fonksiyonelinin en aza indirilmesinde standart en iyileme yöntemleri kullanılamamaktadır. Rassal yöntemlerin kullanımı her ne kadar olası ise de, bu tip yöntemler çok fazla işlem yükü gerektirmektedir [3]. Bu çalışmada standart olmayan nedensel bir yöntem olan aşamalı dış bükeylik algoritması kullanılmaktadır.

Düzgünleştirme teorisindeki yırtılabilir zar modelinde ayrıtları saptanacak resmin, önce parça parça düzlenmiş bir fonksiyonla modellemesi yapılmaktadır. İki boyutta yırtılabilir zar ile modellenen resim aşağıdaki enerji fonksiyoneliyle ifade edilmektedir [2]:

$$
\begin{aligned}
E \;=\; & \sum_i \sum_j (u_{i,j} - d_{i,j})^2 + \\
& \lambda^2 \sum_i \sum_j (u_{i,j} - u_{i-1,j})^2 (1 - m_{i,j}) + (u_{i,j} - u_{i,j-1})^2 (1 - \ell_{i,j}) + \\
& \alpha \sum_i \sum_j (\ell_{i,j} + m_{i,j})
\end{aligned}
\tag{1}
$$

Burada d giriş resmi, u oluşturulan resim, ℓ ve m sadece 0 veya 1 değeri alabilen çizgi işlev değişkenleri, λ ve α da pozitif değerli iki parametredir. Oluşturulan resim u parça parça düzlenmiştir. Zara yırtılabilir özelliğini ***çizgi işlevi*** adı verilen mantıksal değişkenler vermektedir. Çözüm olarak bulunacak u fonksiyonu, bu enerji fonksiyonelinin en aza indirilmesiyle elde edilmektedir. Ancak bu fonksiyonelin içbükeylik (*convexity*) özelliği yoktur [2, 3]. Bu nedenle standart olmayan bir en aza indirme yöntemiyle çözüm yoluna gidilmektedir.

Aşamalı dış bükeylik (*Graduated Non-Convexity*) adı verilen ve gerekimci olan bu yöntem temel olarak, dışbükey bir enerji fonksiyonelinin bir parametreye bağlı olarak içbükeyliğinin kontrol edilmesine dayanmaktadır. Bu parametreyle enerji fonksiyoneli değiştirilmekte ve en aza indirme değiştirilmiş fonksiyonel üzerinde gerçeklenmektedir. Yırtılabilir zar modellemesinden ortaya çıkan dışbükey enerji fonksiyonelinin bu şekilde değiştirilmesi Blake ve Zisserman [2] tarafından önerilmiştir. Değiştirilmiş enerji fonksiyoneli şu şekilde verilmektedir:

$$E^{(p)} = \sum_i \sum_j (u_{i,j} - d_{i,j})^2 + \sum_i \sum_j g^{(p)}_{\alpha,\lambda}(u_{i,j} - u_{i-1,j}) + \sum_i \sum_j g^{(p)}_{\alpha,\lambda}(u_{i,j} - u_{i,j-1}) \tag{2}$$

ve

$$g^{(p)}_{\alpha,\lambda} = \begin{cases} \lambda^2 t^2 & |t| < q \\ \alpha - c(|t| - r)^2/2 & q \le |t| < r \\ \alpha & |t| \ge r. \end{cases} \tag{3}$$

Burada $c = c^*/p$, $r^2 = \alpha(2/c + 1/\lambda^2)$, $q = \alpha/(\lambda^2 r)$ olarak verilmektedir [2, 3]. $g^{(p)}_{\alpha,\lambda}$ fonksiyonuna, yerel etkileşim fonksiyonu adı verilmektedir. Ortadan kaldırılmış çizgi işlevleri,

$$\ell(t) = \begin{cases} 1 & |t| > \sqrt{\alpha}/\lambda \\ 0 & \text{diğer hallerde.} \end{cases} \tag{4}$$

ile tekrar elde edilebilmektedir.

Yırtılabilir zar modelinde kullanılan değiştirilmiş fonksiyonel, sadece bu modele özgüdür ve farklı türdeki enerji fonksiyonelleri için çalışması söz konusu değildir. Daha başka dışbükey enerji fonksiyonellerinin de bu yöntemle en aza indirilmesi gerektiğinde, burada kullanılan yönteme benzer biçimde, enerji fonksiyonelinde dışbükeyliğe neden olan kısımın değiştirilmesi gerekmektedir.

İkinci bölümde, ayrıt saptama amacıyla geliştirilen iki değişik yaklaşım niceliksel ve niteliksel olarak karşılaştırılmakta ve böylece daha önce bu şekilde karşılaştırılan değişik ayrıt saptama yöntemleriyle ilişkileri ortaya çıkarılmaya çalışılmaktadır. Ele alınan yöntemler, ayrıt saptamayı süzgeçleme kullanarak gerçekleştiren uyarlanır düzleme (*adaptive smoothing*) [1] yöntemi ile, bunu yırtılabilir zar (*weak membrane*) [2] modeliyle gerçekleştiren en azlama yöntemidir. Ayrıt saptamada karşılaşılan problemlerin görülebileceği yapay resimler üretilmiş (dama tahtası ve çubuklar), bu resimlere gürültü eklenmiş ve her iki algoritmanın başarımı bu resimler üzerinde incelenmiştir. Algoritmaların ayrıt saptama başarımlarının karşılaştırılmasında görsel değerlendirmenin yanısıra, FoM, P(IE/AE), P(AE/IE) [4] gibi sayısal ölçütler de kullanılmıştır. Bunun yanısıra her iki yöntem gerçek resimler üzerinde de denenmiş, sonuçlar görsel olarak yorumlanmıştır. Elde edilen sonuçlardan, karşılaştırılan iki yöntemden yırtılabilir zar yönteminin, uyarlanır düzlemeye göre, ayrıt saptama açısından daha iyi sonuçlar verdiği gözlenmiştir.

Üçüncü bölümde, resmin bulanıklaşma nedeninin bilinmesi halinde, bulanık resmin netleştirilerek onarılması problemi ele alınmaktadır [5]. Odaklama hatalarından kaynaklanan bulanıklaşmanın ve bunun üzerine eklenen gürültünün ortadan kaldırılarak, net ve gürültüsüz resmin elde edilmesi amaçlanmaktadır. Düzgünleştirme yaklaşımı yardımıyla yüzey kurarak netleşmiş resim elde edilmeye çalışılmaktadır. Düzenlemede, yırtılabilir zar (*weak membrane*) modelinden ortaya çıkan dışbükey enerji fonksiyonelinin, aşamalı dışbükeylik (*Graduated Non-Convexity*) algoritmasıyla çözümü gerçeklenmektedir. Bulanıklık giderilmesi amacıyla değiştirilmiş zara ilişkin değiştirilmiş enerji fonksiyoneli

$$
\begin{aligned}
E \;=\; & \sum_i \sum_j ((f * b)_{i,j} - d_{i,j})^2 + \\
& \lambda^2 \sum_i \sum_j (f_{i,j} - f_{i-1,j})^2 (1 - m_{i,j}) + (f_{i,j} - f_{i,j-1})^2 (1 - \ell_{i,j}) + \\
& \alpha \sum_i \sum_j (\ell_{i,j} + m_{i,j}).
\end{aligned}
\tag{5}
$$

şeklinde ifade edilmektedir. Burada d bozuk resim, f netleştirilmiş resim, b bulanıklaştırma süzgeci, m ve ℓ çizgi işlevleri. λ ve α da yırtılabilir zara ilişkin parametrelerdir. Netleşmiş resim, E fonksiyonelinin en azlanması sonucunda elde edilmektedir. Böylece resimdeki süreksizlik noktalarının korunmasının yanısıra, bulanıklaşmış olan süreksizliklerin de ortaya çıkarılması sağlanmaya çalışılmıştır. Yöntem, ortalama alan süzgeçle bulanıklaştırılan ve üzerine gürültü eklenen yapay ve gerçek resimler üzerinde uygulanmıştır. Elde edilen onarılmış resimlerde süreksizliklerin baskınlaştırılmasıyla bulanıklığın ortadan kalktığı gözlenmiştir.

Dördüncü bölümde, düzgünleştirme teorisine dayanan bir görüntü kodlama algoritması tanıtılmaktadır. Tanıtılan kodlama algoritması kayıplı bir kodlama yöntemidir. Bu yaklaşımda kodlama bölümü, yırtılabilir zar modelinden ortaya

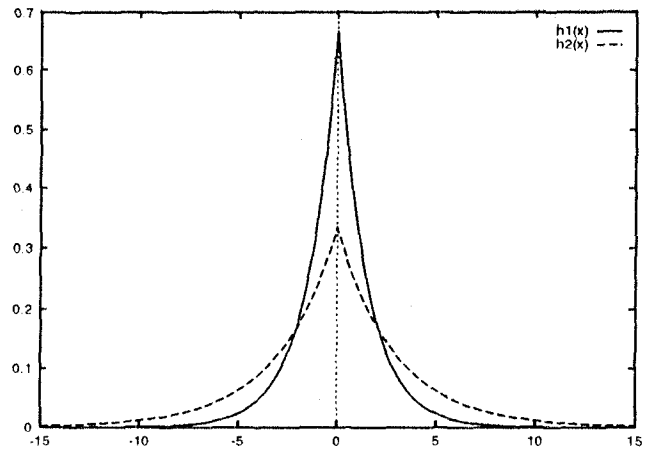
çıkan dışbükey enerji fonksiyonelinin en aza indirilmesine dayanmaktadır. Kod çözme aşamasında ise ayrıtlar dışında kurulan zar modellemesinden ortaya çıkan içbükey enerji fonksiyonelinin en aza indirilerek, seyrek veriden yüzey kurma yöntemi kullanılmaktadır.

Bu bölümde iki aşamalı bir kodlama yöntemi tanıtılmaktadır. Birinci aşamada süreksizlikleri bulmak için yırtılabilir zar yöntemi kullanılarak çizgi işlevleri (*line processes*) bulunmakta, ikinci bölümde ise bu çizgi işlevlerinin mantıksal değerler almasından yola çıkılarak seyirtim uzunluğu (*run-length*) yöntemiyle kodlanmaktadır. Kod çözme için gerekli olan süreksizlik etrafındaki veriler de entropi kodlamasıyla kodlanmaktadır. Kod çözücü bölümde ise öncelikle çizgi işlevleriyle süreksizlik noktalarının yerleri tespit edilmektedir. Daha sonra yerleri bulunan kodlanmış verilerin kodları çözülerek yerlerine yerleştirilmiştir. Bundan sonra yırtılmış zar modeli kullanılarak parça parça sürekli bir enerji fonksiyonelinin en aza indirilmesiyle yüzey kurularak resim elde edilmiştir. Parça parça sürekliliğin burada kullanılması, süreksizlik olmayan yerlerde zar modellemesinin kullanılması şeklinde ortaya çıkmakta, bu tür bölgeler içindeki zarın enerji fonksiyoneli içbükey özellik gösterdiğinden, en aza indirilmesinde zorluk olmamaktadır.

Burada tanıtılan kodlama yöntemi sonucunda, çizgi işlevleri ve bunlarla belirlenen süreksizlik noktaları etrafındaki benek değerleriyle kodlanan resmin, boyut olarak giriş resminden küçük olması, söz konusu yöntemin resim sıkıştırma amacıyla da kullanılabileceğini ortaya çıkarmaktadır. Bu çalışmada yöntem gerçek ve yapay resimler üzerinde denenmiş, farklı resimlerdeki değişik sıkıştırma oranları elde edilmiş ve elde edilen resimlerdeki bozulmalar da nicel ve nitel olarak belirlenmiştir. Elde edilen sıkıştırmada ortalama olarak 5:1 oranında sıkıştırma elde edilmiştir. Bununla birlikte, ayrıtlar çevresindeki verilerin çok daha etkin biçimde kodlanabileceği görülmektedir.

Beşinci bölümde, Gökmen tarafından tanıtılan düzgünleştirmenin kullanıldığı çok ölçekli bir ayrıt gösterilimi ve çok ölçekli bir ayrıt birleştirme algoritması incelenmektedir [6, 7]. *DORS* (*Difference Of Regularized Solutions*) adı verilen ve her seviyesi, farklı düzgünleştirme parametresiyle elde edilen iki düzgünleştirilmiş çözüm arasındaki farkın alınmasıyla elde edilen gösterilim ele alınmaktadır. Her bir düzgünleştirilmiş çözüm, zar fonksiyoneliyle ifade edilen bir enerji fonksiyonelinin en aza indirilmesiyle elde edilmektedir. Bir boyutlu durumda, *DORS* gösterilimi, resmin Vietnam şapkası operatörü adı verilen bir süzgeçle katlanmasıyla da elde edilmektedir [6]. Bir boyutta yay fonksiyoneline karşı düşen, $\lambda_1 = 1$ ve $\lambda_2 = 3$ ile elde edilmiş iki R-süzgeci, Şekil 2'de gösterilmiştir. Yay fonksiyoneline karşı düşen iki R-süzgecinin farkının alınmasıyla elde edilen bir boyuttaki *DORS* süzgeci Şekil 3'de gösterilmektedir.

Ayrıtların, bu gösterilimde farklı ölçeklerdeki davranışlarının incelenmesiyle, daha önce bir boyutlu durum için geliştirilmiş olan çok ölçekli bir birleştirme algoritması iki boyutlu durum için geliştirilmiştir [6]. Ağırlıklı toplama yönteminin kullanıldığı algoritma, ölçek uzayında karşılaşılan ayrıtların kayması, ortaya çık-



Şekil 2. İki R-süzgeci. $h1(x)$ için $\lambda = 1$, $h2(x)$ için $\lambda = 3$ olarak alınmıştır.

ması, ortadan kaybolması ve dallanması problemlerine çözüm getirmektedir.

Zar modellemesiyle elde edilecek çözüm şu şekilde ifade edilmektedir [7]:

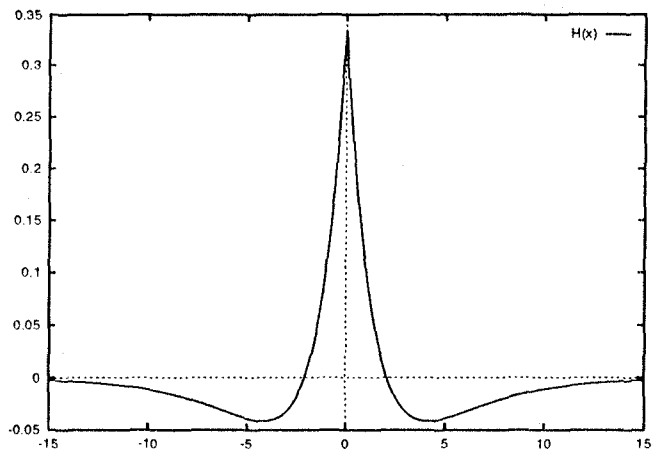
$$\begin{aligned} v(x,y;\lambda_i) &= \{v(x,y) : E_m(f,\lambda_i) = \\ &\quad \inf_{f\in V} \int\int_\Omega (f-d)^2\,dxdy + \lambda_i \int\int_\Omega (f_x^2 + f_y^2)\,dxdy\} \end{aligned} \tag{6}$$

burada f giriş resmi, v modellemeyle elde edilen çözüm, λ düzgünleştirme parametresi, f_x ve f_y de giriş resmi f'nin sırasıyla x ve y'ye göre birinci mertebe türevleridir. Farklı λ düzgünleştirme parametreleri ile elde edilmiş resimlerden *DORS* gösterilimi

$$DORS(x,y;\lambda_i,\lambda_j) = v(x,y;\lambda_i) - v(x,y;\lambda_j), \qquad \lambda_i \neq \lambda_j. \tag{7}$$

şeklinde elde edilmektedir. λ_i ve λ_j birbirinden farklı iki düzgünleştirme parametresi ve $v(x,y,\lambda)$ de λ ile elde edilmiş zar çözümüdür. Şekil 3'de, bir boyutlu durumda *DORS* gösteriliminin, resmin Vietnam şapkası operatörü adı verilen bir süzgeçle katlanmasıyla da elde edildiği gösterilmektedir.

Bu gösterilimde, ayrıtların farklı ölçeklerdeki davranışlarının incelenmesinden yola çıkılarak, çok ölçekli bir birleştirme algoritması geliştirilmiştir. Birleştirmede, çok ölçekli ayrıt resimleri giriş olarak kullanılmaktadır. Giriş olarak alınan ayrıt resimleri en kabadan en ince ölçeğe doğru sırayla kullanılmaktadır. Birleştirme en kaba ölçekten başlamaktadır. Kaba ölçekteki her ayrıt noktası



Şekil 3. $\lambda_1 = 1$ ve $\lambda_2 = 3$ alınarak elde edilen iki R-süzgecinin kullanılmasıyla elde edilmiş Vietnam şapkası operatörü.

için ince ölçekte 3×3 genişliğindeki pencere içinde ayrıt noktası aranmaktadır. Ayrıt bulunması halinde, kaba ölçekteki ayrıt noktası ince ölçekte bulunan noktaya, toplama dizisi yardımıyla bağlanmaktadır. Her bağlama işlemi, bağlanan noktaya karşı düşen toplama dizisi değerinin uygun şekilde artırılmasıyla gerçeklenmektedir.

Bulunulan ölçeğe göre, ince ölçekte ayrıt bulunmaması durumunda kaba ölçekteki ayrıt noktası ince ölçeğe doğru devam ettirilebilmektedir. Bunun tersi olarak kaba ölçekte ayrıt olmayan yerlerde, ince ölçekte ayrıt olması durumunda, yeni ayrıt ortaya çıkmasına izin verilebilmektedir. Giriş olarak alınan ayrıt resimleri dizisindeki ardışıl her resim için bu işlem tekrarlanmaktadır. Bütün ayrıt resimleri işlendikten sonra, belli bir değere gelmiş toplama dizisindeki yerler ayrıt olarak atanmaktadır.

Ağırlıklı toplama kullanılmasıyla gerçeklenen birleştirme algoritması, ölçek uzayında karşılaşılan ayrıtların kayması, ortaya çıkması, ortadan kaybolması ve dallanması problemlerine çözüm getirmektedir. Bu haliyle birleştirme algoritması, çok ölçekli diğer ayrıt saptayıcılarla elde edilmiş ayrıt resimlerine de uygulanabilecek durumdadır. Birleştirme algoritmasının başarımı niceliksel ve niteliksel olarak yapay ve gerçek resimler için çıkartılmıştır.

Sonuç olarak bu çalışma içinde, düzgünleştirmedeki yırtılabilir zar modelinden yola çıkılarak görüntü işleme ve bilgisayarla görme alanındaki ayrıt saptama, yüzey kurma, netleştirme, kodlama ve sıkıştırma, çok ölçekli ayrıt elde etme ve birleştirme problemlerine, aynı yaklaşımlar kullanılarak, çözüm getirilmiştir.

CHAPTER 1.

INTRODUCTION

Vision as a word has a dictionary meaning of "*(the) ability to see*" [1]. Human visual system has this ability and furthermore, allows us to organize and understand the many complex elements of the environment. Computer vision is closely tied to human vision which explains how humans see and perceive the outside world. One approach to solve these problems in this field is to simulate the human vision system which can be realized by using the most powerful machines, but a complete simulation has not been achieved yet. The reasons of this may be that the computers are too primitive, or the developments in the area of computer vision have not reached a mature degree, or the over-complex structure of the human visual system. But the major difficulty is that the solution of vision problems is not unique. Thus the ill-posed nature of these problems is one of the most important obstacles in achieving a complete solution. Whatever the reason is, the research studies in computer vision have been in the leading extreme edge, power consuming, mathematically complex, requiring very high technology and starving for vast computer resources compared to others.

Two dimensional projections of the three dimensional world gives us comparatively small amount of information about the 3-D world around us. This small information is not sufficient to encode the full complexity of a natural 3-D scene. Any vision system must be able to extract the features of a real 3-D object from a single or multiple consequent frames of a scene. In terms of recognition, these features may be used to separate each object from the other one by detecting the

[1] "Longman Dictionary of Contemporary English," Longman Group Ltd. 1978

boundaries or segmenting the whole image into smaller components.

Region is defined as a connected set of pixels in which all of the pixels are adjacent or touching to each other. Image segmentation is the process of partitioning a digital image into meaningful and disjoint (non-overlapping) regions. There are basically two different approaches to segmentation: In the region based approach, the pixels are assigned to particular objects or regions. The boundary (edge) based approach attempts to locate the boundaries between regions.

The edge is a boundary at which a significant change occurs in image intensities. Edge detection is one of the the main research areas of computer vision and image processing since the performance of higher level processes such as object recognition rely heavily on the correctness and completeness of detected edges. The main goal in edge detection is to obtain a complete and meaningful information apart from being sparse. The edges arise from the physical properties of the objects and manifest themselves as intensity changes in the image. Related to this definition, the criteria in edge detection are presented and important properties of a good edge detector are given in the foregoing chapters.

The edge detection methods may be divided into two major classes where one is based on convolution and the other one is based on energy minimization [2]. The investigated methods in this thesis fall into two of the classes. The adaptive smoothing relies on convolution and the weak membrane modeling is based on regularization which also relies on energy minimization.

Weak membrane modeling which is based on regularization is used for edge detection and surface reconstruction, for image coding and restoration, and to obtain a new multiscale image representation. The image coding scheme is based on edge detection which is performed by using weak membrane modeling. The encoding part of this scheme is a regular membrane modeling of regularization theory where sparse data is used as input. Image restoration method is based on a slightly modified energy functional of the weak membrane model. Thus, image restoration is also performed by using the regularization theory where the

[2]Noise smoothing operation performed before edge detection is also an important part of an edge detector. In chapter 2, the noise smoothing operations are classified as *i.* using low-pass filters, *ii.* fitting local curves to noisy image and *iii.* fitting a global curve to image.

solution is obtained by a non-standard algorithm.

The energy functional emerging from weak membrane modeling lacks the mathematical property of convexity. The modeling is realized by finding the minimum of the derived energy functional related to weak membrane. The weak membrane energy functional has many local minima and only one of them is the global minimum. A non-standard method is used in minimizing this functional, hence there exist more than one solution of the functional if a regular optimization method is used. A standard minimization algorithm may easily stick to one of those local minima, thus causing the solution to be the wrong one. Even though there exist stochastic optimization methods to find the minimum of such non-convex functionals, the computation required is very high in such algorithms. The method used in this work is graduated non-convexity algorithm (GNC) which is a deterministic algorithm.

In the second chapter, adaptive smoothing and weak membrane modeling is compared for edge detection performance. Synthetic images are produced and two methods are used to detect edge maps of these images for comparison. Numerical values are calculated for quantitative comparison, such as FoM, P(IE/AE), P(AE/IE) in addition to qualitative results. Real images are also used and edges of these images are obtained for qualitative evaluation of edge detection performance. It is observed that weak membrane modeling has superior edge detection performance than adaptive smoothing method.

A regularized approach to image restoration is derived in the third chapter where energy functional related to weak membrane is used. It is aimed to reduce the blurring caused from misfocusing and eliminate the noise added onto blurred images. In this approach, it is assumed that the blurring effect can be modeled by an appropriate filter. Even though there is no restriction on the size and coefficient values of the degrading filter, 3x3 averaging filter is used to obtain the images presented in this chapter.

A different approach to image coding is presented in the fourth chapter where the coding part is based on edge detection and the decoding part is based on surface reconstruction methods. Since regularization combines edge detection and surface reconstruction in one energy functional, the same regularization functional

is used in the coding and the decoding part.

In the fifth chapter, a multiscale image representation based on difference of two regularized solutions is investigated together with a multiscale edge integration method. In difference of regularized solutions (DORS) representation, two images are obtained by solving the membrane functional with two different regularization parameters [6, 7]. Then the edge image is obtained by finding the zero crossings on this DORS image. By using multiple couples of regularization parameters to obtain multiple DORS images, multiscale edges can be obtained. Then a multiscale edge integration algorithm which is based on weighted accumulation and presented in [6] for one dimension is expanded to two dimensions which can use the edge maps obtained by DORS. The integration algorithm tracks down the edge points starting from the coarsest level down to the finest level by updating an accumulation array. The edge tracking operation can be reduced to finding the appropriate correspondence between two consequent edge images in scale space and updating the accumulation array by using this relation. The integration method allows new edge points to appear by also preventing the old ones to survive and permits branching of one edge point into more edge points in scale space.

In this thesis, image processing problems are integrated under a unified framework based on regularization theory. By using the weak membrane model of regularization, edge detection and surface reconstruction, image restoration, compression, multiscale image representation and multiscale edge integration problems are investigated.

CHAPTER 2.

COMPARISON OF EDGE DETECTION ALGORITHMS BASED ON ADAPTIVE SMOOTHING AND WEAK MEMBRANE MODELING

2.1 Introduction

In this chapter, two different approaches to edge detection are qualitatively and quantitatively compared and thus possible relations with the other edge detection algorithms are tried to be figured out. The methods under consideration are adaptive smoothing which accomplishes detection of edges by nonlinear filtering and weak membrane modelling which performs this process by minimizing a non-convex energy functional. The algorithms are tested on both real and synthetic images. The synthetic images (noisy checkerboard and bar images) are created to reveal the detection and localization performance of the algorithms. In addition to the visual comparison of results, the performances of the algorithms are quantitatively compared by using quantitative measures such as FoM, P(IE/AE) and P(AE/IE). Based on these results, it's concluded that the weak membrane modelling has a superior edge detection performance as compared to the adaptive smoothing method.

2.1.1 Boundary Detection

Detection of boundaries from images which are two dimensional projections of physical objects is one of the most important subjects of computer vision. Edge detection is one of the main research areas of computer vision, since performance of higher level processes such as object recognition relies heavily on the correctness and completeness of edges. It is the main goal of edge detection to obtain complete and meaningful information, while being sparse, where edges resulting from physical structure of objects appear as sharp intensity changes in their images. In this chapter, qualitative and quantitative comparison of edge detection methods realized by two different approaches is evaluated.

An edge detection algorithm is expected to satisfy the following criteria when applied to a noisy image [8]:

c1. Good detection: True edge points must be marked by the detector and probability of falsely marking non-edge points must be reduced.

c2. Good localization: Location of detected edges should be as close as possible to true edges.

c3. Robustness: The algorithm should be robust to noise and perform well for various images.

c4. Efficiency: The implementation of the algorithm should lead itself to be an efficient one. Parallel form of the algorithm with small neighbour interactions improves the efficiency.

c5. Applicability to sparse data: The algorithm should reach a reasonable performance when applied to sparse data which allows usage of depth data.

Main problems of edge detection are noise produced by imaging and sampling, tradeoff between detection and localization, and multiscale behaviour of information obtained from the image. Detection of sharp changes in image intensities requires various derivatives of pixel values on the noisy image. But taking derivatives on a noisy data amplifies noise, thus causing instability. For this reason, many edge detection algorithms perform some kind of noise smoothing method before taking derivatives. Noise smoothing generally is achieved by one of the

following methods [6]:

i. Smoothing with a low pass filter,
ii. fitting edge templates or curves locally to noisy data,
iii. fitting a global curve to data.

After the smoothing process, some method for detection of discontinuities is used to extract boundary information from the image. The most of the edge detection methods use the first or the second order derivatives, but the processing of the derivative differs in each method. In the following subsection, a brief overview of the main edge detection algorithms is presented.

2.1.2 Overview of Edge Detection Methods

Edge detector introduced by Marr and Hildreth [9] smoothes the image by convolving the image with the Gaussian filter where the smoothing is controlled by the standard deviation of Gaussian, σ. In this model, zero crossings of laplacian of the image is assigned as edge pixels. Canny's edge detector also uses Gaussian smoothing [10]. Canny has shown that the first derivative of the Gaussian is a very good approximation to optimal edge detector by formulating the first *c1* and the second *c2* criteria of edge detection. Haralick's step edge detector falls into the second category (*ii*) [11]: A part of the surface is modelled by the interpolation of intensity values with the Chebychev polynomials up to the third degree. Then by using the coefficients of this function, zero crossings of directional second derivative in the direction of the gradient is calculated. And finally the pixel is assigned as an edge point if a zero crossing exists in the neighbourhood of this pixel and the slope of the zero crossing is negative.

Algorithms based on regularization falls into the third category. This sort of algorithms is based on minimization of an energy functional including one or more smoothness constraints [3, 4, 6]. Weak string and membrane models used in this chapter are introduced by Blake and Zisserman [2]. These models use regularization with the first order derivatives as the smoothness constraint, and use ***line processes*** to preserve discontinuities. This non-standard smoothness constraint

with line processes is introduced by Geman and Geman [12]. Terzopoulos [13] has integrated the first and the second order derivatives in one energy functional for discontinuity preserving surface reconstruction. In these type of algorithms, the system is assumed to have high energy at discontinuities, and this energy is minimized by breaking the string or tearing the membrane at sufficiently large discontinuous locations. This kind of energy functional is non-convex and requires special minimization algorithms such as simulated annealing or graduated non-convexity [2]. Gökmen [6, 8] has developed an edge detection scheme based on regularization theory in which the smoothness is controlled spatially over the image space.

Another major difficulty in edge detection is the tradeoff between localization and detection criteria. Correct localization of an edge requires a small neighbourhood to be processed around this pixel. The detection criteria requires that noise should be smoothed and noisy edges should not appear in the resultant edge map. Thus the smoothing of noise requires a large support such as convolving the image with a Gaussian filter with large standard deviation. Another important difficulty is the multisize nature of images. The images are composed of different sized objects with different properties and different distances to the camera. The two dimensional projections of such variety of objects result in an image with spatially changing signal-to-noise ratio and intensity values. Hence images obtained are not homogeneous in itself, as well as not homogeneous in different frames. Detection of edges from such images requires multiscale filters or multiscale methods to be used. But linear edge operators cannot cope with the multisize features since they behave equal in each spatial location in the image. To overcome these problems, three methods are usually used to detect multisized edges [6]:

1. Specify the best scale and use this throughout the whole image.
2. Detect features in multiscales and combine them.
3. Control the scale in the image space.

The best scale corresponds to the best filter size in methods of class (1), and to neighbourhood size in algorithms of class (2). Multiscale solutions in convolution

based methods of class (1) are obtained by using multiple standard deviation values with a Gaussian-like filter. In methods of class (2), multiscale solutions are calculated by fitting polynomials to local image data with different kernel sizes. Adaptive smoothing is based on convolution of the input image with a fixed sized filter where filter coefficients are changed in each spatial location. The multiscale solutions are not obtained by changing the size of the filter, but by changing the coefficients of the constant sized nonlinear filter with the scale parameter k. Adaptive smoothing falls into the third class since it's a filter though being adaptive which imposes local curve fitting. Algorithms based on regularization fall into the third class: In this class, smoothing is achieved by minimizing an energy functional where smoothing is induced into the functional by a constraint which usually includes first and second order derivatives of the signal. In this chapter, convolution based adaptive smoothing where size of the convolving filter is fixed but the contents are changed spatially across the data, and weak membrane modelling of regularization are compared for edge detection performance [14].

2.2 Adaptive Smoothing

In adaptive smoothing, it's aimed to keep discontinuities between regions while performing smoothing operation in the regions [1]. Smoothing is performed by convolving the image with a filter where the coefficients depend on the neighbouring pixel values in a small neighbourhood, such as 3×3, until a predefined convergence criteria is satisfied or a predefined maximum iteration count is reached. Filter coefficients are related to the first order derivative of the input signal in a nonlinear fashion. The method becomes adaptive by the recalculation of filter coefficients at each spatial location as well as in each iteration. The smoothing performed in this way relies on discontinuity preserving nonlinear filtering. The filtering is a iterative process and number of iterations needed for convergence may grow too many even though the smoothing filter size is as small as 3×3. In each iteration two major effect of the convolution is observed: *i.* The discontinuities are preserved and enhanced, *ii.* continuous regions are smoothed. Discontinuities are emphasized after several iterations and are not changed there-

after. But the smoothing of regions require big counts of iterations. Thus in the purpose of edge detection, iterations are terminated after several iterations and edge detection is performed with a convenient thresholding scheme.

2.2.1 The Nonlinear Filtering

If the filter at (t)th iteration is denoted by $w^{(t)}(x,y)$, the signal by $I^{(t)}(x,y)$, smoothed signal $I^{(t+1)}(x,y)$ at $(t+1)$th iteration is expressed as,

$$I^{(t+1)}(x,y) = \frac{1}{N(x,y)^{(t)}} \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I^{(t)}(x+i,y+j) \cdot w^{(t)}(x+i,y+j) \quad (2.1)$$

where,

$$N(x,y)^{(t)} = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} w^{(t)}(x+i,y+j). \quad (2.2)$$

The small size of the filter is expected to be an advantage which should decrease the computation time. The suggested filter is given by the following equation [1]:

$$w^{(t)}(x,y) = f(d^{(t)}(x,y)) = e^{-\frac{|d^{(t)}(x,y)|^2}{2k^2}} \quad (2.3)$$

$$d^{(t)}(x,y) = \sqrt{G_x^2 + G_y^2}. \quad (2.4)$$

where $d^{(t)}(x,y)$ denotes the discontinuity at (x,y) of the signal at the (t)th iteration and the first order derivative is used to express discontinuity. G_x and G_y represent the first order derivatives of the signal in x and y directions respectively. The scale parameter k remains the same over an image and multiscale edges are obtained by changing k.

The filter in Eq. 2.3 is not linear since it does not satisfy the following condition:

$$\text{Linearity} \iff T[ax_1(n_1,n_2) + bx_2(n_1,n_2)] = ay_1(n_1,n_2) + by_2(n_1,n_2), \quad (2.5)$$

where $T[x_1(n_1,n_2)] = y_1(n_1,n_2)$, a and b are any scalar constants, and $A \iff B$ means that A implies B and B implies A [15].

Parameter k signifies magnitude of edges that should survive during smoothing. k is presented to be the scale parameter of this algorithm [1]. Very smoothed results are obtained for large values of k, and small contrasts are preserved for small k values. Effect of scale parameter k may be summarized as following:

- Discontinuities are smoothed for large values of k,
- Discontinuities are preserved for small values of k.

The order of smoothing is not determined solely by k, the first order derivative at that location is also used. First order derivative means a change in intensity values where noise may be one of the reasons of this intensity change. So, very low smoothing may be achieved for even large values of scale parameter, thus causing noise effect to survive. This is only the case for large k values since smoothing is not very strong in small scales. But noise is preserved at even the coarse scales which results in false edge occurrences to appear.

2.2.2 Iterative Weighted Averaging

Weighted averaging is defined as "convolving the image with a filter where coefficients are changed in each iteration, or coefficients differ for each spatial location" [1]. The goal is to smooth continuous regions by taking $d^{(t)}(x,y) = 0$, and taking a positive value for $d^{(t)}$ at discontinuities. By this approach, the filter coefficients are changed by *i.* iterations, and *ii.* spatial locations. The coefficients are modified by the first order derivative which means that they are effected from a very small neighbourhood, i.e. 3x3 sized neighbourhood. Small sized filters are known to have good localization performance [6, 10, 11]. But taking a small sized kernel has the disadvantage of poor detection, since a tradeoff exists between localization and detection performance of edge detectors.

2.3 Weak Membrane Modelling

2.3.1 Piecewise Continuous Reconstruction

Detection of discontinuities from an image is customarily done by first blurring the image with a lowpass filter, and then edges are extracted from the blurred version of the image by finding the large values of the first order derivative or zero crossings of the second derivative. But smoothing the image destroys data as well as smoothes noise. This distortion of the image causes shifting of discontinuities in scale space.

In the weak string which is the 1D model of weak membrane, discontinuities are detected without dislocalization. This is very important since the images generally include noise and this noise is desired to be reduced before an edge detection algorithm is applied. The noise reduction may be done by using large spatial scaled filters for blurring the image. But linear filters cause dislocalization, rounding of corners, disconnection of T-junctions and occurrence of edge shifts near intensity changes. These errors increase as the scale of the filter increases.

The weak string and weak membrane modelling preserves the discontinuities without necessitating any *a priori* knowledge about the location and existence of edge points, and does not exhibit the errors of linear filters even in relatively high noise levels. Edges can be detected without causing serious dislocalization. The edge segments detected are in forms of smooth curves and separate regions. These regions of continuity are smoothed to suppress noise and to produce smooth as well as natural surfaces. Thus, weak membrane modelling establishes two major facilities: The first is the preservation of discontinuities, i.e. detection of edges accomplished by using line processes. The other is to produce smooth and continuous regions where edges do not exist. This behaviour gives weak membrane model the property of piecewise continuous reconstruction.

2.3.2 Weak Membrane Modelling And Line Processes

The membrane modelling is based on regularization theory. String functional is given as [2, 6]:

$$E_{s1} = \int_{\Omega_s} (u(x) - d(x))^2 dx + \lambda^2 \int_{\Omega_s} u_x^2(x) dx \tag{2.6}$$

where $d(x)$ is the input function, $u(x)$ is the reconstructed function, $u_x(x)$ is the first order derivative with respect to x and λ is the regularization parameter. The membrane may be thought as the 2D version of the string functional above. The membrane energy functional is given as:

$$E_{m1} = \int\int_{\Omega} (u(x,y) - d(x,y))^2 dxdy + \lambda^2 \int\int_{\Omega} (u_x^2(x,y) + u_y^2(x,y)) dxdy. \tag{2.7}$$

The first term assures that the reconstructed function $u(x,y)$ to be close to the input function $d(x,y)$. The second term which includes derivatives of $u(x,y)$ with respect to x and y imposes the smoothness constraint. The energy functionals shown in Eq. 2.6 and Eq. 2.7 are known as *string* functional in 1D and *membrane* functional in 2D, respectively. If the smoothness term includes the second order derivatives, then that energy functional will be named as *rod* functional in 1D and *plate* functional in 2D [2]. In this chapter, string and membrane functionals are investigated.

2.3.2.1 Weak String

Detection of discontinuities (edges) on the 1D data can be realized by construction of a 1D function $u(x)$ which is a good fit to some input data $d(x)$ in Eq. 2.6. The construction can be done by modelling $u(x)$ as a *weak elastic string*, which is an elastic string under weak continuity constraints. The discontinuities may be seen as the *breaks* in the string where the weak continuity comes into scene.

The weak form of string functional in Eq. 2.6 is specified by its energy shown below.

$$E_s = \int_{\Omega_s} (u(x) - d(x))^2 dx + \lambda^2 \int_{\Omega_s} u_x^2(x)(1 - \ell(x))dx + \alpha \int_{[0,1]} \ell(x)dx \qquad (2.8)$$

where $u_x(x)$ denotes the first order derivative, and ℓ is the *line process*. Line process marks discontinuities and assures piecewise smoothing. When $\ell(x) = 1$, the second term vanishes and the overall functional will have lower energy, which permits any value for the first derivative to take, hence causing discontinuity. The last term with α coefficient is the penalty term added to prevent all line processes to be 1. In this way, balance between discontinuity occurrence by $\ell = 1$ and continuity preservation by $\ell = 0$ is established by α and λ coefficients. This functional may be written in discrete form as

$$E_s = \sum_i (u_i - d_i)^2 + \lambda^2 \sum_i (u_i - u_{i-1})^2 (1 - \ell_i) + \alpha \sum_i \ell_i. \qquad (2.9)$$

The first summation term assures that the reconstructed function u to be close to the input d. The second and third summation terms which include the first order derivatives and line processes imposes the smoothness constraint together with breaking by line processes.

2.3.2.2 Weak Membrane

In weak membrane modelling, image is first modelled by a piecewise continuous function where the modelling is carried out by non-convex weak membrane functional. This functional is the 2D version of weak string functional in Eq. 2.8 and expressed in continuous form as

$$E_m = \underbrace{\int\int_{\Omega} (u(x,y) - d(x,y))^2 dxdy}_{D} + \qquad (2.10)$$

$$\underbrace{\lambda^2 \int\int_{\Omega} (u_x^2(x,y)(1-m(x,y)) + u_y^2(x,y)(1-\ell(x,y)))dxdy}_{S} +$$

$$\underbrace{\alpha \int\int_{[0,1]} (m(x,y)+\ell(x,y))}_{P}.$$

where $m(x,y)$ and $\ell(x,y)$ are vertical and horizontal line processes, respectively. The image is reconstructed by minimizing this non-convex functional of weak membrane which preserves discontinuities using line processes $\ell(x,y)$ and $m(x,y)$.

The energy functional E of weak membrane is composed of three main parts: D, S and P part as in Eq. 2.10. If $u_{i,j}$ denotes the reconstructed image, and $d_{i,j}$ denotes the original input image, the energy functional in discrete case is expressed as,

$$E = D + S + P \tag{2.11}$$

$$D = \sum_i \sum_j (u_{i,j} - d_{i,j})^2 \tag{2.12}$$

$$S = \lambda^2 \sum_i \sum_j (u_{i,j} - u_{i-1,j})^2 (1 - m_{i,j}) + (u_{i,j} - u_{i,j-1})^2 (1 - \ell_{i,j}) \tag{2.13}$$

$$P = \alpha \sum_i \sum_j (\ell_{i,j} + m_{i,j}) \tag{2.14}$$

where $\ell_{i,j}$ and $m_{i,j}$ denotes horizontal and vertical line processes respectively. D term is a measure of faithfulness to input data. S term represents deformation of the membrane surface and allows the membrane to break at locations where line process is 1. P is α levied measure of penalty incorporated by line processes. S and P terms form the balance between smoothness of surface and occurrence of discontinuities. The locations of edges are the points where the corresponding line process is 1.

2.3.3 Graduated Non-Convexity Algorithm

The minimization of the energy functional in Eq. 2.11 cannot be performed by a regular minimization method since this functional lacks the mathematical

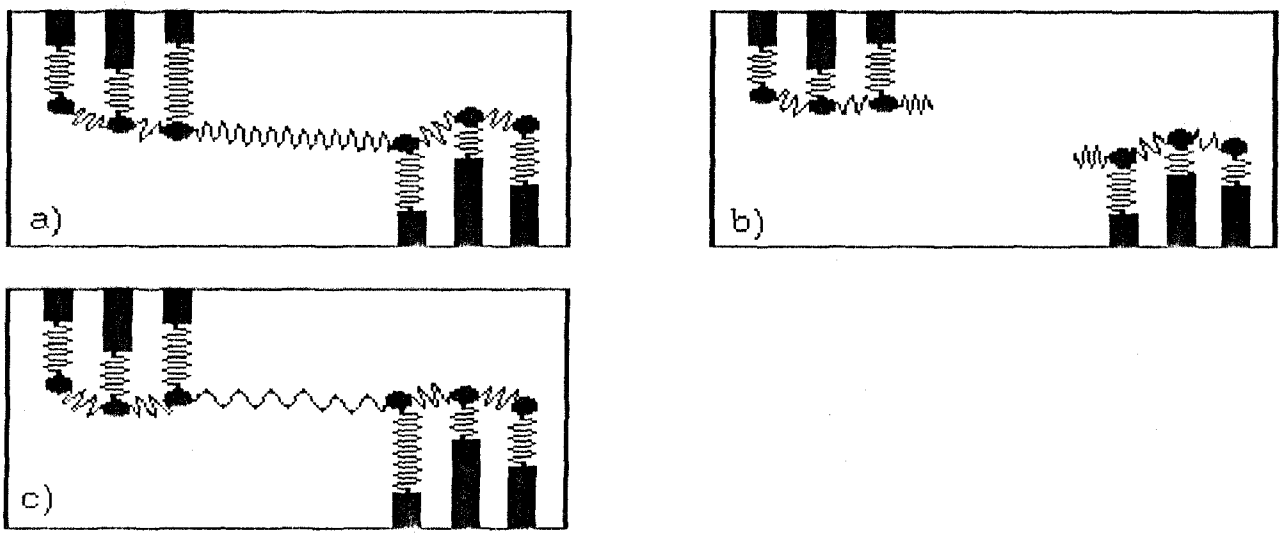


Figure 2.1. Non-convexity. The states of weak string (a) and (b) are stable, the intermediate state (c) has higher energy than those.

property of convexity. This means that the system may have many u minimizing E, but only one of those minima is the global minimum while the others are all the local minima of energy functional E. Such local states resists to small perturbations and resists back to the same stable state when disturbed, but when a larger perturbation is applied, the system jumps and locks onto some other minimum and won't come back. This may be thought as, the string is extended passed its deformation or maybe break point, thus not keeping the old properties (such as Hook's constant k). This is illustrated in Fig. 2.1.

Descent algorithms will not suffice to find the global minimum of E. Local descent algorithms tend to stick to a local minimum point like the myopic fly in Fig. 2.2. Stochastic methods such as simulated annealing searches for the global minimum by random fluctuations. The computation needed may be very intensive in this sort of methods.

A method developed by Blake and Zisserman to solve this problem is *Graduated Non-Convexity* algorithm [2]. A preprocessing is first performed to eliminate line processes, and S and P terms are combined into a single term which exhibit the function of each term. The combined function is called neighbourhood interaction function and denoted by $g_{\alpha,\lambda}$. A further modification is done in the energy

functional by adding a parameter p which controls the convexity, and the local neighbourhood interaction function $g_{\alpha,\lambda}$ is further modified by incorporating this p parameter.

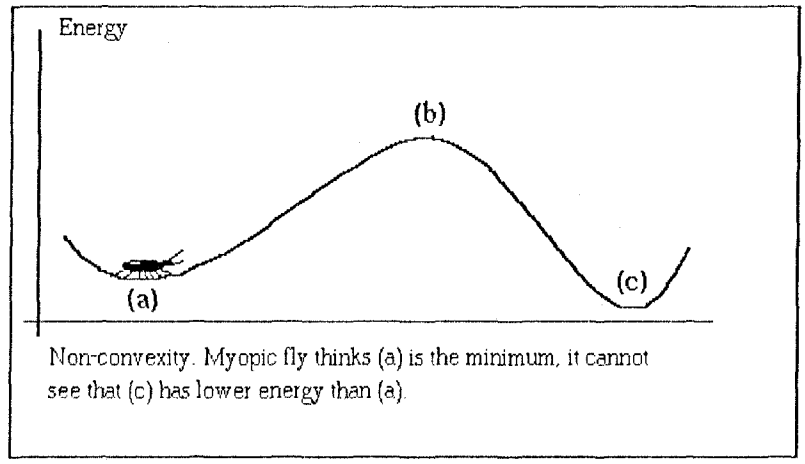


Figure 2.2. The non-convexity. The myopic fly thinks state a is the best, it has no way of seeing state c.

2.3.3.1 Elimination of Line Processes

The energy functional obtained in Eq. 2.9 is a function of the nodal variables u and the line variables ℓ. Since the line variables may be recovered in advance, a first step minimization will be performed on line processes. To eliminate the line processes, the energy functional E_s in Eq. 2.9 is expressed as

$$E_s = D + \sum_i h_{\alpha,\lambda}(u_i - u_{i-1}, \ell_i) \tag{2.15}$$

where

$$D = \sum_i (u_i - d_i)^2 \tag{2.16}$$

$$h_{\alpha,\lambda}(t,\ell) = \lambda^2 t^2 (1-\ell) + \alpha\ell. \tag{2.17}$$

Beware that the function $h_{\alpha,\lambda}$ is obtained by combining the S and P terms and that the overall functional in Eq. 2.16 is the same one as in Eq. 2.9. The depen-

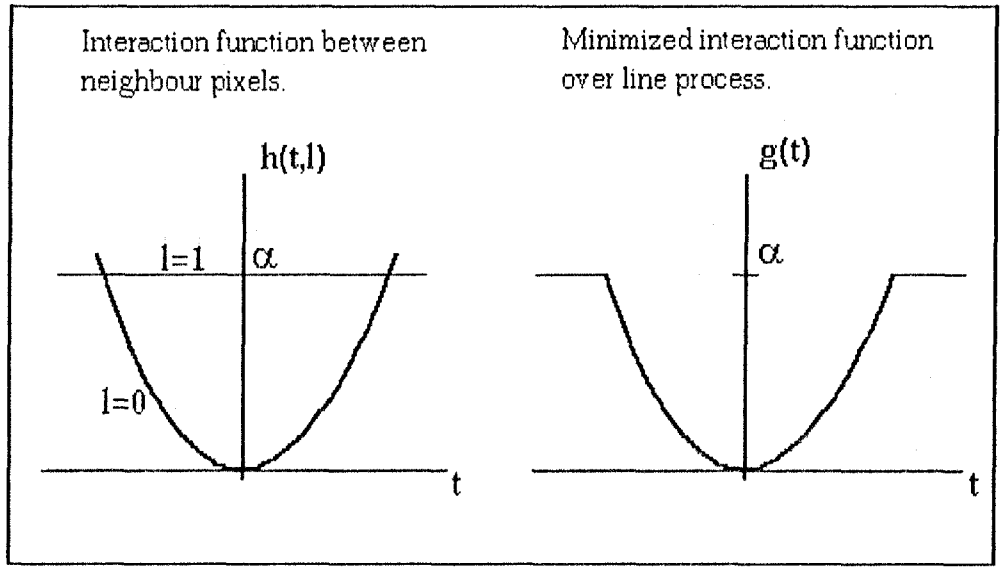


Figure 2.3. Neighbour interaction function and elimination of line variables. The line process ℓ can be eliminated by minimization over $\ell \in [0,1]$. t denotes the first order derivative.

dency on ℓ_i is now contained in the $h_{\alpha,\lambda}$ part. The $h_{\alpha,\lambda}$ function is plotted in Fig. 2.3.

The function $h_{\alpha,\lambda}$ controls the local interactions between u_i nodes. The problem is

$$\min_{\{u_i,\ell_i\}} E_s = \min_{\{u_i\}}(D + \min_{\{\ell_i\}} \sum h_{\alpha,\lambda}(u_i - u_{i-1}, \ell_i)) \tag{2.18}$$

One should note that D term does not include ℓ_i, hence minimization over ℓ can be performed immediately in Eq. 2.18. Then the problem reduces to

$$\begin{aligned} u(x;\lambda,\alpha) &= \{u(x) : E_s(u,\lambda,\alpha) = \\ &\quad \inf_{f\in\Omega_s}\left[\int_{\Omega_s}(f-d)^2dx + \lambda^2\int_{\Omega_s} g_{\alpha,\lambda}(f_x)dx\right]\} \end{aligned} \tag{2.19}$$

and in discrete case,

$$u = \left\{ u : E_s(u, \lambda, \alpha) = \inf_{f \in \Omega_s} \left[\sum_i (f - d)^2 + \lambda^2 \sum_i g_{\alpha,\lambda}(f_i - f_{i-1}) \right] \right\} \tag{2.20}$$

where

$$g_{\alpha,\lambda}(t) = \min_{\{\ell \in [0,1]\}} h_{\alpha,\lambda}(t, \ell). \tag{2.21}$$

The function $g_{\alpha,\lambda}$ is shown on the right side of Fig. 2.3, which is simply the minimum of two graphs on the left side, and named the *neighbourhood interaction function*. This function may be written from Eq. 2.17 as

$$g_{\alpha,\lambda}(t) = \begin{cases} \lambda^2 t^2 & \text{if } |t| < \sqrt{\alpha}/\lambda \qquad (\ell = 0) \\ \alpha & \text{otherwise} \qquad\qquad\quad (\ell = 1). \end{cases} \tag{2.22}$$

The line processes can be recovered at any time from $g_{\alpha,\lambda}(t)$ by

$$\ell(t) = \begin{cases} 1 & |t| > \sqrt{\alpha}/\lambda \\ 0 & \text{otherwise.} \end{cases} \tag{2.23}$$

where t is the first order derivative. The above modification may be extended to 2D case and the 2D functional may be derived from Eq. 2.11 as

$$\begin{aligned} E \;=\; & \sum_{i,j} (u_{i,j} - d_{i,j})^2 + \\ & \lambda^2 \sum_{i,j} h_{\alpha,\lambda}(u_{i,j} - u_{i-1,j}, m_{i,j}) + \lambda^2 \sum_{i,j} h_{\alpha,\lambda}(u_{i,j} - u_{i,j-1}, \ell_{i,j}) \end{aligned} \tag{2.24}$$

and by using neighbourhood interaction function,

$$\begin{aligned} E \;=\; & \sum_{i,j} (u_{i,j} - d_{i,j})^2 + \\ & \lambda^2 \sum_{i,j} g_{\alpha,\lambda}(u_{i,j} - u_{i-1,j}) + \lambda^2 \sum_{i,j} g_{\alpha,\lambda}(u_{i,j} - u_{i,j-1}) \end{aligned} \tag{2.25}$$

is obtained. Recovery of line variables m and ℓ is the same as in Eq. 2.23. The solution of this functional is now expressed as

$$u = \{u : E(u,\lambda,\alpha) = \inf_{f\in\Omega}[\sum_{i,j}(f-d)^2 + \lambda^2 \sum_{i,j} g_{\alpha,\lambda}(f_{i,j}-f_{i-1,j}) + g_{\alpha,\lambda}(f_{i,j}-f_{i,j-1})+]\}. \tag{2.26}$$

2.3.3.2 Modification of Energy Functional

Minimization of the non-convex energy functional in Eq. 2.26 is carried out by GNC which is an iterative scheme. In this method, the cost functional E is first approximated by a convex functional E^*, which has only one minimum that is the global one. Then a sequence of functionals is used to approach to the original non-convex functional E. Thus, a general strategy for small or large λ, is to use a sequence of cost functionals $E^{(p)}$, for $1 \geq p \geq 0$, to approach to the global minimum of E by decreasing p values at each step. These are chosen so that $E^{(1)} = E^*$, the convex approximation to E; and $E^{(0)} = E$, the non-convex functional itself. Here, $E^{(p)}$ is changed continuously from $E^{(1)}$ to $E^{(0)}$ as GNC proceeds. This algorithm will then find the global minimum of E with appropriate steps for p, such as $p = \{1, 1/2, 1/4, 1/8, 1/16\}$, and using the result of one optimization in the succeeding one [2]. A simple GNC algorithm may be tabulated as in Table 2.1

The modified energy functional where the line processes are eliminated and local interaction function used is expressed as,

$$E^{(p)} = \sum_i \sum_j (u_{i,j}-d_{i,j})^2 + \sum_i \sum_j g^{(p)}_{\alpha,\lambda}(u_{i,j}-u_{i-1,j}) + \sum_i \sum_j g^{(p)}_{\alpha,\lambda}(u_{i,j}-u_{i,j-1}), \tag{2.27}$$

Table 2.1. A simple GNC algorithm.

i. Make a good approximation to E^* to E.

   (This problem expands to finding an appropriate $u^{(0)}$ to start with. A good choice to reduce the iteration count is to start with the input data d, i.e. $\forall_i, u_i^{(0)} = d_i$.)

ii. Start with $p = 1$, thus making $e^{(1)} = E^*$, the convex approximation of the energy functional.

iii. Find the local minimum of $E^{(p)}$ by a local minimization method.

   (Direct descent, Successive-Over-Relaxation (SOR) may be chosen. In this implementation, SOR is used.)

iv. Decrease p, e.g. $p = p/2$.

v. if $p \geq p_{min}$ then goto (iii), (e.g. $p_{min} = 1/16$).

vi. done.

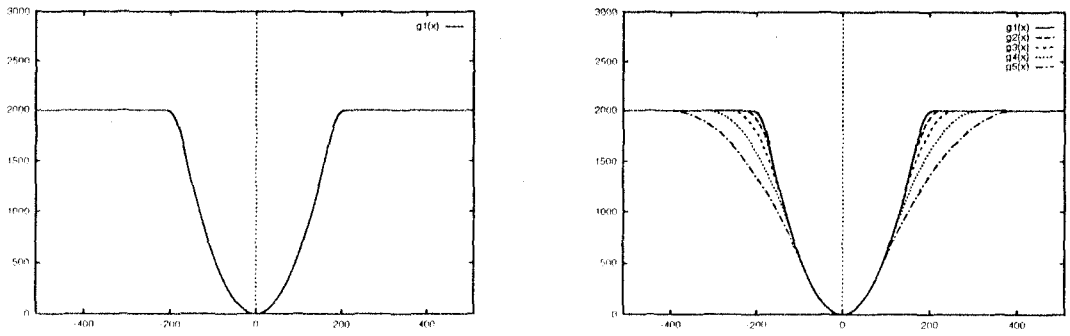


Figure 2.4. The local interaction function in 1D used in GNC. The graph on the left side is the original function. The graph on the right side shows how $g_{\alpha,\lambda}$ is changed by p. g1(x) denotes the unmodified neighbourhood functional when $p = 1$. The value of p is 1 for g1(x), 1/2 for g2(x), 1/4 for g3(x), 1/8 for g4(x) and 1/16 for g5(x). λ is 0.24 and α is 2000.

and

$$g_{\alpha,\lambda}^{(p)} = \begin{cases} \lambda^2 t^2 & |t| < q \\ \alpha - c(|t| - r)^2/2 & q \le |t| < r \\ \alpha & |t| \ge r, \end{cases} \tag{2.28}$$

where $c = c^*/p$, $r^2 = \alpha(2/c + 1/\lambda^2)$, $q = \alpha/(\lambda^2 r)$ [2, 3]. $g_{\alpha,\lambda}^{(p)}$) is called the neighborhood interaction function. The eliminated line processes can be recovered by Eq. 2.23. The modified neighbour interaction function in Eq. 2.28 is plotted in Fig. 2.4.

The energy functional is convex at the start, i.e. $E^{(p=1)}$ is a convex functional. The functional $E^{(p)}$ becomes non-convex as p decreases in each iteration. There are two iterations during the GNC process. During the outer iteration, p is started from 1 and proceeds down to the smallest value in the p set. Then for each value of p, there are inner iterations, which are selected to be the SOR iterations. For each p value, the minimization is realized by SOR (*Successive Over Relaxation*) method. This is a gradient descent method and the minimization of $E^{(p)}$ in each step of p must satisfy

$$\frac{\partial E^{(p)}}{\partial u_{i,j}} = 0 \tag{2.29}$$

condition. The first derivative of $E^{(p)}$ in Eq. 2.27 with respect to $u_{i,j}$ is

$$\begin{aligned} \frac{\partial E^{(p)}}{\partial u_{i,j}} = {} & 2[(u_{i,j} - d_{i,j}) + \\ & g_{\alpha,\lambda}^{(p)'}(u_{i,j} - u_{i-1,j}) + g_{\alpha,\lambda}^{(p)'}(u_{i,j} - u_{i+1,j}) + \\ & g_{\alpha,\lambda}^{(p)'}(u_{i,j} - u_{i,j-1}) + g_{\alpha,\lambda}^{(p)'}(u_{i,j} - u_{i,j+1})]. \end{aligned} \tag{2.30}$$

The first derivative of modified local interaction function in Eq. 2.28 with respect to $u_{i,j}$ can be calculated as

$$g'^{(p)}_{\alpha,\lambda}(t) = \begin{cases} 2\lambda^2 t & |t| < q \\ -c(|t| - r)sign(t) & q \le |t| < r \\ 0 & |t| \ge r \end{cases} \tag{2.31}$$

where q and r are defined in Eq. 2.28.

2.3.3.3 The SOR Iterations

The SOR iterations proceed as

$$u^{(n+1)}_{i,j} = u^{(n)}_{i,j} - \frac{w}{T}\frac{\partial E^{(p)}}{\partial u_{i,j}}. \tag{2.32}$$

The updating of $u_{i,j}$ in serial SOR iterations is performed by

$$\begin{aligned} u^{(n+1)}_{i,j} \;=\; & u^{(n)}_{i,j} - w\,[\,2(u^{(n)}_{i,j} - d_{i,j}) + \\ & g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n+1)}_{i-1,j}) + g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n+1)}_{i,j-1}) + \\ & g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n)}_{i+1,j}) + g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n)}_{i,j+1})\,]\,/\,(2 + 8\lambda^2) \end{aligned} \tag{2.33}$$

where $T = 2(1 + 4\lambda^2)$ for regular elements. The 1 is for D term, and each one of four λ^2 is for each $g'^{(p)}(.)$ term. The multiplier 2 comes from the squares in each term. This iterative equation needs special treatment at the boundaries. For example at the upper left corner the $u_{i,j}$ update becomes

$$\begin{aligned} u^{(n+1)}_{0,0} \;=\; & u^{(n)}_{0,0} - \omega\,[\,2(u^{(n)}_{0,0} - d_{0,0}) + \\ & g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{0,0} - u^{(n)}_{1,0}) + g'^{(p)}_{\alpha,\lambda}(u^{(n)}_{0,0} - u^{(n)}_{0,1})\,]\,/\,(2 + 4\lambda^2). \end{aligned} \tag{2.34}$$

A convenient convergence norm is checked for each p iteration. The convergence norm used in this implementation is the infinite norm and defined as

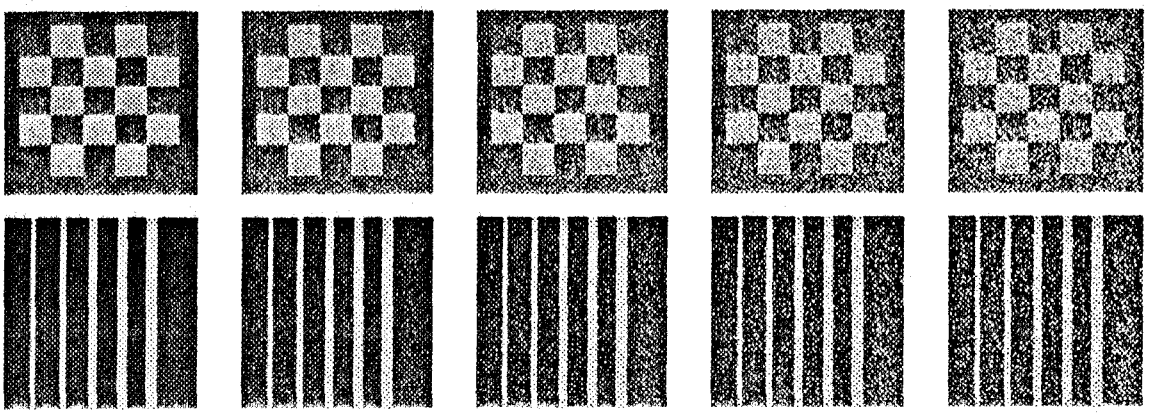


Figure 2.5. Noisy checkerboard and bars images. SNR(dB) values are (from left to right) 10, 7, 5, 4 and 3.

$$L_\infty = \max |u_{i,j}^{(n+1)} - u_{i,j}^{(n)}| \tag{2.35}$$

In addition to this norm, maximum iteration count in each SOR iteration is bounded to an upper value.

2.4 Results and Analysis of Edge Detection Performance of the Methods

Adaptive smoothing and weak membrane modelling methods are applied to various synthetic images to exhibit different problems of edge detection. Gaussian noise is added to synthetic checkerboard and bars images to obtain various SNR valued images, and two methods under consideration are applied on these samples (Fig 2.5).

Synthetic images are used to obtain the reconstructed surfaces and edge images to evaluate the qualitative performance of the methods. The surfaces reconstructed by weak membrane modeling for checkerboard and bars images are given in Fig. 2.6 and Fig. 2.8, respectively. The smoothed surfaces obtained by adaptive smoothing for checkerboard and bars images are given in Fig. 2.10 and Fig. 2.12, respectively. The edges obtained by adaptive smoothing for checkerboard and

Table 2.2. Qualitative analysis of edge detection performance of weak membrane modeling over checkerboard images with different SNR values.

G.Noise		GNC		SOR		CHECKERBOARD				
SNR	σ	λ	α	ω	Iter.	P(AE/IE)	P(IE/AE)	FoM	MSD	MAD
-	0	1.4	2400	1.3	7	1.0000	1.0000	1.0000	0.0000	0.0000
10 dB	10	1.4	2400	1.3	11	1.0000	1.0000	1.0000	0.0000	0.0000
7 dB	20	1.6	2800	1.4	317	0.9980	0.9990	0.9981	0.0201	0.0010
5 dB	30	1.9	4400	1.4	418	0.9939	0.9939	0.9945	0.0493	0.0061
4 dB	40	1.8	4400	1.4	510	0.9565	0.9184	0.9246	0.3244	0.2632
3 dB	50	1.8	4400	1.4	510	0.8836	0.7251	0.7427	0.6514	1.0608

Table 2.3. Qualitative analysis of edge detection performance of weak membrane modeling over bars images with different SNR values.

G.Noise		GNC		SOR		BARS				
SNR	σ	λ	α	ω	Iter.	P(AE/IE)	P(IE/AE)	FoM	MSD	MAD
-	0	1.4	2400	1.3	7	1.0000	1.0000	1.0000	0.0000	0.0000
10 dB	10	1.4	2400	1.3	11	1.0000	1.0000	1.0000	0.0000	0.0000
7 dB	20	1.6	2800	1.4	317	0.9980	0.9990	0.9981	0.0201	0.0010
5 dB	30	1.9	4400	1.4	418	0.9939	0.9939	0.9945	0.0493	0.0061
4 dB	40	1.8	4400	1.4	510	0.9565	0.9184	0.9246	0.3244	0.2632
3 dB	50	1.8	5000	1.4	510	0.8684	0.7974	0.8116	0.5385	0.7249

bars images are presented in Fig. 2.11 and Fig. 2.13, respectively. Edge maps of checkerboards and bars images obtained by weak membrane modelling are shown in Fig. 2.7 and Fig. 2.9, respectively.

In order to evaluate the quantitative performance, some numerics are calculated on the resultant edge images, such as FoM (Figure Of Merit), P(AE/IE), P(IE/AE), and edge location errors (MAD and MSD) [8, 6]. P(AE/IE) denotes the probability of assigned edge occurrence provided that an ideal edge exists. P(IE/AE) is the probability of the reverse condition. Table 2.2 and 2.3 shows these numerical values for checkerboard and bars images for weak membrane modelling. Table 2.4 and 2.5 shows similar results for adaptive smoothing.

From the investigation of edge images and tables showing numerical performance results, it can be seen that the edge image obtained by minimizing the non-convex energy functional with GNC emerging from weak membrane is much better than the edge images obtained by adaptive smoothing. It is seen that

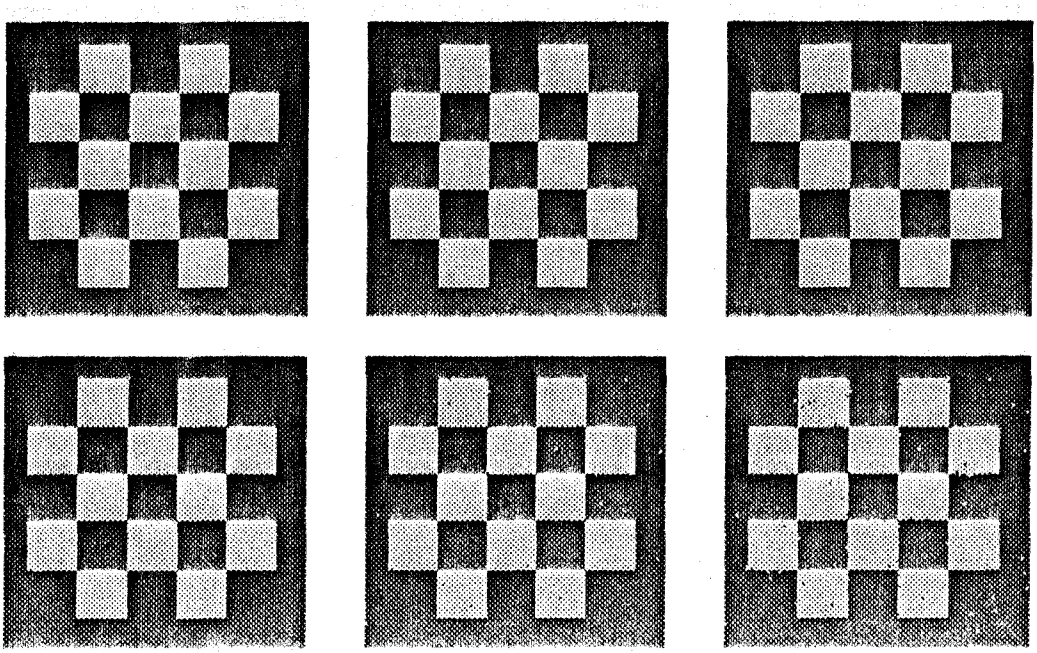


Figure 2.6. Reconstructed surfaces obtained by weak membrane modeling from noisy checkerboard images. SNR(dB) values from left to right: The first row: No noise, 10, 7. The second row: 5, 4, 3.

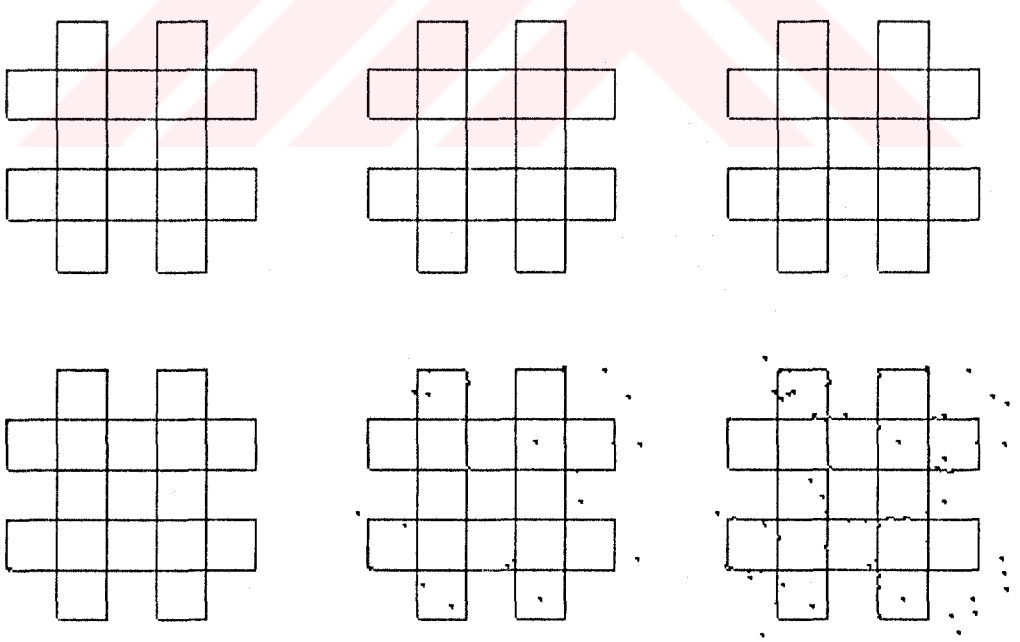


Figure 2.7. Edge maps obtained by weak membrane modeling from noisy checkerboard images. SNR(dB) values from left to right: The first row: No noise, 10, 7. The second row: 5, 4, 3.

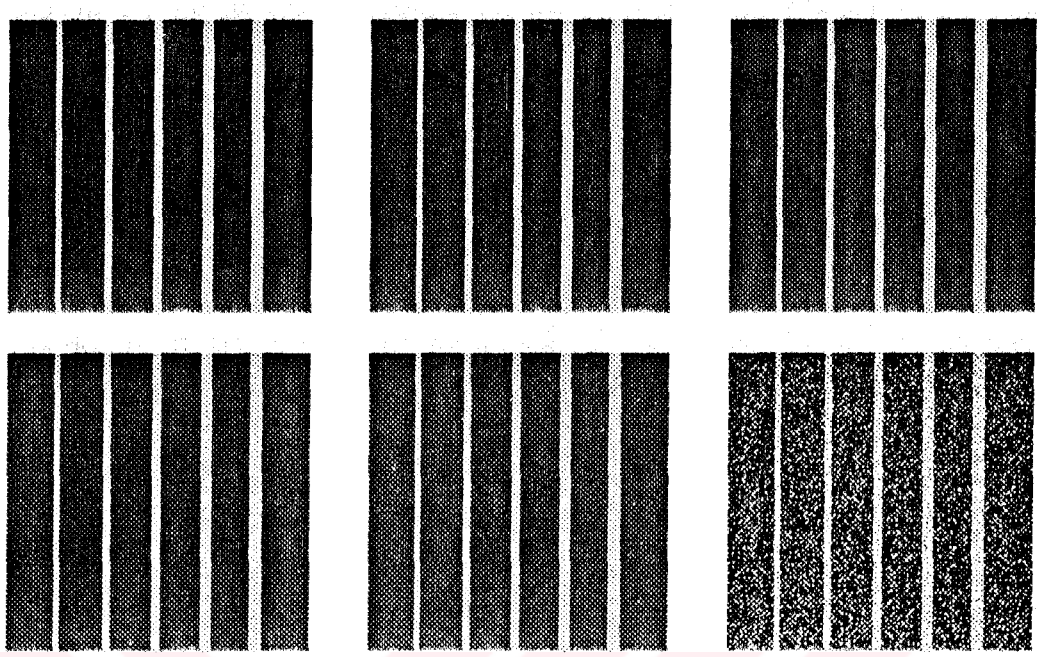


Figure 2.8. Reconstructed surfaces obtained by weak membrane modeling from noisy bars images. SNR(dB) values from left to right: The first row: No noise, 10, 7. The second row: 5, 4, 3.

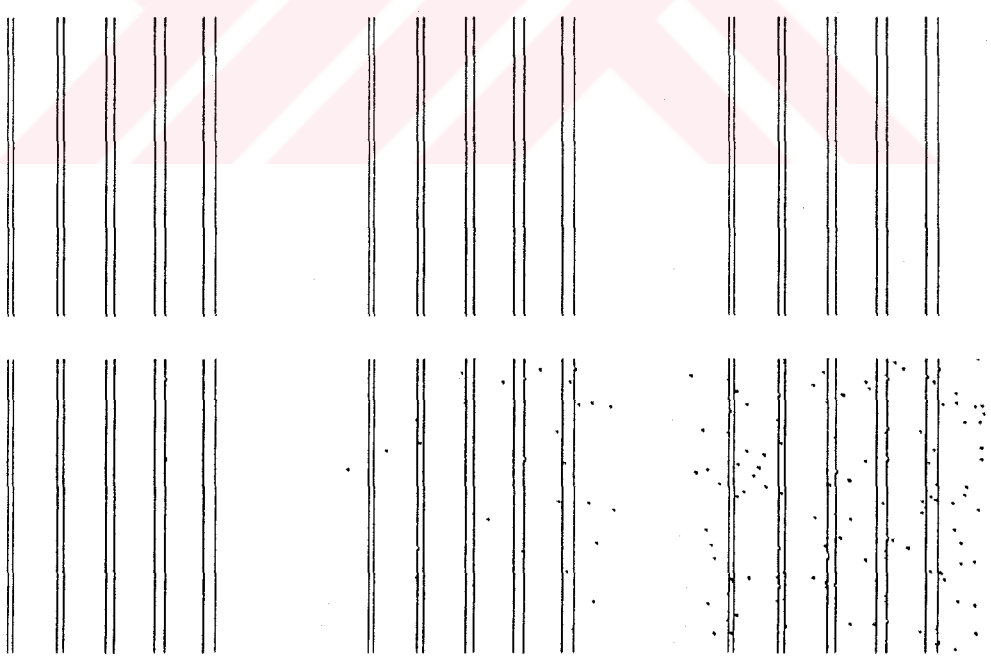


Figure 2.9. Edge maps obtained by weak membrane modeling from noisy bars images. SNR(dB) values from left to right: The first row: No noise, 10, 7. The second row: 5, 4, 3.

Table 2.4. Qualitative analysis of edge detection performance of adaptive smoothing over checkerboard images with different SNR values.

Noise (Gauss)		Scale	Thrs.	CHECKERBOARD				
SNR	σ	k	τ	P(AE/IE)	P(IE/AE)	FoM	MSD	MAD
-	0	2	32	0.4899	0.5000	0.5389	0.4472	0.5000
17 dB	2	2	32	0.4858	0.5000	0.5344	0.4477	0.5010
12 dB	6	2	32	0.4798	0.4938	0.5289	0.4505	0.5073
10 dB	10	6	32	0.4909	0.4393	0.4897	0.6198	0.9603
9 dB	12	6	32	0.4919	0.3408	0.3881	0.8980	2.0161
8 dB	15	4	32	0.4808	0.2167	0.2573	1.2279	3.7693
7 dB	20	10	32	0.3694	0.0996	0.1327	1.4785	5.4650

Table 2.5. Qualitative analysis of edge detection performance of adaptive smoothing over bars images with different SNR values.

Noise (Gauss)		Scale	Thrs.	BARS				
SNR	σ	k	τ	P(AE/IE)	P(IE/AE)	FoM	MSD	MAD
-	0	2	32	1.0000	1.0000	1.0000	0.0000	0.0000
17 dB	2	2	32	0.4858	0.5000	0.5344	0.4477	0.5010
12 dB	6	2	32	0.4798	0.4938	0.5289	0.4505	0.5073
10 dB	10	6	32	0.4909	0.4393	0.4897	0.6198	0.9603
9 dB	12	6	32	0.4919	0.3408	0.3881	0.8980	2.0161
8 dB	15	4	32	0.4808	0.2167	0.2573	1.2279	3.7693
7 dB	20	10	32	0.3694	0.0996	0.1327	1.4785	5.4650

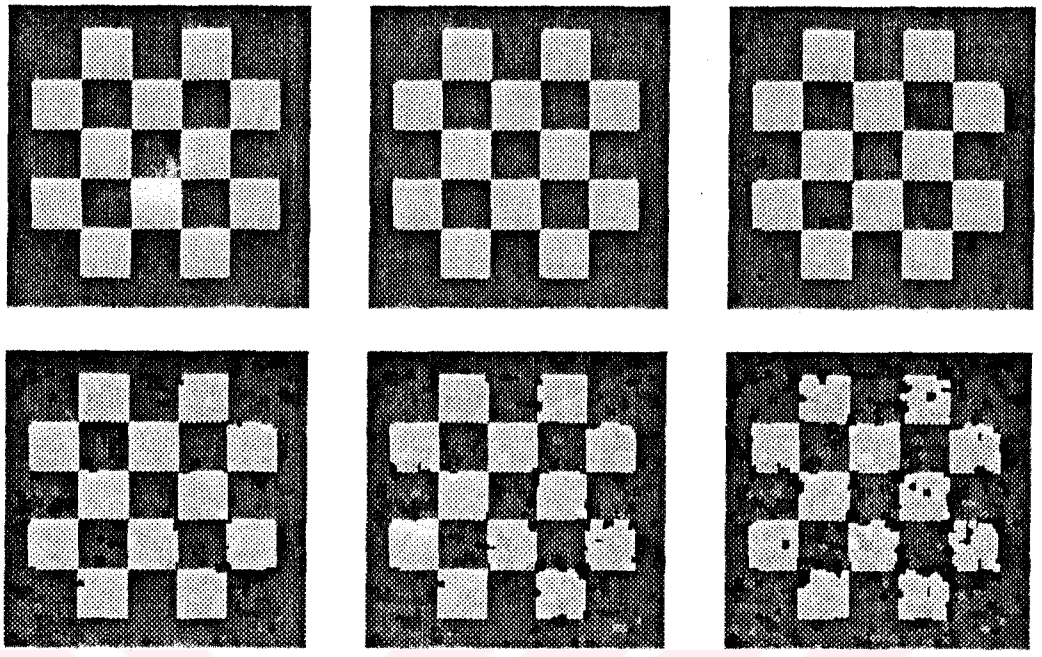


Figure 2.10. Reconstructed surfaces obtained by adaptive smoothing from noisy checkerboard images. SNR(dB) values from left to right: The first row: No noise, 17, 12. The second row: 10, 8, 7.

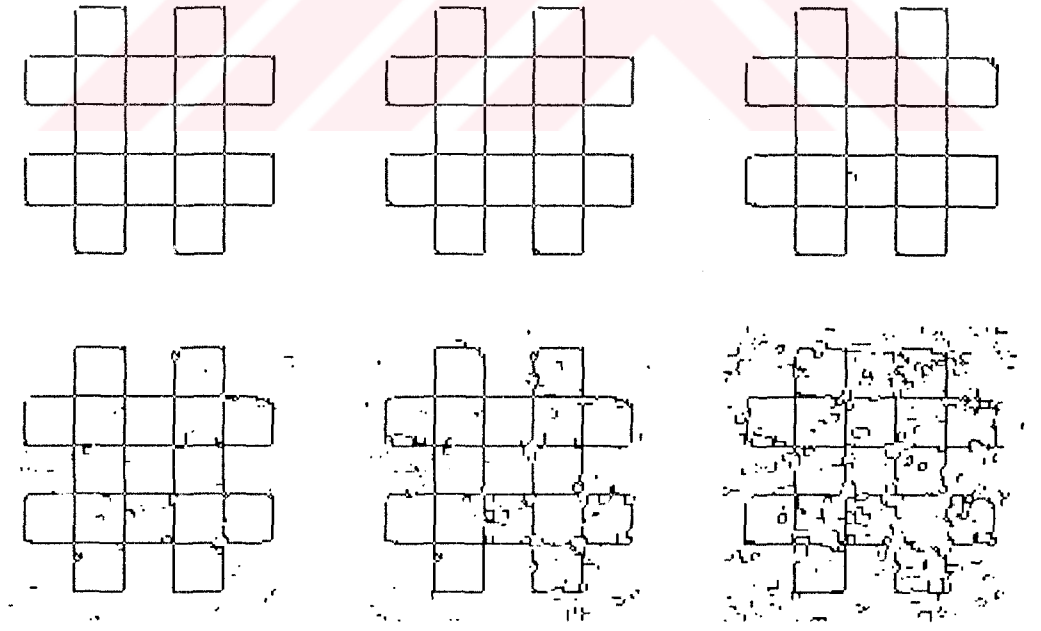


Figure 2.11. Edges obtained by adaptive smoothing from noisy checkerboard images. SNR(dB) values from left to right: The first row: no noise, 17, 12. The second row: 10, 8, 7.

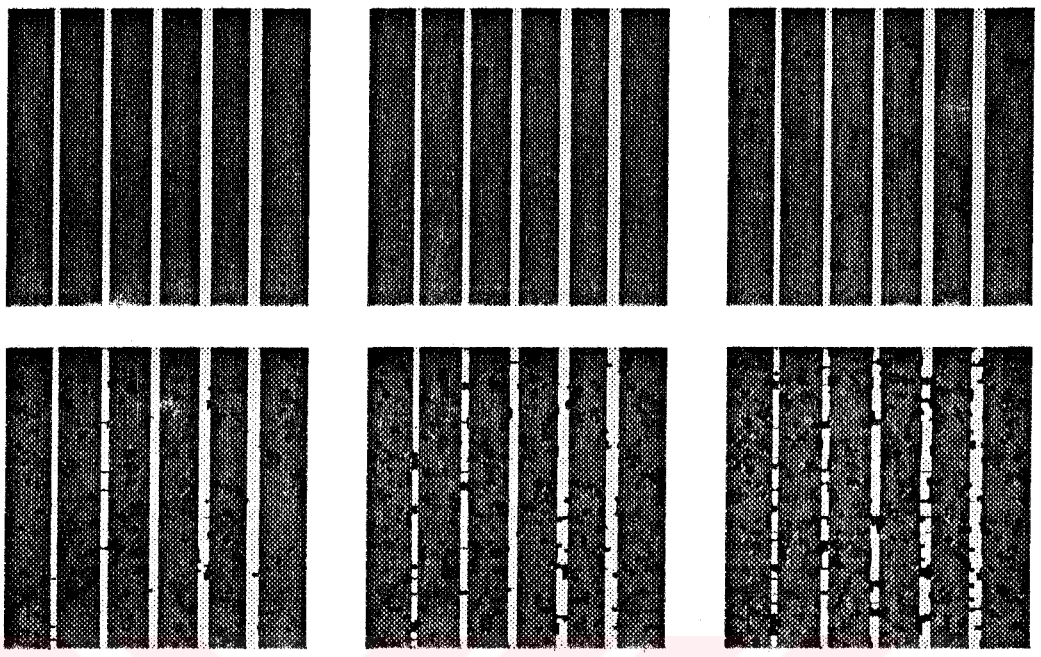


Figure 2.12. Reconstructed surfaces obtained by adaptive smoothing from noisy bars images. SNR(dB) values from left to right: The first row: No noise, 17, 12. The second row: 10, 8, 7.

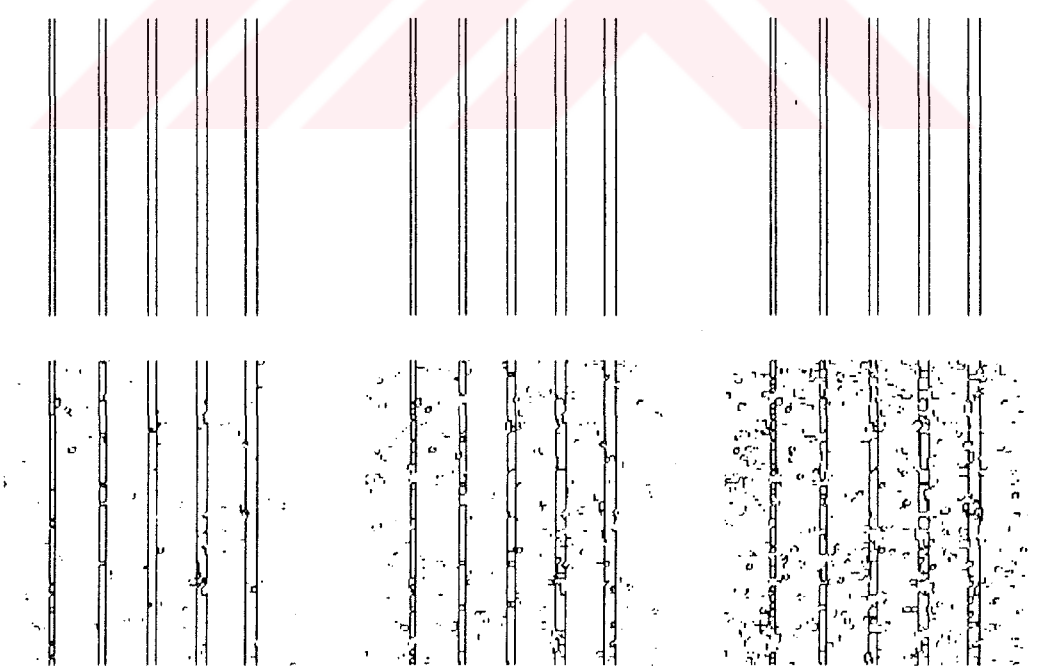


Figure 2.13. Edges obtained by adaptive smoothing from noisy bars images. SNR(dB) values from left to right: The first row: no noise, 17, 12. The second row: 10, 8, 7.

noise immunity of adaptive smoothing is not good. Use of a function including the first derivatives to calculate the filter coefficients is found to be the main reason which decreases the noise immunity of this method. Use of the smoothing term constructed from the local neighbourhood function $g_{\alpha,\lambda}$ in the energy functional of weak membrane model eliminates the effect of noise even for very low SNR values. This neighbourhood function also balances the breaking of the membrane and preservation of continuity in this model.

The weak membrane model is torn in continuous locations without causing excessive speckles in the membrane surface. Hence the edges detected by this model are continuous smooth curves. The smoothing operation is performed in a small neighbourhood and the resultant edge map is locally continuous in adaptive smoothing. For this reason, the detection performance of adaptive smoothing is seriously degraded.

Figure 2.14. Edges obtained by weak membrane modeling (middle row) and adaptive smoothing (last row) from Lenna and House (first row) images where the same parameters are used for two of the images. $\lambda = 1.8$, $\alpha = 1000$ and $SOR - \omega = 1.2$ in weak membrane modeling. Scale parameter $k = 2$ and threshold $\tau = 24$ in adaptive smoothing.

CHAPTER 3.

IMAGE RESTORATION USING REGULARIZATION

3.1 Introduction

In this chapter, restoration of blurred images in the case of a known blurring function is studied and a new approach to this kind of restoration is presented. It's aimed to obtain a clear and noiseless image by removing the blurring caused by misfocusing and noise added onto the degraded image. Regularization is used to obtain the restored image by surface reconstruction using the weak membrane functional. The non-convex energy functional emerging from weak membrane modeling is minimized by using the graduated non-convexity algorithm described in chapter 2. The blurred discontinuities are enhanced while the existing ones are preserved. The method is applied to real and synthetic images blurred by averaging with various filter sizes. The results are given and it is observed qualitatively and quantitatively that the method works quite good for this kind of degraded images.

The main goal in image restoration is to obtain an image as close as to the original one from the degraded one. Degradation is generally caused by the motion of objects or the imaging apparatus such as a camera, misfocusing of the lens system or the atmospheric effects. In addition to blurring, the noise coming from the recording media such as particles on the film, quantization or measurement error is added to the image and thus cause degradation. The degraded image g can be denoted as

$$g = Bf + n \tag{3.1}$$

where B is the blurring function, f is the original image and n is the additive noise. Since the noise is a stochastic process, the original image can be restored to some degree only. For this reason, it's usually impossible to obtain a perfect restoration from the degraded image g. In the restoration process, the aim is to obtain a $\hat{f}$ as close as to the original image f. A restoration method should satisfy the following criteria:

- Deblurring,
- Suppression of noise,
- Preservation of discontinuities,
- Reducing the ringing effect.

The linear restoration methods developed for this purpose generally cause ringing effects at locations of fast intensity changes. These ringing effects are not desired since they don't exist in the original image and some authors have studied to overcome this problem using the regularization approach [16].

3.1.1 Reduction of Image Blurring

When the blurred image model is expressed as

$$g(n1, n2) = f(n1, n2) * b(n1, n2) \tag{3.2}$$

which is a part of the degraded image function Eq. 3.1 without the noise addition. $b(n1, n2)$ is the blurring function (point spread function). The restoration from degraded image g can be performed by two conventional methods [15]:

1. Deconvolution (inverse filtering),
2. Blind deconvolution ($b(n1, n2)$ is not known).

3.1.2 Inverse Filtering

If the point spread function is known, an iterative procedure can be implemented for inverse filtering. If $\hat{f}_k$ is a good estimate to the restored image, $\hat{f}_k * b$ will be close to the degraded image g. Then $\hat{f}_{k+1}$ is estimated by adding a correction term proportional to difference between $\hat{f}_k * b$ and g to $\hat{f}_k$. An iterative function may be written as [15]

$$\hat{f}_{k+1}(n1, n2) = \hat{f}_k(n1, n2) + \lambda[g(n1, n2) - \hat{f}_k(n1, n2) * b] \tag{3.3}$$

where $\lambda > 0$ and the initial estimate for $\hat{f}_0$ may be chosen as

$$\hat{f}_0(n1, n2) = \lambda g(n1, n2). \tag{3.4}$$

As $k \rightarrow \infty$ in Eq. 3.3, the restored solution $\hat{f}_k$ will be closer to f in Eq. 3.2. The difference between the degraded image and the blurred version of the restored image is also used in the presented approach in this chapter.

In this study, the restoration is tried to be solved by modifying the weak membrane functional developed for surface reconstruction and edge detection which is explained in chapter 2 and in [2, 14, 5]. The blurring function is supposed to be known as in inverse filtering. The blurring mainly infects the discontinuities on an image. The edges are smoothed, corner points and T-junctions are rounded and details caused by discontinuities are disappeared. In this method, the discontinuities are tried to be enhanced, edges are sharpened, hence removing the blurring effect. A blurring term is added to the energy functional of weak membrane and the blurring effect is tried to be removed while keeping and enhancing the discontinuities by using line processes. The additive noise is suppressed by constructing piecewise smooth surfaces separated by line processes, and the surface reconstruction property of the method also reduces the ringing effect seen in many restoration methods.

3.2 Regularized Approach to Image Restoration

The ringing effect of restoration algorithms may be divided into two major parts:

1. Ringings near the boundary points of the image,
2. Ringings around the discontinuities on the image.

The ringing effect at discontinuities can be removed by using line processes, hence tearing the membrane. This situation separates smooth regions by line processes and prevents one region to be effected from the other one, which in turn disallows ringing at discontinuities. There are two major reasons to use line processes: The first one is to preserve discontinuities, and the other one is to prevent ringing effect of this type.

The membrane functional of the regularization theory is given in Eq. 2.7. The weak membrane functional emerging from this equation may be written as given in Eq. 2.11,

$$E = D + S + P \tag{3.5}$$

$$D = \sum_i \sum_j (f_{i,j} - d_{i,j})^2 \tag{3.6}$$

$$S = \lambda^2 \sum_i \sum_j (f_{i,j} - f_{i-1,j})^2 (1 - m_{i,j}) + (f_{i,j} - f_{i,j-1})^2 (1 - \ell_{i,j}) \tag{3.7}$$

$$P = \alpha \sum_i \sum_j (\ell_{i,j} + m_{i,j}). \tag{3.8}$$

This functional is modified for the purpose of restoration of blurred images.

3.2.1 Modified Energy Functional for Deblurring

The blurring function b is supposed to be known in this approach. This function corresponds to B in Eq, 3.1. The blurred image is supposed to be degraded by convolving it with this function. The function b is the space-invariant point-spread function (PSF). When the image f is degraded by b, the f in Eq. 3.6 becomes $f * b$. The blurred version of restored image ($f * b$) should be close to the input degraded image, when only blurring is considered [17]. This is the main idea of the proposed method.

The modified energy functional in Eq. 3.6 has its D term changed as [5]

$$D = \sum_i \sum_j ((f * b)_{i,j} - d_{i,j})^2. \tag{3.9}$$

When the neighbourhood interaction function is used as in Eq. 2.27 and Eq. 2.28, the weak membrane functional becomes

$$\begin{aligned} E &= \sum_{i,j} ((f * b)_{i,j} - d_{i,j})^2 + \\ &\quad \lambda^2 \sum_{i,j} g_{\alpha,\lambda}(f_{i,j} - f_{i-1,j}) + \lambda^2 \sum_{i,j} g_{\alpha,\lambda}(f_{i,j} - f_{i,j-1}) \end{aligned} \tag{3.10}$$

where $g_{\alpha,\lambda}$ is the neighbourhood interaction function, b is the space-invariant point-spread function, d is the degraded image and f is the reconstructed image [17]. α and λ are two real-valued parameters of weak membrane [2].

3.3 Discontinuity Preserving Surface Reconstruction for Deblurring

Local neighbourhood function g imposes smoothing by using α and λ parameters at continuous regions where discontinuous regions are selected by line

processes. There is a tradeoff between the first term and the term including interaction functions. The first term in Eq. 3.9 forces the solution to be close to the degraded input image. The term with g functions imposes smoothing operation while line processes ℓ and m shown in Eq. 3.9 try to break the membrane at discontinuous locations of the restored image. This breaking results in piecewise smoothing. Since the sharp changes are selected by line processes and smoothing is not performed at these locations, the restored image will be piecewise continuous. The sharp changes will not be smoothed but will be enhanced.

3.3.1 The Blurring Filter

The blurring function b, in general, can have any form and size. The method, in fact, does not put any burden on the shape of this filter. When b is taken as a low pass filter, the method works as an image deblurring system. Sample low pass filters are tried in this study. The first one if the averaging filter with size 3x3 and this filter may be shown as

$$b = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3.11}$$

This function may be a Gaussian function and may be written as

$$b(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.12}$$

Any other filter which does not need to be a blurring function and does not need to have a functional description can be used to implement for different purposes. For example, if the function b is taken to be a sharpening filter, then the solution of Eq. 3.10 is expected to be a piecewise smoothed version of the input image f.

3.3.2 Iterative Solution of The Modified Energy Functional

In the solution of the non-convex energy functional shown in Eq. 3.10, graduated non-convexity (GNC) algorithm is used. This algorithm is explained in detail in chapter 2. This algorithm uses an iterative approach to solve the functional. In each iteration, successive over relaxation method is used as in chapter 2. The iterations of this method may be written as

$$\begin{aligned} f_{i,j}^{(n+1)} &= f_{i,j}^{(n)} - \omega \, [\, \frac{\partial}{\partial f_{i,j}} ((f^{(n)} * b)_{i,j} - d_{i,j})^2 + \\ & g'_{\alpha,\lambda}(f_{i,j}^{(n)} - f_{i-1,j}^{(n+1)}) + g'_{\alpha,\lambda}(f_{i,j}^{(n)} - f_{i,j-1}^{(n+1)}) + \\ & g'_{\alpha,\lambda}(f_{i,j}^{(n)} - f_{i+1,j}^{(n)}) + g'_{\alpha,\lambda}(f_{i,j}^{(n)} - f_{i,j+1}^{(n)}) \,] \, \frac{1}{2 + 8\lambda^2} \end{aligned} \tag{3.13}$$

where $f^{(n)}$ is the image at nth iteration, λ is the regularization parameter, w is the relaxation parameter of SOR method, d is the degraded image, and $g'_{\alpha,\lambda}$ is the derivative of local interaction function w.r.t $f_{i,j}$ (see Chp. 2).

When the PSF is taken to be the averaging filter as in Eq. 3.11, the $\frac{\partial}{\partial f_{i,j}}(.)$ part of this equation will be [5]

$$\begin{aligned} \frac{\partial}{\partial f_{i,j}} [\frac{1}{9} \; (\; & f_{i-1,j-1} + f_{i-1,j} + f_{i-1,j+1} + f_{i,j-1} + f_{i,j} + f_{i,j+1} + \\ & f_{i+1,j-1} + f_{i+1,j} + f_{i+1,j+1}) - d_{i,j}]^2 = \\ 2\{ \; & \frac{1}{9}[f_{i,j} + f_{i,j+1} + f_{i,j+2} + \\ & f_{i+1,j} + f_{i+1,j+1} + f_{i+1,j+2} + \\ & f_{i+2,j} + f_{i+2,j+1} + f_{i+2,j+2}] + \\ & \ldots \; \ldots \; \ldots \\ & \ldots \; \ldots \; \ldots \\ & \frac{1}{9}[f_{i-2,j-2} + f_{i-2,j-1} + f_{i-2,j} + \\ & f_{i-1,j-2} + f_{i-1,j-1} + f_{i-1,j} + \\ & f_{i,j-2} + f_{i,j-1} + f_{i,j}] \\ & - d_{i,j} \; \}. \end{aligned} \tag{3.14}$$

This iteration is performed all over the image. But this equation must be modified at boundaries appropriately as specified in chapter 2. These iterations are terminated when a pre-defined convergence norm is reached or a maximum number of iteration count is performed. The convergence norm used in this study is the infinite norm.

3.3.3 Effect of The Blurring Filter on Iterative Equations

The structure of iterative equation will remain the same as in Eq. 3.13 for various PSFs. But the open form of the iterative equation will change dramatically. The part specified in Eq. 3.14 will be different. All of the f values will have corresponding coefficients of the blurring filter. In addition to the coefficients, the size of the filter may change. In this situation, this equation will have extra terms of f as the size grows larger than 3x3.

That part of the iterations may be expressed by defining a variable sized and variable coefficient molecule to implement different PSFs. For the averaging filter in Eq. 3.11, the molecule will be a square with each side having a length of 3, and each member having 1. If blurring function b is any other function, such as Gaussian with a changing σ, then the length of each side and coefficient at each corner will be a function of σ.

When the blurring function is represented by b with size $B_1 \times B_2$ (where $B_1 = B_2$ usually), the convolution term in Eq. 3.10 may be formulated as

$$(f * b)_{i,j} = \frac{1}{S} \sum_{m=-B_1/2}^{B_1/2} \sum_{n=-B_2/2}^{B_2/2} f_{i+m,j+n} \cdot b_{m+\frac{B_1}{2},n+\frac{B_2}{2}} \tag{3.15}$$

$$S = \sum_{p=0}^{B_1-1} \sum_{r=0}^{B_2-1} b_{p,r}. \tag{3.16}$$

By locating this equation in Eq. 3.10, the most general functional for $M \times N$ image is expressed as in the following equation

$$E = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(\frac{1}{S}\sum_{m=-B_1/2}^{B_1/2}\sum_{n=-B_2/2}^{B_2/2} f_{i+m,j+n}\cdot b_{m+\frac{B_1}{2},n+\frac{B_2}{2}} - d_{i,j}\right)^2 + \lambda^2\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} g_{\alpha,\lambda}(f_{i,j}-f_{i-1,j}) + \lambda^2\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} g_{\alpha,\lambda}(f_{i,j}-f_{i,j-1}). \quad (3.17)$$

The S term in Eq. 3.17 is calculated as specified in Eq. 3.16.

3.4 Implementation, Results and Discussions

The method is applied to several blurred synthetic and real images. The images are first blurred with a known blurring function and then the restoration algorithm is used to obtain the deblurred versions. Typical parameter values are $\lambda = 1.6$ and $\alpha = 1600$. These are experimental results obtained by using the averaging filter as PSF with real and synthetic degraded images.

Degraded checkerboard images are seen in the top row of Fig. 3.1 which are 128x128 sized images blurred two and three times by the averaging filter given in Eq. 3.11. In the last row, the corresponding restored images that are deblurred with the presented method are given. In Fig. 3.2, similar results are given for 256x256 House image. In Fig. 3.3, the degraded images are obtained by first blurring by the same averaging filter and then adding Gaussian noise with SNR=25dB. The variance of the zero mean Gaussian noise added onto the blurred images is calculated by

$$SNR_{dB} = 10\lg_{10}\left[\frac{\frac{1}{N^2}\sum_{i,j}(f_{i,j}-\mu_f)^2}{\sigma_n^2}\right] \quad (3.18)$$

where μ_f is the mean of the input image onto which noise is to be added and σ_f^2 is the variance of this image. The variance of the Gaussian noise is calculated by

$$\sigma_n^2 = \frac{\sigma_f^2}{10^{SNR_{dB}/10}}. \quad (3.19)$$

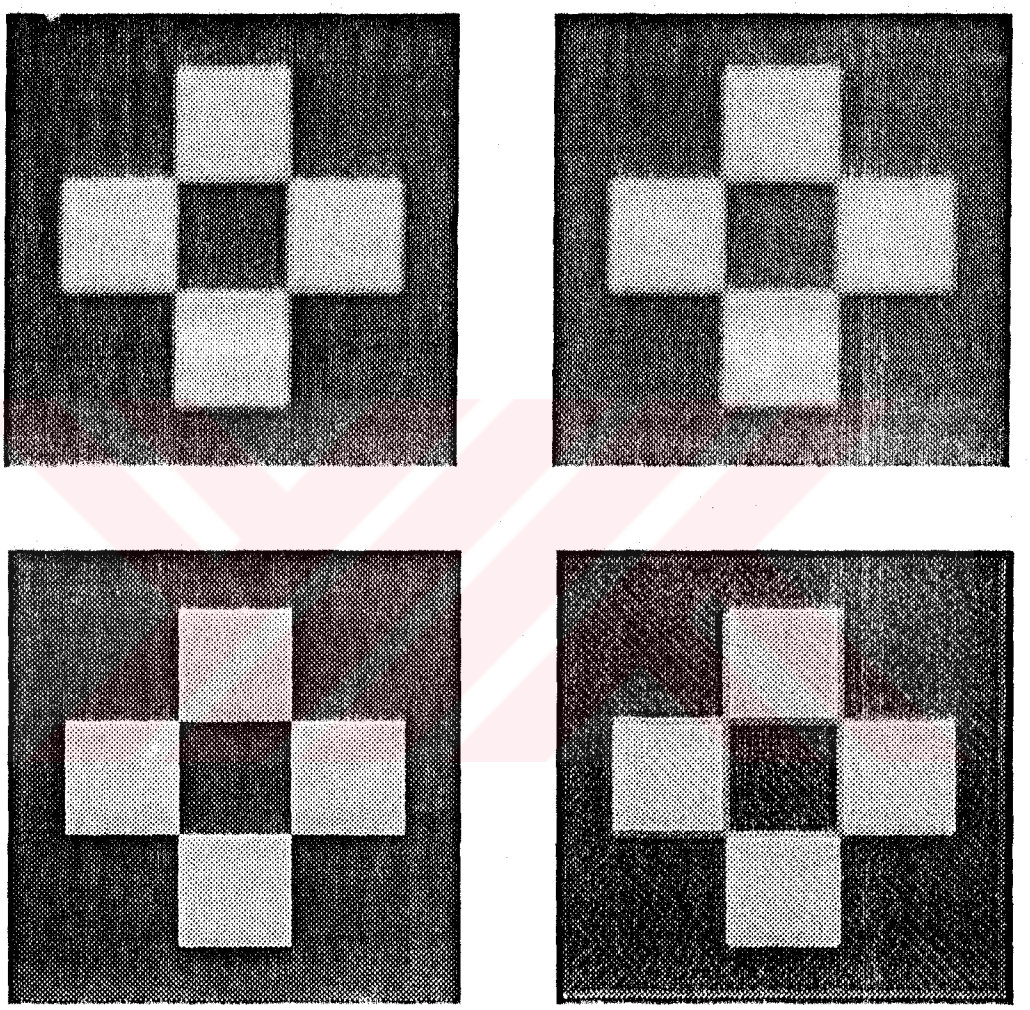


Figure 3.1. Top row: The blurred checkerboard images degraded by applying a 3x3 averaging filter two (left) and three (right) times. Bottom row: Restored versions of the images.

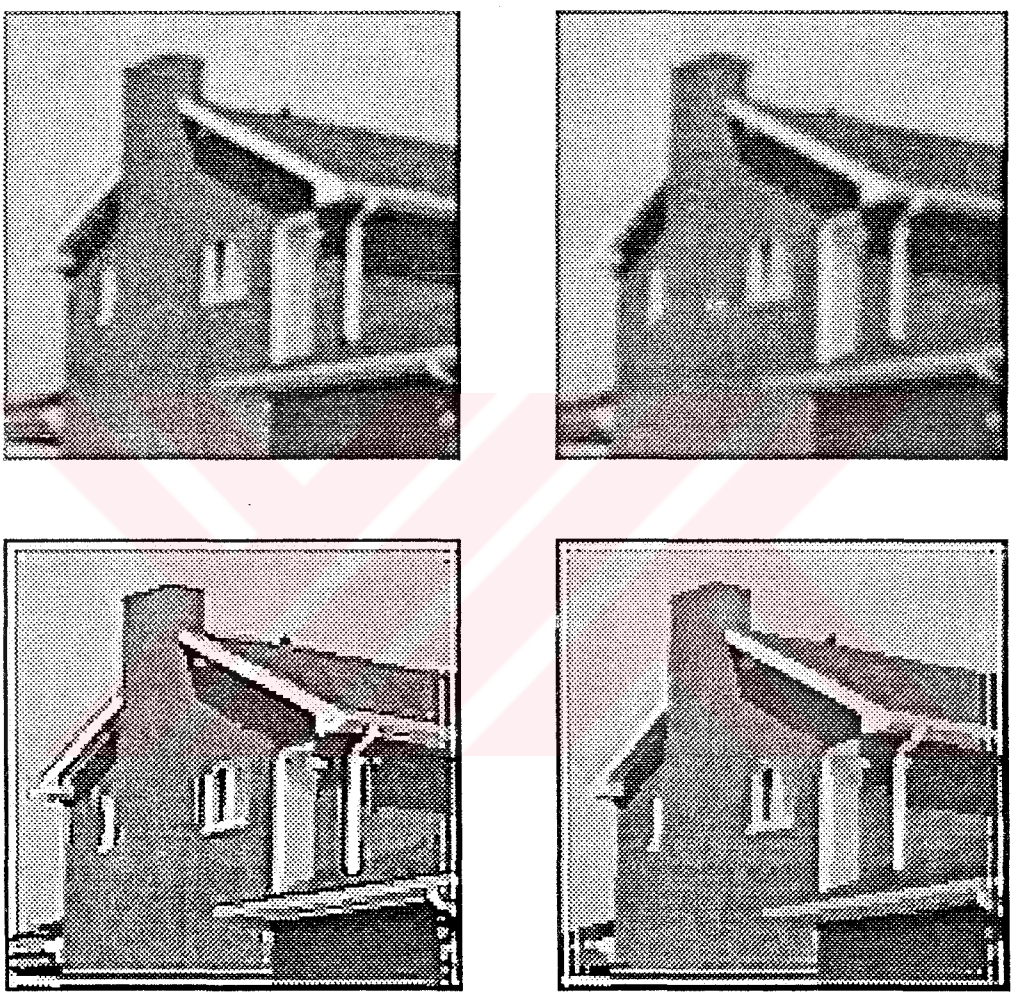


Figure 3.2. Top row: The blurred house images degraded by applying a 3x3 averaging filter two (left) and three (right) times. Bottom row: Restored versions of the house images.

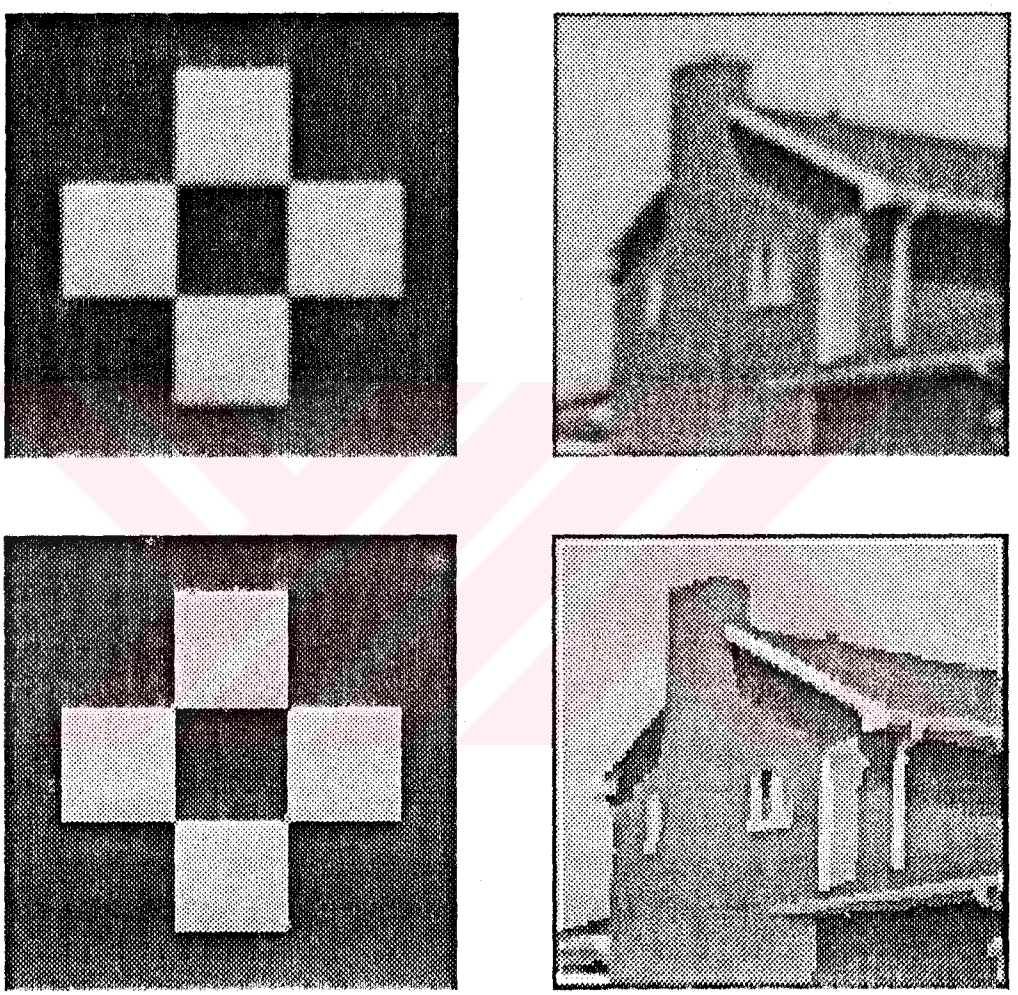


Figure 3.3. Blurred and noise added (SNR=25 dB) checkerboard and house images and their restored versions.

Table 3.1. Normalized mean square error values for blurred and noise added checkerboard and house images.

Image	-		SNR=30 dB	
	Gauss σ_n	NMSE	$Gauss\sigma_n$	NMSE
che1	-	133	7	6563
che2	-	22	7	6509
che3	-	113	6.9	6664
house1	-	798	5.5	8283
house2	-	544	5.4	8222
house3	-	570	5.3	8283
clock1	-	1314	5.6	6533
clock2	-	852	5.5	6501
clock3	-	902	5.5	6723

In Table 3.1, normalized mean square error values are calculated between the original and the restored images from their degraded ones. The sample images are checkerboard (che), house and clock images. The numbers written after the name of the images denote how many times the image is blurred by the averaging filter in Eq. 3.11. From the NMSE values of the results given in Table 3.1, it can be seen that the method works better for two times blurred images. For the noisy images, the NMSE values are almost the same and the method usually amplifies noise if the parameters of weak membrane model are not chosen carefully. It is observed that λ and α parameters have significant effect on the restored image quality and the number of SOR iterations. Even a very small change results in a blurred or noisy restored image and causes the SOR iteration count to grow too large. For example, a 0.1 change in λ value in the restoration of checkerboard image shown in Fig. 3.1 rises the SOR iteration count from 50 to 260. When λ is smaller than 1, it's observed that a projection onto convex set operation is needed for the calculated f values. Otherwise, the calculated value ($f^{(n+1)}$) grows larger than 255 for an 8 bit image. This projection p may be expressed for an 8-bit image as

$$p(f_{i,j}) = min(255,\ max(f_{i,j}, 0)) \tag{3.20}$$

or with a q-bit image

$$p(f_{i,j}) = min(2^q - 1,\ max(f_{i,j}, 0)). \tag{3.21}$$

This operation prevents the calculated f values to stay in meaningful pixel value range especially for small λ values.

The Figures 3.1, 3.2, 3.3 and Table 3.1 show that the restoration using weak membrane modeling successfully enhances discontinuities even in the existence of noise. The blurring effect can be removed, noise can be suppressed, discontinuities on the image can be preserved and ringing effect seen in linear restoration methods can be eliminated in the image. The preservation of discontinuities and elimination of ringing effect is realized by using line processes of weak membrane model.

The main disadvantage of this method is the extravacant sensitivity of noise upon the parameters. In this case, the reasonable results can be obtained for only a narrow range of weak membrane parameters. The values that the parameters can have cannot be chosen arbitrarily and it's found to be hard to find a good parameter set for general usage for noisy images. The main advantage is the removal of ringing effect near the discontinuities which is seen in many of the restoration methods.

CHAPTER 4.

IMAGE COMPRESSION USING WEAK MEMBRANE MODEL OF IMAGES

4.1 Introduction

Discontinuous regions in a scene carry important information to identify and locate the objects from the image data. These boundaries manifest themselves as sharp changes in image intensities [8, 10, 13]. Regions between these discontinuities are smooth and pixel values don't represent sharp changes. This allows us to determine the pixel values of smooth regions by using data along edges which present at discontinuities. In a 2D image, the edge data correspond to a very sparse image where data exist only along the edge points. The non-edge data may be reconstructed by using an appropriate surface reconstruction algorithm which can cope with very sparse data and this sparse data consist of the pixel values on both sides of a discontinuity.

The coding and decoding parts are built on a surface reconstruction algorithm. The weak membrane modeling is used in both parts. Using the same energy functional derived from weak membrane modeling for both coding and decoding results in a compact algorithm. The scheme introduced in this chapter can be synthetically divided into two separate parts where the first part is the coding and the second is the decoding part. The coding part is again divided into two consequent sections: In the first section, edge detection is performed via line processes using weak membrane model of image [2, 14] as explained in

chapter 2. Then the binary valued line processes are run-length coded. In the second section, data around the line processes are coded using entropy coding.

The decoding part accepts 2D sparse data and line processes as input. In this part, encoded data is located by using line processes which are obtained by encoding the run-length coded line processes. Then, this data is used to construct surface between edge points identified by line processes. Thus, the surface reconstruction algorithm must be able to work with sparse data, and must perform piecewise continuous reconstruction while preserving discontinuities.

4.2 The Coding Part

This part is divided into two sections in itself. In the first section, input image is modeled by weak membrane and discontinuous points are detected by line processes as explained in chapter 2. Then two separate information are coded to reconstruct the image at the receiving side. These information are the binary valued line processes and pixel values at both sides of the discontinuities.

4.2.1 Weak Membrane Modelling With Sparse Data

Weak membrane modeling is a discontinuity preserving surface reconstruction method and produces very smooth and continuous edges. Edges are located by a boolean valued array called *line processes*. This model is expressed by a non-convex energy functional [2]. The following energy functional has a slightly different notation from Eq. 2.11:

$$E = D + S + P \tag{4.1}$$

$$D = \sum_i \sum_j (d_{i,j} - u_{i,j})^2 \tag{4.2}$$

$$S = \lambda^2 \sum_i \sum_j (1 - \ell_{h_{i,j}})(u_{i,j} - u_{i-1,j})^2 + (1 - \ell_{v_{i,j}})(u_{i,j} - u_{i,j-1})^2 \tag{4.3}$$

$$P = \alpha \sum_i \sum_j (\ell_{v_{i,j}} + \ell_{h_{i,j}}) \tag{4.4}$$

where, d is the input and u is the reconstructed image, ℓ_v and ℓ_h are boolean valued vertical and horizontal line process arrays respectively. λ is the scale parameter, prescribing the smoothness and edge occurrence threshold together with α which is the penalty coefficient for P term. D is faithfulness to input data, S expresses the deformation or smoothness of reconstructed surface, P is the penalty introduced into the functional by assigning a point as an edge.

Surface reconstruction is performed by finding u which minimizes this non-convex functional in Eq. 4.1. During the minimization process, D term assures that the reconstructed surface u will be close to the original one, d. Smoothing is imposed by the second term S, by using the first order derivatives. Discontinuities are preserved by preventing the smoothing operation at edge points where edges are identified by $\ell_v = 1$ or $\ell_h = 1$, which signifies that the membrane is torn at these locations. The penalty introduced into the overall energy functional by assigning a point as an edge point is maintained by the last term P which prevents all line processes to become 1. Edges are the points where line processes have the value 1, and in smooth regions line processes resemble the value 0.

In minimising this non-convex functional, graduated non-convexity (GNC) algorithm is used [2, 14]. This algorithm uses a modified version of the energy functional given in Eq. 4.1 where the approximation of modified functional to non-convex one is controlled by a parameter. A comprehensive study on this method may be found in chapter 2.

In addition to producing continuous edge segments, this method can be used for both edge detection and surface reconstruction from sparse data. To handle the sparse data, the weak membrane functional may be expressed as:

$$E = D + S + P \tag{4.5}$$

$$D = \sum_i \sum_j \beta_{i,j}(d_{i,j} - u_{i,j})^2 \tag{4.6}$$

$$S = \lambda^2 \sum_i \sum_j (1 - \ell_{h_{i,j}})(u_{i,j} - u_{i-1,j})^2 + (1 - \ell_{v_{i,j}})(u_{i,j} - u_{i,j-1})^2 \tag{4.7}$$

$$P = \alpha \sum_i \sum_j (\ell_{h_{i,j}} + \ell_{v_{i,j}}) \tag{4.8}$$

where,

$$\beta_{i,j} = \begin{cases} 1, & \text{if data is available at } (i,j), \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

The sparse information marked by β can be coded efficiently. The information is only a portion of the entire image.

4.2.2 Sparse Information for Surface Reconstruction

An edge point means that a discontinuity exists at that spatial location and these discontinuities are denoted by two line process arrays. Each line process ($\ell_{h_{i,j}}$ and $\ell_{v_{i,j}}$) resides on one side of a discontinuity which resulted from two very different pixel values (Eq. 4.5). To reconstruct pixel values on either side of a discontinuity, $u_{i,j}$ and $u_{i-1,j}$ for $\ell_{h_{i,j}}$ and $u_{i,j}$ and $u_{i,j-1}$ for $\ell_{v_{i,j}}$ are needed. The location of line processes and related pixels are illustrated in Fig. 4.1. The surface reconstruction algorithm cannot construct surfaces lying on both sides of an edge point without knowing both of these pixel values on these surfaces.

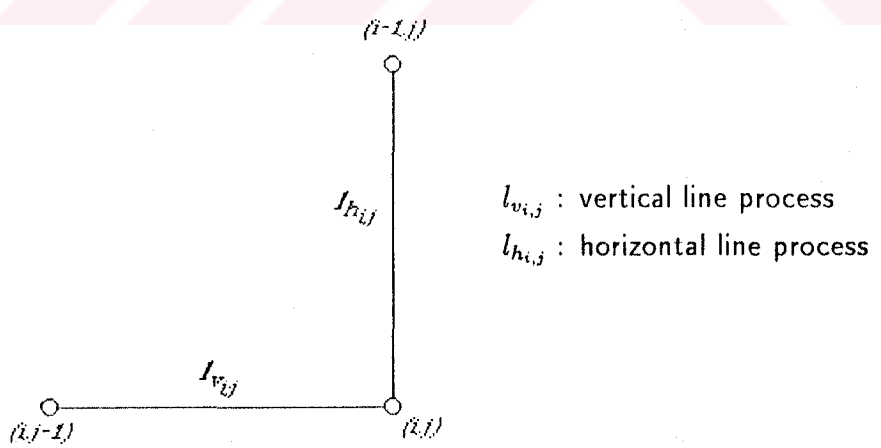


Figure 4.1. Line processes in two dimensions.

4.2.2.1 Line Processes – Rectangular Array Formation

There are two sort of discontinuity, one vertical ($\ell_{v_{i,j}}$), and one horizontal ($\ell_{h_{i,j}}$) line process (Fig. 4.1). A vertical discontinuity means that there is a sharp intensity change from pixel $(i, j-1)$ to (i, j), resulting corresponding line process to become 1, that is $\ell_{v_{i,j}} = 1$. Similarly, A horizontal discontinuity means, that there is a sharp intensity change from pixel $(i-1, j)$ to (i, j), resulting corresponding line process to become 1, that is $\ell_{h_{i,j}} = 1$. Using this kind of two separate line process arrays results in a hexagonal grid of line processes for the final image. This is illustrated in Fig. 4.2.

To obtain a rectangular line process array with the same size as the input image, the point (i, j) is marked to be an edge point if any one of $\ell_{h_{i,j}}$ or $\ell_{v_{i,j}}$ represents a discontinuity. Then, these points are denoted by a single line process array k. The components of k may be computed by simply applying the OR operation to horizontal and vertical line processes, i.e. $k_{i,j} = \ell_{h_{i,j}}\ OR\ \ell_{v_{i,j}}$. The S term in Eq. 4.7 may now be expressed as,

$$S = \lambda^2 \sum_i \sum_j (1 - k_{i,j})[(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2] \tag{4.10}$$

where,

$$k_{i,j} = \begin{cases} 1, & (\ell_{h_{i,j}} = 1)\ OR\ (\ell_{v_{i,j}} = 1), \\ 0, & \text{otherwise.} \end{cases} \tag{4.11}$$

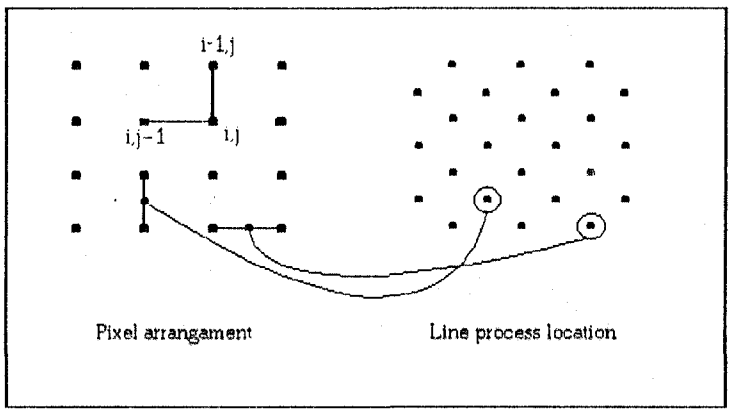


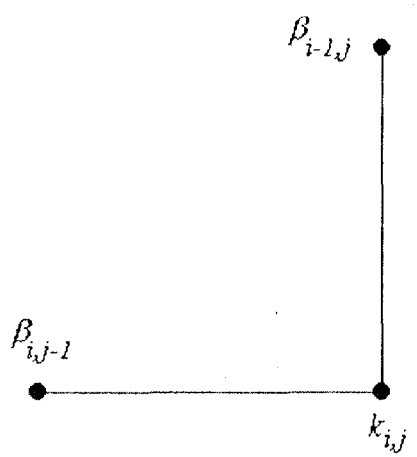
Figure 4.2. Hexagonal grid formation of line processes.

This form attaches just one line process to both of the directional first order derivatives (Fig. 4.3). This results in a rectangular line processes array instead of an hexagonal grid occurred with vertical and horizontal ones. Then, the sparse data may be marked by the rectangular line processes where existing data is marked by line process with value of 1.

4.2.2.2 Marking Sparse Data

If a point on the line process array k is 1 at (i, j), there is a discontinuity and three data values at (i, j), $(i-1, j)$, and $(i, j-1)$ are needed in the surface reconstruction step (Fig. 4.3). The information to code may now be summarized as below:

- Rectangular grid array of line processes which denotes the location of edges. This is a boolean valued 2D array.
- For each line process with a value of 1, i.e. $k_{i,j} = 1$, three data (pixel) values as,
  - *i1.* the current pixel at (i, j),
  - *i2.* neighbouring pixel at $(i-1, j)$ and,
  - *i3.* neighbouring pixel at $(i, j-1)$.



$$\beta_{i,j} = \begin{cases} 1, & k_{i+1,j} = 1 \text{ or} \\ & k_{i,j+1} = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (4.12)$$

Figure 4.3. Extraction of β from k.

Locations with value 1 on the line process array k describes locations of edges, hence the first item. Pixels for $i1$ are marked by $k_{i,j} = 1$, and data for $i2$ and $i3$ are marked by $\beta_{i-1,j} = 1$ and $\beta_{i,j-1} = 1$ respectively (Fig. 4.3). Line process array k is used to locate discontinuities, and β is used to mark pixels which caused discontinuities.

In this way, location of pixels at both sides of discontinuous regions are made available for reconstruction at the decoder. To find a single array to locate sparse data, we use arrays k and β. k may be obtained from ℓ as defined in Eq. 4.11. β array may be extracted uniquely from k as in Fig. 4.3. Then, k is the least space occupying one that can be used to locate discontinuities and needed pixels.

4.2.2.3 The Energy Functional

The energy functional at the coding part deals with non-sparse data, which means that there should be no β term in the functional. The energy functional to be minimized is expressed by combining Eq. 4.5, Eq. 4.10 and Eq. 4.11:

$$E = D + S + P \tag{4.13}$$

$$D = \sum_i \sum_j (d_{i,j} - u_{i,j})^2 \tag{4.14}$$

$$S = \lambda^2 \sum_i \sum_j (1 - k_{i,j}) \cdot [(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2] \tag{4.15}$$

$$P = \alpha \sum_i \sum_j k_{i,j}. \tag{4.16}$$

The functional in Eq. 4.13 is non-convex and the minimization method used for this functional is explained in chapter 2. Graduated non-convexity together with SOR is used to find the solution for this functional. There are two arrays found after minimization: One is the image array u which is the reconstructed surface. The other one is the line process array k which denotes the locations of discontinuities. The coded line processes and sparse data are the two outcomes of this functional.

4.2.3 Coding of Line Processes and Data

The line process and data array are coded separately. The line process array is composed of ones and zeros, that is, one bit is enough to code each member of line process array. For an image with size $M \times N$, the line process array occupies $M \times N$ bits. Another property of line process array is that it is very sparse; most of the entries are zero. That means, there are long runs of zeros in this array. This is the reason of using run-length coding for line process array. Since computation of β array from line processes k is unique (Fig. 4.1), coding of only line processes is enough to obtain the following $\hat{\beta}$ array at the decoder part by Eq. 4.17. $\hat{\beta}$ array marks the locations of needed data and defined as

$$\hat{\beta}_{i,j} = (k_{i,j} = 1) or (\beta_{i-1,j} = 1) or (\beta_{i,j-1} = 1). \tag{4.17}$$

The data portion of the information which is located along the boundaries is sparse. This data is coded using dynamic entropy coding.

4.2.3.1 Run-Length Coding of Line Processes

Each row of the line process array is coded separately, that is, the next row is not treated as the continuation of the current row. Each entry of the line process array is represented by either 0 or 1. Thus, in coding of the binary valued line processes the value is not sent. But only the number of same valued bits are coded in a stream composed of consequent zeros or ones. When the coder–decoder couple has agreed on the starting bit value (either 0 or 1), at arrival of each number, the value is toggled from 0 to 1 or vice-versa. In this way, only the run-length is coded. The maximum allowed size of this stream, which is the blocksize, is simply $2^m - 1$, where m is the number of bits reserved to code this number. To find the optimal blocksize which gives the smallest size of coded line processes, we've tried for a range of blocksizes and chosen the best one which requires the smallest space to code this array. The blocksize is retained at the decoder side by sending m in one byte. Once optimal blocksize is found, this is used for the whole line process array.

Whenever the actual length of the same valued bit stream exceeds the blocksize, the number represented by blocksize is sent. Then a zero is inserted and the counter is reset. The zero causes the line process value to be toggled without receiving a new information. The next number toggles the value once more which returns back to the one before zero is sent. The remaining part of this length is coded in a similar manner. The space wasted by this method is m, which is the number of bits required to represent blocksize. Since the length of the coded line processes is calculated including this kind of wastings, the blocksize used is still the least space occupying one.

4.2.3.2 Huffman Coding of Sparse Data

Once $\hat{\beta}$ array is calculated as in Eq. 4.17, location of pixels to be coded is known. By locating this data on the input image with $\hat{\beta}$ array, all available 2D data where $\hat{\beta}_{i,j} = 1$ is converted to an 1D array. This is a reversible process, since this 1D data can be located onto two dimensional image array by using $\hat{\beta}$ array by scanning the image in the same order.

The 1D array obtained in this way is coded by entropy coding (or Huffman coding) [15]. For each image, a separate coding tree is built, which means that dynamic Huffman coding is used. This coding method does not yield low bit rates for the coded image. In most of the 8-bit real images, the bit rate of entropy coded image has been around 7. The reason of this is that, the data has a broad spanning range, i.e. it has nearly an homogeneous histogram distribution. Data to code are the pixels causing discontinuities, which means that two neighbouring values are too much different. This property results high bit rates without a convenient preprocessing of edge data.

4.3 The Decoding Part

The decoding part is built on a surface reconstruction algorithm. The input is composed of two separate data parts: The first is a very sparse 1D data containing the pixel values along edges. The second part is a binary valued 2D array which is

Table 4.1. The decoding algorithm.

- Retain number of rows and columns of the image and the blocksize.
- Decode run-length coded line processes onto k array.
- Extract $\hat{\beta}$ from k by Eq. 4.3 and Eq. 4.17.
- Decode one pixel value from entropy coded data whenever $\hat{\beta}_{i,j} = 1$ and locate this data at (i, j) on the image array.
- Perform surface reconstruction from sparse data by weak membrane modeling using Eq. 4.25 and Eq. 4.26.

used to locate the sparse data on 2D image array. When sparse data is located on image array, surface reconstruction algorithm based on weak membrane modeling is applied and image is reconstructed. The surface reconstruction method is the one used in the coding part with the addition of sparsity.

4.3.1 The Main Algorithm

At the receiver part, just the reverse process is done for decoding. Run-length coded line processes are extracted and $\hat{\beta}$ array is calculated from the line process array k as in Eq. 4.3. 1D data is decoded and converted to 2D array by locating the exact co-ordinates with the help of $\hat{\beta}_{i,j}$ values: Pixel value is assigned to zero if $\hat{\beta}_{i,j}$ is zero, or a non-zero pixel value is extracted from 1D data when $\hat{\beta}_{i,j} = 1$.

After locating the available data, surface reconstruction from sparse data is realized to construct surfaces between edges.

4.3.2 Extraction of Line Processes and Data

Line process array is obtained by encoding the run-lengths of zeros or ones. Each row is encoded separately. In most of the images, the boundaries need

special treatment, and except pixels on boundaries, probability of edge existence is very small at the outer pixels. For this reason, each row is assumed to start with zero in the implementation. The boundaries are studied in a separate section below.

After line process array, k, has been encoded, $\hat{\beta}$ array is calculated by using Eq. 4.3. Line processes k and sparse data indicator β mark where data exist for surface reconstruction. That's why, these two arrays are combined into a single array $\hat{\beta}$ by Eq. 4.17. This array will be used in the surface reconstruction step explained below.

The sparse data is located by using this $\hat{\beta}$ array by the method given in the main algorithm on Table 4.1. The values of input data which is denoted by d array, will not be modified by the surface reconstruction algorithm furthermore. $\hat{\beta}$ array is used to satisfy this, too.

4.3.3 Boundary Conditions

In general, boundaries do not represent edge information. In the surface reconstruction scheme from sparse data, boundary pixels must be known. If the boundaries are not known, then they are initialized with any value, and presumably with zero. In the case of iterative solution of weak membrane functional, propagation from the nearest available data will take too much time, and even with the tightest convergence criteria, boundaries will still remain dark, if initialized with zero. If a boundary point takes any other value as the initial value, that value won't change if there is not sufficiently near available data to that boundary point. To prevent this, all pixel values at the boundaries are coded. This boundary condition is satisfied by simply assigning 1 to boundary locations on $\hat{\beta}$ array in the decoding part.

As a boundary condition, the four corners and four sides of the image are known to have one in $\hat{\beta}$ array, in advance. That's why, the line processes at the boundaries need not be coded. The size of the line process array to code is thus reduced from $M \times N$ from $(M-2) \times (N-2)$. This is used to obtain the sample

images in this study.

SOR iterations to minimize regular membrane functional need special treatment for boundaries as well as corner points. Assigning 1 to $\hat{\beta}$ at boundaries eliminates this kind of special treatment to boundaries in the reconstruction step, reducing code size as well as time consumed during iterations. The only disadvantage is that data values at the boundaries must also be coded.

4.3.4 Image Reconstruction From Sparse Data

To reconstruct the image at the decoder part, weak membrane modeling is used. The weak membrane modeling is expressed by a non-convex energy functional as given in Eq. 4.5 and in Eq. 4.20. The solution of this equation by GNC, u, is the reconstructed image. SOR (*Successive Over Relaxation*) is used in the solution of energy equation in Eq. 4.5 by GNC. By solving the Euler-Lagrange equation, this iterative scheme is obtained,

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} - \frac{\omega}{T} \cdot \frac{\partial E}{\partial u_{i,j}}, \tag{4.18}$$

where ω is the relaxation parameter and $\omega \in [0, 2]$. This parameter controls convergence together with T and a good selection for ω is found to be 1.25 [18]. T is defined to be $2(\beta_{i,j} + 4\lambda)$ [2]. n denotes the iteration number in increasing order. By taking derivatives and rearranging we obtain,

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} - \lambda^2 \frac{\omega}{2(\beta_{i,j} + 4\lambda^2)} \cdot \left[4u_{i,j} - (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})\right] \tag{4.19}$$

where $\beta_{i,j} = 0$, and $u_{i,j}^{(n+1)} = d_{i,j}$ where $\beta_{i,j} = 1$. At the boundaries β array is 1; this means that pixels at boundaries will not be modified. Thus boundary conditions tell us that SOR iterations will not be performed at boundaries. This condition result in a single SOR iteration as given in Eq. 4.19.

With k in hand and β extracted from k, energy equation for weak membrane in Eq. 4.5 becomes

$$E = D + S + P \tag{4.20}$$

$$D = \sum_i \sum_j \beta_{i,j}(d_{i,j} - u_{i,j})^2 \tag{4.21}$$

$$S = \lambda^2 \sum_i \sum_j (1 - k_{i,j}) \cdot [(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2] + \tag{4.22}$$

$$P = \alpha \sum_i \sum_j k_{i,j}. \tag{4.23}$$

By investigating the modification conditions of SOR iterations, the energy functional will be reduced in the following section. This reduced functional will make the solution easier than the regular iterations used in the graduated non-convexity algorithm in chapter 2.

4.3.5 The Reduced Energy Functional

During the SOR iterations, pre-existing data should not be modified and iterative calculations must be performed only for non-existing data in Eq. 4.19. Similarly, line process array k and β array are not modified, either. If goal is to construct surface from sparse data, it is not important whether the data is at an edge location or in the middle of a smooth region. The important thing is to locate and use available data for reconstruction. The available data is indicated by either β or k array, and it makes no difference whether data is marked by either β or k. The molecule of a membrane has a structure with five elements. This molecule has its center and the four connected components which are only one pixel apart as illustrated in Fig. 4.4. This means that any molecule cannot contain pixels in different regions separated by edges, each pixel in one molecule belongs to the same region. Thus an ordinary sparse membrane functional is enough for reconstruction.

The P term in Eq. 4.20 is always constant since line process array k won't change, so P term can easily be eliminated from the minimization step. D term is

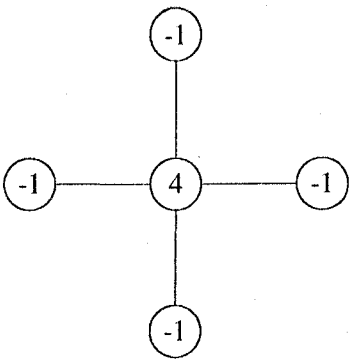


Figure 4.4. Regular membrane molecule.

not processed at available data locations to satisfy $u_{i,j} \equiv d_{i,j}$ where $\hat{\beta}_{i,j} = 1$. This term is not processed for unavailable data either, since no $d_{i,j}$ exists when $\beta_{i,j} = 0$. These two conditions eliminates D term from energy functional. In advance, it's known that surfaces on the regions of unavailable data will be smooth, which indicates S term in Eq. 4.20 to be sufficient for reconstruction.

Hence, only the S term of Eq. 4.20 is left which includes the first order derivatives imposing smoothness, which is what we are looking for. The S term alone results in a yet incomplete functional:

$$E = \lambda^2 \sum_i \sum_j (1 - k_{i,j}) \cdot [(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2]. \quad (4.24)$$

But, using k alone does not provide the restriction for modification based on $\hat{\beta}$ array in Eq. 4.19, which is the reason of incompleteness of equation in 4.24. To satisfy this restriction, $\hat{\beta}$ is used instead of k in Eq. 4.24 which can be obtained by Eq. 4.17. The energy functional then becomes

$$E = \lambda^2 \sum_i \sum_j (1 - \hat{\beta}_{i,j}) \cdot [(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2] \quad (4.25)$$

which means that updating of pixels during iterations is performed only when

$\hat{\beta}_{i,j} = 0$. The SOR iteration in Eq. 4.19 now becomes

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} - \frac{\omega}{8} \cdot [4u_{i,j} - (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})] \tag{4.26}$$

where $\hat{\beta}_{i,j} = 0$ to construct pixel values of unavailable data locations.

4.4 Results and Discussions

The prescribed method is applied to various synthetic and real images. All of the images are coded and encoded using the same parameter values. λ, α and SOR-ω are taken the same values for all of the images for easy comparison between different images. In the coding part, λ is 0.8, α is 250 and SOR-ω is 1.2. In the decoding part, λ is 6 and SOR-ω is 1.2. α is not used in the decoding part.

Various information including the sizes, blocksize and compression ratio of different images is presented in Table 4.2. The first column shows the name of sample images where their respective sizes are given in bytes in the second column. In the compression size section of the table, the first column gives the number of bits used for blocksize. The blocksize may be thus calculated by $2^m - 1$ where m is the run-bits given in the table. The line proc. column is the coded size of the line processes.

Table 4.2. Size and compression ratio information of coded images.

		Compressed Size (bytes)						Compression		
Image	Size	Run bits	Line proc.	Data entropy	Data size	Coded Data	Total	Ratio	%	NMSE
Che.	16kb	6	527	0.876	1258	188	715	23:1	96	62
Bars	16kb	6	708	0.942	1636	234	942	17:1	94	142
House	64kb	6	3153	7.538	8613	9037	12190	5.3:1	81	313
Lenna	64kb	4	5203	7.572	14004	14663	19866	3.2:1	70	238
Clock	64kb	5	4121	7.607	10302	11028	15149	4.3:1	77	4998
Brain	30kb	3	3522	6.281	14693	11954	15476	2:1	50	89

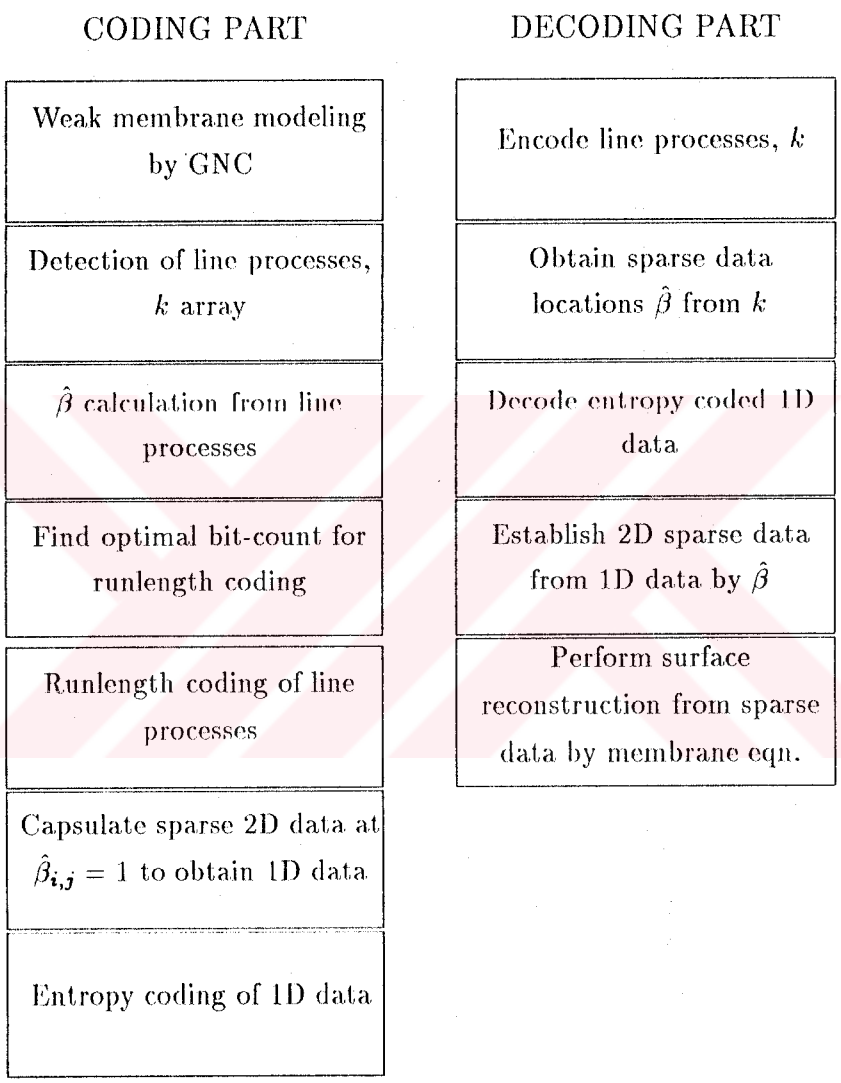


Figure 4.5. Phases of coding (left) and decoding (right) parts.

The data size given is the number of pixel values to code, or the number $\hat{\beta}$ locations with value of 1. Data entropy is the value calculated in the entropy coding part. Coded data is the entropy coded data size and can roughly be calculated by multiplying the entropy by the data size. The total size is given after run-length and entropy coding is done and is the size of the whole compressed image. The ratio section shows simply the ratio of the original image to the compressed images size. The compression percentage is calculated by

$$Percentage = \frac{I_s - C_s}{I_s} \tag{4.27}$$

where I_s denotes the size of the original image and C_s shows the size of the compressed image. NMSE is calculated between the original input image and the reconstructed image at the decoder part.

The overall flow diagram of the presented algorithm is illustrated in Fig. 4.5. The left side of the figure shows the coding part and the right side shows the decoding part.

There are checkerboard and bars images in the synthetic image set. The original synthetic images are shown in the upper row in Fig. 4.6. The images in the lower row are reconstructed versions of the coded ones. From the investigation of reconstructed checkerboard image in this figure, it can be seen that in continuous regions, there are circular slightly darkened areas. This kind of regions appear at locations residing apart from the discontinuities. During the SOR iterations of surface reconstruction, The propagation of pixels to this kind of places will take time, but the convergence criteria will be satisfied before this propogation is fully completed. A similar sinking efect is also seen in the reconstructed bars image. The large darkened circular region near the upper left corner of clock image is also a sinking effect.

These figures show that, the effect of sinking is proportional to the distance of this region to the discontinuities. In this figures, the locations corresponding to $\hat{\beta}_{i,j} = 0$ are filled with zeros before the reconstruction begins. If these locations are filled with all ones, then the sinking effect will be reversed and these regions will be brighter. One solution to prevent this kind of sinking effects, the locations

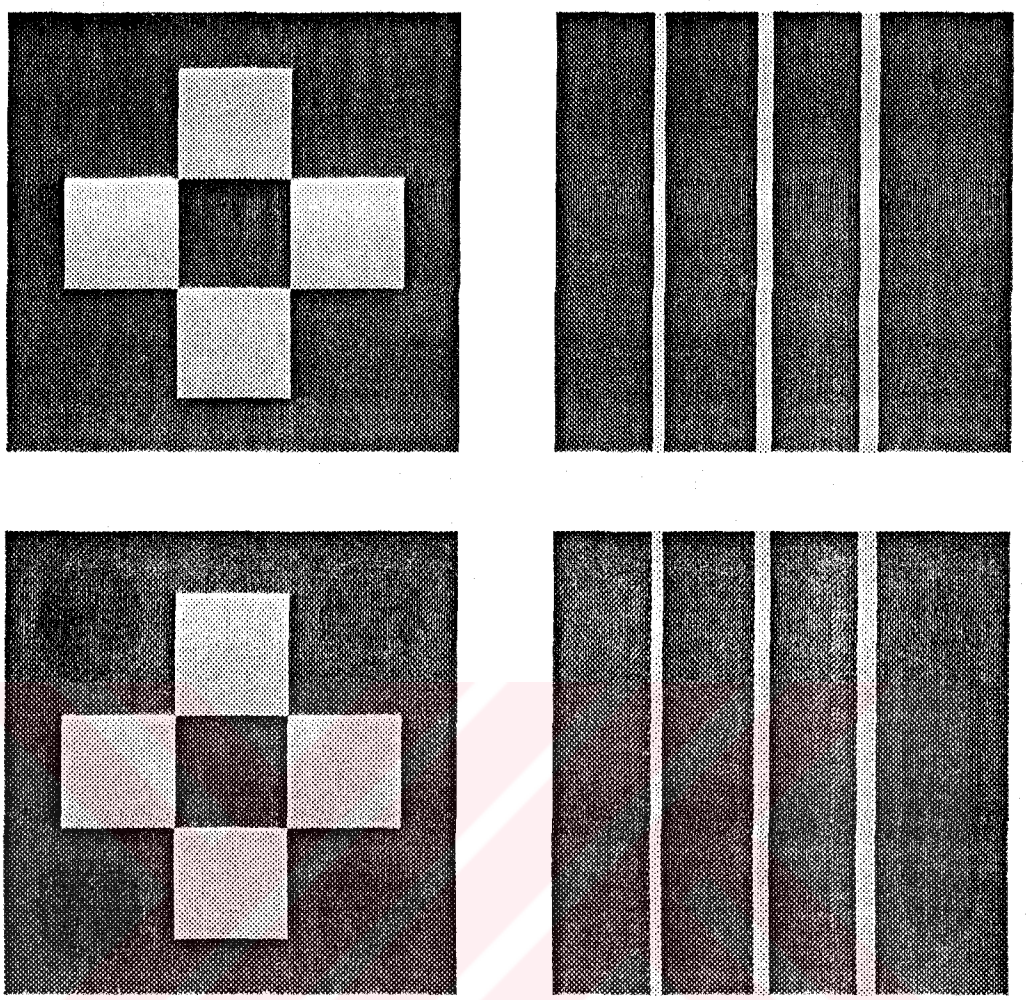


Figure 4.6. First row: Original 128x128 checkerboard and bars images. Second row: Reconstructed versions of checkerboard and bars images.

at $\hat{\beta}_{i,j} = 0$ may be filled by the local mean. But the locality cannot be taken constant over the image as well as for different images. The windows used to calculate the mean must include sufficiently large number of existent data.

In this study, data to be coded is used without any preprocessing. This is the main reason of a rather lower compression ratio. A good approach to make the histogram of this data more narrow is to perform a differentiation along the edges. This transformation should be done along the edges, since at the edges two neighbouring pixel values differ too rapidly. But, since an edge separates two different pixel values and the edges are, in general, smooth and continuous, subtraction process should give very small values which are gathered around

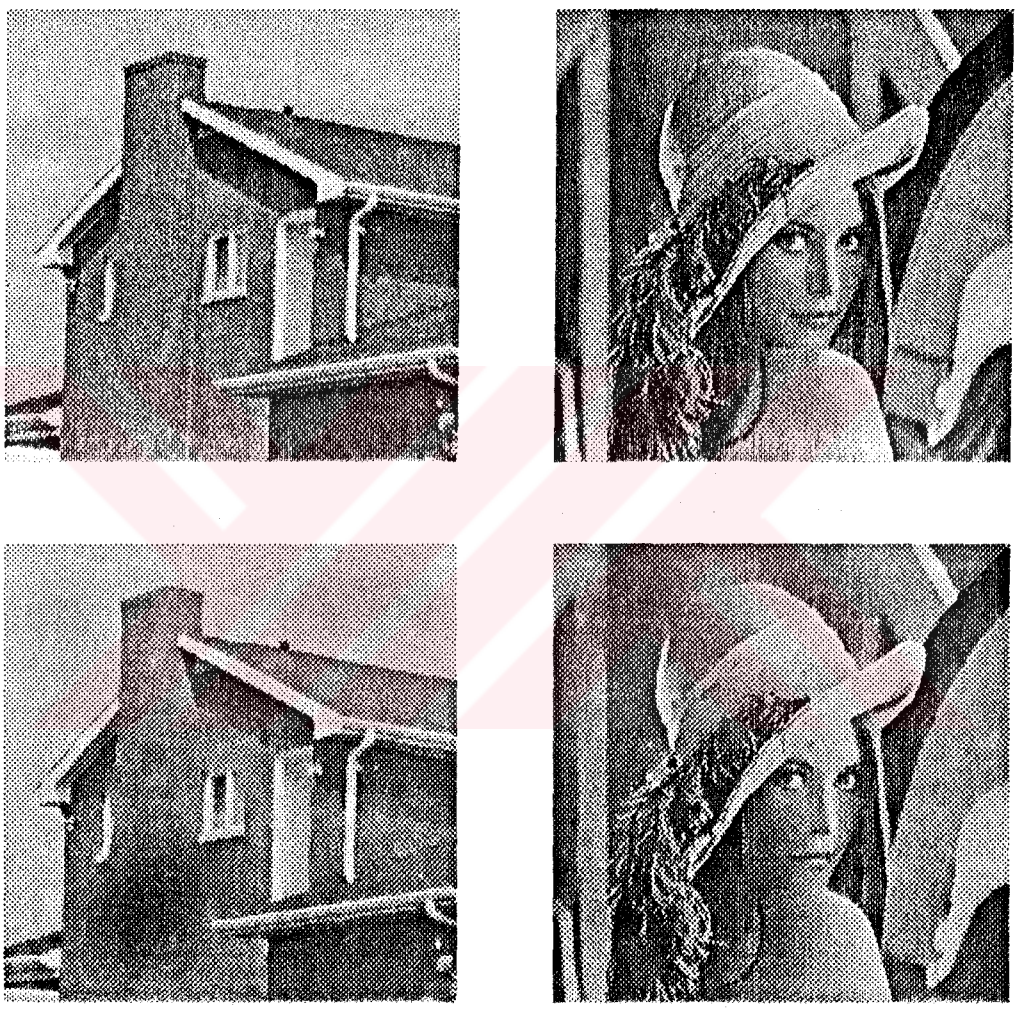


Figure 4.7. First row: Original 256x256 House and Lenna images. Second row: Reconstructed versions of House and Lenna images.

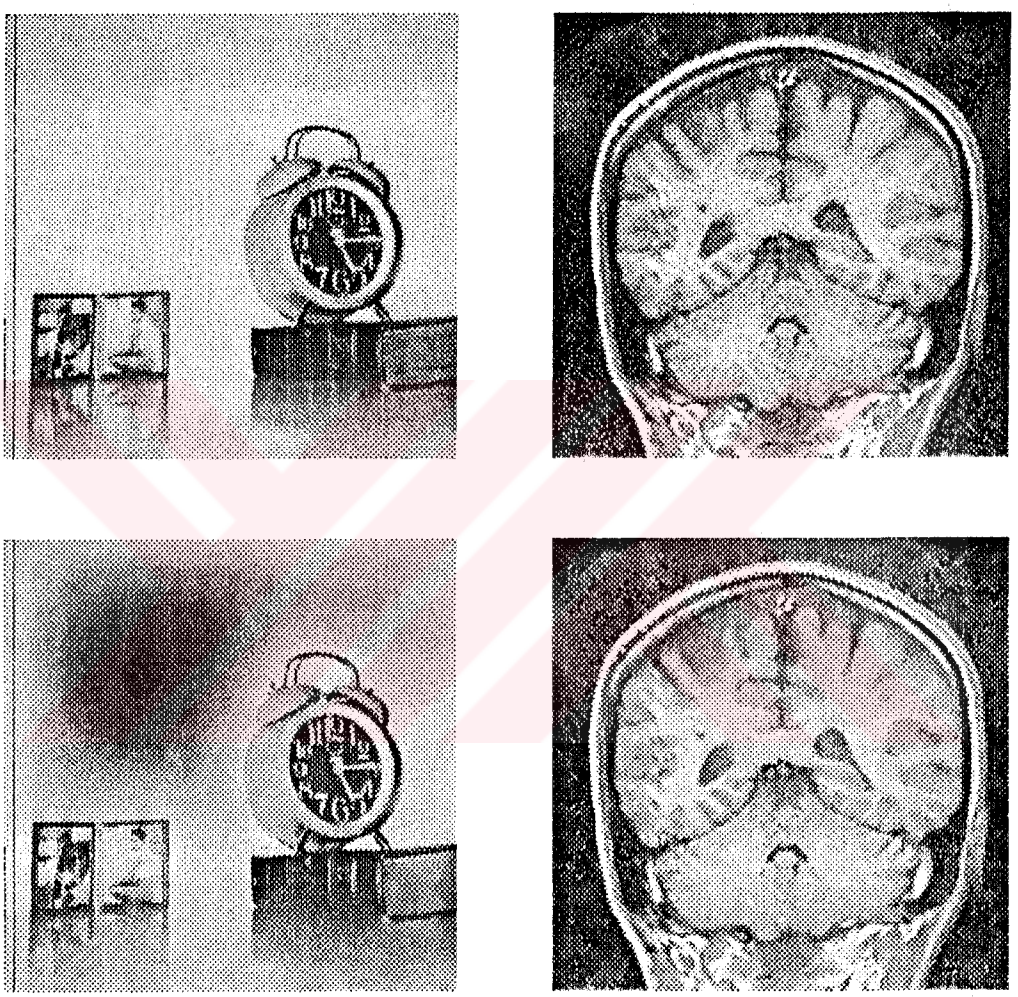


Figure 4.8. First row: Original 256x256 Clock and 175x175 Brain images. Second row: Reconstructed versions of Clock and Brain images.

zero, on either side of an edge. By having a narrower histogram, entropy coding performs better. This is not performed at the current work.

CHAPTER 5.

MULTISCALE IMAGE REPRESENTATION AND EDGE INTEGRATION BY WEIGHTED ACCUMULATION

5.1 Introduction

In this chapter, a multiscale edge representation using regularization is investigated. Each level of this representation, called the Difference of Regularized Solutions (DORS) representation, is obtained by taking the difference between two regularized solutions with different regularization parameter values [6, 7]. Each regularized solution is obtained by minimizing an energy functional associated with membrane functional. It is shown that, in 1-Dimensional case, the DORS representation can be also attained by convolving the image data with a kernel named the Vietnamese hat operator. After analyzing the multiscale behaviour of edges in this representation, a multiscale edge integration scheme is developed. By means of a weighted accumulation process, the algorithm is capable of overcoming problems of shift of edges, appearance and disappearance of edges, and branching in the scale space. Thus it can be applied to the other multiscale edge representations as well. Performance of the integration scheme on different edge representations is evaluated by using synthetic and real images.

Edge detection is a crucial step toward ultimate goal of computer vision, since the performance of the higher level processes such as object recognition relies heavily on the accuracy of the detected edges. In general an edge detector is

expected to have good detection and good localization performance when applied to a noisy image. Major difficulties in edge detection are the noise corrupting data during the imaging and sampling processes, the tradeoff between detection and localization performance and the multisize features to be extracted from image. These are studied in chapter 2in detail.

One approach to overcome these problems is to use locally adjusted schemes [2, 8] and another is to extract edges at various scales and then to integrate the multiscale results into a single edge map, the multiscale edges are combined into a single edge image by solving the matching problem of edges at different scales. However several difficulties are encountered in integration process. First of all, edges may shift as the scale changes and the amount of shift is increased at coarse scales. In addition, some of the edges may disappear at coarse scales and some edges may be split to different branches as the scale parameter increases [19].

Several integration schemes have been proposed to overcome these problems such as edge focusing [19], feature synthesis [10] and edge linking [7]. Among these integration schemes, the edge focusing algorithm appears to be the most complete scheme in terms of utilizing the behaviour of edges in multiple scales. However the fact that the major edges may disappear at coarse scales is disregarded in this scheme.

In this study, a multiscale edge representation called the DORS representation based on regularization is investigated at first. Each level of the DORS representation is obtained by taking the difference between two regularized solutions obtained with different regularization parameter values.

Continuity in the scale space to ease the integration is achieved by using all possible pairs of the regularized solutions. Then a multiscale edge integration scheme using weighted accumulation is expanded from its one dimensional version introduced by Gökmen to two dimensional case [6]. This coarse-to-fine integration scheme attempts to overcome all difficulties encountered in multiscale edge integration, that is, shift, new appearance, disappearance and branching of edges in scale space. The integration scheme was applied to edges from the DORS representation and the multiscale edges obtained by Canny detector.

5.2 Difference of Regularized Solutions

Regularization is a general framework to solve ill posed problems by imposing constraints on the solution. A multiscale image representation can be obtained by imposing the smoothness constraint on the solution and then continuously sweeping the regularization parameter which controls the amount of smoothing [6, 7].

5.2.1 The Membrane Functional

The smoothness constraint can be imposed on the solution by finding the function f that minimizes

$$E_m(f,\lambda) = \int\int_\Omega (f-d)^2\, dxdy + \lambda \int\int_\Omega (f_x^2 + f_y^2)\, dxdy \tag{5.1}$$

or

$$E_p(f,\lambda) = \int\int_\Omega (f-d)^2\, dxdy + \lambda \int\int_\Omega (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2)\, dxdy \tag{5.2}$$

where f_x and f_y are the first order derivatives of f with respect to x and y respectively and f_{xx} and f_{yy} are the second order derivatives similarly. $E_m(f,\lambda)$ and $E_p(f,\lambda)$ are the energy functionals associated with membrane (string in 1-D) and thin plate (rod in 1-D) respectively. In these functionals, the first term on the right side is a measure of the closeness of the solution f to the input data d and the second term, stabilizer, is a measure of the smoothness. The smoothness is imposed by the first order derivatives in membrane and by the second order derivatives in plate functional. The compromise between these two terms is controlled by the regularization parameter λ. In this study, the multiscale representation is obtained using membrane functional $E_m(f,\lambda)$ in Eq. 5.1.

The scale parameter in this representation is the regularization parameter λ so that the coarse levels of representation are obtained by minimizing the string

functional with a large λ, while fine levels are obtained by using small λ. Sweeping the parameter λ continuously generates the scale space representation. Surfaces are reconstructed for each λ value and the reconstructed surfaces are identified by their respective scale parameter values.

5.2.2 Multiscale Edge Detection using DORS

In the following discussion, the sequence of different scales are obtained by using λ_i values from a Λ set consisting of ordered λ_i values. Let $v(x, y; \lambda_i)$ be the regularized solution with the regularization parameter λ_i, that is,

$$\begin{aligned} v(x, y, \lambda_i) \;=\; & \{v(x, y) : E_m(f, \lambda) = \\ & \inf_{f \in V} \int\int_{\Omega} (f - d)^2 \, dxdy + \lambda_i \int\int_{\Omega} (f_x^2 + f_y^2) \, dxdy \} \end{aligned} \tag{5.3}$$

where λ_i is a member of some Λ set defined later in this section, and d is the input data.

Now we define $DORS(x, y; \lambda_i, \lambda_j)$ as the difference of two regularized solutions with regularization parameters λ_i and λ_j, that is,

$$DORS(x, y; \lambda_i, \lambda_j) = v(x, y; \lambda_i) - v(x, y; \lambda_j), \qquad \lambda_i \neq \lambda_j. \tag{5.4}$$

For 1-Dimensional case, it was shown that [2, 7]

$$v(x; \lambda_i) = d * h_r(x; \lambda_i) \tag{5.5}$$

where

$$h_r(x; \lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}, \tag{5.6}$$

which means that minimizing the string functional is equivalent to convolving the data with a filter whose impulse response is $h_r(x; \lambda)$. In this case, $DORS(x, y; \lambda_i, \lambda_j)$ can be written as

$$DORS(x;\lambda_i,\lambda_j) = d * H(x;\lambda_i,\lambda_j) \tag{5.7}$$

where

$$H(x;\lambda_i,\lambda_j) = h_r(x;\lambda_i) - h_r(x;\lambda_j) \tag{5.8}$$

is called the Vietnamese hat operator [6]. Fig. 5.1 shows the filters $h_r(x;1.5)$ and $h_r(x;3)$, and Fig. 5.2 shows the Vietnamese hat operator $H(x;1.5,3)$. The Vietnamese hat operator has one positive lobe at the center and two negative lobes at sides like the Mexican hat operator, so that the zero crossings of the signal convolved with filter can be marked as edge points. The properties of $h_r(x;\lambda_i)$ and $H(x;\lambda_i,\lambda_j)$ filters as well as relationship with other edge operators are considered in [7].

Let $h_r(x;\lambda_i)$ and $h_r(x;\lambda_j)$ denote two regularized solutions obtained by using Eq. 5.6. The DORS representation may then be rewritten as

$$\begin{aligned} DORS(x;\lambda_i,\lambda_j) &= h_r(x;\lambda_i) - h_r(x;\lambda_j)) \\ &= \frac{1}{2\lambda_i} e^{|x|/\lambda_i} - \frac{1}{2\lambda_j} e^{|x|/\lambda_j}. \end{aligned} \tag{5.9}$$

In order to detect zero crossings, the DORS image is equated to zero, that is,

$$\frac{\lambda_j}{\lambda_i} = e^{-|x|/\lambda_j} - e^{-|x|/\lambda_i} \tag{5.10}$$

and by rearranging

$$w_{zc} = \frac{\lambda_i\lambda_j}{\lambda_j - \lambda_i} \cdot \ln(\frac{\lambda_j}{\lambda_i}). \tag{5.11}$$

where w_{zc} is used instead of $|x|$. This means, after convolving the image with Vietnamese Hat operator given in Eq. 5.8, the pixels with value w_{zc} calculated in Eq. 5.11 are assigned as edge points. It can easily be seen that w_{zc} is always a positive number. The two conditions are

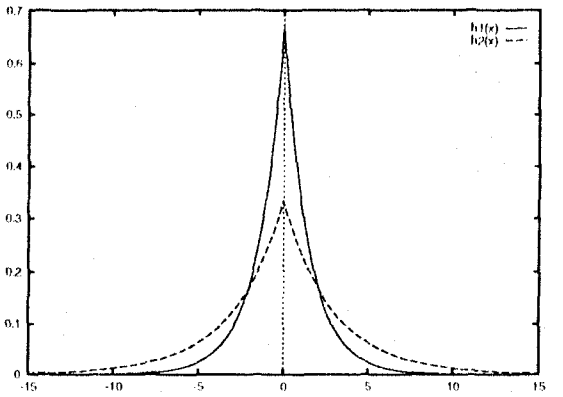


Figure 5.1. R-filters $h1(x) = h_r(x; 1.5)$ and $h2(x) = h_r(x; 3)$.

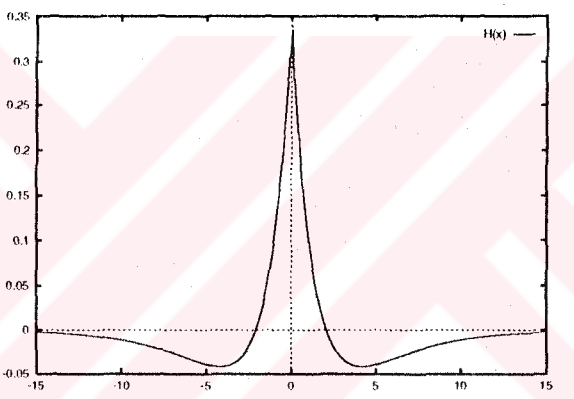


Figure 5.2. The Vietnamese hat operator $H(x; 1.5, 3)$.

1. $\lambda_j > \lambda_i$ leads to $\ln \frac{\lambda_j}{\lambda_i} > 0$, $\lambda_j - \lambda_i > 0$. This gives $w_{zc} > 0$.
2. $\lambda_j < \lambda_i$ leads to $\ln \frac{\lambda_j}{\lambda_i} < 0$, $\lambda_j - \lambda_i < 0$. This gives $w_{zc} > 0$.

Now, given a set of regularization parameters

$$\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}, \qquad \forall_i,\ \lambda_i < \lambda_{i+1}, \tag{5.12}$$

we construct the multiscale image representation V as

$$V = \{v(x,y;\lambda_1), v(x,y;\lambda_2), ..., v(x,y;\lambda_n)\}, \qquad \lambda_i, \lambda_j \in \Lambda. \tag{5.13}$$

and the multiscale edge representation W is formed as

$$W = \{w_{1,2}, w_{1,3}, ..., w_{1,n}, w_{2,3}, w_{2,4}, ..., w_{2,n}, ..., w_{n-1,n}\}, \qquad \lambda_i, \lambda_j \in \Lambda. \tag{5.14}$$

where

$$w_{i,j} = zc\,[\,DORS(x,y;\lambda_i,\lambda_j)\,] \tag{5.15}$$

and

$$zc[DORS(x,y;\lambda_i,\lambda_j)] = \begin{cases} 1 & \text{if } DORS(x,y;\lambda_i,\lambda_j) \\ & \text{has a zero crossing at } (x,y) \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

As seen from Eq. 5.14 the set W contains not only the pairs $w_{i,i+1}$, but also all other pairs provided that $i < j$. Thus $m = n(n-1)/2$ levels are constructed from n regularized solutions. The use of all pairs $w_{i,j}$ with $i < j$ creates intermediate scales and this continuity in scale space makes tracking from course to fine scale easier during integration. Fig. 5.3 shows regularized solutions and the multiscale edges obtained from them.

5.3 Multiscale Edge Integration Using Weighted Accumulation

In the multiscale algorithms, a common approach is to consider a few number of images at different scales and try to match the edge segments at these levels. While a small number of levels is desirable to reduce the computation, a continuous transform of features from level to level is also crucial to solve the correspondence problem in the matching process, which requires a large number

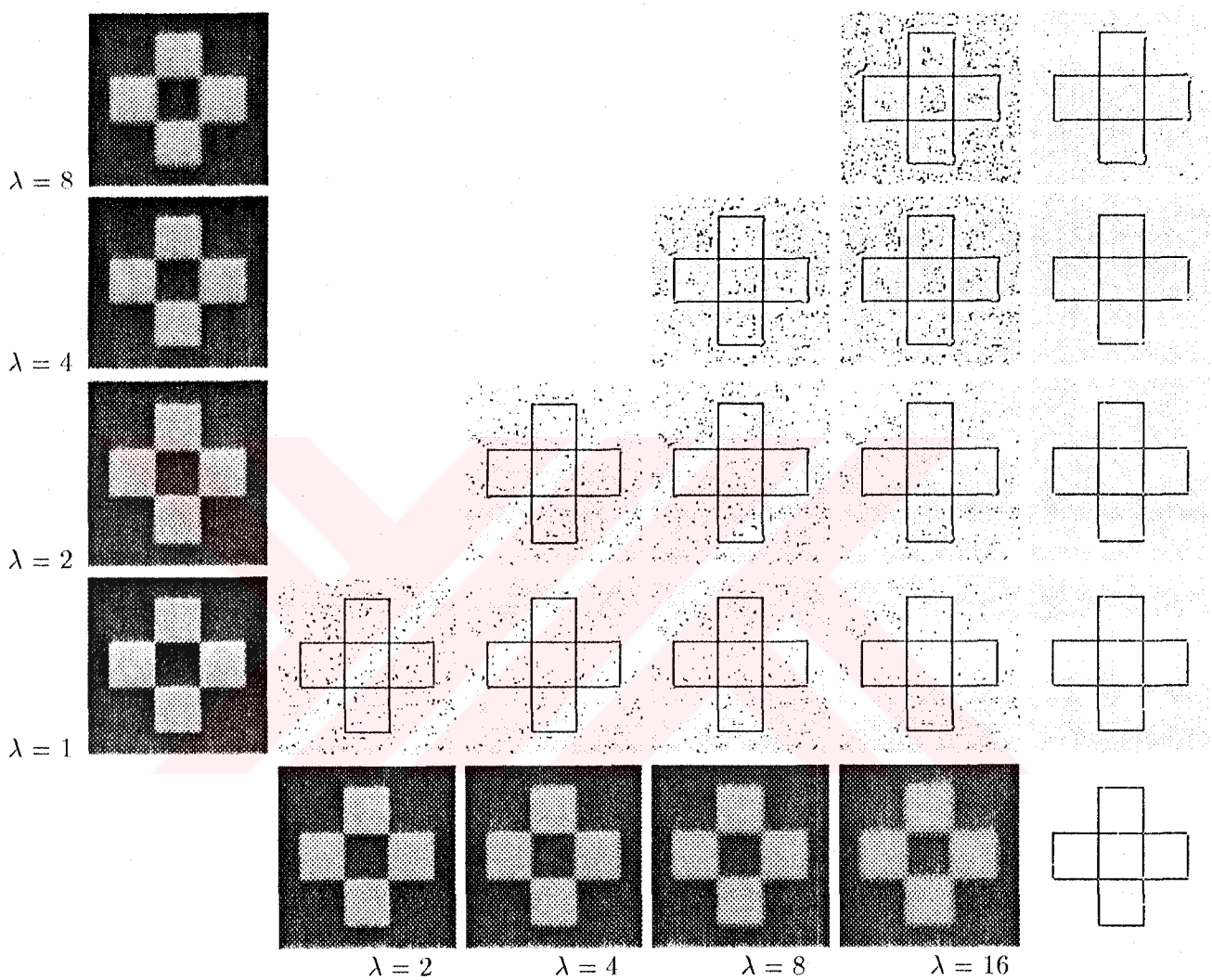


Figure 5.3. Checkerboard images (SNR=7dB) used in DORS and integration. The first column and the last row show regularized solutions with the indicated parameters. The noisy edge images obtained by using two images with DORS in the corresponding row and column are presented in the lower part of the diagonal. In the right-most column, results of intermediate integration for DORS and in the lower-right corne, the final integrated edge image are presented.

of levels in scale space representation. In the presented integration scheme, it's aimed to achieve this continuity in scale space without proportionally increasing the required computation. This is attained by considering all pairs that can be constructed from available regularized solutions.

The integration algorithm uses a set of edge images as input. If the number of these images is denoted by m, then the image set can be represented by W array as

$$W = \{w_1, w_2, ..., w_m\} \tag{5.17}$$

where the scale increases with increasing indices. In this case, w_1 corresponds to the finest scale and w_m corresponds to the coarsest scale.

The edge image set in Eq. 5.17 is obtained from DORS images by an intermediate integration. The edge images for W in Eq. 5.17 are obtained by keeping the edge points which exist in at least two images with the same first parameter in Eq. 5.15. To obtain w_m from DORS images, the intermediated integration of DORS edge images $w_{m,k}$ where $k \in [1, N]$ are performed. The input images to this intermediate level integration and the integrated edges are illustrated in Fig. 5.3.

5.3.1 Behaviour of Edges in Scale Space

In integration scheme, scale space is divided into three categories: Coarse scales, mid scales and fine scales. Based on the behaviour of edges in scale space, we know that coarse scales favour the detection performance of edge detection, i.e., most of the edges are true edges but their locations may be wrong. The fine scales, however, favour the localization performance, i.e., the edge locations are correct but there may be many spurious edges due to noise. In the integration scheme, the edges detected at the coarse scales are tracked down to the finest scale in order to determine the correct position of the actual edges. In order to handle disappearance, appearance of edges and branching in the scale space, a scheme based on the weighted accumulation of edges is developed.

5.3.2 Weighted Accumulation and Edge Tracking in Scale Space

In a coarse-to-fine tracking, first the accumulation array is set to the coarsest scale edge (w_m) image values by

$$acc(i,j) = w_m(i,j) \tag{5.18}$$

where i and j correspond to row and column number, respectively. The coarsest scale image w_m is taken from the multiscale image set given in Eq. 5.17. Then for each edge point in the current level we try to find corresponding edges at the finer level by searching its small neighbourhood such as 3×3. Two kinds of search operation are performed to find one or more corresponding edge points. The first one is called forward search and the other one is called backward search which is performed just after the first one.

5.3.2.1 Forward Search

This is the primary search stage and edge points in the finer level are determined which will be used in the backward search. For each edge point in the coarse level, a search in a 3×3 window is performed to find edge points in the finer level. The detected points are sorted by their distances to the location of the coarse level point. These points are processed in the linking operation starting from the nearest one to prevent false connections.

This forward search operation is illustrated in Fig. 5.4. In this figure, *Level* n is the coarser level and *Level* $n-1$ is the finer level where n denotes the scale number as smaller n corresponds to finer scale. Black points denote the edge points in both levels. Shaded points denote the non-edge points in the 3×3 search window. Search operation is performed between each of two such consequent levels in the multiscale image set. Edge points in *Level* $n-1$ is scanned for correspondence to point $w_n(i,j)$ at *Level* n in the search window. The size of the search window size is taken as 3×3 in this figure and in the

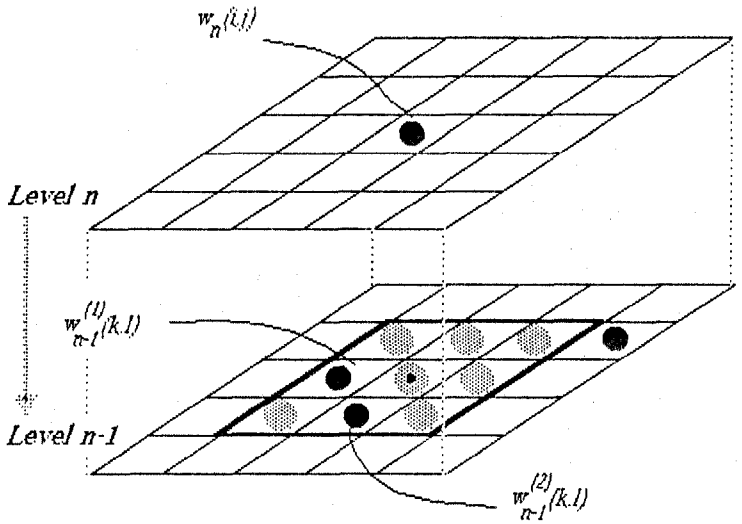


Figure 5.4. The forward search operation to find edge points in the finer level.

implemented integration algorithm. Two edge points are found in *Level* $n-1$, and they are labeled $w^{(1)}_{n-1}(k,l)$ and $w^{(2)}_{n-1}(k,l)$ in this figure. The other edge point in *Level* $n-1$ is not in the active search window which is drawn by thick lines, therefore not processed. $w^{(1)}_{n-1}(k,l)$ may also be denoted by $w_{n-1}(i,j-1)$ and $w^{(2)}_{n-1}(k,l)$ may also be denoted by $w_{n-1}(i+1,j)$. If we assume that a coarser level edge point may be connected upto three points in the finer level, then the point $w_n(i,j)$ will be connected to both of these points.

An edge point may be linked to at most three possible edge points in the finer level in order to handle branching that may occur as the scale parameter decreases. Fig. 5.4 illustrates a branching operation where there are two branches from point $w_n(i,j)$ to $w^{(1)}_{n-1}(k,l)$ and $w^{(2)}_{n-1}(k,l)$.

5.3.2.2 Backward Search

After the candidate edge points at the finer level are found, a backward search operation is performed to connect the finer level points to the nearest coarse level point. A separate backward search operation is performed for each candidate point that has been found at the forward search operation. The backward search is realized to prevent more than one coarse level point to be linked to the same fine level point thus leaving some of the fine level points unconnected.

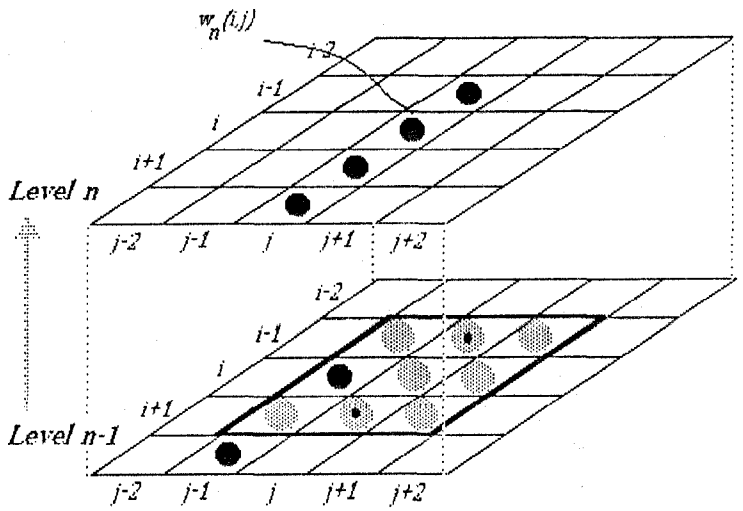


Figure 5.5. The backward search operation for the candidate points detected at the forward search.

The backward search operation is illustrated in Fig. 5.5. In the figure, black points denote edge points in both levels and shaded places are non-edge points. Row numbers are denoted by i and column numbers are denoted by j. There are only two edge points in the fine level and their locations do not coincide with any of the points in the coarse level. In the absence of backward search, point $w_n(i-1,j)$ and $w_n(i,j)$ will be connected to $w_{n-1}(i,j-1)$. The point $w_n(i+1,j)$ and $w_n(i+2,j)$ will be connected to $w_{n-1}(i+2,j-1)$. This causes the line in coarse level to be broken into disconnected points even though this line is continuous in *Level* $n-2$. To prevent this, backward search is used. It's satisfied in backward search that each fine level point may be connected at most one coarse level point. The connected fine level points are marked so that, another search will not link any other coarse level point to this one.

Fig 5.6 represents the difference between edge linking with and without the existence of edge linking. The edge points are shown by X and non-edge points are represented by 0. The arrows denote the linking direction and are directed toward the connected point in the finer level. The coarse level edge points are linked to a finer level edge point if one is found in the search window in the case on the left side. This permits more than one coarse level point to be connected to the same location in the finer level. This condition is illustrated on the left side of Fig. 5.6. In this situation, edges are disconnected after the linking. In the

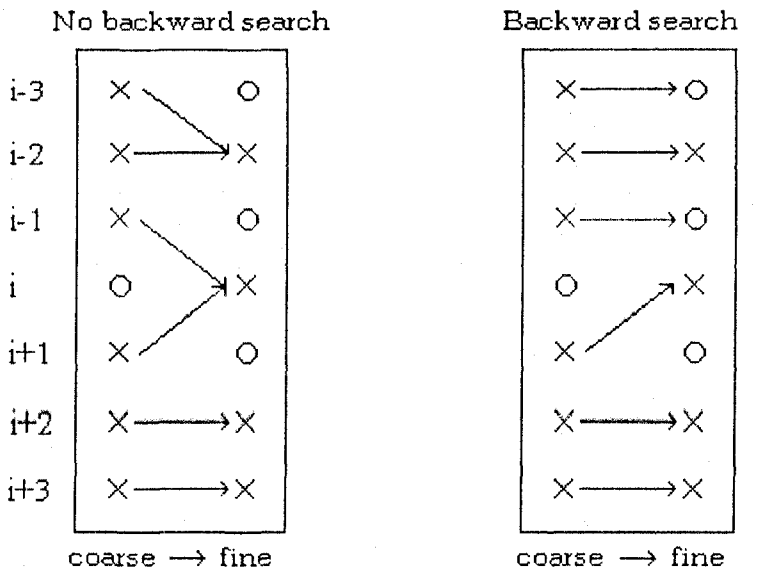


Figure 5.6. The advantage of backward search in edge linking.

existence of backward search, this permission is not granted and thus edges are obtained in forms of continuous curves. This situation is illustrated on the right side of the figure. The propogated locations are $i-3$ and $i-1$. All other linkages are true connections.

The coarse level edge points to which no correspondent fine level point is found, are propogated to the same location. This kind of propogation is illustrated in Fig. 5.5 at points $(i-1,j)$ and $(i+1,j)$. In this figure, there are two connections and two propogations. One of the connections is from $w_n(i+2,j)$ to $w_{n-1}(i+2,j-1)$ and $w_n(i,j)$ to $w_{n-1}(i,j-1)$. The propogations are from $w_n(i-1,j)$ to $w_{n-1}(i-1,j)$ and $w_n(i+1,j)$ to $w_{n-1}(i+1,j)$.

5.3.2.3 Weighted Accumulation

Then, the weight of the edge at the current level is passed to the finer level by updating the value of the accumulation array at the point where the coarse level edge is linked, that is,

$$acc(k,l) = \alpha_n(i,j)w_n(k,l) + acc(i,j) \qquad |k-i| \leq 3, \quad |l-j| \leq 3, \qquad (5.19)$$

which indicates that $w_n(i,j)$ is linked to $w_{n-1}(k,l)$. This accumulation process is done for each fine level connected point. In Fig. 5.4, updating of the accumulation array is done for $w^{(1)}$ and $w(2)$. The weight $\alpha(i,j)$ is determined by the appearance or disappearance of the edge point in scale space.

The updating of the accumulation array is different in coarse, mid and fine scales. At the coarse and mid scales, we allow the appearance of the new edge. If the point (k,l) is the first point that an edge is linked, that $acc(k,l) = 0$, then the weight $\alpha(i,j)$ is set to the distance between the current level and the coarsest level, so that edges that appear at any level of the coarse and mid scales are promoted by this scheme. In coarse and mid scales, an edge is propogated to the same location even if no connection is found in the finer level for that point.

The promotion of the new appearance of edges is terminated at fine scales since most of the new edges at these scales are due to noise. Furthermore at the fine scales an edge at the coarse level is linked to at most one edge at the finer level. In order to promote the edges that disappear during this coarse-to-fine tracking, which is identified when we cannot find any corresponding edges at the finer scale for the edge under consideration, the weight $\alpha(i,j)$ is set to the distance between the current scale and the finest scale.

After the accumulation array is updated at the current level, we move to the finer level and repeat the same process until the finest level is reached. Thus in this integration scheme where edge images are obtained by DORS with n regularization parameters, the accumulation array will contain $n(n-1)/2$ at the correct location of edge even if it does not appear at all levels. The combined edges are detected by picking the points where the accumulation array has a value of $n(n-1)/2$ for a representation obtained from n regularized solutions.

5.4 Results and Discussions

In order to explore the properties of the DORS representation, considered synthetic and real images with different SNR values are considered.

Fig. 5.3 shows the multiscale DORS representation of a noisy checkerboard image with SNR=7 dB. The noise is synthetically added by using Gaussian noise generator [6]. The images in the first column and the last row of this figure show the regularized solutions with specified regularization parameter values. Each image in the second, third, fourth and fifth columns demonstrates the edge image $w_{i,j}$ obtained from the difference of two regularized solutions shown in its associated row and column. The edge images in each row of this representation are integrated by keeping the edge points which exist in at least two edge images in this row. These intermediate edges shown in the last column of this figure are integrated using the weighted accumulation scheme and the final integrated edge image is shown in the lower right corner.

In order to explore the localization and detection performances of the method, this multiscale integration algorithm was applied to various synthetic and real images. The checkerboard and bar images with various SNR values were used in the synthetic data set. Two of these synthetic images with SNR=7 dB are shown in the first and the second columns of Fig. 5.7. Real images, House and Lenna, are shown in the last two columns. The first row shows the original noisy synthetic images and original real images. The last row exhibits the integrated edge images obtained by the weighted accumulation scheme. The intermediate levels show the edges from DORS representation.

DORS edge images are obtained by using

$$\Lambda = \{1, 2, 4, 8, 16\} \tag{5.20}$$

Λ set where $\lambda_1 = 1$ and $\lambda_5 = 16$. λ parameters shown on the left side of each row is the smaller one of two λ values used by DORS. For example, for the second row the $\lambda_4 = 8$ and $\lambda_5 = 16$ where only λ_4 is given.

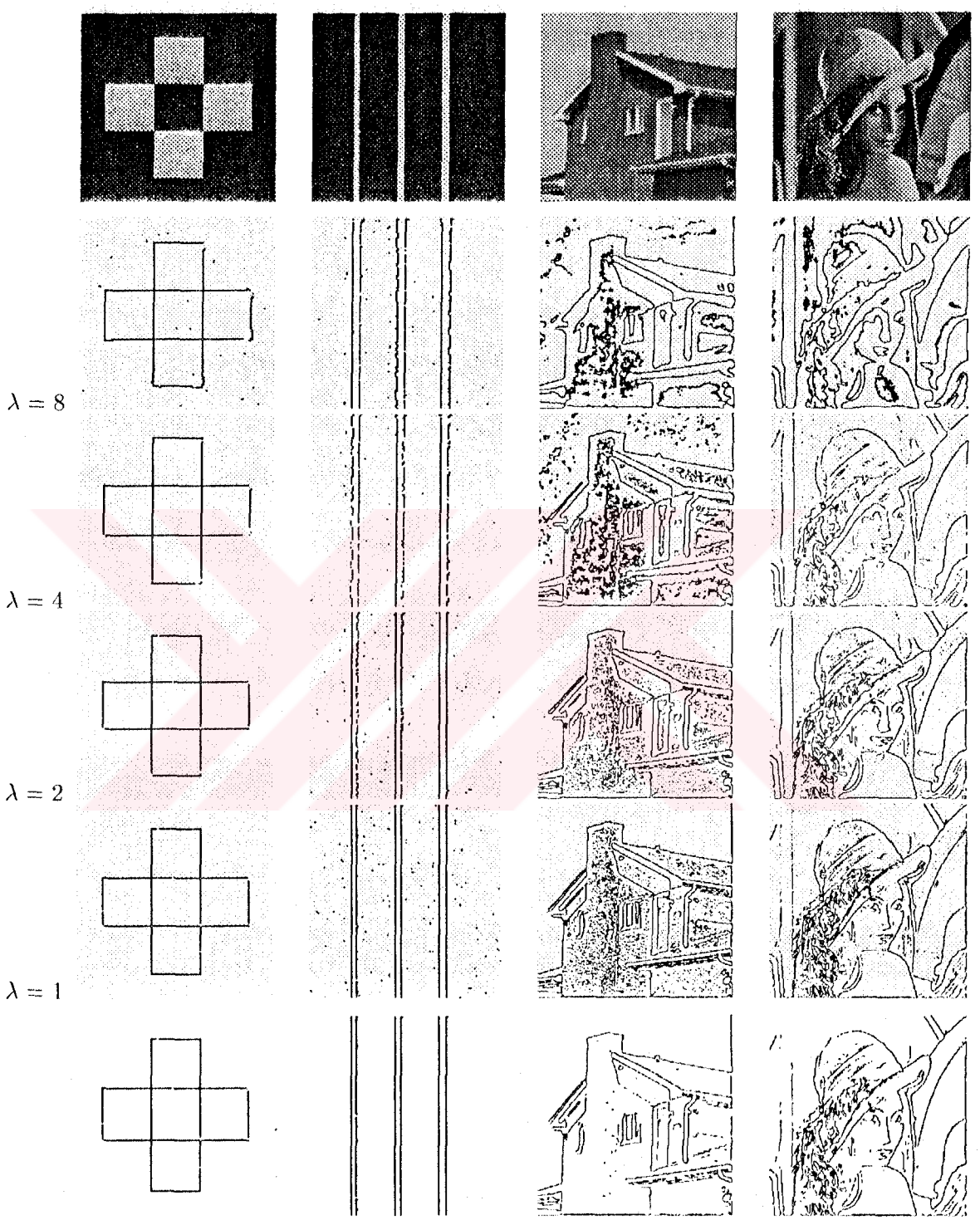


Figure 5.7. Multiscale edge images of checkerboard (128x128, SNR=7dB), bars (128x128, SNR=7dB), house (256x256), lenna (256x256) obtained by DORS, and integrated edges (last row).

As seen from Fig. 5.7, the integration algorithm can combine the results successfully by eliminating the noise and completing the missing parts in some scales. The figure exhibits that the integration algorithm can recover sharp corners of checks and actual thickness of bars. In Lenna and House images, the integration scheme can successfully locate the contours which are heavily dislocated at coarse scales. The algorithm can track down these dislocated edges to actual locations and complete the missing parts.

Fig. 5.8 shows the similar results when the integration scheme applied to the edges obtained with Canny edge detector [10]. The first row shows the original images. The σ parameter of Canny edge detector is given on the left side of each row. The images of the same row is obtained by this parameter. The edge images on last row show the integrated edge images for their corresponding columns.

The integration algorithm can pull down the dislocated edges to the actual locations and results in a complete edge map by properly combining the edges in different scales. These figures exhibit that the integration scheme correctly localizes edges using multiscale information and eliminates noisy edge occurrences. The integration scheme tends to separate the noisy segments into smaller parts while keeping almost all of actual edges, so that noisy and isolated edges can be easily eliminated. It can also combine edge segments presented only at intermediate scales, for which the edge focusing scheme [19] fails.

In conclusion, a multiscale image representation based on the difference of two regularized solutions, and an integration scheme which can handle the shift, appearance, disappearance and branching of edges in the scale space are presented.

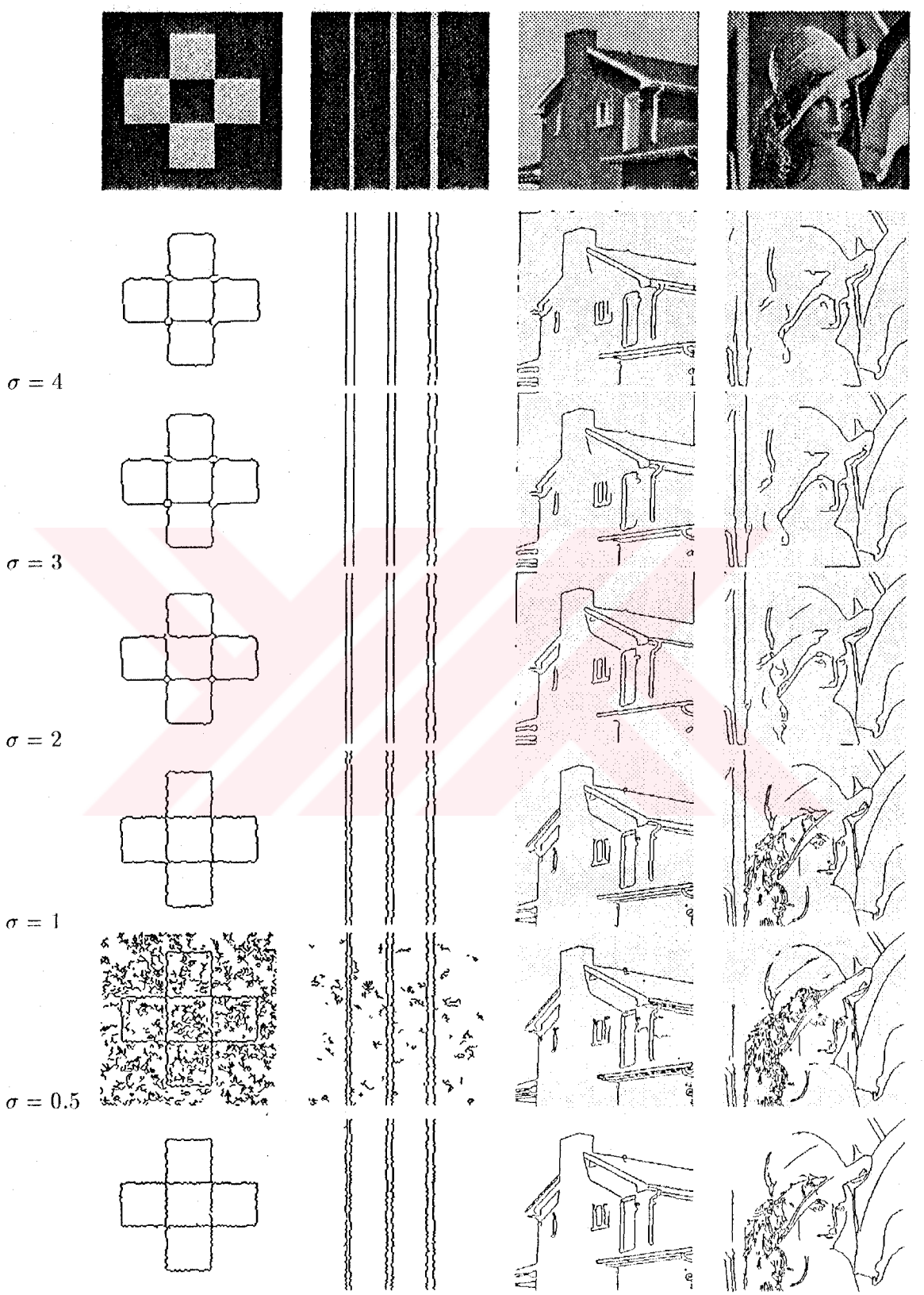


Figure 5.8. Multiscale edges of checkerboard (SNR=7dB), bars (SNR=7dB), House and Lenna images obtained by applying Canny edge detector. Integrated edge images are shown on the last row.

CHAPTER 6.

CONCLUSIONS

In this chapter, the results and contributions of this thesis have been summarized and possible problems for future research have been suggested.

6.1 Summary Of Results

The non-standard regularization has been used to solve image processing problems in this thesis. The investigated problems are edge detection and surface reconstruction, image restoration, image coding and compression, a multiscale image representation and edge integration.

The overview of edge detection algorithms show that the main problems in edge detection are the tradeoff between localization and detection criteria, noise corruption and the multisize nature of the features in the image. It is identified that good detection, good localization, robustness, efficiency and applicability to sparse data are the requirements of good edge detector. The comparison of the studied algorithms are realized by quantitative measures in addition to the qualitative ones to present an objective comparison measure of the edge detectors. The numerical measures are Figure of Merit, P(IE/AE), P(AE/IE), MAD and MSD. Edge maps are obtained from the synthetic and real images which contain Gaussian noise. By inspecting the tables denoting the quantitative results together with the obtained edge images, it's determined that the edge images obtained by weak membrane model are much more satisfactory than the ones

obtained by the adaptive smoothing.

A regularized approach to image restoration which uses discontinuities are presented. The discontinuities are enhanced to remove blurring effect of weak membrane modeling while smoothing is performed in the continuous regions separated by line processes. It's observed that the ringing effect in the image is eliminated significantly by preventing the interaction between separated regions by line processes.

Chapter 4 presents a regularization based coding/decoding scheme which uses membrane modeling. It is assumed that the regions between discontinuities are smooth and continuous, and can be reconstructed by a surface reconstruction algorithm from sparse data. The boundary detection and surface reconstruction are realized by using the same process. The discontinuities are detected by weak membrane model using line processes in the coding part. The data which causes discontinuities are used to reconstruct the smooth regions using membrane modeling. Thus a coder/decoder algorithm is developed where the same model of regularization is used in both parts.

In the last chapter, a new multiscale representation is investigated where the multiscale edges are obtained by using two regularized solutions using membrane modeling. The DORS (Difference Of Regularized Solutions) representation is used by taking the difference between two regularized solutions with different regularization parameters. Then by inspecting the behaviour of edges in scale space, a multiscale edge integration algorithm is expanded from its one dimensional case to two dimensional case. This integration scheme scans the multiscale edge images starting from the coarsest one by using two consequent edge images to update an accumulation array. The experimental results show that the integration method can successfully track down the edges in scale space and locates them in correct positions by eliminating the noise effect.

6.2 Summary Of Contributions

Two existing edge detection algorithms were compared and a number of research contributions were presented in this thesis. These studies can be summarized as follows:

- It is observed that, edge detection performance of the weak membrane modeling which is a non-standard regularization based approach, is superior than that of the adaptive smoothing.
- A non-standard regularization based image restoration algorithm is developed and it is seen that the method can successfully removes blurring and obtain the restored image.
- A new image coding/decoding scheme is introduced where sparse information along discontinuities is used in the surface reconstruction algorithm using membrane model.
- The DORS representation is used in two dimensions to obtain multiscale edge images from two regularized solutions.
- A multiscale edge integration algorithm has been extended for two dimensional images which uses a weighted accumulation array to integrate multiscale edge images.

6.3 Suggestions for Future Research

The results are obtained by using 3×3 averaging filter in chapter 3. The introduced method can handle different blurring filters to obtain the restored images. The application of this scheme to other blurring filters should be considered.

The coding algorithm can be improved to achieve higher compression ratio. The first stage that should be considered is to implement differential Huffman

coding in the coding of data along discontinuities. Since data along an edge contour does not represent abrupt changes, the difference of adjacent values will not yield large values. The differentiation process will result in a narrower histogram and the entropy will have smaller value.

BIBLIOGRAPHY

[1] SAINT-MARC, P., CHEN, J., MEDIONI, G., Adaptive smoothing: A general tool for early vision, Trans. on Pattern Anal. Mac. Int., Vol.13, No.6, pp.514–529, June 1991.

[2] BLAKE, A., ZISSERMAN, A., Visual Reconstruction, MIT Press, Cambridge, MA, 1987.

[3] BLAKE, A., Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction, Trans. on Pattern Anal. Mac. Int., Vol.11, No.1, pp.2–12, Jan 1989.

[4] GÖKMEN, M., LI, C.C., Edge detection using refined regularization, Comp.Vis. and Patt. Recog., Lahaina, Maui, Hawaii, 3-6 June 1991.

[5] ACAR, T., GÖKMEN, M., Yırtılabilir zar modeliyle resim onarımı, Elektrik Müh. 5. Ulusal Kong., pp.247–252, Trabzon, Turkey, Sep, 13-18 1993.

[6] GÖKMEN, M., Edge Detection Using Refined Regularization, Ph.D. thesis, University of Pittssburgh, 1990.

[7] GÖKMEN, M., LI, C.C., Multiscale edge detection using first order R-filter, Int.Conf. on Patt. Recog., The Hague, The Netherlands, 1992.

[8] GÖKMEN, M., LI, C.C., Edge detection and surface reconstruction using refined regularization, Trans. on Pattern Anal. Mac. Int., Vol.15, No.5, pp.492–499, 1993.

[9] HILDRETH, E., MARR, D., Theory of edge detection, Roy. Soc. Press, pp.187–217, London B, 207, 1980.

[10] CANNY, J.F., A computational approach to edge detection, Trans. on Pattern Anal. Mac. Int., Vol.8, pp.679–698, 1986.

[11] HARALICK, R.M., The digital step edge from zero-crossings of second directional derivatives, Trans. on Pattern Anal. Mac. Int., Vol.6, No.1, pp.58–68, 1984.

[12] GEMAN, D., GEMAN, S., Stochastic relaxation, gibbs distributions and the bayesian restoration of images, Trans. on Pattern Anal. Mac. Int., Vol.6, No.6, pp.721–741, 1984.

[13] TERZOPOULOS, D., The computation of visual surface representations, Trans. on Pattern Anal. Mac. Int., Vol.10, No.4, pp.1988, 1988.

[14] ACAR, T., GÖKMEN, M., Uyarlanır düzleme ve yırtılabilir zar modelinin ayrıt saptama açısından karşılaştırılması, 1. Sinyal İşleme ve Uygulamaları Kurultayı, pp.111–116, Bosphorus University, İstanbul, Turkey, Apr, 21-22 1993.

[15] LIM, J.S., Two Dimensional Signal and Image Processing, Prentice Hall Signal Processing Series, PTR Prentice-Hall, Inc. Englewood Cliffs, NJ 07632, 1990.

[16] BOEKEE, D.E., LAGENDIJK, R.L., BIEMOND, J., Regularized iterative image restoration with ringing reduction, IEEE Trans. on Acoust., Speech, and Signal Proc., Vol.36, No.12, pp.1874–1888, Dec. 1988.

[17] ACAR, T., GÖKMEN, M., Ağırlıklı toplama yöntemiyle Çok Ölçekli ayrıt birleştirme, 2. Sinyal İşleme ve Uygulamaları Kurultayı, pp.65–70, Gökova, Muğla, Turkey, Apr, 8-9 1994.

[18] ACAR, T., GÖKMEN, M., Image coding by using weak membrane model of images, to appear in SPIE's symposium on Visual Commuciations and Image Processing '94, Chicago, IL, Sep, 25-29 1994.

[19] BERGHOLM, F., Edge focusing, Trans. on Pattern Anal. Mac. Int., Vol.9, No.6, pp.726–741, 1987.

APPENDIX A.

USER MANUAL OF DEVELOPED PROGRAMS

In each chapter, the method or methods are turned into application programs all written in the C programming language. All of the programs are composed of several modules and should be linked externally after compilation, or one can use the *make* utility to obtain an executable module from the source codes. The source code of the programs may be compiled in any system having an ANSI compatible C compiler. Each program has a *Makefile* to simplify the compiling/linking process. All of the programs work on HIPS formatted images, but may easily be converted to work on any other format. HIPS is a binary format file having a short ASCII text header specifying information such as the size of the image, bits per pixel, number of frames, creator of the image, and optional remark lines. The current versions of the programs work on gray scale HIPS formatted images with any size.

Each program is compiled and executed on SUN workstations running SunOS 4.1.3, Hewlett-Packard 9000 series 400 and 700 running HP/UX 9.02, NCR Tower 32/600 running UNIX Rel. 4, Data General - aviion series 300 running DG/UX and IBM compatible PCs running some version of MS-DOS. The developed programs are expected to run on any system running UNIX System V or derivatives. [1]

C compilers used in UNIX or UNIX-like operating systems are the default C compilers shipped with the operating system, i.e. *cc*. On SUN workstations, C

[1] All brands and names are the property of their respective owners.

and AT&T Rel. 2.1 C++ translator/compilers are used even though the program source codes are not written in C++. Under the MS-DOS operating system, Watcom C compiler v8.0 and Pharlab 386 DOS-Extender are used, thus the system must have at least 386sx or an higher processor. Not all of the programs are experienced with 16-bit C compilers such as Microsoft or Borland series of C/C++ compilers.

The machine named as *media* is a SUN Sparcstation IPX having Internet address of 160.75.26.1. The machine named *cadmain* is an HP 9000 series 400 having Internet address 160.75.26.2. Both of the workstations reside in the computer science department of the university.

A.1 Adaptive Smoothing

Directory	: `/home/acar/edge-detection/adaptive_smooth@media`
Backup	: `/users/acar/backup/adaptive.tar.Z@cadmain`
Program name	: `adaptive`
Usage	: `adaptive` *InputImage* `[` k `[`τ`]` `[`*iterations*`]]`

Default parameter values may be seen by running the program without any parameters. *InputImage* must be an HIPS formatted image. The output image files will have the following names:

Table A.1. Filenames in adaptive smoothing.

Filename	UNIX and derivatives	MS-DOS
Output Image	*InputImage*.adp	*InputImage*.adp
Edge Image	*InputImage*.edge_adp	*InputImage*.edg

The k is the scale of adaptive smoothing (see the related chapter for details). The τ is the threshold value for edge existence: In detecting edges from the smoothed image, a point will become an edge candidate if any one of the first directional derivatives is greater than the τ value. Smoothing of the *InputImage* with the adaptive filter is performed *iteration* times.

An example may be as,

```
adaptive house.hips 0.4 32 20
```

where *InputImage* is `house.hips`, scale k is `0.4`, τ is `32` and the number of smoothing iterations is `20`. Then two images with the same size as the *InputImage* named `house.adp` (`house.adp`) for smoothed image, and `house.edge_adp` (`house.edg`) for edge image will be created. The heading of the created images will contain the parameters used to obtain these images in text form. Thus, one may see the parameters by executing a command like "`head house.adp`" to see these parameters.

A.2 Weak Membrane Modelling with Graduated Non-Convexity

Directory	: `/home/acar/edge-detection/gnc@media`
Backup	: `/users/acar/backup/gnc.tar.Z@cadmain`
Program name	: `gnc`
Usage	: `gnc` *InputImage* `[` λ `[`α`]` `[`*SOR-w*`]]`

Default parameter values may be seen by running the program without any parameters or with the "`-h`" option. This program outputs two image files: One is the reconstructed surface and the other one is the edge map. The naming convention of these output image files is specified in the following table:

Table A.2. Filenames in weak membrane with GNC.

Filename	UNIX and derivatives	MS-DOS
Output Image	*InputImage*.gnc_out	*InputImage*.gno
Edge Image	*InputImage*.edge_gnc	*InputImage*.edg

λ and α values are the coefficients of the energy functional of weak membrane model. *SOR-w* is the relaxation parameter of the SOR (*Successive Over Relax-*

ation) minimization method. This parameter value should be in $[0, 2]$ interval to satisfy convergence.

An example of running this program may be as the following line,

```
gnc lenna.hips 0.9 800
```

where *InputImage* is `lenna.hips`, λ is `0.9`, α is `800` and the *SOR-w* is calculated by the program. After program completion two images with the same size as the *InputImage* will be created: `house.gnc_out` (`lenna.gno`) for reconstructed image, and `lenna.edge_gnc` (`lenna.edg`) for the edge image. The parameters used to create these images are written in the header of each output image in the same way as in the adaptive smoothing.

A.3 Image Restoration Using Regularization

Directory	: `/home/acar/deblurring@media`
Backup	: `/users/acar/backup/deblur-gnc.tar.Z@cadmain`
Program name	: `deblur`
Usage	: `deblur` *InpImg* `[`λ `[`α`][`*SOR-w*`] [average|gauss [`σ`]]]`

The *InpImg* is expected to be blurred by either an averaging or a Gaussian filter with standard deviation of σ. The deblurring program must already know what kind of blurring is realized on the *InpImg*, hence either `average` or `gauss` should be given. If none is given, then the input image is supposed to be blurred by an averaging filter with size 3x3. If `gauss` specified, then the input image is expected to be blurred by a Gaussian filter. The standard deviation of this Gaussian filter may be given optionally in the σ parameter. If no σ parameter is present in the command line, then this value is taken to be 1. λ and α are two parameters of weak membrane modeling. Typical values are, $\lambda = 1.6$ and $\alpha = 1600$. *SOR-w* is the relaxation parameter of the SOR minimization method. Recommended value range of *SOR-w* is $[0, 2]$.

After program completion, one image is created named *InpImg*`.deblur_out` (*In-*

pImg.deb in MS-DOS systems). This is the deblurred image obtained by applying the algorithm onto the *InpImg* with the specified parameters. The parameters used to obtain this image are written into the header part of the output image.

Sample usage may be as,

```
deblur house14.hips 1.6 1600 1.2 gauss 1.4
```

where blurred *InpImg* is `house14.hips`, λ is 1.6, α is 1600 and *SOR-w* is 1.2. The last two parameters denote that the input image is blurred by a Gaussian filter with $\sigma = 1.4$. The output image will be named `house14.deblur_out` (`house14.deb` in MS-DOS systems).

Another sample usage may be,

```
deblur lennaa.hips
```

where the blurred *InpImg* is `lennaa.hips`. The rest of the parameters are taken to have their default values: $\lambda = 1.6$, $\alpha = 1600$, *SOR-w* is calculated, and blurring is performed by a 3x3 averaging filter. The output image will be named `lennaa.deblur_out` (`lennaa.deb` in MS-DOS systems).

A.4 Image Coding and Compression Using Weak Membrane Model of Images

Directory	: `/home/acar/compression/gnc-compress/gnccompress@media`
Backup	: `/users/acar/backup/gnc-compress.tar.Z@cadmain`
Program name	: `gnccompress`
Usage	: `gnccompress -(sr) <InputFile> [λ [α] [SOR-w]]`

The default values are $\lambda = 1$ and $\alpha = 1000$. *SOR-w* is calculated from the given parameters.

The program includes the coder and the decoder parts together. The action is selected by giving any one of the command line parameters of `s` or `r`, but not

both. The `s` corresponds to sender (coder) and the `r` parameter corresponds to receiver (decoder) part. To simplify the sender/receiver program invoking with appropriate parameter lists separate batch files are written.

`send` This batch file needs one parameter which is the name of the input HIPS file with its full file name. This is the coding part. The `gnccompress` program is invoked with $\lambda = 0.8$, $\alpha = 250$ and $w = 1.2$. If the *FileName* is `filename.hips`, then compressed file is created with name `filename.lproc`. Compressed line processes and sparse data reside in this file. This file has a special format. In addition to this file, an HIPS formatted file named `inputfile.betadata` is created (if enabled in the source code) which is the sparse data residing along the discontinuities. This file may be viewed on the screen as an ordinary HIPS file.

`recv` This batch file needs one parameter which is the name of the compressed file. This is the decoding part. The `gnccompress` program is invoked with $\lambda = 6$ and $SOR-w = 1.2$. α parameter is not used in the decoding part. If the *InputFile* is named `inputfile.lproc` typically, then the decoded image will be named `inputfile.compr`.

The program displays appropriate processing information such as the bit count used in run-length coding, the number of data and its percentage in the whole image, the size of the coded line processes, entropy of the sparse data.

An example usage is,

```
gnccompress -s house.hips 0.8 250 1.2
```

or

```
send house.hips
```

for the sender part. After the program completion, `house.lproc` file is created. This is a special format file and cannot be viewed as expected. Then the decoding part is run by

```
gnccompress -r house.lproc 6 1 1.2
```

or

```
recv house.lproc
```

where number 1 for α after 6 signifying λ is given just to obey formal description and then to give $SOR - w$ parameter. After program termination `house.compr` file is created which may be viewed on the screen as an HIPS file. In the remark line of this HIPS file, appropriate parameters are written for later accesses.

A.5 Membrane Modelling

Directory	: `/home/acar/edge-detection/dors/membrane@media`
Backup	: `/users/acar/backup/membrane.tar.Z@cadmain`
Program name	: `membrane`
Usage	: `membrane [-h] [λ [SOR-w]]` < *InputFile* > *OutputFile*

The default values are: $\lambda = 1.6$ and $w = 1.3$. "`-h`" option is used to display a short usage message. The file naming convention is explained below.

> Let *Filename* denote the name of the file without extension, e.g. `.hips`. The constructed image by `membrane` program with λ_i will have the form *Filename*`.ext.`λ_i where,
>
> `.ext.` is the extension (usually `.hips`), thus forming the inputfile-name as *Filename.ext* for *InputFile* parameter of `membrane` program.
>
> $.\lambda_i$ is a number representing the λ value used in reconstruction. Hence the *OutputFile* section of `membrane` program is *Filename.ext.*λ_i.

An example usage is,

```
membrane 1 1.2 < house.hips > house.hips.1
```

The λ is 1, *SOR-w* is 1.2. The parameters used to reconstruct the image are written onto the header of the output image, here `house.hips.1`. The λ value used in reconstruction may also be extracted from the remark line of the header in the output image in the form "`Lambda=`λ". So that, the remark line is searched for the string "`Lambda=`", then the next characters denote the numerical value of λ until a non-numeric character.

A.6 Multiscale Image Representation – DORS

Directory	: `/home/acar/edge-detection/dors/dors@media`
Backup	: `/users/acar/backup/dors.tar.Z@cadmain`
Program name	: `dors`
Usage	: `dors` *InputImage1* `[edge%]` < *InputImage2* > *EdgeImage*

Both *InputImage1* and *InputImage2* must be given. The output is *EdgeImage* which is obtained by detecting the zero crossings of DORS (*InputImage1*, *InputImage2*). The threshold value used in zero crossing detection is calculated in the program. This threshold is calculated so that *edge%* of the difference image (DORS (*InputImage1*, *InputImage2*)) will be edge points. Default value for optional parameter *edge%* is 10. The calculated threshold value is also written onto the header of the *EdgeImage* together with the other parameters.

If `realimage` macro is defined by `-Drealimage` during the compilation phase, the output edge image *EdgeImage* will have black (0) in the background and white (255) in the foreground, i.e. edge locations. If this macro is not defined, then the edge image will have 0 in the background and 1 in the edge locations.

The file naming convention used throughout the study is explained below.

> *InputImage1* and *InputImage2* are typically the two image files of the `membrane` program, thus having the form *Filename.ext.*λ_i. Assume that the names of *InputImage1* and *InputImage2* be `image1.ext.`λ_1 and `image2.ext.`λ_2, respectively. Then the output filename of *DORS* is expected to be `filename.ext.dors.`$\lambda_1.\lambda_2$ where `filename`, `image1`

and `image2` are usually equal to each other, which means that they are processed forms of the same image (e.g. `house`).

A sample usage will be,

```
dors house.hips.1 15 < house.hips.2 > house.hips.dors.1.2
```

where *InputImage1* is `house.hips.1`, *InputImage2* is `house.hips.2`, *edge%* is 15, and the output edge image will be called `house.hips.dors.1.2`. Then λ_1 which is 1, and λ_2 which is 2, may be found in the remark line in the header of the output image in a similar format as of the `membrane` program, having "`Lambda1=`" and "`Lambda2=`" being the specifiers of λ_1 and λ_2 respectively.

A.7 Multiscale Edge Integration Using Weighted Accumulation

Directory	: `/home/acar/edge-detection/dors/integration@media`
Backup	: `/users/acar/backup/integrate.tar.Z@cadmain`
Program name	: `integrate`
Usage	: `integrate [-fh[l` *LogFile*`]] EdgeImage [IntEdgeFile] [> IntEdgeFile]`

The options and their usage is explained in Table A.3.

The file naming convention is explained below.

The integration program accepts a filename from which all filenames that will be used in the integration process is extracted. The series of files may be the output of `dors` program or some other edge detection program. The file names are treated differently in each condition:

- If the edge images are created by `dors`, then they have names like *filename.ext.dors*.λ_1.λ_2. Then only *filename.ext.dors* part must

Table A.3. Command line option for integration program.

Option	Explanation
`-f`	Use weighted accumulation directly on the *EdgeImage*, i.e. don't perform intermediate AND/OR integration process.
`-h`	The help screen like this explanations.
`-l` *LogFile*	writes information messages onto *LogFile* instead of `stderr`.
EdgeImage	is the name of a series of edge files without extensions. These extensions will be appended two integers separated by dors, whose values differ from 0 to 32.

be given to the integration program. The integration program uses all files whose names match *filename.ext.dors.*∗.∗ where ∗ will have numeric values between 0 and 32.

- If the edge images are created by some other edge detection program such as `canny`, then the "`-f`" option must have been given on the command line. If the name of the command line file is *filename.ext*, then the integration program will use all files whose names match *filename.*∗ where ∗ will have numeric values ranging from 0 to 32.

An example may be as the following:

```
integrate house.hips.dors > house.edge.hips
```

The multiscale edge images should be created by `dors` and have names beginning with `house.hips.dors`. Files with names `house.hips.dors.`λ_1`.`λ_2 are searched where λ_1 and λ_2 ranges from 0 to 32. Files with the same λ_1 are processed with intermediate AND/OR integration to give just one edge file with the name `house.hips.dors.`λ_1. Then the outputs of each intermediate integration are

integrated to give the final integrated edge image **house.edge.hips** as indicated on the command line.

APPENDIX B.

VOCABULARY

accumulation ..... toplama
adaptive ..... uyarlanır
averaging ..... ortalama
blurring ..... bulanıklaştırma
convexity ..... içbükeylik
convolution ..... katlama
crease ..... kırılma
criteria ..... ölçüt
detection ..... saptama
deterministic ..... gerekimci
edge ..... ayrıt
efficiency ..... etkinlik
filter ..... filtre, süzgeç
graduated ..... aşamalı
image processing ..... görüntü işleme
immunity ..... bağışıklık
impulse ..... dürtü
integration ..... birleştirme
intensity ..... yoğunluk
interaction ..... etkileşim
isolated ..... yalıtılmış
iterative ..... ardışıl
line process ..... çizgi işlevi

multi-resolution ..... çok ölçekli
nonconvexity ..... dışbükeylik
performance ..... başarım
piecewise ..... parça parça
pixel ..... benek
reconstruction ..... kurma
recursive ..... özyineleme
region ..... bölge
regularization ..... düzgünleştirme
relaxation ..... gevşeme
representation ..... gösterilim
robust ..... gürbüz
run-length ..... seyirtim
scale ..... ölçek
segment ..... bölüt
signal ..... i aret
simulated annealing ..... tavlama benzetimi
simulation ..... benzetişim
smoothing ..... düzleme
step ..... basamak
surface ..... yüzey
threshold ..... eşik
tracking ..... izleme
object ..... nesne
occluding ..... örten
vision ..... görme
weighted ..... ağırlıklı

BIOGRAPHY

Tolga Acar was born on September 28^{th}, 1967 in Balıkesir. He has graduated from Naval Lycee of Turkish Naval Forces on 1985. He has been a cadet in Naval Academy from Aug. 1985 to Jan. 1988. He entered Control and Computer Engineering of Istanbul Technical University on 1988 and graduated as the first in the class on Jul. 1992. His undergraduate thesis is the design and implementation of a multitasking operating system for 386 based computer systems. He has been a research assistant at Computer Science Department, Electrical and Electronics Faculty of Istanbul Technical University on Nov. 1992. He has worked for Center of Defence Studies in Foundation of Istanbul Technical University from Jan. 1993 to Jan. 1994 on target recognition from radar images and as UNIX system and network administrator as a part-time researcher. He has also worked as a lecturer in the data structures course given to Alcatel-Teletaş telecommunications company in spring 1994. His current research areas are image processing, computer vision and operating systems. He is a student member of IEEE.