İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TARANMIŞ RESİMLERİN VEKTÖR FORMATINA DÖNÜŞTÜRÜLMESİ

DOKTORA TEZİ
Y. Müh. Barbaros SOYER
(511920009012)

Tezin Enstitüye Verildiği Tarih : 22 Temmuz 1999
Tezin Savunulduğu Tarih : 02 Haziran 2000

Tez Danışmanı : Doç. Dr. Temel KOTİL
Diğer Jüri Üyeleri Prof. Dr. Eşref ADALI (İ.T.Ü.)
Prof. Dr. Ertuğrul TAÇGIN (M.Ü.)
Doç. Dr. Mehmet KORÜREK (İ.T.Ü.)
Doç. Dr. Ercan Öztemel (Tübitak)

HAZİRAN 2000

ÖNSÖZ

Bu çalışmanın ortaya çıkmasında emeği geçen herkese, özellikle Doç. Dr. Temel Kotil' e teşekkür ederim.

Haziran 1999 Barbaros Soyer

İÇİNDEKİLER

SEMBOL LİSTESİ

P_i	Üzerinde işlem yapılan benek
i,j,k	Sayaç
$\vec{a}_{ik},\vec{b}_{ik}$	i. beneğin k komşuluğunda oluşturulan vektörler
$\cos_{ik}$	i. beneğin k komşuluğundaki vektörlerinden oluşturulan k adet kosinüs değerleri (R-J yöntemi için)
h	i. beneğin komşu sayısını gösterir
$\overline{\cos_{ik}}$	i. beneğin k komşuluğundaki vektörlerinden oluşturulan k adet kosinüs değerleri (R-W yöntemi için)
c_v , c_i	Eğrilik
δ	Sayısal eğri üzerindeki P_i ve $t=0$ iken uydurulan eğri üzerindeki P_{bi} arasındaki mesafe (M-Y yöntemi için)
M	Beneklerin varsayılan teğete olan toplam mesafelerinin bir göstergesi (Bartem yöntemi için)
x^+_{ort},y^+_{ort}	İlgilenilen beneğin önündeki benek koordinatlarının aritmetik ortalaması
x^-_{ort},y^-_{ort}	İlgilenilen beneğin arkasındaki benek koordinatlarının aritmetik ortalaması
m	Sayısal eğriye uydurulan teğetin eğimi
α	Sayısal eğriye uydurulan teğetin açısı
E_i, E_x, E_y	Hata fonksiyonları
$\vec{u},\vec{v}$	İlgilenilen beneğin önündeki ve arkasındaki beneklerden en küçük kareler metoduyla oluşturulan vektörler (LSQ yöntemi için)
r^i_j	i. dış sınır eğrisinin j. parçası
m^i_j	i. dış sınır eğrisinin j. parçasının eğimi

ŞEKİL LİSTESİ

Sayfa No.

TABLO LİSTESİ

Sayfa No.

TARANMIŞ RESİMLERİN VEKTÖR FORMATINA DÖNÜŞTÜRÜLMESİ

ÖZET

Çizimdeki benek desenlerin, doğru parçalarına, yaylara, bileşik çizgilere, çemberlere ve uygun spline eğriler gibi temel geometrik nesnelere dönüştürülmesine vektörleştirme denmektedir. Bu işlem sonucunda ulaşılan resim temsili, noktasal resim temsilinden genellikle çok daha derlitopludur. Fakat elde edilen nesnelerin sınırlarında mevcut küçük düzensizlikler gibi bazı bilgiler, yok edilebilirler.

Bir mühendislik çiziminin, kâğıtta çizili olmasından ziyade elektronik ortamda bilgisayar destekli tasarım (BDT) formatında bulunmasının birçok getirisi vardır: BDT çizimlerinin üzerinde değişikliklerin yapılması daha kolaydır, kopyalarının çıkarılması ve yeniden çizilmeleri daha hızlıdır, üzerlerinden bilgi çıkarmak daha kolaydır ve genel olarak çizimlerin yönetilmesi daha kolaydır. Buna rağmen günümüzde etkin olarak kullanılan ve üretilen mühendislik çizimlerinin çoğu, hâlâ sadece kâğıt üzerindedir. Maliyet-kâr analizleri göstermiştir ki, bir çizimin bir kaç kere düzenleneceği ihtimali varsa o çizimin, kâğıt üzerinde çizilmesinden ziyade elektronik ortamda oluşturulması, oldukça avantaj sağlamaktadır.

Vektörleştirme işlemi, sadece mühendislik çizimlerinin noktasal formattan vektörel formata dönüştürülmesi için kullanılmaz. Aynı zamanda şekil tanıma, şekil sınıflandırılması, nesnelerin geometrik testleri, serbest-el çizimlerinin yorumu gibi değişik alanlarda da vektörleştirme işleminden yararlanılmaktadır.

Bu tezde ilk olarak vektörleştirme işlemindeki basamaklardan biri olan köşe tespiti üzerinde çalışılmıştır. Yedi adet köşe bulma algoritması sunulmuştur. Rosenfeld-Johnston, Rosenfeld-Weszka ve Medioni-Yasumoto köşe tespit yöntemleri de sunulmuştur. Bu köşe bulma yöntemlerinin, on sayısal kapalı eğri üzerindeki performans karşılaştırması yapılmıştır. İkinci olarak farklı, yeni bir vektörleştirme algoritması sunulmuştur.

Kapalı bir sayısal eğri olan C, n adet tamsayı koordinattan oluşan aşağıdaki gibi bir küme kabul edilebilir.

$$C=\{P_i=(x_i,y_i),i=1,\cdots,n\} \tag{1}$$

Burada P_{i+1}, n modülüne göre P_i nin komşusudur. Bir sayısal eğrideki eğriliğin yüksek olduğu noktalar, köşe noktaları olacağı için genellikle köşe tespit yöntemleri ilk olarak herbir benek için eğrilik hesabına giderler. Daha sonra ikinci adımda köşe noktalarına karar verilmeğe çalışılır.

Geliştirdiğimiz köşe bulma yöntemlerinden ilk beşi (Bartem, Moment, Spthr, Spmin, Sp3min), sayısal eğrinin ilgilenilen beneğindeki teğet doğrusunun bulunması esasına dayanır. Çünkü ardışık beneklerdeki teğetlerin eğimleri arasındaki fark, sayısal eğrinin o benekteki eğriliği kabul edilebilir. Hatmin adı verilen altıncı yöntem, bir hata fonksiyonuna göre sayısal eğriye bir parabol uydurur. Sayısal eğrinin ilgilenilen noktasındaki eğriliğinin parabolün aynı noktadaki eğriliği olduğu kabul edilir. Görüldüğü gibi geliştirilen ilk altı yöntemde ilk önce her benek için bir eğrilik değeri bulunmaya çalışılmış ve bu eğriliklerin belirtilen komşulukta maksimum olanlarına köşe noktaları denmiştir.

LSQ adı verilen geliştirilen yedinci yöntemde İlgilenilen P_i noktası için en küçük kareler yöntemi kullanılarak $\vec{u}$ ve $\vec{v}$ vektörleri oluşturur. Öndeki k adet benek, $\vec{u}$ vektörünün oluşumu için arkadaki k adet benek, $\vec{v}$ vektörünün oluşumu için kullanılır. Her benek için oluşturulan bu iki vektörün skaler çarpımından hareketle $\cos\alpha$ değeri bulunur. Her benek için bulunan $\cos\alpha$ değerleri eğriliğin bir göstergesi kabul edilebilir. Dolayısıyla bir beneğin köşe olabilmesi için kendisine ait $\cos\alpha$ değerinin lokal maksimum olması gerekecektir.

$$\cos\alpha=\frac{u1\cdot v1+u2\cdot v2}{\sqrt{u1^2+u2^2}\cdot\sqrt{v1^2+v2^2}} \tag{2}$$

Uygulamalardan hareketle görülmüştür ki LSQ köşe tespit yöntemi gerçeğe en yakın sonuçları vermektedir. Dolayısıyla LSQ yöntemi, ilerideki geliştirmeler için tercih edilmektedir.

Bu tezde köşe bulma yöntemleri üzerindeki çalışmalar tamamlandıktan sonra tesirli bir vektörleştirme işlemi sunulmuştur. Özellikle mühendislik çizimlerinin (şimdilik sadece doğruları içeren) sayısallaştırılmasıyla elde edilen noktasal formattaki resim verilerinin yorumu için bir algoritma tanımlanmıştır. Önerilen algoritma beş basamaktan oluşur; (1) sınır eğrlerin elde edilmesi, (2) sınır eğrilerdeki köşe tespiti, (3) komşulukların oluşturulması, (4) karşı karşıya komşulukların bitirilmesi ve (5) uç noktaların birleştirilmesi. Burada köşe tespiti için LSQ algoritması kullanılmıştır.

Doğrular arasındaki ilişki bulunduktan sonra karşı karşıya olan komşu doğrular bulunur. Dördüncü adımda karşı karşıya komşuluğu olan sınır doğruları, bir doğruya indirilir. Oluşan yeni doğruların uç noktaları üst üste olmayabilir. Bu yüzden son adım doğruların uçlarında oluşan boşlukların giderilmesine yöneliktir.

Sunulan vektörleştirme algoritması, inceltme adımını içermemektedir. Bu önemli bir üstünlüktür. Sonuçta inceltme kaynaklı karmaşıklık, yavaşlık ve istenmeyen yapay bozukluklar gibi problemler, safdışı edilmiş olur. Ayrıca önemli bir zaman tasarrufu da söz konusudur.

VECTORIZATION OF A DIGITIZED CURVE

SUMMARY

Vectorization converts certain pixel patterns on the drawing into basic geometric entities such as straight line segments, arcs, polylines, circles and possibly splines. The resulting representation is usually much more compact than the represantation of the raster image, but some information is discarded, such as small irregularities in the boundaries of the inferred entities

There are many advantages in having engineering drawings in electronic computer-aided design (CAD) format rather than hard copy only: CAD drawings are easier to modify, faster to copy and retrieve, easier to extract information from, and, in general, easier to manage. Nevertheles, most engineering drawings in active use today, and most of those being produced, are still in hard copy format only. Cost-benefit analyses show that, if a drawing is expected to be modified several times, it is advantageous to convert them from hard copy to electronic format.

Vectorization is not only for transformation from raster to vector form of engineering drawing, but also it is used in pattern recognition, pattern classification, geometrical testing of objects, interpretation of freehand drawings etc.

In this thesis firstly corner detection, one of the steps in vectorization process , is studied. We present seven corner detection algorithms. Corner detectors considered here include Rosenfeld-Johnston corner detector, Rosenfeld-Weszka corner detector and Medioni-Yasumoto corner detector. We compare the performance of these algorithms using ten digital closed curves. Secondly a different vectorization algorithm is presented.

Let the sequence of n integer-coordinate points describe a closed curve C,

$$C = \{P_i = (x_i, y_i), i = 1, \cdots, n\} \quad (1)$$

where P_{i+1} is a neighbor of P_i (modulo n). Since dominant points on a curve correspond to points of high curvature, the various existing dominant point detection algorithms first compute an estimate of the curvature at each point on the curve. Next, a two-stage procedure is applied to choose dominant points.

First five (Bartem, Moment, Spthr, Spmin, Sp3min) of our developed seven dominant point detection algorithms are based on constructing the tangent line at the point (pixel) being interested of the digital curve. Because the curvature at each pixel on the digital curve is calculated from difference between the slopes of the tangent line at the successive pixels. In the sixth method called Hatmin a parabola is fitted according to the error function. The curvature of the parabola at point P_i is considered as the curvature of digital curve at P_i. In these six methods the pixels having the maximum slope change (curvature) at specified neighbourhood are called corner points.

In our developed seventh method, called LSQ, for the interesting pixel P_i, vectors $\vec{u}$ and $\vec{v}$ are formed by least squares method. The front k-pixels are considered to form vector $\vec{u}$ and the back k-pixels for vector $\vec{v}$. For each pixel $\cos\alpha$, angle between these two vectors by scaler multiply, is calculated. The point P_i is a corner if $\cos\alpha_i$ is a local maximum.

$$\cos\alpha = \frac{u1 \cdot v1 + u2 \cdot v2}{\sqrt{u1^2 + u2^2} \cdot \sqrt{v1^2 + v2^2}} \quad (2)$$

From the application point of view, it is seen that the performance of the LSQ corner detector is closest to that of a human viewer. So the LSQ method is preferred for feature development.

In this thesis after the completion of the corner detection techniques, an effective vectorization process is proposed. An algorithm for the interpretation of raster data acquired by digitizing especially engineering drawings (in line format for the present) is described. The proposed algorithm includes five steps; (1) obtaining border curves, (2) detecting corners of border curves, (3) forming neighbours, (4)

finishing face to face neighbours and (5) connecting end points. Here, for corner detection LSQ corner detector is used.

After finding the relation between the lines, face-to-face neighbour border lines are caught. In the fourth step, a line is constructed from each face-to-face neighbour border lines. These new lines' end points may not be coincide. Therefore the last step tries to get rid of these gaps.

The proposed vectorization process does not include thinning. This is the most important advantage. As a result, the problems being caused by a thinning algorithm like creating unwanted artifacts, complexity and slowness is eliminated. Additionally a serious time saving is a matter of course.

1. BÖLÜM

GİRİŞ

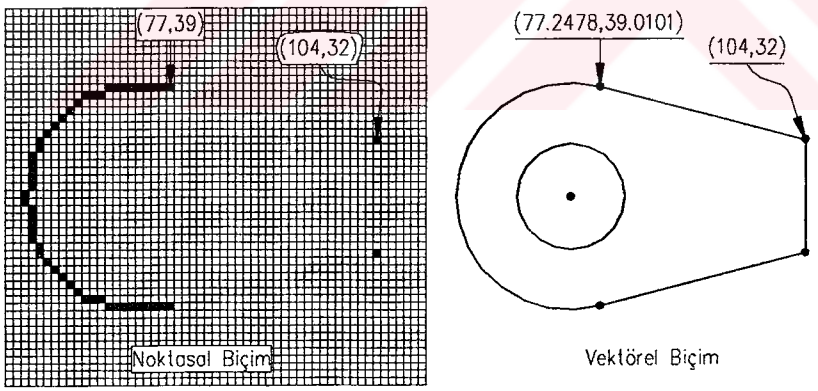
Bilgisayarın yaygın kullanımı, bir çok bilginin de artık elektronik ortamda muhafaza edilmesini beraberinde getirmiştir. Yazı, resim ve ses gibi bilgiler artık kolaylıkla elektronik ortamda uzun süreler saklanabilmekte, düzenlenebilmekte ve iletilebilmektedir. Bu kolaylıklar özellikle arşiv ve iletişim alanlarında inanılmaz boyutlarda gelişmelere sebep olmaktadır. Bu gelişmelerin insanoğlunun çok daha fazla bilgiye ulaşma ve sahip olma duygularını kamçıladığını iddia edebiliriz. Doğal olarak bilginin çokluğu, üzerinde yorum yapabilme gücünde zayıflamayı getirir. Yani insan, 200 sayfalık bir kitap üzerinde hakimiyet kurabilir ama 5 000 000 sayfalık bir kitap karşısında çaresizdir. Çaresizliğini yine kendisi giderecek ve bu bilgi yığınını değerlendirmede kendisine yardımcı olacak yazılımlar ve cihazlar icat edecek veya üretecektir.

Bu çalışmada bizim ilgilendiğimiz bilgi türünü resimler oluşturuyor. Elektronik ortamda resimler, noktasal (raster) ve vektörel (vector) biçimde tutulurlar. Bu biçimlerin birbirlerine göre bazı üstünlükleri ve eksiklikleri vardır [1]. Aşağıdaki tabloda (Tablo 1.1) bu iki biçim arasında bir karşılaştırma verilmeğe çalışılmıştır.

Noktasal biçimdeki resim veya çizim denilince, Şekil 1.1' de görüldüğü gibi bir ızgara düzlemi akla gelir. Benek (pixel) denen ızgara elemanları, temel çizim öğeleridir. Resim veya çizim, bu beneklerden oluşur. Oysa vektörel biçimdeki bir resim veya çizim söz konusu olduğunda sürekli bir ortam (düzlemde) düşünülür. Temel çizim elemanları, analitik denkleme sahip doğru, çember, yay, elips gibi geometrilerdir. Çizim bu nesnelerden oluşur. Şekil 1.1' de iki biçim arasındaki fark anlatılmaya çalışılmıştır.

Tablo 1.1 Vektörel Ve Noktasal Biçim Karşılaştırması

Vektörel Format	Noktasal Format
Parametrik olarak tanımlanmış sembollerin oluşturulması.	Serbest el çizimi ve desenleri gibi parametrik olmayan nesnelerin oluşturulması.
Daha yüksek derecede geometrik hassasiyet sağlanabilmesi.	Kullanıcı tanımlı desenler ve daha hızlı bölge dolguları oluşturulması.
Yüksek hassasiyetli otomatik ölçülendirme yapılabilmesi.	Serbest el ile düzenleme ve silme işlemlerinin rahat yapılabilmesi.
Parametrik olarak tanımlı doğru, yazı ve sembollerin düzenlenmesi.	Çizimlerin istenen bölgelerinin hızlı kesilip yapıştırılması.
Dönme (rotate) ve ölçeklendirme (scale) gibi işlemlerde minimum bilgi kaybı olması.	
Başka uygulamalara girdi için yeterince hassas geometrilerin sağlanması.	



Şekil 1.1 Noktasal ve vektörel biçim karşılaştırması.

Bu iki biçim arasında zaman zaman dönüşüm gerekmektedir. Vektör biçiminden noktasal biçime dönüşümde bir takım zorluklar [2] olmasına rağmen tersine göre çok daha kolay olduğu söylenebilir. Konunun zor olması ve henüz istenen neticelerin alınamamış olmasının yanında, uygulama sahasının da fazlalığı çalışmaların yoğun bir şekilde noktasal biçimden vektör biçime dönüşüm etrafında

toplanmasına sebep olmaktadır. Bu tezin araştırma konusu da vektörleştirme denen bu dönüşüme yöneliktir.

Beneklerden oluşan noktasal biçimdeki bir şeklin, doğru, yay, çember veya elips gibi temel geometrik nesnelere dönüştürülmesi işlemine vektörleştirme (vectorization) denir [3]. Birçok adımdan oluşan vektörleştirme işleminin uygulama sahası oldukça geniştir.

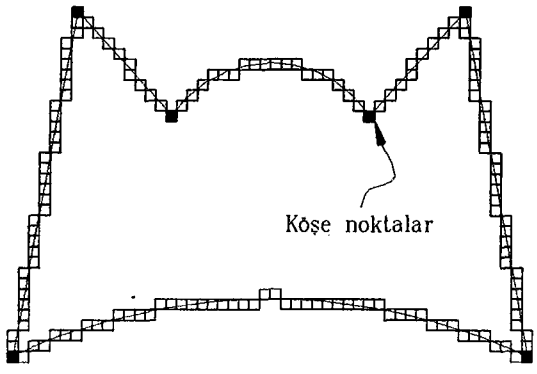
Vektörleştirme işleminin yaygın kullanıldığı alanlardan biri, çizilmiş teknik resimlerin arşivleme zahmetinden kurtulmak ve bu çizimlerden benzer yeni çizimlerin oluşturulabilmesi kolaylığından yararlanmak için bunların elektronik ortama aktarılması çalışmalarıdır. Kâğıt üzerindeki çizimleri elektronik ortama aktarmanın iki yolu vardır: (1) Bir BDT (Bilgisayar Destekli Tasarım) yazılımı kullanarak bu çizimlerin elektronik ortamda yeniden çizilmesi; (2) Bir tarayıcı vasıtasıyla çizimin elektronik ortama aktarılması. Birinci yöntem sonuçta doğrudan vektör biçiminde çizimin oluşmasına sebep olma açısından makbul ama harcanan emek ve zaman açısından oldukça tavizkârdır. Dolayısıyla birinci yol, istisnalar haricinde çoğunlukla tercih edilmez. İkinci yöntem oldukça hızlı çizimin elektronik ortama aktarılmasını sağlar. Fakat çizim, noktasal formattadır. Dolayısıyla üzerinde müdahalelerin yapılması zahmetli olduğundan, özellikle de teknik resim çizimleri bol düzenleme gerektirdiğinden, tercih edilmez. Bu noktada vektörleştirme işlemi devreye girer. Önemine binaen sedece bu konu etrafında vektörleştirme işlemi üzerinde yoğun çalışmalar sürdürülmekte olup önemli gelişmeler [3-14] elde edilmesine rağmen bir çok problem hâlâ tam olarak çözülememiştir.

Amerika Birleşik Devletleri ve Kanada' da yaklaşık olarak 3.5 trilyon teknik resim çiziminin olduğu sanılmaktadır. Her yıl yaklaşık 26 milyon yeni çizim bu sayıya eklenmektedir [3]. Bu çizimlerin hazırlanması, çoğaltılması ve arşivlenmesi için yıllık bir trilyon doları geçen harcamalar yapılmaktadır. Ülkemizde sayıların bu kadar büyük olmayacağı muhakkak. Fakat ülkemiz için bulunacak sayıların da bizim için yeterince büyük olacağı kanaatindeyiz.

Vektörleştirme işleminin uygulama alanının oldukça geniş olduğunu söylemiştik. Örneğin bir fabrikada konveyör üzerinde taşınan cisimlerin hangilerine, robot kollar tarafından hangi işlemlerin uygulanacağına karar verilmesinde vektörleştirme işleminden yararlanılabilir. Yani taşınan cismin kamera yardımıyla elektronik ortama

aktarılan görüntüsü vektörleştirme işleminden sonra daha kolay tanınabilecektir. Aynı şekilde çekilen bir hava fotoğrafında var olan cisimlerin, yolcu, kargo veya savaş uçağı olup olmadığının tesbiti için de vektörleştirme işlemi kullanılabilir. Bazı üç boyutlu cisimlerin (türbin kanatçığı gibi) geometrik testleri [15] yapılırken, serbest elle çizilen şekillerin vektörel biçimde elektronik ortama aktarılmasında [16] vektörleştirme işlemine ihtiyaç duyulabilir. Çoğu coğrafi bilgi sistemleri, üzerlerinde işlem yapacakları bilginin vektör tabanlı temsil edilmesini istedikleri için çekilen uydu fotoğraflarının işlendikten sonra elde edilen sonuçların vektörleştirme işleminden geçirilmesi gerekecektir [17].

Vektörleştirme işleminin önemi üzerinde durduktan sonra bu tezde bir diğer önemli çalışma konusu olan köşe bulma hakkında da birşeyler söylemek gerekir. Vektörleştirme işleminin temel adımlarından biri olan köşe bulma, görüntü işlemede önemli çalışma konularından biridir. Köşe bulma, vektörleştirme işlemindeki temel basamaklardan biri olmakla beraber, noktasal formattaki bir şeklin belirleyici özelliklerinden olması sebebiyle şekil tanıma, şekil sınıflandırma, bilgisayar görü ve görüntü işlemeye yönelik olarak yaygın şekilde kullanılmaktadır. Bir sayısal eğrinin tanımlanması veya sınıflandırılması çalışmalarında köşelerin dikkate alınmadığı durumlarla karşılaşılmakla beraber [18-20], yaygın olarak bu tip çalışmalarda köşeler veya köşelerle yakından ilgili eğrilik işin içine katılmaktadır [21-29]. Vektörleştirmeden bağımsız olarak da köşe bulma yöntemleri üzerinde çalışmaların sürdüğü görülmektedir. Otomatik pilot uygulamasına [30], karakter tanımaya [31], el yazısı analizine [32], parmakizi sınıflandırmasına [28] yönelik algoritmalarda köşe bulmanın bir basamak olarak zaman zaman yer alması kullanım alanının genişliğine örnek olarak gösterilebilir.



Şekil 1.2 Sayısal kapalı bir eğri ve köşeleri.

Köşeler, bir sayısal eğrinin yani noktasal biçimdeki eğrinin baskın veya belirleyici noktalarıdır. Şekil 1.2' de sayısal bir eğrideki köşeler gösterilmiştir. Bu noktaların bulunmasından sonra sayısal eğri elemanlarına bölünmüş olur ve şeklin tanınması kolaylaşır. Bugüne kadar köşe tesbiti için çeşitli yöntemler geliştirilmiştir. Kabaca bu yöntemleri iki sınıfta toplamak mümkündür [33] : (1) Doğrudan gri tonlu görüntüye uygulanan yaklaşımlar [28,33-36] ve (2) gri tonlu görüntüdeki bölgelerin sınırlarının elde edilmesinden sonra uygulanan yaklaşımlar [37-48]. Gri tonlu görüntüye uygulanan yaklaşımlar üzerindeki çalışmaların son zamanlarda artmasına rağmen sayısal sınır eğrileri üzerine yapılan çalışmalar daha yaygındır. Çünkü sayısal eğri üzerinde çalışmak daha basit ve bir çok elektronik görme sistemlerinde başarıyla kullanılmaktadır.

Bu çalışmada iki konu üzerinde yoğunlaşılmıştır; köşe tesbiti ve vektörleştirme. Yukarıda bahsedildiği gibi bu iki konunun da üzerinde çalışmalar yaygın olup kullanım alanları çok geniştir.

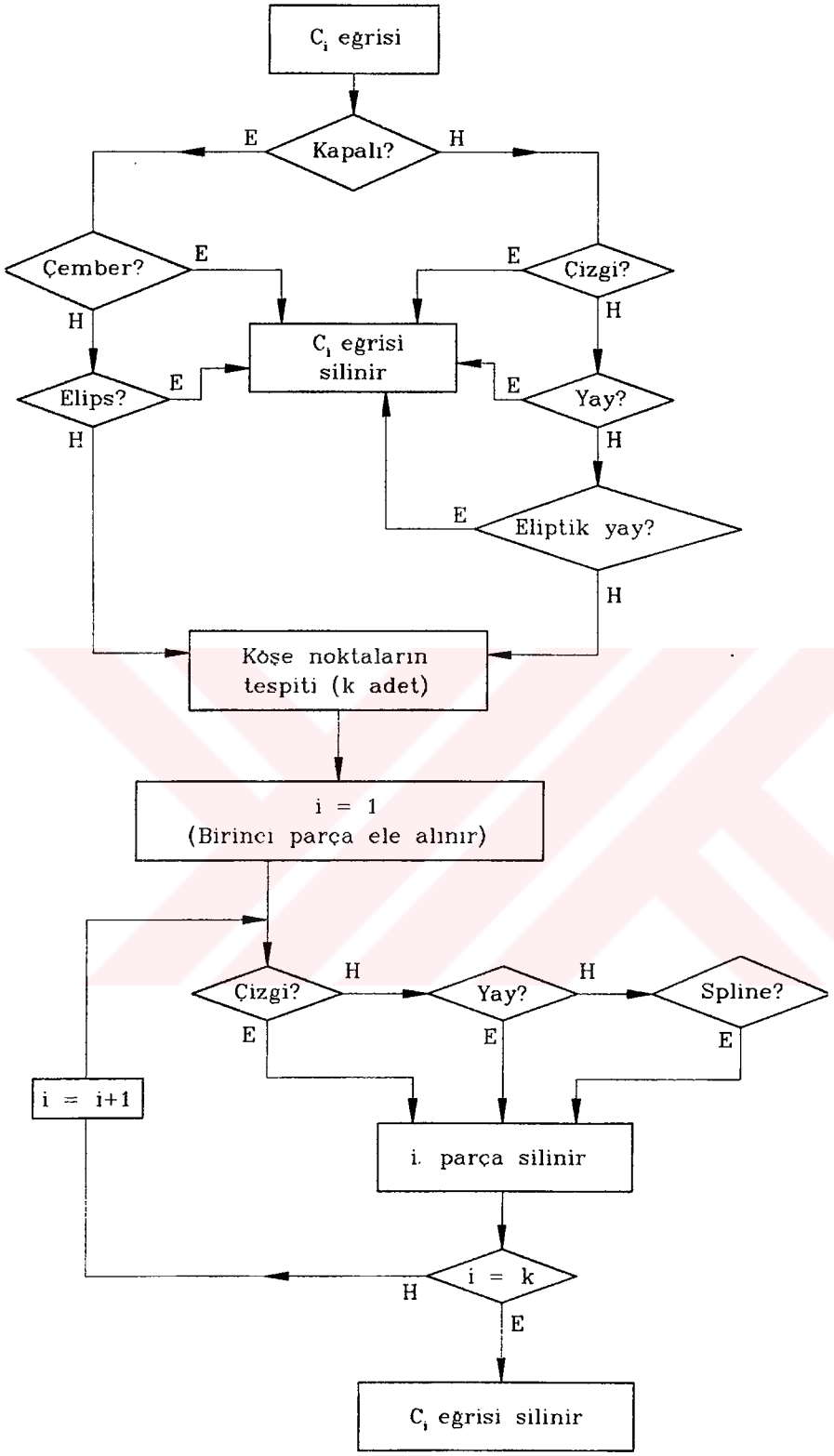
Çalışmamızın birinci bölümü, köşe bulma; ikinci bölümü, vektörleştirme konusuna yöneliktir. Giriş bölümünde vektörleştirme işlemine ve sayısal eğrilerin köşelerinin bulunmasının önemi üzerinde durulmaya çalışılmış ve diğer bölümlerle ilgili kısaca bilgi verilmiştir. İkinci bölüm, kapalı sayısal bir eğrinin köşe noktalarının tesbiti ile ilgilidir. Bahsedilen kapalı sayısal eğri, birim genişliktedir. Yani sayısal eğriye ya bir inceltme (thinning) algoritması sonucunda ya da bir nesnenin dış sınırı izlenerek ulaşılmıştır. Kaynak taramalarımızdan gördüğümüz kadarıyla köşe bulma çalışmalarında genellikle karşılaştırma amacıyla seçilen üç yöntem, burada da referans alınmıştır (Rosenfeld-Johnston [38-40,42,45,50], Rosenfeld-Weszka [38-40,42,43,45], Medioni-Yasumoto [39,41] köşe bulma teknikleri). Değişik geometrilere referans seçilen bu üç yöntem ve tarafımızdan geliştirilen yedi yöntem uygulanarak sonuçlar verilmiştir. Üçüncü bölümde henüz geliştirilme aşamasında olan üstünde çalışmalarımızın devam ettiği ve şimdilik sadece doğrulardan oluşan şekillere uygulanabilen bir vektörleştirme yöntemi sunulmuştur. Dördüncü bölüm, son bölüm olup sonuçların değerlendirilmesine ve ileriye yönelik düşüncelere ayrılmıştır.

2. BÖLÜM

KAPALI SAYISAL BİR EĞRİNİN KÖŞELERİNİN BULUNMASI

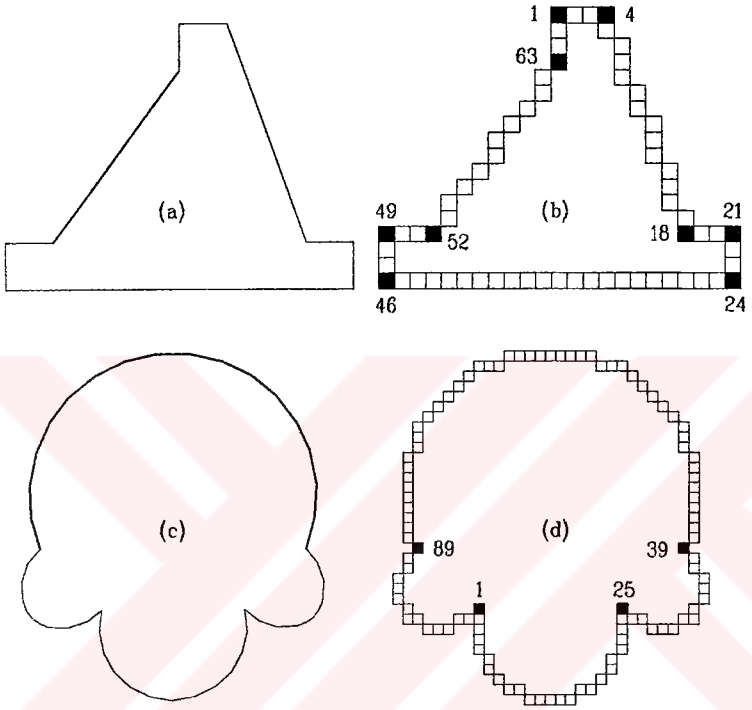
Bu bölümde, tezdeki birinci çalışma konusu olan köşe bulma üzerinde durulmaktadır. Köşeler, noktasal formattaki bir şeklin belirleyici özelliklerinden olması sebebiyle şekil tanıma, şekil sınıflandırma, bilgisayar görü ve görüntü işlemeye yönelik olarak yaygın şekilde kullanılmaktadır. Elektronik ortamın günümüzde kazandığı önem, beraberinde konuya yönelik bütün çalışmaların önem kazanmasına sebep olmuştur. Önceki bölümde vektörleştirmenin önemi üzerinde durulmuş ve uygulama alanının (askeri amaçlı uygulamalar, coğrafi bilgi sistemleri uygulamaları, endüstriye yönelik uygulamalar, kalite kontrol uygulamaları vs.) çok geniş olduğuna değinilmişti. Vektörleştirme işlemi tabii ki bir çok adımdan oluşmaktadır. Basit bir vektörleştirme algoritma akışı Şekil 2.1' de verilmiştir. Görüldüğü gibi ortamdaki (resim düzlemi - ızgara gibi düşünülebilir) sayısal eğrilerin açık mı yoksa kapalı mı oldukları belirlendikten sonra standart geometrilere benzeyip benzemedikleri araştırılmaktadır. Bu konuda da oldukça yoğun çalışmalar sürdürülmektedir [49-63]. Şekil 2.1' de görüldüğü gibi eğer kapalı veya açık sayısal eğri, herhangi bir standart nesneye uydurulamamışsa köşe tesbiti aşamasına geçilir. Bu aşamada köşeler bulunur ve tanınmayan şekil elemanlarına ayrıştırılmış olur. Bu elemanların tanımlanabilmesi sonucunda karmaşık eğri de tanımlanmıştır.

Bu bölümde biz, kaynak taramalarımızdan hareketle gördük ki, köşe bulma ile ilgili makalelerde Rosenfeld-Johnston, Rosenfeld-Weszka ve Medioni-Yasumoto köşe bulma yöntemleri, makalede sunulan yöntemle karşılaştırma amacıyla kullanılıyor [38-43]. Biz de bu üç köşe bulma yöntemini, geliştirdiğimiz yeni köşe bulma yöntemleri ile karşılaştırma yapmak amacıyla kullandık. Dolayısıyla referans olarak seçilen bu üç yöntemin kısa açıklaması aşağıda bulunmaktadır.



Şekil 2.1 Sayısal bir eğrinin vektörleştirilmesinde izlenen yol.

Referans olarak seçilen bu üç yöntemle ilgili kısa açıklamalardan sonra bu çalışma çerçevesinde geliştirdiğimiz yedi adet yeni köşe bulma yöntemi açıklanmıştır. Toplam 10 (3' ü kaynaklardan, 7' si bizim geliştirdiğimiz) adet köşe bulma yöntemi aşağıdaki Şekil 2.2' de görülen geometrilere uygulanmıştır. Diğer uygulama sonuçları, şekil yoğunluğunun dikkati dağıtmasını engellemek amacıyla Ek A' da verilmiştir. Şekil 2.2' deki dört yaydan oluşan şekil, [38], [40] ve [41] tarafından kullanıldığı için alınmıştır.



Şekil 2.2 (a) Orijinal şekil (b) Sayısallaştırma sonucu
(c) Orijinal şekil (d) Sayısallaştırma sonucu

Köşe tesbiti ile ilgili çalışmalar iki sınıfta incelenebilir [33]; doğrudan gri tonlu görüntüye uygulanan yöntemler [33-35] ve gri tonlu görüntüdeki bölgelerin sınırlarının elde edilmesinden sonra uygulanan yöntemler [37-43]. Genellikle gri tonlu görüntüden sınırlar elde ediltikten sonraki çalışmalar daha yoğundur. Çünkü sayısal eğri üzerinde çalışmak daha basittir. Ayrıca daha az miktarda veri üzerinde işlem yapılmasından dolayı sonuçlara daha hızlı ulaşılmaktadır.

2.1 SEÇİLEN KÖŞE BULMA YÖNTEMLERİ

Burada önceden geliştirilmiş ve kaynak taramalarımızdan referans olarak verildiğini gördüğümüz [38-43] üç teknikten (Rosenfeld-Johnston, Rosenfeld-Weszka, Medioni-Yasumoto) kısaca bahsedilmektedir. Bu köşe bulma tekniklerinin kodları yazılmış ve Ek B' de sunulmuştur. Bu teknikler Şekil 2.2' de gösterilen geometrilere uygulanarak sonuçları açıklamaların arkasından verilmiştir. Diğer uygulamaları Ek A' bulunmaktadır.

Kapalı sayısal eğri olan *C*, *n* adet tamsayı koordinattan oluşan bir dizi olsun. Yani

$$C = \{P_i = (x_i, y_i), i = 1, \ldots, n\} \tag{2.1}$$

P_{i+1} , n modülüne göre P_i ' nin komşusudur.

2.1.1 Rosenfeld-Johnston Köşe Bulma Yöntemi

Bu metoda göre ilk olarak aşağıdaki gibi $2k$ adet vektör tanımlanır.

$$\vec{a}_{ik} = (x_i - x_{i+k}, y_i - y_{i+k}) \tag{2.2a}$$

$$\vec{b}_{ik} = (x_i - x_{i-k}, y_i - y_{i-k}) \tag{2.2b}$$

Bu vektörlerden hareketle her P_i noktası için k adet kosinüs değeri aşağıdaki formülden elde edilir.

$$\cos_{ik} = \frac{\vec{a}_{ik} \cdot \vec{b}_{ik}}{|\vec{a}_{ik}| \cdot |\vec{b}_{ik}|} \tag{2.3}$$

Burada

$$\vec{a}_{ik} \cdot \vec{b}_{ik} = (x_i - x_{i+k})(x_i - x_{i-k}) + (y_i - y_{i+k})(y_i - y_{i-k}) \tag{2.4a}$$

$$|\vec{a}_{ik}| = \sqrt{(x_i - x_{i+k})^2 + (y_i - y_{i+k})^2} \tag{2.4b}$$

Bilindiği gibi k adet vektör çifti arasındaki açının kosinüs değerleri için $-1 \leq \cos_{ik} \leq 1$ ifadesi geçerlidir ve en dar açı (0°) için $\cos_{ik} = 1$ ve düz bir doğru (180°) için $\cos_{ik} = -1$ dir. Bir yumuşatma unsuru olan *m* seçilerek (*n/10* veya *n/15*; *n*, kapalı sayısal eğrideki nokta sayısı [38]) her P_i noktası için *m* adet kosinüs değeri hesaplanır. Sonuçta aşağıdaki küme oluşur.

$$\{\cos_{ik}, k = 1,2,\ldots,m\} \tag{2.5}$$

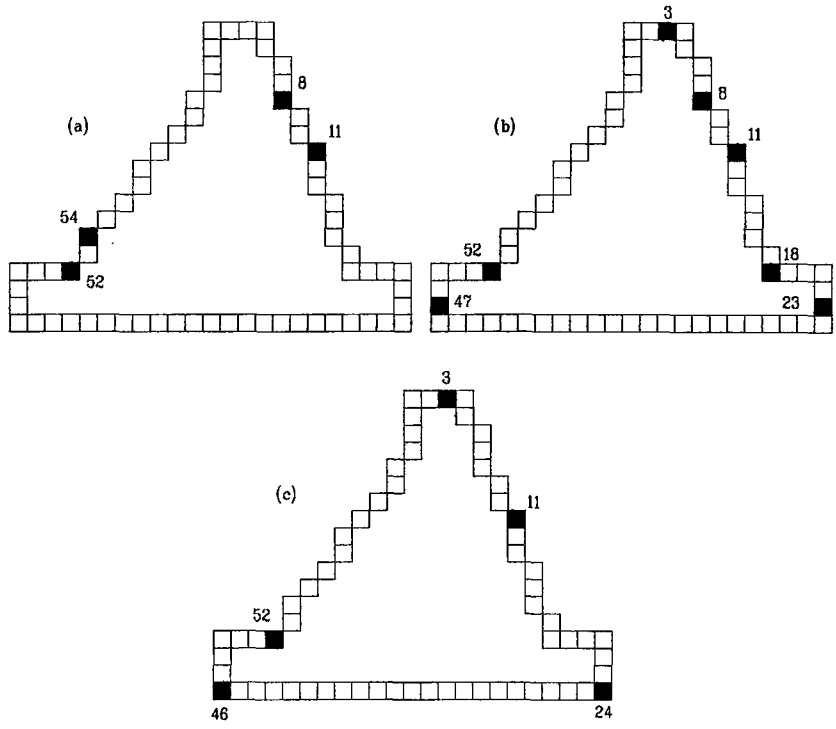
Her nokta için oluşturulan kosinüs değerleri kümesinden hareketle ilgilenilen noktanın kaç tane komşu noktası olduğunu belirten *h* değeri ve eğriliğin bir göstergesi kabul edilen kosinüs değeri aşağıdaki ifade yardımıyla bulunur.

$$\cos_{i,m} < \cos_{i,m-1} < \cos_{i,m-2} < \cdots < \cos_{i,h_i} \geq \cos_{i,h_i-1} \tag{2.6}$$

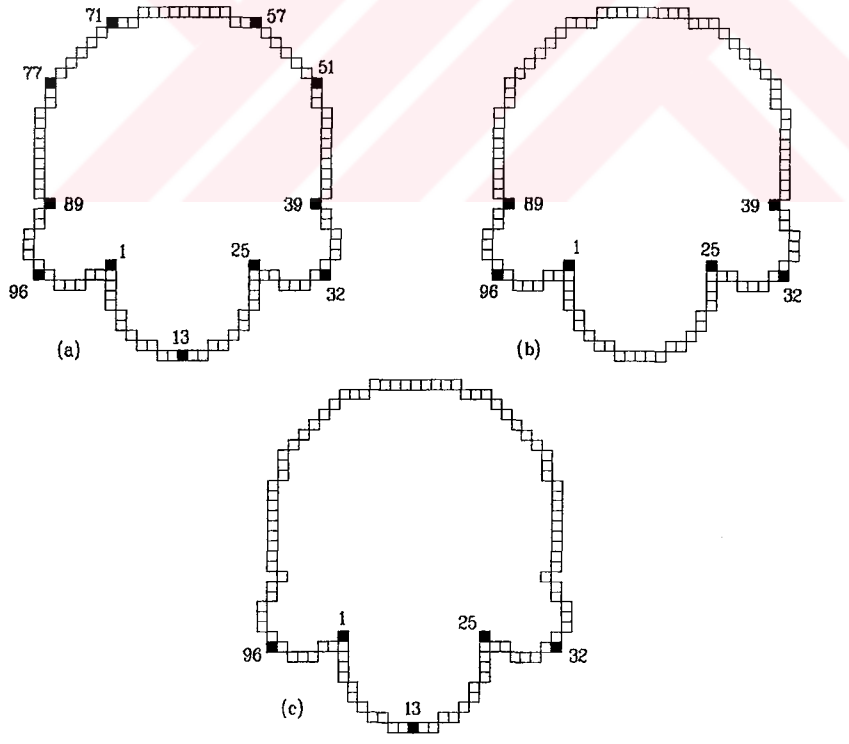
Son olarak P_i noktasının köşe olup olmadığı bir karşılaştırma ile belirlenir. P_i noktasının eğriliğini temsil eden $\cos_{i,h_i}$ değeri, h_i komşuluğundaki (2.7 ifadesi) noktaların $\cos_{j,h_j}$ değerleri ile karşılaştırılır. Eğer

$$\cos_{i,h_i} \geq \cos_{j,h_j}, \qquad |i-j| \leq \frac{h_i}{2} \tag{2.7}$$

eşitsizliği sağlanıyorsa P_i noktası belirtilen komşulukta lokal maksimum olur ve köşedir denir. Rosenfeld-Johnston köşe bulma yöntemi için bir akış şeması Ek B' de verilmiştir. Şekil 2.3 ve Şekil 2.4 bu yöntemin Şekil 2.2' deki geometrilere uygulama sonuçlarını göstermektedir. Ek A' da diğer uygulama sonuçları bulunmaktadır.



Şekil 2.3 Rosenfeld-Johnston yönteminin uygulaması. (a) m=3 iken bulunan köşeler. (a) m=5 iken bulunan köşeler. (c) m=7 iken bulunan köşeler.



Şekil 2.4 Rosenfeld-Johnston yönteminin uygulaması. (a) m=7 iken bulunan köşeler. (a) m=9 iken bulunan köşeler. (c) m=11 iken bulunan köşeler.

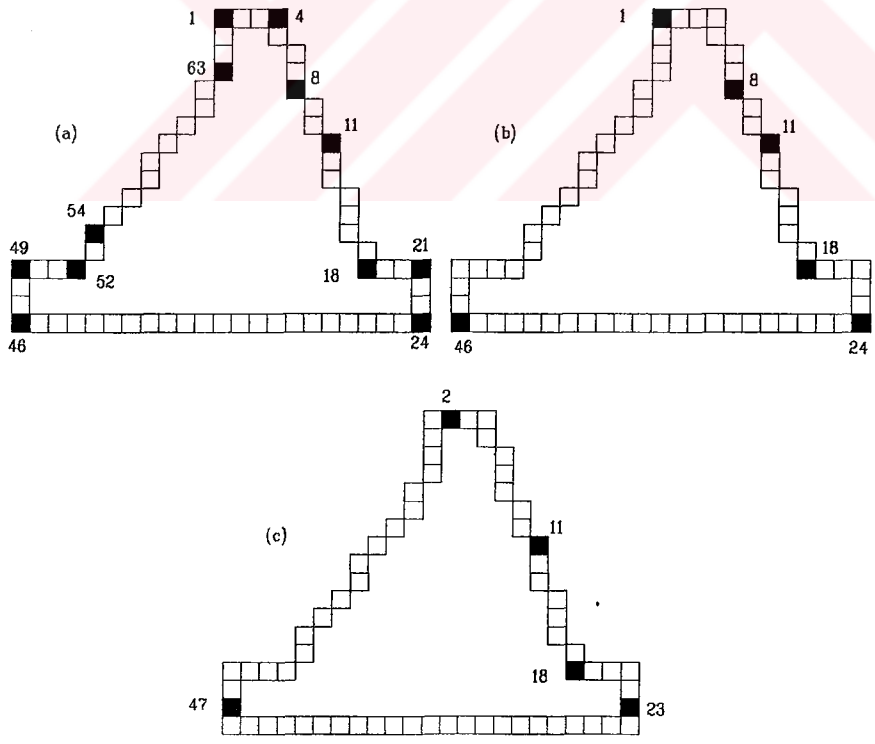
2.1.2 Rosenfeld-Weszka Köşe Bulma Yöntemi

Rosenfeld-Johnston köşe bulma yöntemi, özellikle köşeler birbirine yakınsa yanlış noktaların köşe olarak tesbit edilmesine sebep olur. Rosenfeld-Weszka yöntemi, bu problemin üstesinden gelmek için geliştirilmiştir [38]. Bu yöntem Rosenfeld-Johnston yönteminin aşağıda anlatılan şekilde değiştirilmiş hâlidir. Bir önceki yöntemde her nokta için k adet kosinüs değeri hesaplanmaktaydı. Bunlardan hareketle her nokta için yeni kosinüs değerleri aşağıdaki gibi hesaplanır ($k > 1$).

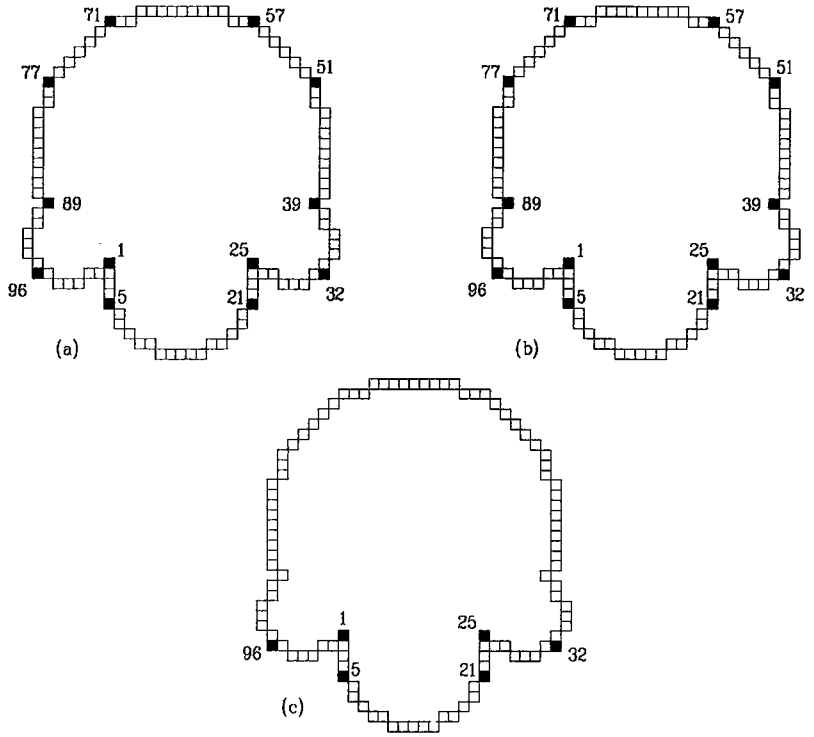
$$\overline{\cos}_{ik} = \frac{2}{k+2} \sum_{j=k/2}^{k} \cos_{ij} \quad , \qquad k = çift \tag{2.8a}$$

$$\overline{\cos}_{ik} = \frac{2}{k+3} \sum_{j=(k-1)/2}^{k} \cos_{ij} \, , \qquad k = tek \tag{2.8b}$$

Bundan sonra yeni kosinüs değerleri ile Rosenfeld-Johnston yöntemine devam edilir. Arkadaki Şekil 2.5 ve Şekil 2.6' da yöntemin Şekil 2.2' deki geometriye uygulama sonuçları verilmiştir. Diğer uygulamalar Ek A' da bulunmaktadır.



Şekil 2.5 Rosenfeld-Weszka yönteminin uygulaması. (a) m=3 iken bulunan köşeler. (b) m=5 iken bulunan köşeler. (c) m=7 iken bulunan köşeler.



Şekil 2.6 Rosenfeld-Weszka yönteminin uygulaması. (a) m=7 iken bulunan köşeler. (b) m=9 iken bulunan köşeler (m=7 ile aynı). (c) m=11 iken bulunan köşeler.

2.1.3 Medioni-Yasumoto Köşe Bulma Yöntemi

Bu yöntem, sayısal eğrimize, parametrik üçüncü derece B-spline eğri uydurma esasına dayanır. İlgilenilen noktanın uydurulan eğriye mesafesi ve eğriliği, belirtilen bir sınırı geçiyorsa ilgilenilen noktanın köşe olduğu sonucuna varılır [41].

Parametrik eğrinin aşağıdaki gibi olduğu düşünülürse

$$x = f(t) = a_1 t^3 + b_1 t^2 + c_1 t + d_1 \qquad (2.9a)$$

$$y = g(t) = a_2 t^3 + b_2 t^2 + c_2 t + d_2 \quad , \quad 0 \le t \le 1 \qquad (2.9b)$$

$t = 0$ daki eğrilik

$$c_v(o) = \frac{2(b_2 c_1 - b_1 c_2)}{(c_1^{\,2} + c_2^{\,2})^{3/2}} \qquad (2.10)$$

Eğriye üçüncü dereceden bir B-spline fonksiyon uydurulduğunda

$$x(t) = TM_b G_x \tag{2.11a}$$

$$y(t) = TM_b G_y \tag{2.11b}$$

Burada

$$M_b = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}, \quad T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

ve i noktasındaki G_x ve G_y matrisleri

$$G_x^{\,i} = \begin{bmatrix} x_{i-1} & x_i & x_{i+1} & x_{i+2} \end{bmatrix}^T \tag{2.12a}$$

$$G_y^{\,i} = \begin{bmatrix} y_{i-1} & y_i & y_{i+1} & y_{i+2} \end{bmatrix}^T \tag{2.12b}$$

ve uydurulan eğrinin P_{i-1}, P_i, P_{i+1} ve P_{i+2} kontrol noktalarından geçtiği kabul edilir.

(2.12) matrislerini (2.11) denklemine koyup (2.9) denklemleri ile karşılaştırma yapıldığında

$$a_1 = (-x_{i-1} + 3x_i - 3x_{i+1} + x_{i+2})/6 \tag{2.13a}$$

$$b_1 = (x_{i-1} - 2x_i + x_{i+1})/2 \tag{2.13b}$$

$$c_1 = (-x_{i-1} + x_{i+1})/2 \tag{2.13c}$$

$$d_1 = (x_{i-1} + 4x_i + x_{i+1})/6 \tag{2.13d}$$

$$a_2 = (-y_{i-1} + 3y_i - 3y_{i+1} + y_{i+2})/6 \tag{2.13e}$$

$$b_2 = (y_{i-1} - 2y_i + y_{i+1})/2 \tag{2.13f}$$

$$c_2 = (-y_{i-1} + y_{i+1})/2 \tag{2.13g}$$

$$d_2 = (y_{i-1} + 4y_i + y_{i+1})/6 \tag{2.13h}$$

ve $t = 0$ daki eğrilik için

$$c_v = \frac{4\left[(x_{i+1}-x_{i-1})(y_{i-1}-2y_i+y_{i+1})-(y_{i+1}-y_{i-1})(x_{i-1}-2x_i+x_{i+1})\right]}{\left[(x_{i+1}-x_{i-1})^2+(y_{i+1}-y_{i-1})^2\right]^{3/2}} \quad (2.14)$$

Sayısal eğrimiz üzerindeki P_i ve $t=0$ iken uydurulan eğri üzerindeki P_{bi} arasındaki $\delta=(\delta_x,\delta_y)$ mesafesini aşağıdaki gibi hesaplayabiliriz.

$$\delta_x = d_1 - x_i = \frac{x_{i-1}}{6} - \frac{x_i}{3} + \frac{x_{i+1}}{6} \quad (2.15a)$$

$$\delta_y = d_2 - y_i = \frac{y_{i-1}}{6} - \frac{y_i}{3} + \frac{y_{i+1}}{6} \quad (2.15b)$$

Uydurulan eğri yeni bir B-spline eğri oluşturmak için δ kadar taşınırsa yeni eğrideki noktalar $\tilde{P}_{i-1}$, $\tilde{P}_i$, $\tilde{P}_{i+1}$ ve $\tilde{P}_{i+2}$, eski eğrideki P_{i-1}, P_i, P_{i+1} ve P_{i+2} cinsinden aşağıdaki gibi ifade edilir.

$$\tilde{P}_{i-1} = \frac{P_{i-2}}{6} + \frac{2P_{i-1}}{3} + \frac{P_i}{6} \quad (2.16a)$$

$$\tilde{P}_i = \frac{P_{i-1}}{6} + \frac{2P_i}{3} + \frac{P_{i+1}}{6} \quad (2.16b)$$

$$\tilde{P}_{i+1} = \frac{P_i}{6} + \frac{2P_{i+1}}{3} + \frac{P_{i+2}}{6} \quad (2.16c)$$

$$\tilde{P}_{i+2} = \frac{P_{i+1}}{6} + \frac{2P_{i+2}}{3} + \frac{P_{i+3}}{6} \quad (2.16d)$$

Yeni B-spline eğrimiz

$$x_n(t) = a_1't^3 + b_1't^2 + c_1't + d_1' \quad (2.17a)$$

$$y_n(t) = a_2't^3 + b_2't^2 + c_2't + d_2' \quad (2.17b)$$

Yukarıdaki (2.17) denklemlerindeki katsayılar

$$a_1' = \frac{x_{i+3}-x_{i-1}+x_{i+2}-x_{i-2}}{36} + \frac{2(x_i-x_{i+1})}{27} \quad (2.18a)$$

$$a_2' = \frac{y_{i+3}-y_{i-1}+y_{i+2}-y_{i-2}}{36} + \frac{2(y_i-y_{i+1})}{27} \quad (2.18b)$$

$$b_1' = \frac{x_{i+2} + x_{i-2}}{12} + \frac{x_{i+1} + x_{i-1}}{6} - \frac{x_i}{2} \tag{2.18c}$$

$$b_2' = \frac{y_{i+2} + y_{i-2}}{12} + \frac{y_{i+1} + y_{i-1}}{6} - \frac{y_i}{2} \tag{2.18d}$$

$$c_1' = \frac{x_{i+1} - x_{i-1}}{3} + \frac{x_{i+2} - x_{i-2}}{12} \tag{2.18e}$$

$$c_2' = \frac{y_{i+1} - y_{i-1}}{3} + \frac{y_{i+2} - y_{i-2}}{12} \tag{2.18f}$$

$$d_1' = \frac{x_{i+2} + x_{i-2}}{36} + \frac{2(x_{i+1} + x_{i-1})}{9} + \frac{x_i}{2} \tag{2.18g}$$

$$d_2' = \frac{y_{i+2} + y_{i-2}}{36} + \frac{2(y_{i+1} + y_{i-1})}{9} + \frac{y_i}{2} \tag{2.18h}$$

Yeni eğrimizin katsayılarını (2.18 denklemleri) ve noktalarını (2.16 denklemleri) hesaplarken P_{i-2} ve P_{i+3} noktaları da devreye girmektedir. Yeni B-spline eğri üzerindeki P_{bi}' ve sayısal eğrimizdeki P_i noktaları arasındaki mesafe $\delta' = (\delta_x', \delta_y')$

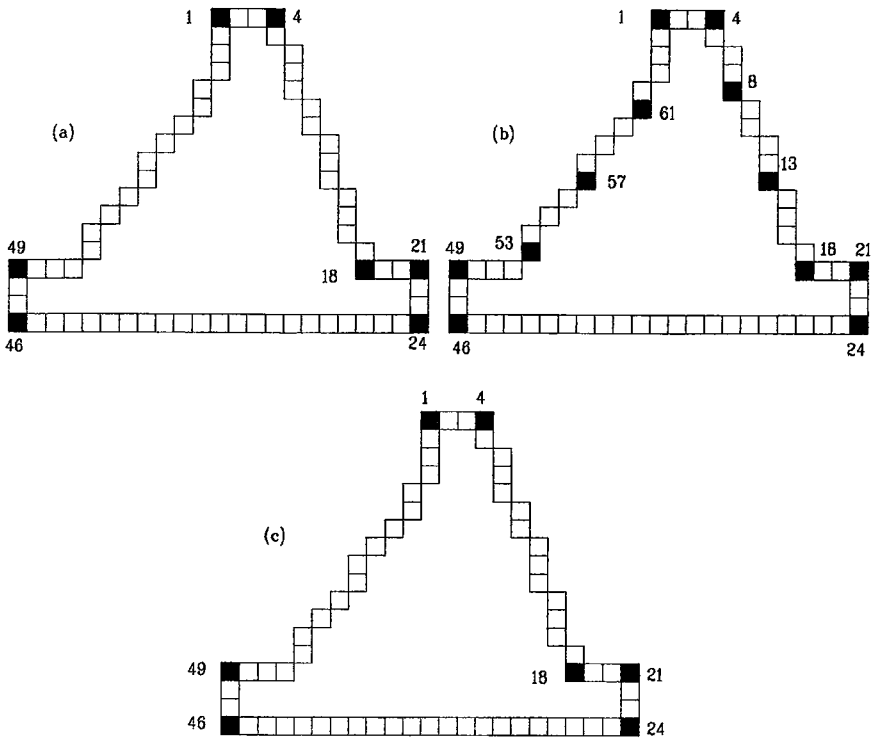
$$\delta_x' = d_1' - x_i = \frac{x_{i-2} + x_{i+2}}{36} + \frac{2(x_{i+1} + x_{i-1})}{9} - \frac{x_i}{2} \tag{2.19a}$$

$$\delta_y' = d_2' - y_i = \frac{y_{i-2} + y_{i+2}}{36} + \frac{2(y_{i+1} + y_{i-1})}{9} - \frac{y_i}{2} \tag{2.19b}$$

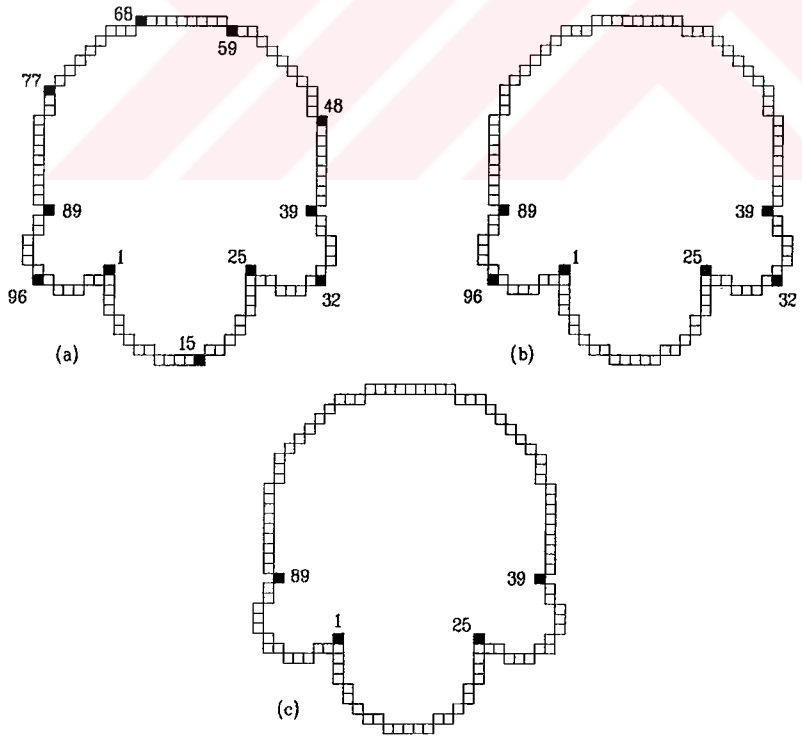
P_{bi}' noktasındaki eğrilik

$$c_v' = \frac{2(b_2'c_1' - b_1'c_2')}{(c_1'^2 + c_2'^2)^{3/2}} \tag{2.20}$$

Kontrol noktalarındaki düzensizlikler B-spline eğri tarafından yumuşatılırlar ve c_v' azalır. Diğer yandan sayısal eğrimizdeki yön değişimi artarsa δ farkı da artacaktır. Dolayısıyla köşe tesbitinde hem eğrilik (c_v') hem de yer değiştirme (δ) dikkate alınırsa sonuçlar daha gerçeğe yakın olur. Yani ilgilenilen noktanın köşe olabilmesi için: (1) δ farkı, verilen bir eşik değerinden büyük olmalı yani $\|\delta'\| \geq \delta_t$; (2) eğrilik ($c_v'$) verilen bir eşik değerden büyük olmalı; (3) eğrilik belirtilen komşulukta maksimum olmalı. Aşağıda yöntemin uygulama sonuçları verilmiştir (Şekil 2.7 ve Şekil 2.8).



Şekil 2.7 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.5, δ_t=0.25, $|i-j|$=2 (b) c_t=0.3, δ_t=0.25, $|i-j|$=2 (c) c_t=0.3, δ_t=0.3, $|i-j|$=2 iken köşeler.



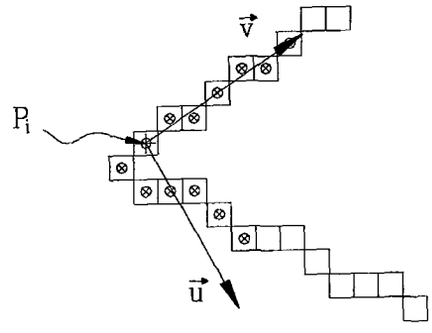
Şekil 2.8 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.2, $|i-j|$=5 (b) c_t=0.3, δ_t=0.3, $|i-j|$=5 (c) c_t=0.3, δ_t=0.4, $|i-j|$=5 iken köşeler.

2.2 GELİŞTİRİLEN KÖŞE BULMA YÖNTEMLERİ

Bu kısımda köşe tesbiti için geliştirdiğimiz yöntemlerden en başarılı olanları etkinlik sırasına göre anlatılmaktadır. Her yöntem Şekil 2.2' deki geometrilere uygulanarak sonuçlar verilmiştir. Ayrıca Ek A' da başka şekiller üzerinde uygulama sonuçları sunulmuştur. Sayısal eğrinin üzerindeki bir beneğin köşe kabul edilebilmesi için o benekteki eğriliğin belli bir değeri aşması beklenir. Geliştirdiğimiz köşe bulma yöntemlerinde her bir benek için eğrilik bulunmaya çalışılır. Aşağıda açıklanan ilk ve son yöntemler haricindekilerde (sp3min, bartem, moment, spthr ve spmin) her benek için teğet bulunmaya çalışılmıştır. Çünkü eğriliğin geometrik anlamından hareketle, komşu beneklerin teğetlerinin eğimleri arasındaki farkı, eğriliğin bir göstergesi olarak kabul edebiliriz. Birinci (Lsq) ve yedinci (hatmin) yöntemlerde doğrudan her benek için eğrilik bulunmaya çalışılmaktadır.

2.2.1 Birinci Yöntem (LSQ)

Bu yöntemde ilgilenilen P_i noktası için $\vec{u}$ ve $\vec{v}$ vektörleri oluşturulur. $\vec{u}$ nün oluşturulması için öndeki k adet noktadan (P_{i+1}, P_{i+2}, ..., P_{i+k}) en küçük kareler metoduyla (LSQ) geçen doğru bulunur. Şekil 2.9' da görüldüğü gibi bu doğru, $\vec{u}$ vektörünün de yönüdür. $\vec{v}$ de aynı mantıkla P_{i-1}, P_{i-2}, ..., P_{i-k} noktaları dikkate alınarak oluşturulur.



Şekil 2.9 $\vec{u}$ ve $\vec{v}$ nin yönlerinin bulunması

$\vec{u}$ vektörünün doğrultusunu bulmak için

$$\chi = \sum_{k} \{y_{i+k} - F(x_{i+k})\}^2 \qquad (2.21)$$

fonksiyonu minimum yapılır. Burada $F(x)=ax+b$ lineer formunu seçelim. Ayrıca bir şartımız da $y_i=ax_i+b$, yani doğrunun P_i noktasından geçmesi istenir. Dolayısıyla (2.21) denklemi

$$\begin{aligned}\chi &= (y_{i+1}-y_i-a(x_{i+1}-x_i))^2+(y_{i+2}-y_i-a(x_{i+2}-x_i))^2+\ldots+\\ &\quad (y_{i+k}-y_i-a(x_{i+k}-x_i))^2\end{aligned} \tag{2.22}$$

durumuna gelir. Fonksiyonun minimum olması için

$$\frac{d\chi}{da}=0 \tag{2.23}$$

şartını sağlayan a değeri bulunmalı. (2.23) denklemindeki türev alınıp denklem düzenlendiğinde

$$a=\frac{\sum_{j=1}^{k}(y_{i+j}-y_i)(x_{i+j}-x_i)}{\sum_{j=1}^{k}(x_{i+j}-x_i)^2} \tag{2.24}$$

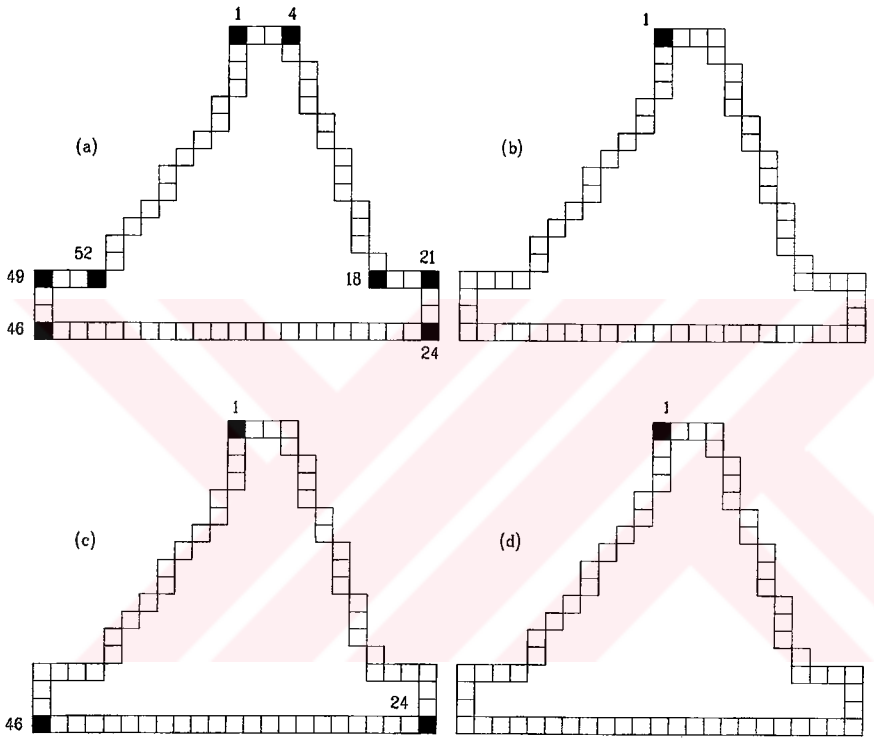
ifadesi çıkar. $\vec{v}$ için de aynı yol izlenerek vektörün üzerinde bulunduğu doğrunun eğimi ($y=cx+d$ kabuluyle)

$$c=\frac{\sum_{j=1}^{k}(y_{i-j}-y_i)(x_{i-j}-x_i)}{\sum_{j=1}^{k}(x_{i-j}-x_i)^2} \tag{2.25}$$

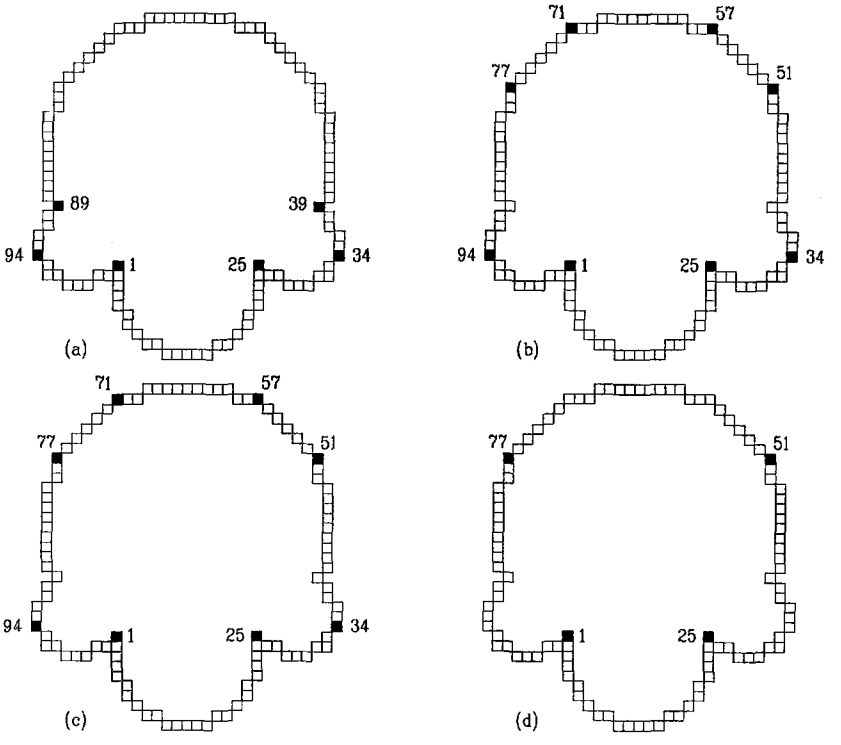
bulunur. $\vec{u}(u1,u2)$ ve $\vec{v}(v1,v2)$ vektörlerinin skaler çarpımından bu iki vektör arasındaki açı

$$\cos\alpha=\frac{u1.v1+u2.v2}{\sqrt{u1^2+u2^2}.\sqrt{v1^2+v2^2}} \tag{2.26}$$

bulunur. Her i noktası için bulunan $\cos\alpha_i$ değerleri arasından, belirtilen komşulukta lokal maksimum olan ve komşu noktadakilerden belirli bir t değeri kadar büyük olan noktalara, köşe noktaları denir. Aşağıdaki Şekil 2.10 ve Şekil 2.11, yöntemi kullanan "lsq.c" programının sonuçlarıdır. Ek A' da diğer uygulama sonuçları bulunmaktadır. Şekillerden görüldüğü gibi yöntem, geliştirilmiş olan diğer yöntemlerden oldukça başarılıdır. Son bölümde bu yöntemin, kaynaklardaki yöntemler ile karşılaştırması verilmiştir.



Şekil 2.10 LSQ yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



Şekil 2.11 LSQ yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6

2.2.2 İkinci Yöntem (Sp3min)

Bu yöntemde P_i noktasındaki teğet bulunmaya çalışılır. P_i noktasındaki teğeti bulmak için aşağıdaki formda üçüncü dereceden parametrik bir eğri, sayısal eğrimize uydurulmaya çalışılır. P_i noktasının önündeki k adet ve arkasındaki k adet nokta parabole minimum uzaklıkta olacak şekilde parabol katsayıları hesaplanacaktır. Spmin yönteminde de aynı mantığın kullanıldığını belirtelim.

$$x(t) = at^3 + bt^2 + ct + d \tag{2.27a}$$

$$y(t) = et^3 + ft^2 + gt + h \tag{2.27b}$$

Dolayısıyla P_i noktası için açık formda hata fonksiyonu aşağıdaki denklemdeki gibi olacaktır.

$$
\begin{aligned}
E_i = \; & (-k^3a+k^2b-kc+x_i-x_{i-k})^2+\ldots+(-27a+9b-3c+x_i-x_{i-3})^2 + \\
& (-8a+4b-2c+x_i-x_{i-2})^2+(-a+b-c+x_i-x_{i-1})^2 + \\
& (a+b+c+x_i-x_{i+1})^2+(8a+4b+2c+x_i-x_{i+2})^2 + \\
& (27a+9b+3c+x_i-x_{i+3})^2+\ldots+(k^3a+k^2b+kc+x_i-x_{i+k})^2 + \\
& (-k^3e+k^2f-kg+y_i-y_{i-k})^2+\ldots+(-27e+9f-3g+y_i-y_{i-3})^2 + \\
& (-8e+4f-2g+y_i-y_{i-2})^2+(-e+f-g+y_i-y_{i-1})^2 + \\
& (e+f+g+y_i-y_{i+1})^2+(8e+4f+2g+y_i-y_{i+2})^2 + \\
& (27e+9f+3g+y_i-y_{i+3})^2+\ldots+(k^3e+k^2f+kg+y_i-y_{i+k})^2 \qquad (2.28)
\end{aligned}
$$

P_i noktasının 5 adet önünde ve 4 adet arkasındaki noktaların dikkate alınmasıyla hazırlanan hata fonksiyonunun a, b ve c katsayılarına göre türevlerinin alınıp sıfıra eşitlenmesi sonucunda aşağıdaki denklem sistemi elde edilir.

$$
\begin{bmatrix} 50810 & -6250 & 2666 \\ -6250 & 2666 & -250 \\ 2666 & -250 & 170 \end{bmatrix}
\begin{bmatrix} a \\ b \\ c \end{bmatrix} =
\begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad (2.29)
$$

Yukarıdaki sistemde A, B ve C aşağıdaki gibi tanımlıdır.

$$A = 2x_{i+1}+16x_{i+2}+54x_{i+3}+128x_{i+4}-2x_{i-1}-16x_{i-2}-54x_{i-3}-128x_{i-4}-250x_{i-5}+250x_i \qquad (2.30a)$$

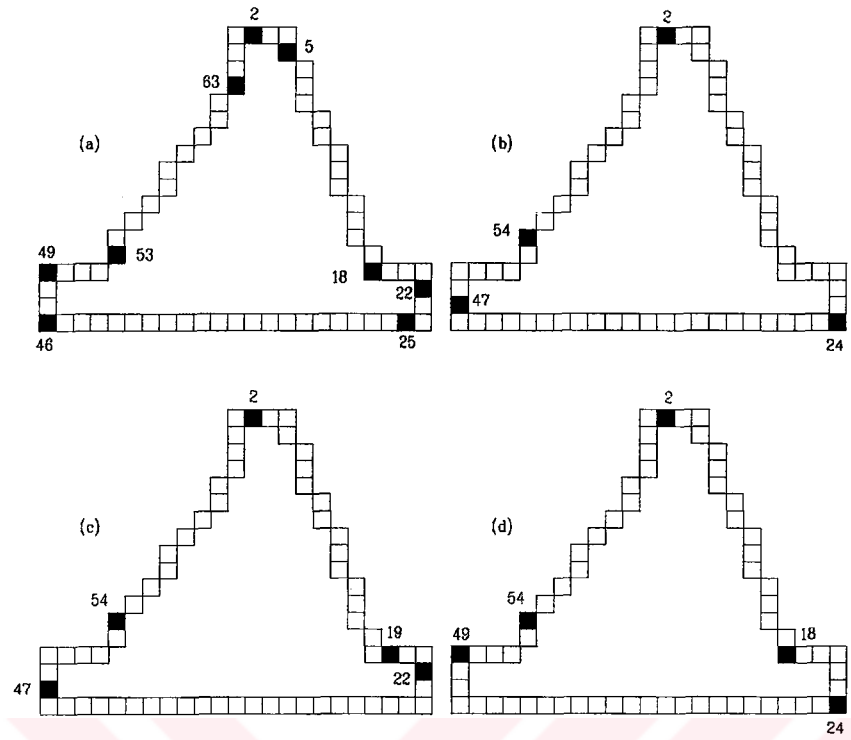
$$B = 2x_{i+1}+8x_{i+2}+18x_{i+3}+32x_{i+4}+2x_{i-1}+8x_{i-2}+18x_{i-3}+32x_{i-4}+50x_{i-5}-170x_i \qquad (2.30b)$$

$$C = 2x_{i+1}+4x_{i+2}+6x_{i+3}+8x_{i+4}-2x_{i-1}-4x_{i-2}-6x_{i-3}-8x_{i-4}-10x_{i-5}+10x_i \qquad (2.30c)$$

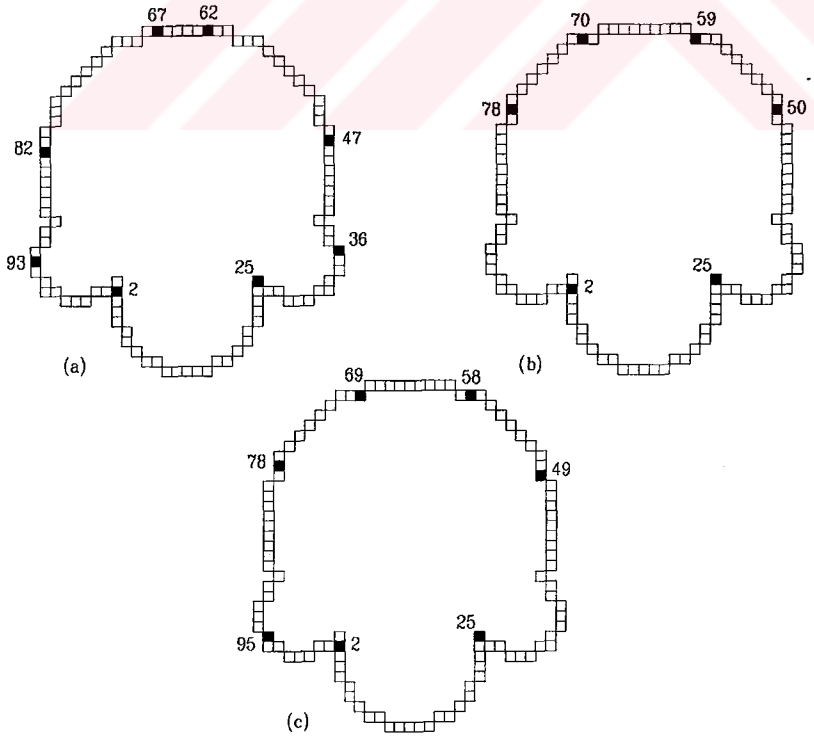
(2.29) denklemlerinin çözümüyle a, b ve c katsayıları bulunur. Hata fonksiyonunun e, f ve g katsayılarına göre türevleri alınıp sıfıra eşitlendiğinde (2.29) denklemlerine benzer denklem takımı çıkar. Buradan da e, f ve g katsayıları bulunur.

Katsayıların belirlenmesi sonucunda (2.27) denkleminden hareketle P_i noktasındaki teğet aşağıdaki denklemden bulunabilir. Sonuç (2.32) denklemindeki gibidir.

$$\frac{x-x_i}{3at^2+2bt+c\big|_{t=0}} = \frac{y-y_i}{3et^2+2ft+g\big|_{t=0}} \qquad (2.31)$$



Şekil 2.12 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



Şekil 2.13 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4

$$y = \frac{g}{c}x - \frac{g}{c}x_i + y_i \tag{2.32}$$

olarak bulunur. Artık P_i noktasındaki teğetin belli olmasıyla c_i değerleri elde edilir. P_i noktasının köşe olabilmesi için c_i ' nin bir eşik değerinden büyük ve belirtilen bir komşulukta maksimum olması gerekir. Şekil 2.12 ve Şekil 2.13, bu yöntemi kullanan "sp3min.c" programının sonuçlarını gösterir. Ek A' da yöntemin diğer geometriler üzerindeki uygulama sonuçları bulunmaktadır. Şekillerden görülmektedir ki, bu yöntem de çoğunlukla gerçek köşelere yakın benekleri doğru köşe olarak yakalıyor. Fakat istenen hassasiyete bu yöntem ulaşamamaktadır.

2.2.3 Üçüncü Yöntem (Bartem)

Bartem adını verdiğimiz üçüncü yöntemde kapalı sayısal eğrimizin her noktası için bir teğet oluşturulmaya çalışılır. Daha sonra her nokta (benek) için bulunan teğet eğimlerinin, belirtilen komşulukta fazlaca değişim gösterdiği nokta, köşedir denir.

P_i noktasındaki teğeti oluşturmak için şöyle bir yol izlenir: P_i noktasındaki teğeti, ilgilenilen noktanın önündeki k adet ve arkasındaki k adet nokta etkiliyor olsun. P_i noktasındaki teğet

$$y = mx + n \tag{2.33}$$

$2k$ adet noktanın teğete olan toplam mesafesinin bir göstergesi

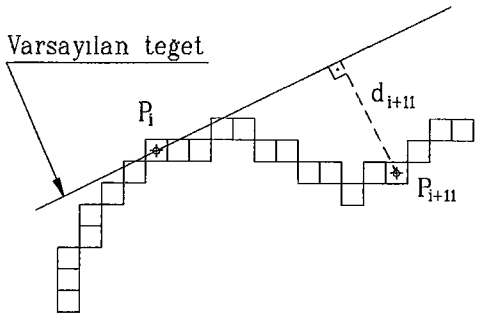
$$M = d_{i-k}^2 + d_{i-k+1}^2 + \ldots + d_{i-1}^2 + d_{i+1}^2 + \ldots + d_{i+k-1}^2 + d_{i+k}^2 \tag{2.34}$$

P_{i+j} noktasının teğete olan mesafesi

$$d_{i+j}^2 = \left(\frac{mx_{i+j} - y_{i+j} + n}{m + \frac{1}{m}} \right)^2 \left(\frac{m^2 + 1}{m^2} \right) \tag{2.35a}$$

$$n = y_i - mx_i \tag{2.35b}$$

Şekil 2.14' de varsayılan teğete örneğin P_{i+11} noktasının uzaklığı gösterilmektedir.



Şekil 2.14 P_{i+11} noktasının P_i noktasındaki teğete mesafesi.

(2.34) ve (2.35) denklemlerinden toplam mesafenin bir ölçüsü olan *M* için

$$M = \sum_{\substack{j=-k \\ j\neq 0}}^{k} d_{i+j}^2 \tag{2.36}$$

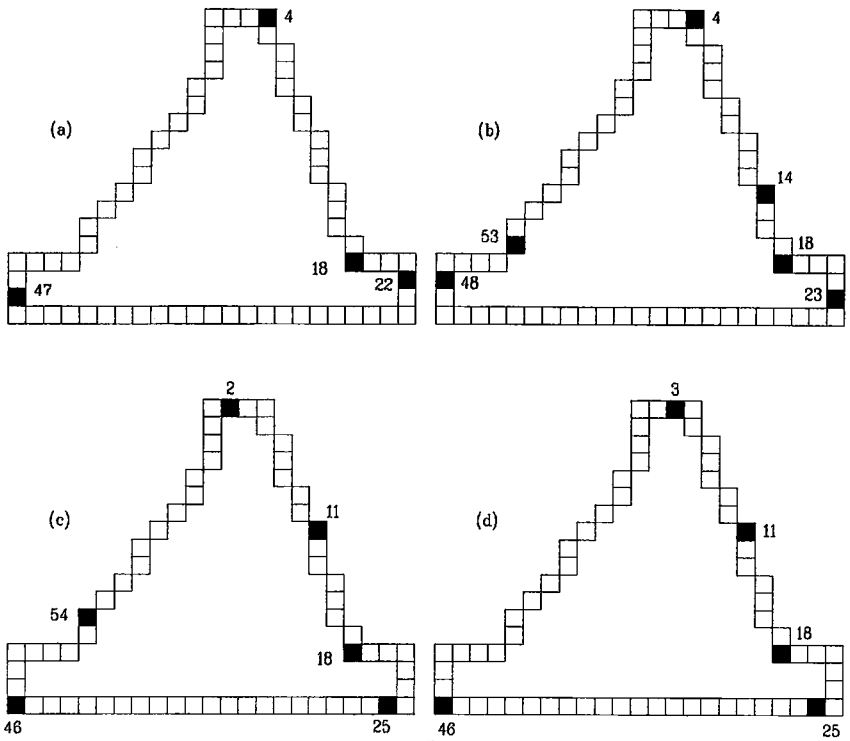
yazılabilir. M' nin minimum olabilmesi için aşağıdaki denklem teğetin eğimi olan m tarafından sağlanmalıdır.

$$\frac{dM}{dm} = \sum_{\substack{j=-k \\ j\neq 0}}^{k} \frac{d}{dm}(d_{i+j}^2) = 0 \tag{2.37}$$

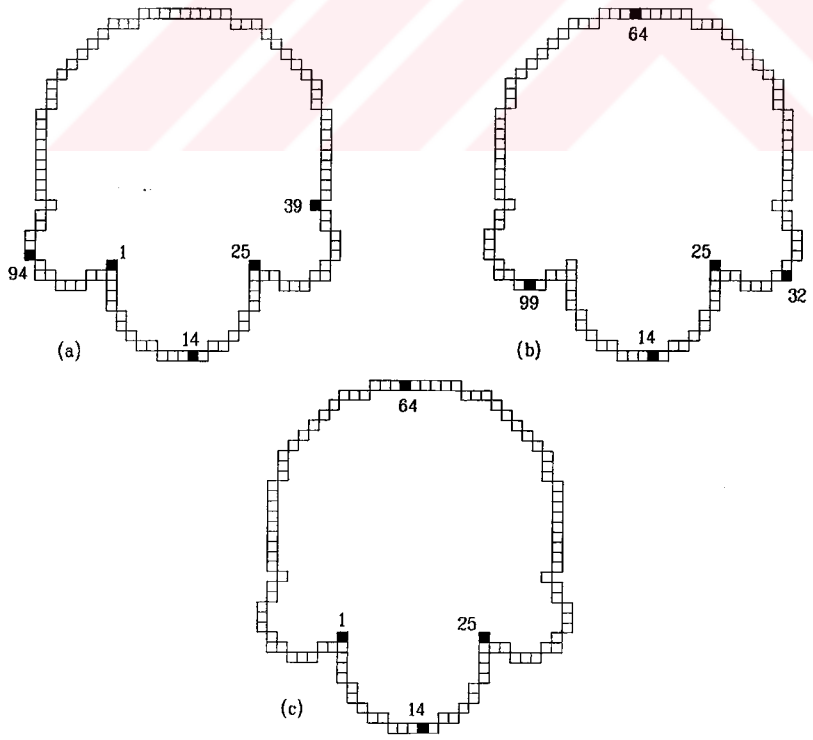
Yukarıdaki denklem uygulandığında ve bir takım ara işlemlerden sonra

$$\left(\sum_{j=-k}^{k} X_{i+j}^2\right)(2m-1) + \left(\sum_{j=-k}^{k} X_{i+j}Y_{i+j}\right)(m^2+1) - \left(\sum_{j=-k}^{k} Y_{i+j}^2\right) m = 0 \tag{2.38}$$

elde edilir. Burada $X_{i+j} = x_{i+j} - x_i$ ve $Y_{i+j} = y_{i+j} - y_i$ tanımları yapılmıştır.



Şekil 2.15 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



Şekil 2.16 Bartem yönteminin uygulaması. (a) k_1=4, k_2=3 (b) k_1=4, k_2=4 (c) k_1=5, k_2=4

(2.38) denkleminden yararlanılarak kapalı eğrimizin her noktası için eğim (m) değerleri hesaplanır. Bu eğim değerlerinin değişiminin, belirtilen komşuluk içinde maksimum olduğu noktalar köşe noktaları olarak atanır. Şekil 2.15 ve Şekil 2.16, bu yöntemi kullanan "bartem.c" programının sonuçlarını gösterir. Ek A' da diğer uygulama sonuçları bulunmaktadır. Geliştirilen bu yöntem köşelerin açıkca belli olmadığı ve köşelerin birbirine yakın olması durumunda yanlış benekleri köşe olarak bulmaktadır. Dolayısıyla bu yöntemden istenen performansın alınamadığını söyleyebiliriz.

2.2.4 Dördüncü Yöntem (Moment)

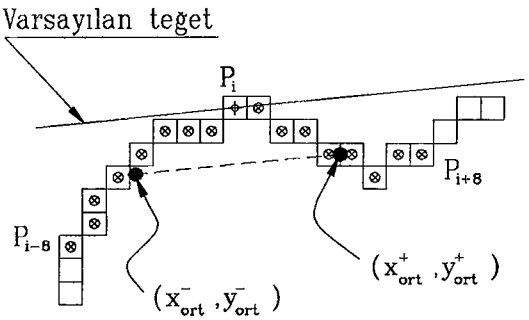
Bu yöntemde de kapalı sayısal eğrimizin her noktası için bir teğet oluşturulur. Sonraki aşamada her nokta için bulunan teğet eğimlerinin değişimine göre ilgilenilen noktanın köşe olup olmadığı kararına varılır.

İlgilenilen P_i noktasındaki teğet şöyle oluşturulur: P_i noktasının önündeki k adet ve arkasındaki k adet nokta dikkate alınır. Öndeki k adet noktanın $(P_{i+1}, P_{i+2}, \ldots, P_{i+k})$ ağırlık merkezi

$$x_{ort}^{+} = \left(\sum_{j=1}^{k} x_{i+j}\right) \Big/ k \quad , \quad y_{ort}^{+} = \left(\sum_{j=1}^{k} y_{i+j}\right) \Big/ k \tag{2.39}$$

Arkadaki k adet noktanın $(P_{i-1}, P_{i-2}, \ldots, P_{i-k})$ ağırlık merkezi ise

$$x_{ort}^{-} = \left(\sum_{j=1}^{k} x_{i-j}\right) \Big/ k \quad , \quad y_{ort}^{-} = \left(\sum_{j=1}^{k} y_{i-j}\right) \Big/ k \tag{2.40}$$



Şekil 2.17 P_i noktasında farzedilen teğet

Şekil 2.17' de görüldüğü gibi P_i noktasındaki teğetin $(x_{ort}^{+}, y_{ort}^{+})$ ve $(x_{ort}^{-}, y_{ort}^{-})$ noktalarından geçen doğruya paralel olduğu farzedilmiştir.

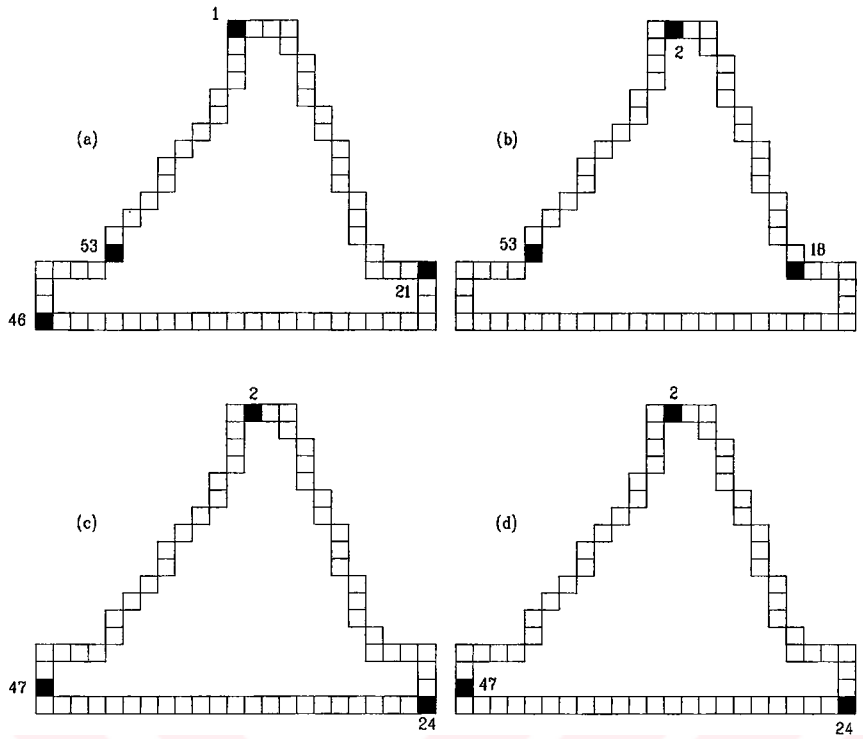
Dolayısıyla farzedilen teğetin eğimi

$$m = (y_{ort}^{+} - y_{ort}^{-}) / (x_{ort}^{+} - x_{ort}^{-}) \tag{2.41}$$

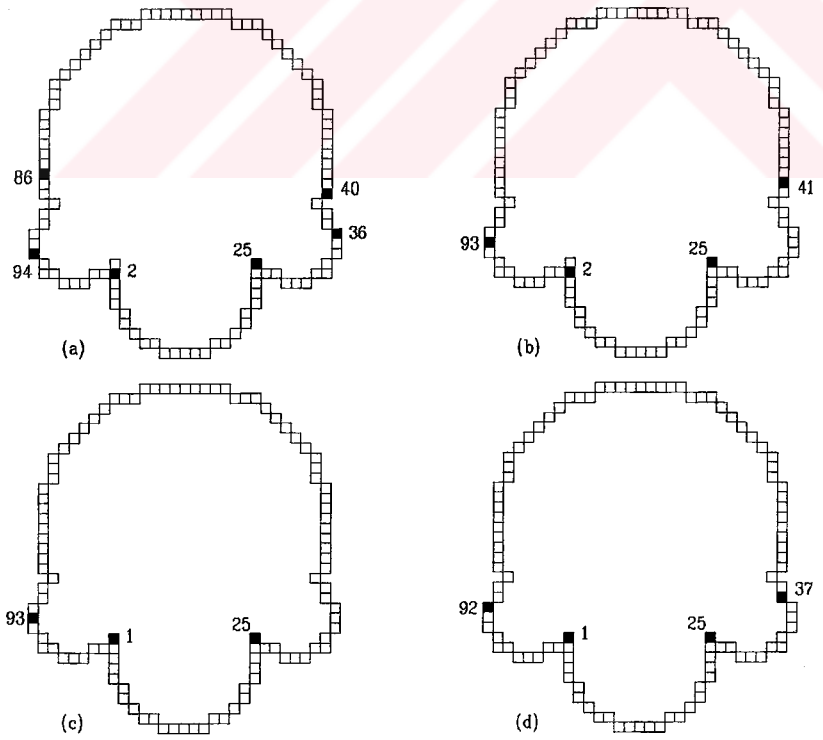
Her noktanın eğimi bulunduktan sonra eğriliğin göstergesi olan c_i

$$c_i = \alpha_i - \alpha_{i-1} \quad , \qquad \alpha = \text{atan}(m) \tag{2.42}$$

hesaplanır. P_i noktasının köşe olabilmesi için c_i, bir eşik değerinden büyük olmalı ve belirlenen komşulukta (k komşuluğu) maksimum olmalıdır. Aşağıdaki Şekil 2.18 ve Şekil 2.19 bu yöntem için yazılan "moment.c" programının uygulamaları verilmiştir. Değişik geometriler üzerindeki sonuçlar, Ek A' da gösterilmiştir. Sonuçların istenen hassasiyette bulunamaması, başka algoritmalar geliştirmemize sebep olmaktadır. Burada da maalesef köşelerin çok belirgin olduğu durumlarda iyi neticeler alınmaktadır.



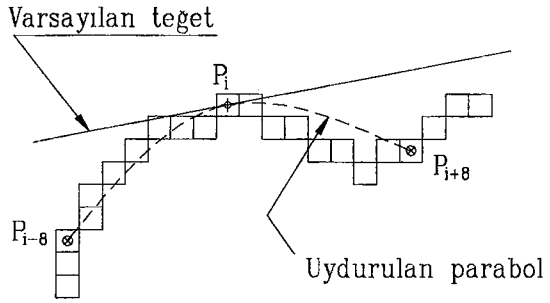
Şekil 2.18 Moment yönteminin uygulaması. (a) k=2 (b) k=3 (c) k=4 (d) k=5



Şekil 2.19 Moment yönteminin uygulaması. (a) k=3 (b) k=5 (c) k=6 (d) k=7

2.2.5 Beşinci Yöntem (Spthr)

Bu yöntemde P_i noktasındaki varsayılan teğet, P_i noktasından geçen parametrik bir parabolden yararlanılarak elde edilir. Şekil 2.20, bu yöntemde kullanılan teğetin nasıl bir mantıkla oluşturulduğunu göstermektedir.



Şekil 2.20 P_i noktasında farzedilen teğet

P_i, P_{i+k} ve P_{i-k} noktalarından geçen parabolümüz

$$x(t) = at^2 + bt + c \tag{2.43a}$$

$$y(t) = dt^2 + et + f \tag{2.43b}$$

olsun. k, parabolün P_i noktasından ne kadar ilerideki ve gerideki noktadan geçeceğini belirtir. Parabolün geçeceği noktalar belli olduğu için katsayılar aşağıdaki şekilde bulunur.

$$a = \frac{x_{i+k} + x_{i-k} - 2x_i}{32} \tag{2.44a}$$

$$b = \frac{x_{i+k} - 16a - x_i}{4} \tag{2.44b}$$

$$c = x_i \tag{2.44c}$$

$$d = \frac{y_{i+k} + y_{i-k} - 2y_i}{32} \tag{2.44d}$$

$$e = \frac{y_{i+k} - 16d - y_i}{4} \tag{2.44e}$$

$$f = y_i \tag{2.44f}$$

Parabolün P_i noktasındaki teğet denklemi [64]

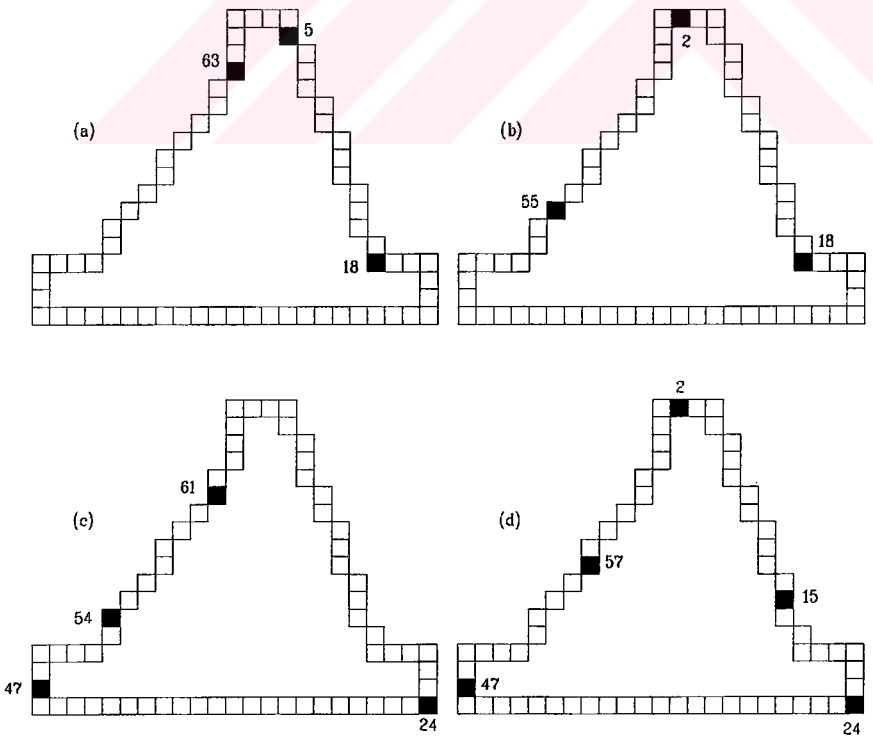
$$y = \frac{e}{b}x - \frac{e}{b}x_i + y_i \tag{2.45}$$

P_i noktasının köşe olabilmesi için eğriliğin bir göstergesi olan

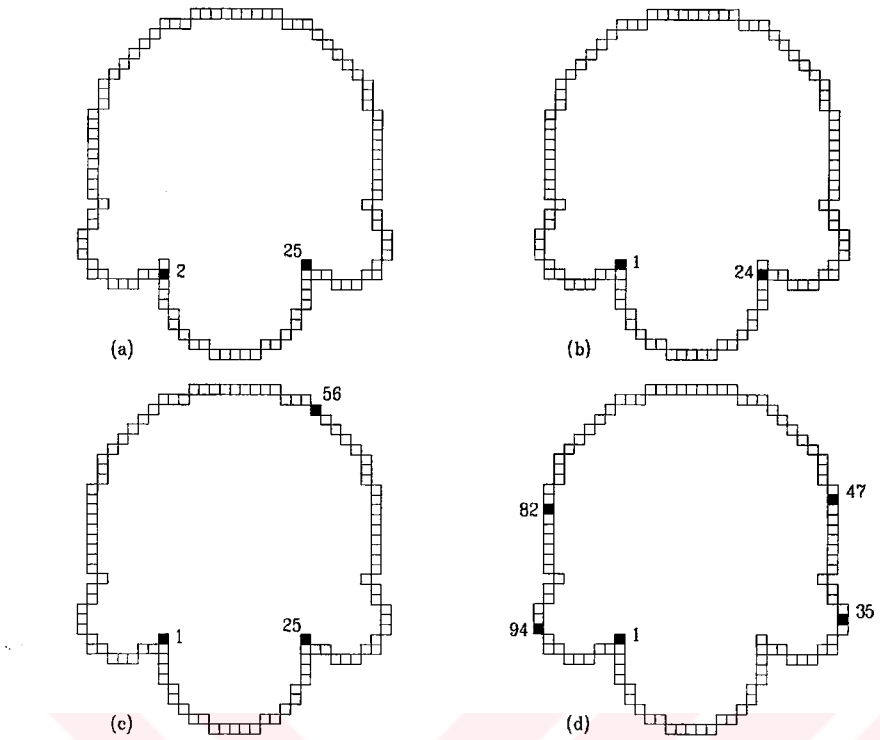
$$c_i = \alpha_i - \alpha_{i-1} \tag{2.46}$$

bir eşik değerinden büyük ve belirlenen bir komşulukta maksimum olmalı. Dolayısıyla (2.46) denklemindeki α

$$\alpha = \operatorname{atan}\frac{e}{b} \tag{2.47}$$



Şekil 2.21 Spthr yönteminin uygulaması. (a) k=2 (b) k=3 (c) k=4 (d) k=5

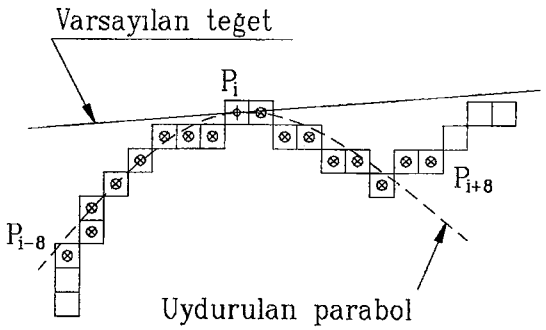


Şekil 2.22 Spthr yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=7

Şekil 2.21 ve Şekil 2.22, "spthr.c" programının sonuçlarını göstermektedir. Ek A' da diğer sonuçlar bulunabilir. Bu yöntemde de köşelerin açık olduğu durumlarda bile genellikle gerçek köşelerden sapmalar ortaya çıkmaktadır. Bu da bizi yeni köşe bulma yöntemleri araştırmasına sevketmektedir.

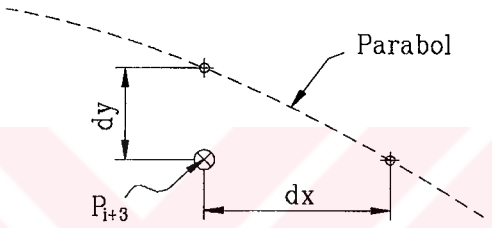
2.2.6 Altıncı Yöntem (Spmin)

Bu yöntemde P_i noktasındaki teğeti bulmak için P_i noktasından (2.43) denklemlerinde belirtilen formda bir parabol geçtiği varsayılır. P_i noktasının önündeki k adet ve arkasındaki k adet nokta parabole minimum uzaklıkta olacak şekilde parabol katsayıları hesaplanacaktır.



Şekil 2.23 P_i **noktasında farz edilen teğet**

Şekil 2.23' te görüldüğü gibi daha sonra parabolün P_i noktasındaki teğeti elde edilir. Örneğin P_{i+3} noktası için hata fonksiyonu şöyle tanımlanabilir (Şekil 2.24).



Şekil 2.24 P_{i+3} **noktasının parabole uzaklığı**

$$E_i = \sqrt{dx^2 + dy^2} = \left\{(x(3) - x_{i+3})^2 + (y(3) - y_{i+3})^2\right\}^{1/2} \tag{2.48}$$

Hata fonksiyonunu P_i noktasının önündeki k adet ve arkasındaki k adet nokta için yazarsak

$$E_i = \sum_{\substack{j=-k \\ j\neq 0}}^{k} \left\{\left(x(j) - x_{i+j}\right)^2 + \left(y(j) - y_{i+j}\right)^2\right\}^{1/2} \tag{2.49}$$

Açık formda hata fonksiyonu

$$
\begin{aligned}
E_i = & \left\{\left(a-b+x_i-x_{i-1}\right)^2+\left(d-e+y_i-y_{i-1}\right)^2\right\}^{1/2} \\
& +\left\{\left(4a-2b+x_i-x_{i-2}\right)^2+\left(4d-2e+y_i-y_{i-2}\right)^2\right\}^{1/2} \\
& +\left\{\left(9a-3b+x_i-x_{i-3}\right)^2+\left(9d-3e+y_i-y_{i-3}\right)^2\right\}^{1/2} \\
& +..... \\
& +\left\{\left(k^2a-kb+x_i-x_{i-k}\right)^2+\left(k^2d-ke+y_i-y_{i-k}\right)^2\right\}^{1/2} \\
& +\left\{\left(a+b+x_i-x_{i+1}\right)^2+\left(d+e+y_i-y_{i+1}\right)^2\right\}^{1/2} \\
& +\left\{\left(4a+2b+x_i-x_{i+2}\right)^2+\left(4d+2e+y_i-y_{i+2}\right)^2\right\}^{1/2} \\
& +\left\{\left(9a+3b+x_i-x_{i+3}\right)^2+\left(9d+3e+y_i-y_{i+3}\right)^2\right\}^{1/2} \\
& +..... \\
& +\left\{\left(k^2a+kb+x_i-x_{i+k}\right)^2+\left(k^2d+ke+y_i-y_{i+k}\right)^2\right\}^{1/2}
\end{aligned}
\qquad (2.50)
$$

Hata fonksiyonunun minimum olabilmesi için

$$
\frac{\partial E_i}{\partial a}=0\ ,\quad \frac{\partial E_i}{\partial b}=0\ ,\quad \frac{\partial E_i}{\partial d}=0\ ,\quad \frac{\partial E_i}{\partial e}=0 \qquad (2.51)
$$

olmalı. Bu türevlerin sonucunda dört denklem ve dört bilinmeyen ortaya çıkar. Fakat bu denklemler oldukça karmaşık ve sadece nümerik yollarla çözülebilirler ve çözümleri oldukça vakit alır. Bu yüzden hata fonksiyonunu (2.49) denklemindeki gibi değil, aşağıdaki formda almak daha uygun olacaktır.

$$
E_i=\sum_{\substack{j=-k\\ j\neq 0}}^{k}\left\{\left(x(j)-x_{i+j}\right)^2+\left(y(j)-y_{i+j}\right)^2\right\} \qquad (2.52)
$$

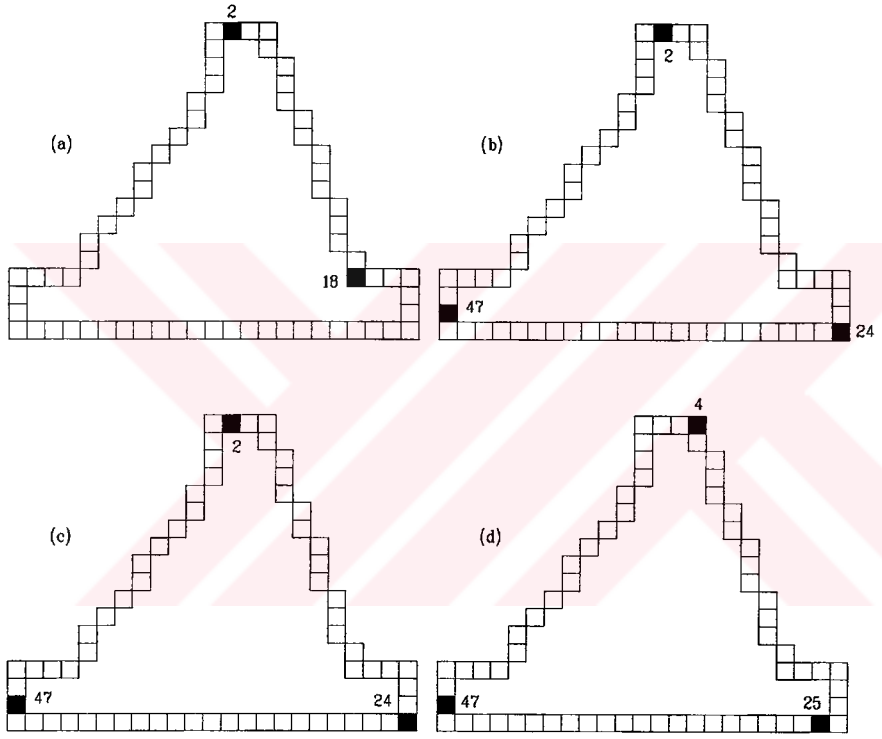
Örneğin $k=5$ alınırsa parabolün katsayılarından b ve e aşağıdaki gibi elde edilir.

$$
b=\frac{1}{110}\left[(x_{i+1}-x_{i-1})+2(x_{i+2}-x_{i-2})+3(x_{i+3}-x_{i-3})+4(x_{i+4}-x_{i-4})+5(x_{i+5}-x_{i-5})\right]
$$

(2.53a)

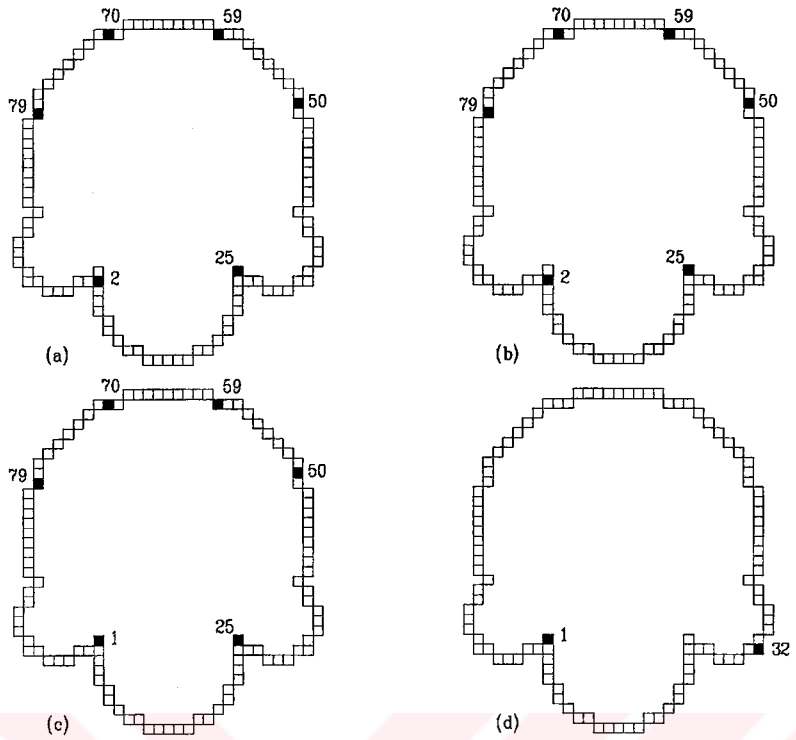
$$e=\frac{1}{110}\left[(y_{i+1}-y_{i-1})+2(y_{i+2}-y_{i-2})+3(y_{i+3}-y_{i-3})+4(y_{i+4}-y_{i-4})+5(y_{i+5}-y_{i-5})\right] \quad (2.53b)$$

(2.45) denkleminden yararlanarak P_i noktasında parabolümüze çizilen teğetin eğimi bulunur. Sonra her nokta için bulunacak c_i (2.46 denkleminden), eğrilik hakkında bilgi taşıyacaktır. Bir noktanın köşe olabilmesi için diğer yöntemlerdeki gibi c_i' nin bir eşik değerinden büyük ve belirtilen bir komşulukta maksimum olması gerekir.



Şekil 2.25 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6

Şekil 25 ve Şekil 26' da bu yöntemi kullanan "spmin.c" programının sonuçları bulunmaktadır. Başka geometrilere uygulama sonuçları Ek A' da verilmiştir. Bu yöntemde de gerçek köşelere yakın benekler köşe olarak bulunabilmesine rağmen istenen başarı sağlanamamıştır. Dolayısıyla yeni köşe bulma yöntemleri geliştirilmeye devam edilmektedir.



Şekil 2.26 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6

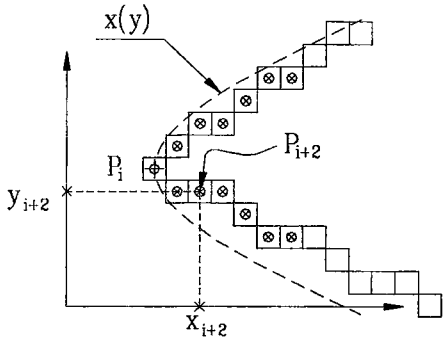
2.2.7 Yedinci Yöntem (Hatmin)

P_i noktasından aşağıdaki iki parabolün geçtiğini varsayalım.

$$y(x) = ax^2 + bx + c \qquad (2.54a)$$

$$x(y) = dy^2 + ey + f \qquad (2.54b)$$

Bu iki parabol P_i noktasının k adet önündeki ve k adet arkasındaki noktalara en yakın yerden geçerler.



Şekil 2.27 Sayısal eğriye uydurulan parabol

Şekil 2.27' de görüldüğü gibi P_i noktasından (2.54b) denklemindeki parabolün geçmesi daha uygun. P_i noktası için $y(x)$ veya $x(y)$ parabolünün kullanımına hata fonksiyonu karar verir. Örneğin Şekil 2.27' deki P_{i+2} noktası için x yönündeki hata

$$E_x = (x(y_{i+2}) - x_{i+2})^2 \tag{2.55}$$

Bu denklemi genelleyerek aşağıdaki hata fonksiyonuna ulaşılır.

$$E_x = \sum_{j=-k}^{k} (x(y_{i+j}) - x_{i+j})^2 \tag{2.56}$$

y yönündeki hata fonksiyonu da aynı şekilde

$$E_y = \sum_{j=-k}^{k} (y(x_{i+j}) - y_{i+j})^2 \tag{2.57}$$

Amacımız E_x'i minimum yapan *d*, *e* ve *f* 'yi bulmaktadır. Aynı zamanda

$$x_i = dy_i^2 + ey_i + f \tag{2.58}$$

şartı da sağlanmalıdır. Dolayısıyla yeni fonksiyonumuz [65]

$$F = E_x - \lambda\,(dy_i^2 + ey_i + f - x_i) \tag{2.59}$$

d, *e* ve *f* katsayılarının belirlenmesi için

$$\frac{\partial F}{\partial d}=\frac{\partial F}{\partial e}=\frac{\partial F}{\partial f}=\frac{\partial F}{\partial \lambda}=0 \qquad (2.60)$$

olmalı. Bu türevler alınıp çıkan denklemler düzenlendiğinde aşağıdaki takım oluşur.

$$\begin{pmatrix} 2\sum_{j=-k}^{k} y_{i+j}^4 & 2\sum_{j=-k}^{k} y_{i+j}^3 & 2\sum_{j=-k}^{k} y_{i+j}^2 & -y_i^2 \\ 2\sum_{j=-k}^{k} y_{i+j}^3 & 2\sum_{j=-k}^{k} y_{i+j}^2 & 2\sum_{j=-k}^{k} y_{i+j} & -y_i \\ 2\sum_{j=-k}^{k} y_{i+j}^2 & 2\sum_{j=-k}^{k} y_{i+j} & 2\sum_{j=-k}^{k} 1 & -1 \\ -y_i^2 & -y_i & -1 & 0 \end{pmatrix} \begin{pmatrix} d \\ e \\ f \\ \lambda \end{pmatrix} = \begin{pmatrix} 2\sum_{j=-k}^{k} x_{i+j} y_{i+j}^2 \\ 2\sum_{j=-k}^{k} x_{i+j} y_{i+j} \\ 2\sum_{j=-k}^{k} x_{i+j} \\ -x_i \end{pmatrix} \qquad (2.61)$$

Anolojiden yararlanarak y doğrultusundaki hata fonksiyonu ile ilgili olarak

$$G = E_y - \mu\ (ax_i^2 + bx_i + c - y_i) \qquad (2.62)$$

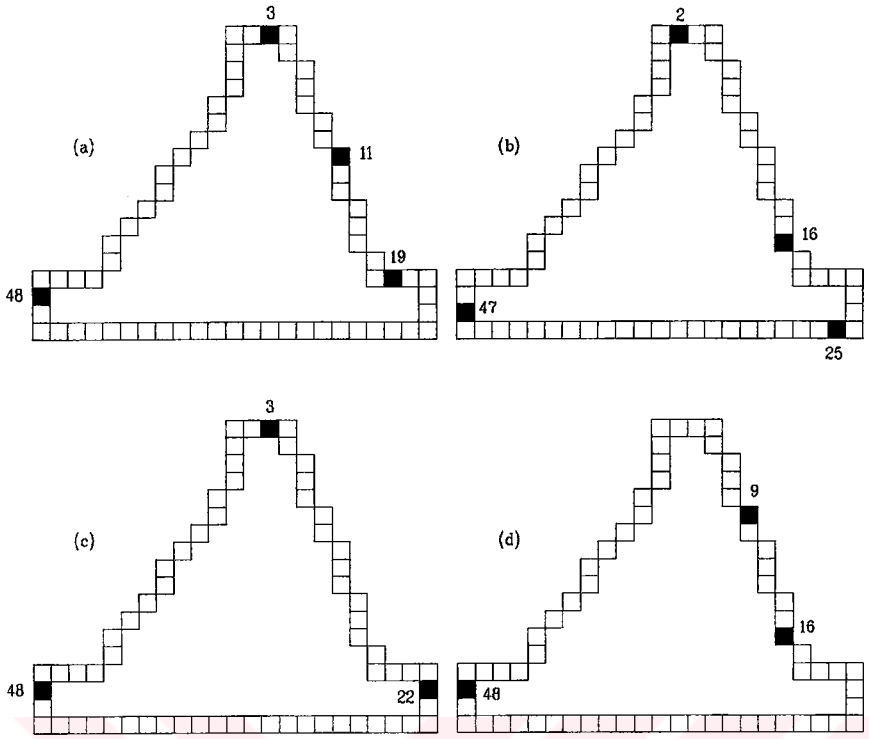
yazılabilir. Aynı şekilde *a*, *b*, *c* ve μ katsayılarını bulmak için

$$\frac{\partial G}{\partial a}=\frac{\partial G}{\partial b}=\frac{\partial G}{\partial c}=\frac{\partial G}{\partial \mu}=0 \qquad (2.63)$$

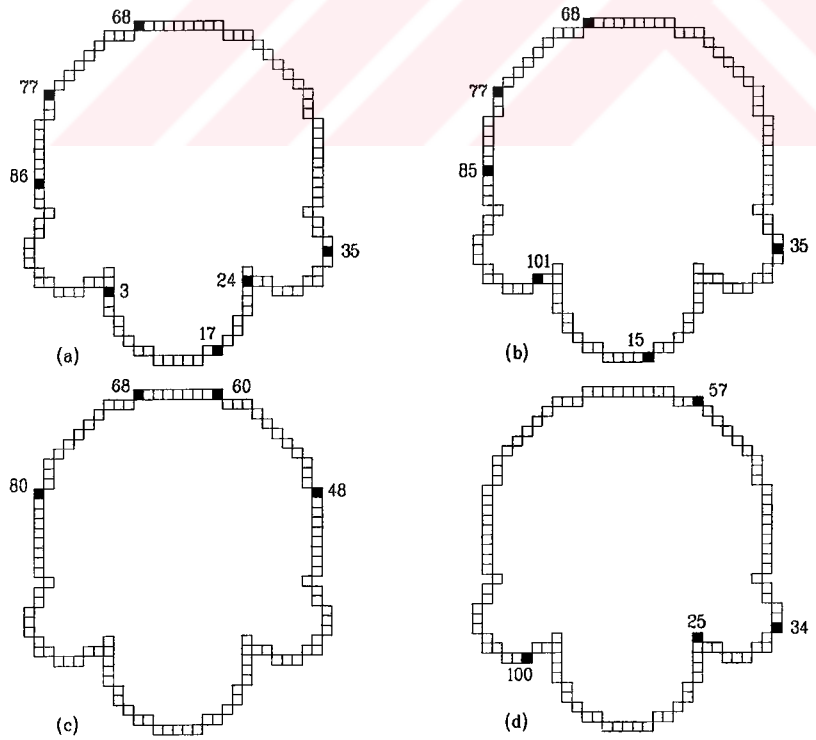
türevleri alınıp çıkan denklemler düzenlendiğinde aşağıdaki takım elde edilir.

$$\begin{pmatrix} 2\sum_{j=-k}^{k} x_{i+j}^4 & 2\sum_{j=-k}^{k} x_{i+j}^3 & 2\sum_{j=-k}^{k} x_{i+j}^2 & -x_i^2 \\ 2\sum_{j=-k}^{k} x_{i+j}^3 & 2\sum_{j=-k}^{k} x_{i+j}^2 & 2\sum_{j=-k}^{k} x_{i+j} & -x_i \\ 2\sum_{j=-k}^{k} x_{i+j}^2 & 2\sum_{j=-k}^{k} x_{i+j} & 2\sum_{j=-k}^{k} 1 & -1 \\ -x_i^2 & -x_i & -1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ \mu \end{pmatrix} = \begin{pmatrix} 2\sum_{j=-k}^{k} y_{i+j} x_{i+j}^2 \\ 2\sum_{j=-k}^{k} y_{i+j} x_{i+j} \\ 2\sum_{j=-k}^{k} y_{i+j} \\ -y_i \end{pmatrix} \qquad (2.64)$$

Bu denklem takımından (2.54a) parabolünün katsayıları bulunur. (2.54) parabolleri belirlendikten sonra E_x ve E_y hata fonksiyonları hesaplanır.



Şekil 2.28 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



Şekil 2.29 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6

P_i noktasındaki eğrilik için de aşağıdaki yöntem kullanılır.

$$c_i = \frac{2d}{(1+(2dy_i+e)^2)^{3/2}} \quad , \quad E_x < E_y \text{ ise} \qquad (2.65a)$$

$$c_i = \frac{2a}{(1+(2ax_i+b)^2)^{3/2}} \quad , \quad E_x > E_y \text{ ise} \qquad (2.65b)$$

Her nokta için (2.65) denklemleri yardımıyla hesaplanan eğrilik değerleri arasından belirtilen komşulukta en büyük olanlar köşe noktası olarak atanır. Şekil 2.28 ve Şekil 2.29' da "hatmin.c" programının sonuçları görülmektedir. Diğer uygulama sonuçları Ek A' da verilmiştir. Görüldüğü gibi bu yöntemin sonuçları da yeterince hassas değil.

Bu bölümde köşe bulma üzerindeki çalışmalarımız anlatıldı. Kaynak taramalarımızdan çıkardığımız üç köşe bulma yöntemi kısaca özetlendi. Köşe bulmaya yönelik bizim çalışmalarımızdaki başarılı yöntemler etkinlik sıralamasıyla sunuldu. Bundan sonraki üçüncü bölümde çalışmamızın devamı niteliğinde etkin bir vektörleştirme yaklaşımı bulunmaktadır.

3. BÖLÜM

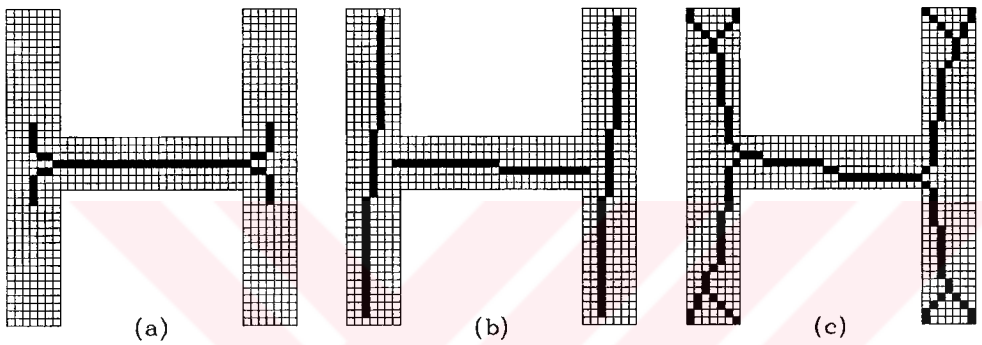
ETKİN BİR VEKTÖRLEŞTİRME YÖNTEMİ

Beneklerden oluşan noktasal biçimdeki bir şeklin, doğru, yay, çember veya elips gibi temel geometrik nesnelere dönüştürülmesi işlemine vektörleştirme denir [3]. Çok değişik nedenlerden dolayı vektörleştirme işlemine ihtiyaç duyulur. Diğer bir deyişle uygulama alanı oldukça geniş bir çalışma konusudur ve henüz istenen düzeyde başarı yakalanabilmiş değildir.

Vektörleştirme yaklaşımlarının uygulama alanlarının çok geniş bir yelpazeyi kapsar. Kâğıt üzerindeki bir çizimin elektronik ortama aktarılmasında [3-9,13,14], karmaşık parçaların bir takım testleri yapılırken [15], coğrafi bilgi sistemlerinde [17], bir fotoğraftaki cisimlerin tanınmasında veya konveyörde taşınan ürünlerin tanınmasında vektörleştirme algoritmalarından yararlanılmaktadır. Yani askeri amaçlı uygulamalardan, kalite-kontrol uygulamalarına; endüstriyel uygulamalardan, otomatik pilot uygulamalarına kadar bir çok konuda vektörleştirme işlemi devreye sokulabilir. Ayrıca vektörleştirme sonucu oluşan vektör tabanlı çizimin büyüklüğü ciddi oranda azalır.

Bu bölümde özellikle mühendislik çizimlerine uygulanabilecek (kalın çizgilerden oluştuğu için) etkin bir vektörleştirme yaklaşımı sunulmaktadır. Sunulan vektörleştirme yaklaşımı henüz geliştirilme aşamasında olup sadece doğrulardan oluşan mühendislik veya mimari tarzındaki çizimlere uygulanabilmektedir. Tabii ki bir teknik resim çiziminde doğrulardan başka geometriler olduğu gibi bazı özel semboller, ölçülendirme öğeleri ve yazılar içerilir. Vektörleştirme uygulamalarında ilk önce yazıların [3,8,9-12], sembollerin [3,7,9,14] ve ölçülendirme öğelerinin [13] ayrıştırılması işlemi yer alır. Dolayısıyla resim düzleminde sadece çizim öğeleri (doğru, çember, yay gibi) bırakılır. Bu çizim öğeleri de bir benek kalınlığında değildir ve çoğunlukla inceltme (thinning) işleminden geçirilirler. İnceltme işlemi sonucunda

noktasal formattaki çizim öğeleri artık bir benek kalınlığına getirilmiştir. Bu durumdaki çizime Şekil 2.1 de kabaca gösterilen akış şemasındaki adımlar tatbik edilir. Genellikle kâğıt üzerindeki bir çizime uygulanan klasik vektörleştirme yaklaşımı bu çerçevede ortaya çıkar. Bu çalışmada geliştirilen yaklaşımın, klasik yaklaşımlara karşı en temel üstünlüğü, vektörleştirme işlemindeki inceltme adımının çıkarılmış olmasıdır. Bu da, son bölümde de bahsedilen inceltme işleminden kaynaklanabilecek problemlerin [66-73] yok edilmesini beraberinde getirir. Dolayısıyla ciddi bir zaman tassarrufu söz konusu olmakla beraber Şekil 3.1' de özetlenen inceltme adımından kaynaklanabilecek hatalardan da korunulmuş olur.

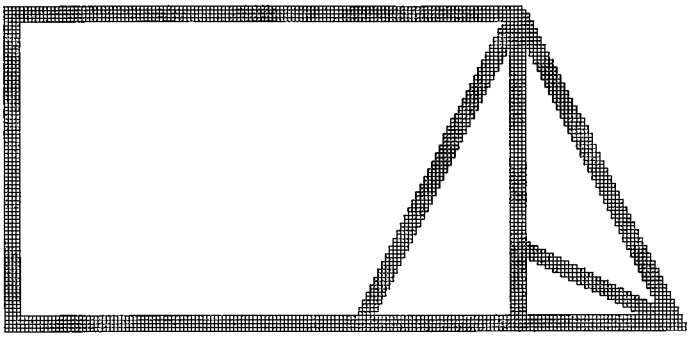


Şekil 3.1 İnceltme kaynaklı olabilecek hatalar. (a) Uçların yok edilişi. (b) Kopukluk oluşumu. (c) Erozyon oluşumu.

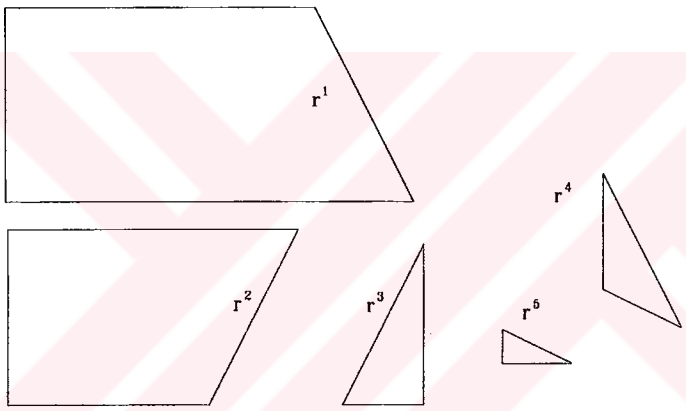
Bu bölümde sunulan yeni vektörleştirme yaklaşımımız beş temel basamak içerir. Bu beş basamak: (1) Sınır eğrilerin elde edilmesi; (2) birinci adımda bulunan sınır eğrilerin köşelerinin tesbiti; (3) komşulukların oluşturulması; (4) karşı karşıya komşu olma niteliklerinin yok edilmesi; (5) uç noktaların birleştirilmesi. İkinci adımda LSQ köşe bulma yöntemi kullanılmıştır. Köşelerin bulunmasıyla beraber şekli oluşturan sınır eğriler de elemanlarına ayrılmış olurlar. Dolayısıyla her bir elemana artık bireysel olarak müdahale yapılabilir. Böylece üçüncü adıma gönderilen parçacıklara komşuluk etiketleri verilir. Komşuluklar karşı karşıya (KKK) ve uç uca (UUK) olmak üzere iki türlüdür. Dördüncü adımda KKK etiketine sahip parçalar yok edilir ve yerlerine bunların orta hatları çizilir. Çizilen bu orta hatlar, vektör formatındadır. Yani dördüncü adımın sonunda çizim vektör formatına dönüşür. Fakat bazı uç noktalar problemin doğasından dolayı üst üste olmayabilir. Böylece bir son adıma (beşinci adım) gereksinim vardır. Beşinci adımda da bu açıklıklar giderilir.

3.1 SINIR EĞRİLERİN ÇIKARILMASI

Sunulan yeni vektörleştirme yaklaşımındaki ilk adım sınır eğrilerin elde edilmesidir. Örneğin Şekil 3.2.a' da görülen geometrinin sınır eğrileri Şekil 3.2.b' de verilmiştir.



(a)



(b)

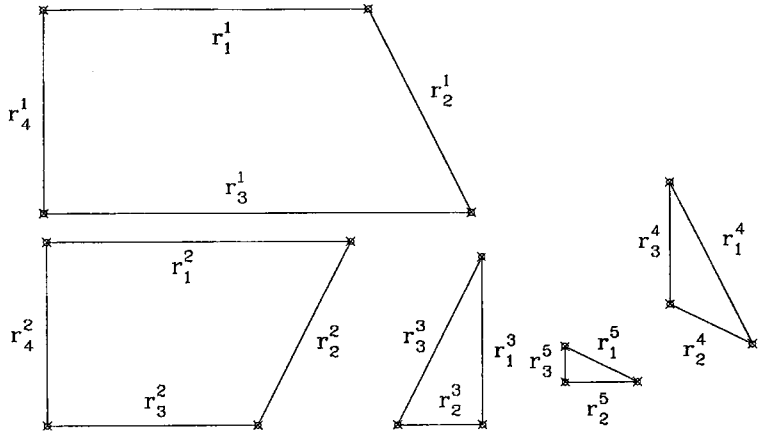
Şekil 3.2 Sınır eğrilerinin çıkarılışı (a) Sayısal biçimde bir resim (b)Çıkarılan sınır eğrileri

Her zaman "1" indisli eğri, dış sınır eğrisi olup diğerleri iç yapıyı oluşturan sınır eğrileridir.

3.2 SINIR EĞRİLERDEKİ KÖŞELERİN TESBİTİ

Algoritmamızın ikinci adımını sınır eğrilerin parçalarına ayrılması oluşturmaktadır. Parçalara ayırma işlemi, eğrinin köşelerinin tesbitini gerektirir. Köşe tesbiti için de ikinci bölümde geliştirilen metodlardan en iyi sonuçları veren "LSQ" metodu

kullanılmıştır. Şekil 3.3, çıkarılan sınır eğrilerde bulunan köşeleri ve parçaları göstermektedir. Parçalara indis verme şekli U. Cugini' den [5] alınmıştır.



Şekil 3.3 Sınır eğrilerin köşe ve parçaları

3.3 KOMŞULUKLARIN OLUŞTURULMASI

Artık sınır eğrilerinin parçaları elimizde. Şimdi bu parçaların hangilerinin komşu olduklarını bulmalıyız. Bunun için aşağıdaki algoritma kullanılır.

Adım 1: Dış sınır eğrisinin parçalarından birini al (r_i^1) ve eğimini hesapla (m_i^1). Örneğin r_1^1 i ele aldığımızı varsayalım.

Adım 2: Eğimi, birinci adımda bulunan eğime eşit olan sınır eğrilerinin parçalarını bul. Örneğin r_1^1 parçasının eğimine eşit eğimi olan parçalar r_1^2, r_3^2, r_2^3, r_2^5 dir.

Adım 3: Eşit eğimli parça sayısı 1' den fazla ise uzaklık testi yapılır. Yani birinci adımdaki parçaya, ikinci adımda bulunan parçaların ne kadar uzakta olduğuna bakılır. Önceden belirtilen bir mesafeden daha yakın olanlar dikkate alınırlar. Örneğin ikinci adımda bulunan parçalardan sadece r_1^2, uzaklık testini gerçekler.

Adım 4 : Uzaklık testinden geçen parça sayıs 1' den fazla ise uç noktaların izüşümlerinin (UNİ) testi yapılır. UNİ testinin adımları şöyledir (L1 ve L2

gibi iki doğrunun ele alındığı düşünülürse):

(1) L1 doğrusunun uç noktaları, L2 doğrusu üzerine izdüşürülür.

(2) İzdüşüm noktaları, L2 doğrusunun uç noktaları arasında mı?

(3) Arasında değilse izdüşüm noktasının veya noktalarının, L2 doğrusunun uç noktalarına mesafesi δ' dan küçük mü?

(4) Arasında ise L1 ve L2 doğrularına karşı karşıya komşu (KKK) denir ve yedinci adıma geçilir.

(5) Küçük değilse L2 ve L1 doğruları yer değiştirilir ve yukarıdaki adımlar tekrarlanır.

(6) Küçükse L1 ve L2 doğrularına karşı karşıya komşu (KKK) denir.

(7) Başka L1 ve L2 doğrularını al ve birinci adıma git.

Şekil 3.1' deki geometride sınır eğrilerinin parçaları, Şekil 3.3 te gösterilmiştir. Buradaki doğrulara bahsedilen algoritma uygulandığında karşı karşıya komşu (KKK) olma listesi aşağıdaki gibi çıkarılmış olacaktır:

r_1^1 : r_1^2

r_2^1 : r_1^4

r_3^1 : r_3^2, r_2^3, r_2^5

r_4^1 : r_4^2

r_2^2 : r_3^3

r_1^3 : r_3^4, r_3^5

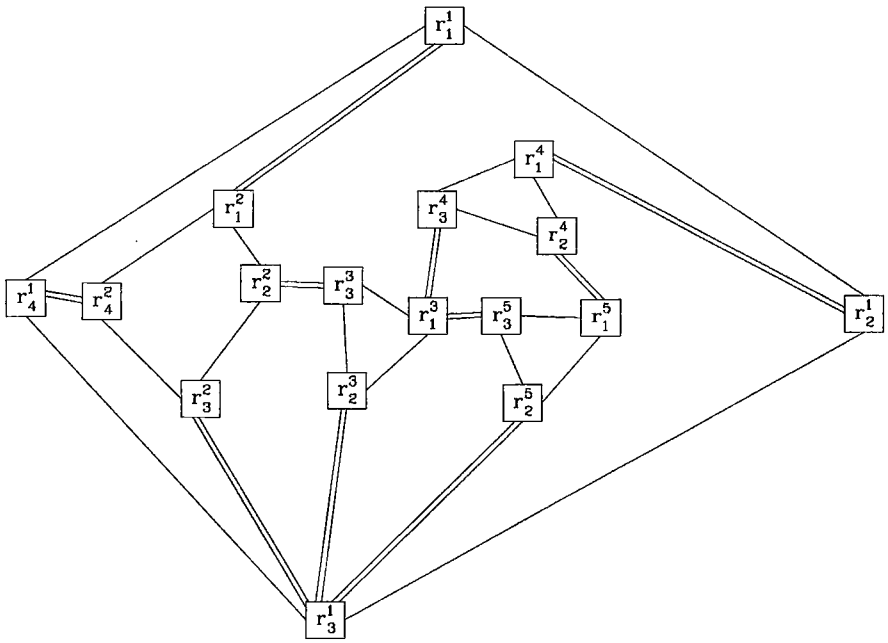
r_2^4 : r_1^5

Ayrıca bir de uç uca komşu (UUK) olma listesinden bahsedilebilir. Aynı örnek için UUK listesi aşağıdaki gibidir.

$r_1^1 - r_2^1 - r_3^1 - r_4^1$ $\quad r_1^2 - r_2^2 - r_3^2 - r_4^2$ $\quad r_1^3 - r_2^3 - r_3^3$

$r_1^4 - r_2^4 - r_3^4$ $\quad r_1^5 - r_2^5 - r_3^5$

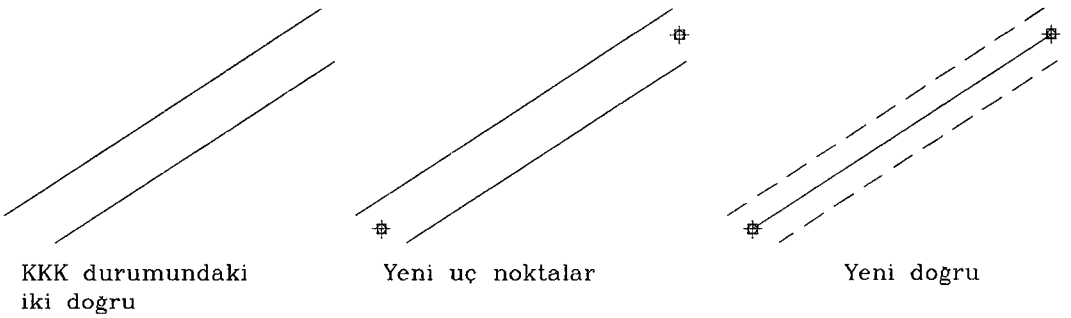
KKK ve UUK listelerinden hareketle şekil 3.4' te görülen grafik elde edilebilir.



Şekil 3.4 Komşuluk durum grafiği (-, UUK ; =, KKK)

3.4 KKK HALİNİN BİTİRİLMESİ

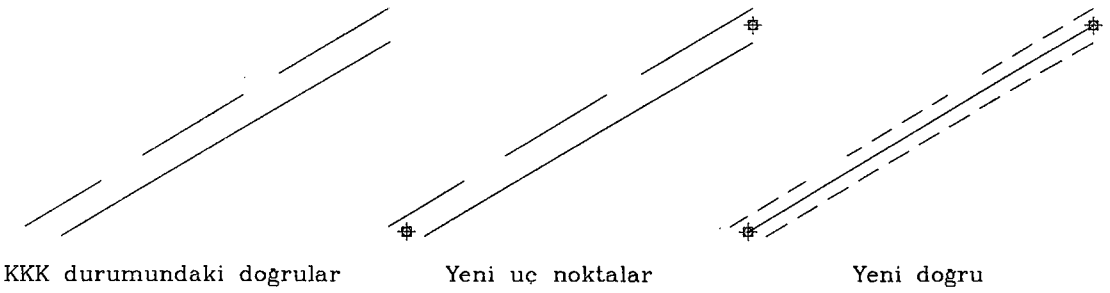
Algoritmanın bu basamağında karşı karşıya komşu olan doğrulardan tek doğru üretilecektir. Örneğin (r_1^1 : r_1^2 veya r_2^1 : r_1^4 deki gibi) sadece iki doğru KKK durumunda ise birbirine yakın uçların aritmetik ortalaması alınarak yeni uç noktalar bulunur. Dolayısıyla yeni doğruya ulaşılmıştır. Şekil 3.5' te bu durum özetlenmeye çalışılmıştır.



Şekil 3.5 Yeni doğrunun oluşumu (2 doğru KKK)

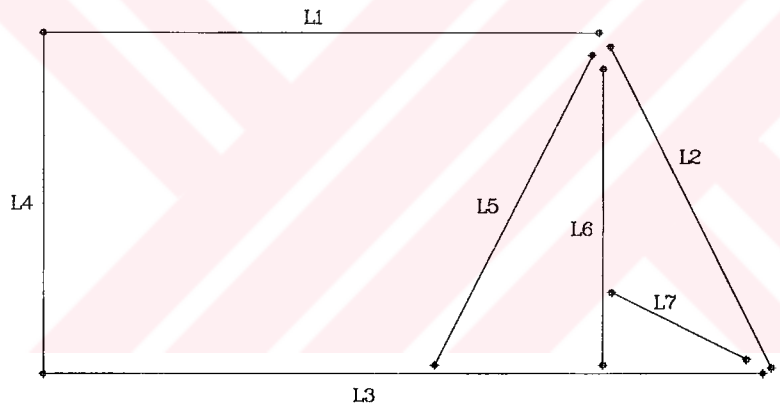
İkiden fazla doğru KKK durumunda (r_3^1 : r_3^2, r_2^3 : r_2^5 veya r_1^3 : r_3^4, r_3^5 deki gibi) ise işlem sayısı artmakla beraber, yukarıdakine benzer bir yol izlenir. İlk olarak uzun olan doğrunun uçlarına, kısa olan doğrulardan hangilerinin uçlarının en yakın olduğu bulunur. Elde edilen iki çift noktanın her bir çifti, aritmetik ortalamaya tabi

tutularak yeni çizilecek doğrunun uç noktaları bulunur. Şekil 3.6, bu durumu özetlemektedir.

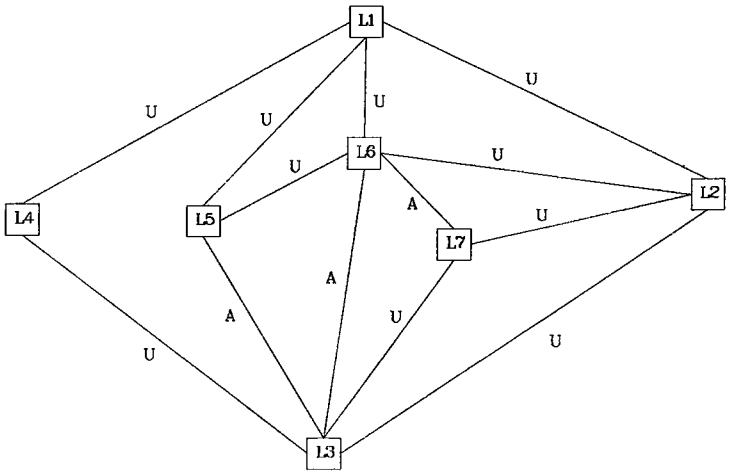


Şekil 3.6 Yeni doğrunun oluşumu (2' den fazla doğru KKK)

Şekil 3.1' deki geometri için elde edilen KKK durumundaki doğrulardan yeni doğrulara ulaşıldığında Şekil 3.7' deki geometri oluşur.



Şekil 3.7 KKK doğruların yok edilmiş hali



Şekil 3.8 Komşuluk durum grafiği (KKK yok)

Şekil 3.7' deki geometriye ilişkin grafik yukarıda Şekil 3.8' de gösterilmiştir. Şekil 3.8' deki grafik elde edilirken doğrular arasında aşağıdaki ilişkiden yararlanıldı. Aşağıdaki listede bazı doğruların U, bazılarının ise A ilişkisi ile birbirine bağlı olduğu görülmektedir. U, uç uca bağlılığı; A ise aradan bağlılığı ifade eder. Doğrular arasında U veya A bağlılığının olduğu ise aşağıdaki algoritma vasıtasıyla tayin edilir:

Adım 1: İlk doğruyu al. Örneğin L1.
Adım 2 : Birinci adımda ele alınan doğru(lar)dan farklı başka bir doğruyu al (L2).
Adım 3 : İkinci adımda ele alınan doğrunun (L2 doğrusunun) uçlarından biri, birinci adımdaki doğrunun (L1 doğrusunun) uçları ile üst üste mi? Evet ise bu iki doğru arasında U bağlılığı vardır denir ve son adıma geçilir.
Adım 4 : Adım 2 deki doğrunun uçlarından biri adım 1 deki doğrunun uçlarından birine yeterince (seçilen bir ε mesafesinden küçük) yakın mı? Yeterince yakınsa bu iki doğru arasında U bağlılığı vardır denir ve son adıma geçilir.
Adım 5 : Adım 2 deki doğrunun uçlarından biri adım 1 deki doğruya yeterince (seçilen birε mesafesinden küçük) yakın mı? Yeterince yakınsa bu iki doğru arasında A bağlılığı var denir ve son adıma geçilir.
Adım 6 : İkinci ele alınan doğruyu değiştir ve ikinci adıma git.

L1 : L2, L4, L5, L6 (U)
L2 : L3, L7 (U)
L3 : L4 (U)
L3 : L5, L6 (A)
L6 : L7 (A)

3.5 UÇ NOKTALARIN BİRLEŞTİRİLMESİ

Şekil 3.7' de görüldüğü gibi bazı doğruların uçları mecburen açıkta kalmıştır. Bu aşamada bu açıklıklar giderilmeye çalışılacaktır. Bu açıklıkların bertaraf edilebilmesi için Şekil 3.8' deki grafikten veya aynı şeyi ifade eden yukarıdaki listeden yararlanılmaktadır. Açıklıklar giderilirken dışarıdan içeriye doğrular dikkate alınmıştır. Doğruların uç noktaları başlangıçta H (hareket edebilir) tipindedir. H tipine sahip uç noktası olan doğru kalmayıncaya kadar (açıklıklar giderilinceye

kadar) geliştirilen algoritma sürdürülür. Uç noktaları birleştirmek için geliştirilen algoritmanın adımları aşağıdadır.

Adım 1 : Aralarında U bağlantısı olan iki doğruyu al. Örneğin L1 ve L2.

Adım 2 : Bu iki doğrunun üst üste uç noktası var mı? Varsa üst üste olan noktaları H tipi olmaktan çıkar; bu iki doğru arasındaki bağlantıyı kopar; adım 4 e git.

Adım 3 : Bu iki doğrunun kesişim noktasını bul. Doğruların kesişim noktasına yakın olan uç noktalarını, bulunan kesişim noktası yap; bu uç noktaları H tipi olmaktan çıkar; bu iki doğru arasındaki bağlantıyı kopar; adım 4 e git.

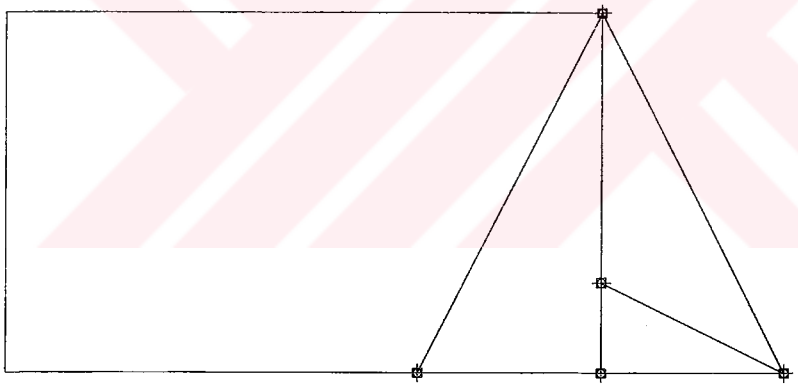
Adım 4 : Aralarında U bağlantısı olan başka iki doğru al ve adım 2 ye git.

Adım 5 : Aralarında A bağlantısı olan iki doğruyu al. Örneğin L3 ve L5 doğruları.

Adım 6 : Bu iki doğrunun kesişim noktasını bul. Kesişim noktasına yakın uç noktasını değiştir ve bu ucu H tipi olmaktan çıkar; bu iki doğru arasındaki bağlantıyı kopar; adım 7 ye git.

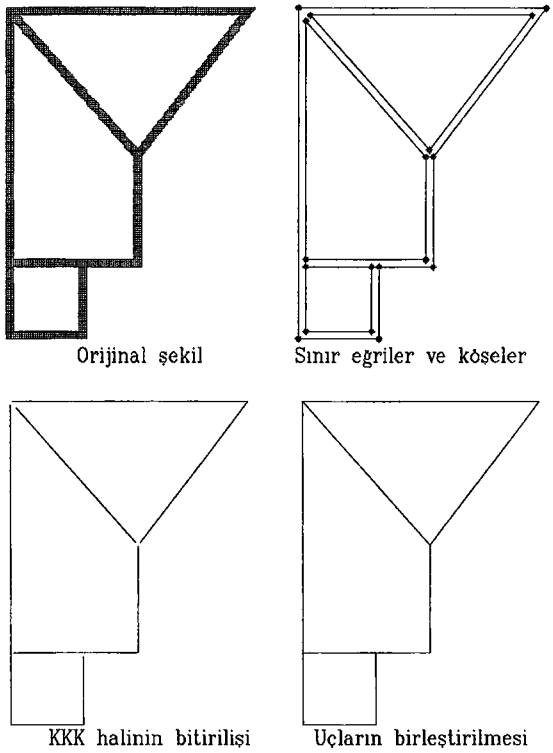
Adım 7 : Aralarında A bağlantısı olan başka iki doğru al ve adım 6 ya git.

Şekil 3.9' da örneğimizdeki uç noktaların birleştirilmiş durumu gösterilmiştir.

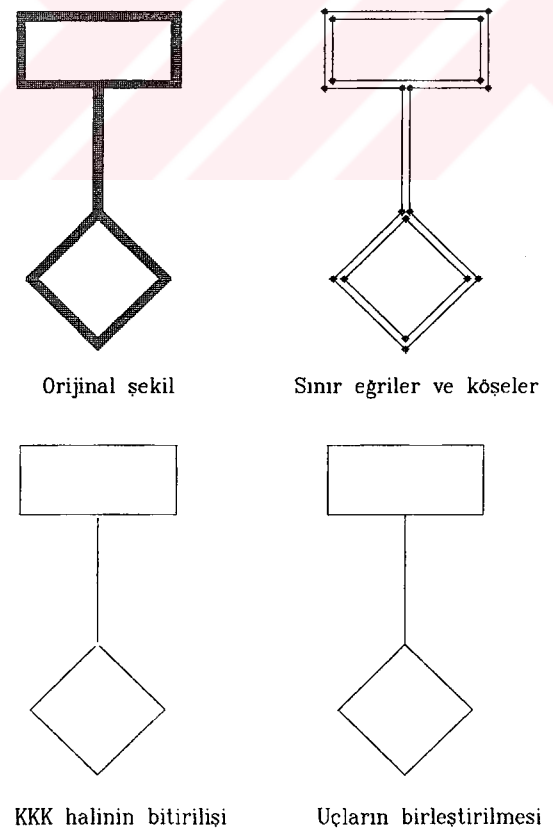


Şekil 3.9 Uçların birleştirilmesi

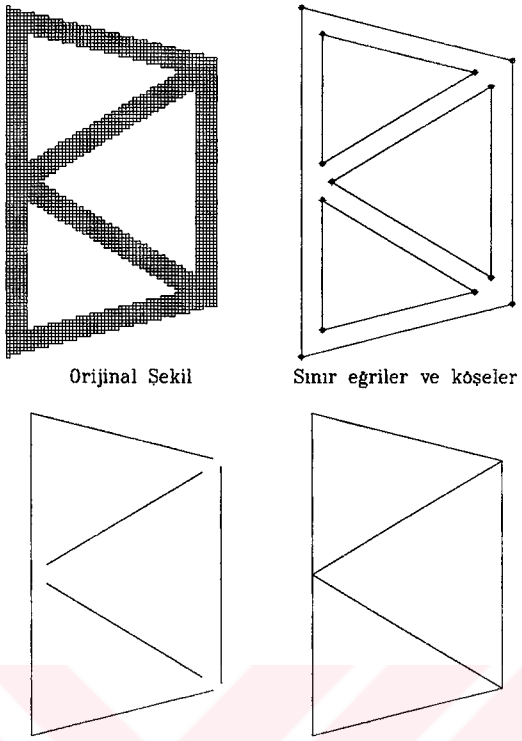
Aşağıdaki Şekil 3.10, Şekil 3.11 ve Şekil 3.12 bu yöntemin üç farklı geometri üzerine uygulama sonuçlarını göstermektedir.



Şekil 3.10 Doğrulardan oluşan bir geometriye geliştirilen algoritmanın uygulanışı.



Şekil 3.11 Algoritmanın farklı bir geometriye uygulanışı.



Şekil 3.12 Algoritmanın farklı bir geometriye uygulanışı.

4. BÖLÜM

SONUÇLAR VE ÖNERİLER

Bu çalışmamızın ilk aşamasında köşe bulma, devamın da vektörleştirme konusu üzerinde çalışılmıştır. Vektörleştirme işleminin ve sayısal eğrilerin köşelerinin bulunmasının önemi vurgulanmış ve konuyla ilgili yeni yaklaşımlarımız sunulmuştur.

Sayısal eğrilerin köşelerinin bulunmasına yönelik olarak referans seçtiğimiz üç yöntem (Rosenfels-Johnston, Rosenfeld-Weszka ve Medioni-Yasumoto) ve geliştirdiğimiz yöntemler arasından başarılı bulunulanlar (Lsq, Sp3min, Bartem, Moment, Spthr, Spmin ve Hatmin) değişik geometrilere uygulanmış ve sonuçlar elde edilmiştir. Uygulamalardan görülmektedir ki bu yöntemler içinden Lsq köşe bulma yöntemi en iyi sonuçları vermektedir. Dolayısıyla geliştirdiklerimiz içinden Lsq yöntemi haricindekiler bir tarafa bırakılmıştır. Uygulamalardan görülmektedir ki Lsq köşe bulma yönteminin, özellikle köşelerin doğru yerde ve doğru sayıda olma şartını sağlaması dikkate değerdir. Tablo 4.1 ve Tablo 4.2' de sunulan köşe bulma yöntemleri için performans karşılaştırması bulunmaktadır.

Tablo 4.1 Şekil 2.2.a' ya uygulanan yöntemlerin başarı karşılaştırması.

Sınır eğrisi benek sayısı, $n=65$; gerçek köşe sayısı, $n_{k,g}=9$

	Bulunan Köşe Sayısı	Doğru Yerde Olanlar	Doğru Yerde Olanlar (±1)
Lsq	8 (%88.89)	8 (%88.89)	8 (%88.89)
Sp3min	9 (%100.00)	4 (%44.44)	9 (%100.00)
Bartem	6 (%66.67)	2 (%22.22)	5 (%55.56)
Moment	4 (%44.44)	3 (%33.33)	4 (%44.44)
Spthr	5 (%55.56)	1 (%11.11)	3 (%33.33)
Spmin	3 (%33.33)	1 (%11.11)	3 (%33.33)
Hatmin	4 (%44.44)	0 (%0.00)	3 (%33.33)

Tablo 4.2 Şekil 2.2.c' ye uygulanan yöntemlerin başarı karşılaştırması.

Sınır eğrisi benek sayısı, $n=102$; gerçek köşe sayısı, $n_{k,g}=4$

	Bulunan Köşe Sayısı	Doğru Yerde Olanlar	Doğru Yerde Olanlar (±1)
Lsq	6	4 (%100.00)	4 (%100.00)
Sp3min	6	1 (%25.00)	2 (%50.00)
Bartem	5	3 (%75.00)	3 (%75.00)
Moment	6	1 (%25.00)	3 (%75.00)
Spthr	3	2 (%50.00)	2 (%50.00)
Spmin	6	1 (%25.00)	2 (%50.00)
Hatmin	4	1 (%25.00)	1 (%25.00)

Uygulamalardaki sonuçlar ışığında, sunulan yöntemler içinden Lsq yönteminin başarısı, Tablo 4.1 ve Tablo 4.2' de de görülmektedir. Dolayısıyla referans olarak seçilen üç yöntem ile Lsq karşılaştırılmıştır. Ayrıca aynı sebepten dolayı tezin devamındaki vektörleştirme yöntemimizdeki köşe bulma aşamasında Lsq kullanılmıştır.

Aşağıdaki Tablo 4.3 ve Tablo 4.4' de Lsq yöntemimizin referans olarak seçilen üç köşe bulma yöntemiyle karşılaştırması yapılmaktadır. Görüldüğü gibi sonuçlar Lsq yöntemi için oldukça iyi seviyededir.

Tablo 4.3 Şekil A.72' ye uygulanan yöntemler için sonuçlar.

Sınır eğrisi benek sayısı, $n = 456$; gerçek köşe sayısı, $n_{k,g} = 32$

	Bulunan Köşe Sayısı	Doğru Yerde Olanlar	Doğru Yerde Olanlar (±1)
Rosenfeld-Johnston	28 (%87.50)	16 (%50.00)	20 (%62.50)
Rosenfeld-Weszka	28 (%87.50)	24 (%75.00)	26 (%81.25)
Medioni-Yasumoto	30 (%93.75)	26 (%81.25)	28 (%87.50)
LSQ	34 (%93.75)	28 (%87.50)	30 (%93.75)

Tablo 4.4 Şekil 2.2.a' ya uygulanan yöntemler için sonuçlar.

Sınır eğrisi benek sayısı, $n = 65$; gerçek köşe sayısı, $n_{k,g} = 9$

	Bulunan Köşe Sayısı	Doğru Yerde Olanlar	Doğru Yerde Olanlar (±1)
Rosenfeld-Johnston	5 (%55.56)	3 (%33.33	4 (%44.44)
Rosenfeld-Weszka	5 (%55.56)	1 (%11.11)	4 (%44.44)
Medioni-Yasumoto	7 (%77.78)	7 (%77.78)	7 (%77.78)
LSQ	8 (%88.89)	8 (%88.89)	8 (%88.89)

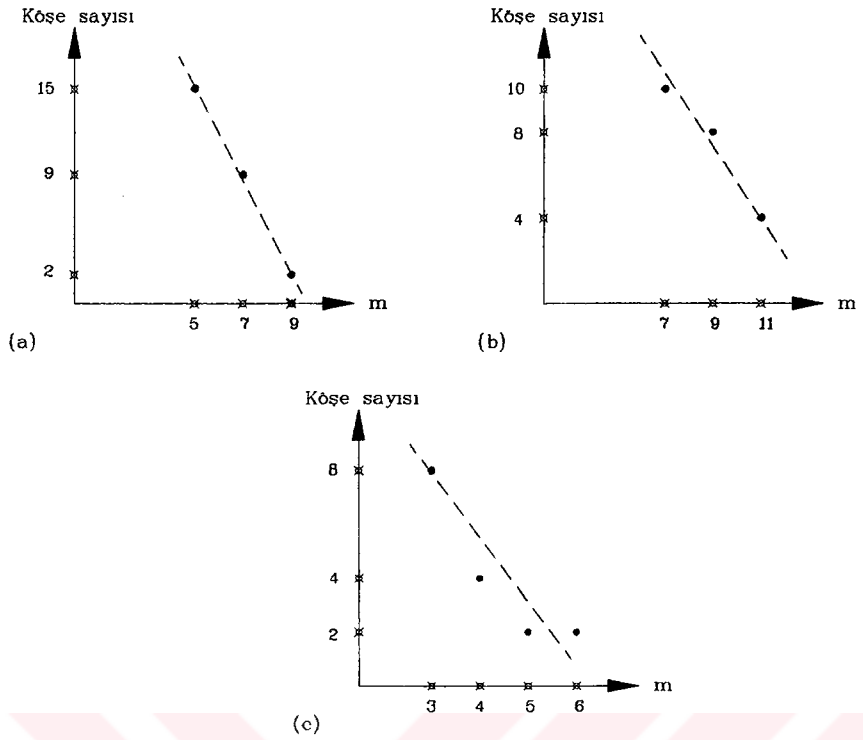
Tablo 4.5' te köşe bulma metodlarının girdi parametreleri gösterilmektedir. *m* ve *k* parametreleri, ilgilenilen beneğin etrafındaki kaç adet beneğin dikkate alınacağını gösterir.

Tablo 4.5 Köşe bulma metodları için girdi parametreleri.

Metod	Girdi Parametreleri
Rosenfeld-Johnston	*m*
Rosenfeld-Weszka	*m*
Medioni-Yasumoto	$\delta_{eşik}, c_{eşik}$
LSQ	*k*

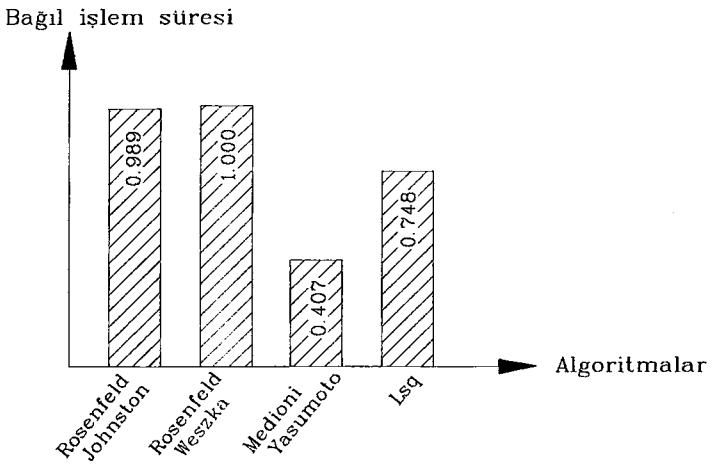
Girdi parametrelerinin az olması yöntem için bir üstünlük olarak dikkate alınabilir. Çünkü fazla sayıda girdi parametresinin seçilmesi, parametrelerin doğru aralıkta tahmin edilmediği taktirde yanlış beneklerin köşe olarak bulunmasına sebep olabilir.

Ayrıca Şekil 4.1' de girdi parametrelerinin, bulunan köşe noktalarını nasıl etkilediği gösterilmiştir. Bu grafikler Şekil A.7, Şekil A.14 ve Şekil A.70 dikkate alınarak çizilmiştir. Çoğunlukla benzer davranışlar diğer uygulamalarda da geçerlidir. Anlaşılacağı gibi ilgilenilen beneğin etrafında dikkate alınan benek sayısındaki artış, yöntemlerin bulduğu köşe sayısında azalmaya sebep oluyor. Çünkü *m* küçükken yani sadece yakındaki benekler işin içine katılıyorken beneklerin yerlerindeki değişimlerin büyük olarak algılanmasından dolayı ve köşeler bulunurken değişimlerin büyük olmasının belirleyici etken olmasından dolayı bulunan köşe noktasında artma gözlenir.



Şekil 4.1 Girdi parametrelerinin köşe noktalarını etkilemesi.
(a) Rosenfeld-Johnston (b) Rosenfeld-Weszka (c) LSQ yöntemleri için

Rosenfeld-Johnston, Rosenfeld-Weszka, Medioni-Yasumoto ve Lsq köşe bulma yöntemleri, Şekil 4.2' de birbirleriyle hız açısından karşılaştırılmaktadır. Bu grafik, yöntemlerin Şekil 2.2a' ya uygulanmasıyla elde edilmiştir. Farklı geometriler için de hemen hemen aynı sonuçlara ulaşılmaktadır.



Şekil 4.2 Yöntemlerin işlem süresi açısından karşılaştırılması.

Sunulan Lsq köşe bulma yöntemine ileriye yönelik olarak aşağıdakiler eklenebilir:

1. Şu an sadece kapalı sayısal eğrilere uygulanabilen yöntem, açık sayısal eğrilere de uygulanabilir duruma getirilebilir.
2. Yöntemdeki girdi parametresi *k*, işlem boyunca değişmemektedir. Oysa bölgesel olarak işlem içinde değiştirilebilir duruma getirilebilir. Bu ekleme birinciye kıyasla daha zor ama yöntemin verimini oldukça artıracağı aşikardır. *k* girdi parametresinin sayısal eğrideki dalgalanmaya bağlanmasıyla bu özelliğin yönteme kazandırılabileceği düşünülmektedir.
3. İki nesnenin birbiriyle keskin köşe oluşturacak şekilde birleşmediği yani sürekliliğin sağlandığı birleşim noktalarının da bulunması sağlanabilir.

Tezdeki çalışmamızın devamı niteliğinde vektörleştirme işlemine yönelik henüz geliştirilme aşamasında olan etkin bir yöntem (üçüncü bölümde) sunulmuştur. Sunulan yöntem şimdilik sadece doğrulardan (kalın çizgiler) oluşan geometrilere tatbik edilebilmektedir. Lsq köşe bulma tekniğini kullanan yöntemimizin klasik vektörleştirme yöntemlerinden en önemli farkı, inceltme adımının kaldırılmış olmasıdır. Dolayısıyla ciddi bir zaman tassarrufu söz konusu olmakla beraber Şekil 3.1' de özetlenen inceltme adımından kaynaklanabilecek hatalardan da korunulmuş olur.

Sunulan vektörleştirme yönteminde ileriye yönelik olarak düşünülenler aşağıdaki gibi özetlenebilir:

1. Sadece doğrulardan oluşan mühendislik veya mimari çizimlere uygulanabilen yöntem, yay, elips, çember gibi eğrileri de kapsayacak şekilde düzenlenebilir.
2. Algoritmamızın çizgi tipleri ve çizgi kalınlıklarını tanıyabilmesi sağlanabilir.

Ayrıca vektörleştirme yöntemimiz aşağıdaki kullanım alanlarına hizmet verebilir:

1. Karakter ve parmakizi analizi uygulamaları içinde kullanılabilir.
2. Otomatik pilot uygulamalarında, örneğin yol çizgilerinin tanınarak hareketin sağlanmasına yönelik olarak kullanılabilir.
3. Askeri ve endüstriyel uygulamalarda, örneğin otomatik hedef tanıma veya robotik çalışmalarına yönelik olarak hazırlanmış yazılımlara eklenebilir.

KAYNAKLAR

[1] **Eastman, C. M.**, 1990. Vector versus Raster: A functional Comparison of Drawing Technologies, IEEE Computer Graphics and Applications, 68-80.

[2] **Bresenham J. E.**, 1987. Ambiguities in Incremental Line versus Rastering, IEEE Computer Graphics and Applications, 31-43.

[3] **Filipski, A. J. and Flandera R.**, 1992. Automated Conversion of Engineering Drawings to CAD Form, Proceedings of the IEEE, **80**, 1195-1209.

[4] **Clement, T. P.**, 1981. The Extraction Of Line-Structured Data From Engineering Drawings, Pattern Recognition, **14**, 43-52.

[5] **Cugini, U., Ferri, U. C., Mussio, P. and Protti, M., 1984.** Pattern-Directed Restoration and Vectorization of Digitized Engineering Drawings, Comput. And Graphics, **8**, 337-350.

[6] **Sanford J. P. and Bixler J. P.**, 1985. A Technique For Encoding Lines And Regions In Engineering Drawings, Pattern Recognition, **18**, 367-377.

[7] **Okazaki, A., Kondo, T., Mori, K., Tsunekawa, S. and Kawamoto, E.**, 1988. An Automatic Circuit Diagram Reader With Loop-Structure-Based Symbol Recognition, IEEE Transactions On PAMI, **10**, 331-341.

[8] **Kasturi, R. and Fletcher, L. A.**, 1988. A Robust Algorithm For Text String Separation From Mixed Text/Graphic Images, IEEE Transactions On PAMI, **10**, 910-918.

[9] **Kasturi, R., Bow, S. T., El-Masri, W., Shah, J., Gattiker, J. R. and Mokate, U. M.**, 1990. A System For Interpretation Of Line Drawings, IEEE Transactions On PAMI, **12**, 978-992.

[10] **Messelodi, S. and Modena, C. M.**, 1999. Automatic Identification And Skew Estimation Of Text Lines In Real Scene Images, Pattern Recognition, **32**, 791-810.

[11] **Lu, Z.**, 1998. Detection Of Text Regions From Digital Engineering Drawings, IEEE Transactions On PAMI, **20**, 431-439.

[12] **Wu, V., Manmatha, R., and Riseman, E. M.**, 1999. Text Finder: An Automatic System To Detect And Recognize Text In Images, IEEE Transactions On PAMI, **21**, 1224-1229.

[13] **Min, W., Tang, Z., and Tang, L.**, 1993. Using Web Grammer To Recognize Dimensions In Engineering Drawings, Pattern Recognition, **26**, 1407-1416.

[14] **Yu, Y., Samal, A. and Seth, S.**, 1994. Isolating Symbols From Connection Lines In A Class Of Engineering Drawings, Pattern Recognition, **27**, 391-404.

[15] **Cosmos, J. P. and Hibbert, R.**, 1991. Geometrical testing of three dimensional objects with the aid of pattern recognition, IEEE Proceedings-E, **138**, 250-254.

[16] **Jansen, H. and Krause, F. L.**, 1984. Interpretation of freehand drawings for mechanical design processes, Computer and Graphics, **8**, 351-369.

[17] **Riekert, W. F.**, 1993. Extracting Area Objects from Raster Image Data, IEEE Computer Graphics and Applications, 68-73.

[18] **Zahn, C. T. and Roskies, R. Z.**, 1972. Fouriew Descriptors For Plane Closed Curves, IEEE Transactions On Computers, **c-21**, 269-281.

[19] **Lindenbaum, M. and Bruckstein, A.**, 1993. On Recursive, O(N) Partitioning Of A Digitized Straight Segments, IEEE Transactions On PAMI, **15**, 949-953.

[20] **Flickner, M., Hafner, J., Rodriguez, E. J. and Sanz, J. L. C.**, 1996. Periodic Quasi-Orthogonal Spline Bases and Applications to Least-Squares Curve Fitting of Digital Images, IEEE Transactions on Image Processing, **5**, 71-88.

[21] **Mokhtarian, F. and Mackworth, A. K.**, 1986. Scale-based description and recognition of planar curves and two-dimensional shapes, IEEE Transactions on PAMI, **8**, 34-43.

[22] **Mokhtarian, F. and Mackworth, A. K.**, 1992. A Theory of Multiscale, Curvature Based Shape Representation for Planar Curves, IEEE Transactions on PAMI, **14**, 789-805.

[23] **Liu, H. C. and Srinath, M. D.**, 1990. Partial Shape Classification Using Contour Matching in Distance Transformation, Transactions on PAMI, **12**, 1072-1079.

[24] **Huang, Z. and Cohen, F. S.**, 1996. Affine-Invariant B-Spline Moments For Curve Matching, IEEE Transactions On Image Processing, **5**, 1473-1480.

[25] **Wang, Y. P., Lee, S. L. and Toraichi, K.**, 1999. Multiscale Curvature-Based Shape Representation Using B-Spline Wave, IEEE Transactions On Image Processing, **8**, 1586-1592.

[26] **Rajpal, N., Chaudhury, S. and Banerjee, S.**, 1999. Recognition Of Partially Occluded Objects Using Neural Network Based Indexing, Pattern Recognition, **32**, 1737-1749.

[27] **Bebis, G., Papadourakis, G. and Orphanoudakis, S.**, 1999. Curvature Scale-Space-Driven Object Recognition With An Indexing Scheme Based On Artificial Neural Networks, Pattern Recognition, **32**, 1175-1201.

[28] **Ando, S.**, 2000. Image Field Categorization And Edge/Corner Detection From Gradient Covarience, IEEE Transactions On PAMI, **22**, 179-190.

[29] **Jia, L., Kitchen, L.**, 2000. Object-Based Image Similarity Computation Using Inductive Learning Of Contour-Segment Relations, IEEE Transactions On Image Processing, **9**, 80-87.

[30] **Broggi, A.**, 1995. Parallel and Local Feature Extraction: A Real Time Approach To Road Boundary Detection, IEEE Transactions On Image Processing, **4**, 217-223.

[31] **Jung, K. and Kim, H. J.**, 2000. On-line Recognition Of Cursive Korean Characters Using Graph Representation, Pattern Recognition, **33**, 399-412.

[32] **Plamondon, R., and Privitera, C. R.**, 1999. The Segmentation Of Cursive Handwriting: An Approach Based On Off-Line Recovery Of The Motor-Temporal Information, IEEE Transactions On Image Processing, **8**, 80-91.

[33] **Xie, X., Sudhakar, R. and Zhuang, H.**, 1993. Corner detection by a cost minimization approach, Pattern Recognition, **26**, 1235-1243.

[34] **Mehrotra, R. and Nichani, S.**, 1990. Corner Detection, Pattern Recognition, **23**, 1223-1233.

[35] **Cooper, J., Venkatesh, S. and Kitchen, L.**, 1993. Early Jump-Out Corner Detectors, IEEE Transactions on PAMI, **15**, 823-828.

[36] **Mokhtarian, F. and Suomela, R.**, 1998. Robust Image Corner Detection Through Curvature Scale Space, IEEE Transactions on PAMI, **20**, 1376-1381.

[37] **Beus, H. L. and Tiu, S. S. H.**, 1987. An improved corner detection algorithm based on chain-coded plane curves, Pattern Recognition, **3**, 291-296.

[38] **The, C. H. and Chin, R. T.**, 1989. On the detection of dominant points on digital curves, Transactions on PAMI, **11**, 859-872.

[39] **Liu H. C. and Srinath, M. D.**, 1990. Corner detedtion from chain-code, Pattern Recognition, **23**, 51-68.

[40] **Rattarangsi, A. and Chin, R. T.**, 1992. Scale-based Detection of Corners of Planar Curves, Transactions on PAMI, **14**, 430-449.

[41] **Sheu, H. T. and Yang, H. Z.**, 1993. Open curve segmentation via a two-phase scheme, Pattern Recognition, **26**, 1839-1844.

[42] **Lee, J. S., Sun, Y. N., Chen, C. H. and Tsai, C. T.**, 1993. Wavelet based corner detection, Pattern Recognition, **26**, 853-865.

[43] **Lee, J. S., Sun, Y. N. and Chen, C. H.**, 1995. Multiscale Corner Detection by Using Wavelet Transform, IEEE Transactions On Image Processing, **4**, 100-104.

[44] **Huang, S. C. and Sun, Y. N.**, 1999. Polygonal Approximation Using Genetic Algorithms, Pattern Recognition, **32**, 1409-1420.

[45] **Li, L. and Chen, W.**, 1999. Corner Detection and Interpretation On Planar Curves Using Fuzzy Reasoning, IEEE Transactions On PAMI, **21**, 1204-1210.

[46] **Sheu, H. T. and Hu, W. C.**, 1999. Multiprimitive Segmentation Of Planar Curves-A Two-Level Breakpoint Classification And Tuning Approach, IEEE Transactions On PAMI, **21**, 791-797.

[47] **Ji, Q. and Haralick, R. M.**, 1998. Breakpoint Detection Using Covariance Propagation, IEEE Transactions On PAMI, **20**, 845-851.

[48] **Ip, H. H. S. and Wong, W. H.**, 1997. Corner Detection and Interpretation On Planar Curves Using Fuzzy Reasoning, IEE Proceedings Vision, Image and Signal Processing, **144**, 23-30.

[49] **Pavlidis, T. and Horowitz, S. L.**, 1974. Segmentation of Plane Curves, IEEE Transactions on Computers, **c-23**, 860-870.

[50] **Shapiro, S. D.**, 1978. Feature Space Transforms For Curve Detection, Pattern Recognition, **10**, 129-143.

[51] **Gao, Q. G. and Wong, A. K. C.**, 1993. Curve Detection Based On Perceptual Organization, Pattern Recognition, **26**, 1039-1046.

[52] **Wang, M. J. J. and Wu, W. Y.**, 1993. Elliptical Object Detection By Using Its Geometric Properties, Pattern Recognition, **26**, 1499-1509.

[53] **Sethi, I. K. and Yoo, J. H.**, 1993. An Ellipse Detection Method From The Polar And Pole Definition Of Conics, Pattern Recognition, **26**, 307-315.

[54] **Goldgof, D., Ranganathan, N. and Kumar, S.**, 1994. Parallel Algorithms For Circle Detection In Images, Pattern Recognition, **27**, 1019-1028.

[55] **Palmer, P. L., Kittler, J. and Petrou, M.**, 1994. Using Focus Of Attention With The Hough Transform For Accurate Line Parameter Estimation, Pattern Recognition, **27**, 1127-1134.

[56] **Chen L. H. and Ho, C. T.**, 1995. A Fast Ellipse/Circle Detector Using Geometric Symmetry, Pattern Recognition, **28**, 117-124.

[57] **Zapata, E. L., Villalba, J. and Guil, N.**, 1995. A Fast Hough Transform For Segment Detection, IEEE Transactions On Image Processing, **4**, 1541-1548.

[58] **Cheng, Y. C. and Lee, S. C.**, 1994. A New Method For Quadratic Curve Detection Using K-Ransac With Acceleration Techniques, Pattern Recognition, **28**, 663-682.

[59] **Tsai, W. H. and Lo, R. C.**, 1995. Gray-Scale Hough Transform For Thick Line Detection In Gray Scale Images, Pattern Recognition, **28**, 647-661.

[60] **Breuel, T. M.**, 1996. Finding Lines Under Bounded Error, Pattern Recognition, **29**, 167-178.

[61] **Chen, L. H. and Ho, C. T.**, 1996. A High Speed Algorithm For Elliptical Object Detection, IEEE Transactions On Image Processing, **5**, 547-550.

[62] **Wang, S., Liu, B. and Kulkarni, S. R.**, 1996. Model-Based Reconstruction of Multiple Circular and Elliptical Objects From A Limited Number Of Projections, IEEE Transactions On Image Processing, **5**, 1386-1390.

[63] **Fitzgibbon, A., Pilu, M. and Fisher, R. B.**, 1999. Direct Least Square Fitting Of Ellipses, IEEE Transactions On PAMI, **21**, 476-480.

[64] **Şenatalar, M.**, 1978. Diferansiyel Geometri, İstanbul Devlet Müh. ve Mim. Ak. Yayınları, Kutulmuş Matbaası, İstanbul.

[65] **Gürdal, Z. ve Kamat, M. P.**, 1990. Elements of Structural Optimization, Kluwer Academic Publishers.

[66] **Lam, L., Lee, S. W. and Suen, C. Y., 1992.** Thinning Methodologies-A Comprehensive Survey, IEEE Transactions on PAMI, **14**, 869-885.

[67] **Davies, E. R. and Plummer, A. P. N.**, 1981. Thinning Algorithms: A Critique And A New Methodology, Pattern Recognition, **14**, 53-63.

[68] **Shivaprasad, A. P. and Govindan, V. K.**, 1987. A Pattern Adaptive Thinning Algorithm, Pattern Recognition, **20**, 623-637.

[69] **Bourbakis, N. G.**, 1989. A Parallel-Symmetric Thinning Algorithm, Pattern Recognition, **22**, 387-396.

[70] **Lam, L., Lee, S. W. and Suen C. Y.**, 1992. Thinning Methodologies-A Comprehensive Survey, IEEE Transactions on PAMI, **14**, 869-885.

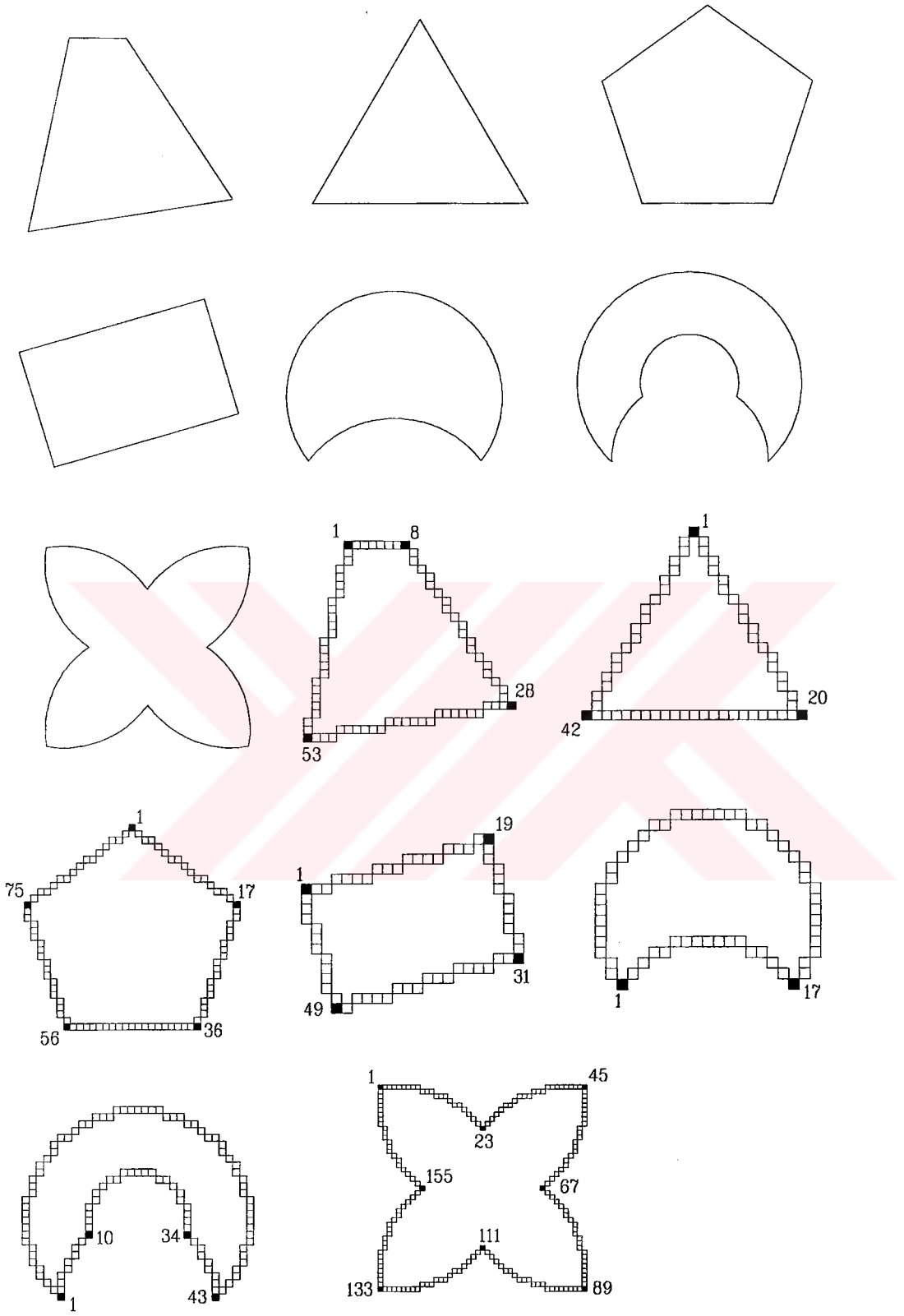
[71] **Chin, R. T. and Jang, B. K.**, 1992. One-Pass Parallel Thinning: Analyses, Properties, and Quantitative Evaluation, IEEE Transavtions on PAMI, **14**, 1129-1140

[72] **Parui, S. K. and Datta, A.**, 1994. A Robust Parallel Thinning Algorithm For Binary Images, Pattern Recognition, **27**, 1181-1192.

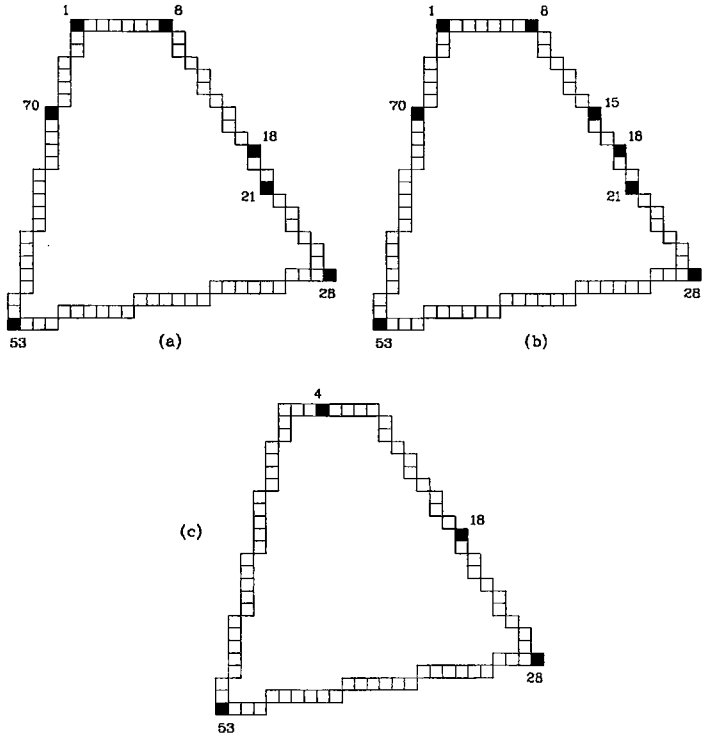
[73] **Petrosino, A. and Salvi, G.**, 2000. A Two-Subcycle Thinning Algorithm And Its Parallel Implementation On SIMD Machines, IEEE Transactions On Image Processing, **9**, 277-283.

EK A

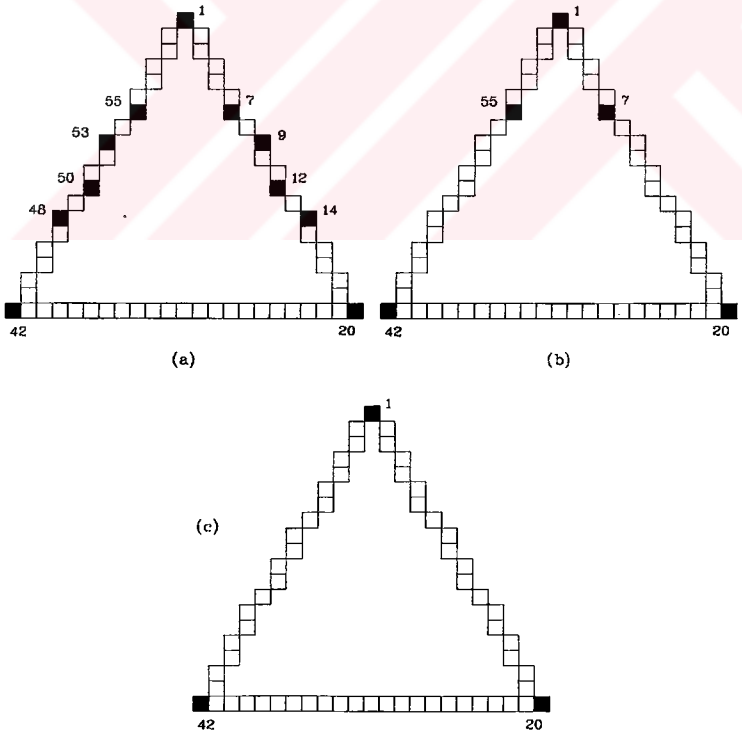
Burada kaynak taramalarımızdan hareketle karşılaştırma amacıyla seçtiğimiz üç köşe bulma yöntemi (Rosenfeld-Johnston, Rosenfeld-Weszka ve Medioni-Yasumoto köşe bulma yöntemleri) ile bizim geliştirdiğimiz yedi köşe bulma yönteminin (kronolojik sırada Bartem, Moment, Spthr, Spmin, Sp3min, Hatmin ve Lsq köşe bulma yöntemleri) Şekil A.1' de verilen sayısal eğrilere uygulama sonuçları verilmiştir. Ayrıca Şekil A.72 ve Şekil A.73' deki resimlere sadece Rosenfeld-Johnston, Rosenfeld-Weszka, Medioni-Yasumoto ve geliştirdiğimiz Lsq (en iyi sonuçları verdiği için) köşe bulma yöntemi uygulanmıştır.



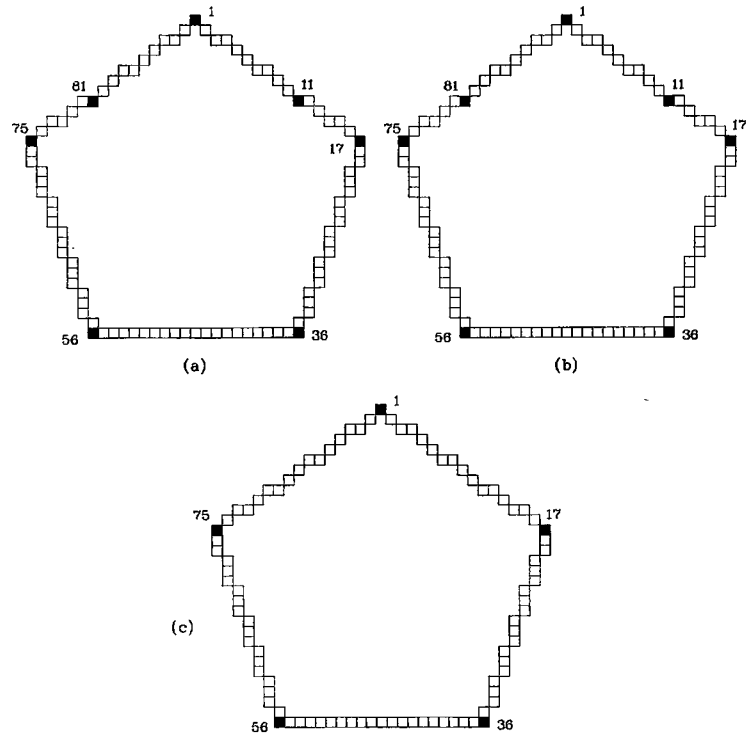
Şekil A.1 Rosenfeld-Johnston, Rosenfeld-Weszka, Medioni-Yasumoto, Bartem, Moment, Spthr, Spmin, Sp3min, Hatmin ve Lsq köşe bulma yöntemlerinin uygulandıkları geometrilerin orijinal ve sayısallaştırılmış durumları.



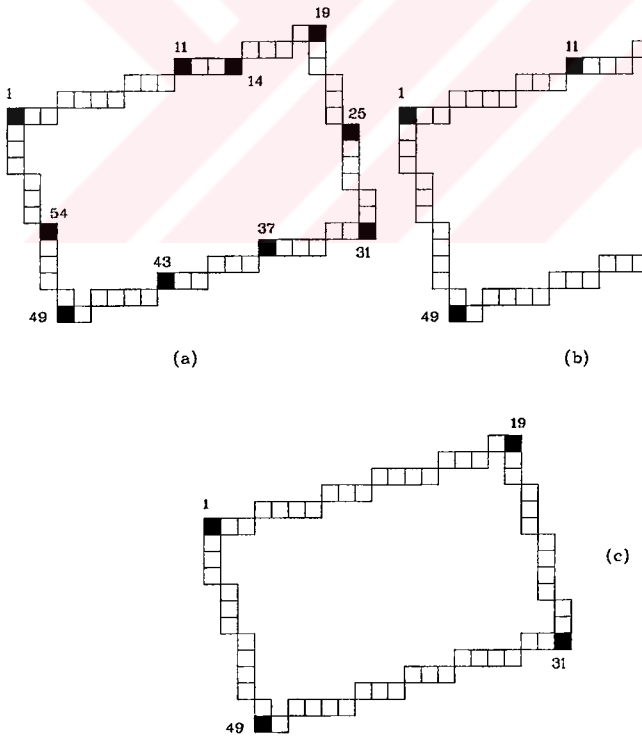
Şekil A.2 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



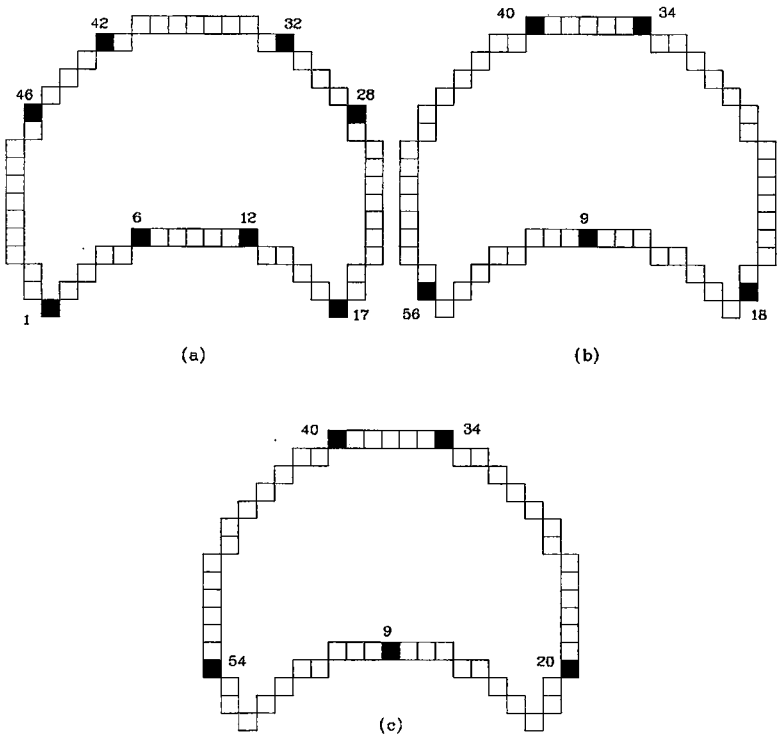
Şekil A.3 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



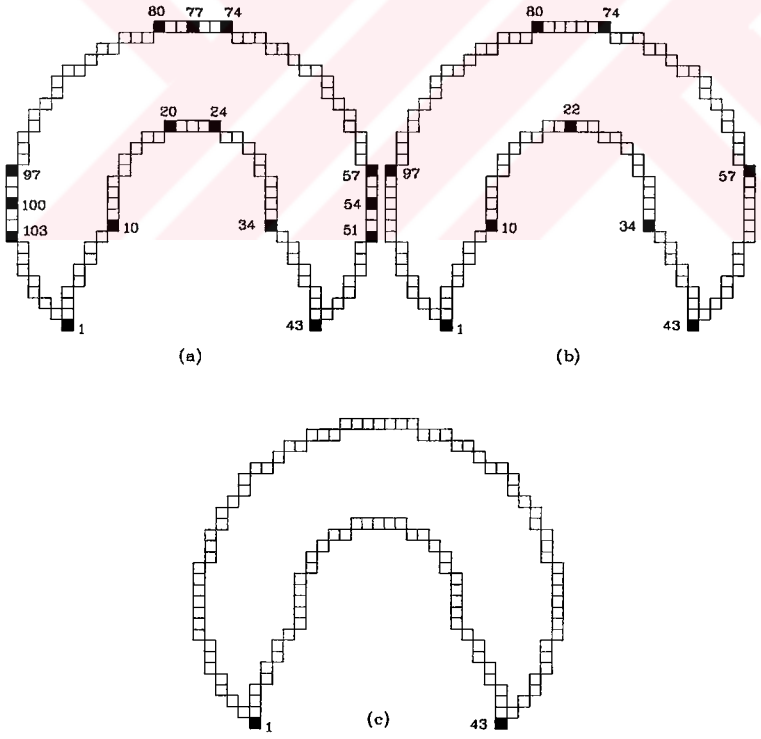
Şekil A.4 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



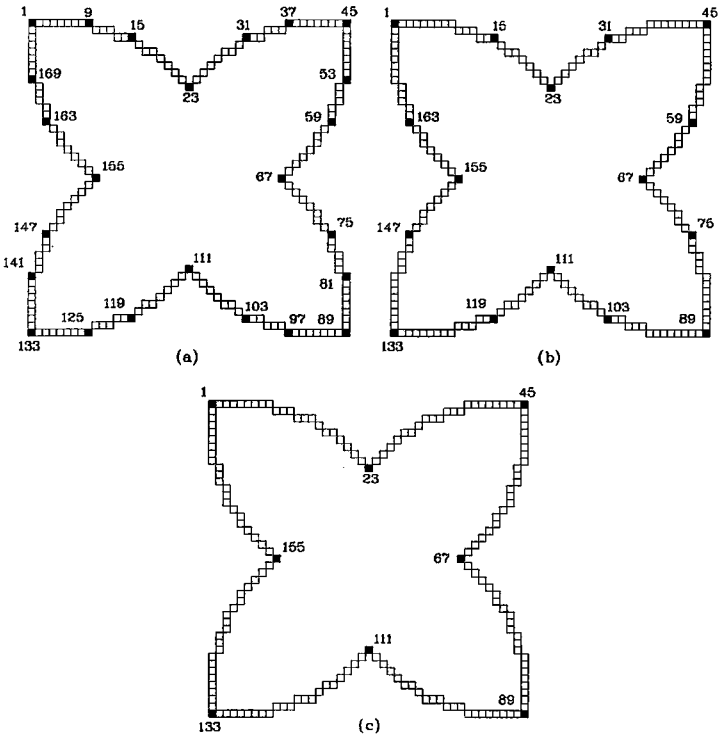
Şekil A.5 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



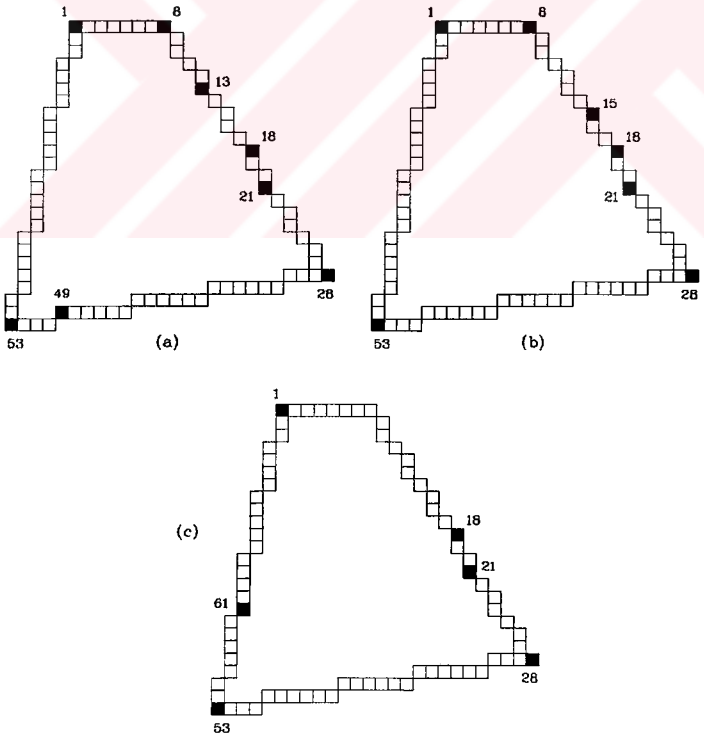
Şekil A.6 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



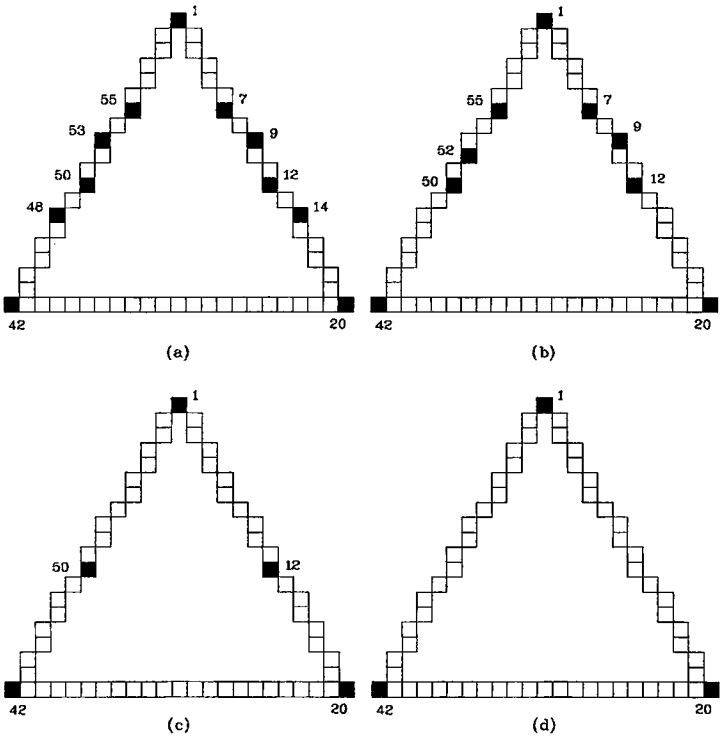
Şekil A.7 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



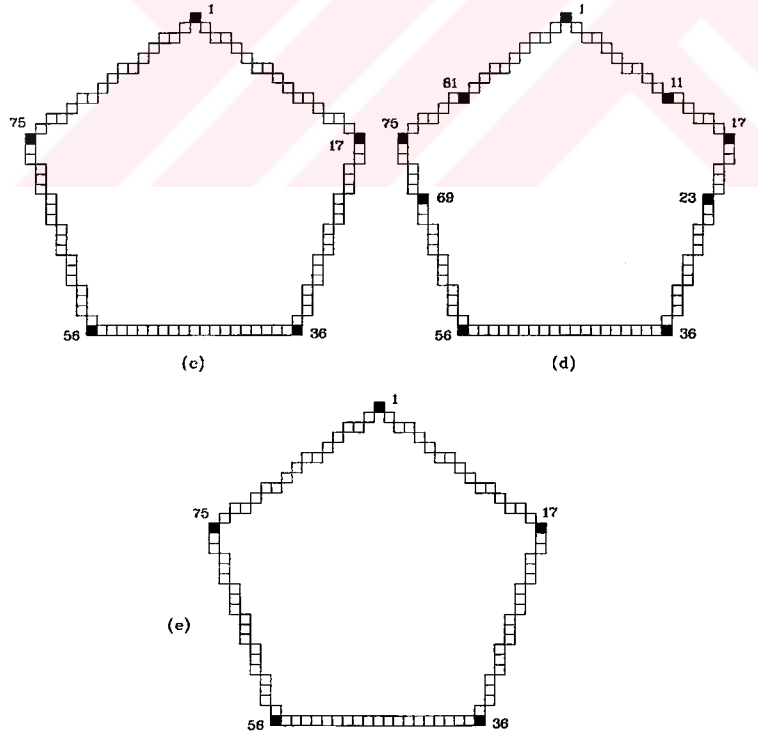
Şekil A.8 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=8 (c) m=9



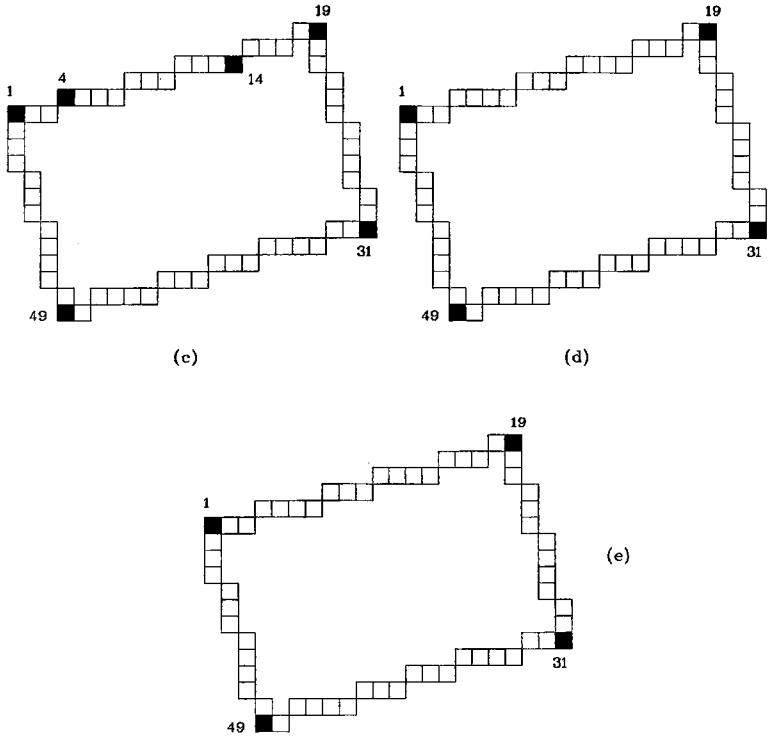
Şekil A.9 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=8 (c) m=9



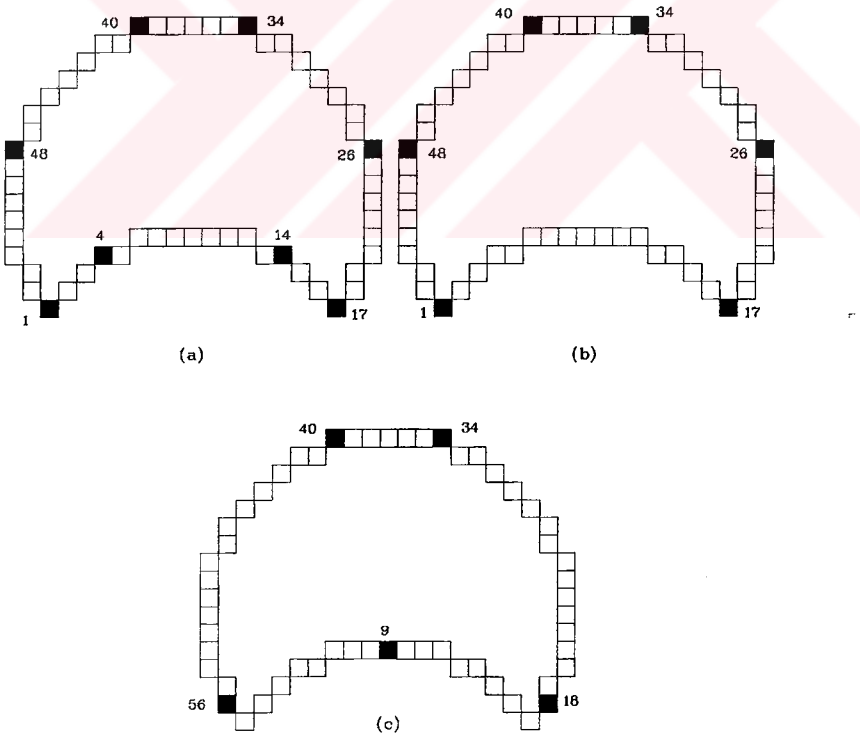
Şekil A.10 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9 (d) m=11



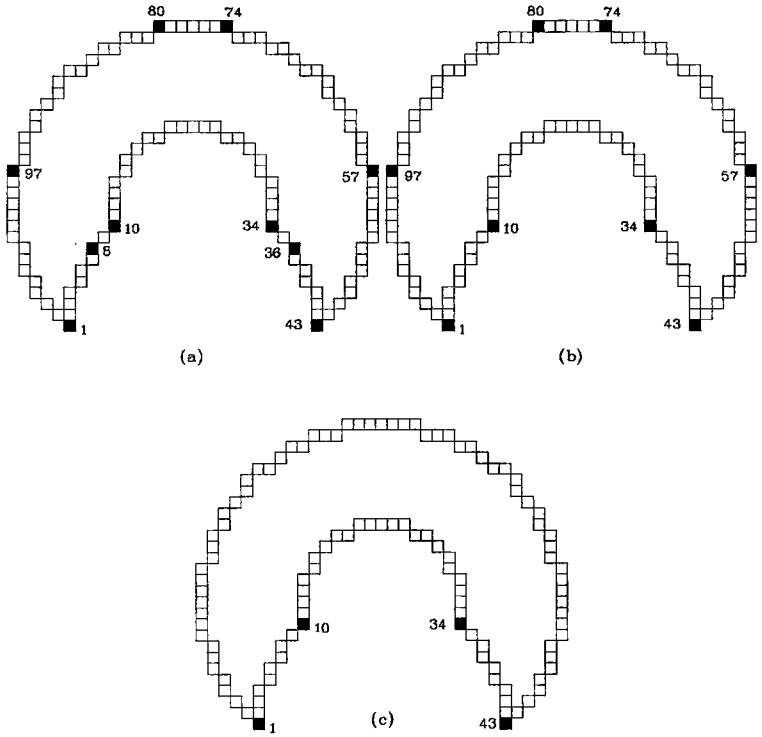
Şekil A.11 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



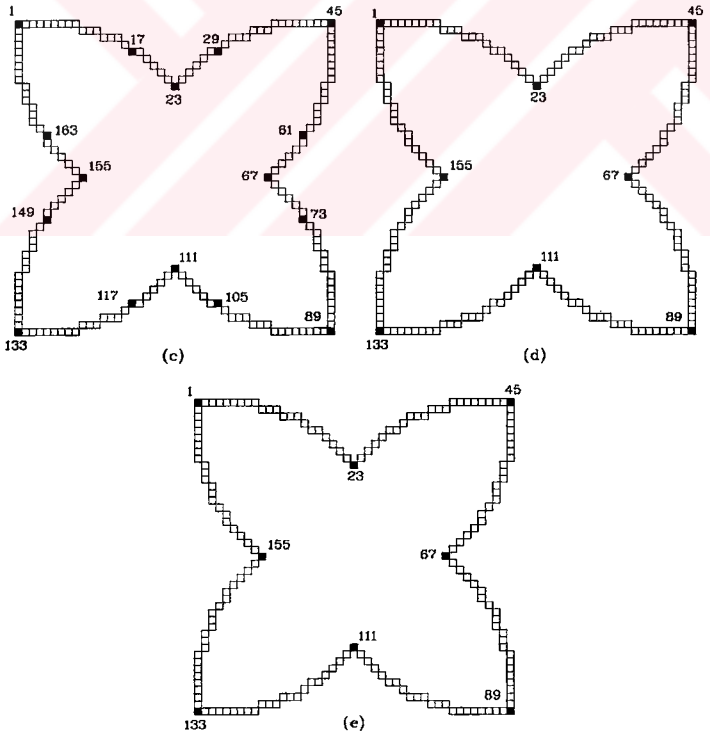
Şekil A.12 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



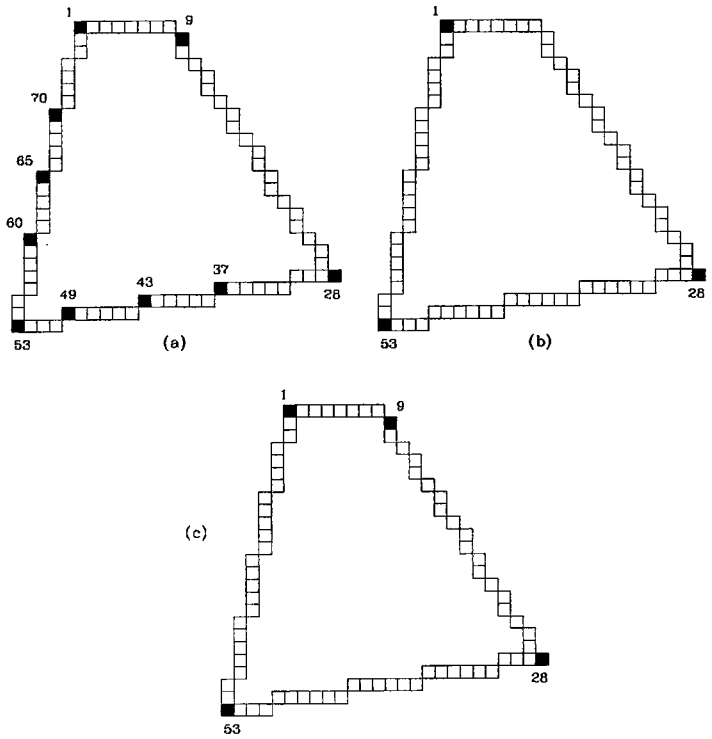
Şekil A.13 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9



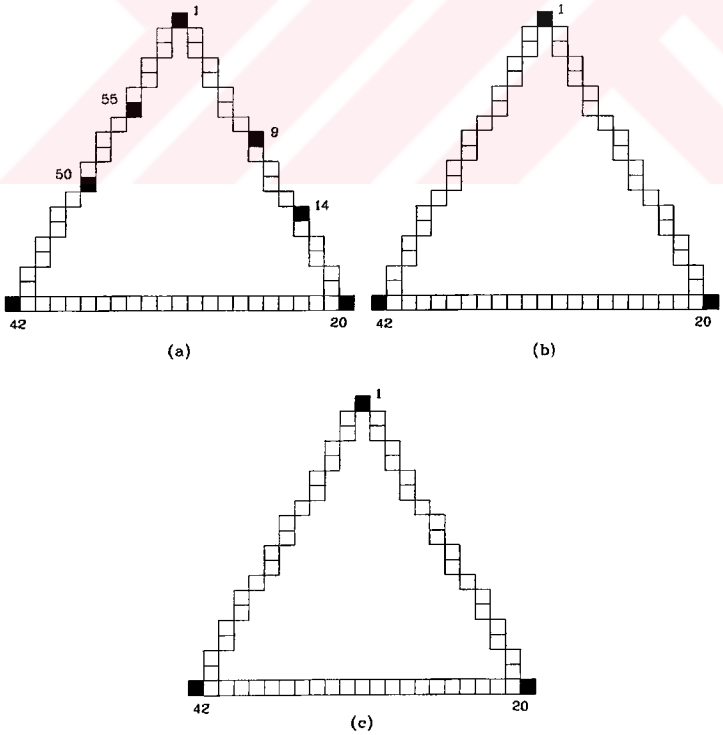
Şekil A.14 Rosenfeld-Weszka yönteminin uygulaması. (a) m=7 (b) m=9 (c) m=11



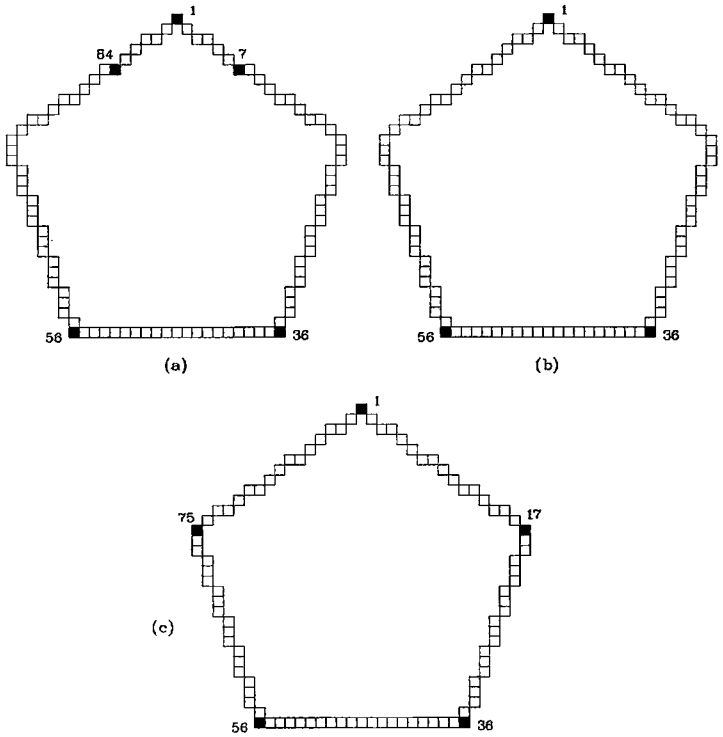
Şekil A.15 Rosenfeld-Weszka yönteminin uygulaması. (a) m=7 (b) m=8 (c) m=9



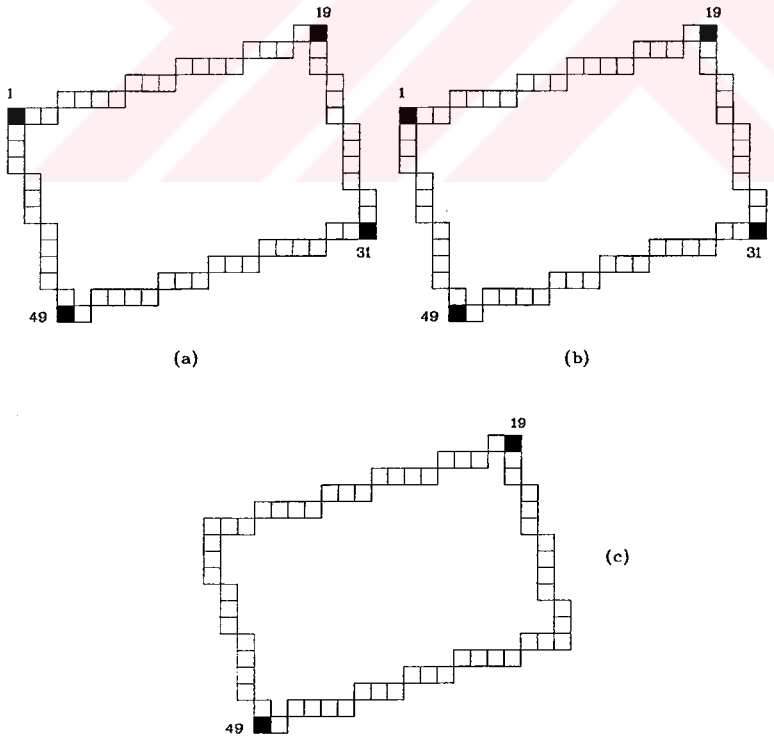
Şekil A.16 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.2, $|i-j|$=3 **(b)** c_t=0.3, δ_t=0.3, $|i-j|$=3 **(c)** c_t=0.45, δ_t=0.2, $|i-j|$=3 **iken köşeler.**



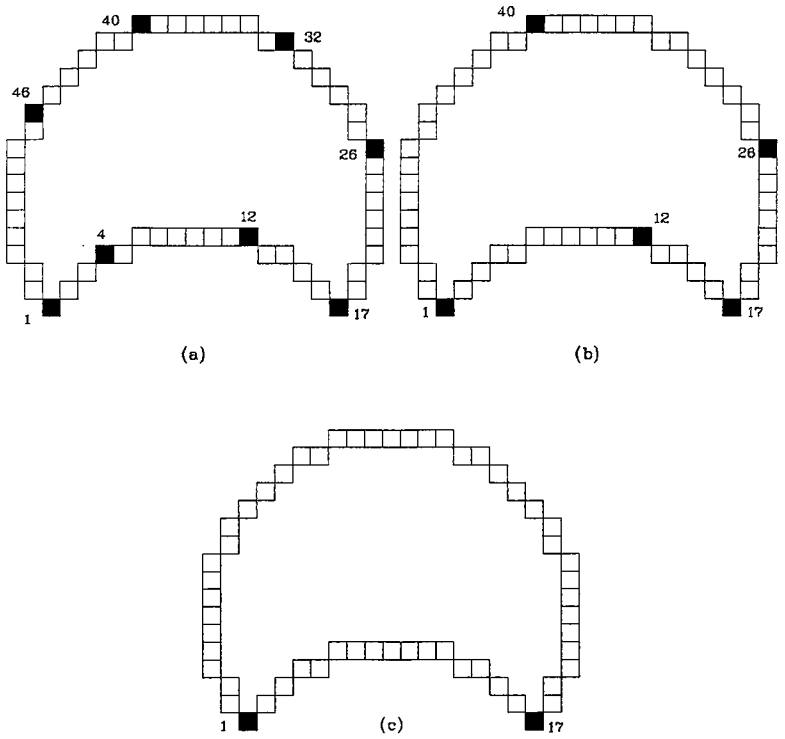
Şekil A.17 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.2, $|i-j|$=3 **(b)** c_t=0.5, δ_t=0.2, $|i-j|$=3 **(c)** c_t=0.3, δ_t=0.2, $|i-j|$=5 **iken köşeler.**



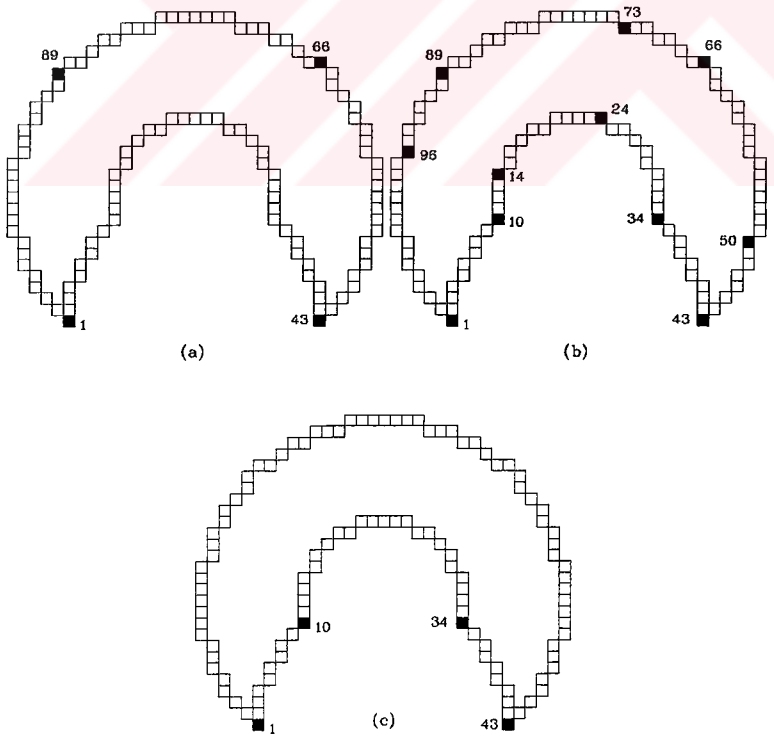
Şekil A.18 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.25, $|i-j|$=3 **(b)** c_t=0.3, δ_t=0.3, $|i-j|$=3 **(c)** c_t=0.3, δ_t=0.26, $|i-j|$=2 **iken köşeler.**



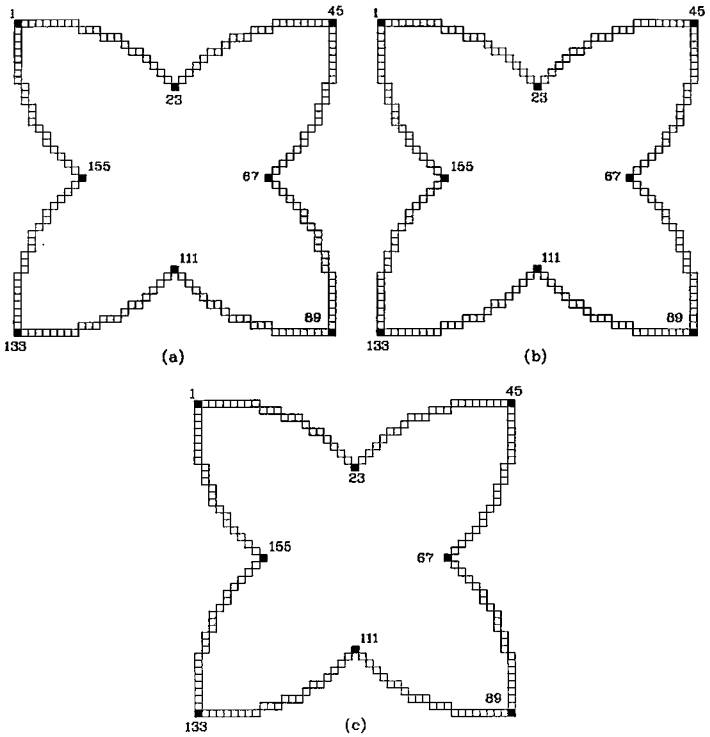
Şekil A.19 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.2, $|i-j|$=3 **(b)** c_t=0.4, δ_t=0.3, $|i-j|$=3 **(c)** c_t=1.7, δ_t=0.3, $|i-j|$=3 **iken köşeler.**



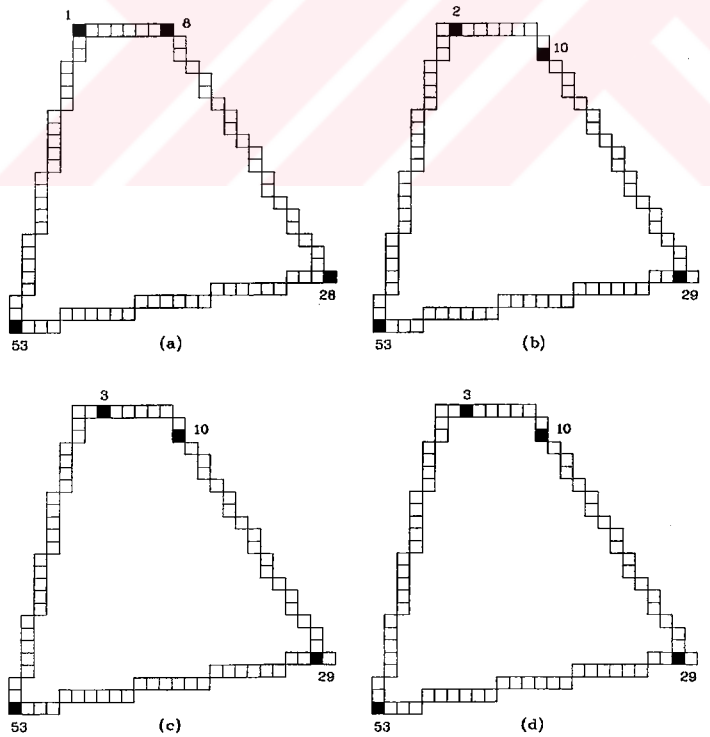
Şekil A.20 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.2, $|i-j|$=3 **(b)** c_t=0.4, δ_t=0.2, $|i-j|$=3 **(c)** c_t=0.3, δ_t=0.3, $|i-j|$=3 **iken köşeler.**



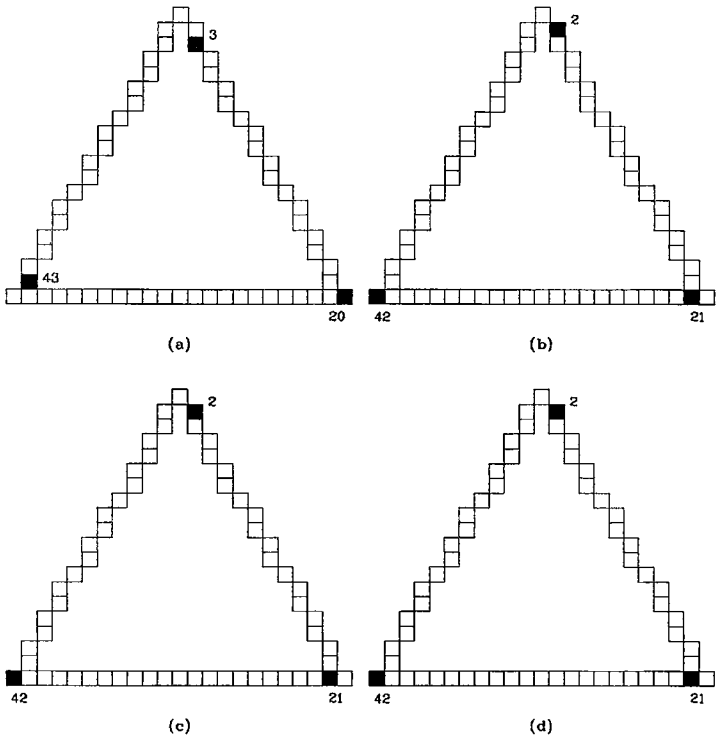
Şekil A.21 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.45, δ_t=0.2, $|i-j|$=3 **(b)** c_t=0.4, δ_t=0.2, $|i-j|$=3 **(c)** c_t=0.3, δ_t=0.26, $|i-j|$=3 **iken köşeler.**



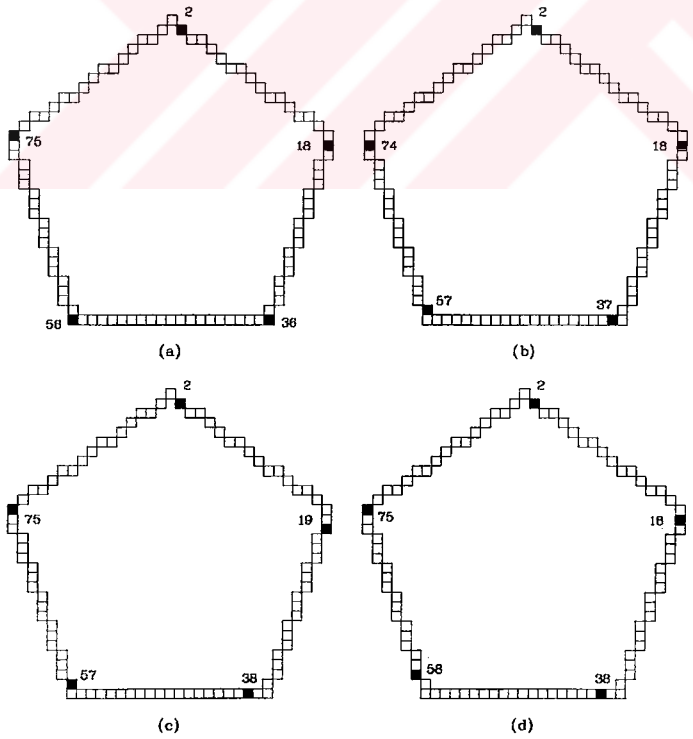
Şekil A.22 Medioni-Yasumoto yönteminin uygulaması. (a) c_t=0.3, δ_t=0.3, $|i-j|$=3 (b) c_t=0.4, δ_t=0.3, $|i-j|$=3 (c) c_t=0.55, δ_t=0.3, $|i-j|$=3 iken köşeler.



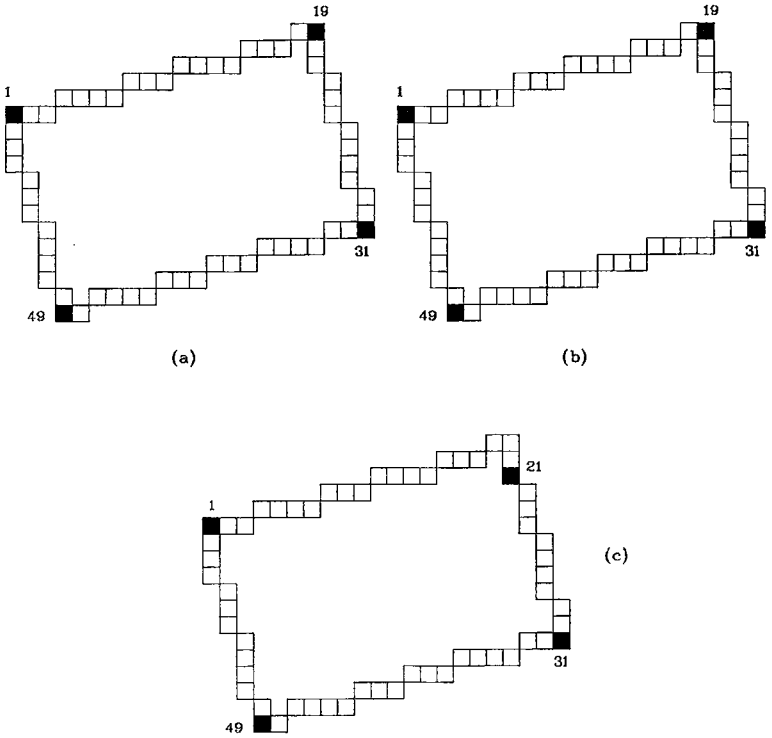
Şekil A.23 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



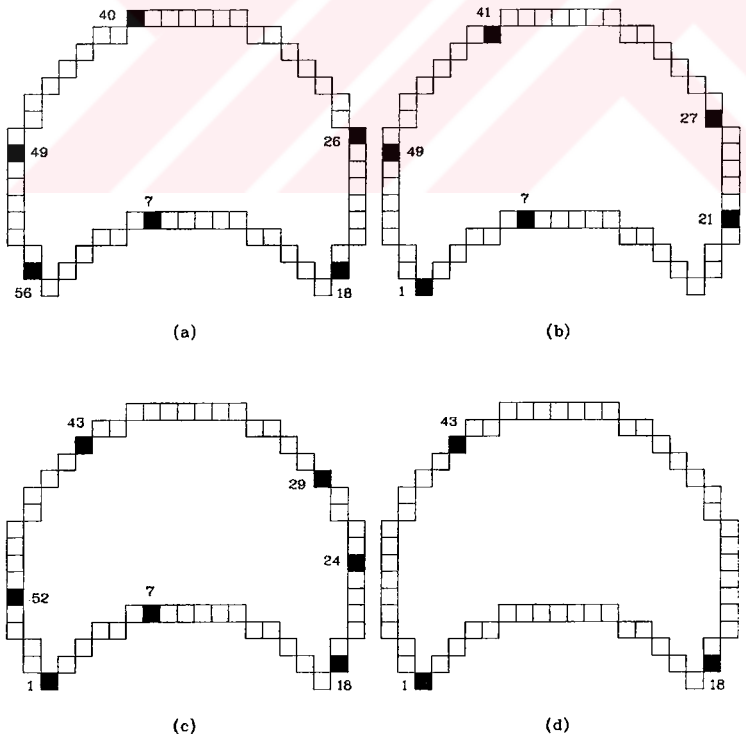
Şekil A.24 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



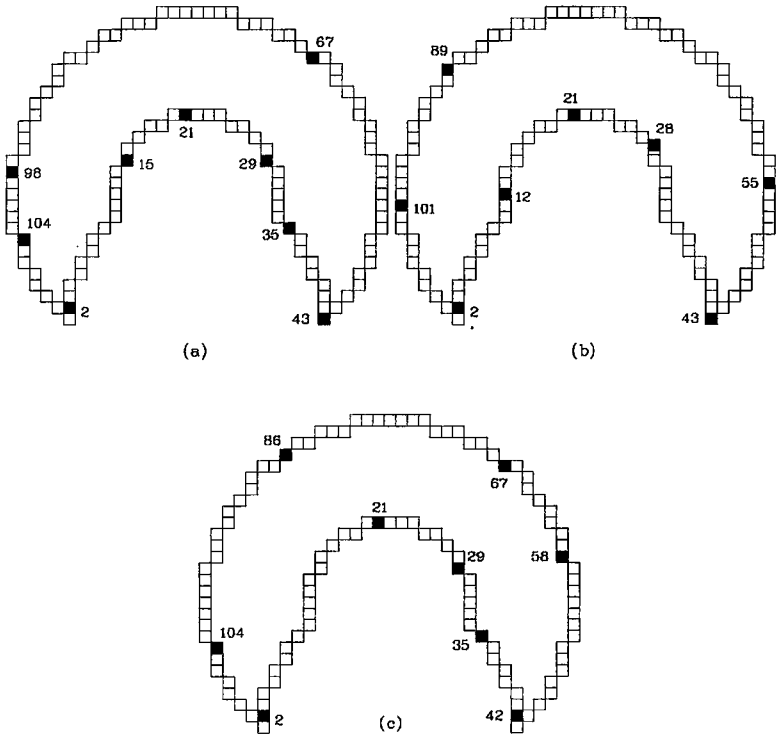
Şekil A.25 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



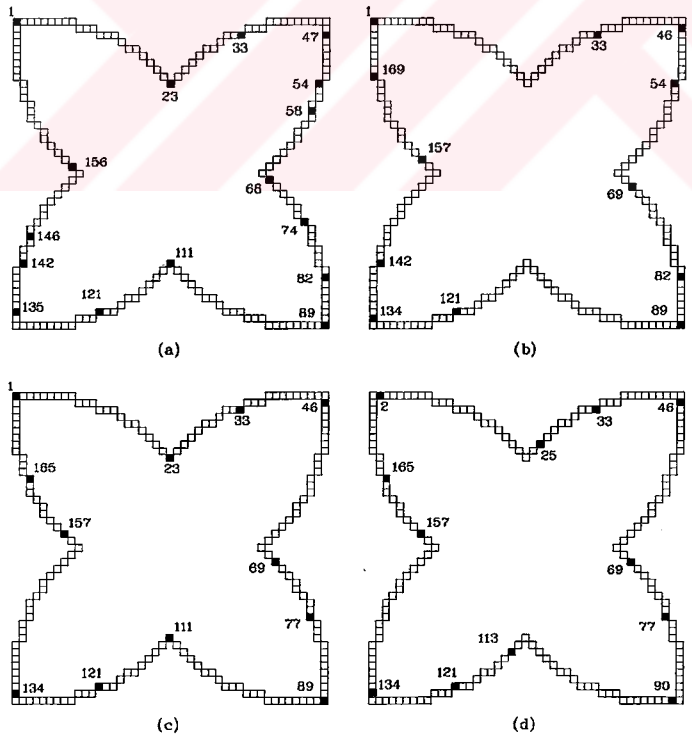
Şekil A.26 Bartem yönteminin uygulaması. (a) k_1=4, k_2=3 (b) k_1=4, k_2=4 (c) k_1=5, k_2=4



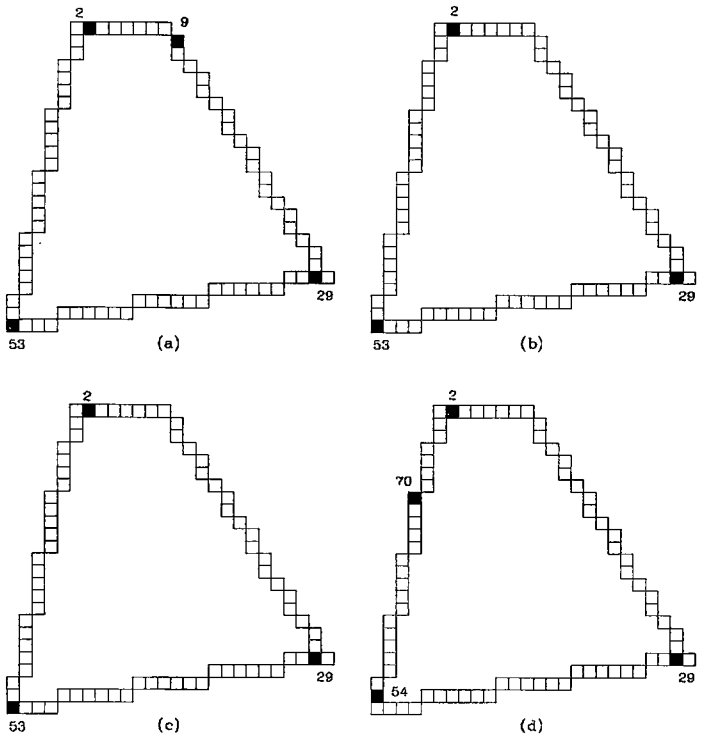
Şekil A.27 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



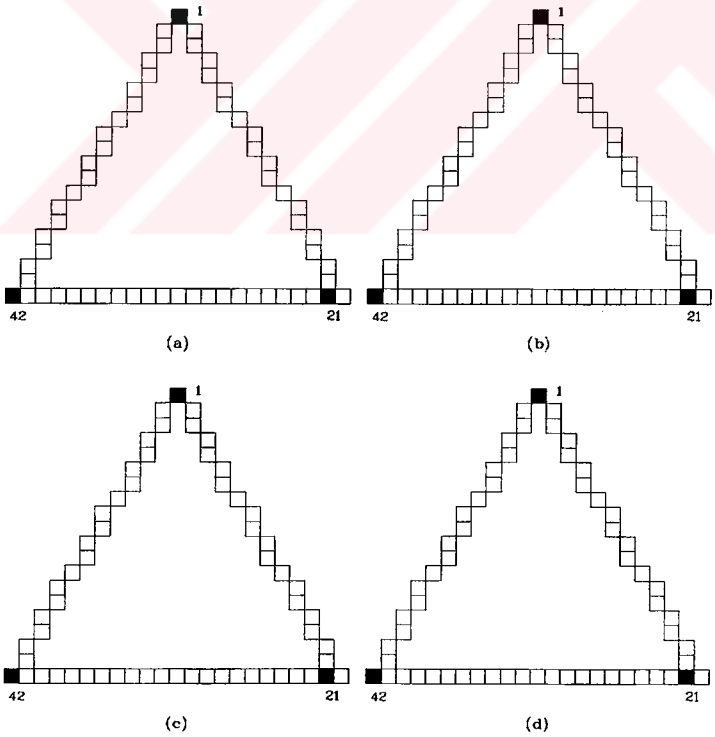
Şekil A.28 Bartem yönteminin uygulaması. (a) k_1=4, k_2=3 (b) k_1=4, k_2=4 (c) k_1=5, k_2=4



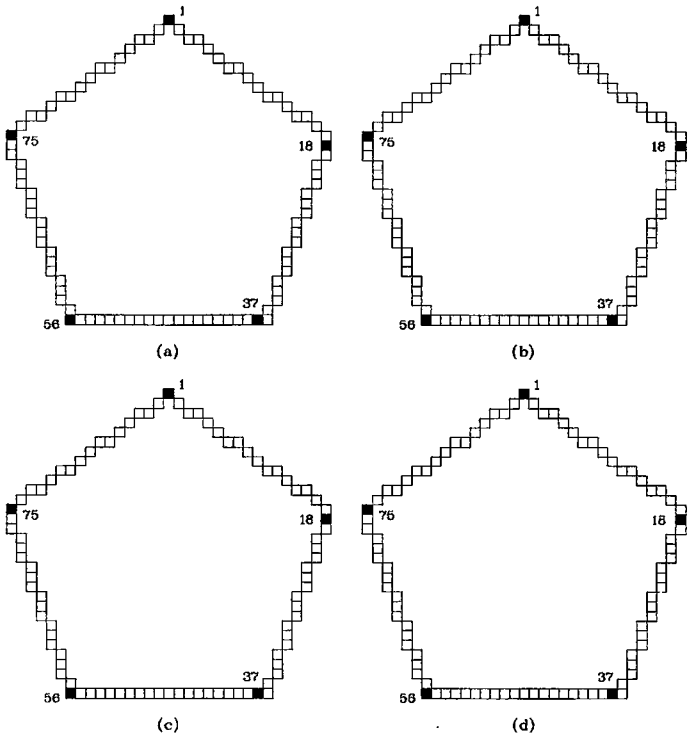
Şekil A.29 Bartem yönteminin uygulaması. (a) k_1=3, k_2=3 (b) k_1=4, k_2=3 (c) k_1=4, k_2=4 (d) k_1=5, k_2=4



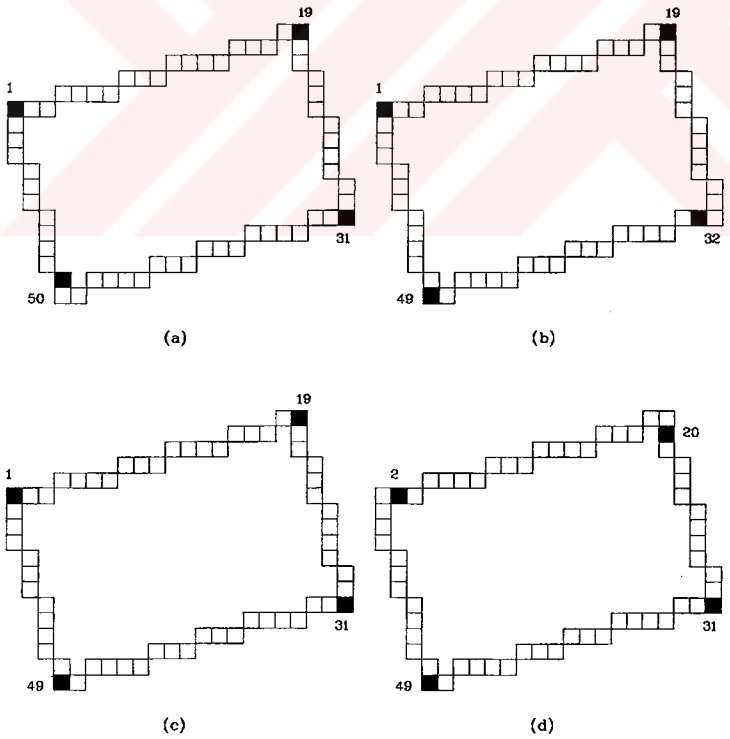
Şekil A.30 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



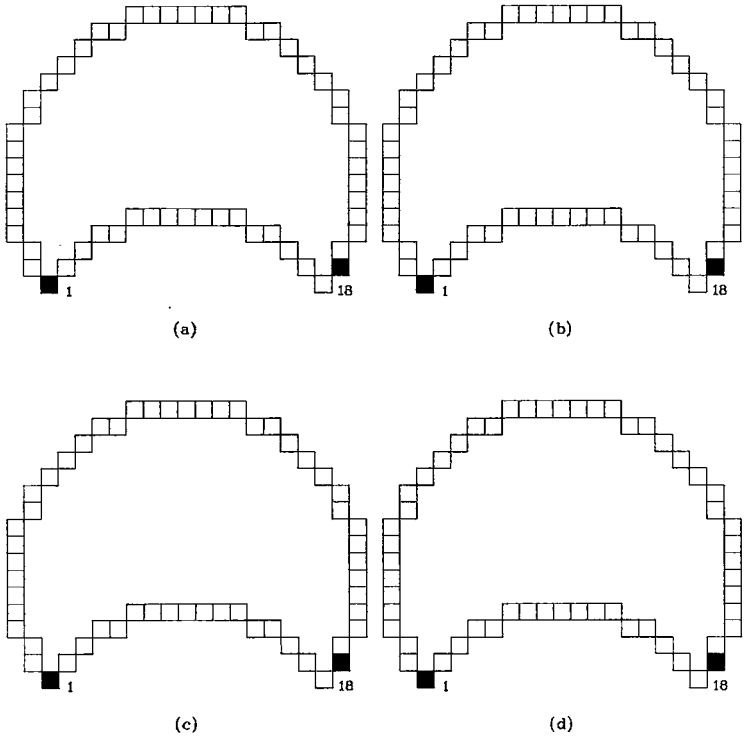
Şekil A.31 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



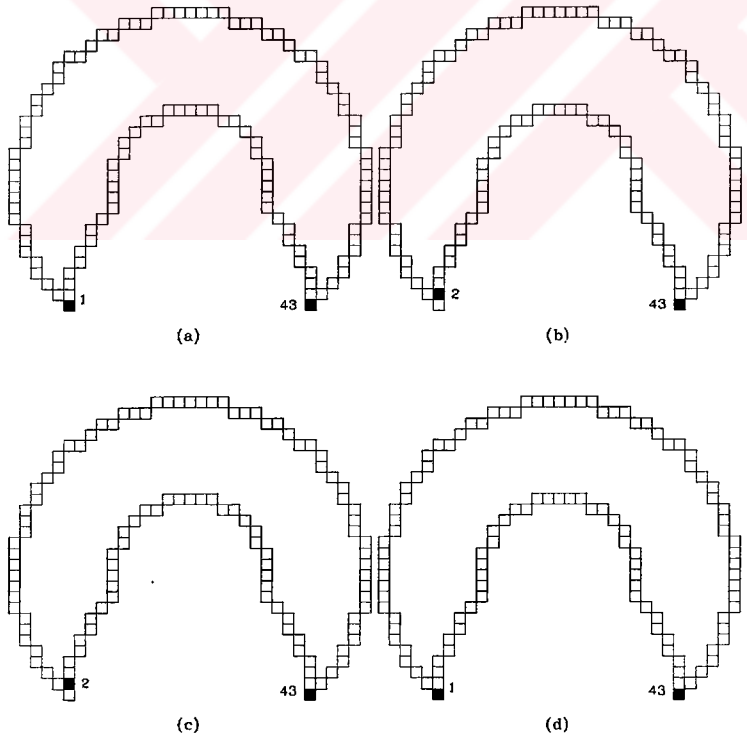
Şekil A.32 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



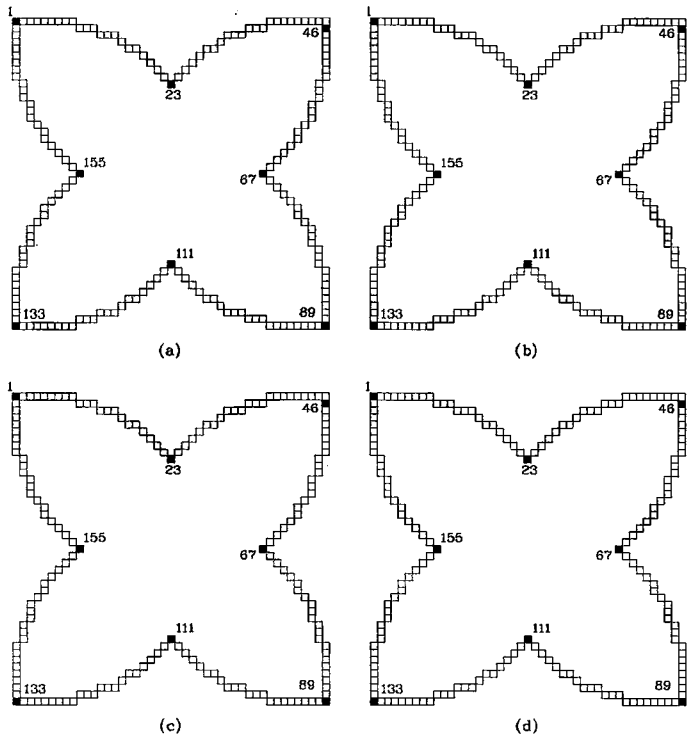
Şekil A.33 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



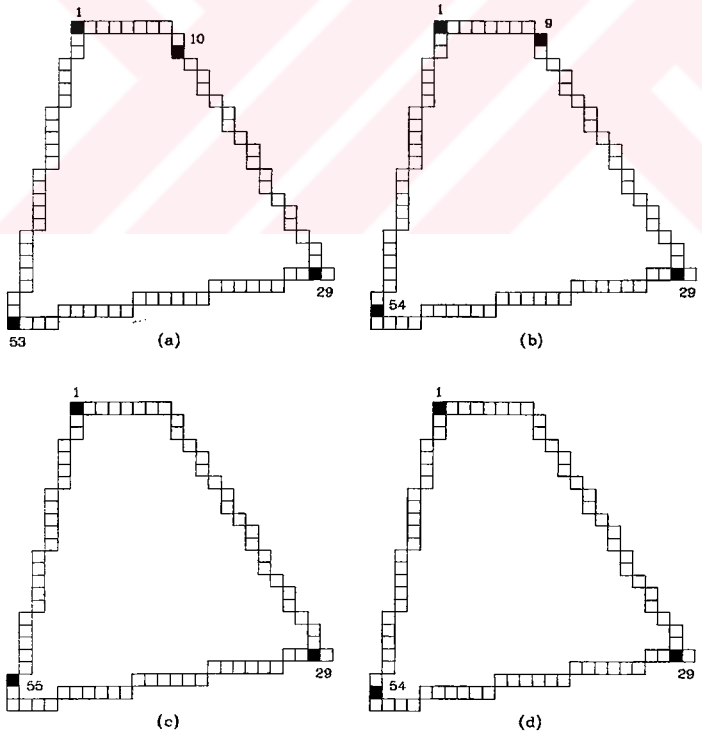
Şekil A.34 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



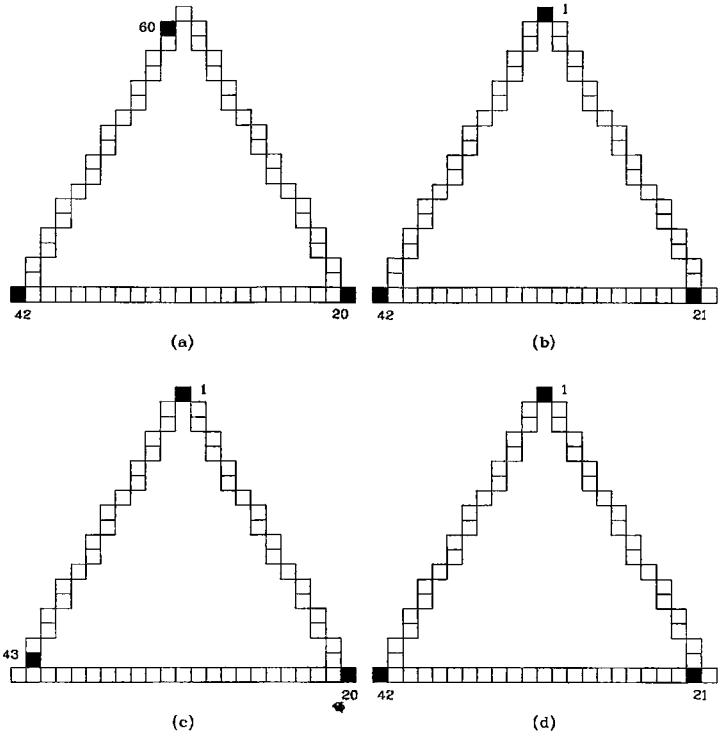
Şekil A.35 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



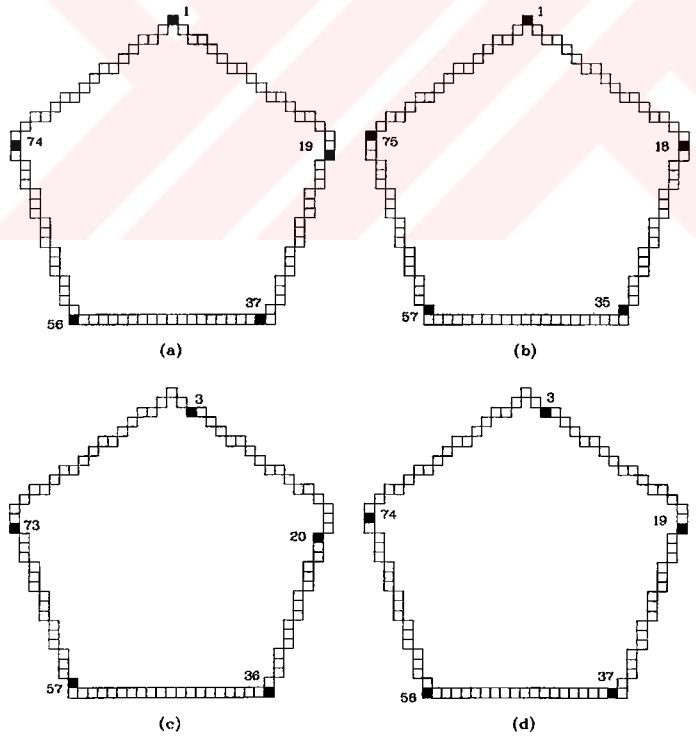
Şekil A.36 Moment yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



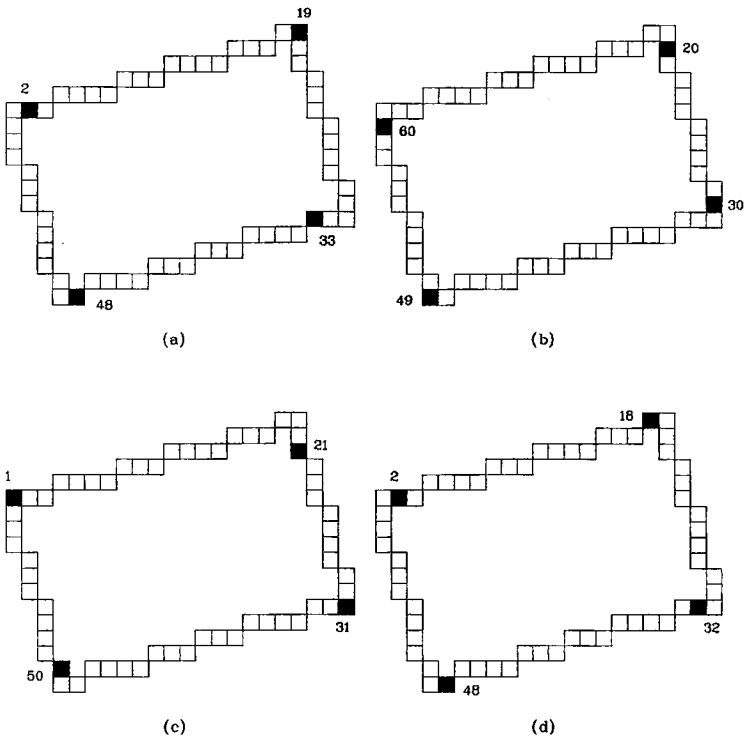
Şekil A.37 Spthr yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=6 (d) k=7



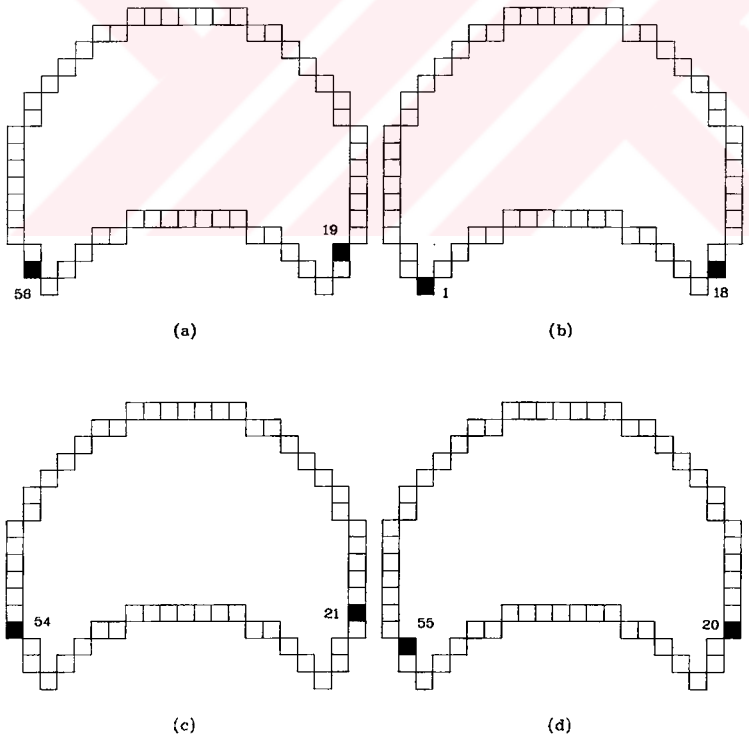
Şekil A.38 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



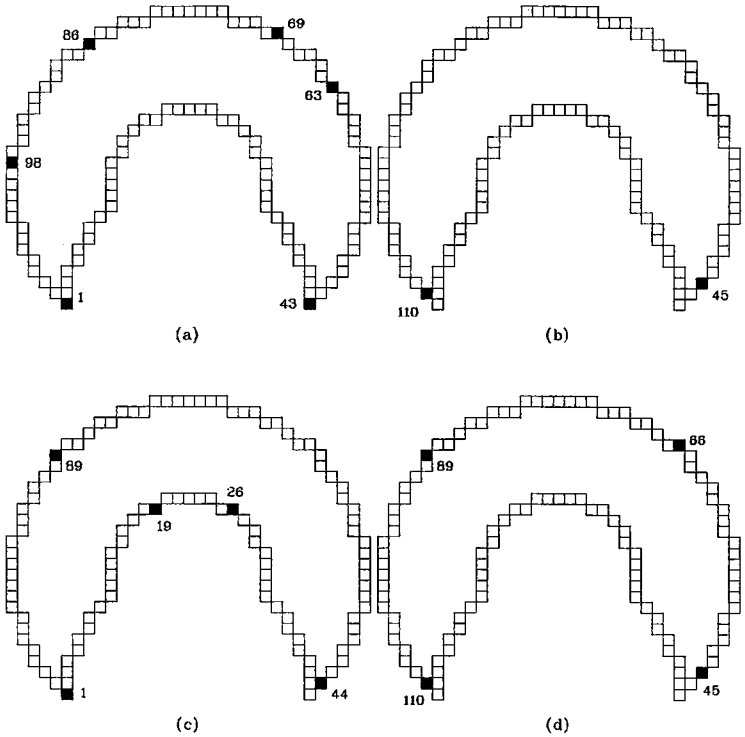
Şekil A.39 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



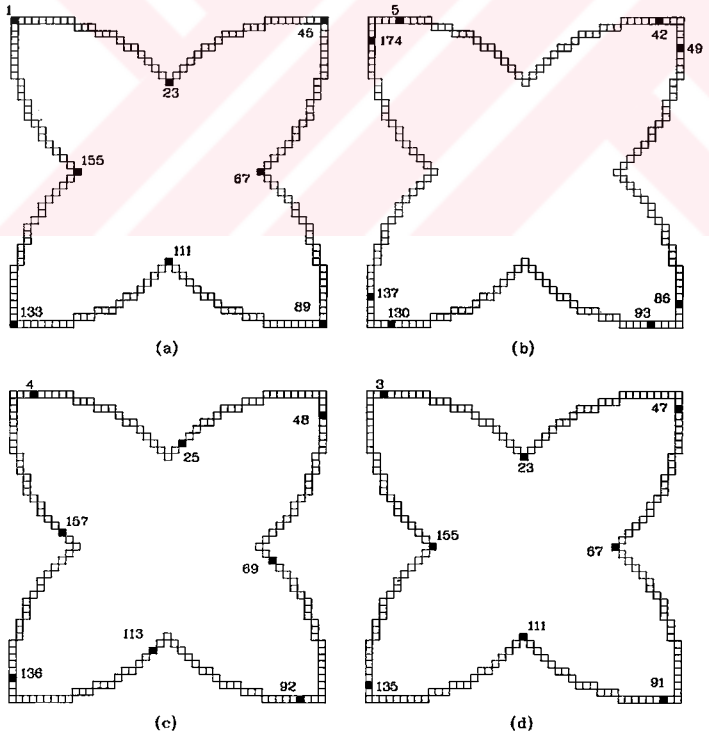
Şekil A.40 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



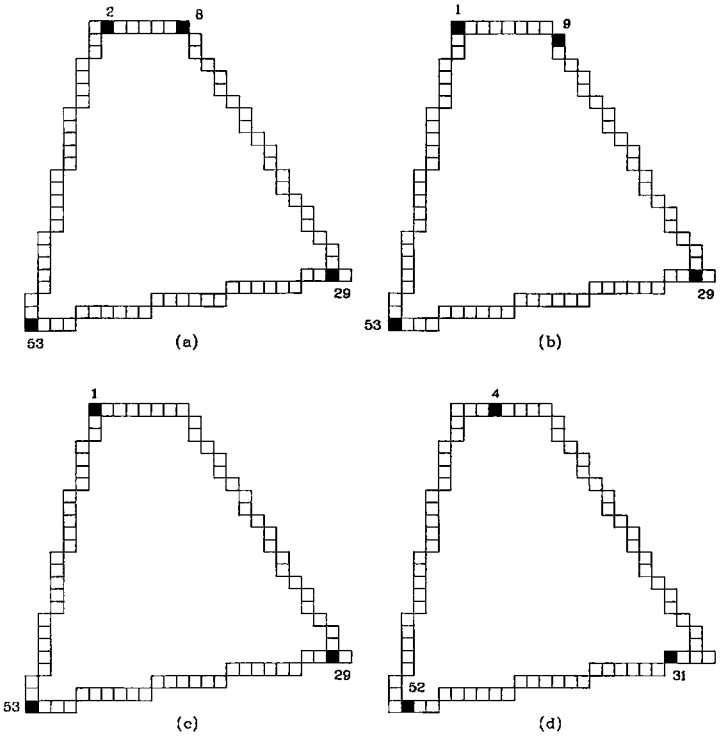
Şekil A.41 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



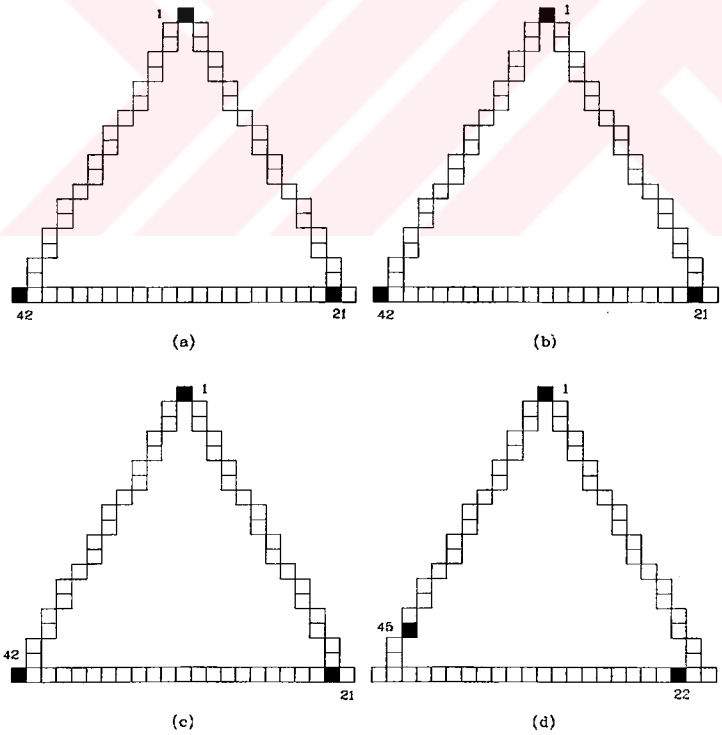
Şekil A.42 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



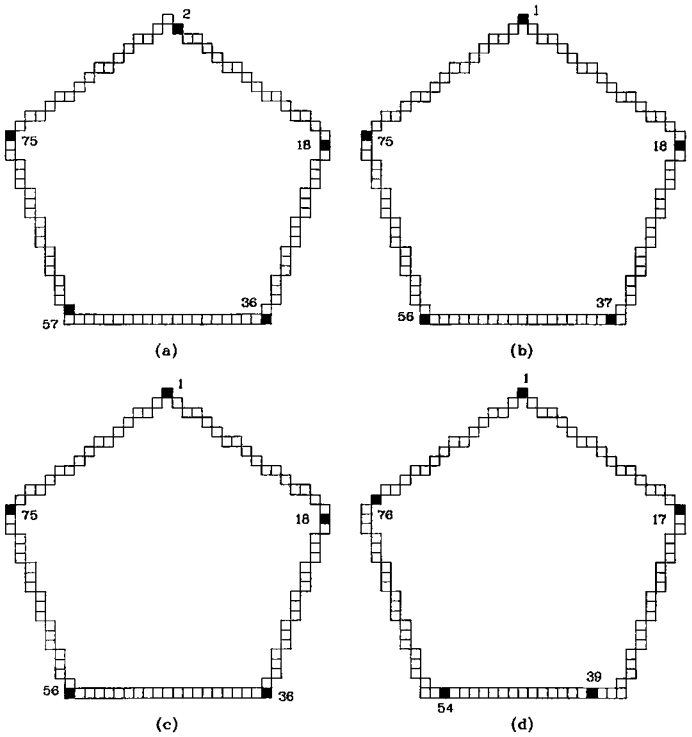
Şekil A.43 Spthr yönteminin uygulaması. (a) k=4 (b) k=5 (c) k=6 (d) k=7



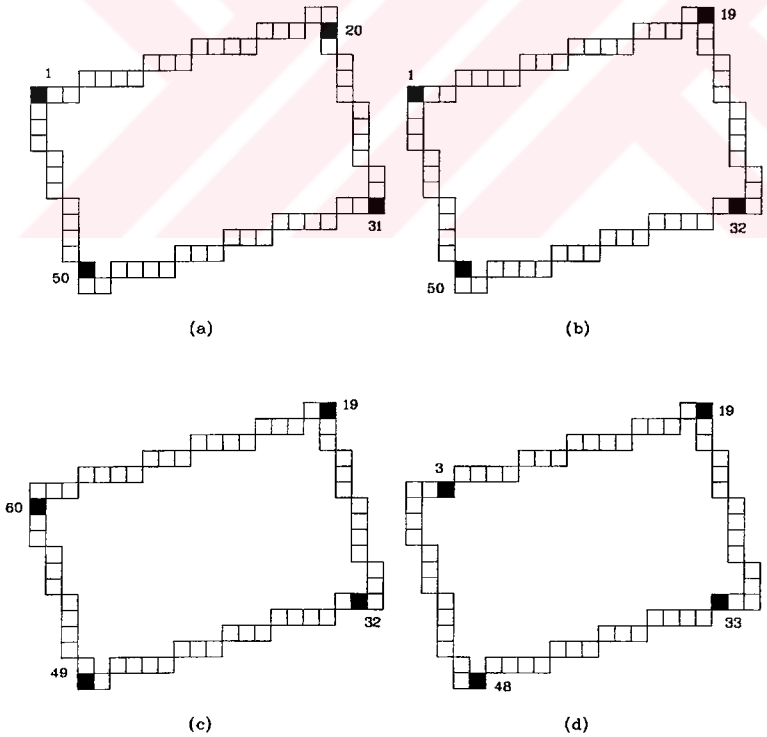
Şekil A.44 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



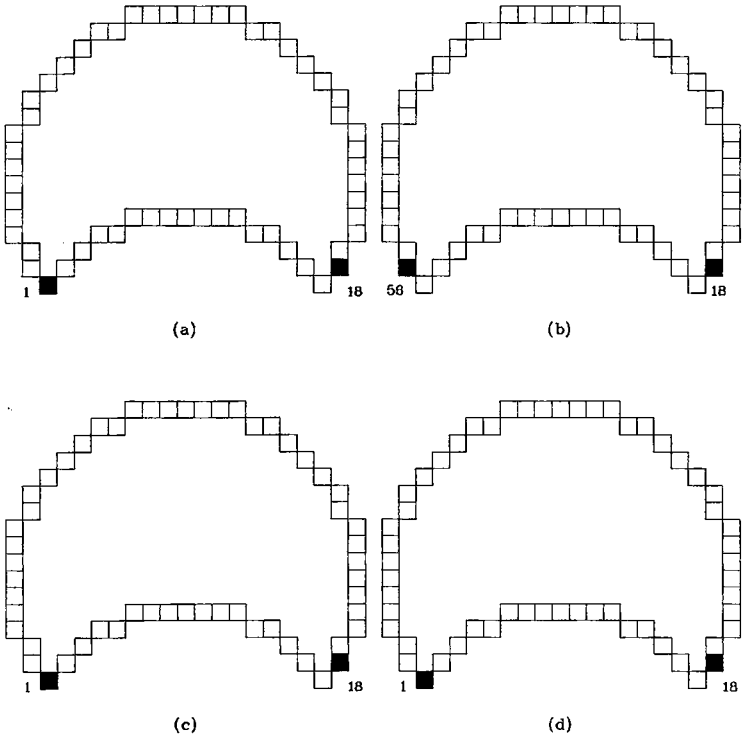
Şekil A.45 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



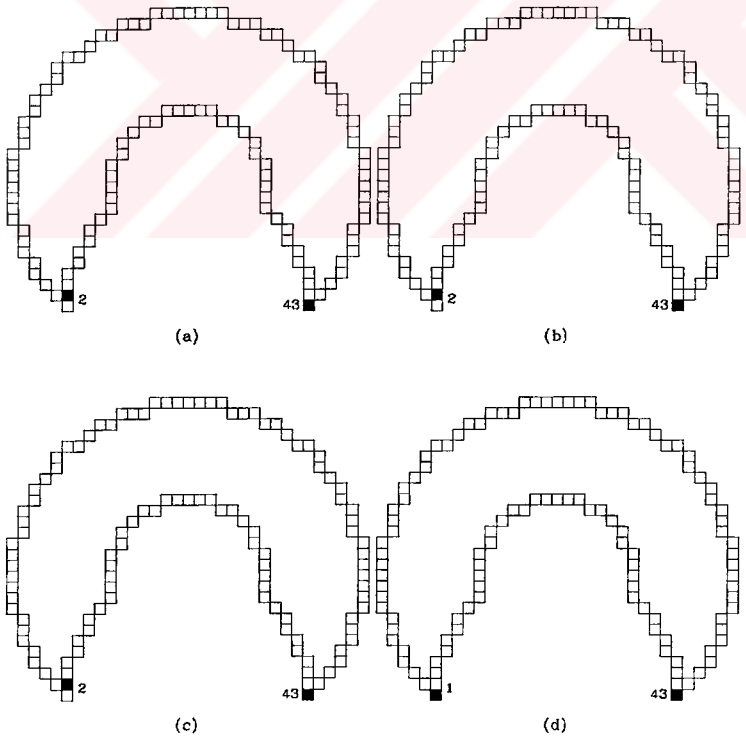
Şekil A.46 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



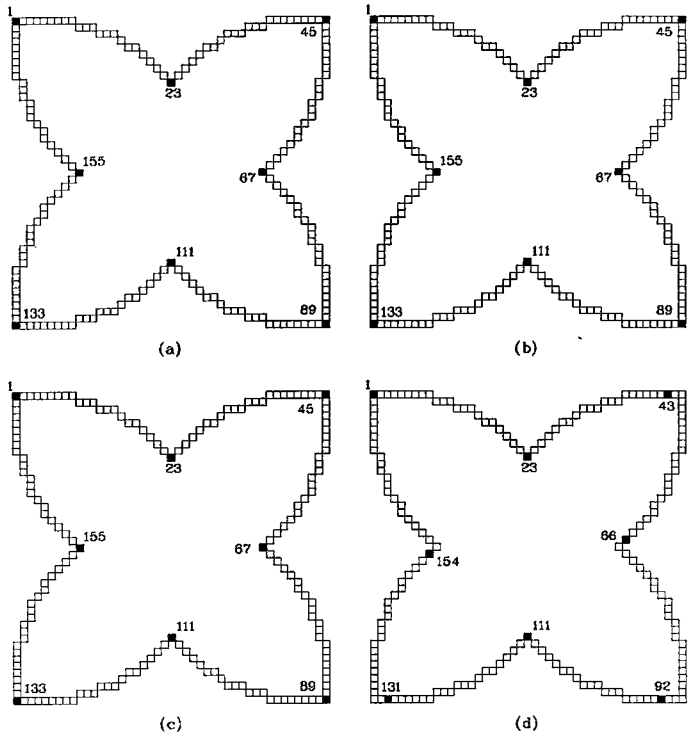
Şekil A.47 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



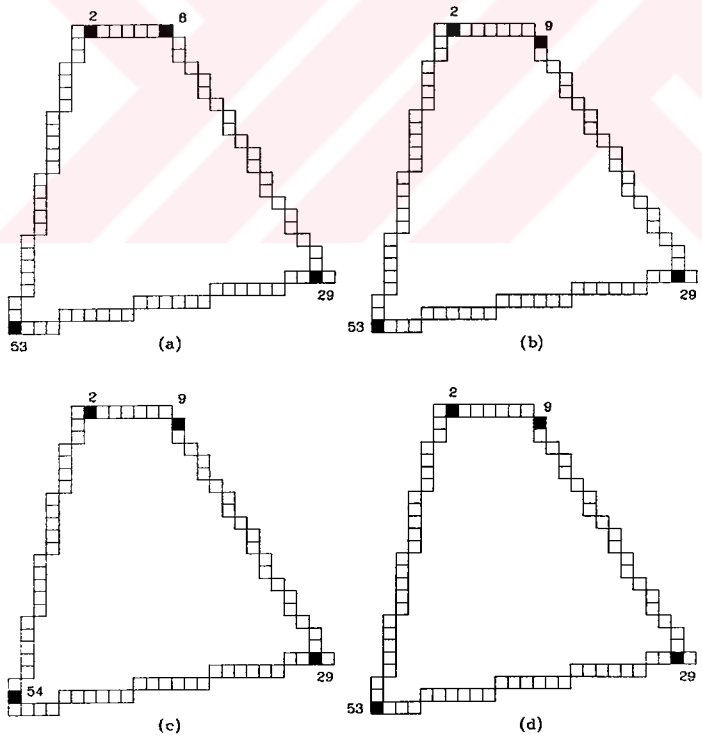
Şekil A.48 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



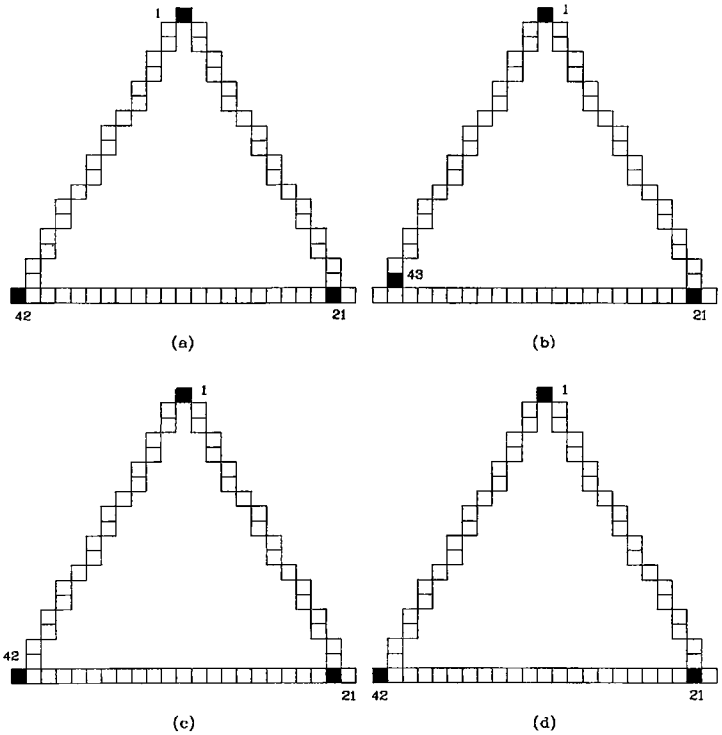
Şekil A.49 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



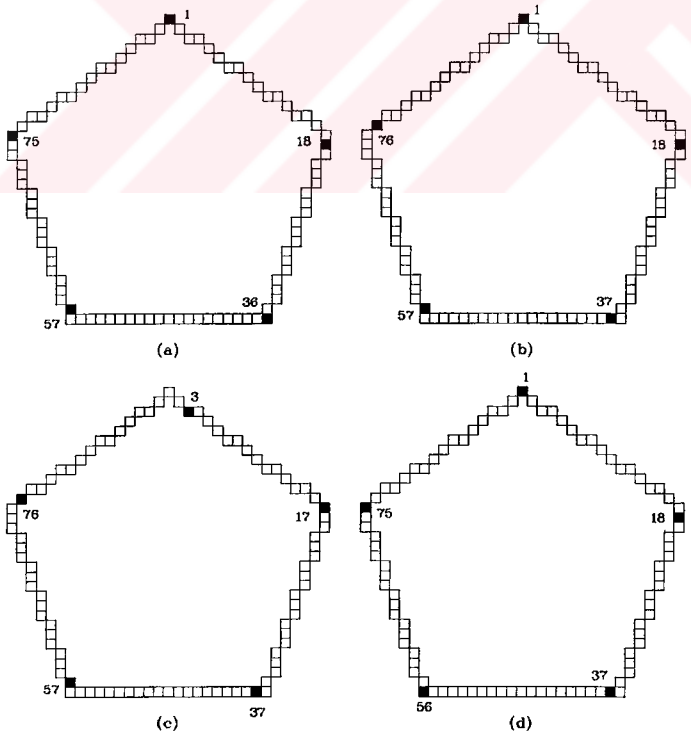
Şekil A.50 Spmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



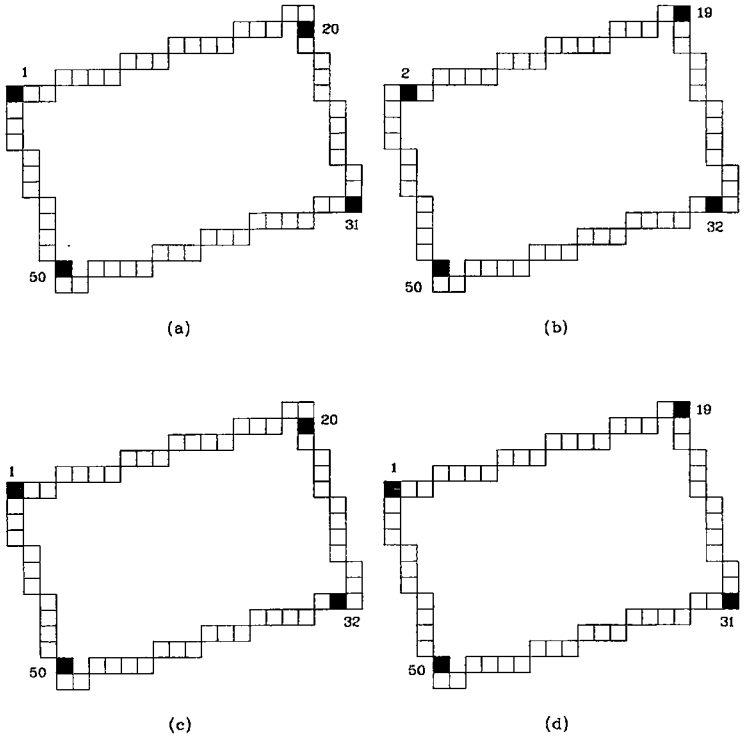
Şekil A.51 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



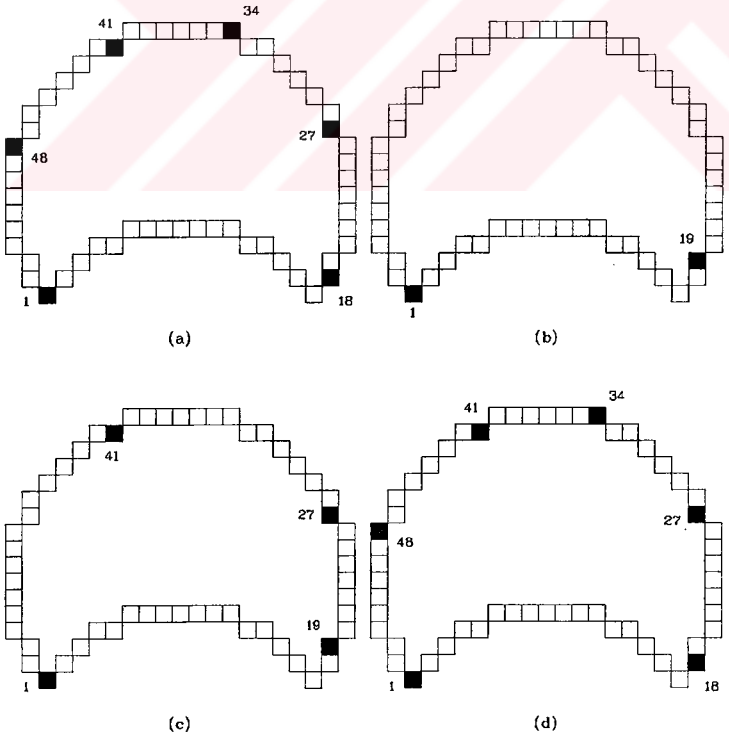
Şekil A.52 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



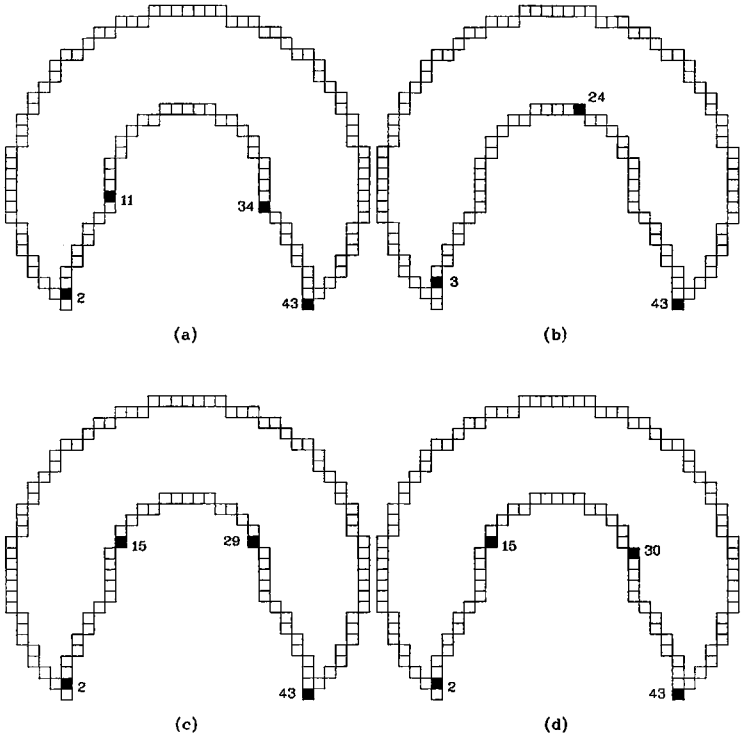
Şekil A.53 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



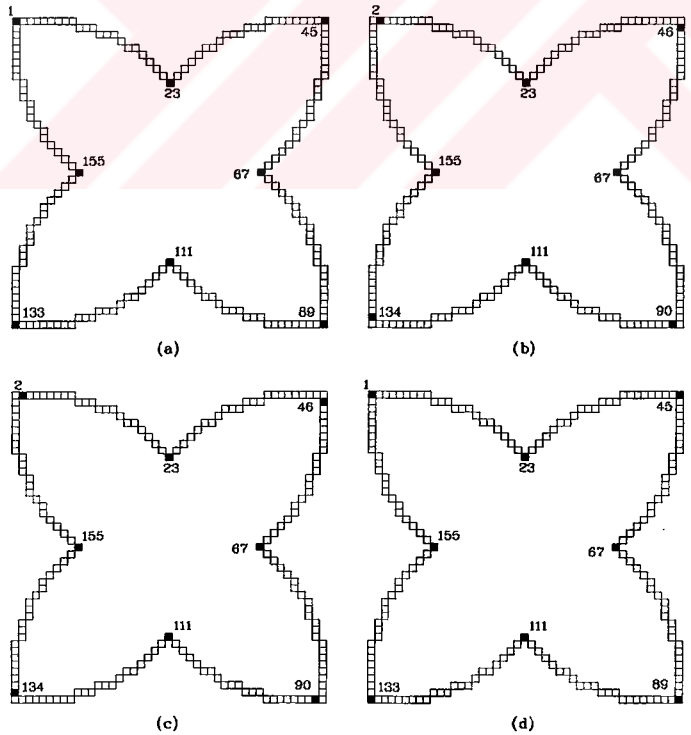
Şekil A.54 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



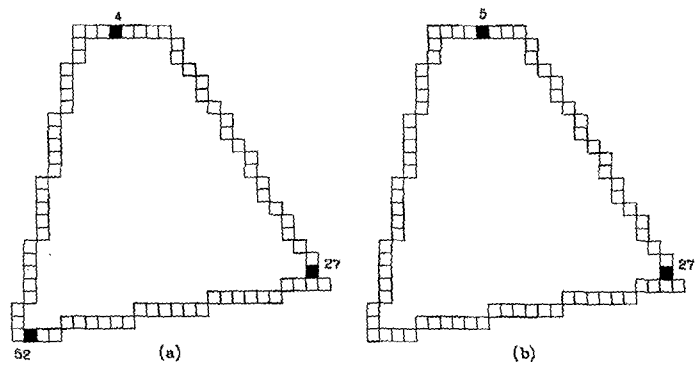
Şekil A.55 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



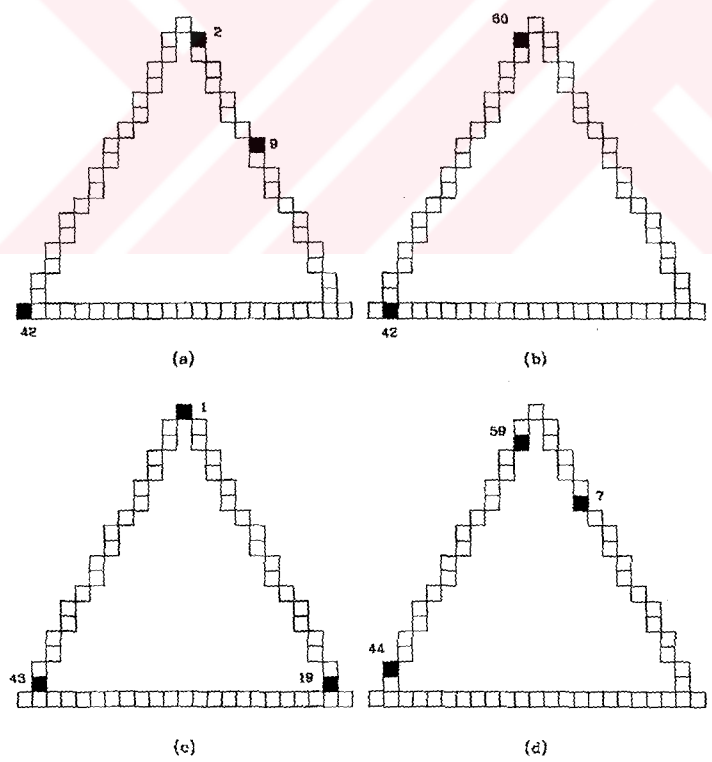
Şekil A.56 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



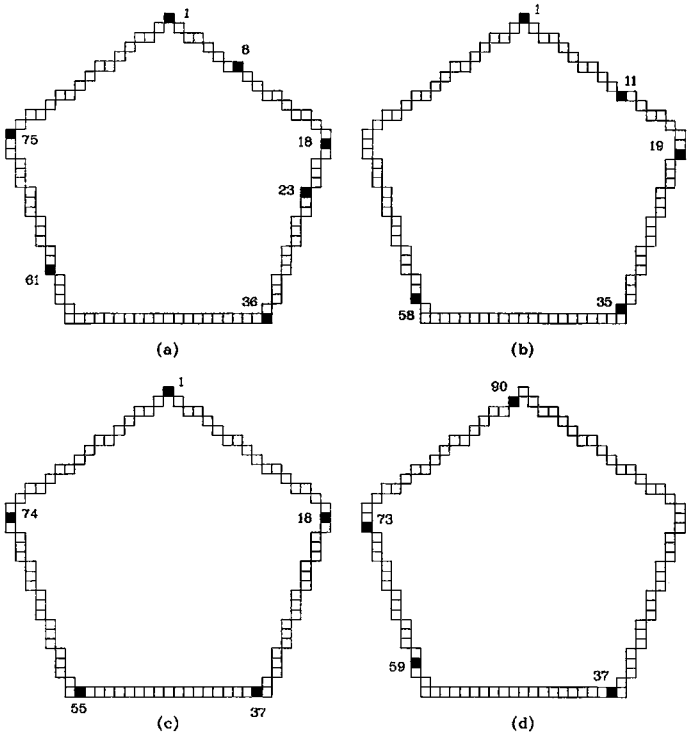
Şekil A.57 Sp3min yönteminin uygulaması. (a) k_1=4, k_2=4 (b) k_1=4, k_2=5 (c) k_1=5, k_2=4 (d) k_1=5, k_2=5



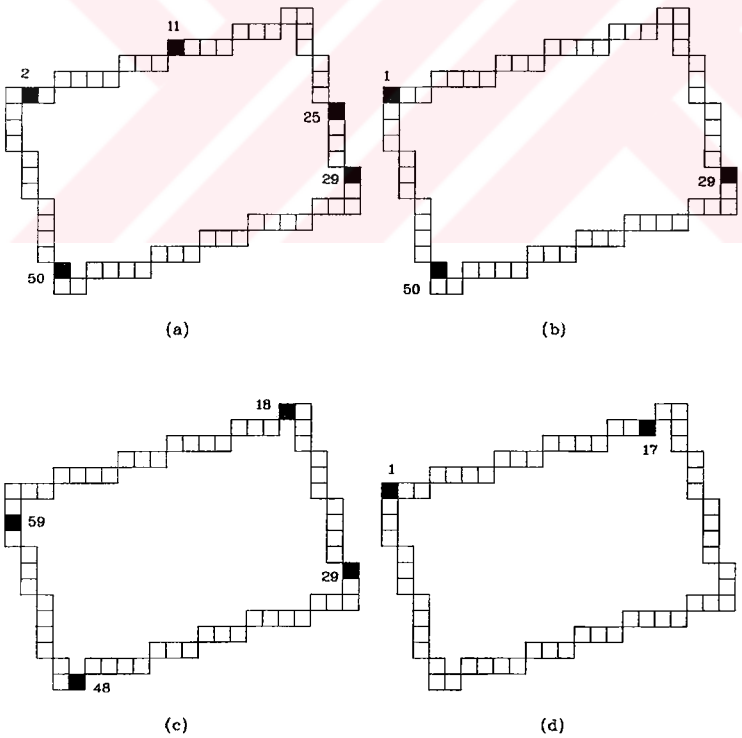
Şekil A.58 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



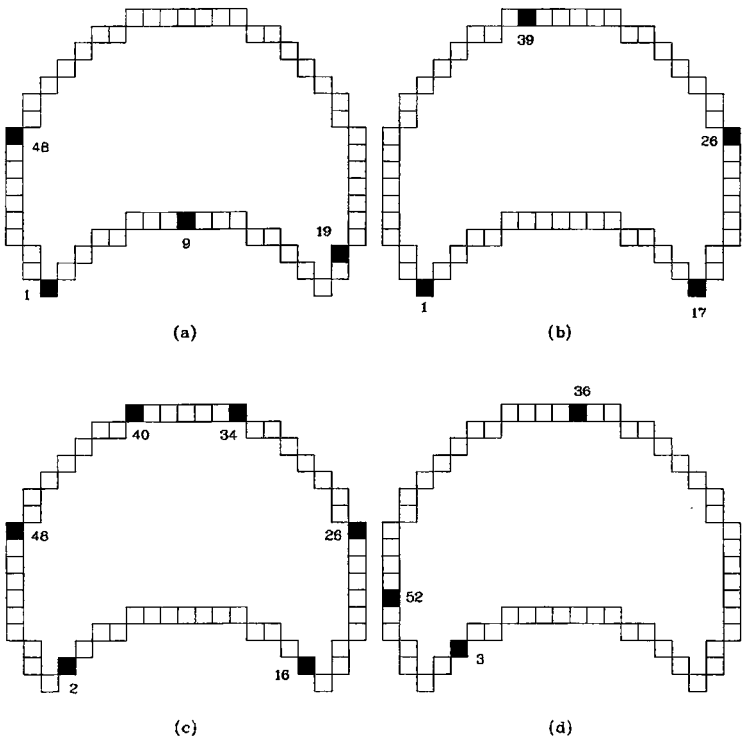
Şekil A.59 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



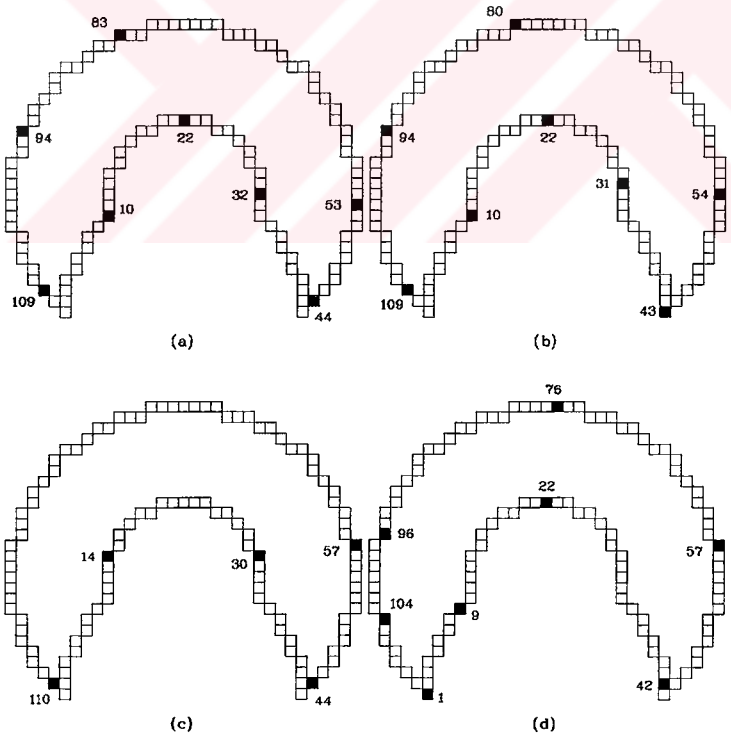
Şekil A.60 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



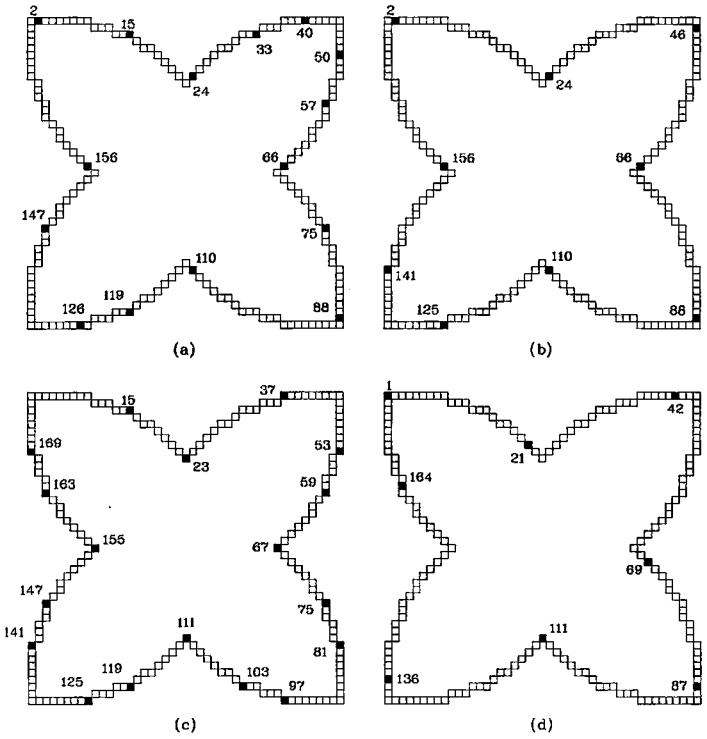
Şekil A.61 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



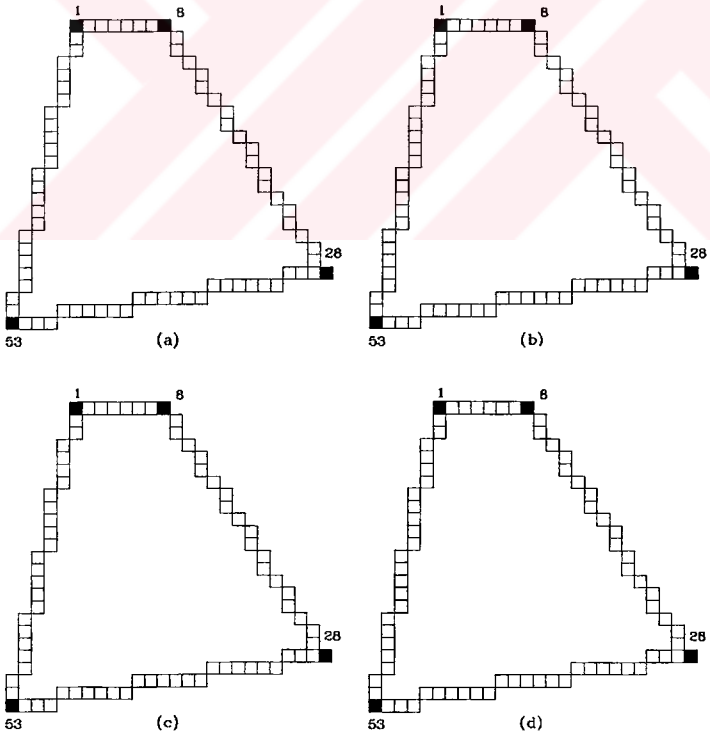
Şekil A.62 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



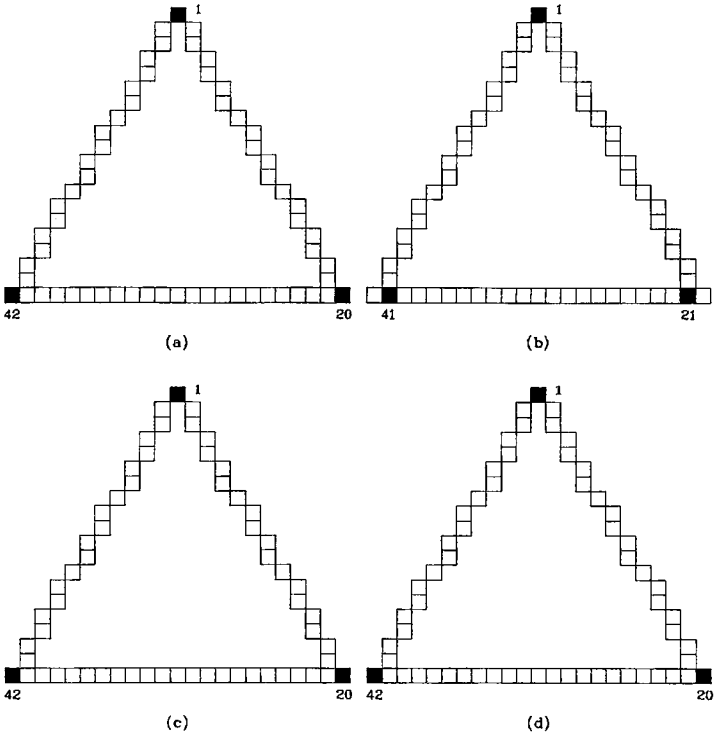
Şekil A.63 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



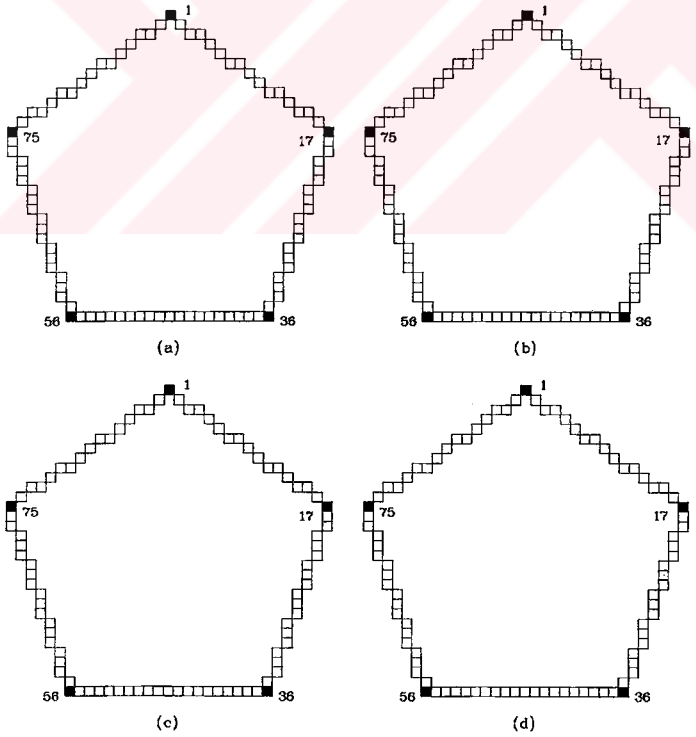
Şekil A.64 Hatmin yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



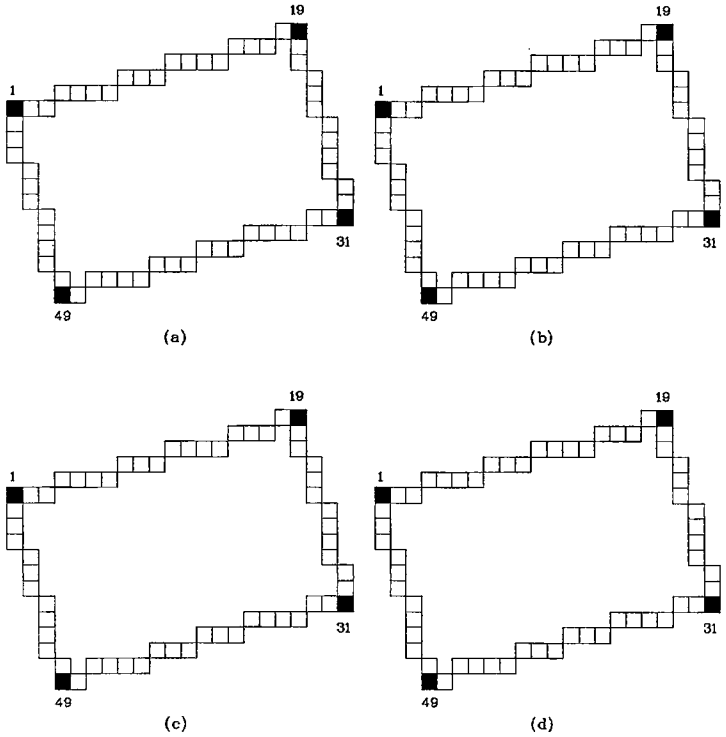
Şekil A.65 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



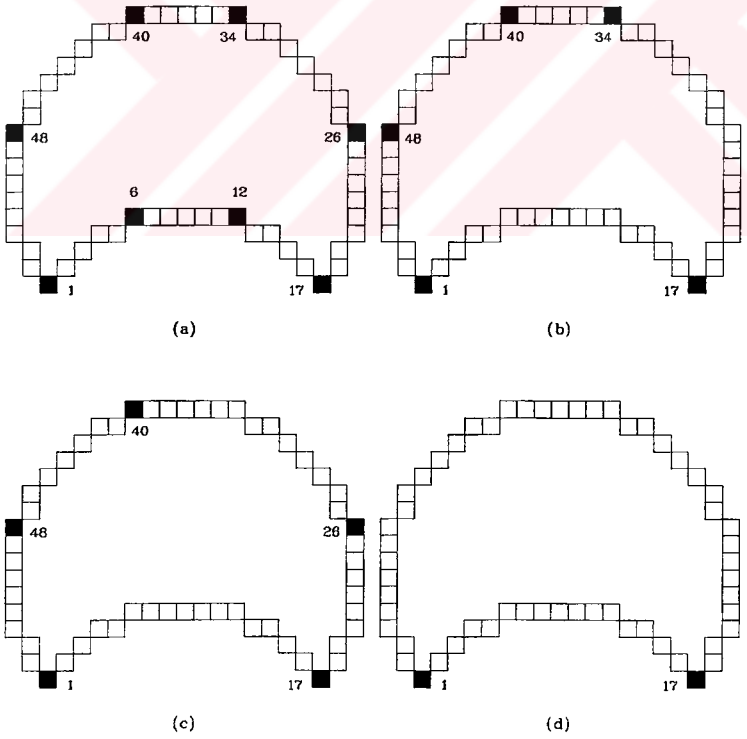
Şekil A.66 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



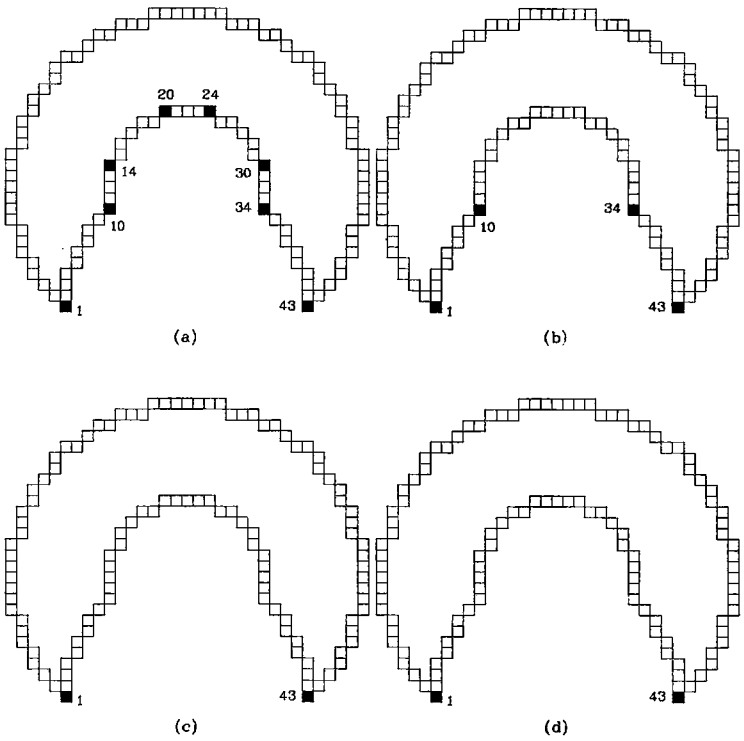
Şekil A.67 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



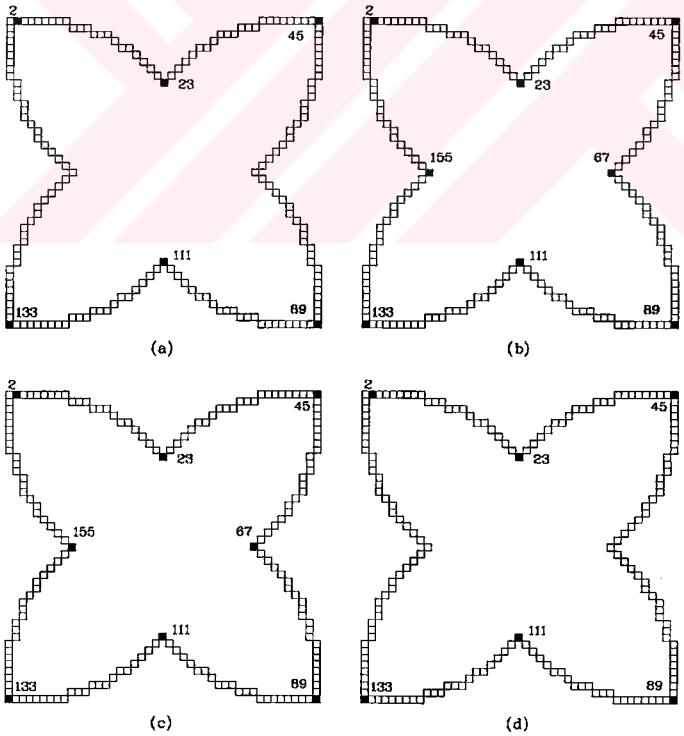
Şekil A.68 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



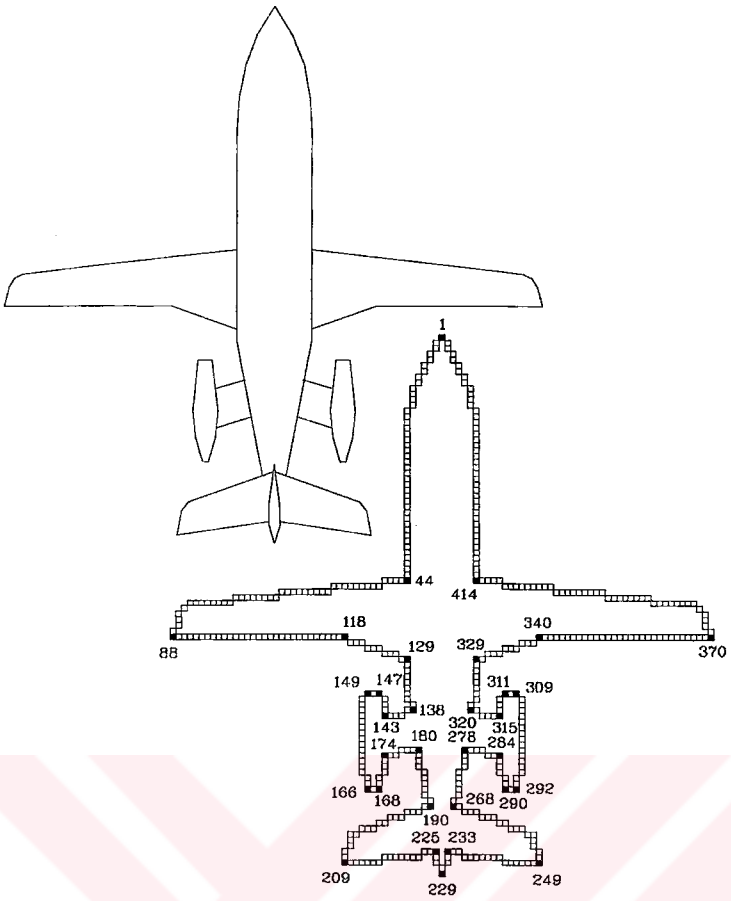
Şekil A.69 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



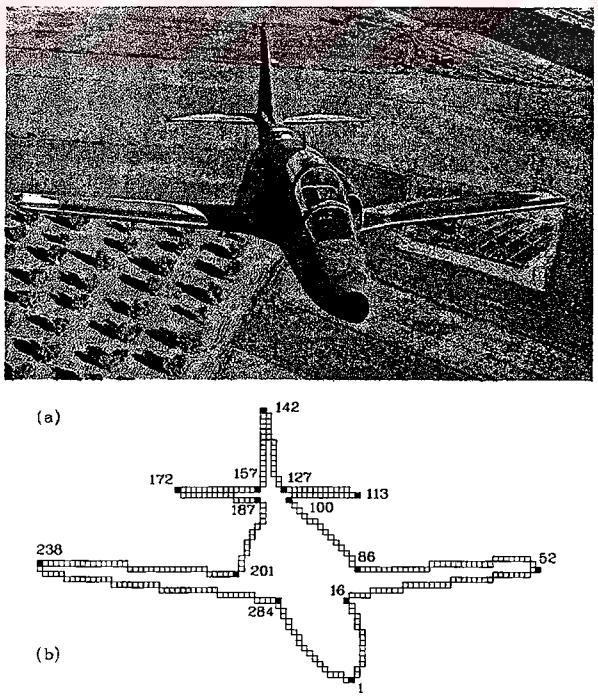
Şekil A.70 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



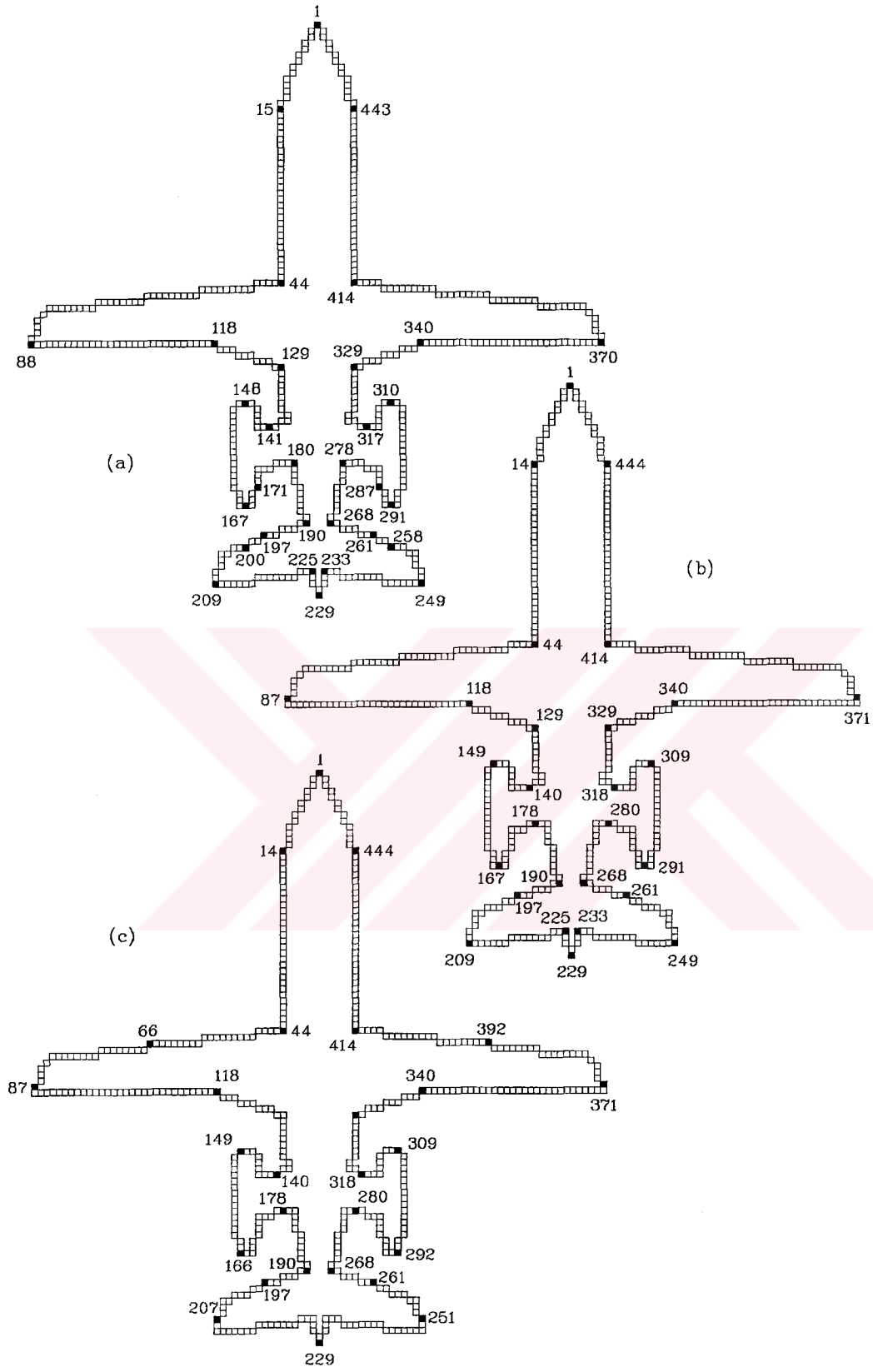
Şekil A.71 Lsq yönteminin uygulaması. (a) k=3 (b) k=4 (c) k=5 (d) k=6



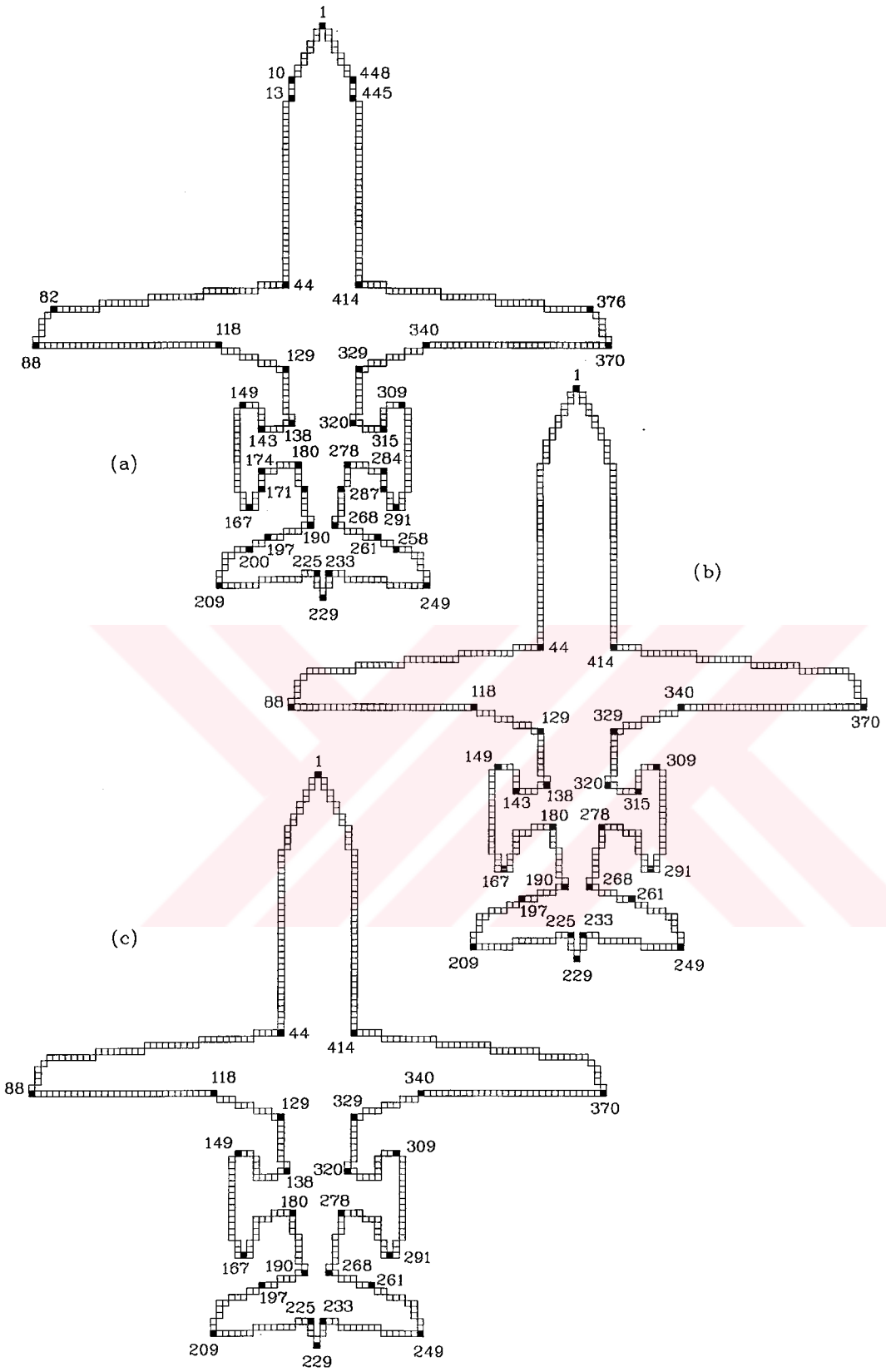
Şekil A.72 Orijinal şekil ve sayısallaştırma sonucu.



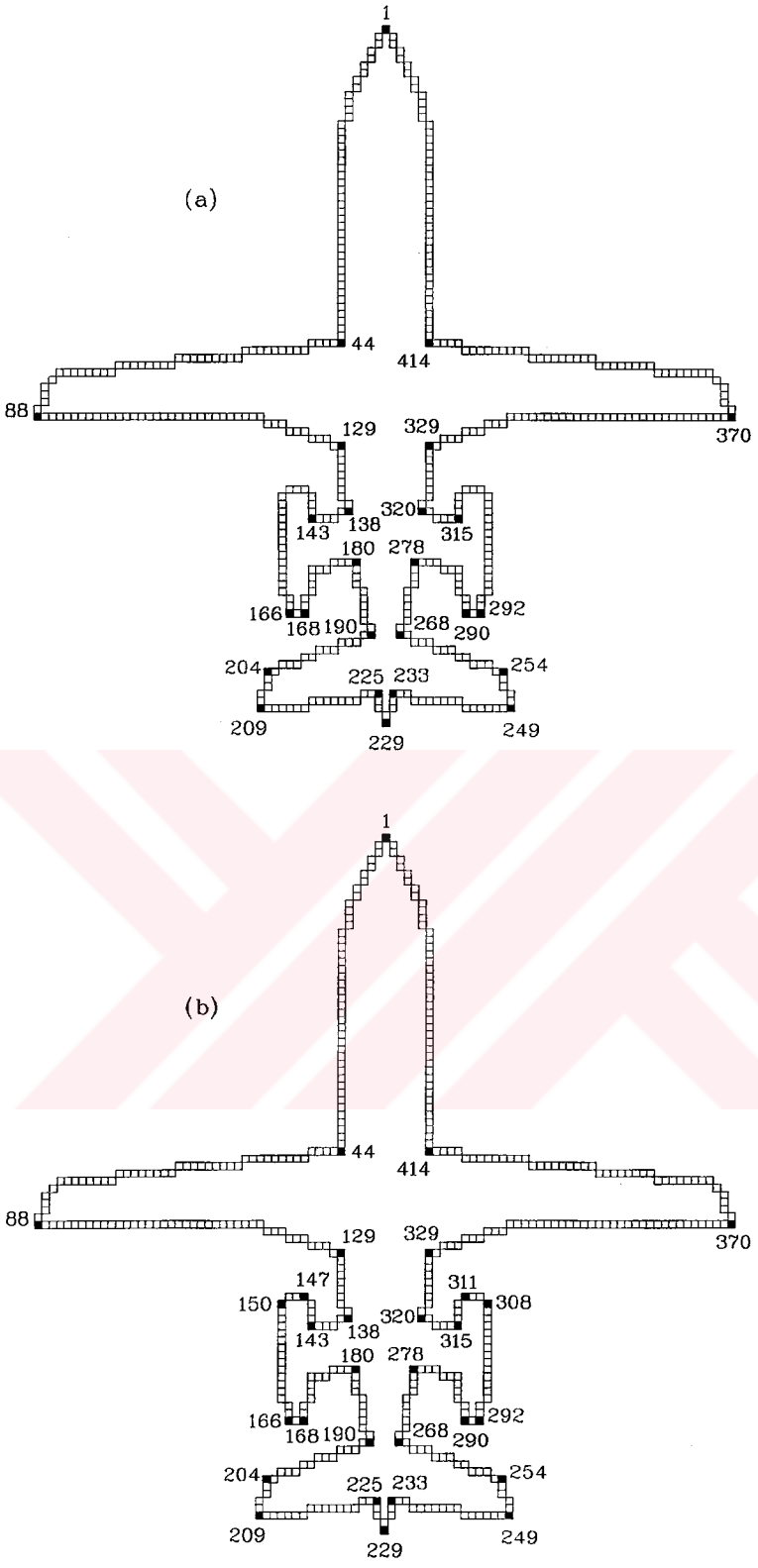
Şekil A.73 (a) Orijinal şekil (b)Sayısallaştırılmış eğri



Şekil A.74 Rosenfeld-Johnston yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9 iken bulunan köşeler.

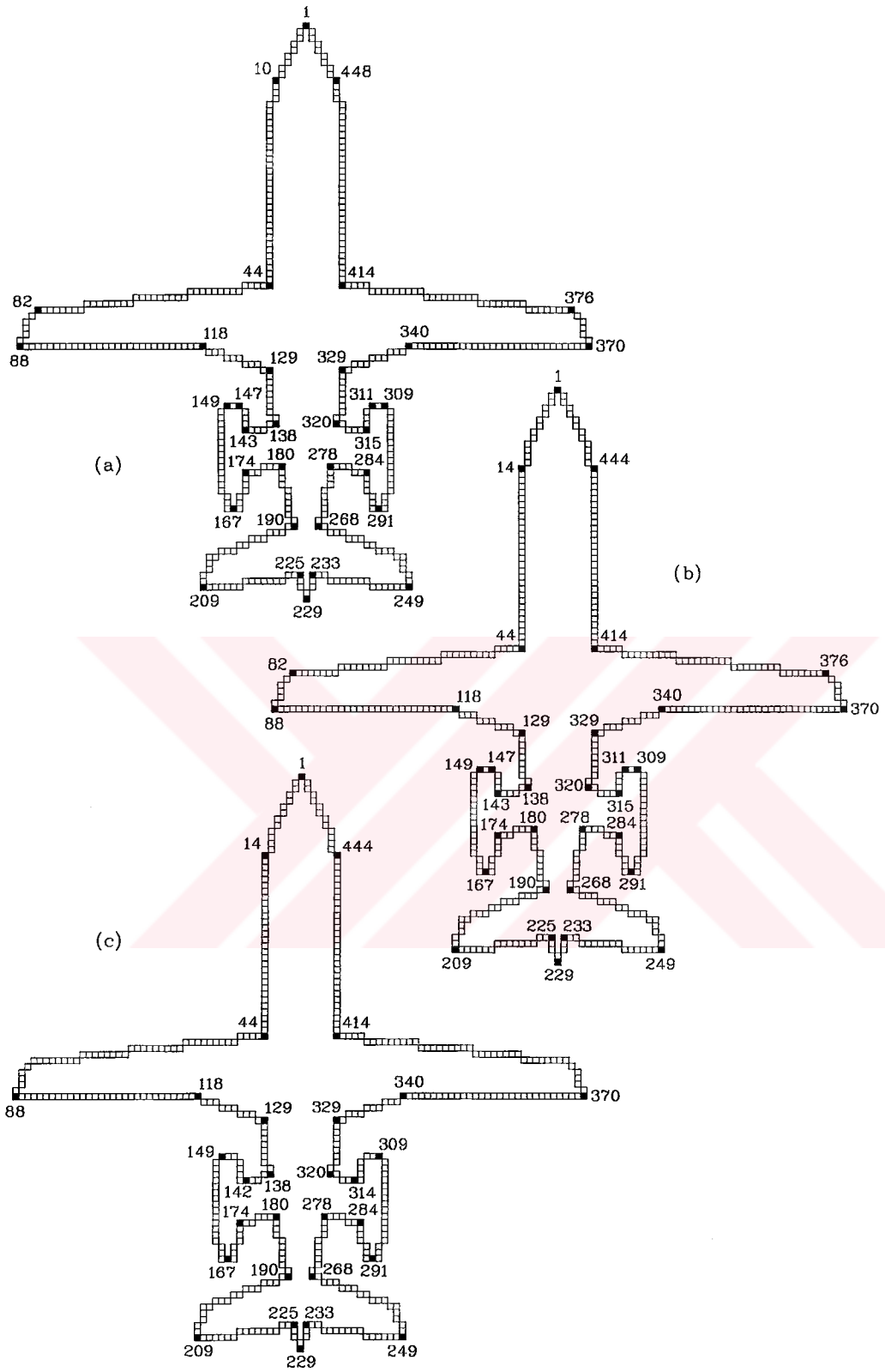


Şekil A.75 Rosenfeld-Weszka yönteminin uygulaması. (a) m=5 (b) m=7 (c) m=9 iken bulunan köşeler.

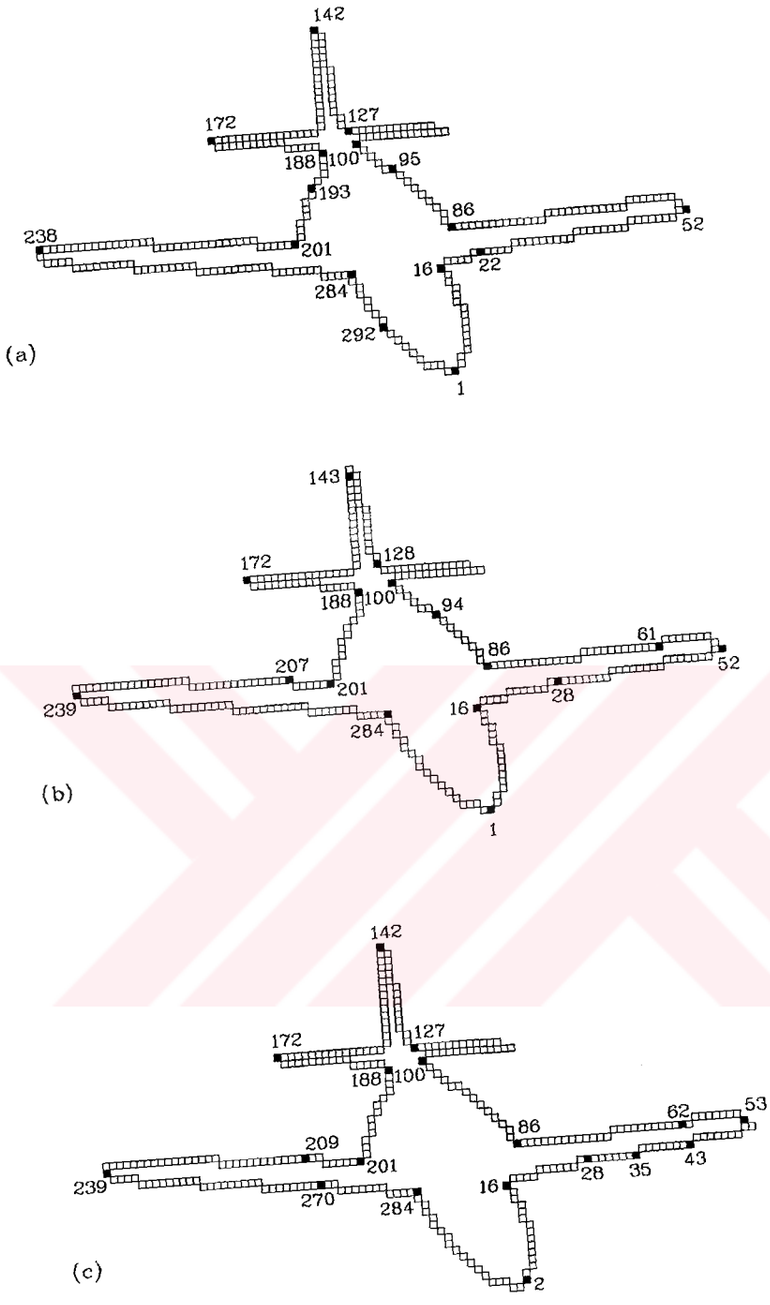


Şekil A.76 Medioni-Yasumoto yönteminin uygulaması.

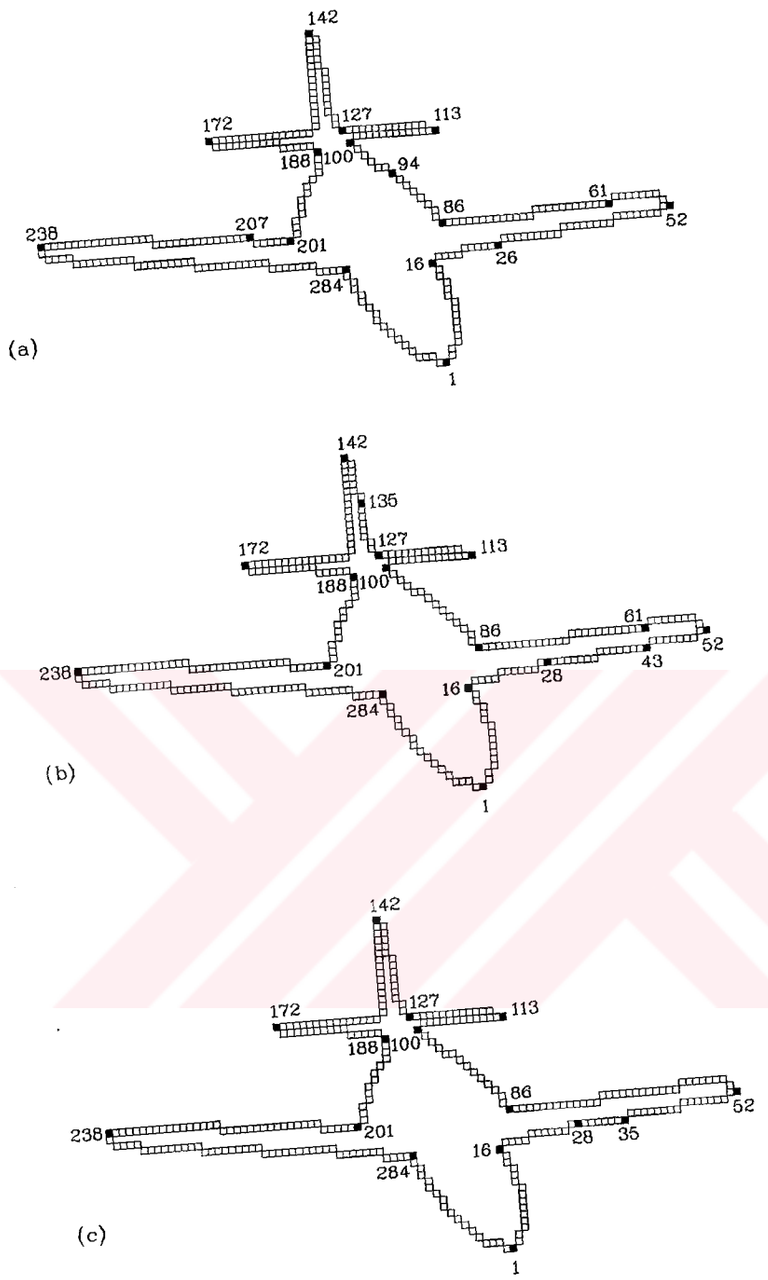
(a) c_t=1, δ_t=0.3, $|i-j|$=1 (b) c_t=0.5, δ_t=0.25, $|i-j|$=2 iken köşeler.



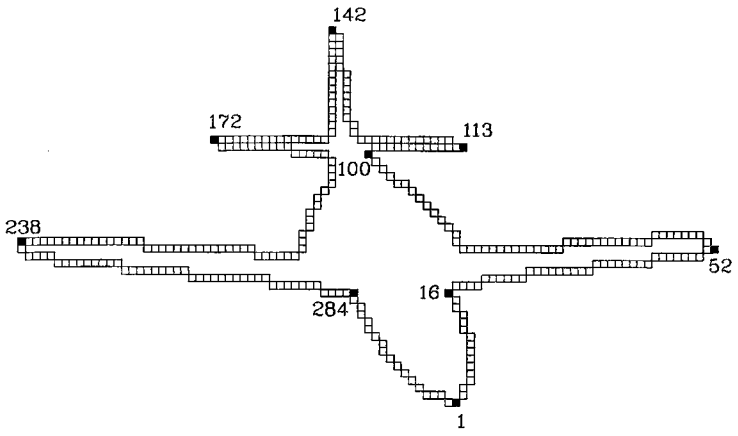
Şekil A.77 LSQ yönteminin uygulaması. (a) m=3 (b) m=4 (c) m=5 iken bulunan köşeler.



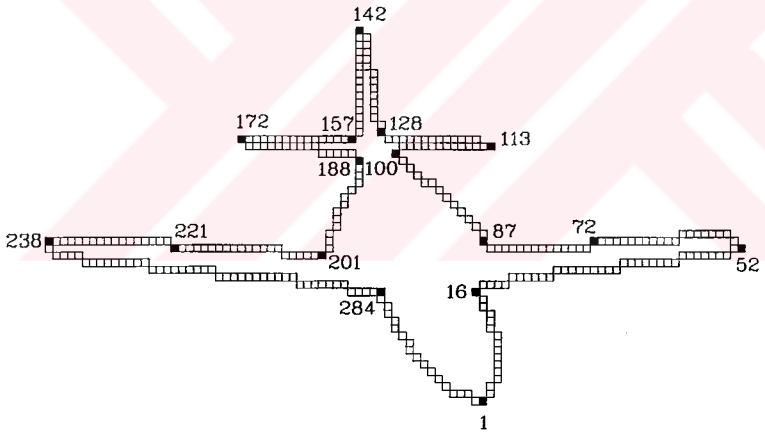
Şekil A.78 Rosenfeld-Johnston yönteminin uygulaması.
(a) m=5 (b) m=7 (c) m=9 iken bulunan köşeler.



Şekil A.79 Rosenfeld-Weszka yönteminin uygulaması.
(a) m=7 (b) m=9 (c) m=11 iken bulunan köşeler.

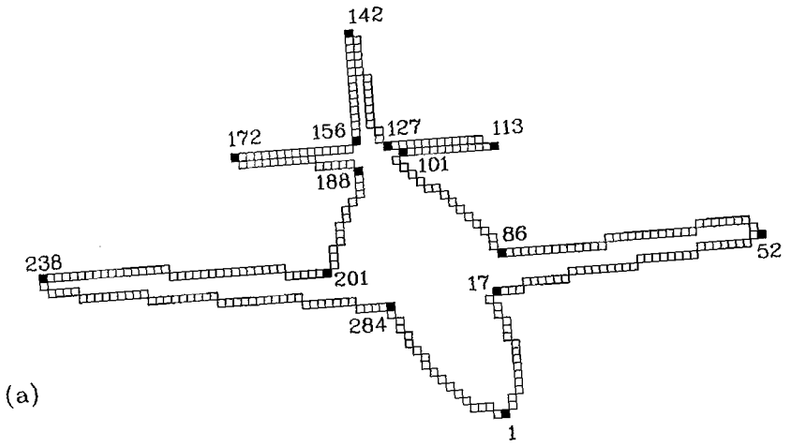


(a)

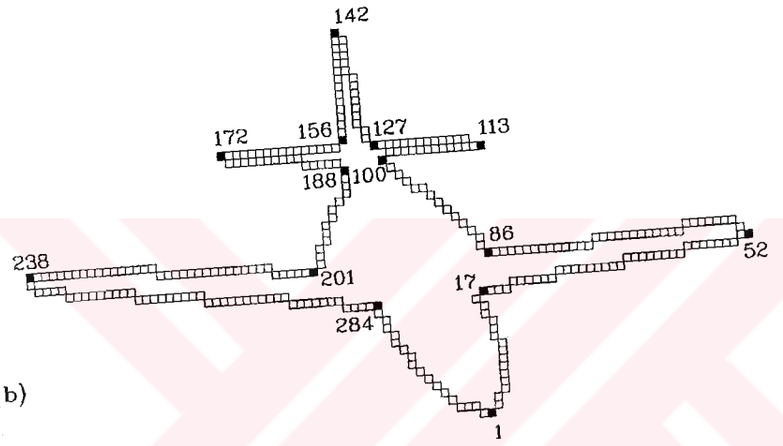


(b)

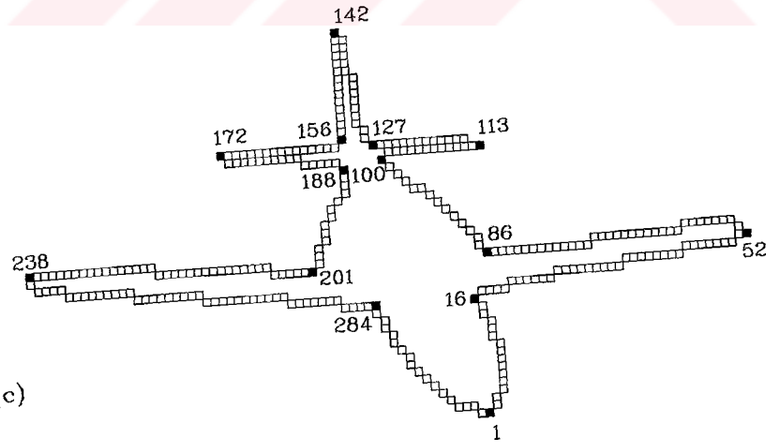
Şekil A.80 Medioni-Yasumoto yönteminin uygulaması.
(a) c_t=0.5, δ_t=0.3, $|i-j|$=2 (b) c_t=0.4, δ_t=0.25, $|i-j|$=9 iken köşeler.



(a)



(b)



(c)

Şekil A.81 LSQ yönteminin uygulaması.
(a) m=4 (b) m=5 (c) m=6 iken bulunan köşeler.

EK B

Burada karşılaştırma amacıyla kullanılan Rosenfeld-Johnston, Rosenfeld-Weszka ve Medioni-Yasumoto köşe bulma algoritmaları verilmiştir.

Rosenfeld-Johnston ve Rosenfeld-Weszka köşe bulma algoritmaları tarafından kullanılan kod listesi:

```
#include <stdio.h>
#include <math.h>
#include <dos.h>

main()
{       int i=0,sat_say,stream=1,m,n,pix,piy,k,next,prev,pnextx,pnexty;
        int pprevx,pprevy,ax,ay,bx,by,regioni=0;
        float *cosi,curvi;
        FILE *fptr,*fptr1;
        struct  time t;
          gettime(&t);
          printf("The current time is: %2d:%02d:%02d.%02d\n",
                          t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);

        fptr = fopen("p.dat","rt");fptr1 = fopen("curv.txt","a");
        fscanf(fptr,"%d\n",&sat_say);  /*ilk satir okunur*/
        while (stream)
                {i=i+1;fscanf(fptr,"%d\n",&sat_say);stream=!feof(fptr);}
        fscanf(fptr,"%d\n",&sat_say);i=i+1; /*son satir okunur*/
        fclose(fptr);
        n=i/2; /*nokta sayisi*/
if( (n/20.0-n/20) != 0.0) m = n/20 + 1; else m = n/20; /*m=7*/

        for(i=1;i<=n;i++){
          for(k=1;k<=m;k++){
            next=i+m-k+1;prev=i-m+k-1; /*on ve arkadaki nokta indisleri*/
            if(next>n) next=next-n; /*on indis nokta sayisini gecerse*/
            if(prev<1) prev=prev+n; /*arka indis ilk noktayi atlarsa*/
            pix=okux(i);piy=okuy(i); /*i. noktanin koordinatlari okundu*/
            pnextx=okux(next);pnexty=okuy(next); /*sonraki nokta koor. oku.*/
            pprevx=okux(prev);pprevy=okuy(prev); /*onceki nokta koor. oku.*/
            ax=pix-pnextx;ay=piy-pnexty; /*A vektoru olustu*/
            bx=pix-pprevx;by=piy-pprevy; /*B vektoru olustu*/
            /*A ve B vektorleri arasindaki aci bulunur*/
            *(cosi+m-k)=(ax*bx+ay*by)/sqrt((ax*ax+ay*ay)*(bx*bx+by*by));
                          }/*** k<=m ***/
          /* yeni cosi 'ler (ncosi) hesap edilecek (Rosenfeld-Weszka için lazım) */
          for(k=2;k<=m;k++){
            if( (k/2.0-k/2) == 0.0 ) /* tek cift testi */
              {sum = 0.0;
               for(j=k/2;j<=k;j++) sum = sum + *(cosi+j-1);
               *(ncosi+k-2) = (2.0/(k+2.0)) * sum;
              }/* cift olma durumunda */
```

```
            else
              {sum = 0.0;
               for(j=(k-1)/2;j<=k;j++) sum = sum + *(cosi+j-1);
               *(ncosi+k-2) = (2.0/(k+3.0)) * sum;
              }/* tek olma durumunda */
                              }/* yeni cosi 'lerin hesabi bitirildi */

           /*cos(i,m)<cos(i,m-1)<cos(i,m-2)<...<cos(i,h(i))>=cos(i,h(i-1))
             siralamasi yapilacak ve egrilik ile bolge tesbit edilecek*/
           for(k=1;k<m;k++){
             if(*(cosi+m-k)>=*(cosi+m-k-1)){
                    curvi=*(cosi+m-k);
                    regioni=m-k+1;break;     }
                              }/*** k<m ***/
           if(regioni==0){curvi=*cosi;regioni=1;}

           /*bulunan egrilik ve bolge degerleri ilgili dosyaya yazilacak*/
           fprintf(fptr1,"%f\t%d\n",curvi,regioni);
           regioni=0;
                           }/*** i<=n ***/
         fclose(fptr1);/* "curv.txt" dosyasi */

           gettime(&t);
           printf("The current time is: %2d:%02d:%02d.%02d\n",
                           t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
}
/*********Verilen nokta numarasina ait x koordinatini okur**********/
okux (int numara)
{
        int i,pnx;
        FILE *stream;

        stream = fopen("p.dat","rt");
        for(i=1;i<=2*numara-1;i++) fscanf(stream,"%d\n",&pnx);
        fclose(stream);
        return(pnx);
}/*********************************************************************/
/*********Verilen nokta numarasina ait y koordinatini okur**********/
okuy (int numara)
{
        int i,pny;
        FILE *stream;

        stream = fopen("p.dat","rt");
        for(i=1;i<=2*numara;i++) fscanf(stream,"%d\n",&pny);
        fclose(stream);
        return(pny);
}/*********************************************************************/
```

Medioni-Yasumoto köşe bulma yöntemi için yazılan kod listesi:

```
#include <stdio.h>
#include <math.h>

main()
{       int i=0,sat_say,stream=1,n,pix,piy,on,ikion,son,ikison;
        int ponx,pony,pikionx,pikiony,psonx,psony,pikisonx,pikisony;
        float b1,b2,c1,c2,alt,cv,deltatx,deltaty,deltat;
        FILE *fptr,*fptr1;
```

```
        fptr = fopen("p.dat","rt");fptr1 = fopen("curv.txt","a");
        fscanf(fptr,"%d\n",&sat_say);  /*ilk satir okunur*/
        while (stream)
                {i=i+1;fscanf(fptr,"%d\n",&sat_say);stream=!feof(fptr);}
        fscanf(fptr,"%d\n",&sat_say);i=i+1; /*son satir okunur*/
        fclose(fptr);
        n=i/2; /*nokta sayisi*/

        for(i=1;i<=n;i++){
          on=i-1;ikion=i-2;son=i+1;ikison=i+2; /*nokta indisleri tesbiti*/
          if(on<1) on=n+on; /*on indis sifir ve negatif olmasin*/
          if(ikion<1) ikion=ikion+n; /*ikion indis icin duzenleme*/
          if(son>n) son=son-n; /*son indis n' den buyuk olmasin*/
          if(ikison>n) ikison=ikison-n; /*ikison indis icin duzenleme*/
          pix=okux(i);piy=okuy(i); /*i. noktanin koordinatlari okundu*/
          ponx=okux(on);pony=okuy(on); /*i-1. nokta koordinatlari*/
          pikionx=okux(ikion);pikiony=okuy(ikion); /*i-2. nokta koord.*/
          psonx=okux(son);psony=okuy(son); /*i+1. nokta koordinatlari*/
          pikisonx=okux(ikison);pikisony=okuy(ikison); /*i+2. nokta koord.*/
          b1=pikionx/12.0+ponx/6.0-pix/2.0+psonx/6.0+psonx/12.0;
          b2=pikiony/12.0+pony/6.0-piy/2.0+psony/6.0+psony/12.0;
          c1=(psonx-ponx)/3.0+(pikisonx-pikionx)/12.0;
          c2=(psony-pony)/3.0+(pikisony-pikiony)/12.0;
          alt=c1*c1+c2*c2;
          cv=2.0*(c1*b2-c2*b1)/sqrt(alt*alt*alt);/* curvature */

    deltatx=pikionx/36.0+2.0*ponx/9.0-pix/2.0+2.0*psonx/9.0+pikisonx/36.0;
    deltaty=pikiony/36.0+2.0*pony/9.0-piy/2.0+2.0*psony/9.0+pikisony/36.0;
    deltat=sqrt(deltatx*deltatx+deltaty*deltaty);/* displacement */
          /*bulunan egrilik ve oteleme degerleri ilgili dosyaya yazilacak*/
          fprintf(fptr1,"%f\t%f\n",cv,deltat);
                        }/*** i<=n ***/

        fclose(fptr1);/* "curv.txt" dosyasi */
}
/*********Verilen nokta numarasina ait x koordinatini okur**********/
okux (int numara)
{
        int i,pnx;
        FILE *stream;

        stream = fopen("p.dat","rt");
        for(i=1;i<=2*numara-1;i++) fscanf(stream,"%d\n",&pnx);
        fclose(stream);
        return(pnx);
}/*********************************************************************/
/*********Verilen nokta numarasina ait y koordinatini okur**********/
okuy (int numara)
{
        int i,pny;
        FILE *stream;

        stream = fopen("p.dat","rt");
        for(i=1;i<=2*numara;i++) fscanf(stream,"%d\n",&pny);
        fclose(stream);
        return(pny);
}/*********************************************************************/
```

ÖZGEÇMİŞ

Barbaros Soyer, 1967 Ankara doğumludur. 1984 yılında Çorum Lisesinden mezun olmuş ve 1984-1988 yıllarında İ.T.Ü. Uçak ve Uzay Bil. Fak. Uçak Mühendisliği Bölümünde lisans eğitimini görmüştür. 1988-1990 yılları arasında O.D.T.Ü. Fen Bilimleri Enstitüsü Havacılık Bölümünde yüksek lisans eğitimine devam etmiştir. 1990 yılında İ.T.Ü. Fen Bil. Ens. Uçak Müh. Bölümüne yatay geçiş yapmış ve 1990-1992 tarihleri arasında Doç. Dr. Temel Kotil' in danışmanlığında, titreşim konusunda çalışarak yüksek lisans eğitimini tamamlamıştır. 1993 tarihinden bu yana İ.T.Ü. Fen Bil. Ens. Uçak Müh. Bölümünde Doç. Dr. Temel Kotil ile doktora çalışmasına devam etmektedir.