İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HAVA FOTOĞRAFLARINDAN YARI OTOMATİK OLARAK ÇİZGİSEL DETAYLARIN BELİRLENMESİ

DOKTORA TEZİ

Y. Müh. Oktay EKER

Anabilim Dalı : JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ

Programı : JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ

OCAK 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HAVA FOTOĞRAFLARINDAN YARI OTOMATİK OLARAK ÇİZGİSEL DETAYLARIN BELİRLENMESİ

DOKTORA TEZİ

Y. Müh. Oktay EKER
(501002300)

Tezin Enstitüye Verildiği Tarih: 8 Ağustos 2005
Tezin Savunulduğu Tarih: 18 Ocak 2006

Tez Danışmanı :	**Prof. Dr. Dursun Zafer ŞEKER**
Diğer Jüri Üyeleri	**Prof. Dr. Ahmet YAŞAYAN (Y.T.Ü)**
	Prof. Dr. Sıtkı KÜLÜR (İ.T.Ü)
	Doç. Dr. Fatma Gül BATUK (Y.T.Ü)
	Doç. Dr. Nebiye MUSAOĞLU (İ.T.Ü)

OCAK 2006

ÖNSÖZ

Hava fotoğraflarından yarı otomatik olarak çizgisel detayların belirlenmesi konulu bu doktora çalışmasının her aşamasında bana yardım ve desteğini esirgemeyen, tez yürütücülüğümü üstlenen sayın hocam Prof. Dr. Dursun Zafer ŞEKER başta olmak üzere, Prof. Dr. Ahmet YAŞAYAN'a ve Prof. Dr. Sıtkı KÜLÜR'e teşekkürü bir borç bilirim.

Bu çalışma boyunca sıkıntıya düştüğüm zamanlarda bana yardımcı olan, düşünceleriyle ve bilgileriyle tartışma ortamı yaratıp, doğrulara ulaşılmasını sağlayan mesai arkadaşlarıma da sonsuz teşekkürlerimi sunarım.

Çalışmam sırasında gece gündüz sürekli yanımda olan eşim Nilüfer ve kızım Elif İdil'e bana katlandıkları için ne kadar teşekkür etsem azdır, onların destekleri olmasaydı ben bu cümleleri yazıyor olamazdım.

Ağustos 2005 Oktay EKER

İÇİNDEKİLER

KISALTMALAR

AKM	: Aktif Kontur Modeller (snakes)
DYO	: Duda Yol Operatörü
OYİ	: Otomatik Yol İzleyicisi
CBS	: Coğrafi Bilgi Sistemi
CVT	: Coğrafi Veri Tabanı
SYM	: Sayısal Yükseklik Modeli

TABLO LİSTESİ

ŞEKİL LİSTESİ

SEMBOL LİSTESİ

$\mathbf{D}$	: İleri ve geri operatörler
$\mathbf{E}$	: Bir noktaya etkiyen enerjilerin toplamı
$\mathbf{E}_{ext}$, $\mathbf{E}_{int}$, $\mathbf{E}_{img}$	: Dış, iç ve görüntü enerjileri
$\mathbf{F}$	: Bir noktaya etkiyen bütün kuvvetlerin toplamı
$\vec{\boldsymbol{v}}$	: İki boyutlu kıvrım fonksiyonu
$\mathbf{P}$	: Yüksek değerli bir fonksiyon
λ	: İç ve görüntü enerjileri arasındaki fark
$\mathbf{K}$	: Beşgen diyagonal matris
Δ	: Yer değiştirme miktarı
ϕ	: Düzey kümesi fonksiyonu
Γ	: Başlangıç eğrisi
$\mathbf{n}$	: Normal vektör
$\mathbf{u}$	: değişken (grid noktası)
∇	: Görüntünün bir noktadaki gradyeni
$\boldsymbol{K}$	: Eğrilik

HAVA FOTOĞRAFLARINDAN YARI OTOMATİK OLARAK ÇİZGİSEL DETAYLARIN BELİRLENMESİ

ÖZET

Hava fotoğrafı ve uydu görüntülerindeki veriler, çok uzun zamandır klasik yollarla ve operatörler tarafından elle çizilmekteydi. Bilgisayar teknolojisi ve dijital görüntü işleme alanlarındaki gelişmeler, günümüzde bu işlemlerin otomatikleşmesine olanak sağlamaktadır. Otomatikleşmenin hedefi hızı arttırmak ve değerlendirme masraflarını azaltmaktır. Bu kapsamda yapılan araştırma çalışmaları, öncelikle, yüzey kaplaması, geometrik şekil, genişlik gibi karakteristik özelliklere sahip olan ve modellenebilen, yollar gibi detayların, dijital görüntülerden otomatik ve yarı otomatik olarak çizilmesi üzerine yoğunlaşmıştır.

Gerçekleştirilen çalışmalar, otomatik ve yarı otomatik yol çiziminde, kullanılan dijital görüntünün mekansal ayırma gücünün çok önemli bir rolü olduğunu göstermiştir. Ortaya konan yöntemler, kullanılan dijital görüntünün çözünürlüğüne göre; düşük, yüksek ve her ikisinin birlikte kullanıldığı çok çözünürlükte yol çizim yöntemleri olarak sınıflandırılabilir. Bu yöntemler arasında özellikle yol izleme ve Aktif Kontur Modeller (snakes) tabanlı yöntemler öne çıkmaktadır.

Otomatik ve yarı otomatik olarak detay çizme ve görüntüler üzerinde sınıflandırma işlemlerinde kullanılan bir başka yöntem de görüntü bölümlemedir (image segmentation). Görüntü bölümleme, dijital görüntüler üzerinde detay çıkarma ve yorumlama konularında çok sık başvurulan bir yöntemdir.

Son yıllarda, görüntü üzerindeki bölümleme sonucunda elde edilen yüzeylerin görüntü üzerinde gelişmesini ve yayılmasını sağlamak için Düzey Kümesi ve Hızlı İlerleme (Level Set and Fast Marching) yöntemleri başarıyla kullanılmaktadır.

Bu çalışmada; dijital hava fotoğrafları üzerinden yollar, binalar, dereler ve göller gibi çizgisel ve alan detayların sınırlarının ve merkez hatlarının yarı otomatik olarak çizim yöntemlerinin ve ayrıca Coğrafi Bilgi Sistemlerinde (CBS) bu bilgilerin güncelleştirilmesi çalışmalarının araştırılması ve bunu sağlayacak uygun bir yöntem geliştirerek, bir bilgisayar programı yardımıyla uygulanabilirliğinin değerlendirilmesi amaçlanmıştır.

Bu amaç ve hedef doğrultusunda, renk farkları ile düzey kümesi yöntemlerini temel alan algoritmaları kullanan bir yöntem ve bu yöntemin uygulanmasına yönelik bir uygulama yazılımı tasarlanmış ve geliştirilmiştir. Söz konusu yazılım, Borland C++ ve Visual C++ görsel yazılım dilleri kullanılarak nesneye dayalı bir kodlama ile gerçekleştirilmiştir.

Geliştirilen yazılımın test edilmesi amacıyla 4 farklı ölçekte ve özelliklerde seçilmiş hava fotoğraflarıyla birlikte 1 adet IKONOS ve 1 adet SPOT uydu görüntüsü üzerinde çizgisel detayların değerlendirilmesi çalışmaları gerçekleştirilmiştir.

Bu yazılımın, harita üretiminde ve CBS için gerekli olan vektör verilerin toplanmasında kullanılabileceği düşünüldüğünden; İstanbul Teknik Üniversitesi, Ayazağa Kampüsünün bir bölümünü içeren 1:16000 ölçekli renkli hava fotoğraflarından oluşturulmuş 0.5 m mekansal ayırma gücüne sahip, iki adet 1:5000 ölçekli ortofoto görüntüden, koordinatlı olarak yol ve bina detayları geliştirilen yazılımla yarı otomatik olarak çizilmiştir. Elde edilen verilerin geometrik doğruluklarının tespit edilmesi maksadıyla aynı detaylar bir operatör yardımıyla tamamen elle çizilmiş ve bu veriler doğru olarak kabul edilerek, yarı otomatik toplanan verilerle karşılaştırılarak bir doğruluk araştırması yapılmıştır. Veriler, yollar ve binalar olmak üzere iki ayrı grupta değerlendirilmiştir. Yollar için 422 noktada, binalar için 281 noktada ölçümler gerçekleştirilmiş olup, yollar için karesel ortalama hata ±0.663 m, binalar için ise ±0.463 m olarak hesaplanmıştır.

Yapılan doğruluk araştırması sonucunda, geliştirilen yöntemin, kullanılan dijital hava fotoğrafının, uydu görüntüsünün veya raster görüntünün ±1 pikselinin boyutuna eşit olan bir hata kriterine sahip olduğu sonucuna ulaşılmıştır. Bununla birlikte; bu yöntemin fotogrametrik harita üretiminde ve CBS için fotogrametrik veri sağlanmasında yeni bir yöntem olarak kullanılabileceği değerlendirilmiştir. Özellikle: Göller, sulu dereler ve binalar gibi homojen yapıdaki detayların sınırlarına ait vektör verilerin toplanmasında çok başarılı ve etkili bir şekilde kullanılabileceği görülmüştür. İstenildiği takdirde, tolerans değerinin uygun olarak belirlenmesiyle, söz konusu detaylar üzerinde gözle ayırt edilemeyen sınıflandırmalar ve bölümlemeler gerçekleştirilebileceği tespit edilmiştir. Kaliteli yolların sınırları ve/veya merkez hatları (kullanılan fotoğrafın ölçeğine ve mekansal ayırma gücüne bağlı olarak) etkili ve hızlı bir şekilde çizilebileceği, ayrıca kırıklık toleransı değerleri değiştirilerek istenilen kırıklıkta vektör veriler elde edilebileceği sonucuna varılmıştır. Raster veriden vektör veriye dönüşümde hem sınırların hemde merkez hatların kullanılabilmesinin etkinliğe çok katkı sağlayacağı düşünülmektedir.

Test çalışmaları sırasında yazılımın olumsuz olarak etkilendiği ve eksik kaldığı bazı faktörler de tespit edilmiş olup, bunlarda aşağıda sunulmuştur:

- Özellikle büyük ölçekli görüntülerde detaylar üzerinde bulunan engeller (ağaç, araba, gölge gibi) detay çıkarma ve çizimşlemini olumsuz olarak etkilemektedir. Bu etki ölçek küçüldükçe azalmaktadır.

- Tolerans değerlerinin uygun olarak belirlenmediği durumlarda yanlış detay çizimleri gerçekleşebilmektedir. Tolerans yüksek tutulduğunda ilgilenilen detaylar atlanabilmekte, küçük tutulduğunda ise çok fazla operatör müdahalesine gerek duyulup, küçük küçük adımlarla ilerleme sağlandığından yöntemin etkinliği düşmektedir.

- Çok büyük boyutlu görüntü dosyaları kullanıldığında, tasarlanan algoritmada piksel değerleri bilgisayar hafızasına kayıt edildiğinden, çok fazla hafızaya ihtiyaç duyulacağından bazı donanımsal hatalarla karşılaşılabilinmektedir.

- Görüntülerin kalitesi, kontrastlığı ve gürültü oranları algoritmanın başarısını önemli ölçüde etkilemektedir.

- Çizgisel detayların yüzey ve kaplama özellikleri ile kontrastlıkları da algoritmanın başarısını etkilemektedir.

Söz konusu problemlerin giderilmesi ve yazılımın daha etkin bir hale getirilebilmesi için; görüntülerdeki kontrastlığın arttırılması, gürültü oranlarının azaltılması için görüntü işleme algoritmalarından faydalanılarak anisotropik difüzyon gibi filtreler ile kenar zenginleştirme algoritmalarının uygulanmasının, engellerden kaynaklanan boşlukların doldurulması için farklı enterpolasyon yöntemlerinin kullanılmasının, bölümleme işlemlerinde renk farkı yerine daha gelişmiş (AKM vb.) algoritmaların kullanılmasının ve büyük boyutlu görüntülerin bilgisayar ortamlarında daha kolay elde alınabilmesi için piramit seviyelerinin kullanılmasının faydalı olacağı değerlendirilmektedir.

SEMI-AUTOMATIC EXTRACTION OF LINE FEATURES FROM AERIAL PHOTOGRAPHS

SUMMARY

Aerial photographs and satellite imagery have been evaluated manually by the operators for a long time for the extraction of the vector data. Computer technology and digital image processing technologies have been developed and this development provides to perform these extraction processes automatically or semi-automatically. The aim of making the processes automatic is to increase the speed of collecting the data and to reduce the cost. Automatic feature extraction studies are firstly motivated to carry out the extraction of roads from digital images because roads have characteristic attributes like width, surface type and geometrical shape which can be modeled more easily than the others.

These studies are showed that the resolution of the images has a very important role in the automatic and semi-automatic extraction of the roads. The developed methods can be classified in three groups: road extraction in low resolution, road extraction in high resolution and multi-resolution road extraction. Most known methods are based on the road tracing and the snakes algorithms.

Another method of automatic and semi-automatic feature extraction and classification of images is the image segmentation. Image segmentation is commonly used the interpretation of the medical imagery like mr-scans.

In recent years, image segmentation and the front propagation of the segments have been carried out successfully by the Level Set and Fast Marching methods.

The goal of this study is to develop a semi-automatic extraction method of the line and area features like roads, rivers and lakes from digital aerial photographs and updating of these features in Geographic Information Systems (GIS) and also to develop application software to test the capabilities of this method.

According to this goal, a semi-automatic line extraction method, based on the segmentation of the images using color-differences of the pixels and the propagation of fronts by the Level Set algorithms, is developed. An object-oriented application software is also developed using Borland C++ and Visual C++ languages to test the capabilities of the developed method.

Several applications are made to test the software. 4 different kinds and scale aerial photographs, 1 IKONOS and 1 SPOT satellite imagery are used or semi-automatic extraction of lines and the boundaries of area features.

It is thought that this software can be used in producing maps and in collecting the vector data for GIS. Another application with 1:5000 scaled two Roth images which

have 0.5m resolution of Ayazağa Campus of Istanbul Technical University. These ortho images are generated from 1: 16000 scaled color aerial photographs. In this test area, an accuracy test is also carried out to find the accuracy of the developed method. In this accuracy test, vector data of roads and buildings are collected semi-automatically with the developed software and also manually with an experienced operator. The data collected by the operator are assumed the correct ones and they are compared with the others collected by the software. The accuracy test is carried out in two groups. In the first group, roads are used. On 422 road check points, measurements are made and the found square mean root of this group is ±0.663m. In the second group, buildings are used and 281 check points are measured and the square mean root of this group is equal to ±0.463m

As the results of the applications and tests, it can be said that the accuracy of this developed method is ±1 pixel size of the used imagery. It can be used correctly for producing maps and collecting vector data for GIS. Especially for lakes, rivers and buildings can be collected very efficiently. Different classifications and segmentations, which an operator's can not see, can be made also with the adjusting of the tolerance value. Roads which have good quality can be vectorized from their center lines and/or boundaries according to the scale of the image used.

Some weak sides of this developed method and software are also found out. These are:

- Especially on big scale aerial photographs, the obstacles on the features, as trees, cars and shadows, effects the extraction of the features negatively. Effects of this factor are reduced whether the scale of the image gets smaller.

- If the tolerance value is not be adjusted to the correct values, wrong features can be extracted.

- When a big size image is used, the software gives back some errors because the size of the arrays is directly proportional to the number of the pixels.

- The quality, contrast and noise of the image affect the feature extraction process.

- The surface attributes of the features also affect the success degree of the feature extraction.

If the noise and the contrast of the images are eliminated by the image process algorithms like edge detection algorithms and filters as anisotropic diffusion and the blanks that are generated by the obstacles on the feature can be interpolated by the different kinds of interpolation methods, more good results can be achieved by the developed method and the software. Also, for the image segmentation different types of segmentation like snakes, instead of color difference and for big size images pyramid levels can be used to increase the success degree of this method.

1. GİRİŞ VE ÇALIŞMANIN AMACI

Hava fotoğrafı ve uydu görüntüleri; binalar, yollar ve köprüler gibi insan yapısı objeler, bitki örtüsünün karakteristiği ve konumu gibi yeryüzünün şekli hakkında bir çok bilgi sunmaktadır. Hava fotoğrafı ve uydu görüntüleri olmaksızın bu bilginin toplanması ve güncellenmesi çok pahalı ve zaman alıcı bir işlemdir.

Hava fotoğrafı ve uydu görüntülerindeki veriler, çok uzun zamandır klasik yollarla ve operatörler tarafından elle tespit edilmekteydi. Bilgisayar teknolojisi ve dijital görüntü işleme alanlarındaki gelişmeler, günümüzde bu işlemlerin otomatikleşmesine olanak sağlamaktadır. Otomatikleşmenin hedefi hızı arttırmak ve değerlendirme masraflarını azaltmaktır. Bu kapsamda yapılan araştırma çalışmaları, öncelikle binalar ve yolların, dijital görüntülerden otomatik ve yarı otomatik olarak çizilmesi üzerine yoğunlaşmıştır. Yollar ve binaların, yüzey kaplaması, geometrik şekil, genişlik gibi karakteristik özelliklere sahip olması bu detayların tanımlanabilmesi ve belirlenebilmesini diğer detaylara göre daha kolay bir hale getirmektedir.

Bu çalışmada; dijital hava fotoğrafları üzerinden yollar, binalar, dereler ve göller gibi çizgisel ve alan detay sınırlarının ve merkez hatlarının yarı otomatik olarak çizim yöntemlerinin ve ayrıca Coğrafi Bilgi Sistemlerinde (CBS) bu bilgilerin güncelleştirilmesi çalışmalarının araştırılması ve bunu sağlayacak uygun bir yöntem geliştirerek, bir bilgisayar programı yardımıyla uygulanabilirliğinin değerlendirilmesi amaçlanmıştır.

Dijital fotoğraflardan otomatik ve yarı otomatik olarak detay çizme konusunda gerçekleştirilen çalışmalarda öncelikle karakteristik özellikleriyle modellenebilir detaylar ele alınmıştır. Söz konusu detayların başında yol detayı gelmektedir. Bu alanda gerçekleştirilen çalışmalar, dijital fotoğraflardan detay çizme işleminin kullanılan fotoğrafın mekansal ayırma gücüne doğrudan bağlı olduğunu göstermiştir. İkinci bölümde; bu konuda yapılan çalışmalar, kullanılan dijital görüntünün mekansal ayırma

gücüne göre, düşük, yüksek ve her ikisinin birlikte kullanıldığı çok çözünürlükte otomatik ve yarı otomatik yol çizimyöntemleri olarak sınıflandırılmış ve herbir yöntem hakkında bilgi verilerek birbirleriyle karşılaştırılmıştır.

Çalışmanın üçüncü bölümünde ise otomatik ve yarı otomatik detay çizme yaklaşımlarında çok önemli bir yeri olan ve yeterince Türkçe kaynak bulunmayan Aktif Kontur Modelleri (snakes) temel alan algoritmalar hakkında bilgi verilerek bu eksiklik biraz olsun giderilmeye çalışılmıştır.

Otomatik ve yarı otomatik olarak detay çizme araştırmalarında kullanılan başka bir yöntem de görüntü bölümlemesidir (image segmentation). Özellikle görüntüler üzerindeki sınıflandırma işlemlerinde ve dijital görüntülerin yorumlanmasında oldukça sık başvurulan ve bu çalışmada geliştirilen uygulamanın temellerinden birisini oluşturan, görüntü bölümleme algoritmaları dördüncü bölümde geniş olarak incelenmiştir.

Son yıllarda, görüntü üzerindeki bölümleme sonucunda elde edilen yüzeylerin görüntü üzerinde gelişmesini ve yayılmasını sağlamak için Düzey Kümesi ve Hızlı İlerleme (Level Set and Fast Marching) yöntemleri kullanılmaktadır. Bu yöntemler aynı zamanda bu araştırma çalışması sonucunda geliştirilen yöntemde ve uygulama yazılımında da başarıyla kullanılmıştır. Düzey Kümesi ve Hızlı İlerleme yöntemleri ve bu yöntemlerin matematiksel eşitlikleri beşinci bölümde ayrıntılarıyla ele alınmıştır.

Altıncı bölümde coğrafi veri kavramı, coğrafi veri kaynakları ve coğrafi veri toplama yöntemleri tanıtılmış ve geliştirilen yöntemin CBS için bir veri toplama ve güncelleştirme yöntemi olarak kullanılabileceği vurgulanmıştır.

Geliştirilen yöntem ve uygulama yazılımı yedinci bölümde ayrıntılarıyla tanıtılmıştır. Yazılımın arayüzleri, kullanımı ve çeşitli ölçeklerdeki siyah/beyaz ve renkli hava fotoğrafları ile uydu görüntüleri üzerindeki test çalışmaları ve yöntemin doğruluk araştırması bu bölümde sunulmuştur.

Sekizinci ve son bölümde ise yapılan test çalışmaları sonucunda elde edilen sonuçlar ve öneriler tartışılmıştır.

2. OTOMATİK VE YARI OTOMATİK ÇİZGİSEL DETAY ÇİZİM PROBLEMİNE GENEL BİR BAKIŞ

Son yirmi yıldır, otomatik ve yarı otomatik çigisel detay çizimiyle ilgili literatürde birçok yaklaşım yayınlanmıştır. Yayınların büyük bir çoğunluğunda çizgisel detaylardan yol detayının otomatik ve yarı otomatik olarak çizimine çözüm bulunmaya çalışılmıştır. Yol detayının yüzeysel ve geometrik özellikleri bir model geliştirmek için diğer çizgisel detayların özelliklerinden daha elverişli olduğundan öncelikle ve çoğunlukla yolların otomatik ve yarı otomatik olarak çizilmesi ele alınmıştır. Bunlar birbirlerinden; amaçları, elde edilebilir bilgileri ve detaylar hakkındaki kabulleri açısından küçük farklılıklar göstermektedirler. Bu yaklaşımları sınıflandırmak ve analiz etmek için çeşitli kriterler öngörülebilir ancak yolların otomatik ve yarı otomatik olarak çiziminde en etkili iki faktör: Görüntülerin mekansal ayırma gücü ve çizim işlemini başlatıp, takip edecek operatör ihtiyacının olup olmamasıdır.

Görüntünün mekansal ayırma gücünün, yollar ve diğer objelerin tanımlanabilmesinde güçlü bir etkisi vardır. Dijital görüntülerin, sonsuz dünyanın sınırlı sayıdaki görüntü pikselleri üzerine izdüşümü olarak kabul edildiği düşünüldüğünde yüksek çözünürlüklü görüntüler üzerinde ayırt edilebilir bir çok detayın, görüntü çözünürlüğü düştükçe ayırt edilemeyeceğini söylemek yanlış olmaz. Örneğin; 0.2m çözünürlüğündeki görüntüde kolaylıkla ayırt edilebilen yol kenarları, araçlar ve hatta yol işaretleri, görüntü çözünürlüğü 2m veya daha düşük bir değer olduğunda teşhis edilememektedirler. Yüksek çözünürlüklü görüntülerde çok çeşitli yollar tanımlanabilirken, düşük çözünürlüklü görüntülerdeki yollar ise binalar gibi diğer çizgi detaylardan güçlükle ayırt edilebilecek şekilde çizgisel yapıda görünmektedirler. Bu kısıtlamalardan yol tanımlama yöntemleri oldukça etkilenmektedir.

Otomatik ve yarı otomatik yol çizimi için diğer bir önemli faktör de yol çizim algoritmaları ile insan arasındaki etkileşimin kabulüdür. Bir operatörün müdahalesi gereken algoritmalara **yarı-otomatik algoritmalar** adı verilir. Tam otomatik

algoritmaların aksine, bir operatör tarafından etkileşimli olarak seçilen **kaynak** noktalarını temel alırlar. Yarı otomatik algoritmalar bu kaynak noktalarını gerçek yola en çok benzeyen çizgi ile birleştirilerek yolun tamamının çizilmesine olanak sağlarlar (bunu gerçekleştirirken en küçük kareler yöntemi, korelasyon analizi ve kalman filtrelerini kullanırlar). Girilen bu kaynak noktaları, yarı otomatik algoritmaların problemlerini oldukça azaltır. Tam otomatik algoritmalardan farklı olarak yol bulma işlemleri ile uğraşmayan ve genelde operatör tarafından bulunan yolların nasıl izlenmesini gerektiğini konu alan bu algoritmalar, **yol izleme algoritmaları** olarak adlandırılırlar.

Otomatik ve yaro otomatik çizim yöntemleri, kullanılan görüntülerin çözünürlüklerine göre **düşük, yüksek** ve **çok çözünürlükte** kullanılabilen yöntemler olarak sınıflandırılacak ve her sınıfa giren yöntemler tektek ele alınarak daha önce bu konuda yapılan çalışmalara ait özet bilgiler verilerek karşılaştırma yapmaya olanak sağlanacaktır..

2.1. Düşük Çözünürlükte Yol Çizimi

Düşük çözünürlükte yollar çizgisel yapıda görünmekte ve görüntüdeki diğer detaylardan ayırt edilmeleri oldukça zor olmaktadır. Küçük detayların görülememesi nedeniyle yollar, homojen yüzeylerdeki çizgiler şeklinde temsil edilmektedirler.

İlk bakışta bu şekildeki bir basitleştirme avantaj olarak değerlendirilebilir fakat gerçekte, bu şekildeki bir temsilde, yollar görüntüdeki diğer çizgi detaylarla kolaylıkla karıştırılabilmektedir [3]. Bununla birlikte, kapalı ve gölgeli görüntü parçalarındaki yol izlemede, yolun tanımlanabilmesi için yeterli iz olmaması ve yol genişliklerinin tespit edilememesi nedeniyle, otomatik algoritmalarla yol çiziminde güçlükler ortaya çıkmaktadır.

Dezavantajlarına rağmen, düşük çözünürlükte yol çiziminin bazı avantajları da vardır. Görülemeyen küçük yol detayları bir soyutlama yoluyla ihmal edilebilmektedirler. Soyutlama; altyapıların ihmal edilmesi ve değişik görüntü parçalarının birleştirilmesi yoluyla gerçekleştirilmektedir. Yolların farklılık yaratan özel durumları daha az dikkate alındığından bu şekildeki bir soyutlama yolun tanımlanmasına yardımcı olmaktadır [8]. Düşük çözünürlükte yol çizim probleminin çözülmesi, çizginin bulunması ve çizilmesi

genel problemini büyük ölçüde çözer [3]. Öte yandan, yüksek çözünürlüklü görüntülerde daha fazla bilgi olmasına rağmen bu bilgilerden faydalanabilmek için daha fazla işleme gerek duyulmaktadır. Başka bir deyişle, yüksek çözünürlüklü görüntülerden otomatik yol çizimi, düşük çözünürlüklü görüntülerden otomatik yol çizimine oranla daha karışık algoritmalar gerektirmekte ve daha fazla zaman almaktadır [9].

2.1.1. Kalıp eşleştirme yöntemi

Bajcsy ve Tavakoli tarafından 1976'da yapılan otomatik yol çizimi ile ilgili ilk çalışmalardan birinde yaklaşık 57m x 79m mekansal ayırma gücüne sahip görüntüler kullanılmıştır. Düşük çözünürlük nedeniyle, sadece 3 veya daha fazla şeritli ana yollar tespit edilebilmiştir. Algoritma tarafından ilk önce bir eşik (threshold) işlemi gerçekleştirir, beklenen yol yoğunluğuna sahip noktalara 1, diğerlerine 0 değeri verilir. Sonra, bir yolun bulunduğuna işaret eden, öncedentanımlanmış, 52 adet kalıpile eşleşen görüntü parçaları taranır. Eşleşen parçalar, yol noktaları arasındaki mesafeler ve eğriliklerdeki sabitler kullanılarak yol parçaları ile birleştirilir. En sonunda algoritma, çizgileri bir piksel genişliğine kadar inceltir ve yolun anlamlı bir uzunluğa sahip olması gerektiğini kabul ederek kısa çizgileri yok sayar [1].

Yöntem düşük eğrilik, uzunluk ve genişlik gibi yola has bilgileri kullanmaktadır. Yol yüzeyinin eşit yoğunluğa sahip olduğu kabulü, farklı görüntülerdeki yollar için geçerli olmamaktadır.

2.1.2. Dinamik programlama yöntemleri

Fischler'in 1981 yılında geliştirdiği bu yöntemde, başlangıçtaki zayıf eşikişlemi, mutlak yol yoğunluğu değerini sabitleyen Duda Yol Operatörü (DYO) gibi yol ve köşe bulan işlemciler ile değiştirilmiştir. Bu çalışmada işlemciler iki sınıfa ayrılmaktadır: **Birinci çeşit işlemciler**, yolların dış hatlarıyla hassas olarak ilgilenmeksizin, yolun bulunmasında ve belirlenmesinde yüksek doğruluk sunmaktadırlar. **İkinci çeşit işlemciler**, doğru ve kesin tanımlamayı amaçlamamakta fakat detayların çizgiselleştirilmesinde yüksek hassasiyet sağlamaktadırlar. Birinci çeşit işlemcilerin uygulanmasında, yüksek olasılıkla yol parçalarına ait olan görüntü detayları elde edilmektedir (çok sayıda atlama hatası yapma pahasına bile olsa). Elde edilen yol

parçaları, ikinci çeşit işlemciler kullanılarak birleştirilmektedir. Birleştirme, bir dinamik programlama algoritması (F^* algoritması) kullanılarak gerçekleştirilmektedir [3].

Dinamik programlamaya dayanan farklı yaklaşımlar da ortaya çıkmıştır [10]. Grün ve Li tarafından Wavelet Dönüşümü ile yol keskinleştirme, bir ikinci çeşit işlemci çeşidi olarak kullanılmış ve ayrıca sabit genişlik, düşük eğrilik, bağlantı yüzey homojenliği ve komşu araziyle kontrastlık gibi parametrelerle tanımlanan genel yol modeli ortaya konmuştur. Model bir dinamik programlama algoritması içine oturtulmuş ve sabitler atanarak (örneğin yol eğriliği) kontrol edilmiştir [5]. Her iki yöntemde de yarı otomatik olup, birinci çeşit işlemcinin yol tanımlama görevi operatör tarafından elle gerçekleştirilmektedir.

2.1.3. Diferansiyel geometrik strateji

1996'da Steger tarafından otomatik çizgi ve yol çizimine farklı bir yaklaşım getirilmiştir. Yukarıda belirtilen yöntemlerde, lokal gri değeri değişimi gibi sadece lokal kriterler kullanılmaktadır, bu da çizgiyi oluşturan noktalar için bir çok hatalı hipotez üretilmesine ve görüntülerdeki önemli yolların seçilmesi için maliyetli yöntemlerin gereksinimine neden olmaktadır. Bu yöntemde, çizgilerin konumunu alt piksel doğruluğunda tanımlayan bir diferansiyel geometrik yaklaşım önerilmektedir. Her bir piksel için, görüntü fonksiyonunun Gauss yumuşatma fonksiyonu türevi ile ikinci derece Taylor polinomu hesaplanmaktadır. Çizgiye dik doğrultudaki yüksek eğrilik ve kaybolan eğim için çizgi noktalarına ihtiyaç duyulmaktadır. Yöntem, ölçek uzayında bir yapay çizginin çizimini analiz ederek otomatik çizgi çizimi için bir teorik temel sunmaktadır. Görüntü ölçeği ve aranan yolların genişliği arasındaki ilişki belirtilmekte ve uygun görüntü ölçeğinin otomatik hesaplanması için bir yöntem verilmektedir. Ek olarak, maksimum çizgi noktası sayısını koruyan, bağımsız çizgi noktalarını çizgiler ve bağlantı noktalarıyla ilişkilendiren bir algoritma sunulmaktadır [13].

2.1.4. Düşük çözünürlükte yol çiziminin özeti

Sonuçları özetlemek gerekirse, en son yaklaşım, otomatik yol çizimindeki en güvenli yöntemi göstermektedir. Diferansiyel geometrik strateji yöntemi, keyfi kalınlıktaki çizgi çizimine ölçeklendirilebilmekte ve böylelikle yüksek çözünürlüklü görüntülere de

uygulanabilmektedir. Yine de, bu yöntem, bağlantılar gibi yol özelliklerinin avantajlarını kullanmamaktadır. Bu yüzden, engellerden, gölgelerden veya zayıf kontrasttan dolayı yerel olarak yol özellikleri kaybedilebilmekte ve yol parçaları arasında boşluklar görülebilmektedir.

Bu dezavantaj, dinamik programlamaya dayalı algoritmalar için bir problem teşkil etmemektedir. Dinamik programlama algoritmaları, algoritmaların yarı-otomatik doğası sayesinde yolları ormanlık ve yerleşim yerleri gibi engelli bölgelerde başarıyla izlemektedirler. Yollar üzerindeki kaynak noktaları iyi belirlendiği durumlarda, dinamik programlama algoritmaları verilen noktalar arasında yolu takibe zorlanmaktadırlar. Kaynak noktalarının izlenmesi istenen yol üzerinde olmaması durumunda, algoritmalar tarafından izlenen ve çizilen yol parçası, çizilmesi istenen yol üzerinden sapmaktadır. Algoritmalar, tespit edilen yol parçalarını kabul etmeye veya reddetmeye yarayan kalite bilgisi vermemekte, sadece görüntü içerisinde modellerine en uygun şekilde yol parçalarını birleştirmeyi garanti etmektedirler.

2.2. Yüksek çözünürlükte yol çizimi

Düşük çözünürlükteki duruma benzer olarak, yüksek çözünürlükteki yol çiziminin hem avantajları hem de dezavantajları vardır. Kullanılan görüntünün mekansal ayırma gücü arttıkça yol detaylarının karakteristikleri, diğer çizgisel detaylara oranla, yolun tespiti için daha çok ipucu vermektedir.

Yol kenarları kontrastlığın değişmesine neden olmakta, yol üzerinde görüntü alındığı sırada farklı boyutlarda nesneler (örneğin farklı boyuttaki arabalar) bulunabilmekte ve ayırt edilebilmekte, yol yüzey kaplamasının cinsi ve kalınlığı farklılıklar göstermektedir. Çok sayıda ayırt edilebilebilen özelliklerin bulunması, otomatik yol çizim algoritmalarının karmaşıklığının artmasına neden olmaktadır. Algoritmanın karmaşıklığını düşük seviyede tutmak için değerlendirmeye alınacak yol özellikleri dikkatlice ve özenle seçilmelidir.

Yol özelliklerinin ve karakteristiklerinin seçiminin yanı sıra, bu özelliklerin yol çizim hipotezine katkılarının ağırlıklarının problemi vardır. Örneğin, eğer bir algoritma yol

kenarını değil de bir araba tespit ederse görüntünün yorumlanan kısmı bir yol parçası olarak kabul edilebilir mi?

Bu soruyu cevaplayabilmek için iki temel soruna çözüm bulmak gerekmektedir:

- Mevcut özelliklerin hangileri kullanılabilir?
- Seçilen özelliklerle ilgili bilgi, yolları belirleyebilmek ve çizebilmek için nasıl birleştirilebilir?

Bu sorulara farklı cevaplar veren farklı yaklaşımlar ve yöntemler ortaya konmuştur. Stratejilerine göre bu yöntemler iki ayrı grupta toplanabilir:

Birinci sınıf, bir operatör yardımıyla çizilen yolun başlangıcının çeşitli algoritmalar ile devamettirilmesini konu alan yol izleme algoritmalarıdır.

Diğer bir sınıf ise, yolun çevresinde ince bir şerit halinde tespit edilen nokta kümesi içinde en uygun noktaların seçilmesini ve bu noktaların bir enterpolasyon ile birleştirilmesini inceleyen AKM tabanlı algoritmalardır.

2.2.1. Yol izleme yöntemleri

Quam ve Lynn tarafından 1978 yılında geliştirilen yol izleyicisi, yol yoğunluk profilinin korelasyonunu kullanmaktadır. Yolun başlangıç noktası, yönü ve bu noktadaki kalınlığı verildiği takdirde, görüntü fonksiyonu yol boyunca analiz edilmekte ve olası yol noktaları arasında çapraz korelasyon hesaplanmaktadır. Çapraz korelasyonda hesaplanan hatalar, analiz edilen noktaların kabul veya reddedilmesinde kullanılmaktadır. Gerçek yolun konumu, korelasyon kayıklığından hesaplanmaktadır. Korelasyonun tepe noktası zayıf ise yol güzergahında yeni noktalar belirlenmekte ve yöntem tarafından inceleme tekrarlanmaktadır. Bu işlem, iyi bir uyum bulana kadar devam edilerek yolun bulunmasını sağlarken, daha önceden bulunan yoldan sapılarak yolun çok ötesine gidilmesine de neden olabilmektedir. Yol yoğunluk profilinin değiştiği gerçeği dikkate alınmalıdır. Yeni bir yol noktası için araştırma çok saparsa, yol profili için yeni bir hipotez oluşturulmakta ve araştırma uygunluk bulunamayıncaya kadar tekrar edilmektedir. Uygunluk bulunmayınca izleme durdurulmaktadır [15].

Yöntem, yollar hakkında, değişmeyen, tutarlı fakat anomalileri olan bilgiler kullanmaktadır. Anomaliler genellikle yollar üzerindeki arabalardan veya işaretlerden

oluşmaktadır. Yöntem ayrıca, yolların genişliklerindeki ve yüzey kaplama maddelerindeki keskin değişikliklere güvenmektedir. Oldukça çok bilgi dikkate alınmış olmakla birlikte, bu kadar çok bilgi her zaman istenilen sonuçları verecek anlamına gelmemektedir. Çapraz korelasyon tekniği, korelasyon hatalarının nedenleri hakkında kaliteli bilgi sağlamamaktadır. Bu nedenle yöntem, anomalilerin varlığını açıklayamamakta ve onları yol dışındaki noktalardan ayırt edememektedir. Yol eğriliğindeki keskin değişiklikler olası olmadığından, algoritma kısa anomalileri yolun konumunu tahmin ederek birleştirmektedir. Buna rağmen, eğer bir anomali, gölgelere, arabalara vb. engellere bağlı olarak artarsa, algoritmanın yolu izleme konusunda başarısız olma olasılığı oldukça yüksek olmaktadır.

Tanımlanan problemleri çözmek amacıyla McKeown ve Denlinger 1988 yılında farklı yol özelliklerine dayalı birkaç bağımsız yol izleyicisi önermişlerdir. Bunlardan birisi yol özelliklerinden bazılarına bağlı olarak başarısız olduğu durumda, izleme ilkinden farklı özelliklerdeki bir algoritmaya sahip bir başka yol izleyicisi tarafından devam ettirilmektedir [9].

"Otomatik Yol İzleyicisi" (OYİ) adı verilen algoritma iki adet yol izleyicisi kullanmaktadır. Birincisi; Quam ve Lynn tarafından geliştirilen yöntemdeki gibi korelasyon bazlı bir izleyicidir. Diğeri; kenar bazlı bir izleyicidir. Bu izleyici, yolların oldukça az bir eğrilikte ve düz kenarları olduğunu kabul etmektedir. Bir yolun başlangıç noktası, yönü ve kalınlığı verilmekte ve yolun yönü boyunca birleşmiş kenar noktaları aranmaktadır. Bu yöntemde kenar noktalarının seçiminde yapılan kabuller şu şekildedir:

- Görüntü gradyen kuvveti, sabitlenmiş bir eşik değerinden daha büyüktür.
- Nokta, kenara dik yönde yerel maksimumdadır.
- Komşu pikseller merkez pikselle en fazla 30 derece kenar açısı yapmaktadır [12].

Her bir basamakta birkaç kenar noktası alınmakta ve değerleri hesaplanmaktadır. En büyük değere sahip nokta kenar noktası olarak seçilmektedir. Minimum bir eşik değerinden hiçbir noktanın değeri yüksek değilse, tahmin etme işlemi yüzey izleyicisinde olduğu gibi yapılmaktadır. Tahmin, geçerli bir nokta bulmaksızın çok ileri gittiğinde izleme durdurulmaktadır.

OYİ nin kontrol süreci her bir yol izleyicisini bağımsız olarak uyarmakta ve her birine uyumsuz olarak sadece bir basamak ilerlemesine olanak tanımaktadır. OYİ, yol farklılığının olup olmadığını sorgulamaktadır. Eğer 7.5 m'den daha geniş bir farklılık tespit edilirse, OYİ her bir izleyicinin bağıl iyiliğini hesaplamakta ve iki izleyicide de hesaplanan güven değeri düşük ise tüm işlem iptal edilmekte veya yüksek güven değerine sahip izleyiciyi seçilmekte ve diğer izleyiciyi ilkinin pozisyonundan tekrara başlatılmaktadır.

OYİ ile oldukça iyi sonuçlar elde edilmektedir. Kenar izleyicisi, yüzey izleyicisine anomaliler boyunca, yollara ilişkin az bilgi olmasına rağmen yardımcı olmakta ve bu da sistemin performansını arttırmaktadır. McKeown ve Denlinger, OYİ'ye diğer izleme yöntemlerinin entegrasyonunun tüm sistem performansını geliştirebileceğini iddia etmişlerdir.

Bu yöntemin temel dezavantajı, çok sayıdaki parametrenin elle ayarlama gerektirmesidir. Maksimal yol ayrılığı, yüzey izleyici için kabul edilen çapraz korelasyon hatası, kenarların minimum büyüklüğe sahip gradyeni sadece birkaç örnektir. Parametreler arasındaki ilişkinin kesin olarak belirtilmemiş olması, sonuç üzerindeki etkilerinin analizini zorlaştırmaktadır. Parametreler görüntü alanı değiştikçe (örneğin yerleşim yerinden kırsal alana geçiş gibi) yeniden ayarlanmaktadır,.

2.2.2. Aktif kontur modeller tabanlı yöntemler

Aktif Kontur Modeller (AKM) tabanlı yöntemler görüntülerdeki çizgisel detayların yarı otomatik ve otomatik olarak çiziminde kullanılmaktadır [7]. AKM tabanlı yöntemler kullanılarak çok sayıdaki parametre sayısı azaltılabilmektedir. Çizgisel detayların çizimine ilişkin diğer yöntemlerle karşılaştırıldığında, AKM tabanlı yöntemlerin temel avantajı, detayların geometrik özelliklerinin sınırlandırıcılar formunda saklanıyor olması ve araştırmaya yön vermede doğrudan kullanılmasıdır.

AKM tabanlı yöntemler; yol kenarlarının ve merkez hatlarının belirlenmesinde başarıyla uygulanmaktadır. Bir yandan; bağlantı, az eğrilik ve sabit genişlik gibi yol özelliklerine karşılık gelen geometrik sınırlandırıcılar AKM ile işbirliğine sokulur ki bu da tanımlanmış yol özelliklerini AKM ile karşılayacak iç güçleri harekete geçirmektedir. Diğer yandan; fotometrik sınırlandırıcılar, AKM ile görüntüler arasındaki ilişkiyi

belirlemede kullanılmaktadır. Örneğin AKM yol kenarlarının belirlenmesi için uygulandığında, fotometrik sınırlandırıcılar, AKM'in eğriliği boyunca bir yerel maksimum belirlemede görüntü gradyen büyüklüğüne ihtiyaç duymaktadırlar [14].

İç ve dış güçler belirlendiğinde, yol çizimi işlemi AKM'in konumunu optimize etmek suretiyle gerçekleştirilmektedir. Başlangıçta AKM, aranılan yolun eğriliğinde ayarlanmakta ve daha sonra, iç ve dış güçlerin uyum sağladığı AKM'in şekline ve konumuna uymak için çeşitli hesaplamalar kullanılmaktadır. Güçlerin uyumu, AKM'ile yerleştirilen görüntü detayının, ona yakın diğer şeylerden daha çok yol özellikleriyle maksimum uyumunu garantilemektedir. Böylelikle, eğer AKM'in başlangıç konumu aranan yolunkine bağıl olarak yakınsa, optimizasyon işlemi yolun çizimini garantilemektedir. AKM tabanlı yöntemlerde otomatik olarak anomaliler dikkate alınmamakta ve yollar gölgeler ve engeller içinden de izlenebilmektedir [4]. Aynı yere ait çok ve farklı görüntü kullanımında AKM tabanlı algoritmaların başarı derecesi yükseltilebilmektedir [6].

Bir önceki bölümde dikkat çekilen parametrelerin hiçbirisine burada gerek duyulmamaktadır. Bununla birlikte; geleneksel AKM tabanlı yöntemler, araştırılan yolların doğru başlangıç tahminine ihtiyaç duymaktadırlar. Başlangıç tahminini basitleştirmek için Neuenschwander tarafından 1996 yılında geliştirilmiş bir yaklaşım ortaya konmuştur. Ziplock AKM adı verilen bu yöntemde bir yolun iki uç noktasının (başlangıç ve son) belirtilmesiyle yol çizim başlangıç tahminin karmaşıklığının azaltılması amaçlanmıştır [11]. AKM ile ilgili formülasyonlar bir sonraki bölümde ayrıntılarıyla anlatılacaktır.

2.2.3. Yüksek çözünürlükte yol çiziminin özeti

Yüksek çözünürlükteki görüntülerde görülebilen detayların çok olması bu detayların otomatik olarak çizilmesi işlemlerini zor ve karmaşık bir hale getirmektedir. AKM tabanlı yöntemler bütün elverişli bilgileri kullanmak yerine yol çizimini paralel kenarların konumlandırılmasına indirgemişlerdir. Yol yüzeyine ait bilgiler dikkate alınmamıştır. Optimize edilmiş AKM, engellere ve gölgelere maruz kalmış yolların kenarlarını sıklıkla doğru olarak tahmin etmektedirler. Bununla beraber bu bölümde sözü geçen yöntemler, konumlandırılacak detaylar hakkında azda olsa nitelik bilgisine

ihtiyaç duymaktadırlar. Dinamik programlama yöntemlerine benzer olarak, yol modelini en iyi şekilde karşılayan kaynak noktaları arasındaki aralığı izlemektedirler. Doğru bir yol çizimini gerçekleştirebilmek için AKM bir operatör yada daha üst düzey bir program tarafından başlatılmalıdır.

Yol izleme yöntemlerinin doğası yarı otomatik olmakla birlikte, başlangıç için daha az bilgi gerektirmektedirler. AKM tabanlı yöntemlerin tersine, yol yüzeyini analiz edip, kenarları görülemeyen veya yol çizimi için zayıf belirtiler gösteren yollarında belirlenebilmesine olanak tanımaktadırlar. Dezavantajı ise birçok eşik değeri ayarlamak için operatör eğitimine ihtiyaç duyulmasıdır çünkü parametreler elle ayarlanmaktadır.

2.3. Çok Çözünürlüklü Yol Çizimi

Yukarıda sunulan yöntemlerin hiçbiri güvenilir ve tam otomatik bir yol çizimine olanak sağlamamaktadırlar. Hepsinin avantaj ve dezavantajları vardır. Düşük çözünürlükte kullanılan yöntemler basit yol modelleri kullanarak birçok yolu belirleyebilmekle birlikte bilgi yetersizliği nedeniyle ortaya çıkan hatalara engel olamamaktadırlar. Yüksek çözünürlükte kullanılan yöntemlerde daha çok bilgiye ulaşabilmek ve yolları daha detaylı olarak belirleyebilmek olanaklı hale getirilmiştir. Bununla beraber, görüntüdeki diğer objeler de birçok çizgisel detayla temsil edildiğinden, bu detayların ayırt edilebilmesi için yol çizimi şi karmaşık yöntemlere ihtiyaç duymaktadır.

Her iki çözünürlükteki otomatik yol çizimlerinin avantajları çok çözünürlüklü otomatik yol çizim yaklaşımında birleştirilmiştir [2]. Bu yöntem, yolların tespiti ve bunların doğrulanması olmak üzere iki aşamayı içermektedir. Yolların tespiti düşük çözünürlükte kenar belirleme ile yapılırken, yüksek çözünürlükte paralel kenarlar belirlenmekte ve bu kenarlar arasında homojen yüzeyler kontrol edilerek dikdörtgenler şeklinde gruplanmaktadır. Yakın ve doğrusal dikdörtgen setleri yol parçaları şeklinde birleştirilip, yolları doğrulamak için, her iki çözünürlükteki sonuçlar bir araya getirilmektedir.

Algoritmanın avantajı, yüksek çözünürlükteki detaylı görüntü analizinin düşük çözünürlükte gerçekleştirilen yol bulma işlemlerince yönlendirilmesi ve bu yönlendirme dolayısıyla elde edilen zaman maliyetinin ve işlem miktarlarının azaltılmasıdır. Bununla

birlikte, kapalı veya gölgeli yol parçaları dikdörtgenlerle modellenemediğinden yorumlamada sıkıntıların ortaya çıkması algoritmanın dezavantajı olarak belirtilebilir.

Genel olarak, çok çözünürlükte otomatik yol çizim yaklaşımı tam otomatik yol çizimi için uygun bir iskelet oluşturmaktadır. Yüksek çözünürlükte yol çizimi zor fakat güvenilir olduğundan, bu işlemin muhtemel hatalarını göze alarak daha az çalışmayla düşük çözünürlükte gerçekleştirilen çizimler tarafından yönlendirilmesi iyi bir çözüm olarak ortaya çıkmaktadır. Çok çözünürlüklü otomatik yol çiziminin, yolların çoğunun düşük çözünürlüklü görüntüde çizgisel detaylar olarak görüldüğü durumlarda uygulanabilir olduğuna dikkat edilmelidir. Bu çoğunlukla kırsal alan görüntülerinde geçerli olabilmektedir. Kentsel alanlarda, yoğun diğer komşu detaylar genellikle yolları tespit edilemez hale getirmektedir. Bu tür alanlarda diğer otomatik yol çizim algoritmaları uygulanmalıdır. Bununla birlikte kırsal alanlardaki çoklu çözünürlükte otomatik yol çizimi kentsel ve ormanlık alanlardaki çizimlerde yol gösterici olarak kullanılabileceği düşünülmektedir.

3. AKTİF KONTUR MODELLER TEORİSİ

Aktif Kontur Modeller (AKM) algoritması, en bilinen otomatik ve yarı otomatik detay çizim yaklaşımlarından bir tanesidir. Bu konsept; bir optimizasyon işleminden geçirilen eğri belirleme algoritmalarına dayanmaktadır. Bu optimizasyon; elastodinamik modelleri ve bu modellerin iç ve dış kuvvetlerin etkisi altındaki davranışlarını kullanan eğrinin kontrastlık ve pürüzsüzlük modellerinin optimizasyonunu içermektedir. Bazı çalışmalarda, görüntüler ile enerji de birleştirilmektedir.

AKM yöntemlerinin uygulanmasındaki ana tanımlamalar, çizginin başlangıcı ve bundan sonraki çözümün doğru yakınsanmasına bağlıdır. Bu sorunların çözümü için çeşitli yöntemler öngörülmüştür. Bu yöntemler:

- Çok ölçekli yaklaşımlar
- Basınç kuvvetleri yaklaşımı
- Sonlu elemanlar yöntemi
- Düzey kümesi yaklaşımı
- Çok düzey yaklaşımı
- Gradyan vektör akışını içermektedirler.

AKM teorisi, görüntülerden kontur ve kenar çizmek amacıyla dizayn edilmiştir. Çizme sırasında, AKM kıvrımının durumu bölgesel olarak başlangıç konumuna yakın olarak optimize edilmektedir. AKM fotometrik ve geometrik kısıtlamaları bütünleştirmektedir. Fotometrik kısıtlamalar kullanılarak istenen özellikleri sağlayan görüntü detayları kullanılarak, AKM oluşturulmaktadır. Geometrik kısıtlamalar AKM optimizasyonunun gerginliğini ve katılığını kontrol etmekte ve kıvrımın yumuşak (smooth) olmasını sağlamaktadır.

AKM esasına dayanan kontur ve kenar çizme, AKM'in başlangıç konumuna ve fotometrik ve geometrik kısıtlamaların gücüne odaklı bir kontur araştırmasıdır. Bu yöntemde, görüntüyle ilgili detay bilgileri kontur çizmeye öncülük etmek üzere

kullanılabilmektedir. Bu olasılık bazen AKMi çok yararlı ve kullanışlı hale getirmektedir.

3.1. Analitik Sunum ve Optimizasyon

Kass ve arkadaşları tarafından 1987'de tanıtılan orijinal AKM algoritmasında, zamana bağımlı olarak modellenen iki boyutlu kıvrımlar aşağıdaki gibi ifade edilmektedir [7].

$$\vec{v}(s,t) = (x(s,t), y(s,t)) \qquad 0 \le s \le 1 \qquad (3.1)$$

Bu eşitlikte, s; orantılı çizgi uzunluğunu, t; şimdiki zamanı, x ve y; kıvrımın görüntü koordinatlarını temsil etmektedir. Zaman ilerledikçe AKM algoritması konumunu ve şeklini, etkileyen kuvvetlere göre deforme etmektedir. AKMi etkileyen kuvvetler üç genel sınıfta toplanmaktadır. İlki; **görüntü kuvvetleri** olup, görüntü fonksiyonu tarafından aktif hale getirilmektedirler. Bu kuvvetler, AKMi çizgi ve kenarlar gibi görüntü detaylarına yönlendirmektedirler. İkincisi; **iç kuvvetler** olarak nitelendirilmektedir ve parça parça yumuşaklık kısıtlarını zorlayarak AKM'in şeklini kontrol etmektedirler. Son olarak **dış kuvvetler**, AKMin ilgilenilen bölgeye en yakın noktaya başka bir ifadeyle yerel minimuma olan hareketinden sorumludur. Bu kuvvetler genellikle AKMi, özelliklerinin değiştirilmesi suretiyle, farklı işler için esnek ve uyumlu hale getirmek için kullanılmaktadırlar.

Bir AKM noktasına $v(s_0,t_0)$ etkiyen bütün kuvvetlerin toplamı olan F vektörü "0" olmadıkça, noktanın konumu, F vektörünün yönü ve büyüklüğüne bağlı olarak değişiklik göstermektedir. Sürecin bölümlenmesi sırasında AKMinin kıvrımlarındaki bütün kuvvetlerin dengelendiği durumlar araştırılmaktadır. Uygulanan görüntü kuvvetleri, istenilen özelliklerdeki çizgisel görüntü detaylarına karşılık gelen AKMin konumunu garantilemektedir veya çizgisel görüntü detaylarına karşılık gelen AKMin konumunun istenilen özelliklerde olmasını garantilemektedir.

3.1.1. AKMin enerjisi

AKMin konumu, optimal konumunu bulmak için, etkiyen enerjilerin toplamı olarak ifade edilmektedir.

$$E(\vec{v}) = E_{img}(\vec{v}) + E_{int}(\vec{v}) + E_{ext}(\vec{v}) \quad (3.2)$$

Bu eşitlikte E_{int}, kıvrımdan kaynaklan AKM'in iç enerjisini göstermektedir. Görüntü enerjisi, E_{img}, görüntü kuvvetlerinin artmasına yol açarken, dış enerjiler, E_{ext}, dış kuvvetlerin artmasına neden olmaktadır. Bütün kuvvetlerin dengelendiği yerde AKM'in konumu, toplam yerel minimuma karşılık gelmektedir.

AKM'in görüntü enerjisi aşağıdaki gibi ifade edilebilir.

$$E_{img}(\vec{v}) = -\int_0^1 P(\vec{v}(s,t))ds \quad (3.3)$$

P(v(s,t)) ilgili detaylara karşılık gelen yüksek değerli bir fonksiyonu temsil etmektedir. Görüntü üzerinde AKM'i kenarlara doğru yönlendirirken, P(v(s,t)) fonksiyonu genellikle görüntü gradyanının büyüklüğüne eşit olarak alınmaktadır.

$$P(\vec{v}(s,t)) = \left|\nabla I(\vec{v}(s,t))\right| \quad (3.4)$$

I(v(s,t)) bir ham görüntü yada Gauss Kernel filtresinden geçirilmiş bir görüntü olarak kabul edilmektedir. Görüntünün yumuşatılmasında ve önemsiz detayların kaldırılmasında Gauss Kernel filtresi kullanılmakta olup, böylece AKM'in daha düşük görüntü enerjili konumlara hareketinin önlenmesi ve daha belirgin detaylara yönelmesi sağlanmaktadır.

İç enerji AKM'in şekli üzerindeki geometrik kısıtlamaları yerine getirmeyi olanaklı kılmaktadır ve aşağıdaki gibi ifade edilmektedir.

$$E_{\text{int}}(\vec{v}) = \frac{1}{2}\int_0^1 \alpha(s)\left|\frac{\partial\vec{v}(s,t)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2\vec{v}(s,t)}{\partial s^2}\right|^2 ds \quad (3.5)$$

α(s) ve β(s) AKM'in gerginliğini ve katılığını düzenleyen keyfi fonksiyonlardır. Gerginlik için kısıtlama birinci derece terimler (ilk sıradaki terimler) tarafından yerine getirilmekte ve AKM'in zar gibi davranmasını sağlamaktadır. Katılık, ikinci

derece terimler ile kısıtlanmalta ve AKM'in ince bir tabaka gibi davranmasını sağlamaktadır.

3.1.2. AKM enerjisinin minimizasyonu

Dış kuvvetlerin yokluğunda, (3.3) ve (3.5) eşitliklerinin (3.2) eşitliğinde yerine konmasıyla, AKM'in toplam enerji fonksiyonu aşağıdaki duruma gelmektedir [11].

$$E(\vec{v}) = -\int_0^1 P(\vec{v}(s,t))ds + \frac{1}{2}\int_0^1 \alpha(s)\left|\frac{\partial\vec{v}(s,t)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2\vec{v}(s,t)}{\partial s^2}\right|^2 ds \qquad (3.6)$$

AKM algoritmasının enerjisi, minimize edilirse, hareketin Euler-Lagrange diferansiyel eşitliğine göre bir çözümü bulunabilme olasılığı ortaya çıkmaktadır [16].

$$-\frac{\partial}{\partial s}(\alpha(s)\frac{\partial\vec{v}(s,t)}{\partial s}) + \frac{\partial^2}{\partial s^2}(\beta(s)\frac{\partial^2\vec{v}(s,t)}{\partial s^2}) = -\nabla P(\vec{v}(s,t)) \qquad (3.7)$$

(3.5) eşitliğindeki belirli deformasyon enerjisi seçilerek, AKM algoritmasının hareketini yöneten diferansiyel eşitlik doğrusal hale getirilmektedir. Bu, minimizasyon problemini kolaylaştırmaktadır, çünkü (3.7) eşitliği iki diferansiyel eşitliğe ayrılabilmekte olup, x(s,t) ve y(s,t) için bağımsız olarak çözülebilmektedir.

$$\begin{aligned} -\frac{\partial}{\partial s}(\alpha(s)\frac{\partial x(s,t)}{\partial s}) + \frac{\partial^2}{\partial s^2}(\beta(s)\frac{\partial^2 x(s,t)}{\partial s^2}) &= -\frac{\partial P}{\partial x} \\ -\frac{\partial}{\partial s}(\alpha(s)\frac{\partial y(s,t)}{\partial s}) + \frac{\partial^2}{\partial s^2}(\beta(s)\frac{\partial^2 y(s,t)}{\partial s^2}) &= -\frac{\partial P}{\partial y} \end{aligned} \qquad (3.8)$$

Buna rağmen bu eşitliklerin tek bir çözümünün olabilmesi için, bu eşitliklerden birisinin, s=0 ve s=1 için x(s,t), y(s,t)'nin türevleri ve değerleri gibi, sınır koşullarının belirlenmesi gerekmektedir. Uygulamada P ayrı bir fonksiyon olduğundan (3.8) eşitlikleri analitik olarak çözülememekte ve sayısal iteratif bir çözüm gerektirmektedir.

(3.8) eşitliğinin çözümü için sayısal iteratif metodun yanında AKM algoritması uygulamaları için görüntü ve iç enerjiler arasındaki denge de önemlidir. Eğer

enerjilerden biri diğerine daha baskın olursa, bu AKM algoritmasının yumuşaklık kısıtlamasını durduracak ya da AKM algoritmasını görüntü kuvvetlerini ihmal etmek için zorlayacaktır. Bunlar istenmeyen etkilerdir. AKM algoritmasının iç enerjisi kontrol edilebilirse, denge, AKM'nin gerilimini ve katılığını kontrol eden $\alpha(s)$ ve $\beta(s)$ fonksiyonlarının dengelenmesiyle sağlanabilmektedir.

İç ve görüntü enerjileri arasındaki denge, $\alpha(s)$ ve $\beta(s)$ fonksiyonlarının yerine değeri, otomatik olarak hesaplanabilen sabit bir λ faktörü ile etkili olarak sağlanabilmektedir [4].

$$\lambda = \frac{\left|\delta E_{img}(\vec{v})\right|}{\left|\delta E_{\text{int}}(\vec{v})\right|} \tag{3.9}$$

δ, fark operatörüdür. Çözüm, AKM'in konumunun, başlangıç tahminin sonuç çözümüne yakınlığına bağlıdır. λ'nın otomatik olarak tekrar dengelenerek optimizasyonu, AKM'in konumunun bir öncekine şeklinden ve görüntü fonksiyonundan bağımsız olarak yaklaşmasıyla sonuçlanmaktadır. (3.8) eşitliğindeki $\alpha(s)$ ve $\beta(s)$ nin yerine λ'nın kullanılmasıyla aşağıdaki eşitlik elde edilmektedir.

$$\lambda\left(-\frac{\partial^2 v(s,t)}{\partial s^2} + \frac{\partial^4 v(s,t)}{\partial s^4}\right) = -\frac{\partial P}{\partial v} \tag{3.10}$$

Burada v, x veya y'den birini ayrı ayrı temsil etmektedir.

3.2 AKM'in Farklı Sunumları

(3.10) eşitliğinin bilgisayar sisteminde çözümü "AKM'in gelişimini ifade eder", [t] ile gösterilen farklı zaman aşamalarında uygulanarak gerçekleştirilebilir. AKM'in türevlerini ve konumunu yaklaştırmak için (3.10) eşitliğinde, AKM'in gösteriminin farklılaştırılmasına ihtiyaç duyulmaktadır. Bu sorun, düzenli aralıklarla AKM'i n köşeli bir poligonal eğriye dönüştürmekle giderilebilir.

$$\vec{v}^{[t]} = \left\{\vec{v}_i^{[t]}\right\} = \left\{\left(x_i^{[t]}, y_i^{[t]}\right) = \left(x(i/n,t), y(i/n,t)\right)\right\} \qquad i = 0,\ldots,n-1 \tag{3.11}$$

O halde, zaman basamağının 1 olarak alınmasıyla, (3.10) eşitliği farklı bir formda tekrar yazılabilmektedir.

$$\lambda(-(v_i^{[t]})_{ss} + (v_i^{[t]})_{ssss}) = -P_v\Big|_{\substack{\rightarrow[t-1]\\ v_i}} \tag{3.12}$$

v, x ya da y'den birini ayrı ayrı ve $(v_i^{[t]})_{ss}$, $(v_i^{[t]})_{ssss}$ $v_i^{[t]}$'nin ikinci ve dördüncü dereceden türevini ifade etmektedir. $(v_i^{[t]})_{ss}$ ve $(v_i^{[t]})_{ssss}$ türevlerini yaklaştırmak için merkezi farklar kullanılmaktadır. Böylece, (3.12) eşitliğinin sol tarafı aşağıdaki gibi yazılabilmektedir.

$$\begin{aligned}
&\lambda(-(v_i^{[t]})_{ss} + (v_i^{[t]})_{ssss}) \approx \\
&-\lambda(v_{i-1}^{[t]} - 2v_i^{[t]} + v_{i+1}^{[t]}) + \lambda(v_{i-2}^{[t]} - 4v_{i-1}^{[t]} + 6v_i^{[t]} - 4v_{i+1}^{[t]} + v_{i+2}^{[t]}) = \\
&\lambda(v_{i-2}^{[t]} - 5v_{i-1}^{[t]} + 8v_i^{[t]} - 5v_{i+1}^{[t]} + v_{i+2}^{[t]})
\end{aligned} \tag{3.13}$$

v; x ya da y'den birini ayrı ayrı temsil etmektedir. Merkezi farklar simetrik olduğunda, başlangıç köşesi (I=0, 1) ve son köşe (I=n-2, n-1) hesaplanamamaktadır. Bunun yerine ileri ve geri farklar kullanılmalıdır. Bu $\lambda(-(v_i^{[t]})_{ss}+(v_i^{[t]})_{ssss})$'nın aşağıdaki tahminlerini vermektedir.

$$\begin{aligned}
&v_0^{[t]} : \lambda\left[(v_0^{[t]} - v_1^{[t]}) + (v_0^{[t]} - 2v_1^{[t]})\right] = \lambda(2v_0^{[t]} - 3v_1^{[t]} + v_2^{[t]}), \\
&v_1^{[t]} : \lambda\left[(-v_0^{[t]} + 2v_1^{[t]} - v_2^{[t]}) + (-2v_0^{[t]} + 5v_1^{[t]} - 4v_2^{[t]} + v_3^{[t]})\right] = \\
&\lambda(-3_0^{[t]} + 7v_1^{[t]} - 5v_2^{[t]} + v_3^{[t]}), \\
&v_{n-1}^{[t]} : \lambda\left[(v_{n-1}^{[t]} - v_{n-2}^{[t]}) + (v_{n-1}^{[t]} - 2v_{n-2}^{[t]} + v_{n-3}^{[t]})\right] = \lambda(2v_{n-1}^{[t]} - 3v_{n-2}^{[t]} + v_{n-3}^{[t]}), \\
&v_{n-2}^{[t]} : \lambda\left[(-v_{n-1}^{[t]} + 2v_{n-2}^{[t]} - v_{n-3}^{[t]}) + (-2v_{n-1}^{[t]} + 5v_{n-2}^{[t]} - 4v_{n-3}^{[t]} + v_{n-4}^{[t]})\right] = \\
&\lambda(-3v_{n-1}^{[t]} + 7v_{n-2}^{[t]} - 5v_{n-3}^{[t]} + v_{n-4}^{[t]})
\end{aligned} \tag{3.14}$$

(3.12) eşitliğinin bir $v_i^{[t]}$ köşesi için çözümü aynı zaman basamağında [t], komşu köşenin konumuna ihtiyaç duymaktadır. Bu da (3.12) eşitliğinin her $v_i^{[t]}$(I=0, .,.n-1) köşesi için çözülmesi gerektiği anlamına gelmektedir. Bütün köşeler için eşitlik kümeleri biraraya getirildiğinde matris formunda yazılabilen bir sistem meydana getirilebilmektedir.

$$K.v^{[t]} = -P_v\Big|_{v^{[t-1]}} \tag{3.15}$$

$v^{[t]}$ her bir $(x_0^{[t]}, .,.x_{n-1}^{[t]})^T$ yada $(y_0^{[t]}, .,.y_{n-1}^{[t]})^T$'i ifade etmekte ve K ise n*n simetrik bir beşgen diyagonal matrisi göstermektedir.

$$K = \begin{bmatrix} 2\lambda & -3\lambda & \lambda & & & & & & \\ -3\lambda & 7\lambda & -5\lambda & \lambda & & & & & \\ \lambda & -5\lambda & 8\lambda & -5\lambda & \lambda & & & & \\ & \lambda & -5\lambda & 8\lambda & -5\lambda & \lambda & & & \\ & & & & & & & & \\ & & & \lambda & -5\lambda & 8\lambda & -5\lambda & \lambda & \\ & & & & \lambda & -5\lambda & 8\lambda & -5\lambda & \lambda \\ & & & & & \lambda & -5\lambda & 7\lambda & -3\lambda \\ & & & & & & \lambda & -3\lambda & 2\lambda \end{bmatrix} \tag{3.16}$$

K singüler bir matristir ve tersi alınamaz. Bu yüzden (3.15) eşitliği $v^{[t]}$ için doğrudan çözülememektedir. Bu da, (3.8) diferansiyel eşitliğinin, tek bir çözümelde etmek için 4 sınır değeriyle sağlanması gerçeğini ortaya koymaktadır. Bu sınırlar iki noktayı ve bunların tanjant vektörlerini AKM'i sabitleyerek elde edilebilmektedir. Ancak, orijinal AKM yaklaşımında, AKM'in yerleşimini kısıtlamak için bir sürtünme kuvveti kullanılmaktadır. Bu; eşitliğin sağ tarafının, bir γ iterasyon basamağının (yada viskozite faktörünün) sonucunda negatif zaman türevlerine ($v^{[t-1]}$-$v^{[t]}$) ayarlanması (eşitlenmesi) ile gerçekleştirilmektedir.

$$K.v^{[t]} + P_v\Big|_{v^{[t-1]}} = 0 \tag{3.17}$$

Sonuç olarak AKM'in hareketinin eşitliği aşağıdaki formda yazılabilmektedir.

$$(K + \gamma I).v^{[t]} = \gamma v^{[t-1]} - P_v\Big|_{v^{[t-1]}} \tag{3.18}$$

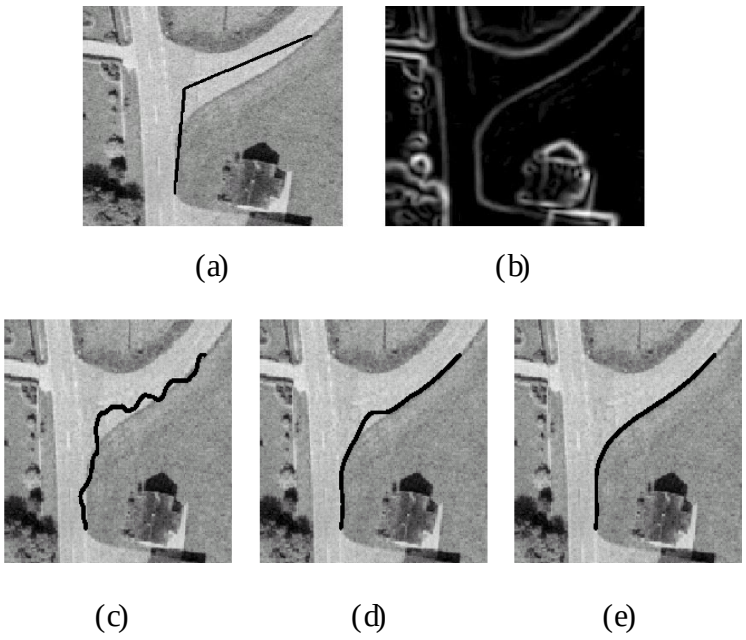
Burada I, n*n birim matris olduğundan (K+γI) sonucunda singüler olmayan bir matris elde edilmekte ve böylece sonuç matrisin tersi alınabilmektedir. Bu demektir ki, $v^{[t]}$ nümerik olarak çözülerek direkt elde edilebilmektedir. Cholesky yöntemi

kullanılarak (3.18) eşitliği, o(n) zamanında, simetrik beşgen diagonal matris (K+γI) için etkili bir biçimde çözülebilmektedir.

Enerjinin bölgesel minimumunda, AKM'in konumunu bulmak için (3.18) eşitliği iteratif olarak çözülmektedir. İterasyon, AKM'in verilen v[0] başlangıç konumu ile başlamakta ve daha sonra viskozite (γ) aşağıdaki gibi hesaplanmaktadır [4].

$$\gamma = \frac{\sqrt{2n}}{\Delta}\left|\frac{\partial E(\vec{v})}{\partial \vec{v}}\right| \qquad (3.19)$$

Δ; AKM'inizin verilen ortalama yer değiştirme miktarını ifade etmektedir. (3.18)'in çözümü, toplam enerjisi doğrulanan AKM'in yeni konumu ile sonuçlanmaktadır. Enerjinin bir önceki zaman basamağındaki enerjiye göre düşüşü, AKM'in hareketinin minimum enerjiye doğru hareketinin doğru yönde olduğunu göstermektedir. Diğer taraftan enerji yükselirse, AKM bir önceki konumunda bekletilmekte ve önerilen yer değiştirme (Δ) azaltılmaktadır. Bu işlem, Δ eşik değerinden az olana kadar devam etmektedir. AKM böyle durumlarda, P fonksiyonu ile vurgulanan belirgin görüntü detaylarında konumlandırılmaktadır.

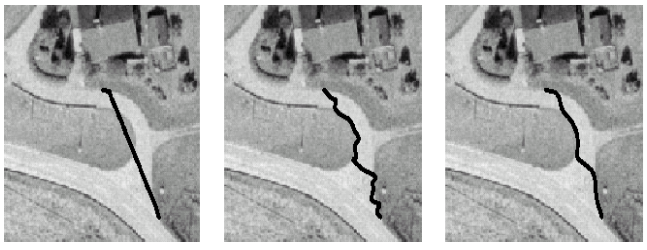


Şekil 3.1 Yol kavşağının bir kenarının çizilmesi. (a) Aranan kontura yakın olan AKM'in başlangıcı. (b) Görüntünün eğim büyüklük haritası. (c-d-e) Optimizasyonun 5, 10, 15 nci zaman aşamalarındaki evreleri [17].

Bu yaklaşımın bir hava fotoğrafına uygulanması Şekil 3.1'de gösterilmiştir [17]. Öncelikle AKM bir yol kavşağının sınırlarına yerleştirilir. Şekil 3.1(a) aranan kontura yakın konumlandırılmış yaklaşık 100 köşe noktası uzunluğundaki AKM'in başlangıç konumunu göstermektedir. Kavşak sınırı güçlü görüntü kontrastlığı ile karakterize edildiğinden, (3.4)'e göre AKM'in hareketi (3.7)'deki P fonksiyonuyla seçilir. Bu fonksiyon, görüntüye uygulandığında Şekil 3.1(b)' de gösterilen görüntü elde edilir. Şekil 3.1(b)' deki parlak noktalar, görüntüde gradyan büyüklükleri ile karakterize edilen yüksek kontrastlı noktalara karşılık gelmektedir. Böyle noktalar görüntü kuvvetlerini, AKM'i çekmesi için uyarmaktadır. Şekil 3.1(c), (d), (e), optimizasyonun beşinci, onuncu, onbeşinci iterasyondaki AKM'in konumlarını göstermektedir. Görülebileceği gibi onbeşinci iterasyonda AKM'in hassas olarak yol kavşağı sınırına yerleştirilmesi başarılmıştır. Bu, yaklaşımın hızlı yakınsadığını göstermektedir [17].

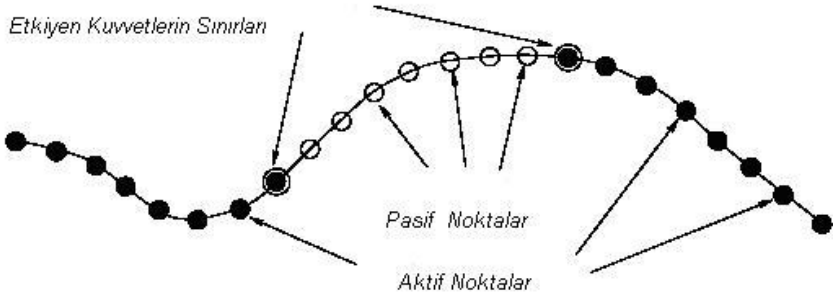
3.3. Ziplock AKM

Yukarıda açıklanan yaklaşımın dezavantajlarından biri, bölümlemenin kalitesinin genelde AKM'in başlangıç konumunun seçimine bağlı olmasıdır. Şekil 3.2'de gösterildiği gibi diğer detayların varlığı, AKM'in başlangıç konumu istenilenden uzak olduğunda, AKM'in hareketini engellemektedir [17]. Bu sorunu çözmek için, yalnız bitiş noktasına sahip AKM'in başlangıcına izin veren, genişletilmiş AKM geliştirilmiştir [11]. Aranılan konturun bulunması için AKM'in doğru başlangıç ve bitiş noktalarıyla başlatılması, optimizasyonun bu noktalardan AKM'in ortasına doğru kıvrım boyunca kenar bilgilerini çoğaltarak ilerlemesine olanak sağlamaktadır.



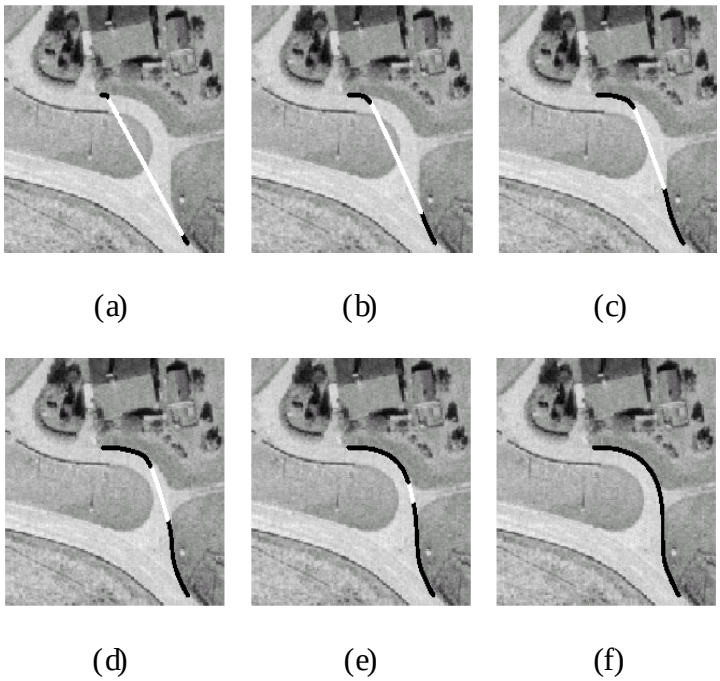
(a) (b) (c)

Şekil 3.2 Orjinal AKM yaklaşımıyla kontur çıkarma. (a) AKM'in başlangıcı yolun diğer tarafı tarafından bloke edilir. (b,c) AKM'in optimizasyonunu, yolun doğru tarafına yerleştirme başarısız sonuçlanmıştır [17].



Şekil 3.3 Ziplock AKM Optimizasyon boyunca kuvvet sınırları son noktalardan merkeze doğru yavaş yavaş artar. Kuvvet sınırlarının her hareketinde, kıvrımın konumu yeniden optimize edilir ve köşe noktaları doğru kontura yerleşir [17].

Ziplock AKMi iki kuvvet sınırı ile üç parçaya bölünmektedir (Şekil 3.3) [17]. AKMin bitiş noktaları ile kuvvet sınırları arasında aktif köşe noktaları, diğer bölümde ise pasif köşe noktaları bulunmaktadır. Optimizasyon boyunca aktif köşe noktalarının konumu, orijinal AKM için kullanılan yola benzer bir yolla hesaplanmaktadır. Ancak pasif köşe noktalarında hiçbir görüntü kuvveti hissedilmemekte ve bu yüzden hareketleri hiçbir görüntü detayından etkilenmemektedir. Optimizasyonun başlangıcında, kuvvet sınırları AKMin son noktalarında başlatılmaktadır. Optimizasyon ilerlerken aktif köşe noktalarında çoğaltılan AKM parçaları, görüntü detayına optimize edilmekte ve AKMin merkezine doğru yavaş yavaş ilerlemektedirler.



Şekil 3.4 Ziplock AKMile yol çizimi. (a) İki son nokta ile AKMin başlatılması. (b-e) Optimizasyon sürecinin ardışık evreleri. Pasif köşe noktaları beyaz, aktif köşe noktaları siyah olarak gösterilmiştir. (f) Sonuç; doğru yol kenarı çizilmiştir [17].

Kuvvet sınırlarının yavaş yavaş büyümesi demek, aktif köşe noktalarının bir önceki çıkarılan kontura yaklaşıyor olması demektir. Bu yolla bütün optimizasyon boyunca görüntü bilgileri, sadece aranılan kontura yakın olarak bilinen noktalarda alınmaktadır. Optimizasyon, kuvvet sınırları birbirleriyle buluşunca sona ermektedir. Orijinal ve Ziplock AKM'inin karşılaştırılması ve Ziplock yaklaşımına göre yol kenarı çizimi Şekil 3.4'de gösterilmiştir [17]. Şekil 3.2(a)'dakine benzer bir kontur başlangıcı, yol kenarının doğru çizimi ile sonuçlanmaktadır. AKM'in pasif bölümü görüntü kuvvetlerini hissetmediğinde, orijinal AKM'de olduğu gibi yolun diğer kenarı tarafından bloke edilmemekte ve böylece doğru çizim yapılmış olmaktadır.

4. GÖRÜNTÜ BÖLÜMLEME

Görüntü bölümleme, ilgi alanındaki yapıların arka plandan veya diğerlerinden ayrıştırılması işlemi olup, görüntü işleme alanında bu konuda bir çok algoritmanın geliştirildiği temel analiz fonksiyonudur. Görüntü bölümleme, görüntünün anlamlı bileşenlerinin tanımlanması şeklinde de tarif edilebilir [18]. Bölümleme, üç boyutlu verilerin görüntülenmesi ve bilgisayar ortamında depolanması işlemlerinde de yoğun olarak kullanılmaktadır [19].

Görüntü bölümleme işleminin temel hedefi, verinin bir veya daha fazla özellik veya özniteliğine göre homojen olan parçalar, sınıflar yada altbölümler şeklinde tasnif edilmesi ve bilgisayarda depolanmasıdır [20-26]. Görüntü bölümleme, *piksellerin* sınıflandırılması işlemini gerektirdiğinden, daha çok desen tanıma problemi gibi ele alınmakta ve ilgili tekniklerden yararlanılmaktadır.

Bölümleme; piksellerin lokal ışık şiddetleri, uzaydaki konumları, komşulukları veya şekil özeliklerine göre nesne sınıflarına ayrılması ile ilgilenmektedir, bu nedenle sınıflandırma terimi kimi zaman bölümleme terimi yerine kullanılmaktadır [27]. Görüntü bölümleme teknikleri, sınıflandırma özelliklerine göre değişik şekillerde yapılabilir.

- El ile, yarı-otomatik yada otomatik [28],
- Piksel-tabanlı (lokal yöntemler) ve bölge-tabanlı (bölgesel yöntemler) [29],
- El ile betimleme, alt düzey sınıflandırma (eşik-değer kullanma, bölge oluşturma vb.), model-tabanlı bölümleme (çok bantlı veya özellik eşleme teknikleri),
- Klasik (eşik-değer kullanma, kenar-tabanlı ve bölge-tabanlı teknikler), istatistiksel, bulanık mantık ve sinir ağı teknikleri [30].

Yaygın olarak kullanılan bölümleme teknikleri genel olarak iki sınıfa ayrılabilir:

- **Bölgesel bölümleme:** Belirli bir homojenlik kriterini sağlayan bölgelerin belirlenmesi tekniğidir.

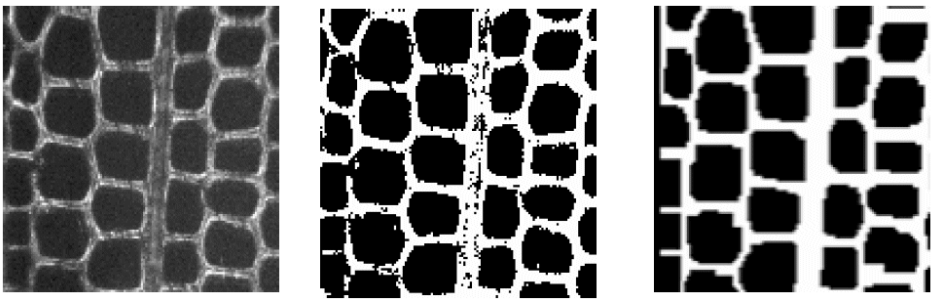
- **Kenar tabanlı bölümleme:** Değişik özelliklere sahip bölgelerin arasındaki kenarın belirlenmesi tekniğidir [22, 25].

4.1. Bölgesel Bölümleme

Bölgesel bölümleme yöntemlerinin yaygın uygulama alanı bulanlarından birisi eşik-değer kullanmadır. Bu yöntemde seçilen eşik değer dikkate alınarak veri; eşik değere eşit olan, eşik değerden küçük olan ve eşik değerden büyük olan veri alt kümelerine ayrılır. Eşik-değer kullanma konusunda değişik yöntemler mevcuttur. Bu yöntemler, üç ana başlık altında toplanabilir:

- Bölgesel yöntemler; gri seviyeli histogramlara dayalı veya lokal özelliklere dayalı,
- Lokal eşik değer seçimi,
- Dinamik eşik değer kullanımı.

Bölgesel eşik-değer tekniği ile görüntü bölümleme örneği Şekil 4.1'de gösterilmiştir. Orijinal görüntüde (Şekil 4.1(a)) siyah bir arka planda beyaz doku yapısı görülmektedir. Piksellerin yoğunluk değerleri 0 ila 255 arasında değişmektedir. Bu örnekte, uygun olarak seçilen bir eşik değer sayesinde arka plan ile nesne ayrımı yapılmıştır (Şekil 4.1(b)). Daha sonra çalıştırılan bölge oluşturma algoritması ile elde edilen sonuç Şekil 4.1(c)'de gösterilmiştir [19].



(a) (b) (c)

Şekil 4.1 Görüntü bölümleme örneği [19].

Kümeleme (Clustring) algoritmaları, görüntüyü oluşturan piksellerin birbirleri ile çok büyük benzerlik taşıyanlarını alt kümeler şeklinde düzenleyerek sınıflandırma işlemini gerçekleştirmektedirler [19]. Bu teknikte yapılan işlem tüm piksellerin tek tek ele alınarak, özelliklerinin uygun olduğu kümede yer almasının sağlanmasıdır.

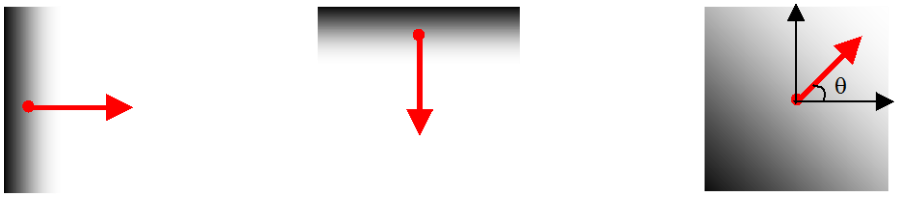
Bölge oluşturma tekniği ise; tanımlanan bir özelliğe sahip veya çok yakın olan birbirlerine komşu piksellerin tespit edilerek, bunların oluşturduğu grupları belirleme işlemidir.

4.2. Kenar Tabanlı Bölümleme

Kenar tabanlı bölümleme, veriye ait değerlerin keskin değişim gösterdiği hatların veya tabakaların bulunması işlemidir. Değişimin olduğu yerler nesnelerin kenarlarını veya çevrelerini oluşturmaktadır. Bir fonksiyonun değerinin çabuk değişim gösterdiği yeri ve bu değişimin değerini gösteren ölçüte gradyan denilmektedir. Bu tür hatların ve tabakaların tespiti, bir eksen boyunca görüntüyü oluşturan verilerin temsil ettiği fonksiyonun birinci ve ikinci dereceden türevlerinin alınması ile gradyan hesabının yapılmasını gerektirir. Birinci dereceden türevlerin maksimum olduğu ya da ikinci dereceden türevlerin sıfır olduğu noktalar değişimin olduğu yerleri temsil etmektedir. Bir görüntünün bir P noktasındaki gradyanı aşağıdaki şekilde ifade edilebilir:

$$\nabla f(x_P, y_P) = \left[\frac{\partial f(x_P, y_P)}{\partial x}, \frac{\partial f(x_P, y_P)}{\partial y} \right] \tag{4.1}$$

Değişik örnek görüntüler için ışık şiddetinin hızlı değişim gösterdiği noktalardaki gradyanların vektör gösterimleri Şekil 4.2'de verilmiştir [19].



(a) (b) (c)

Şekil 4.2 Işık şiddetinin hızlı değişim gösterdiği noktalardaki gradyanlar [19].

Şekil 4.2(a)'daki gradyan değeri için (4.1) eşitliğinin ikinci elemanı, (b)'deki gradyan değeri için ise eşitliğin birinci elemanı 0 olmaktadır. Gradyanın büyüklüğü aşağıdaki şekilde ifade edilmektedir:

$$|\nabla f(x_P, y_P)| \cong \sqrt{\left(\frac{\partial f(x_P, y_P)}{\partial x}\right)^2 + \left(\frac{\partial f(x_P, y_P)}{\partial y}\right)^2} \qquad (4.2)$$

Gradyanın yönü (θ) ise (4.3) eşitliği ile hesaplanabilir.

$$\theta = \arctan\left(\frac{\partial f(x_P, y_P)}{\partial x} / \frac{\partial f(x_P, y_P)}{\partial y}\right) \qquad (4.3)$$

Yukarıda verilen formüller süreklilik arz eden fonksiyonlar için geçerlidir. Sayısal görüntüler piksellerden oluşmaları nedeniyle kesiklilik arz ederler. Hesaplamaların daha hızlı yapılabilmesi amacıyla, kesikli değişkenler için (4.2) formülü yerine aşağıdaki yaklaşıklık kullanılabilir:

$$|\nabla f(x_P, y_P)| \cong \left|\frac{\partial f(x_P, y_P)}{\partial x}\right| + \left|\frac{\partial f(x_P, y_P)}{\partial y}\right| \qquad (4.4)$$

(4.4) eşitliği formül içerisindeki türevler aşağıdaki şekilde ifade edilebilir:

$$\frac{\partial f(x_P, y_P)}{\partial x} \cong f(x_P + n, y_P) - f(x_P - n, y_P) \qquad (4.5)$$

$$\frac{\partial f(x_P, y_P)}{\partial y} \cong f(x_P, y_P + n) - f(x_P, y_P - n) \qquad (4.6)$$

Burada n en küçük birimdeki tamsayıyı ifade etmekte olup genellikle 1 alınır. (4.5) ve (4.6) eşitliklerindeki türevlerin yerine x ve y yönündeki gradyanları ifade etmeleri nedeniyle sırasıyla ∇x ve ∇y, ikisinin bileşeni için ise ∇f kısaltmaları kullanılabilir. (4.5) ve (4.6) eşitliklerinin basit şekilde uygulanabilmesi için her bir eksen boyunca aşağıdaki kesikli evrişim maskesinin kullanılması yeterli olacaktır.

-1 | 0| 1

Bu konudaki değişik yaklaşımlar ile operatör adı verilen farklı maskeler oluşturulmuştur. Bu yaklaşımlarda 3x1 lik komşuluk ilişkileri daha da genişletilerek,

3x3 veya daha fazla sayıda komşulukları dikkate alan operatörler de geliştirilmiştir. Operatörler yardımı ile basit evrişi mişlemi aşağıdaki eşitlik ile yapılabilir:

$$T \cdot I(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} T(i,j) \cdot I(x+i, y+j) \qquad (4.7)$$

(4.7) eşitliğinde, $I(x,y)$ görüntün piksel elemanlarını, $T(i,j)$ ise nxm boyutundaki evrişim operatörünün elemanlarını temsil etmektedir. Birinci derece türevle ilgili olarak söz konusu operatörlerden yaygın kullanılanları Roberts, Prewitt, Sobel ve izotropik (Frei-Chen) operatörleridir. Bu operatörlerden Robert çapraz operatörü, diğerlerine göre daha küçük etki alanına sahip olup, görüntü üzerindeki kirliliklerden kolay etkilenmektedir. Robert operatörleri, gradyan belirleme yönleri ve noktaları aşağıdaki şekildedir:

$$\begin{bmatrix} \mathbf{0} & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} \mathbf{0} & 1 \\ -1 & 0 \end{bmatrix}$$

Aşağıda, gradyan belirleme yön ve noktaları verilen Prewit operatörleri köşegenlerden çok yatay ve düşey kenarlarda daha duyarlıdırlar:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \downarrow$$

Yatay ve düşey kenarlardan çok köşegenlere daha duyarlı olan Sobel operatörü, gradyan belirleme yönleri ve noktaları aşağıdaki şekildedir:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \downarrow$$

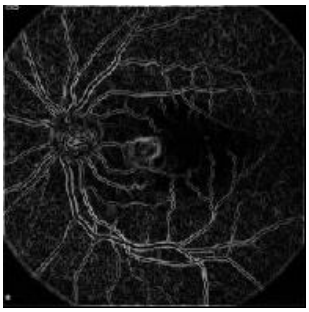
Isotropik operatör olarak da adlandırılan Frei-Chen operatörleri, gradyan belirleme yönleri ve noktaları ise aşağıdaki şekildedir:

$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \mathbf{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \quad \downarrow$$

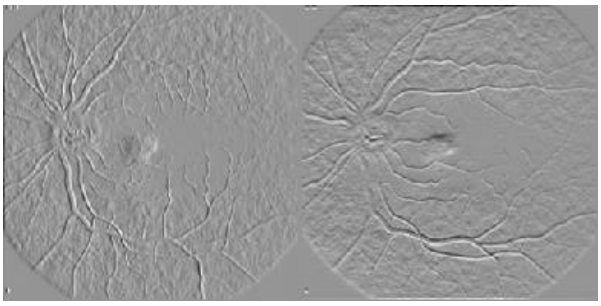
$\longrightarrow$

Bu operatörler arasında kenar bulmada en yaygın kullanılanı ise Sobel operatörüdür. Bir göz retinasındaki damarlara ait kenarların tespitinde Sobel operatörünün kullanılması ile elde edilen sonuçlar Şekil 4.3'de görülmektedir [19].

Orijinal Görüntü



∇f ile elde edilen görüntü



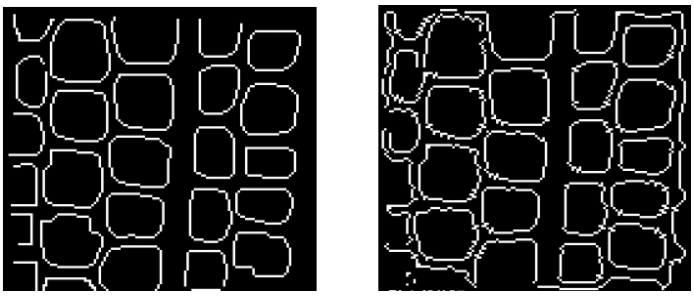
∇x bileşeni ile elde edilen görüntü, ∇y bileşeni ile elde edilen görüntü

Şekil 4.3 Sobel operatörü ile kenar bulma [19].

İkinci derece türevle ilgili olarak ise daha çok Laplacian operatörü kullanılmaktadır. Kenarların birkaç piksel kalınlığında olması durumunda, bu kalınlığın ortasından geçen çizginin bulunması amacıyla ikinci dereceden türevleri kullanan operatörlerden faydalanılmaktadır. İkinci dereceden türevlerin bir diğer avantajı kenar çizgilerinin kapalı şekiller oluşturmasıdır. Bu durum görüntü bölütlemede çok önemlidir. Yaygın olarak kullanılan üç Laplacian operatörü aşağıda verilmiştir.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Laplacian operatörünü diğer operatörlerden ayıran en önemli özellik onun çok yönlü olmasıdır. Bu operatör her yöndeki kenarları ortaya çıkarmaktadır. Laplacian operatörü diğer operatörlere nazaran daha keskin kenarlar belirlemektedir. Laplacian ve Sobel operatörleri kullanılarak Şekil 4.1(c)'deki görüntü için yapılan kenar bulma işlemleri sonucu elde edilen veriler Şekil 4.4'de gösterilmiştir [19].



Laplacian operatörü ile Sobel operatörü ile

Şekil 4.4 Kenar bulma operatörlerinin karşılaştırılması [19].

4.3. Üç Boyutlu Bölümleme

Üç boyutlu bölümleme, bir obje içerisinde farklı yapıda olan ve farklı özellik taşıyan bölgeler arasındaki sınır veya kenar piksellerin belirlenmesi işlemidir. Üç boyutlu bölümleme, yukarıda sıralanan operatörlerin üçüncü boyuta genişletilmiş şekillerinin kullanılması ile yapılmaktadır.

Genellikle, hesaplamalarda pratiklik sağlamak açısından, algoritmalarda; her üç eksen yönünde komşu piksel değerleri arasındaki farklılıklardan yararlanan gri-seviyeli gradyan hesabı kullanılmaktadır. Bu hesaplamada; (x,y,z) konumlu bir pikselin gradyan değeri (4.8) eşitliği ile hesaplanır [31].

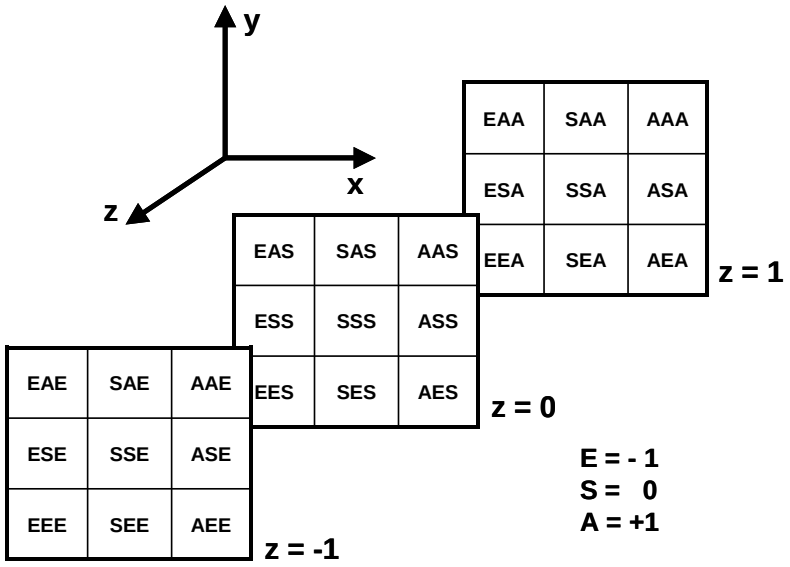
$$\nabla f(x,y,z)=\begin{bmatrix}\nabla f_x\\ \nabla f_y\\ \nabla f_z\end{bmatrix}=\frac{1}{2}\begin{bmatrix}f(x+1,y,z)-f(x-1,y,z)\\ f(x,y+1,z)-f(x,y-1,z)\\ f(x,y,z+1)-f(x,y,z-1)\end{bmatrix} \tag{4.8}$$

(4.8) eşitliğinin basit şekilde uygulanabilmesi için her bir eksen boyunca kesikli evrişim maskelerinin kullanılması yeterli olacaktır. Sobel tarafından, 3x3x3 lük bir gradyan operatörünü geliştirmek maksadıyla, 26 komşu pikseli birbirine birleştiren ve küpün merkezinden geçen muhtemel 13 farklı doğrultuda renk değişiminden yararlanılmıştır.

Bu yaklaşımda, bu doğrultular için koordinat eksenleri boyunca yönlü türev alınmış, bu türevlerin vektör toplamından faydalanılmış olup, küp üzerindeki konumları Şekil 4.5'de olduğu gibi harfler ile kodlanmış **g** gradyan vektörünün yönlü türevinin belirlenmesi için aşağıdaki eşitlikten yararlanılmıştır [19]:

$$|\mathbf{g}| = \langle \text{ışık yoğunluğu farkı} \rangle / \langle \text{komşuya uzaklık} \rangle \tag{4.9}$$

Bu yaklaşımda, komşu piksele olan birim vektör **g** gradyan vektörünün doğrultusunun temsilinde kullanılarak gradyan değeri için aşağıdaki vektör toplamı kullanılmıştır:



Şekil 4.5 Üç boyutlu gradyan hesabı için kodlama [19].

```
G=        (AAS-EES)/4 * [+1,+1, 0]
 +        (SAS-SES)/2 * [ 0,+1, 0]
 +        (EAS-AES)/4 * [-1,+1, 0]
 +        (ASS-ESS)/2 * [+1, 0, 0]
 +        (SSA-SSE)/2 * [ 0, 0,+1]
 +        (SAA-SEE)/4 * [ 0,+1,+1]
 +        (SEA-SAE)/4 * [ 0,-1,+1]
 +        (ASA-ESE)/4 * [+1, 0,+1]
 +        (ASE-ESA)/4 * [+1, 0,-1]
 +        (AAA-EEE)/6 * [+1,+1,+1]
 +        (AEA-EAE)/6 * [+1,-1,+1]
 +        (EEA-AAE)/6 * [-1,-1,+1]
 +        (EAA-AEE)/6 * [-1,+1,+1]
```

(4.10)

(4.10) eşitliğinin hesaplanmasında karekök işlemleri kullanılmamıştır. Bu eşitlikte x değerine etki eden vektörlere ait elemanların kullanılmasıyla aşağıdaki biçimde gradyanın *x* bileşeni elde edilebilir:

$$
\begin{aligned}
G_x = \quad & (AAS-EES)/4 - (EAS-AES)/4 + (ASS-ESS)/2 \\
+ \quad & (ASA-ESE)/4 + (ASE-ESA)/4 + (AAA-EEE)/6 \\
+ \quad & (AEA-EAE)/6 - (EEA-AAE)/6 - (EAA-AEE)/6
\end{aligned}
\tag{4.11}
$$

(4.11) eşitliğinin basit şekilde uygulanabilmesi için her bir doğrultu boyunca ağırlıklandırılmış ışık yoğunluklarından oluşan, kesikli evrişim maskesinin kullanılması yeterli olacaktır. (4.11)'deki eşitliği ifade edebilmek amacıyla G_x bileşeni için bu maske aşağıdaki şekilde tanımlanmıştır [32].

$$
G_x = \begin{bmatrix} -2 & 0 & 2 \\ -3 & 0 & 3 \\ -2 & 0 & 2 \end{bmatrix}, \quad \begin{bmatrix} -3 & 0 & 3 \\ -6 & 0 & 6 \\ -3 & 0 & 3 \end{bmatrix}, \quad \begin{bmatrix} -2 & 0 & 2 \\ -3 & 0 & 3 \\ -2 & 0 & 2 \end{bmatrix}
$$

$$
z = -1 \qquad z = 0 \qquad z = +1
$$

Her ne kadar görüntü bölümleme üç boyutlu veri işleme ve görüntülemenin temel işlemadımlarından biri ise de, araştırmacıların yoğun çabalarına rağmen bu işlemi hatasız ve tamotomatik olarak yapabilecek bir uygulama henüz geliştirilememiştir. Bölümlemenin en iyi şekilde yapılabilmesi için genellikle kullanıcı etkileşimi gerektiren yarı otomatik algoritmalar geliştirilmektedir.

Tüm bu otomatik ve yarı-otomatik tekniklere rağmen, iyi sonuçların alınması için en iyi yöntem el ile yapılan bölümlemedir. Operatörlerin; karmaşık fizyolojik kavrama yeteneklerinin yanında, anatomik beceri gibi model-tabanlı ilave bilgilerini kullanabilmeleri nedeniyle, el ile bölümlemede daha iyi sonuçlar alınabilmektedir [18].

5. DÜZEY KÜMESİ YÖNTEMLERİNİN TEORİ VE ALGORİTMALARI

5.1. Giriş

Dünya üzerindeki tüm nesneler fiziksel sınırlara sahiptir. Bu sınırları; dinamik ve statik olmak üzere iki ayrı sınıfta değerlendirmek mümkündür. Dinamik sınırlara örnek olarak okyanustaki dalgaların kırılması verilebilir. Hava fotoğraflarındaki detayların sınırları ise statik sınırlara örnek olarak gösterilebilir. Sınırlarla gösterilen her bir nesne bir yüzeydir. Bu yüzeyler gelişen ya da büyüyen yüzeyler olarak kabul edilirse; bunların gelişme hızları ve yönlerinin çok iyi bilindiği durumlarda bile şekillerinin doğru olarak izlenmesi oldukça zordur. Birinci zor nokta; bir kar tanesindeki kadar karışık şekillerin keskin köşelerinin belirlenmesidir. İkinci zor nokta ise uzak kenarların birbirlerine dolanması ve karışması durumunda ortaya çıkan problemin çözümüdür. Bu probleme örnek olarak, bir orman yangının sınırlarının, birbirinden ayrı alevlerin birlikte yanması ve kıvılcımların rüzgarla birlikte saçılmasıyla değişiklik göstermesi verilebilir. Üçüncü zor nokta ise, uygun bir şekilde temsil yolu bulunması durumunda bile üç ve daha büyük boyutlu dalgalı bir sınırdan söz etmenin zorluğudur [33].

Düzey kümesi yöntemleri, keskin köşeli, topolojik değişimlere uğramış ve üç boyutlu gelişen yüzeylerin izlenmesine matematiksel ve programlanabilir araçlar sağlamaktadırlar. Bunlarla birlikte; engeller arasında robotlar için en uygun güzergahı belirlemede, görüntü işleme aracı olarak ve dijital görüntülerden detay belirlemede kullanılabilmesi olanaklıdır.

Düzey kümesi yöntemlerinin formülasyonlarının anlaşılabilmesi için bir örnek vermek faydalı olabilir. Bir parça ip düşünülsün ve her iki ucu birbirine yapıştırıldıktan sonra yere rastgele atılsın. Bu aradaipin dolaşmasına ve kendi üstüne geçmesine engel olunsun. Böyle bir ip parçasının yerde oluşturduğu şekle basit kapalı eğri adı verilmektedir (Şekil 5.1) [33].

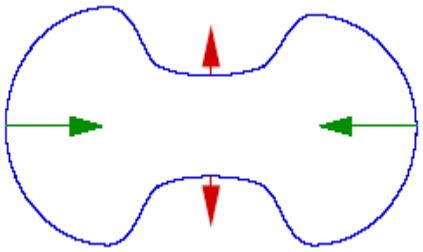
Bir eğrinin karakteristik özelliği, her hangi bir noktadaki eğriliğidir. Eğrilik, eğrinin kıvrılma hızını gösterir. Örneğin bir dairenin sabit bir eğriliği vardır çünkü her zaman

aynı hızda ve oranda kıvrılmaktadır. Daha küçük bir dairenin ise büyük daireden daha hızlı kıvrıldığından daha büyük bir sabit eğriliği vardır.

(a) (b)

Şekil 5.1 (a) Basit kapalı bir eğri. (b) Kapalı olmayan bir eğri [33].

Eğrinin her bir parçasının, eğrilikle orantılı bir hızda eğriye dik olarak hareket ettiği varsayılsın. Eğrinin saat yönünde veya saat yönünün tersinde hareket etmesine bağlı olarak eğrilik pozitif ya da negatif değer alabilir. Bunun sonucunda eğrinin bazı bölümleri dışa hareket ederken diğer bölümleri de içeri doğru hareket eder. Şekil 5.2'deki gibi kırmızı oklar eğriliğin negatif olduğu bölgeleri, yeşil oklar ise eğriliğin pozitif olduğu bölgeleri göstermektedir. Oklar farklı uzunluktadırlar çünkü yeşil oklardaki kuvvet büyüklüğü, kırmızıdakilerden daha büyüktür [33].

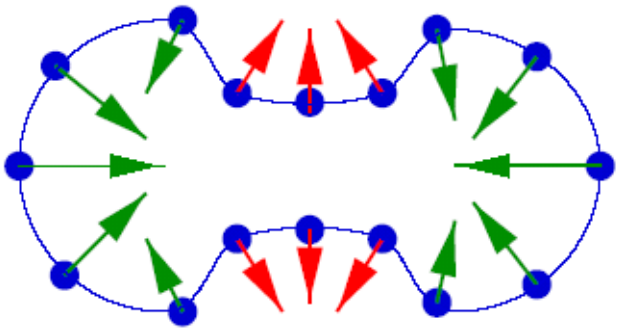


Şekil 5.2 Eğriliğe bağımlı olarak değişen hareketlerin yön ve büyüklükleri [33].

Bir yüzey boyunca yürüdüğünü ve yürürken de geçilen noktaların x ve y koordinatlarını, xy düzleminde bu noktaları çizen bir başka kişiye söylendiği varsayılsın. Yürüme hızı "parameterizasyon", diğer kişi tarafından noktaların birleştirilmesiyle çizilen eğriye de "görüntü" adı verilir. Hızlı veya yavaş olarak yürünse de hep aynı sınır çizilecektir. Bu yaklaşımın avantajı, koordinat sistemine olan bağımlılığın ortadan kaldırılmasıdır.

Bir yüzeyin, parameterize edilmiş sunumunun bir sayısal algoritmanın omurgasını oluşturacak şekilde kullanıldığı düşünülebilir. Eğrinin etrafında yürünsün ve düzenli

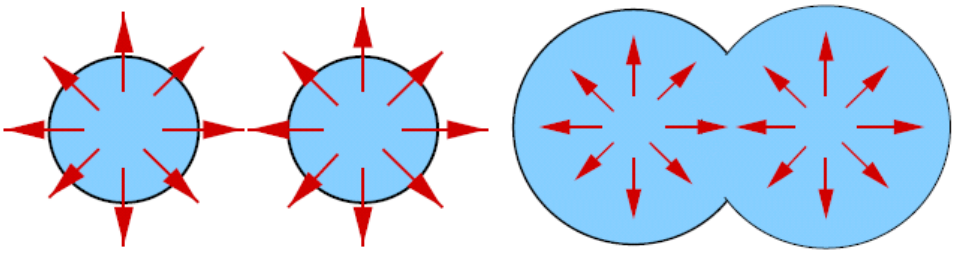
aralıklarla mavi bir işaret yerleştirilsin ve bu işaretler bir iple birbirine bağlansın. İşaretler ve bağlı oldukları ip bir sınır oluşturacaktır (Şekil 5.3) [33].



Şekil 5.3 Birbirine bağlı mavi işaretlerin oluşturduğu eğrinin sınırı [33].

Okların uzunlukları ve yönleri bulundukları noktalardaki eğriliklerle belirlenir. Burada şöyle bir genişleme ya da ilerleme stratejisi uygulanabilir. Okların yönlerine bakarak işaretler ilerletilebilir ve yeni oklar hesaplanabilir ve işaretler yeniden ilerletilebilir. Böylece ne kadar çok işaret kullanılırsa, o kadar doğru bir cevap alınacağı beklentisi içine girilebilir.

Ne yazık ki bu yaklaşım, bazıları kaçınılmaz bazıları da düzeltilebilir bir takım hatalar içermektedir. Şekil 5.3'e dikkatlice bakıldığında düzeltilebilecek hatalardan birini görmek mümkündür. İşaretler birbirlerinin üzerine geçmeye çalışacaklar ve geçeceklerdir. Bu durumda işaretleri birbirine bağlayan ipi düzenli bir şekilde bir arada tutmak zorlaşacaktır, ip dolanacak ve işaretler birbirlerine karışacaklardır. Bir çözüm olarak; yüzeyin ilerlemesini periyodik olarak durdurmak ve eğri boyunca yeniden yürüyerek yeni eşit aralıklı işaretler yerleştirmek verilebilir.

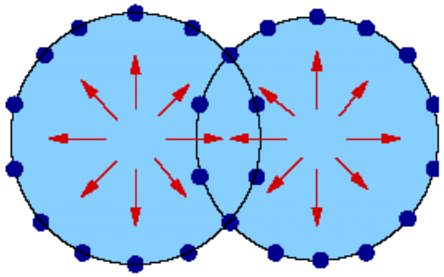


(a) (b)

Şekil 5.4 (a) Başlangıç alevleri (b) Yayılan alevlerin sonraki zamandaki durumu [33].

Yayılan sınırın topolojisinin değiştirilmesi istendiğinde ise çok daha ciddi bir problem ortaya çıkacaktır. İki ayrı dairesel alev ele alınsın. Her ikisinin de sabit hızda dışarıya doğru yandığı ve genişlediği varsayılsın. Yayılan yüzeyin şekli kolayca öngörülebilir (Şekil 5.4) [33].

İki ayrı alev birlikte büyüdüğünden, gelişen yüzeyler birleşerek tek bir yayılan yüzey oluşturmaktadırlar. Bununla birlikte; parameterizasyona dayanan bir sayısal algoritma burada büyük bir hataya neden olacaktır. İlerleyen alevin gerçek kenarı izlenmek isteniyorsa iki alevin kesişim bölgesindeki iki çift işaret bir şekilde ortadan kaldırılmalıdır (Şekil 5.5) [33].

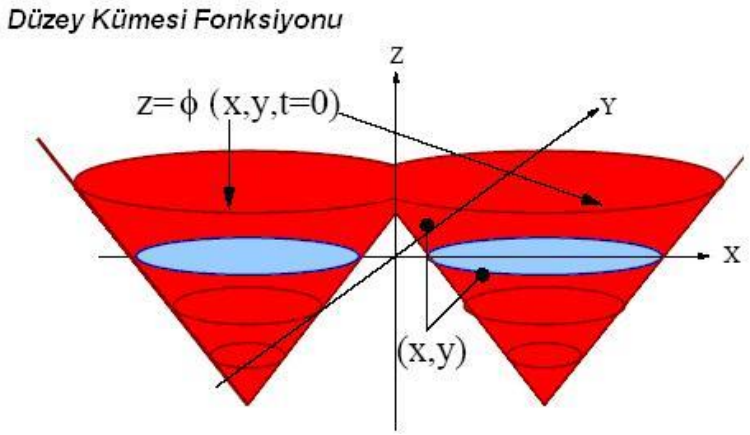


Şekil 5.5 Yayılan yüzey üzerindeki işaretler [33].

Düzey kümesi yaklaşımı bu probleme farklı bir çözüm getirmektedir. Arayüzeyin kendisini izlemek yerine, orijinal arayüzey ele alınmakta ve probleme ilave bir boyut daha eklenmektedir. Arayüzeyin içinde bulunduğu xy düzlemine ek olarak yüksekliğin ölçüldüğü z boyutunun da dahil edildiği yeni bir koordinat sistemi oluşturulmaktadır. Problem iki boyutlu bir problemden üç boyutlu bir probleme dönüştürülmüş olmaktadır.

$z=\phi(x,y,t=0)$ olan bir fonksiyon düşünülsün. Bu fonksiyonda (x,y) koordinatlarıyla temsil edilen bir nokta, girdi verisi olarak alınmakta ve bu noktaya z yüksekliği atanmaktadır. Z yüksekliği, t=0 anındaki arayüzeyi ilet anındaki arayüzey arasındaki uzunluğu ifade etmektedir. Başka bir deyişle t anındaki arayüzeyin t=0 anındaki arayüzeyden olan yüksekliği z değeri ile ifade edilmektedir. Böylece xy düzlemini arayüzeyde mutlak olarak kesen ve kırmızı ile renklendirilmiş bir yüzey oluşturulmaktadır (Şekil 5.6) [33]. Kırmızı yüzey, düzey kümesi fonksiyonu olarak isimlendirilmektedir çünkü düzlem içerisindeki herhangi bir noktayı girdi olarak kabul etmekte ve çıktı olarak bir yükseklik değeri vermektedir. Mavi arayüzey sıfır

düzey kümesi olarak adlandırılmaktadır çünkü yüksekliği sıfır olan tüm noktaların birleşiminden oluşmaktadır (Şekil 5.6) [33].



Şekil 5.6. Düzey kümesi yüzeyi (kırmızı) her (x, y) noktasından mavi arayüzeye olan uzaklığı çıktı olarak vermektedir [33].

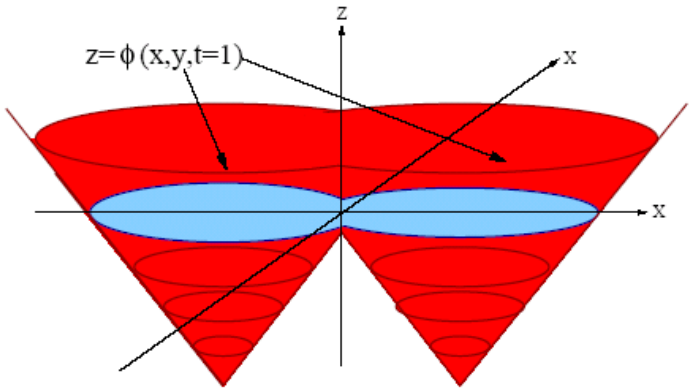
Arayüzeyin gelişimini zamanında yakalayabilmek için $\phi(x,y,t)$ yüzeyinin yüksekliğinin nasıl değiştirileceği belirlenmelidir. Düzey kümesi fonksiyonunun genişlemesine, yükselmesine, alçalmasına izin vermek ve bütün işlemleri yerine getirmesini sağlamak amaç olarak belirlenmektedir. Arayüzeyin herhangi bir zamandaki konumunu bulmak için yüzey sıfır yükseklikte kesilmektedir. Başka bir deyişle sıfır konturu çizilmektedir.

İlk bakışta, hareket eden eğrinin problemini, hareket eden yüzeyinkiyle değiştirmek akıllıca görünmeyebilir. Daha çok boyut daha çok iş demektir denebilir fakat burada dahil edilen ek boyut o kadar yararlıdır ki çarpışabilen ve birbirinden uzaklaşabilen işaretler yerine, şimdi her (x, y) noktasında durulup, düzey kümesi fonksiyonun o noktadaki yüksekliği ayarlanabilecektir. Böylece topolojik hatalar ortadan kaldırılabilecektir. Yayıldıklarında birleşerek tek bir alev meydana getiren iki ayrı alevin belirli bir zamandaki sıfır düzey kümeleri iki değil sadece bir eğri oluşturacaktır (Şekil 5.7) [33].

Basketboldan örnek verilecek olunursa, işaretleme yöntemleri adam adama savunmaya, düzey kümesi yöntemleri ise alan savunmasına benzetilebilir.

Düzey kümesi yaklaşımının üç veya daha büyük boyutlu problemlerin çözümünde kullanılmasının hiçbir farkı yoktur. Daha büyük boyutlu problemlerde görüntülemenin zorlaşması dışında stratejide hiçbir değişiklik yoktur. Dikkat

edilmesi gereken konu hangi boyutta çalışılırsa çalışılsın düzey kümesi fonksiyonu oluşturulurken ilave olarak bir boyut eklenmektedir.

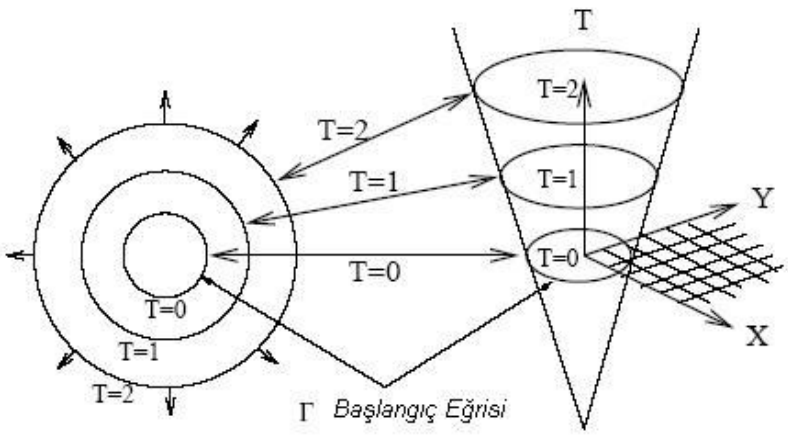


Şekil 5.7 Kırmızı düzey kümesi fonksiyonu hareket ettirilerek yeni mavi arayüzey oluşturulur [33].

Düzey kümesi yaklaşımları ile birlikte dar bant uygulamaları ve hızlı ilerleme yöntemleri gibi daha gelişmiş düzey kümesi uygulamalarının teori ve çözümleri hakkında daha detaylı bilgiler ilerleyen bölümlerde ele alınacaktır.

5.2. Yüzey Yayılımında Başlangıç ve Sınır Değer Yaklaşımları

Yüzeylerin hangi hızda, hangi yönde ve nereye kadar yayılabileceği sorularının çözümü için iki yaklaşım öne sürülmüştür. Başlangıç değeri ve sınır değeri yaklaşımları. Bu yaklaşımların teori ve çözümleri bu bölümün temel konusu olarak ele alınacak ve açıklanmaya çalışılacaktır.

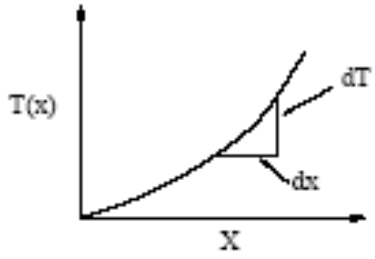


Şekil 5.8 Yüzeyin hareketinin sınır değeri problemine dönüştürülmesi [34].

5.2.1. Sınır değeri yaklaşımı

F hızıyla, kendisine dik bir düzlemiçerisinde yayılan kapalı bir Γ eğrisini ele alalım. F>0 olarak kabul edersek, yüzey her zaman dışa doğru hareket edecektir. Bu genişleyen yüzeyin konumunu tanımlamanın bir yolu Şekil 5.8'de de gösterildiği üzere, yüzeyin her (x,y) noktasından geçişindeki, T(x,y) varış zamanını hesaplamaktır [34].

T(x,y) varış yüzeyini tanımlayan denklem kolayca türetilebilir. Uzaklık=hız*zaman eşitliğini kullanarak $d\chi = F(dT)$ ve buradan $1 = F\frac{dT}{dx}$ sonucunu elde edebiliriz (Şekil 5.9) [34].



Şekil 5.9 Sınır değeri formülasyonu için hazırlık [34].

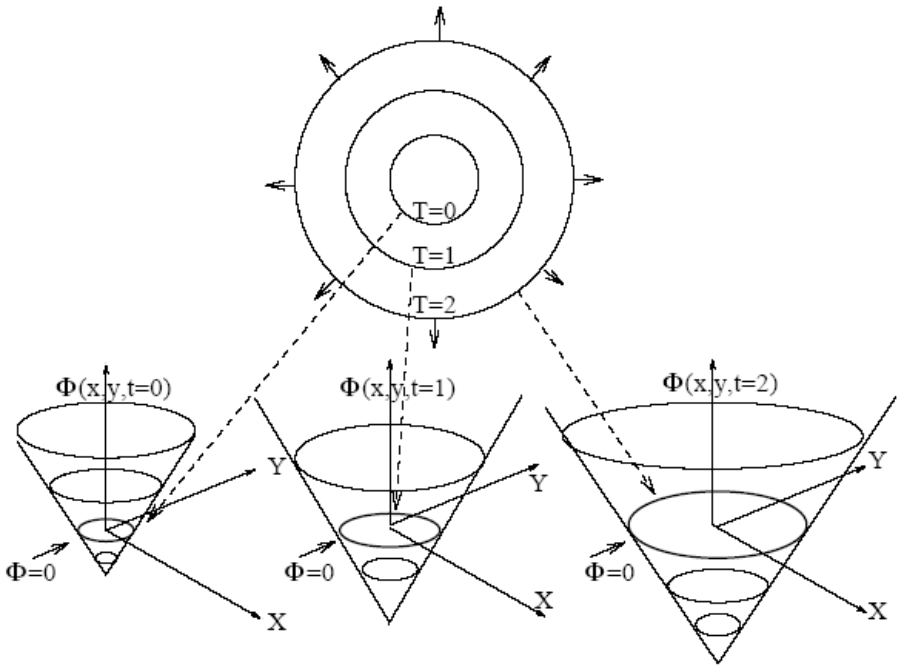
T sonuç yüzeyinin kısmi türevi çok boyutlu durumlar düşünüldüğünde, gradyan olarak alınabilir ve buradan aşağıdaki eşitlik elde edilir.

$$\Gamma \; egrisi \;\; üzerinde \;\; T = 0 \;\; aninda \;\; |\nabla T| F = 1 \;\; dir. \qquad (5.1)$$

Böylece, yüzey hareketi, sınır değeri probleminin bir çözümü olarak tanımlanabilir; eğer F hızı sadece konuma bağlı olarak değişiklik gösterirse, eşitlik çok bilinen Eikonal denklemine indirgenebilir [34].

5.2.2. Başlangıç değeri yaklaşımı

Yüzeyin başlangıç konumunun, daha yüksek dereceden bir ϕ fonksiyonunun sıfır düzey kümesi olarak alındığı varsayılsın. Böylece ϕ fonksiyonunun gelişimi zamana bağımlı başlangıç değeri problemi sayesinde yüzeyin yayılmasıyla tanımlanabilir. Yüzey, herhangi bir zamanda, zamana bağımlı düzey kümesi fonksiyonu ϕ'nin sıfır düzey kümesiyle verilebilir (Şekil 5.10) [34].



Şekil 5.10 Yüzey hareketinin başlangıç değeri problemine dönüşümü [34].

Bu ϕ düzey kümesi fonksiyonu için hareketin denklemini elde etmek için (5.2) eşitliği ile verilen, gelişen ϕ fonksiyonunun sıfır düzey kümesi her zaman yayılan hiper yüzey ile çakışır şartını dikkate almak gerekir.

$$\phi(\chi(t),t)=0 \tag{5.2}$$

zincir kuralı ile (5.3) eşitliği elde edilebilir.

$$\phi_t+\nabla\phi(\chi(t),t).\chi'(t)=0 \tag{5.3}$$

F, dışa doğru olan normal hızı temsil ettiğinden, $\chi'(t).n=F$ diyebiliriz. Burada $n=\nabla\phi/|\nabla\phi|$ 'dir ve bunun sonucunda ϕ için bir gelişme eşitliği elde edilir. Verilen

$$\phi(\chi,t=0)\ için \tag{5.4}$$

$$\phi_t+F|\nabla\phi|=0 \tag{5.5}$$

Bu eşitlik, Osher ve Sethian tarafından ortaya konan düzey kümesi eşitliğidir [39]. F hız fonksiyonunun belirli formları için standart Hamilton-Jacobi eşitliği elde edilebilir.

Bu eşitlik, ϕ düzey yüzey fonksiyonunun zamanla gelişimini, bu gelişen fonksiyonun sıfır düzey kümesinin her zaman yayılan arayüzey ile tanımlanmasıyla açıklamaktadır.

Ortaya konan bu formülasyonlar (5.6) da özetlenmiş olup, bunların çözümü gerekmektedir.

Başlangıç Değeri Formülasyonu	***Sınır Değeri formülasyonu***	
$\phi_t + F\lvert\nabla\phi\rvert = 0$	$\lvert\nabla T\rvert F = 1$	
$Yüzey = \Gamma(t) = \{(x,y) \mid \phi(x,y,t) = 0$	$Yüzey = \Gamma(t) = \{(x,y) \mid \Gamma(x,y) = t$	(5.6)
(F keyfi olarak seçilir)	$(F > 0)$	

5.2.3. Yaklaşımların avantajları

Yayılan yüzeyler üzerine ortaya atılan bu iki yaklaşıma aşağıda sıralanan avantajlar elde edilir:

- Her iki yaklaşımda çok boyutlu ortamlarda, diğer bir deyişle üç boyutlu ve daha üst boyutlu ortamlarda değişiklik göstermezler.
- Γ gelişen yüzeyindeki topolojik değişimler doğal olarak ele alınır. Yüzeyin t anındaki konumu ya gelişen düzey kümesi fonksiyonunun $\phi(\chi, y, t) = 0$ sıfır düzey kümesi ile ya da sınır değeri çözümünün T=t konturuyla verilir. Bu küme bağlanmak zorunda değildir ve t aralıklarıyla birleştirilebilir ve kırılabilir. Her iki durumda da, asıl anahtar nokta şudur: T(x,y) sınır değeri çözümü ve ϕ düzey kümesi fonksiyonu her zaman tek değer alır.
- Her ikisi de, uygulanabilir, hesaplanabilir ve programlanabilir stratejilerin kullanımını etkili kılmaktadırlar.

Aynı zamanda her iki yaklaşım arasında önemli farklarda mevcuttur. Bu farklar aşağıda anlatılmaya çalışılmıştır:

- İki yaklaşım arasındaki en belirgin farklılık, başlangıç değeri düzey kümesi yaklaşımının hem pozitif hem de negatif F hız fonksiyonlarına izin vermesidir. Böylece yüzey yayılma sırasında hem ileriye hem de geriye doğru hareket edebilir. Sınır değeri yaklaşımı ise yüzeyleri, her zaman aynı yönde yayılmaları konusunda zorlar ve diğer yönde yayılmaya izin vermez. Çünkü bu yaklaşıma göre yayılan yüzey, her bir grid noktasından sadece tek bir T geçiş zamanında ve tek bir kez geçilebilir.
- Eğrilerde olduğu gibi daha karmaşık F hız fonksiyonlarının yakınsaması, genelde başlangıç değeri düzey kümesi yaklaşımıyla gerçekleştirilir. Örneğin, düzey kümesi formülasyonunda, bir noktanın normal vektörü (5.7) eşitliğinde olduğu gibi verilir.

$$n = \frac{\nabla \phi}{|\nabla \phi|} \tag{5.7}$$

Her düzey kümesinin eğriliği, birim normal vektörün yüzeye uzaksamasından elde edilebilir.

$$K = \nabla . \frac{\nabla \phi}{|\nabla \phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_x \phi_y \phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}} \tag{5.8}$$

- Diğer taraftan; konuma bağlı olan ve/veya çok büyük değişimler gösteren F hız fonksiyonlarının çözümünün ise genellikle hızlı ilerleme yöntemlerinin kullanılmasıyla başka bir deyişle sınır değeri yaklaşımıyla gerçekleştirilmesi gerektiği söylenebilir. Çünkü:
  - Hızlı ilerleme yöntemleri zaman adımları kullanmazlar.
  - Yığın sıralama algoritmalarının kullanılmasıyla, hızlı ilerleme yöntemleri, hesaplama ve programlama tekniklerine, düzey kümesi yöntemlerinden daha elverişlidir.

5.2.4. Sınır ve başlangıç değeri yaklaşımlarının çözümü

Yukarıda sözü geçen her iki yaklaşımı, aşağıdaki gibi genel kısmi diferansiyel eşitliği altında toplayabiliriz.

$$\alpha u_t + H(Du, x) = 0 \qquad (5.9)$$

Bu eşitlikte, Du, u_x ve u_y gibi, u'nun her değişkene göre kısmi türevlerini ifade etmektedir. Örneğin, Eikonal eşitlik durumunda α=0 dır ve H fonksiyonu H=F$|\nabla\phi|$ fonksiyonuna indirgenebilir.

Yukarıdaki eşitliklerin çözümünde karşılaşılan en büyük zorluklardan bir tanesi, keyfi yumuşak (smooth) sınır verisiyle bile çözümün diferansiyellenebilir olması gerekmediğidir. Burada amaç, fiziksel olarak doğru, yumuşak olmayan çözümler elde etmektir. Bu çözümlere ulaşmak için gerekli olan etkili ve doğru yaklaşma tasarılarının inşası için diferansiyellenememezlik durumunu çözecek sayısal teknikler oluşturmak gerekmektedir.

5.3. Tekillik Geliştirme, Akışkanlık (Viscousity) Çözümleri ve Sayısal Kabuller

5.3.1. Temel eşitlikler ve diferansiyellenememezlik durumu

Temelini basit Eikonal eşitlikten alan bir örnekle başlanabilir.

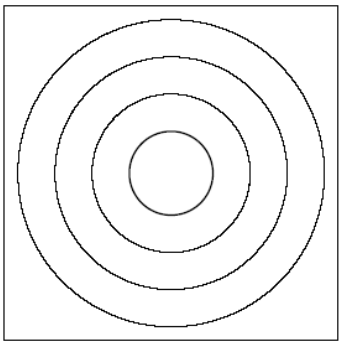
$$|\nabla u| = f(x,y) \qquad f=1 \qquad (5.10)$$

Çözüm yumuşak başlangıç verilerine rağmen diferansiyellenemeyecektir. Durumun canlandırılması amacıyla;.

birim dairenin dışında	*birim daire üzerinde*
$\|\nabla u\| = 1$	$u = 0$

(5.11)

olduğu varsayılsın. Birim çemberinin uzunluk fonksiyonuna karşılık gelen $u(x,y) = (x^2 + y^2)^{1/2} - 1$ fonksiyonunun verilen Eikonal eşitliği çözdüğü görülmüştür. Bununla beraber, fonksiyonun sağ el $\int(x)=1$ ve sınır değer olarak sıfırı sağlayan bir yapıda olması çözümün başlangıç Γ eğrisine olan uzaklık olduğunu göstermektedir. Çözüm Şekil 5.11'deki konsantrik daireler ailesi olarak gösterilebilir. Bu, sonucun her noktada diferansiyellenebileceğinin kolayca denetlenmesini sağlar [34].

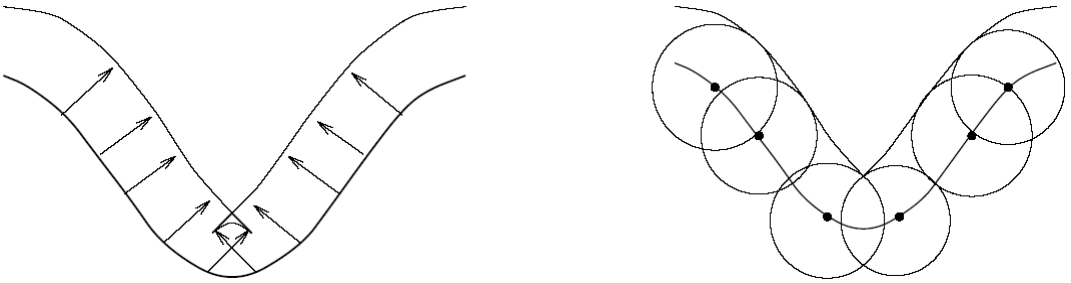


Şekil 5.11 $|\nabla u| = 1$ Eikonal eşitliğine olan uzaklık fonksiyonun çözümü [34].

Buna karşılık; $y = x^2$ parabolü olarak verilen sınır verisi şimdi ele alınsın ve parabolün üst bölgesindeki herhangi bir noktadan sınır eğrisine (parabolün kendisi) olan uzaklığın bulunmaya çalışıldığı varsayılsın. Sınır eğrisinin her noktasına bir parçacık koyulsun ve her parçacığın parabole dik olacak şekilde dışarıya doğru sabit bir hızla hareket ettirildiği varsayılsın. Herhangi bir C anında söz konusu parçacıkların konumları, sınır eğrisinden C uzaklığına sahip tüm noktaların oluşturduğu kümeyi verir. Bu çözüm, kırlangıç kuyruk (swallowtail) grafiği olarak gösterilebilir (Şekil 5.12(a)) [34]. Bu çözüme, sınır verisi üzerindeki her bir noktaya çizilecek diklerle (normal) ulaşılır ve her bir dik boyunca u uzaklığında yayılır.

Bu çözüm her ne kadar olanaklı ise de sınır verisine yakın noktalarda problemleri vardır. Yukarıda anlatılan yapı, sınır eğrisine her noktadan yerel bir C uzaklığında bulunan bir nokta kümesi oluşturur. En yakın noktaların gerçek kümesini veren farklı bir yaklaşım, verilen C uzaklığında eğriye en yakın noktaların oluşturduğu kümenin sorgulanmasından ortaya çıkar. Bu çözümü oluşturmanın bir yolu Huyghen prensipleriyle oluşturulan bir yapıya dayanır. Sınır eğrisi üzerindeki her noktada birim hızda dairelerin (dalga yüzeyleri de olabilir) doğduğu varsayılırsa, bu dairelerin oluşturduğu zarfın her zaman ilk varışlara karşılık geldiği ve bununda otomatik olarak bir çözüm oluşturduğu görülebilir (Şekil 5.12(b)) [34]. Bu yaklaşım, Huyghens prensiplerini temel alan yüzey yayılımı için ortaya konan entropi koşulundan etkilenilerek ortaya atılmıştır [43].

Sınır eğrisi üzerinde aynı uzaklığa sahip olan iki nokta boyunca bir dik sırt meydana gelir (Şekil 5.12(b)). Bu sırt boyunca, çözüm diferansiyellenemez ve ∇u gradyanı tanımlanmaz [34].



(a) (b)

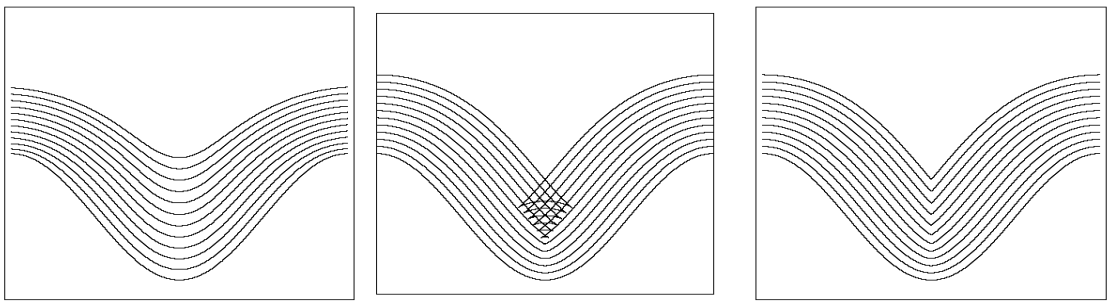
Şekil 5.12 $|\nabla u| = 1$ Eikonal eşitliğinin olası çözümleri. (a) Sınır verisinden "bölgesel uzaklıkta" olan noktalar (b) Global olarak sınır verisine en yakın noktalar [34].

Yukarıdaki her iki yapıda Eikonal eşitliğin çözümleri olarak düşünülebilir. Bununla birlikte, arzu edilen çözüm, en kısa mesafeye yada ilk varışa uygun olan Huyghens yapısı ile elde edilendir.

Bu entropi-sağlayıcı Huyghens yapısını elde etmenin bir başka yolu da eşitliğe bir yumuşaklaştırıcı teri meklemektir. (5.12) eşitliği ile verilen viskoz kısmi diferansiyel eşitliği ele alınsın,

$$|\nabla u(x)| = f(x) + \in \nabla^2 u \qquad (5.12)$$

$\in \nabla^2 u$ ile verilen viskoz terimi, çözümde keskin köşeleri yumuşatır ve Ω bütün alanında yumuşak olarak kalmasını garanti altına alır [43, 45]. $\in$ sıfıra gittiğinde çözüm, Şekil 5.12(b)'de verilen ilk varış çözümüne yakınsar. Böylece sınırlandırma durumları ortaya çıkar (Şekil 6) [34].



(a) (b) (c)

Şekil 5.13 $|\nabla u(x)| = f(x) + \in \nabla^2 u$ için $\in \to 0$ yaklaşırken akışkanlık (viskoz) sınırı (a) $\in = 0.005$ iken elde edilen yumuşatılmış çözüm (b) Kırlangıç kuyruk çözümü (c) $\in = 0$ iken elde edilen yumuşatılmış çözüm [34].

5.3.2. İlerleme yönünün tersindeki (upwind) tasarılar ve sayısal kabuller

5.3.2.1. İlerleme yönünün tersindeki tasarılar, kareselleştirme (quadrature)

Gradyan operatörün tahmini için ilerleme yönünün tersindeki tasarıların kullanımında (5.13) eşitliğiyle verilen tek boyutlu Eikonal eşitliği ele alınabilir.

$$u_x^2 = f^2(x) \qquad u(0) = 0 \tag{5.13}$$

Burada sağ el yönlü $f(x) > 0$ verilmiş olup, amaç $u(0) = 0$ sınır koşulundan uzak olan $u(x)$ fonksiyonunu oluşturmaktır. Unutmamak gerekir ki; bu fonksiyonun çözümü tek taraflı değildir. Eğer $u(x)$ problemi çözer ise $-u(x)$de çözer. Bundan sonra u'nun negatif olmayan çözümleri dikkate alınmayacaktır.

Çözümü oluşturmak için orijinden dışarıya doğru negatif ve pozitif x ekseni boyunca düşünülebilir. Aslında her bir problem ayrı ayrı çözülür. Birinci dereceden tahminler kullanılarak, kısaca basit diferansiyel eşitlikler çözülmeye çalışılır.

$$\frac{du}{dx} = \sqrt{f(x)} \qquad u(0) = 0 x \geq 0$$

$$\frac{du}{dx} = -\sqrt{f(x)} \qquad u(0) = 0 x \geq 0 \tag{5.14}$$

Basit diferansiyel eşitliğin sağ el tarafı sadece x'in bir fonksiyonudur ve esasen sayısal kareselleştirme gerçekleştirilmektedir. Standart sonlu fark görüşü, $u_i \approx u(i\Delta x)$ ve $f_i = f(i\Delta x)$, kullanılarak, iki problemin yaklaşımını Euler's yöntemini kullanarak çözmek mümkündür:

$$\frac{u_{i+1} - u_i}{\Delta r} = \sqrt{f_i} \quad i > 0$$

$$\frac{u_{i+1} - u_i}{\Delta r} = \sqrt{f_i} \quad i < 0 \tag{5.15}$$

Burada $u_0 = 0$'dır. Bu bir ilerleme yönünün tersindeki tasarıdır. Bu tasarıda; "ilerleme yönünün tersindeki" ya da sınır eğrisine doğru olan noktalar kullanılarak türevler hesaplanır.

5.3.2.2 İlerleme yönünün tersindeki tasarılar: Arayüzlerin geliştirilmesi

İlerleme yönünün tersindeki farkları kullanarak ileri gradyan tahmin motivasyonları, öğretici bir örnekten gelir. Bu pozisyonu her zaman, bir grafiğin fonksiyonları olarak tanımlanabilen gelişen eğri örneğidir. Başlangıç yüzeyi, $g(x)$'in grafiği olarak verilen bir eğri düşünelim; g ve g' [0,1] aralığında periyodik olsun. Yüzeyin F=1 hızıyla normali doğrultusunda yayıldığını ve her an için bir fonksiyon bıraktığını varsayalım. ψ, t anında yayılan fonksiyonun yüksekliği ise, $\psi(x,0) = g(x)$dir. Şimdi başlangıç değeri problemini ele alalım (Şekil 5.14) [34].

$$\psi_t = \left(1 + \psi_x^2\right)^{1/2}, \qquad \psi(x,0) = g(x) = \begin{cases} 1/2 - x & x \le 1/2 \\ x - 1/2 & x \rangle 1/2 \end{cases} \tag{5.16}$$

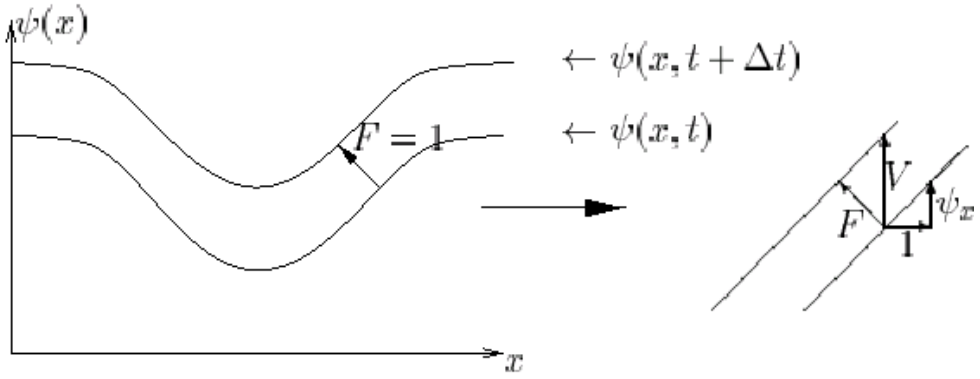
Başlangıç yüzeyi, (1/2,0)'da kesişen ışınlardan oluşan bir "V" şeklidir. Entropi koşulu ve Huyghens prensibi ile, herhangi bir t anındaki çözüm, "V" başlangıç yüzeyinden t uzaklığında konumlandırılmış noktaların oluşturduğu kümedir. Amaç, $\left(1 + \psi_x^2\right)^{1/2}$ terimi için sayısal kabullerin seçiminin bazı zorluklarının olduğunu göstermektedir.

Bir yaklaşım, $[0,1]$ aralığını 2M-1 noktaya bölmek ve (5.16) eşitliğindeki ψ_x kısmi türevi için merkezsel farklılık tahminleri oluşturmaktır. (5.17) eşitliğinde yapılan son tanımlamada merkezsel farklılık için standart isimlendirme kullanılmıştır.

$$\psi_t \approx \frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} = \left[1 + \left[\frac{\psi_{i+1}^n - \psi_{i-1}^n}{2\Delta r}\right]^2\right]^{1/2} = \left[1 + \left[D_i^{0x}\psi\right]^2\right]^{1/2} \tag{5.17}$$

$x_M = 1/2$ ve simetriden dolayı $\psi_M + 1 = \psi_M - 1$ ise, sağ el tarafı 1 dir. Bununla birlikte; grafiğin köşenin her iki tarafında da doğrusal olamaması durumunda her $x \neq 1/2 \, için \, \psi_t$ nin değeri $\sqrt{2}$ olarak hesaplanmıştır. Böylece merkezsel farklılık tahmini doğru olarak yapılmış olur. Δx adımı veya Δt zaman adımı için yapılabilecek

bir şey yoktur. Sayısal parametreleri ne kadar küçük alsak da, tek rakamlar kümesi kullansak da, ψ_t'nin $x = 1/2$ noktasındaki tahmini daha iyi olamaz. Basit olarak ψ_x türevinin tahmin yoluna bağlıdır. Şekil 5.15'de bu tasarımın kullanımı ile elde edilen sonuçlar görülmektedir. Zaman türevi ψ_x, ileri fark tasarısı ile yer değiştirilmiştir [34].



Şekil 5.14 Yayılan grafik için değişkenler [34].

(a) (b) (c)

Şekil 5.15 Gradyan'a göre merkezsel farklılık yaklaşımları (a) Kesin çözüm (b) $\Delta t = 0.005$ durumunda oluşan merkezsel farklılıklar (c) $\Delta t = 0.0005$ durumunda oluşan merkezsel farklılıklar [34].

Neyin yanlış gittiğini görmek çok kolaydır. Kesin çözümde, bütün $x \neq 1/2$ değerleri için $\psi_t = \sqrt{2}$ değerini alır. Maalesef, merkezsel farklılık tahmini farklı fakat yanlış bir sınır çözümü seçmektedir. Tanımlanmayan ψ_x eğimini, sol ve sağ eğimlerin ortalamasına eşit olarak alır. Hesaplama ilerledikçe, yanlış eğim hesaplamaları, sivri uçtan dışarı doğru inişli çıkışlı dalgalanmalar halinde dağılır. Bu dalgalanmalar kod da boğulmalara neden olur [45].

5.3.3. Katı çözümler için tasarılar

Gelişen arayüz örneği ile devam ederken, $\left(1+\psi_x^2\right)$ gradyan terimi üzerine yoğunlaşılmaktadır. (5.18) eşitliğinde verilen sonlu farklılık tahmini ele alındığında ve tekrar standart sonlu farklılık isimlendirilmesi kullanıldığında (5.19) eşitliği elde edilir.

$$\psi_x^2 \approx \left(\max\left(D_i^{+x}\psi, 0\right)^2 + \min\left(D_i^{-x}\psi, 0\right)^2\right) \tag{5.18}$$

$$D_i^{-x}\psi = \frac{\psi_i - \psi_{i-1}}{h} \quad D_i^{+x}\psi = \frac{\psi_{i+1} - \psi_i}{h} \tag{5.19}$$

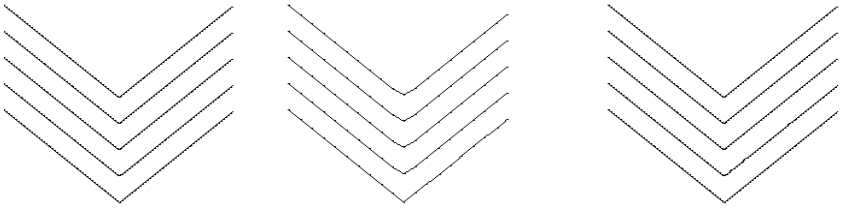
(5.19) eşiltliğinde ψ_i değeri ψ nin grid üzerindeki ih noktasının h grid aralığındaki değeridir.

(5.18) eşitliği bir ilerleme yönünün tersindeki tasarıdır. Tahmindeki grid noktalarını bilgi akış doğrultusunu kullanarak seçmektedir. Eğer bir yüzey soldan sağa gelişirse, sağa ilerleme yönündeki çözümü oluşturmak için bilgiye ulaşmak maksadıyla soldaki ilerleme yönünün tersindekine ulaşan farklılık tasarısı kullanılır.

Yukarıdaki örnekteki yayılan "V" eğrisi ele alınırsa, simetrik noktada tasarının simetrisinin değiştiğini ve sıfıra eşit olmayan bir değerin seçildiği görülür. Kesin çözüm iki simülasyon ile birlikte verilmiştir. Birincisi, entropi-sağlayıcı tasarıyı yalnız 20 nokta ile kullanır, ikincisi ise 100 nokta ile kullanır. Birinci tahminde, entropi durumu tamamlanmış, fakat köşeler az sayıda nokta kullanıldığı için keskin oluşmamıştır.

İyileştirilen hesaplamada, köşeler keskinleşmekte ve kesin çözüme daha çok yaklaşılmaktadır. Böylece bu tasarının entropi-koşulunun sağlanmasında doğru iş yaptığı görülmektedir (Şekil 5.16) [34].

Diğer birçok ilerleme yönünün tersindeki dizilerine karşın gradyan tahmini için yukarıdaki tahminler (ve bir küçük varyasyon) yeterli olacaktır; diğer tasarılar için ayrıntılar [41, 45] de bulunabilir.



(a) (b) (c)

Şekil 5.16 İlerleme yönünün tersi, gradyan için entropi-sağlayıcı tahminler (a) Tamçözüm (b) 20 noktalı tasarı (c) 100 noktalı tasarı [34].

5.3.4. İlerleme yönünün tersi, sınır değer ve başlangıç değer yaklaşımları için entropi- sağlayıcı akışkanlık tasarıları

Bir yayılan yüzey için hareket eşitliklerinin sınır değeri formülasyonu ve başlangıç değeri formülasyonu için uygun tasarıları oluşturmak amacıyla yukarıda anlatılan düşünceleri kullanabiliriz.

5.3.4.1. Sınır değeri formülasyonu

Çözmek istediğimiz problemi (5.21) eşitliği ile hatırlayalım

$$|\nabla u(x,y,z)| = \int(x,y,z) \tag{5.20}$$

Gradyan için yukarıdaki bölümlerde tanımlanan ilerleme yönünün tersi tahmin düşüncelerini çok-boyutlu bir ortama genişletirsek (5.21) eşitliğini kullanabiliriz.

$$\begin{aligned} |\nabla u| \approx \Big(&\max\left(D_{ijk}^{-x}u,0\right)^2 + \min\left(D_{ijk}^{+x}T,0\right)^2 \\ &+ max\left(D_{ijk}^{-y}u,0\right)^2 + min\left(D_{ijk}^{+y}T,0\right)^2 \\ &+ max\left(D_{ijk}^{-z}u,0\right)^2 + min\left(D_{ijk}^{+z}T,0\right)^2\Big)^{1/2} \\ &= \int_{ijk} \end{aligned} \tag{5.21}$$

Diğer koordinat yönlerindeki ileri ve geri operatörleri D^y, D^{+y}, D^z ve D^{+z} daha önce x yönü için tanımlanan ile benzerdir.

[40] da verilen ve daha uygun hale dönüşecek olan çok az farklı ilerleme yönünün tersindeki tasarı (5.22) eşitliğinde verilmiştir.

$$\begin{bmatrix} max\left(D_{ijk}^{-x}u,-D_{ijk}^{+z}u,0\right)^2+ \\ max\left(D_{ijk}^{-y}u,-D_{ijk}^{+y}u,0\right)^2+ \\ max\left(D_{ijk}^{-z}u,-D_{ijk}^{+z}u,0\right)^2 \end{bmatrix}^{1/2} = \int_{ijk} \qquad (5.22)$$

D⁻ ve D⁺ aynı ileri ve geri operatörler olup, $\int_{ijk}$, *ijk* grid noktasındaki yavaşlılıktır.

5.3.4.2. Başlangıç değeri formülasyonu

(5.23) eşitliği başlangıç değeri formülasyonu için bir entropi-sağlayıcı akışkanlık tasarısı olarak gösterilmiştir, düzey kümesi yöntemi olarak bilinir ve sayısal yöntemlerin önde gelenlerindendir [39].

$$\phi_{ijk}^{+1} = \phi_{ijk}^{n} - \Delta t\left[max\left(F_{ijk},0\right)^{\nabla+} + min\left(F_{ijk},0\right)^{\nabla-}\right] \qquad (5.23)$$

(5.23) eşitliğindeki ∇^+ ve ∇^- terimlerinin karşılıkları (5.24) ve (5.25) eşitlikleri ile verilebilir.

$$\begin{aligned} \nabla^+ = \Big[& max\left(D_{ijk}^{-x},0\right)^2 + min\left(D_{ijk}^{+z},0\right)^2 + \\ & \max\left(D_{ijk}^{-y},0\right)^2 + \min\left(D_{ijk}^{+y},0\right)^2 + \\ & \max\left(D_{ijk}^{-z},0\right)^2 + \min\left(D_{ijk}^{+z},0\right)^2 \Big]^{1/2} \end{aligned} \qquad (5.24)$$

$$\begin{aligned} \nabla^- = \Big[& max\left(D_{ijk}^{+z},0\right)^2 + min\left(D_{ijk}^{-z},0\right)^2 + \\ & \max\left(D_{ijk}^{+y},0\right)^2 + \min\left(D_{ijk}^{-y},0\right)^2 + \\ & \max\left(D_{ijk}^{+z},0\right)^2 + \min\left(D_{ijk}^{-z},0\right)^2 \Big]^{1/2} \end{aligned} \qquad (5.25)$$

Burada $D^+\phi_i^n$ yerine D^{+x} vb. gibi kısaltmalar kullanılmıştır. Detaylı bilgiler için [39,45] numaralı kaynaklara bakabilirsiniz.

5.4. Uyarlanabilirlik (Adaptivity)

Sınır değeri ve başlangıç değeri formülasyonları için verilen tasarılar hesapsal olarak yetersizdir. Bu bölümde, uyarlanabilirlik ve nedensellik olarak değişik kullanımlarla iki tasarıyı da optimal hale getirmeye çalışacağız.

5.4.1. Tasarılar ve işlem sayıları

5.4.1.1. Düzey kümesi tasarısı

(5.23) eşitliği açık bir tasarıdır, bu yüzden doğrudan çözülebilir. Zaman adımı gereksinimi F hız fonksiyonun doğasına bağlıdır. Sadece konuma bağlı olan F için zaman adımı $\frac{\Delta t}{\Delta x} F \leq 1$ olarak davranır. F hız fonksiyonu, eğrinin terimlerine bağlı olduğunda (örneğin $F = -\kappa$) eşitliğin iki parabolik bileşeni vardır ve bundan dolayı zaman adımı gereksinimi doğrusal olmayanısı eşitliğine benzer; zaman adımı sıkı sıkıya $\frac{\Delta y}{\Delta x^2}$ ye bağlıdır.

Bu düzey kümesi formülasyonu iki merkezi parçayı ortaya çıkarır.

- İlk olarak, başlangıç adımında, belirlenen uzaklık fonksiyonu düzey kümesi ϕ=0 da arayüzeye karşılık gelen ϕ fonksiyonunu kurmak için kullanılır. Bu adıma başlangıç yada başlatma adımı ismi verilmektedir.
- İkincisi, (5.23) eşitliğinde verilen fonksiyonun başlangıç değerinin inşası, F hızının, sadece kendi arayüzüne karşılık gelen sıfır düzey kümesi için değil bütün düzey kümeleri için tanımlanmış olduğu anlamına gelir. Düzey kümesi eşitliğini (5.26)' daki gibi daha hassas olarak yeniden yazabiliriz.

$$\phi_t + F_{ext} |\nabla \phi| = 0 \qquad (5.26)$$

Bu eşitlikte F_{ext}, 0 düzey kümesinde, verilen F hızına eşit olan hız alanıdır. Diğer bir deyişle:

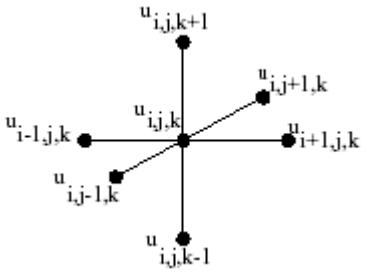
$$\phi = 0 \quad \text{yüzeyi üzerinde} \quad F_{ext} = F \qquad (5.27)$$

Burada tanımlanan, F_{ext} yeni hız alanı, genişletilmiş hız olarak bilinir.

Bunun anlamı; düzey kümesi yönteminin, ϕ^n düzey kümesi fonksiyonunun, hesapsal ilgi alanındaki her noktada tanımlanmış olan F hız fonksiyonu kullanılarak her grid noktasında güncellenmesine ihtiyaç duyduğudur. Bazı problemlerde örneğin eğri akışı gibi ($F = -\kappa$), genişletilmiş hız için uygun tanım açık ve nettir; (i,j) grid

noktasından geçen düzey kümesinin eğriliğini güncellemede o noktadaki hız fonksiyonu gibi kullanılır. Buna rağmen çoğu problemlerde (akışkan ve yanan yüzeyleri içerenler gibi) yüzeyin genişletilmiş hızının uygun tanımı açık değildir; fazla esneklik olduğunda yinede bütün grid noktalarında genişletilmiş hızın inşa edilmesi gerekmektedir.

Düzey kümesi yönteminin sayısı için kaba bir işlem yapabiliriz. Üç boyutlu problemin her uzaysal boyutunda N tane grid noktası olduğunu varsayalım. F=1 hızı ile bir yüzeyin ileri doğru yayılmasını konu eden basit bir problemi ele alalım, ilgi alanında yüzeyin yayılımı için N kadar zaman adımına ihtiyacı olduğunu kabul dersek, işlem sayısı $O(N^4)$ yöntemi ile bulunabilir [34].



Şekil 5.17 Grid noktasının güncellenmesi [34].

5.4.1.2 Sınır değeri yaklaşımı

Karşıt olarak, (5.21) eşitliği kapalı bir eşitliktir. Buradaki çözüm iteratif bir çözümdür [40]. Bir grid noktası ve altı komşusunun olduğu bir şablon düşünelim (Şekil 5.17) [34].

İterasyon Başlangıcı (1 den, nokta sayısı kadar)

İterasyon Başlangıcı (i,j,k için 1 den başlayarak boyut sayısı kadar)

$$(u_{ijk} \text{ için verilen } u_{i-1,j,k}, u_{i-1,j,k}, u_{i,j-1,k}, u_{i,j+1,k}, u_{i,j,k-1}, u_{i,j,k+1} \text{ ile kareselliği çöz}) \quad (5.28)$$

İteresyonun Sonu

İterasyonun Sonu

(5.21) eşitliği, u için komşu grid değerlerinin verildiği kabul edildiğinde, u_{ijk} için bir karesel eşitliktir. Böylece, herbir grid noktasında, çözüme ulaşılana kadar u'nun değerini, bu kareselliğe göre güncellemekle çözüme ulaşılır. Örnek bir iteratif çözüm (5.28) eşitliğinde verilmiştir.

Bu yaklaşımda, her yönde N kadar nokta olduğu ve çözüme yakınsamak için N tane adım gerekli olduğu kabul edilirse, işlem sayısı $O(N^4)$ tekniği ile bulunabilir.

5.4.2. Dar bant düzey kümesi yöntemi

Düzey kümesi yöntemiyle gerçekleştirilen işlemler, "Dar Bant Seviye Yöntemi"nin kullanılmasıyla hızlandırılabilir [35]. Hesapları bütün hesapsal ilgi alanı üzerinde yapmak yerine sadece sıfır düzey kümesinin komşuluğunda oluşturulacak dar bir alanda gerçekleştirmek işlem sürelerini kısaltacak ve bu yöntemi daha uygulanabilirliğini arttıracaktır. Başlangıç değeri düzey kümesi yaklaşımında bu yöntemin seçilmesinin üç ana nedeni vardır.

- İlk olarak üç boyutlu problemlerin çözümündeki işlem sayısı $O(kN^2)$ ye iner ve önemli bir azalma gösterir. Buradaki k dar banttaki hücrelerin sayısıdır.

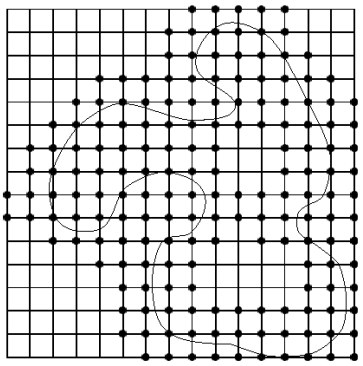
- İkincisi; dar bant yöntemi, gelişen hız yapısının, hesapsal ilgi alanındaki bütün noktalara değil sadece dar bant üzerindeki noktalara uygulanmasını gerektirir. Böylece hesaplama karmaşıklığı ve işlem sayısı önemli ölçüde azaltılabilir.

- Üçüncüsü; bir dar bant uygulamasında, maksimum hız alanına karşın sadece dar bant içindeki hız, zaman adımı olarak uygulanabilir. Hareket ettikçe yüzey hızının çok değişkenlik gösterdiği durumlarda (örneğin eğri akışındaki gibi) bu yaklaşım çok avantajlıdır.

Dar bant yöntemi görüntü işlemede de kullanım alanları bulmuştur [36]. Özellikle görüntülerdeki gürültü seviyesinin azaltılmasında, görüntülerin iyileştirilmesinde ve görüntülerden detay çıkarımında kullanılmıştır [38].

Dar bant yöntemindeki ana düşünce oldukça açıktır ve iki şekil yardımıyla açıklanabilir [35]. Bir başlangıç yüzeyinin çevresine bir dar bant yerleştirelim (Şekil 5.18) [35]. Verinin tamamının iki boyutlu grid noktaları, bir kare dizi içerisine depolanmaktadır. Daha sonra tek boyutlu bir nesne, bu dizi içerisindeki noktaları izlemek için kullanılmaktadır. Şekil 5.18'deki koyu renkli grid noktaları, kullanıcı tarafından genişliği tanımlanan bir bant içerisine yerleştirilmektedir (Şekil 5.19) [35]. Buradaki dar bant bir tüpü andırmaktadır. ϕ düzey kümesi fonksiyonunun sadece bu tüp içerisinde kalan noktaları güncellemeye tabi tutulmaktadır. Dar bantın üzerinde bulunan grid noktalarında ϕ'nin değerleri dondurulmaktadır. Yüzey hareketi

sırasında, tüpün sınırlarına yaklaştığında hesaplama durdurulur ve sıfır düzey kümesi arayüzeyinin sınırı merkezde olacak şekilde yeni bir tüp oluşturulmaktadır [35].

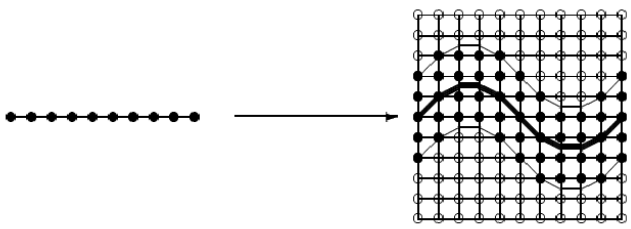


Şekil 5.18 Koyu renkli grid noktaları, dar bantın elemanlarıdır [35].

Sonuç olarak; dar bant yöntemini şu şekilde bir iteratif döngüden oluştuğunu söylemek yanlış olmaz:

- Dar bant içerisindeki noktalar "Canlı" olarak etiketlenir,
- Tüpün alanını belirlemek için sınırlar yapılandırılır,
- Dar bant dışında kalan "Uzak Noktalar" için başlangıç değeri olarak büyük pozitif veya negatif değerler atanır,
- Sınırlara ulaşılıncaya kadar düzey kümesi eşitliği çözülür,
- Döngü tekrar başlatılır.

Gerçekleştirilen uygulamalar, 160x160 gridlik bir alanda, dar bant yöntemiyle yapılan işlemler, dolu matris yöntemiyle yapılanlardan on kat daha hızlı olduğunu göstermiştir [35].



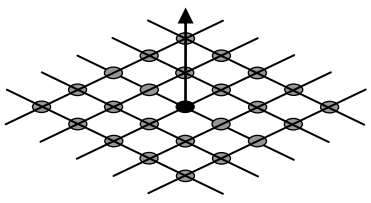
Şekil 5.19 İşaretçi dizisi tarafından, dar bantın içerisinde ve sınırındaki bant noktalarının etiketlenmesi [35].

5.4.3. Hızlı ilerleme yöntemi

Sınır değeri formülasyonu daha hızlı bir şekilde gerçekleştirilebilir ve böylece dar bant düzey kümesi yönteminden daha hızlı çalışan bir yöntem ortaya konabilir. Bu yönteme Hızlı İlerleme Yöntemi (Fast Marching Method) adı verilir. Hızlı ilerleme yönteminin arkasındaki temel düşünce u'nun çözümünü üretmek için, çözümü ilerleme yönünde olacak biçimde sistematik olarak inşa etmektedir. Anahtar, (5.22) eşitliğinin ilerleme yönünün tersindeki farklı yapısını gözlemlemektir: Bilgi, tek yönde, u' nun küçük değerlerinden büyük değerlerine doğru yayılır. Böylece, hızlı ilerleme algoritması, (5.22) eşitliğinin çözümünü, en küçük u değerinden dışarıya doğru inşa ederek bulma temeline dayanır.

Aslında bu daha önce tanımlanan ilerleme yönünün tersindeki kareselleştirme bakışının arkasındadır; ilerleme yönünde sınır şartından dışa doğru çözümü ilerletebiliriz. İnşa alanını, yüzey etrafında dar bir bant şeklinde sınırlandırarak algoritma hızlandırılabilir. Amaç, mevcut yüzey etrafındaki bir takım noktaları dar bantta olarak düşünüp yüzeyi ilerleme yönünün tersindeki şeklinde öne doğru süpürmektir ve bu dar bandı ileri doğru ilerletmek, mevcut noktaların değerlerini dondurmak ve dar bant yapısı içerisine yenilerini getirmektir. Önemli olan, dar bant içerisindeki hangi grid noktasının güncellemek için seçileceğidir, ki bundan sonra bu konu incelenecektir.

Sınır değerinin orijinde olduğu Eikonal eşitliğinin 2 boyutlu versiyonunu ele alalım. Bu şematik olarak Şekil 5.20'de gösterilmektedir [34]. $u_{0,0}$ daki siyah küre u değerinin bilindiği (başlangıç ve ilk değerdir) grid noktasını belirtir ve açık gri küreler de çözüm değerinin bilinmediği grid noktalarını gösterirler.



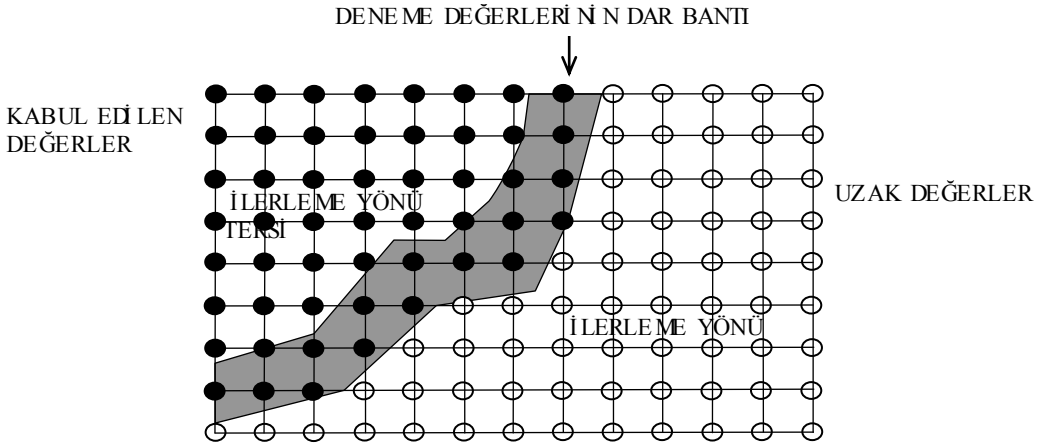
Şekil 5.20 Hızlı ilerleme yönteminin başlangıcı [34].

Şekil 5.22'de gösterildiği gibi, bilinen değerden ilerleme yönünde ilerleyip, komşu 4 grid noktasındaki yeni değerleri hesaplayarak algoritmayı başlatabiliriz.

Bu, Şekil 5.22'de koyu gri küreler olarak gösterilen $u_{-1,0}$, $u_{1,0}$, $u_{0,-1}$, $u_{0,1}$ grid noktalarında, u için muhtemel değerleri sağlar.

Şimdi koyu gri kürelerde verilen değerlerden ilerleme yönünde yayılmak istiyoruz, fakat hangisini seçeceğimizi bilmiyoruz. Cevap; "*bu koyu gri kürelerdeki en küçük u değeri doğru olmak zorundadır*" gözleminde yatar. İlerleme yönünün tersindekinden dolayı, hiçbir nokta, u'nun daha büyük değerlerini içeren grid noktalarından etkilenmez. Böylece, bu en küçük koyu gri küredeki u nun değeri dondurulabilir (yani, bu onu siyah küreye çevirmek ve değeri biliniyor olarak değerlendirmek demektir) ve algoritmaya uygun olarak ilerlenebilir. Bu şematik olarak Şekil 5.22'de gösterilmiştir [34].

İlerleme yönünün tersindeki komşu noktalarındaki T değerlerini yeniden hesaplama işleminden elde edilen değer, kabul edilen noktaların hiç birindekinden daha küçük olamayacağından, bu algoritma çalışır. Böylece, her zaman dar bant grid noktasını T için minimum deneme değeri ile seçerek ve komşuları tekrar ayarlayarak çözüm dışarıya doğru ilerletilebilir (Şekil 5.21) [34].



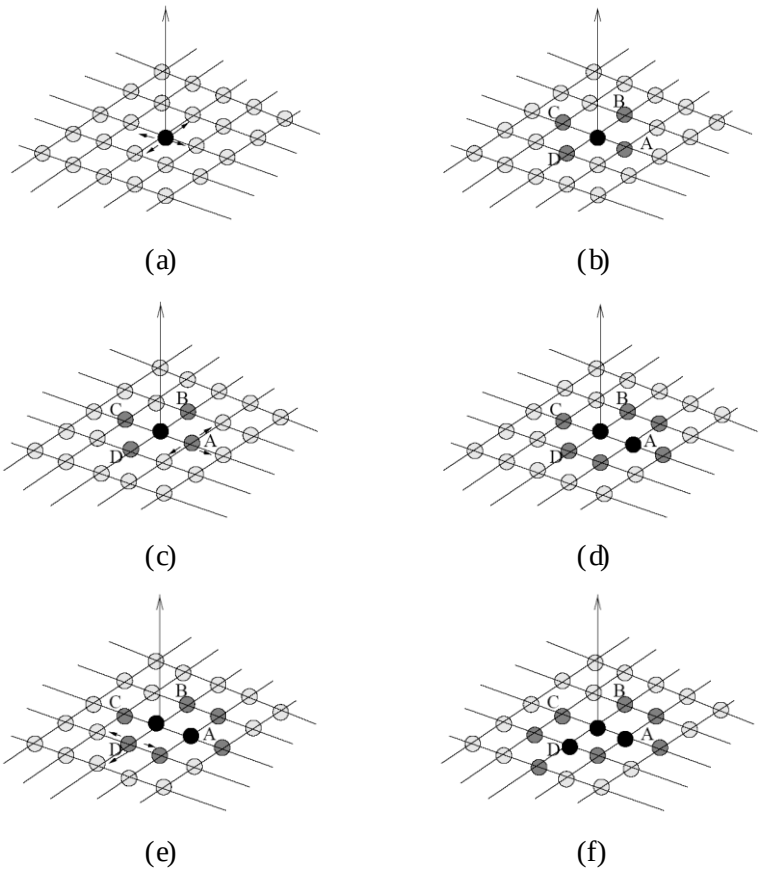
Şekil 5.21 Kabul edilen değerlerin ilerleme yönünün tersindeki yapısı [34].

Bir başka yolda, her bir minimum deneme (T) değeri, Huyghen's prensibinin bir uygulaması olarak başlar ve yayılan dalga şeklinde sınıra değer ve diğerlerini günceller [41, 45].

Böylece, hızlı ilerleme yöntemi şu şekilde gerçekleştirmek için ilk olarak, başlangıç durumundaki noktalar "Canlı" olarak etiketlenir. Sonra, bir grid nokta uzaklığındaki

tüm noktalar, "Kapalı" olarak etiketlenir. Son olarak da, diğer tüm grid noktaları "Uzak" olarak etiketlenir. Böylece iteratif döngü şu şekilde gerçekleştirilebilir:

- Döngüye Başla: *Deneme* değeri, *kapalı* bölgedeki en küçük T değerine sahip nokta olsun.
- *Deneme* noktasını *canlı* bölgeye ekleyin; *kapalı* bölgeden çıkarın.
- *Deneme* noktasının, *canlı* olmayan tüm komşularını *kapalı* olarak etiketleyin. Eğer komşunun etiketi *uzak* ise onu bu listeden çıkarın ve *kapalı* kümesine ekleyin.
- Tüm komşularda, kareselleştirme eşitliğini çözerek, (5.22) eşitliğine göre, T değerlerini tekrar hesaplayın.
- Döngünün başına dönün.



Şekil 5.22 Hızlı ilerleme yöntemi için güncelleme işlem adımları (a) İlerleme yönünde güncelleme işlemi (b) Yeni olası değerlerin hesaplanması (c) En küçük değere sahip koyu gri kürenin seçilmesi (Örneğin "A") (d) A' nın değerinin dondurulması, ilerleme yönündeki komşu noktaların güncellenmesi (e) En küçük değere sahip koyu gri kürenin seçilmesi (Örneğin "D') (f) D' nin değerinin dondurulması, ilerleme yönündeki komşu noktaların güncellenmesi [34].

5.4.4. Yığın sıralama ve hesap etkinliği

Yukarıdaki tekniğin etkili bir versiyonuna çözüm yolu, u için en küçük değer ile, grid noktasını dar bant içine hızlı bir şekilde yerleştirmede yatar. Bunu bu şekilde yapmak için etkili bir tasarı ortaya konabilir [45]. Bu bölümde bu gerçekleştirilmeye çalışılacaktır.

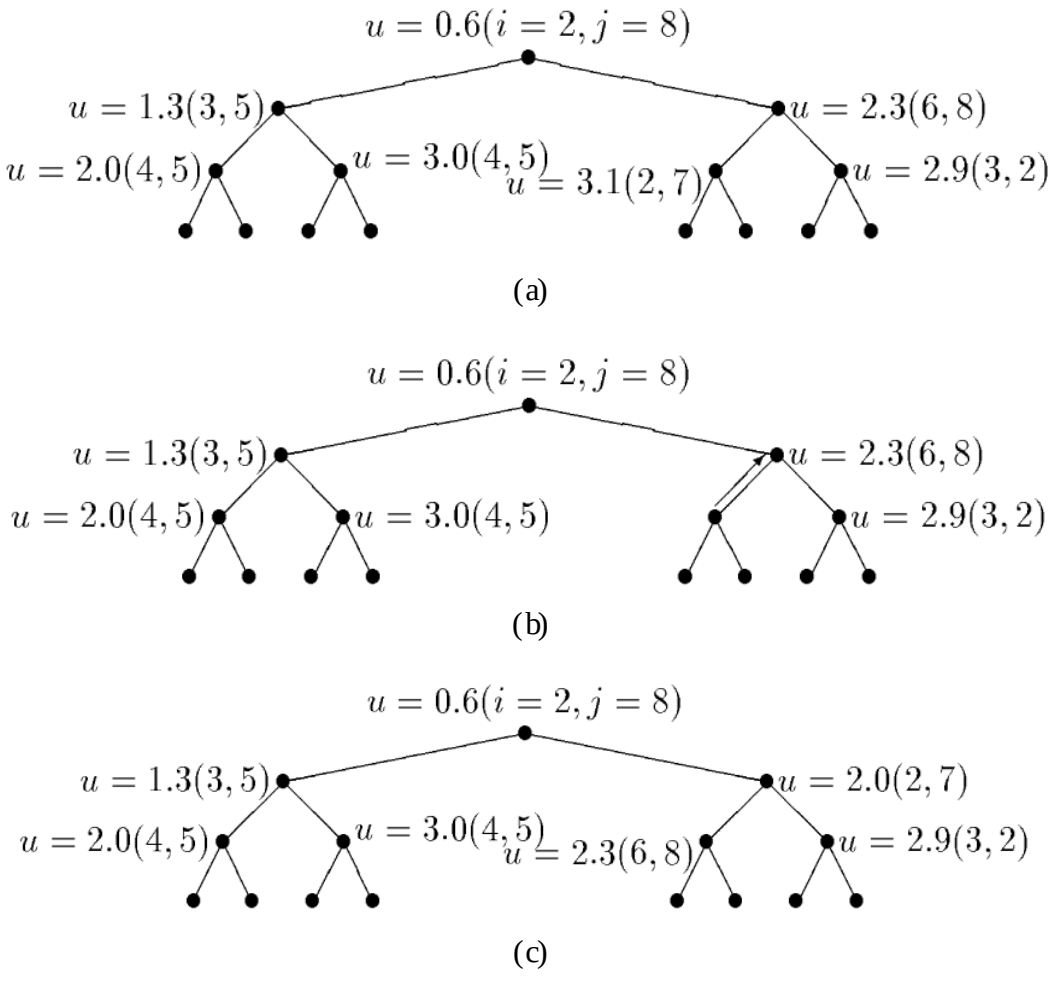
u değerlerini depolamak için geri işaretçilerle birlikte yığın algoritmasındaki bir değişimi kullanılabilir ve özellikle minimum-yığın veri yapısını kullanılır. Özetle, minimum-yığın yapısı, herhangi bir düğümdeki değerin, çocuklarındaki değerlerden daha küçük veya eşit olduğu 'tamikili ağaç'tır. Pratikte, yığın, ardışık olarak, düğümü k konumunda, çocuklarını $2k$ ve $2k+1$ konumlarında depolayan bir dizi olarak temsil edilebilir. Bu tanımdan, k'daki belirli bir düğümün babası $k/2$ konumunda olur. Bu nedenle, en küçük elemanı içeren kök, dizi içerisinde, $k=1$ konumunda depolanır. Verilen bir elemanın babasını veya çocuklarını bulmak, $O(1)$ kadar zaman süren basit dizi ulaşımlarıdır.

u nun değerleri, grid yapısı içerisinde konumlarını belirten indislerle birlikte depolanır. İlerleme algoritması ilk bakışta, dar bant içindeki en küçük eleman için çalışır; bu "en küçüğü bulma" işlemi kökü silme ve kalan elemanların yığın özelliğine uyup uymadığından emin olmak için yığın en tepeden itibaren araştırılmasını gerektirir. Algoritma, canlı olmayan komşu noktalarını etiketleyerek ilerler. Uzak komşular "ekleme" işlemi kullanılarak yığına eklenir ve kalan noktalardaki değerler (5.22) eşitliği ile güncellenir. "Ekleme" işlemi yığın boyutunu bir arttırarak ve yığında aşağıdan yukarıya doğru yeni elemanı üst tarafa, doğru konumuna iteleyerek çalışır. Son olarak, güncellenen u değerlerinin yığın özelliğini bozmadığından emin olmak için, bu konumdan başlayan ve yığının tepesine doğru ilerleyen bir kontrol işlemine ihtiyaç vardır.

Yığında (en kötü durumda), kökten dibe doğru veya tersi yönde, tüm yol boyunca bir eleman taşınır. Bu nedenle, yığında N tane eleman olduğu düşünülürse, bu $O(\log N)$ kadar zaman alır. Şu unutulmamalıdır ki, tamikili ağaç olan yığının, her zaman için dengeli kalması garanti edilir. Geriye kalan, yığın içindeki en küçük elemanın dar bant içindeki komşularını arama işlemidir. Bu, gridden yığın dizinine geri işaretçileri, muhafaza ederek $O(1)$ kadar zamanda gerçekleştirilir. Geri işaretçiler olmadan yukarıdaki arama işlemi en kötü durumda $O(N)$ kadar zaman alır.

Örneğin, Şekil 5.23 tipik bir yığın yapısını ve (2,7) konumundaki elemanın değerinin 3.1 den 2.0 a değiştikten sonraki yığında yukarı taşınma işlemini gösterir.

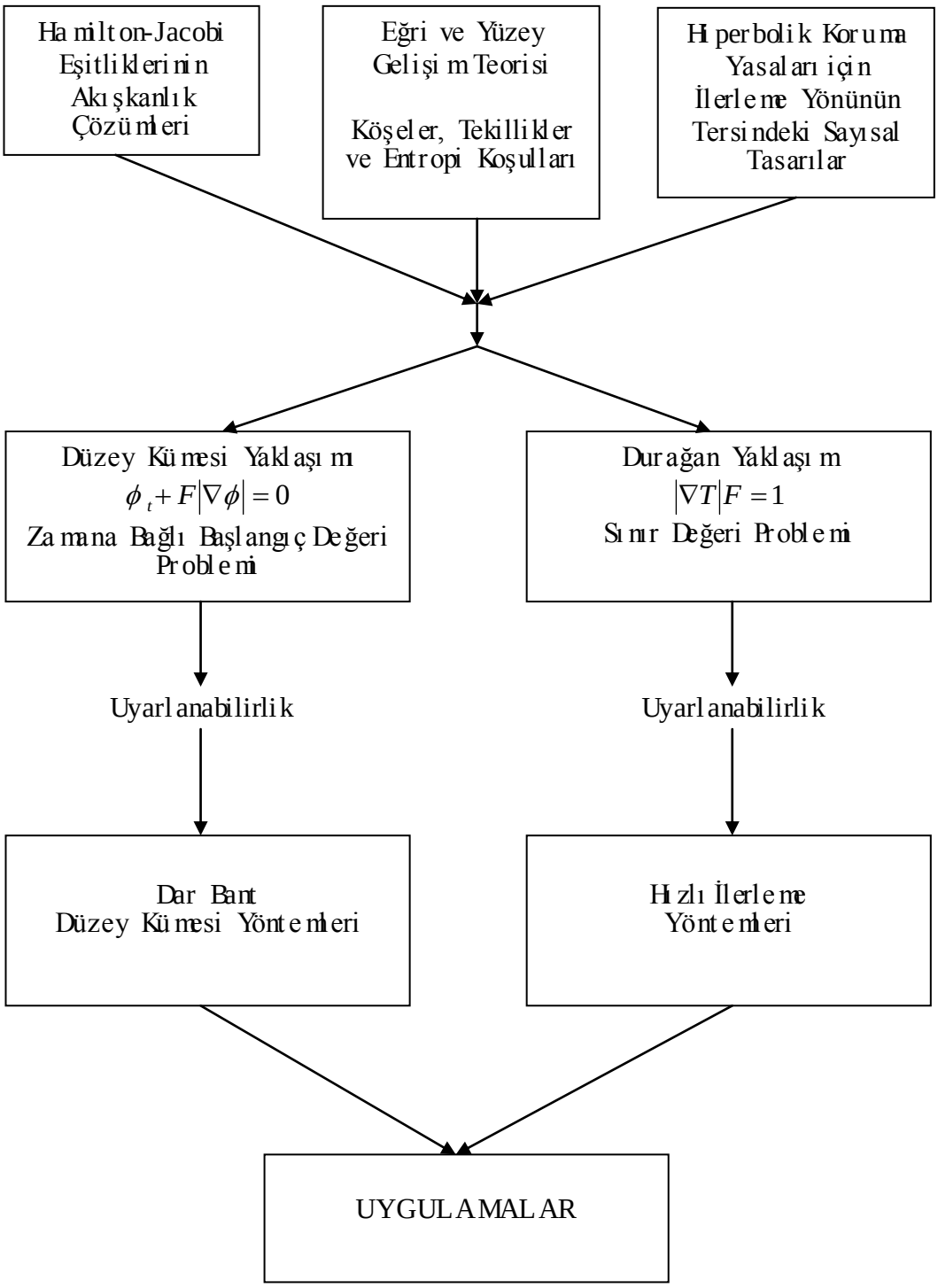
Böylece, yığının bir elemanının değerini değiştirmek ve değerini yukarı doğru çıkartmak için toplamış $O(\log M)$ kadar olduğundan, burada M yığının boyutudur, toplam M tane noktanın olduğu bir grid de hızlı ilerleme yöntemi için toplamışlem süresi $M \log M$ olur. Böylece her yönde N tane noktanın olduğu 3 boyutlu bir grid düşündüğümüzde, hızlı ilerleme yöntemindeki toplamışlem sayısı N^4 'den $N^3 \log N$'e düşürülür; zaten, varış zamanı değerini hesaplamak için, her bir grid noktası bir kez ziyaret edilir.



Şekil 5.23 Yığın yapısı ve yığına yeni eleman ekleme işlemi (a) (2,7)' deki u değeri değişir ve 2.0 değerini alır (b) Yeni değer eskisinden daha küçük ve aynı zamanda babası (2,3) değerinden küçük olduğu için yığında bir üst sıraya yerleşir (c) Böylece yığın özelliği korunmuş olur.

5.4.5. Yöntemlerin akış grafiği

Düzey kümesi ile hızlı ilerleme yöntemleri arasındaki ilişki ve bu yöntemlerin uygulanabilir versiyonları Şekil 5.24' te grafik olarak gösterilmiştir [34].



Şekil 5.24 Hızlı ilerleme ve düzey kümesi yöntemlerinin ilişkileri [34].

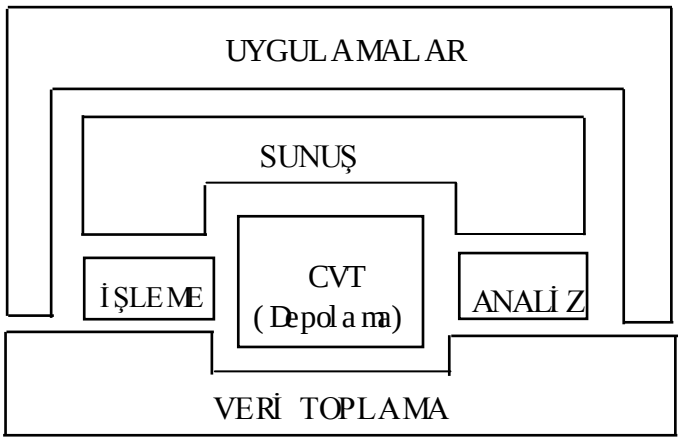
6. CBS İÇİN VERİ TOPLAMA YÖNTEMLERİ

CBS, Coğrafya ile ilgili grafik ve grafik olmayan verilerin kullanıcı ihtiyaçlarını karşılayacak biçimde çeşitli kaynaklardan toplanması, depolanması, işlenmesi, analiz edilmesi, yönetilmesi ve sunulması fonksiyonlarını bütünleşik olarak yerine getiren coğrafi veri, donanım, yazılım ve personel bileşenlerinden oluşan bir organizasyondur [49].

CBS'nin tanımından yola çıkarak bir CBS'de bulunması gereken temel fonksiyonlar aşağıdaki gibi sıralanabilir:

- Coğrafi verinin toplanması
- Coğrafi verinin depolanması
- Coğrafi verinin işlenmesi
- Coğrafi verinin analizi
- Coğrafi verinin sunumu

CBS'nin temel fonksiyonları ışığında bir CBS temel mimarisi şöyle çizilebilir (Şekil 6.1) [50]:



Şekil 6.1 CBS temel mimarisi [50].

CBS'nin kurulması ve yaşatılması için gerekli olan bu beş temel fonksiyon maliyet, emek ve zaman açısından değerlendirildiğinde; yaklaşık ¾'lük bölümün veri toplama işlemleri için harcandığı gözlenmektedir.

Bu çalışmada, hava fotoğraflarından yarı otomatik olarak çizgisel detayların çizilmesi probleminin çözülmesi amaçlanmaktadır. Bu problemin çözümü sadece fotogrametri değil CBS içinde gerekli olan coğrafi verilerin toplanmasına farklı bir yaklaşım getirecektir. Bu yaklaşımın CBS içindeki yeri ve diğer coğrafi veri toplama yöntemlerine göre üstün ve/veya zayıf yönlerinin daha kolay bir şekilde anlaşılabilmesi için coğrafi veri kavramı, coğrafi veri kaynakları ve coğrafi veri toplama yöntemleri ele alınacaktır.

6.1. Coğrafi Veri Kaynakları

CBS için veri toplama, gerekli grafik ve grafik olmayan bilgilerin ilişkili, tutarlı ve anlamlı bir küme olarak derlenmesi ve sayısallaştırılması olarak tanımlanabilir [50].

CBS için gerekeli olan verilerin toplanması için değişik veri kaynakları kullanılmakta olup, veri kaynaklarının yapısına bağlı olarak çok değişik teknolojik yaklaşımlar geliştirilmiş ve farklı coğrafi veri toplama yöntemleri ortaya konmuştur. Bu yöntemler açıklanmadan önce coğrafi veri kavramının ve bu verilerin elde edilebileceği veri kaynaklarının açıklanmasında yarar vardır.

6.1.1. Coğrafi veri kavramı

Yeryüzü üzerinde veya yakınında belli bir anlama sahip somut veya soyut her şey coğrafi varlıktır [49]. Coğrafi veri, belirli bir konumu ve biçimi olan somut ya da soyut, doğal ya da insan yapısı bütün nesnelere diğer bir deyişle coğrafi varlıklara ait her türlü bilgiyi içermektedir. Coğrafi varlıklar ve bunlara ait bilgiler CBS'inde detaylar ve bu detaylara ait öznitelikler olarak temsil edilmektedirler.

Detaylar boyutsal özelliklerine göre 0 boyutlu (noktasal detaylar), 1 boyutlu (çizgisel detaylar), 2 boyutlu (alansal detaylar) ve 3 boyutlu (hacimsel detaylar) olmak üzere sınıflandırılmaktadır. Detayların boyutsal özelliğine "detay tipi" de denir. Detayların CBS'lerinde sayısal olarak temsil edilmelerinde üç temel geometrik

eleman kullanılır: Nokta, çizgi ve alan. Burada sözü edilen geometrik elemanlar bilgisayar ortamında vektör veya raster olarak iki değişik formatta temsil edilirler.

6.1.2. Coğrafi verilerin sınıflandırılması

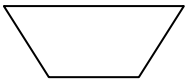
CBS içindeki detayları ve bu detaylara ait öznitelik bilgilerini oluşturmak amacıyla gerekli olan coğrafi verileri mantıksal tiplerine göre ve veri kaynaklarına göre iki ayrı sınıfta toplamak mümkündür.

6.1.2.1. Mantıksal tiplerine göre coğrafi veriler

CBS için gerekli olan coğrafi veriler mantıksal tiplerine göre üç grupta ele alınabilir.

- Konum Verileri
- Öznitelik Verileri
- Topolojik Veriler

Konum verileri coğrafi varlığın (detayın) belli bir referans sistemine göre yerini ve biçimini belirten koordinat veya piksel değerleridir. Konum ve biçim bilgisi iki boyutlu olabileceği gibi üç boyutlu da olabilir. Geometrik veri olarak da adlandırılmaktadır. Bilgisayar belleğinde ve depolama birimlerinde vektör veya raster formda temsil edilirler (Şekil 6.2) [52].

COĞRAFİ VERİ TÜRÜ		COĞRAFİ VARLIK (DETAY) TÜRÜ		
		Nokta	Çizgi	Alan
GRAFİK VERİ	Şekil Verisi (Sembol)	●		
	Konum Verisi (Koordinat)	Detay koordinatı (X, Y)	Detayı temsil eden noktaların koordinatları (Xi, Yi)	Detayı temsil eden noktaların koordinatları (Xi, Yi)
GRAFİK OLMAYAN VERİ (ÖZNİTELİK)		Detay Kodu : 8 Rakımı : 500 m. Çeşidi : 1 No : 27374	Detay Kodu : 34 Yüzey : Asfalt Çeşidi : Cadde Şerit Sayısı : 2	Detay Kodu : 174 Parsel No : 5378 Ada No : 45 Sahibi: Çiğdem PAK

Şekil 6.2 Coğrafi veri türleri [52].

Öznitelik verileri ise konuma bağlı olmayan, topolojik olmayan, doğrudan detaya bağlı ve detayı tanımlayıcı grafik olmayan verilerdir. Örneğin ormandaki ağaç cinsi, akarsuyun debisi, parselin sahibi vb. öznitelik bilgileridir.

Topoloji, şekillerin büyüklük ve biçim özellikleri ile değil; komşuluk, birleşme ve içerme gibi şekil bozulması karşısında değişmeden kalan özellikleri ile, yani geometrik olmayan özellikleri ile ilgilenir. Topolojik veri yapısında detayların hem konum bilgilerinin hem de konuma ait ilişkilerinin (komşuluk, çakışıklık, yön, bağlantılar) bilgisayarda temsil edilebilmesi için kenar ve düğüm elemanları kullanılmaktadır. Yani topolojik veriler detaylar arasında ölçülebilir olmayan uzaysal ilişkileri; komşuluk, çakışıklık, içerme, bağlantı vb. ilişkileri ifade eder. Topoloji sayesinde alan içinde alan, bir detayı kesen detaylar, bir alan detayın komşuları gibi sorgulamalar mümkün olmaktadır. Topolojik verilerin doğrudan toplanması, başka bir deyişle CBS ortamına dışarıdan getirilmesi yerine CBS ortamına aktarılmış olan konum verilerinin analizi ile türetilmesi daha uygundur.

Tablo 6.1. Coğrafi veri kaynakları [50].

KAYNAK SINIFI	KAYNAK CİNSİ
Mevcut Harita ve Dokümanlar	* Çizgisel Haritalar * Tematik Haritalar * Grafik Çizimler (Bilgisayar Destekli Tasarım ve Çizim Ürünleri * Harita Baskı ve Revizyon Bilgi Kalıpları * Ortofoto Haritalar * Dokümanlar
Fotoğraf ve Uzaktan Algılama Verileri	* Hava Fotoğrafları * Yersel fotoğraflar * Uzaktan Algılama Verileri * Uçaklara Takılı Sistemlerden Algılanan Veriler
Arazi Ölçü Verileri	* Klasik Ölçme Kayıtları * Manyetik Ortamdaki Arazi Ölçmeleri * GPS Ölçüleri
Hazır Sayısal Coğrafi Bilgiler	* Standart Formatta Sayısal Coğrafi Bilgi Kütükleri * On-line Bağlantılı Diğer Coğrafi Bilgi Sistemleri

6.1.2.2 Veri kaynaklarına göre coğrafi veriler

Coğrafi veriler, CBS'ne aktarılmadan önceki bulundukları ortama, başka bir deyişle kaynaklarına göre de sınıflandırılırlar. Coğrafi veri toplama yöntemini ve teknolojisini belirleyen en önemli unsur verinin kaynağıdır. Coğrafi verilerin

toplanabileceği başlıca kaynaklar klasik olarak beş ana gruptatoplanmaktadır (Tablo 6.1) [50].

6.2. Coğrafi Veri Toplama Yöntemleri

CBS'de girdi olarak kullanılacak coğrafi verilerin toplanması için değişik veri kaynaklarına yönelik olarak çok değişik teknolojik yaklaşımlar geliştirilmiştir. CBS uygulamalarının pek çoğu, birden çok kaynaktan veri toplanmasını ve bu verilerin entegrasyonunu gerektirmektedir. Buna göre coğrafi veri toplama yöntemleri aşağıdaki şekilde sınıflandırılabilir:

- Sayısallaştırma ile coğrafi veri toplama
  - Manuel (el-ile) sayısallaştırma
  - Otomatik çizgi izleyerek sayısallaştırma
  - Ekrandan sayısallaştırma (Heads-up digitizing)
- Raster'dan vektöre dönüşüm yöntemi ile coğrafi veri toplama
- Video kayıt ile coğrafi veri toplama
- Fotogrametrik yöntemile coğrafi veri toplama
- Uzaktan algılama yöntemi ile coğrafi veri toplama
- Doğrudan arazi ölçmeleri ile coğrafi veri toplama
- Alfasayısal bilgi girişi ile coğrafi veri toplama
- Mevcut sayısal verilerin ithali ile coğrafi veri toplama
- Hava fotoğraflarından ve uydu görüntülerinden otomatik ve/veya yarı otomatik detay çizme yöntemleri ile veri toplama

CBS için gerekli olan tüm verileri toplamaya yeterli tek başına kullanılan bir yöntem mevcut değildir. Bunun yerine, kurulacak CBS'nin ölçeğine ve doğruluğuna uygun, CBS'deki tüm beklentilere yanıt verebilen, birbiriyle nitelik, doğruluk ve içerik bakımından uyumlu yukarıda sıralanan birkaç yöntemin beraberce kullanılması ile daha anlamlı, doğru ve tatmin edici sonuçların elde edilmesi olanaklıdır. Bu yöntemlerin teknikleri ile birlikte, avantaj ve dezavantajları aşağıda incelenmiştir.

6.2.1. Sayısallaştırma ile coğrafi veri toplama

Sayısallaştırma tekniği ile veri toplamada konum verileri iki boyutta x, y koordinat çiftleri halinde, başka bir ifade ile vektörel formda toplanırlar. Sayısallaştırmada, manuel veya otomatik çizgi izleyici sayısallaştırıcılar kullanılır [48].

6.2.1.1. Elle sayısallaştırma yöntemi ile veri toplama

Manuel sayısallaştırma ile vektör yapıda grafik veriler toplanır. Manuel sayısallaştırmada kaynak materyal olarak herhangi bir altlık üzerinde yer alan her türlü çizim, örneğin çizgisel veya tematik haritalar kullanılabilir. Mevcut kaynak materyal; PC, çalışma istasyonu veya sayısal görüntü işleme sistemi ile birlikte kullanılan sayısallaştırma masası üzerine konur, gerekli yönelt me işlemleri yapılır ve sonra da sayısallaştırma cihazı ile kaynak materyal üzerindeki istenen veriler manuel olarak toplanır. Manuel sayısallaştırmanın esası, sayısallaştırma cihazının tıklanmasıyla veya ucunun sayısallaştırma masasına dokundurulmasıyla, masa üzerindeki düzlem(x, y) koordinatlarını veri depolama donanımelemanına iletmektir.

Çizgilerin sayısallaştırmasında çizgiyi temsil edecek karakteristik noktaların seçimi operatörün kontrolündedir. Nokta modundaki bu sayısallaştırmaya alternatif yöntemler geliştirilmiştir. Dört farklı sayısallaştırma modu vardır. Genelde bu modların uygun kombinasyonları ile sayısallaştırma gerçekleştirilir. Manuel sayısallaştırma türleri şunlardır [49]:

- *Nokta sayısallaştırma:* Her tuşa basışta bir nokta sayısallaştırılır. Çizgi detayların sayısallaştırılmasından çok, nokta detayların sayısallaştırılmasına uygundur.
- *Zaman arttırımlı sayısallaştırma:* Tuşa basılı olarak çizgi izlenirken belli zaman aralıklarında noktalar kendiliğinden sayısallaştırılır. Bu mod operatörü belli bir baskı altında tutmakla beraber daha iyi sonuç vermektedir. Özellikle münhani karakterindeki çizgilerin sayısallaştırılmasında çizginin dönüş aldığı yerlerde operatörün izlemesi ister istemez yavaşladığından buralarda daha sık nokta sayısallaştırılmaktadır.
- *Mesafe artımlı sayısallaştırma:* Son sayısallaştırılan noktadan belli bir mesafe uzaklaşıldığında yeni bir nokta kendiliğinden sayısallaştırılır. Artım miktarı çok verildiğinde girinti çıkıntılar iyi temsil edilmemekte, az verildiğinde ise düz yerlerde gereksiz fazlalıkta nokta alınmaktadır.

- *Mesafe Artımlı ve Doğrultudan Sapma Kontrollü Sayısallaştırma:* Son sayısallaştırılan noktadan belli bir mesafe uzaklaşıldığında veya sayısallaştırılmış son iki nokta oluşturduğu doğrultudan belli bir mesafeden fazla ayrıldığında yeni bir nokta kendiliğinden sayısallaştırılır. Bu yöntemle sayısallaştırmada sayısallaştırılan her noktada bir dikdörtgen başlatıldığı düşünülür. Bu dikdörtgenin uzun ekseni bir önceki sayısallaştırma noktası ile son noktayı birleştiren doğrultudadır ve eni ile boyu programla belirlidir. İmlecin bu dikdörtgeni kısa veya uzun kenarından terk ettiği noktada yeni bir nokta kendiliğinden sayısallaştırılır ve yeni bir dikdörtgen oluşturulur.

6.2.1.2 Otomatik çizgi izleyerek sayısallaştırma

Bu sayısallaştırma sisteminin en önemli özelliği raster tarama ve etkileşimli sayısallaştırmanın üstün yanlarının birleşmesidir. Raster taramanın hız ve duyarlılığı ile operatörün düşünce ve kontrol becerilerinin birleştirildiği bu yöntemde operatörün gösterdiği çizgiler otomatik olarak bir raster tarama şeridi ile izlenerek sayısallaştırılmaktadır. Başka bir deyişle bilgisayarca okunabilir ortama dönüşüm tekniği raster iken sonuç doğrudan vektörel olarak elde edilmektedir.

Tipik bir sayısallaştırma esnasında operatör kalibrasyon işlemlerinden sonra, konsol üzerindeki izleme topu ile ekrandaki imleçe komuta ederek ilk sayısallaştırılacak çizgiyi gösterir. Tarayıcı çizgiyi bulur ve yine operatörün gösterdiği yönde enine taramalarla izler. Sayısallaştırma esnasında sayısallaştırma doğrultusu kontrol edilmek suretiyle seyrekleştirme anında yapılabilmektedir. Sayısallaştırma operatör kontrolünde yapıldığından eksik veya tekrarlı sayısallaştırma genellikle olmaz ve sayısallaştırılan çizginin orijinali ile uyumu çok iyidir. Ancak yine de bir kontrol çizimi alınarak gerekiyorsa etkileşimli grafik editleme istasyonunda gerekli düzeltmeler yapılabilir [49].

6.2.1.3 Ekrandan sayısallaştırma (heads-up digitizing) yöntemi ile coğrafi veri toplama

Operatör mouse yardımıyla, toplayacağı detayları bilgisayar ekranında görülen taranmış (raster) veya sayısal formda bulunan görüntü üzerinden sayısallaştırır. Bu işlemi çinileç yardımıyla detay üzerinde veya detayın sınırlarında hareket ettirilir ve bu sırada da fare tuşuna basılarak belli aralıklarda nokta sayısallaştırılır.

Sayısallaştırma sıklığı detayın şekline, çalışılan ölçeğe ve istenilen hassasiyete göre değişir. Sonuçta iki boyutlu sayısal vektör harita elde edilir. İki boyutlu veriler elde mevcut SYM verileri veya sayısal eş yükseklik eğrisi verileri ile birleştirilerek üç boyutlu hale dönüştürülebilir.

6.2.2. Raster'dan vektöre dönüşüm yöntemi ile coğrafi veri toplama

Kaynak materyal yüzeyinin taranması sonunda bilgisayar ortamında elde edilen raster formattaki verilerin, uygun yazılımlar kullanılarak otomatik veya yarı-otomatik biçimde vektörel konum verilerine dönüştürülmesi işlemidir. Daha çok çizgi detayların sayısallaştırılması için uygun olan bu yöntemin girdileri; her türlü basılı materyal, baskı kalıbı ve güncel revizyon bilgi kalıbıdır.

Üzerinden veri toplanacak olan kaynak materyal, büyük boyutlu ve hassas tarayıcılarda taranır. Tarama siyah-beyaz, gri skala veya renkli gerçekleştirilir. Tarama sonucunda hiçbir anlamı ifade etmeyen bir raster dosya elde edilmiş olur. Daha sonra bir yazılım yardımıyla bu raster dosya vektör hale dönüştürülür. Raster'dan vektöre dönüşüm yapan bir çok ticari yazılım bulunmaktadır. Dönüşüm sonrası oluşan vektör veride; kapanmayan alanlar, yanlış yerlere birleşmiş çizgiler, dönüşmemiş detaylar, istenmeyen detaylar gibi bir çok problemler bulunur. Sorunsuz bir dönüşüm sağlamak pratikte imkansızdır. Bu nedenle yöntem ne kadar az ilave editleme yükü getirirse o kadar başarılı olmuş olur. Dönüşüm sonrasında oluşan hataları düzeltmek için gerekli süre, o kaynak dokümanın en baştan sayısallaştırılması için gerekli süreden az ise, raster'dan vektöre dönüşüm yöntemi kullanılmalıdır.

6.2.3. Video kayıt ile coğrafi veri toplama

Video sayısallaştırıcılar 'framegrabber' adlı özel bir analog-sayısal dönüştürücüye bağlı bir kameradan oluşmaktadır. Framegrabber video cihazından (video kasette kayıtlı) veya televizyondan gelen NTSC veya PAL gibi bir formattaki analog sinyali alır. Bu sinyal üzerindeki renk bileşenlerini ve senkronizasyon bilgisini önce bir sinyal üzerine kodlar. Daha sonra bu sinyaldeki senkronizasyon bilgisini çıkarır ve renkleri kırmızı, yeşil, mavi (RGB) bileşenlerine ayırır. Son olarak her renk kanalındaki 8 bitlik video sayısallaştırma mekanizması bu üç analog renk sinyalini

sayısala çevirerek manyetik ortama kaydeder veya anında bir grafik ekrandan (bu artık bir televizyon ekranı değil bilgisayar ekranıdır) görüntüler. İşlemcisi hızlı bir çalışma istasyonunun ekranında hareketli sayısal film görüntüsü bile elde etmek mümkündür. Tek resim veya resim dizisi kullanılarak sayısallaştırma yapılabilir. Çok fazla kullanılmayan bir yöntemdir [51].

6.2.4. Fotogrametrik yöntemle coğrafi veri toplama

Fotogrametri, CBS'nin temel veri kaynaklarından bir tanesidir. Fotogrametri aynı zamanda CBS oluşturulmasında gereksinme duyulan üç boyutlu sayısal verilerin toplanmasında kullanılan en önemli veri toplama yöntemlerinden birisidir. Fotogrametri bir yandan vektör verileri en gelişmiş teknolojilerle CBS kullanımına sunarken, diğer yandan da yüksek doğrulukta hazırlanmış raster formdaki görüntülerle CBS'ni geometrik veri açısından desteklemektedir.

CBS için veri toplama yöntemleri sınıflandırılmasında fotogrametrik yöntemle veri toplama ve uzaktan algılama ile veri toplama yöntemleri birbirlerinden ayrılmıştır. Bundan sonraki bölümlerde anlatılacak fotogrametrik yöntemle veri toplama tekniğinde, veri toplama kaynağı olarak sadece fotoğraf ele alınmıştır.

Fotogrametrik olarak üretilen çeşitli klasik analog ve sayısal harita verileri ile görüntüler, CBS'nde güncel birer altlık olarak kullanılmaktadır. Fotogrametri ve CBS temelde birbirinden ayrık olarak gelişen iki ayrı teknoloji olup, her ikisi de bilgisayar, donanım, yazılım, algılayıcı ve çıktı aletleri ile bilgi iletişim teknolojilerinden çok fazla etkilenmişlerdir. CBS amaçlı veri toplamada fotogrametrik yöntemlerin sağladığı belirgin olanaklar şunlardır [51]:

- Yüksek doğruluk
- Arazinin eksiksiz olarak kapatılması ve ölçümü
- Ölçüm ve değerlendirme işlemlerinin oldukça kolay ve hızlı olması
- Arazi çalışmalarının çok az olması ve çalışmaların büyük çoğunluğunun büroda yapılabilmesi
- Jeodezik yöntemlerle eşdeğerde doğruluklar elde edilmesi
- Fotogrametrik çalışmaların her zaman her koşulda yapılabilmesi
- Otomasyona yatkın olması

Fotogrametrik yöntemlerle sayısal ürünler elde edilmesi sonucunda, fotogrametri ile CBS arasında yakın bir ilişki doğmuştur. CBS, hızlı ve otomatik olarak güncel veri toplama işleminin yapılması amacıyla fotogrametriye gereksinme duymaktadır. Güncel görüntü verilerini içermeyen bir CBS'nin mevcut halihazır durumu gösterdiği söylenemez. Aynı şekilde, SYM verilerini içermeyen bir CBS, düz bir arazi ile sınırlandırılmış olur. Fotogrametri bu aşamada CBS'ni tamamlayıcı, eksikliklerini giderici, vektörel veri yanında bazı öznitelik bilgilerini de CBS'ye sunması nedeniyle CBS'yi destekleyici bir rol oynamaktadır. Kısaca, fotogrametri ile CBS birbirleriyle bütünleşmiş durumdadır.

CBS için fotogrametrik yöntemle veri toplamada kaynak materyal olarak stereo veya monoskopik hava fotoğrafları kullanılmaktadır. Stereo görüntüler analitik ve/veya dijital stereo kıymetlendirme aletleri ile değerlendirilerek detaylara ilişkin X, Y, Z koordinatları kaydedilir.

Fotogrametrik yöntemle coğrafi veri toplama yöntemi aşağıdaki başlıklar ile gruplandırılabilir [51]:

- Sayısal çıkışlı analog aletlerde, hava fotoğraflarından sayısal veri toplama
- Analitik aletlerde hava fotoğraflarından sayısal veri toplama
- Dijital aletlerde (soft-copy), taranmış hava fotoğraflarından veya dijital kamera ile çekilmiş görüntülerden sayısal veri toplama
- Taranmış monoskopik hava fotoğraflarından elde edilen sayısal ortofoto görüntülerden sayısal veri toplama
- Mevcut sayısal harita, CVT veya CBS verilerinin hava fotoğrafları kullanılarak bilgisayar destekli revizyon yoluile veri toplama

Dijital aletlerle veri toplama işlemi tamamen sayısal olarak uygulanan bir yöntemdir. Artık fotogrametrik olarak sayısal veri toplama işlemi bu teknoloji ile gerçekleştirilmektedir. Bu yöntemin girdisi dijital kameralar ile elde edilmiş veya mevcut fotoğrafların taranmasıyla elde edilmiş sayısal görüntülerdir. Dijital aletlerle SYM verileri otomatik olarak elde edilmekte, bu verilerle otomatik olarak eş yükseklik eğrileri üretilebilmektedir. Dijital alet ve yöntemlerle, sayısal vektör veriler yanında, hava fotoğrafları ve uydu görüntüleri yardımıyla raster yapıdaki verilerin (ortofoto görüntü, ortofoto harita, foto-mozaik, zenginleştirilmiş ortofoto gibi) üretimi de olanaklıdır. Donanımolarak dijital aletlerin bakımı, onarımı, yedek

parça temini ve arıza giderme işlemleri, diğer stereo kıymetlendirme aletlerine oranla daha kolaydır.

6.2.5. Uzaktan algılama yöntemi ile coğrafi veri toplama

Uzaktan algılama yöntemi ile fotogrametrik yöntem arasındaki temel fark veri toplamak için kullanılan materyalin farklı oluşudur. Fotogrametrik yöntemde siyah/beyaz, renkli (sadece üç bant: kırmızı, mavi ve yeşil) ve kızılötesi hava fotoğrafları kullanılırken uzaktan algılama yönteminde kullanılan görüntüler çok daha geniş bir yelpazeye yayılmaktadır. Elektromanyetik spektrumun çok geniş bir bölümü uzaktan algılamada görüntü verisi oluşturmak için kullanılabilir. Görünen dalgaboylarından, termal ve radar görüntülere kadar çok farklı amaçlar için kullanılabilecek uzaktan algılama görüntüleri elde edilebilir.

Bu görüntüler kullanılarak; görüntü işleme istasyonlarında çeşitli algoritmalar uygulanarak sınıflandırma, değişiklik izleme vb. yöntemlerle farklı disiplinler için çeşitli veriler elde edilebilmesi mümkündür.

Çeşitli elektromanyetik bantlarda toplanan bu veriler çeşitli tekniklerle işlendiğinde;

- Planimetrik (x,y) konumu,
- Topoğrafik/batimetrik (z) yüksekliği,
- Obje rengi,
- Bitki klorofil soğurma karakteristikleri,
- Bitki örtüsü biokütlesi,
- Bitki nemiçeriği,
- Toprak nemiçeriği,
- Isı,
- Yeryüzü dokusu, gibi zengin biyofizik bilgileri türetilebilmektedir.

6.2.6. Doğrudan arazi ölçmeleri ile coğrafi veri toplama

Bu yöntemde, çeşitli ölçümlerle arazide detay ve yüksekliklere ilişkin doğrudan üç boyutlu sayısal konum verisi ve detaya ilişkin grafik olmayan öznitelik bilgilerini toplamaya olanak sağlayan total station, GPS gibi gelişmiş aletler kullanılır. Bu aletlerden elde edilen verilerin doğruluğu son derece yüksek olup, doğrudan CBS

ortamına aktarılabilir. Bu aletlerin yanısıra, elektronik takeometre, takeometre, teodolit, topoğraf alidatı gibi klasik topoğrafik ve jeodezik aletlerle de detay ve yüksekliklere ilişkin üç boyutlu veriler toplanabilir. Elde edilen bu çizgisel verilerin sayısal hale dönüştürülmesi ve CBS ortamına aktarılması, daha önce açıklanan yöntemlerden herhangi birisi ile sağlanır.

6.2.7. Alfasayısal bilgi girişi ile coğrafi veri toplama

Bu veri kaynakları ile, CBS'nde yer alan grafik verilere ilişkin öznitelik bilgileri toplanır. Öznitelik bilgilerinin grafik verilerle ilişkilendirilmesi ile CBS'ne veri girişi tam olarak sağlanmış olur. CBS'ne veri girişi tamamlandıktan sonra yapılması gereken şey, güncel veri kaynakları kullanılarak CBS'nin grafik ve öznitelik verilerinin güncelleştirilmesidir.

CBS'ne girilecek olan grafik olmayan öznitelik bilgilerinin büyük bir kısmı kağıt formlar üzerine yazılıdır. Kaynak materyal olarak bu basılı formlar ve altlıklar kullanılır. Bir bilgisayara bağlı alfasayısal terminal ve bir editleme yazılımı, öznitelik bilgilerinin girişi için yeterlidir. Yerinde bilgi kaydı için bugünkü el bilgisayarları veya diz üstü bilgisayarlar büyük avantajlar sağlamaktadır. ASCII veya EBCDIC karakterler olarak girilen alfasayısal bilgiler daha sonra grafik verilerle ilişkilendirilmek üzere CBS'nin kurulacağı bilgisayara transfer edilmektedir [51].

Bir veri giriş operatörünün ortalama hızı saatte 12000 tuştur. Öznitelik değerlerinin belli kodlarla ifadesi girilecek veri hacmini azaltır. Öznitelik verileri yığın kütükler halinde hazırlanıp grafik verilerle yığın olarak ilişkilendirilebileceği gibi, ekran başında etkileşimli olarak da ilişkilendirilebilir. Veri girişinde yapılabilecek hatalara karşı çeşitli kontrol mekanizmaları geliştirilmiştir [53]. Yöntemi tek başına kullanmak yerine, genellikle grafik veri toplamaya olanak sağlayan diğer veri toplama yöntemlerinin yanında, öznitelik bilgileri açısından tamamlayıcı ve bütünleyici bir şekilde kullanmak gerekmektedir.

6.2.8. Mevcut sayısal verilerin ithali yöntemi ile coğrafi veri toplama

CBS oluşturulması ve sayısal harita üretimi uygulamaları giderek yaygınlaştıkça, her geçen gün üretilen sayısal coğrafi bilgi kütüklerinin sayısı da artmaktadır. Coğrafi bilgi toplamının en kolay ve ucuz yolu, daha önce üretilmiş olan coğrafi bilgi

kütüklerini, sayısal haritaları veya sayısal coğrafi veri tabanlarını; teyp, disket, CD-ROM gibi off-line ortamlarda veya bilgisayar ağları üzerinden on-line olarak transfer programları yardımıyla CBS ortamına aktarmaktır [53]. Bu aşamada başka ortamlardan alınan bu sayısal verilerin, mevcut CBS yapısına uygun hale getirilmesine gerek duyulmaktadır.

Özellikle kamu kurum ve kuruluşlarının gerçekleştirmiş olduğu karayolu, demiryolu, enerji nakil hattı, baraj, gölet, sulama kanalı gibi projelere ilişkin grafik çizgisel ve sayısal veriler, CBS oluşturulmasında veri kaynakları olarak kullanılabilir. Başlı başına bir CBS kurulmasında bu veri kaynaklarından tamamen yararlanmaktan ziyade, daha önce üretimi yapılan sayısal ve çizgisel haritalarda meydana gelen yeni değişikliklerin işlenmesi, eksik olan grafik verilerin büro revizyonu ile tamamlanması amacıyla kullanılabilirler [51].

6.2.9. Hava fotoğraflarından ve uydu görüntülerinden otomatik ve/veya yarı otomatik detay çıkarma yöntemleri ile veri toplama

Son yıllarda üzerinde çalışılan ve farklı başarı derecelerine sahip, hava fotoğraflarından, uydu görüntülerinden ve raster görüntülerden otomatik ve yarı otomatik vektör veri elde etme yöntemleri, CBS için birer veri toplama yöntemi olarak değerlendirilebilir.

Söz konusu yöntemlerin araştırma ve uygulama çalışmalarına, bilgisayar ve yazılım teknolojilerinin son yirmi yılda gelişmesiyle birlikte motive kazandırılmıştır. Özellikle yarı otomatik yöntemler üzerindeki çalışmaların sayısı oldukça artmıştır.

Bu yöntemler ile ilgili bilgiler 2, 3 ve 4'üncü bölümlerde detaylarıyla incelenmiştir. Bu araştırma çalışması kapsamında geliştirilen "Düzey Kümesi ile Bölümleme" yöntemi yarı otomatik detay çıkarma yöntemi olarak ele alınabilir. Bu yeni yöntem, bir sonraki bölümde ayrıntıları ile tanıtılacak ve CBS için veri toplama yöntemi olarak etkin bir şekilde kullanılıp, kullanılamayacağı ve hangi hata aralığında veri toplama yeteneğine sahip olacağı bir sonraki bölümde gerçekleştirilecek test çalışmaları sonucunda tespit edilecektir.

Kullanılacak yöntemister otomatik olsun ister yarı otomatik olsun istenen geometrik doğruluk sınırları içerisinde operatör tarafından elle kıymetlendirilen verileri

bilgisayar yardımıyla daha hızlı ve kolay bir şekilde toplayıp, veri toplama süresini azaltıp maliyetleri düşürmesi gerekmektedir. Aksi durumda bu yöntemlerin başarısından söz etmek anlamsız olacaktır.

6.3. Coğrafi Veri Toplama Yöntemlerinin Değerlendirilmesi

Manuel sayısallaştırma yönteminde veri toplama hızı ve elde edilen doğruluk oldukça düşüktür. Kullanılan kaynak materyalin güncel olmaması durumunda toplanacak sayısal veriler de güncel olmayacaktır. Bu yöntemile elde edilen grafik kütük çoğunlukla sphagetti yapıdadır. Bu nedenle bu kütüklerin daha sonra yapılandırma programları ile kenar-düğüm veya daha ileri düzeyde topolojik veri yapılarına çevrilmeleri gerekir. Manuel sayısallaştırma yöntemi, nokta ve çizgi detayların sayısallaştırılması için uygun bir yöntemdir. Sonuçta iki boyutlu bir vektörel kütük üretilir. Duyarlığı büyük ölçüde operatörün kartoğrafik çizim becerisine ve dikkatine bağlıdır. Daha çok detay yoğunluğu az olan kaynakların sayısallaştırılmasında kullanılır. Pek çok dezavantajlara sahip olan bu yöntemin, CBS oluşturulmasında bağımsız bir veri toplama yöntemi olarak kullanılması mümkün görülmemektedir. Yöntemin, diğer yöntemleri tamamlayıcı ve bütünleyici bir özelliğe sahip olan yardımcı bir yöntem olarak kullanılması düşünülebilir. Yöntem günümüzde nadir olarak kullanılmaktadır [48].

Otomatik çizgi izleyerek sayısallaştırma yöntemi, çizgi detayların sayısallaştırılmasına uygundur. Manuel sayısallaştırmaya göre daha hızlı veri toplanır. Özellikle münhani (eş yükselti eğrisi) kalıplarının sayısallaştırılması için ideal bir çözümdür. Tematik haritaların bu yöntemle sayısallaştırılması oldukça güçtür. Duyarlığı ve hızı çok iyidir. Ancak sayısallaştırma öncesi filme kopyalama zorunluluğu işletim maliyetini arttırmaktadır. Fazla kullanılan bir yöntem değildir [48].

Ekrandan heads-up sayısallaştırma yönteminde, güncel kaynaklar sayısallaştırılıp kullanılırsa güncel veriler elde edilir. Çalışma istasyonu, PC veya görüntü işleme sisteminde ekrandan sayısallaştırma yapacak operatörlerin eğitimi çok kısa sürede sağlanmakta olup, bu alanda deneyimli operatöre gereksinme duyulmamaktadır. Veri toplama maliyeti düşüktür. Eğer ekrandaki görüntü üzerinde, grafik verilere ilişkin non-grafik veriler mevcut ise, öznitelik bilgileri de grafik verilerin toplanması

esnasında toplanabilir. CBS kurulması ve veri tabanı oluşturulmasında halen oldukça yaygın olarak kullanılan bir yöntemdir. Dezavantajları ise; zaman alıcı olması, elde edilen sonuç sayısal verilerin doğruluğunun nispeten düşük düzeylerde olması, bazı kaçınılmaz geometrik hataların oluşması, çalışmanın hızının donanımle direkt bağlantılı olması, kullanılan tarayıcının hassasiyetinin ekrana gelen görüntünün kalitesine, dolayısıyla da verinin doğruluğuna etki etmesi sayılabilir. Ayrıca aynı bölgeye ilişkin olarak biri çizgisel (basılı veya kalıp), diğeri sayısal olmak üzere iki kez veri toplanmaktadır [48].

Raster'dan vektöre dönüşümle coğrafi veri toplama, gerek sürekli tonda (fotoğraf, tematik harita gibi) gerekse çizgisel kaynak materyalin sayısallaştırılması için hızlı ve duyarlı bir yöntemdir. Daha çok çizgi tipindeki detayların toplanmasına uygun olan bu yöntem, doğru yerde ve doğru şekilde kullanıldığında sayısallaştırma ile veri toplamadan daha hızlı sonuçlar vermekte, böylece de maliyeti düşürmektedir. Bu yöntemde kaynak materyalin kalitesi ile içerdiği detay çeşitliliği ve kullanılan yazılımın özellikleri çok önemlidir. Kartoğrafik kalitesi düşük bir kaynak materyal duyarlığı etkilemesinin yanısıra editleme için çok zaman gerektirmektedir. Ayrıca bu yöntemin etkin biçimde kullanılabilmesi için kaliteli bir tarayıcıya da ihtiyaç duyulmaktadır [48].

Video kayıt ile coğrafi veri toplama yöntemi bugün için çok yaygın olmamakla beraber, gelecekte kullanımının artacağı değerlendirilmektedir. Hava fotoğraflarının gerek tek resim olarak gerekse dijital fotogrametride çift resim olarak kullanılması, video sayısallaştırma yönteminin gelişmesine yol açacaktır [48].

Fotoğrafik yöntemle toplanan detay verileri doğrudan CBS'ye aktarılabilecek üç boyutlu sayısal verilerdir. Toplanan veriler vektörel formda ve sphagetti yapıdadır. Sphagetti yapıdaki sayısal verilerde görülen kopukluk, taşma, boşluk ve üst üste binme (undershoot, overshoot, gap, sliver) gibi geometrik sorunlar etkileşimli editleme ile giderilse dahi CBS'ne aktarıldıktan sonra yapılandırılmaları ve kenar-düğüm veri yapısına çevrilmeleri gerekir. Fotogrametrik yöntemle toplanan bu vektörel verilerin harita sayısallaştırması ile toplanan vektörel verilerden en önemli farkı üç boyutlu (X, Y, Z üçlülerinden oluşan vektörler) olmalarıdır. Fotogrametrik yöntemle toplanan verilerin bir diğer önemli özelliği de duyarlılığıdır. Bugünkü fotogrametrik kıymetlendirme cihazlarında, resim ölçeğine ve aletin duyarlılığına

bağlı olarak, örneğin 1/ 25 000 ölçekli sayısallaştırmada detaylar maksimum ± 1 m gibi bir duyarlıkta sayısallaştırılabilmektedir. CBS için fotogrametrik yöntemle veri toplamanın bir diğer avantajı ise daha güncel bilgiler toplayabilme olanağıdır. Ayrıca dijital alet ve yöntemlerle, sayısal vektör veriler yanında, hava fotoğrafları ve uydu görüntüleri yardımıyla raster yapıdaki verilerin (ortofoto görüntü, ortofoto harita, foto-mozaik, zenginleştirilmiş ortofoto gibi) üretimi de olanaklıdır. Fotogrametrik yöntemle coğrafi veri toplama yöntemi deneyimli operatöre ihtiyaç duyar. İyi bir operatörün yetişmesi uzun zaman alır. Ayrıca veri toplama işlemi yorucudur ve diğer yöntemlere göre daha uzun sürer. Fotogrametrik yöntemle veri toplama işleminde grafik veriler toplandıktan sonra, bir kısım verilerin doğruluğu, eksiklerin giderilmesi ve öznitelik bilgilerinin toplanabilmesi için arazide topoğrafik bütünleme yapılması gerekir [51].

Günümüzde CBS yazılımları giderek artan oranda uzaktan algılama kaynaklı coğrafi verilerin kullanımına olanak sağlamaktadır. Çünkü, uzaktan algılama özellikle tematik amaçlı çalışmalarda alan detayların toplanması için küçük ölçeklerde etkin ve ekonomik bir yöntem durumundadır. Günümüzde bu amaçla en yaygın olarak kullanılan uydu görüntüleri; LANDSAT-TM, LANDSAT-MSS ve SPOT-HRV görüntüleridir. Bu görüntüler sayısal görüntü işleme (DIP) teknikleri ile işlenerek, raster veya vektör formda birçok coğrafi veri üretilmektedir. Güncel stereo uydu görüntüleri ile veri toplandığı taktirde, elde edilen sonuç veriler de güncel olmaktadır. Uydu görüntüleri ile daha geniş alanların haritasının yapılması kısa sürede olanaklı olduğundan, yöntemin maliyeti düşük ve ekonomiktir. Ayrıca doğrudan CBS'nde girdi verisi olarak kullanılabilecek sayısal veriler üretilmektedir. Günümüzde mevcut mono veya stereo uydu görüntüleri ile 1/ 25 000 ve daha büyük ölçekli sayısal harita üretimi ile CBS oluşturulması olanaklı görülmemektedir. Stereo uydu görüntüleri yurtiçi bölgelerin yanısıra, daha ziyade yurtdışı ve hudut bölgelerine ait topoğrafik harita üretimi ve revizyonunda kullanılırlar. Ancak bu durumda güncel, güvenilir, doğru ve uluslararası datum ve projeksiyonlarla uyumlu altlık haritalara gereksinim duyulur. Ayrıca, yurtdışı bölgelere ait topoğrafik harita üretimi sonucunda, arazide topoğrafik bütünleme yapılması ve öznitelik bilgilerinin toplanması olanaklı değildir. Uzaktan algılama ile veri toplama yöntemi için şu dezavantajları da sıralayabiliriz. Yöntemin doğruluğu, kullanılan stereo uydu görüntülerinin arazideki ayırma gücüne bağlıdır. Uydu görüntüleri ile oluşturulan

stereo modeller yardımıyla orta ve küçük ölçekli topoğrafik harita üretiminde, tüm detayların görülebilmesi ve yorumlanabilmesi olanaklı değildir. Arazide topoğrafik harita bütünlemesine çok fazla gereksinme gösterir [51].

CBS oluşturmada birçok öznitelik bilgisini haritalardan okuyarak toplamak imkansızdır. Ayrıca öznitelik bilgilerinin hava fotoğraflarından toplanması, çeşitli yorumlama zorlukları nedeniyle oldukça sınırlıdır. Dolayısıyla öznitelik verilerinin çoğunun arazide toplanması gerekir [48].

Alfasayısal öznitelik bilgileri CBS projelerinde önemsenmemekle beraber, yüksek maliyeti dikkate alınarak, bu bilgilerin etkili yöntemlerle toplanmasına özen gösterilmesi gerekmektedir [48].

Coğrafi bilgi toplamanın en ucuz yolu, daha önce üretilmiş olan coğrafi bilgi kütükleri, sayısal harita verileri ve sayısal topoğrafik veri tabanlarını mevcut CBS içerisine aktarmaktır. Ancak coğrafi bilgi kütüklerinin özellikleri için ne yazık ki etkin bir standardizasyona ulaşılamamıştır. Birbirinden çok farklı özelliklerdeki coğrafi verileri aynı CBS ortamında kullanabilmek için, her şeyden önce belirli standartların tanımlanması gerekir. Üretilen sayısal coğrafi veriler birçok bakımdan birbirlerinden farklılaşmaktadırlar. Bu farklılıklar şunlardır:

- Detay ve öznitelik sınıflandırması ve kodlaması
- Coğrafi veri yapısı ve geometrik özellikler
- Ölçek (veri sıklığı)
- Sembolleştirme
- Sayısal veri transferi formatı.

Kartoğrafik sayısallaştırma yöntemleriyle veri toplama, jeodezik ve fotogrametrik veri toplama yöntemlerine oranla daha hızlı, düşük maliyetli olup, doğrulukları da düşüktür [48].

Jeodezik yöntemde kullanılan gelişmiş elektronik aletler ile kullanıcının deneyimine bağlı olarak, istenen duyarlıkta sayısal veri toplanması mümkündür. Jeodezik yöntemlerle, ulaşılabilen her arazi kesiminde ölçüm yapılması olanaklıdır. Yanına yaklaşılamayan ayrıntılar ölçülemez, bu nedenle jeodezik yöntemler doğa koşullarından çok fazla etkilenmektedir. Fotogrametrik yöntemle hava fotoğrafları üzerinde bulunan bütün ayrıntıları duyarlı bir şekilde sayısallaştırmak mümkündür.

Fotogrametrik yöntemler daha çok büro çalışmaları ile yürütülmekte olup, sayısallaştırma işlemleri kolayca yapılmaktadır. Bilgi sisteminin temelini oluşturan verilerin güncel olması gerekmektedir. Güncelleştirme çalışmalarının hızlı bir şekilde yapılabilmesi için fotogrametrik yöntemlerden yararlanılmaktadır [48].

Otomatik ve yarı otomatik yöntemler, hava fotoğrafları ve uydu görüntüleri üzerindeki tüm detayları değil sadece çizgi detaylar ve alan detayların sınırlarının sayısallaştırılmasına uygundur. Manuel sayısallaştırmaya göre daha hızlı ve daha duyarlı olduğu durumlar söz konusudur ancak özellikle otomatik yöntemlerin daha geliştirilmeye ihtiyacı vardır. Önümüzdeki yıllarda bu yöntemlerin geliştirilmesiyle birlikte çok daha fazla kullanılacağı düşünülmektedir. Toplanan veriler topolojik yapıda olmadığı ve bazı kesiklikleri içerdiği düşünüldüğünde mutlaksurette toplanan verilerin bir operatör veya başka araçlar kullanılarak düzeltilmesi ve yapılandırılması gerekmektedir.

7. UYGULAMA

Bu çalışmada, dijital hava fotoğraflarından çizgisel detayların yarı otomatik olarak çizilmesi amaçlanmıştır. Böylece insan zekası ve değerlendirme yeteneğiyle birlikte bilgisayarların hesaplama hızının birleştirilmesi suretiyle harita üretiminde ve CBS için veri toplama işlemlerinde en çok zaman alan fotogrametrik veri toplama sürelerinin kısaltılması hedeflenmiştir.

Bu amaç ve hedef doğrultusunda, renk farkları ile düzey kümesi yöntemlerini baz alan algoritmaları kullanan bir uygulama yazılımı tasarlanmış ve geliştirilmiştir. Söz konusu yazılım, Borland C++ ve Visual C++ görsel yazılım dilleri kullanılarak nesneye dayalı bir kodlama ile gerçekleştirilmiştir.

Geliştirilen yazılımın test edilmesi amacıyla 4 farklı ölçekte ve özelliklerde seçilmiş hava fotoğraflarıyla birlikte 1 adet IKONOS ve 1 adet SPOT uydu görüntüsü üzerinde çizgisel detayların değerlendirilmesi çalışmaları gerçekleştirilmiştir.

Bu yazılımın, CBS için veri toplama kaynaklarından biri olabileceği ve CBS için gerekli olan vektör verilerin toplanmasında kullanılabileceği düşünüldüğünden; İstanbul Teknik Üniversitesi, Ayazağa Kampüsünün bir bölümünü içeren 1:16 000 ölçekli renkli hava fotoğraflarından oluşturulmuş 0.5m mekansal ayırma gücüne sahip, iki adet 1:5000 ölçekli ortofoto görüntüden koordinatlı olarak yol ve bina detayları geliştirilen yazılımla yarı otomatik olarak çizilmiştir. Elde edilen verilerin geometrik doğruluklarının tespit edilmesi maksadıyla aynı detaylar bir operatör yardımıyla tamamen elle çizilmiş ve bu veriler doğru olarak kabul edilerek, yarı otomatik toplanan verilerle karşılaştırılarak bir doğruluk araştırması yapılarak geliştirilen yöntemin hata kriterleri belirlenmeye çalışılmıştır.

7.1. Yazılımda Kullanılan Algoritmalar ve Kodlama

Tasarlanan yarı otomatik veri toplama yöntemi 5' inci bölümde açıklanan düzey kümesiyle görüntü bölümleme algoritmalarına dayandırılmıştır. Burada çözülmesi gereken üç problemle karşılaşılmıştır. Birincisi; algoritmanın nasıl başlatılacağıdır.

Bu problem, operatör tarafından çizilmek istenen detayın üzerinde herhangi bir noktanın (pikselin) işaretlenmesi suretiyle çözülmüştür. Böylece düzey kümesi algoritması operatörün seçmiş olduğu noktadan itibaren çalışmaya başlayacaktır.

İkinci problemi ise işaretlenen noktadan itibaren hangi kritere göre detayın çizilmesi işleminin ilerleyip ilerlemeyeceğine karar verileceğidir. Bu problem ise raster görüntülerin meydana geldiği her bir pikselin sahip olduğu renk değerlerinden faydalanılarak çözülmüştür. İşaretlenen noktanın sahip olduğu renk değeri veya belirlenen komşuluk derecesinde, komşu piksellerin renk değerleri ile ortalamasının alınarak bulunan renk değeri, komşu piksellerin renk değerlerleri ile karşılaştırılmış ve renk farkı belirlenen bir tolerans değeri içinde kalmışsa algoritma devam ettirilmiş, değilse durdurulmuştur. Başka bir deyişle renk farklılığına bağlı bir bölümleme gerçekleştirilmiştir.

Çizilmesi istenen detay belirlendikten ve işaretleme işleri tamamlandıktan sonra bu detayın bir vektör veri olarak elde edilmesi gerekmektedir ki; bu sayede harita üretiminde veya CBS'nde kullanılabilsin. Bu da üçüncü problemi oluşturmaktadır. Bu problemin çözümü içinde raster veriden vektör veriye dönüşüm işlemi gerçekleştirilmiştir.

Yukarıda bahsedilen üç problemi çözen yarı otomatik veri toplama yönteminin iş akışı, 5 ana adımdan oluşan bir işlemler bütünü şeklinde tasarlanmıştır. Bu işlem adımları:

- Operatör tarafından değerlendirilmek istenen detayın çizilmesi için başlangıç noktasının veya pikselinin seçimi,
- Seçilen görüntü pikselinin, komşu piksellerle olan renk farkından faydalanılarak görüntü bölümlemesinin gerçekleştirilmesi,
- Düzey kümesi yöntemi kullanılarak bölümlemenin yayılması ve yığın yapısı şeklinde depolanması,
- Bu yapıdaki piksellerden 1 bitlik (1 renkli) raster maske görüntüsünün elde edilmesi
- Elde edilen raster maskeden koordinatlı olarak, raster görüntüden vektöre dönüştürme işleminin gerçekleştirilmesini müteakiben detaylara ait bilinen bir formatta vektör verilerinin elde edilmesidir.

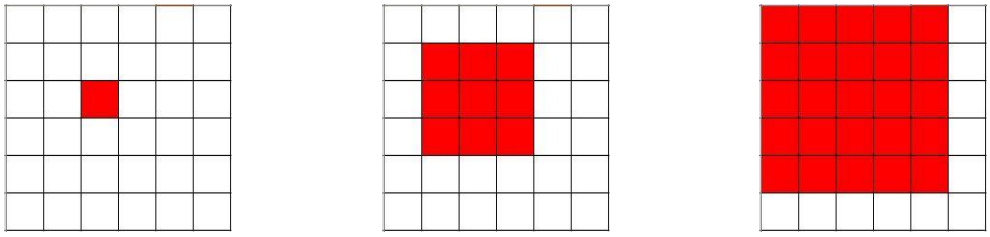
İlk 4 işlem adımı Borland C++ programlama dili kullanılarak ve sistem kütüphaneleri dışında hiçbir kütüphane dosyası kullanılmadan gerçekleştirilmiştir [54]. 5'nci adımi se internet ortamında [55] de bulunan, bir raster görüntüden vektöre dönüşüm açık kodunun, Visual C++ kullanılarak yeniden düzenlenip, kırıklık toleransı ve köşe koordinatları giriş seçeneklerinin ilave edilerek işlevsellik kazandırılmasıyla gerçekleştirilmiştir.

Tasarlanan ve uygulanan yazılımının iş akış grafiği EK-A'da ve ana program parçası ise EK-B'de verilmiştir.

Problemlerin çözümü için ortaya konan tasarılar ve bunların uygulamaya dökülerek bilgisayar programı haline konması, sırasıyla ayrı başlıklar altında tartışılmıştır.

7.1.1. Görüntü bölümleme algoritması

Görüntü bölümlemesi piksellerdeki renk farkına dayandırılmış olup, bu basit algoritmaya operatör tarafından ayarlanabilir bir tolerans değeri eklenerek esneklik kazandırılmıştır. Böylece operatöre, kontrastlığın fazla olduğu yerlerdeki detaylarda toleransı yüksek tutarak sadece bir kez işaretlemek suretiyle büyük alanları seçebilme kabiliyeti kazandırılması öngörülmüştür. Bölümleme algoritmasının eşik değeri olarak görüntünün genişliği ve boyunun çarpımının 2 katı kadar bir değer belirlenmiştir.



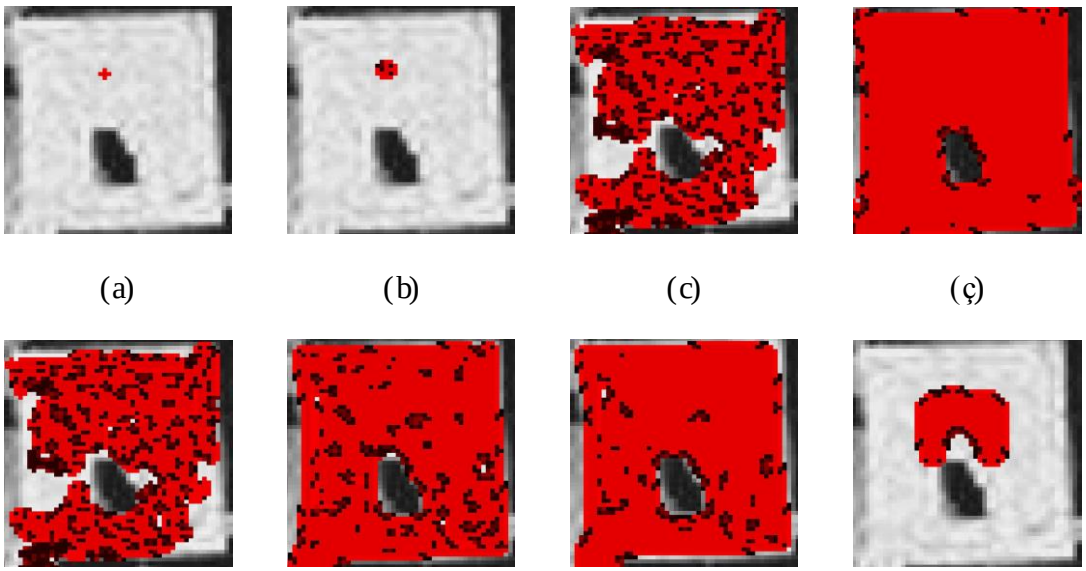
(a) (b) (c)

Şekil 7.1 Komşuluk derecesi kavramı (a) Komşuluk derecesi 0 (b) Komşuluk derecesi 1 (c) Komşuluk derecesi 2

Bölümleme, ilk olarak işaretlenen detay noktası temel alınarak başlatılır. Bu detay noktasına denk gelen pikselin ve/veya belirlenen komşuluk derecesine göre komşu piksellerin renk değerleri ile ortalaması alınması sonucunda elde edilen renk değeri referans olarak alınır. Komşuluk derecesi 0 ise, o pikselin renk değeri, komşuluk derecesi 1 ise etrafındaki 8 piksel ile birlikte ortalaması alınması sonucunda bulunan renk değeri başlangıç olarak alınır (Şekil 7.1).

Burada referans noktasının renk değerleri ile komşu piksellerin renk değerleri arasındaki fark hesaplanırken üç banttaki (kırmızı, mavi ve yeşil) renk farklılıkları ayrı ayrı hesaplanır ve her üçünün de tolerans değerinden küçük olup olmadığı araştırılır. Bu algoritmaya göre Borland C++ ortamında kodlanan görüntü bölümlemesine (renk farklarının hesaplanmasına) ait program parçası EK-C' de verilmiştir.

Tolerans değeri ve komşu piksel derecesi değerlerinin detay belirleme işlemine etkileri Şekil 7.2'de gösterilmiştir. İlk dört şekilde komşuluk derecesi 0 olarak alınmış, başka bir deyişle işaretlenen noktanın renk değerleri referans olarak alınmış olup, diğer piksellerle bir ortalama alma işlemi gerçekleştirilmemiş ve sadece tolerans değerleri sırasıyla 1, 5, 10 ve 25 olarak belirlenmiştir (Şekil 7.2(a), (b), (c), (ç)). İkinci dört şekilde ise tolerans değeri 10 olarak ayarlanmış olup, sabit olarak alınmış ve sadece komşuluk derecesi sırasıyla 1, 2 ve 16 olarak değiştirilmiştir (Şekil 7.2(d), (e), (f), (g)). Burada farklılıkların hatasız olarak tespit edilebilmesi amacıyla her şekilde aynı piksel başlangıç pikseli olarak seçilmiştir (Şekil 7.2(a)). Bu araştırma göstermiştir ki; komşu piksel derecesinin arttırılması her zaman iyi sonuçlar vermemektedir. Özellikle seçilen piksele yakın pikseller değerlendirilmeye katıldığında sonuçlar oldukça iyileşmekte fakat pikselden çok uzaklaşınca aynı sonuçları elde etmek için daha büyük tolerans değerine ihtiyaç duyulmaktadır.



(a) (b) (c) (ç)

(d) (e) (f) (g)

Şekil 7.2 Tolerans değeri ve komşu piksel derecesi değerlerinin detay belirleme ve çizimişlemlerine etkileri.

7.1.2. Düzey kümesi fonksiyonu ile yayılma ve yığın şeklinde depolama

Düzey kümesi algoritması ile yayılmanın yada detay boyunca ilerlemenin gerçekleştirilebilmesi için gerekli olan başlangıç değerine, geliştirilen yöntemin yarı otomatik doğasıyla birlikte çözüm bulunmuştur. Yüzeyin yayılmasına, operatör tarafından işaretlenen pikselden itibaren başlanacaktır ve sıfır düzey kümesi fonksiyonu bu pikselin konumu ile tanımlanacaktır.

Düzey kümesi fonksiyonu için gerekli olan diğer bir bileşen ise yayılmanın kontrolünü sağlayan sınır değeridir. Sınır değeri için çözüm yukarıda anlatılan renk farkının hesaplanmasıyla gerçekleştirilmiştir.

Geriye, yayılmanın başlatıldıktan sonra, sınır değerini sağlayan komşu piksellerden hangisinden devam edeceği probleminin çözümü kalmıştır. Aslında bu problemin çözümü, Bölüm 5.4.3'de anlatılan "Hızlı İlerleme Yönteminde" açıkça ifade edilmişti. Fonksiyonun, en küçük değere sahip pikselden itibaren yayılmaya devam edeceği çok açıktır. Fakat burada ifade edilmeyen, en küçük değerin ne olacağıdır?

Geliştirilen yöntemde, en küçük değer neyi temsil etmelidir sorusu için operatör tarafından işaretlenen ilk piksel sıfır düzey kümesi fonksiyonu olarak kabul edilmiş ve her komşu piksel (doğu, batı, kuzey, güney yönündeki ilk pikseller) yukarıda açıklanan renk farkı algoritmasıyla kontrol edilmiş ve şartı sağlayan pikseller için sıfır düzey kümesinden (ilk işaretlenen piksel) olan mesafeler (d) hesaplanmıştır ve böylece en küçük mesafeye sahip pikselden itibaren yayılmanın devam ettirilmesi sağlanmıştır.

Burada kendimize şu soruyu sorabiliriz: Mesafeler eşitse durum ne olacaktır? Bu başlangıçtan sonraki ilk adımda ortaya çıkabilecek bir problemdir çünkü başlangıçta dört komşu piksel vardır ve şartı sağlayan pikseller aynı uzaklığa (sadece bir birim uzunluğa) sahiptirler. Böyle bir durumda, en küçük renk farkına sahip pikselden itibaren yayılma devam ettirilerek probleme adil bir çözüm getirilmiştir.

Söz konusu çözümleri gerçekleştirebilmek için ilgilenilen her bir piksel için bir **görüntü hücresi** nesnesi yaratılması gerekmiştir, çünkü tek başına bir piksel gerekli olan tüm özellikleri sağlamak için yeterli olamamaktadır. Söz konusu nesnenin sahip olduğu özellikler ve başlangıç değerleri Tablo 7.1'de gösterilmiştir.

Tablo 7.1. Görüntü hücresi nesnesi ve sahip olduğu özellikler ile başlangıç değerleri.

GÖRÜNTÜ HÜCRESİ	
ÖZELLİKLERİ	**BAŞLANGIÇ DEĞERLERİ**
X koordinatı	İşaretlenen pikselin x yönündeki resim koordinatı
Y koordinatı	İşaretlenen pikselin y yönündeki resim koordinatı
Kırmızı renk değeri	İşaretlenen pikselin kırmızı renk değeri
Mavi renk değeri	İşaretlenen pikselin mavi renk değeri
Yeşil renk değeri	İşaretlenen pikselin yeşil renk değeri
Mesafesi	0.0
Konumu	-1
İşaretlenme durumu	Yanlış
İşleme durumu	Yanlış

Yayılmanın tamamlanabilmesi için önemli bir işlem olan ve Bölüm 5.4.3'de anlatılan bir güncelleme işleminin yapılması gerekmektedir. Güncelleme işleminde, mesafenin hesaplanabilmesi için gerekli olan karaselleştirme algoritmaları kullanılmıştır.

Yayılma programı EK-Ç'de, güncelleme, yenileme ve karaselleştirme programları sırasıyla EK-D'de verilmiştir.

İşaretlenen piksellerin depolanması ve erişimi için Bölüm 5.4.4' de anlatılan minimum yığın yapısı kullanılmıştır. Minimum yığın yapısında, en küçük mesafeye sahip görüntü hücresi en tepede olacak şekilde bir yapılanma sağlanmıştır. Yığın yapısının korunması için bazı fonksiyonlara ihtiyaç duyulmaktadır. Yığına görüntü hücresinin eklenmesi, görüntü hücresinin yığında sabitlenmesi, en küçük mesafeye sahip görüntü hücresinin yığından çıkartılması ve yığının her yeni görüntü hücresi eklendiğinde veya çıkartıldığında yapısının yeniden güncellenmesi gerekmektedir. Yığın yapısı ve bu fonksiyonların kodları EK-E'de bulunabilir.

Bu depolama yapısı sayesinde görüntü hücrelerine ulaşılması, yayılmanın test edilmesi ve hesaplanması ile görüntü hücrelerinin işaretlenmesi ve işaretlenen görüntü hücrelerinin depolanması gibi büyük hacimli işlemlerde hesaplama etkinliğinin arttırılması amaçlanmıştır.

7.1.3. Raster veriden vektör veriye dönüşüm

Seçilen detayın yarı otomatik olarak ekran üzerinde işaretlenmesinden sonra bu verilerin harita üretiminde, CBS için gerekli olan coğrafi veri tabanlarının oluşturulmasında kullanılabilmesi ve bu verilere anlam kazandırılabilmesi için vektör veri haline dönüştürülmesi gerekmektedir.

Vektör veri yapısında coğrafi varlıklar nokta, çizgi veya alanlar olarak temsil edilirler. Bu modelde bir coğrafi varlık nokta olarak temsil edilecekse sadece bir koordinat çifti, çizgi şeklinde temsil edilecekse koordinat çiftlerinden oluşan bir küme, alan şeklinde temsil edilecekse başlangıç ve son koordinatları aynı olan koordinat çiftleri kümesi kullanılır [46].

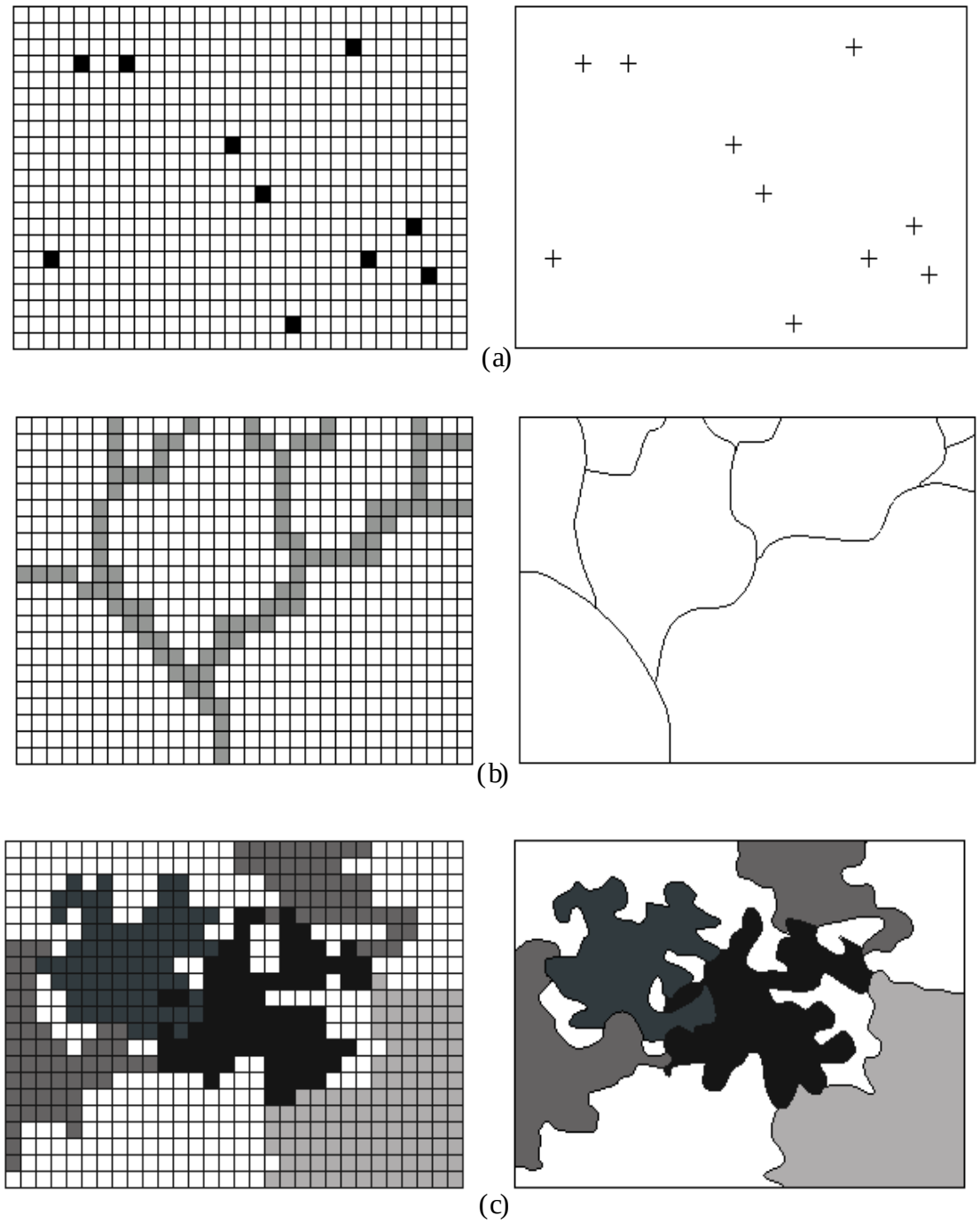
Raster'dan dönüşüm ile vektör veri elde etmek, doğru yerde ve doğru şekilde kullanıldığı zaman hızlı ve etkili bir yöntemdir [48].

Raster'dan nokta detaya dönüşüm yapılırken, her bir piksele karşılık bir nokta detay oluşturulur (Şekil 7.3(a)). Çizgi detaya dönüşüm yapılırken, birbirine komşu sıralı pikselleri çevreleyen alanın merkezinden geçen çizgi elde edilir (Şekil 7.3(b)). Raster veri içerisinde aynı değere sahip birbirine komşu pikseller topluluğu bir alan oluştururlar. Bu alanın dış sınırı çevrilerek, raster veri alan tipindeki vektör detaya dönüştürülür (Şekil 7.3(c)) [47].

Raster'dan vektöre dönüşüm algoritmaları sadece iki tip veri (0 veya 1) içeren 1 bitlik görüntü dosyaları için geçerlidir. Bu yüzden geliştirilen yazılımda, görüntü hücrelerinin seçimi ve işaretlenmesi işlemleri tamamlandıktan sonra, işaretli pikseller dolu, diğerleri ise boş olacak şekilde bir görüntü dizisi oluşturulmaktadır. Bu görüntü dizisinin oluşturulması bir çeşit maskeleme işlemidir. Elde edilen maske raster görüntü daha sonra 1 bitlik (tek renkli) BMP formatında raster görüntü şeklinde kaydedilmektedir.

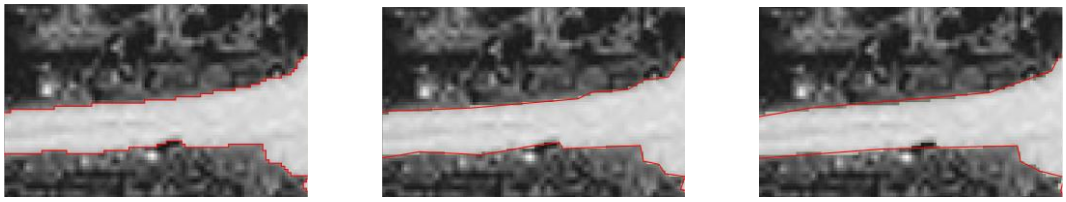
Elde edilen maske görüntü dosyasının, vektör veriye dönüştürülmesi amacıyla Davide Libenzi tarafından geliştirilen ve kendi internet sitesinde açık olarak bulunan Visual C++ kodu yeniden düzenlenmiş ve ilave imkanlar kazandırılmıştır [55]. Ana programla bağlantısı kurularak, etkileşimli bir arayüzde raster veriden, detayların merkez hatlarının ve sınır hatlarının ayrı ayrı vektöre dönüştürülmesi sağlanmış, ana program üzerinde sol alt köşe koordinatları ile her iki boyutta (x,y) görüntü çözünürlüklerinin (mekansal ayırma gücü değeri) girilmesi suretiyle koordinatlı bir

vektör veri elde edilmesi sağlanmıştır. Ayrıca etkileşimli arayüz üzerinde kırıklık toleransının girilmesiyle istenilen yumuşaklıkta ve kırıklıkta vektör veri elde etme olanağı sağlanmıştır. Etkileşimli arayüzü ve fonksiyonlarını içeren kod EK-F'de, Davide Libenzi tarafından geliştirilen kodun değişiklik yapılan ve ilave edilen bölümleri EK-G'de bulunabilir.



Şekil 7.3 Rasterdan vektöre dönüşüm[47].

Kırıklık toleransı ile rasterdan vektöre dönüşüm sonucunda elde edilen vektörün kırıklığı ayarlanmaktadır. Kırıklık toleransı 0 olduğunda hiç yumuşatma yapılmadan tüm pikseller hesaplamaya dahil edilir, kırıklık toleransı arttırıldıkça tüm pikseller değil artan aralıklarla pikseller dikkate alınır ve elde edilen vektör daha düz bir hale gelir fakat kırıklık toleransının çok fazla arttırılması durumunda da geometrik doğruluğun bozulması söz konusudur (Şekil 7.4(a), (b), (c)).

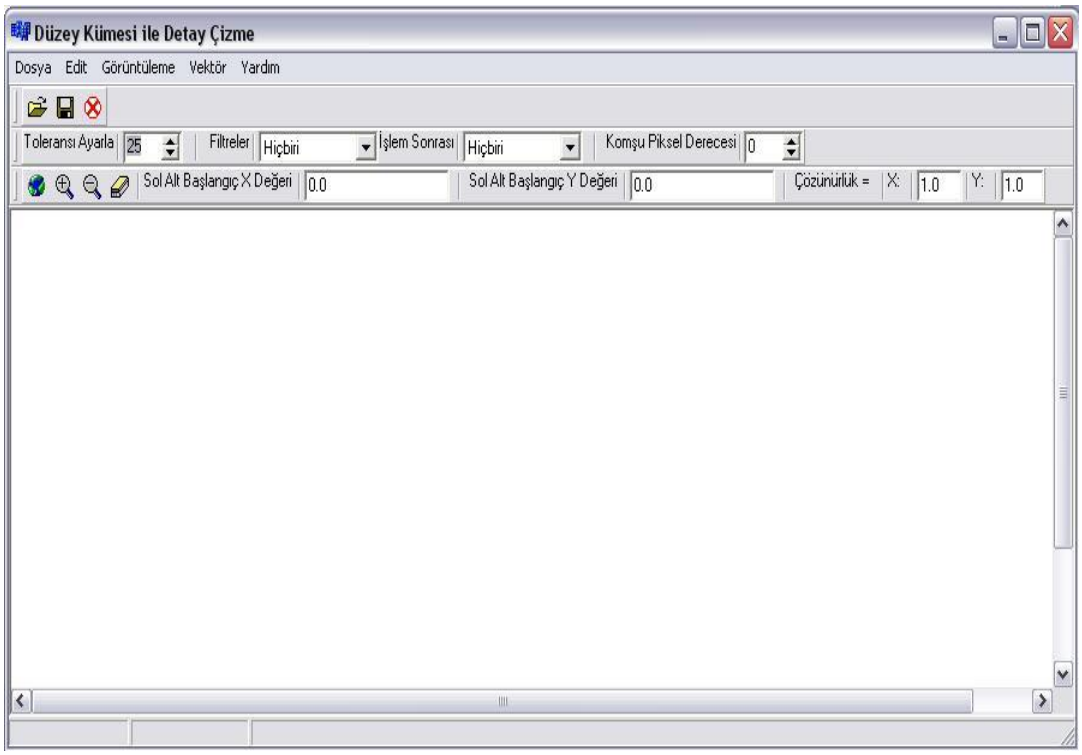


(a) (b) (c)

Şekil 7.4 Kırıklık toleransının vektör veriye dönüştürme işlemine etkisi (a) Kırıklık toleransı = 0 (b) Kırıklık toleransı = 1 (c) Kırıklık toleransı = 2

7.2. Yazılımın Kullanımı

Yazılım iki ayrı çalıştırılabilir (Duzeykumesi.exe ve ras2vek.exe) dosyadan oluşmaktadır. Bu iki dosyanın bilgisayara kopyalanmasıyla her ortamda çalıştırılabilir. Bundan başka herhangi bir kurulum işlemine ihtiyaç duymaz.



Şekil 7.5 Ana arayüz.

Yazılımda iki farklı arayüz mevcuttur. İlki; raster görüntü dosyasının açıldığı ve üzerinde detayların seçilmesi ile işaretlenmesi ve maske raster dosya (bmp formatında) şeklinde kaydetme işlemlerinin gerçekleştirildiği ana arayüzdür (Şekil 7.5). İkinci arayüze (vektör işlemleri penceresi) ise ana arayüzdeki "Vektör" menüsünden ulaşılmakta olup, raster görüntüden vektör veriye dönüştürme işlemleri ise bu etkileşimli arayüzde gerçekleştirilmektedir (Şekil 7.6).

Şekil 7.6 İkinci arayüz (vektör işlemleri penceresi).

Ana arayüzdeki menü, düğme ve kutucukların fonksiyonları ve işlevleri aşağıda açıklanmıştır:

- **Dosya / Aç:** Yazılıma raster görüntü dosyası yükler (JPG, JPEG, BMP, ICO, EMF, WMF formatlarını destekler).
- **Dosya / Kaydet:** Maske raster görüntüyü kaydeder (Tek bitlik BMP formatında).
- **Dosya / Çıkış :** Yazılımı kapatır.
- **Edit / Sil:** Değerlendirme amacıyla seçilen pikselleri temizler.
- **Görüntüleme / Büyütme:** Ekrandaki görüntüyü 2 katı oranında büyütür.
- **Görüntüleme / Küçültme:** Ekrandaki görüntüyü 1/2 katı oranında küçültür. Orijinal boyuttan daha fazla küçültme gerçekleştirilemez.
- **Görüntüleme / Tüm Görüntü:** Büyütülmüş ve/veya küçültülmüş görüntüyü orijinal boyutunda görüntüler.
- **Vektör / Vektör İşlem Penceresi:** İkinci arayüzü (vektör işlemlerinin gerçekleştirildiği pencere) görüntüler ve aktif hale getirir.
- **Yardım / Yardım Konuları :** Yazılımın kullanımı, menü , düğme ve kutucukların fonksiyonları ve işlevleri hakkında bilgileri içerir.

- **Toleransı Ayarla:** Pikseller arasındaki renk farkının alabileceği maksimum değerinin operatör tarafından ayarlanmasını sağlar.

- **Filtreler / Hçbiri / Yumuşatma Filtresi / Medyan Filtresi:** Görüntülerdeki gürültü oranını azaltarak görüntülerin iyileştirilmesi için istenilen filtrenin seçilmesini sağlar. Bir filtre uygulandığında veri toplama işlemine baştan başlanması gerekmektedir çünkü piksel değerlerinde değişiklikler ortaya çıkmaktadır.

- **İşlem Sonrası / Hçbiri / Morfoloji:** İşaretlenen pikseller bmp dosyası olarak kaydedilmeden önce bu seçenek seçilebilir. Böylece oluşturulacak maske raster görüntü kenarlarındaki pikseller yumuşatılabilir.

- **Komşu Piksel Derecesi:** Referans olarak kullanılacak renk değerlerlerinin hesaplanması için kullanılacak değer. 0 durumunda sadece seçilen pikselin renk değerleri alınır, 0 dan faklı olması durumunda etrafindaki komuşu piksel derecesi x 8 adet piksel ile seçilen pikselin renk değerlerinin ortalamasının alınması sonucunda elde edilen renk değerleri kullanılır.

- **Sol Alt Başlangıç X Değeri:** Veri toplama işlemi ortofoto gibi yataylanmış ve koordinatlandırılmış bir görüntüden gerçekleştiriliyorsa maske raster görüntüden vektöre dönüştürme işlemi gerçekleştirilmeden önce mutlaka görüntünün sol alt noktasının sağa değeri metre veya derece cinsinden girilmelidir böylece vektör dosya (DXF formatında) koordinatlı bir şekilde oluşturulabilir.

- **Sol Alt Başlangıç Y Değeri:** Veri toplama işlemi ortofoto gibi yataylanmış ve koordinatlandırılmış bir görüntüden gerçekleştiriliyorsa maske raster görüntüden vektöre dönüştürme işlemi gerçekleştirilmeden önce mutlaka görüntünün sol alt noktasının yukarı değeri metre veya derece cinsinden girilmelidir böylece vektör dosya (DXF formatında) koordinatlı bir şekilde oluşturulabilir.

- **Çözünürlük X Değeri:** Veri toplama işlemi ortofoto gibi yataylanmış ve koordinatlandırılmış bir görüntüden gerçekleştiriliyorsa maske raster görüntüden vektöre dönüştürme işlemi gerçekleştirilmeden önce mutlaka görüntünün X yönündeki çözünürlük değeri (mekansal ayırma gücü) metre veya derece cinsinden girilmelidir böylece vektör dosya (DXF formatında) koordinatlı bir şekilde oluşturulabilir.

- **Çözünürlük Y Değeri:** Veri toplama işlemi ortofoto gibi yataylanmış ve koordinatlandırılmış bir görüntüden gerçekleştiriliyorsa maske raster görüntüden vektöre dönüştürme işlemi gerçekleştirilmeden önce mutlaka görüntünün Y yönündeki çözünürlük (mekansal ayırma gücü) değeri metre veya derece cinsinden girilmelidir böylece vektör dosya (DXF formatında) koordinatlı bir şekilde oluşturulabilir.

Görüntü açıldıktan ve tolerans, filtereler, sol alt nokta ve çözünürlük (mekansal ayırma gücü) değerleri girildikten sonra operatör tarafından istenilen detay üzerine çift data verilerek işlemler başlatılır. Seçme işlemleri tamamlandıktan sonra eğer merkez hatlar vektöre dönüştürülmek isteniyorsa maske raster görüntü cizgi_1.bmp olarak, sınırlar vektöre dönüştürülmek isteniyorsa cizgi_p.bmp olarak kaydedilmelidir.

İkinci arayüzdeki menü, düğme ve kutucukların fonksiyonları ve işlevleri ise şu şekildedir:

- **Vektör İşlemleri / Merkez Hatları → DXF:** Cizgi_1.bmp olarak kaydedilen maske raster görüntüyü merkez hatlarından cizgi olarak DXF formatında vektör dosyaya dönüştürür.

- **Vektör İşlemleri / Sınırları → DXF:** Cizgi_p.bmp olarak kaydedilen maske raster görüntüyü sınırlarından cizgi ve alan olarak DXF formatında vektör dosyaya dönüştürür.

- **Vektör İşlemleri / Vektör Penceresini Kapat:** Sadece ikinci arayüzü kapatır ve ana arayüze geri döner.

- **Vektör İşlemleri / Programdan Çıkış:** Ana arayüz ve ikinci arayüzü birlikte kapatır ve yazılımı sonlandırır.

- **Yardım / Yardım Konuları:** Yazılımın kullanımı, menü , düğme ve kutucukların fonksiyonları ve işlevleri hakkında bilgileri içerir.

- **Kırıklık Toleransı [0..10]:** 0 ile 10 arasında bir reel sayı girilir. Oluşturulacak vektörlerin kırıklıklarının ayarlanmasına yarar.

Vektöre dönüştürme işlemlerine başlamadan önce ana arayüzdeki Sol Alt X ve Y başlangıç değerleri ile çözünürlük (mekansal ayırma gücü) değerleri metre veya derece cinsinden girilmek zorundadır.

Kırıklık toleransı ayarlandıktan sonra hangi tür vektöre dönüştürme işlemi kullanılmak isteniyorsa o düğmeye basılır. Sonuçlar, DOS komut ekranında listelenir, bu ekrandan çıkıp aktif arayüze dönmek için klavyeden herhangi bir karakter girilip, daha sonra Enter tuşuna basılmalıdır. Sonuç vektörler DXF formatında ve merkez hatlar ise cizgi_l.dxf, sınırlar ise cizgi_p.dxf olacak şekilde otomatik olarak kaydedilmektedir

7.3. Yazılıma Gerçekleştirilen Testler

Hava fotoğraflarından yarı otomatik olarak çizgisel detayların çıkarılabilmesi hedefi doğrultusunda geliştirilen ve önceki bölümlerde detaylarıyla açıklanan yazılımın, kullanılabilirliğinin, etkinliğinin ve geometrik hata aralıklarının belirlenebilmesi amacıyla iki grup test çalışması gerçekleştirilmiştir.

Birinci grup test çalışmasında çeşitli ölçeklerde ve özelliklerde dijital hava fotoğrafları ve uydu görüntüleri kullanılmıştır.

İkinci grup çalışmada ise İstanbul Teknik Üniversitesi, Ayazağa Kampüsü'nün bir bölümünü içeren 1:16 000 ölçekli renkli hava fotoğraflarından oluşturulan 0.5m mekansal ayırma gücüne sahip 1:5000 ölçekli iki adet ortofoto görüntü kullanılarak geometrik hata kriterleri belirlenmeye çalışılmıştır.

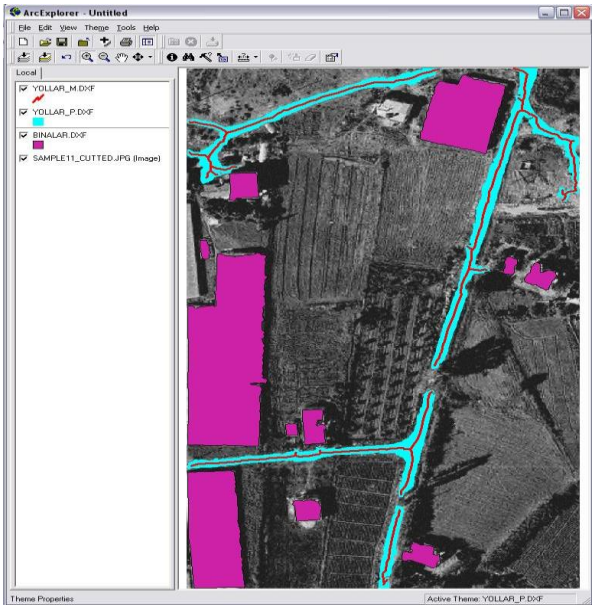
7.3.1. Hava fotoğrafları ve uydu görüntüleri kullanılarak gerçekleştirilen test çalışmaları

Geliştirilen yazılımın test edilmesi amacıyla; 1:4000, 1:16000, 1:25000 ve 1:35000 olamak üzere 4 farklı ölçekte ve özelliklerde seçilmiş hava fotoğraflarıyla birlikte 1 adet IKONOS ve 1 adet SPOT uydu görüntüsü üzerinde çizgisel detayların değerlendirilmesi çalışmaları gerçekleştirilmiştir. Test görüntülerinin seçiminde arazinin farklı dokularına sahip olmasına dikkat edilmiştir. Böylece yazılımile farklı başarı ölçütlerinde elde edilen vektör verilerin hangi arazi dokusuna sahip fotoğraflarda ve hangi ölçeklerde harita üretiminde etkin bir şekilde kullanılabileceğinin tespit edilmesi amaçlanmıştır.

Bu amaç doğrultusunda yapılan çalışmalar ve elde edilen çalışmalar aşağıda ayrı ayrı maddeler olarak anlatılmıştır:

- 1:4000 ölçekli, 20 mikronda taranarak sayısallaştırılmış siyah/beyaz hava fotoğraflarının kullanıldığı birinci test çalışmasında, fotoğraf üzerinde çizgisel detay olarak yollar ve binalar seçilmiştir. Bu seçilen detaylara ait yapılan sayısallaştırmalar sonucunda maske raster görüntüler ve vektör veriler elde edilmiştir (Şekil 7.7). Burada yollar merkez hatlarından ve sınırlarından vektöre dönüştürülmüştür ama aslında merkez hatlarının vektöre dönüştürülmesi çok önemli değildir çünkü bu ölçekteki hava fotoğraflarının kullanıldığı büyük ölçekli fotogrametrik uygulamalarda yollar kenarlarından kıymetlendirilmektedirler. Elde dilen sonuçlar oldukça umut vericidir. Yolların ve binaların üzerine düşen ağaç vb. engeller dışında en küçük tali yollar bile kolaylıkla çıkarılabilmiştir. Detaylar üzerine gelen engellerin neden olduğu kesiklikler bu algoritmayla çözülememiştir ve bu eksiklerinin operatör tarafından elle düzeltilmesi gerekmektedir.

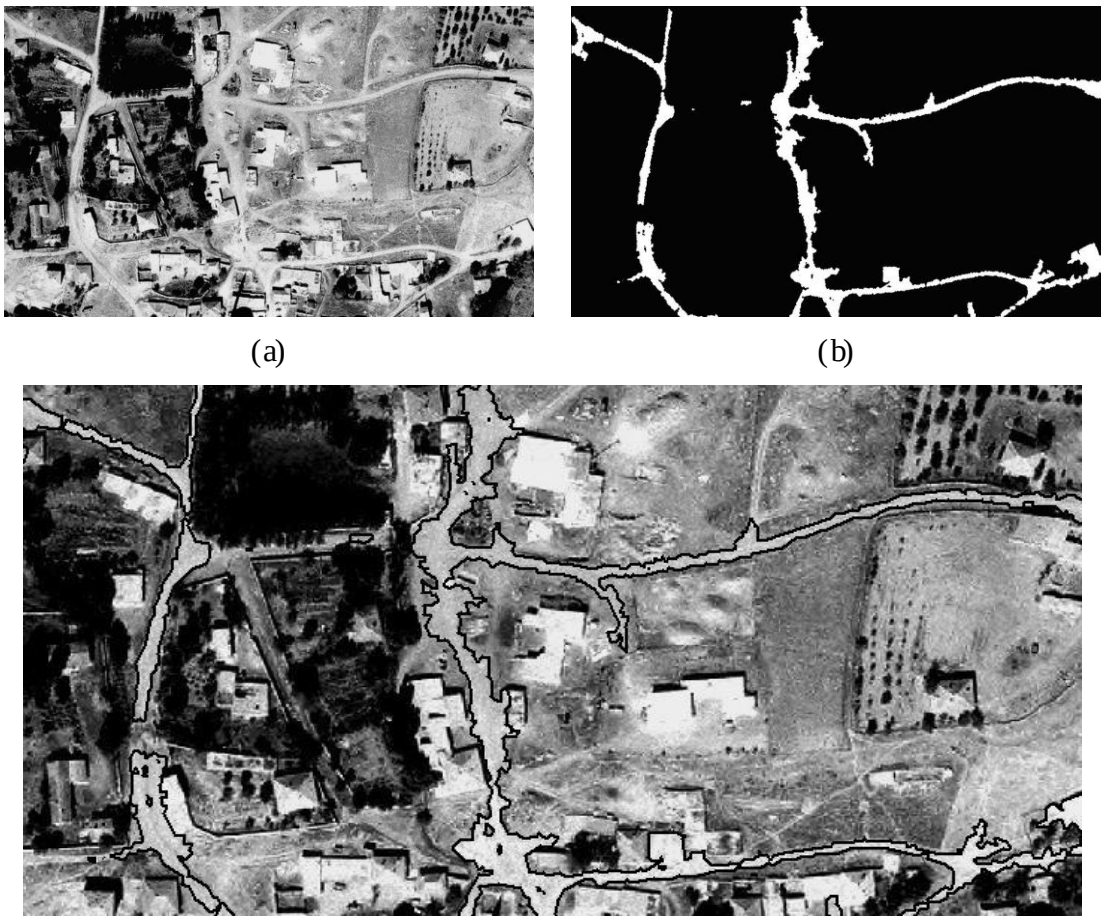
(a) (b) (c)



(ç)

Şekil 7.7 (a) Orijinal görüntü (b) Yollara ait maske raster görüntü (c) Binalara ait raster maske görüntü (ç) Elde edilen vektör veriler

- 1:16000 ölçekli, 20 mikronda taranarak sayısallaştırılmış ve bir köy merkezini içeren siyah/beyaz hava fotoğrafının kullanıldığı ikinci test çalışmasında ise çizgisel detay olarak yollarla ilgilenilmiştir. Binalar burada dikkate alınmamıştır çünkü başarı derecesinin çok az olduğu baştan bellidir. Binaların çatıları çok belirgin değildir ve hava fotoğrafının kontrastlığı binaların kolayca çıkarılabilmesi için elverişli değildir. Köy merkezindeki yolların, yüzey kaplamalarının olmamasının çevrelerindeki detaylardan ayrılmalarını zorlaştırdığı görülmüştür. Burada tolerans değerinin yüksek tutulması durumunda algoritmanın yolların dışına doğru taştığı ve yol olmayan çevre detaylara ait piksellerinde işaretlendiği gözlenmiştir (Şekil 7.8). Bu durumdan kaçınmak için düşük tutulan tolerans değeri de yolların doğru olarak çıkarılması için oldukça fazla işaretleme yapılmasına ve algoritmanın küçük küçük adımlarla ilerlemesine neden olmuştur. Bu istenmeyen bir sonuçtur çünkü yarı otomatik bir algortimanın felsefesine ters düşmektedir. İşlem adımları hızlanmamış bilakis çok yavaşlamıştır.

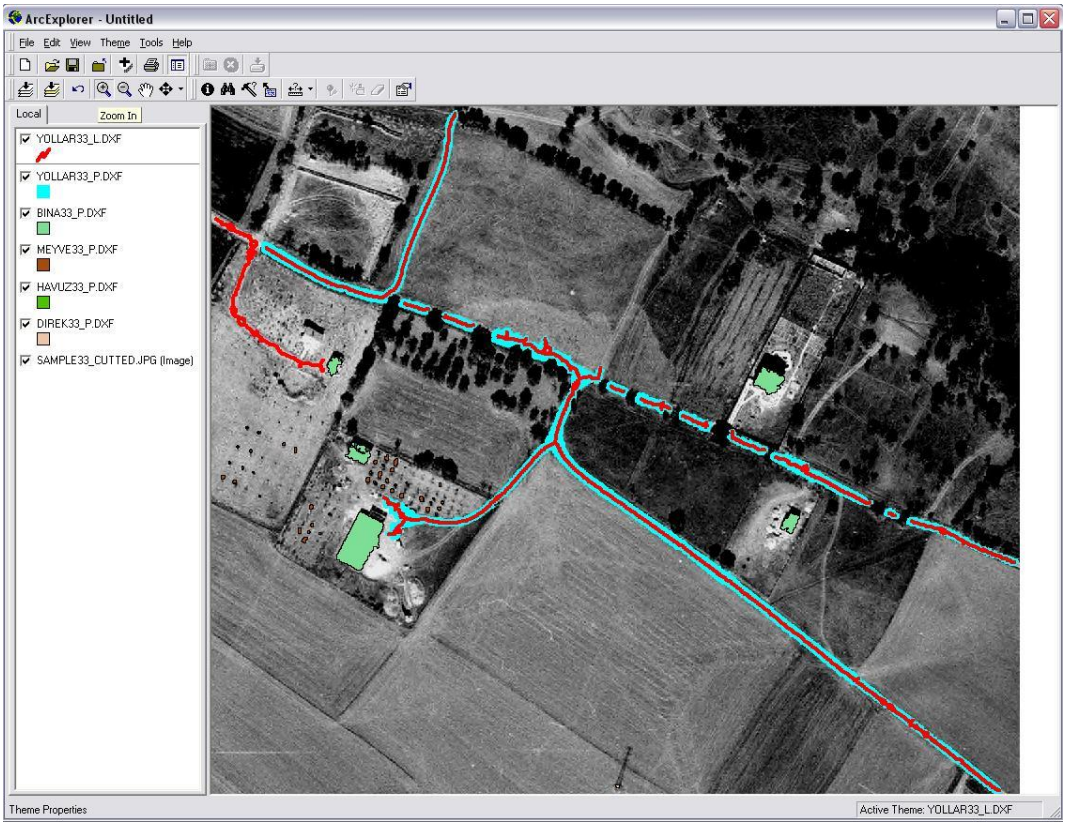


(a) (b)

(c)

Şekil 7.8 (a) Orijinal görüntü (b) Yollara ait maske raster görüntü (c) Elde edilen vektör veriler.

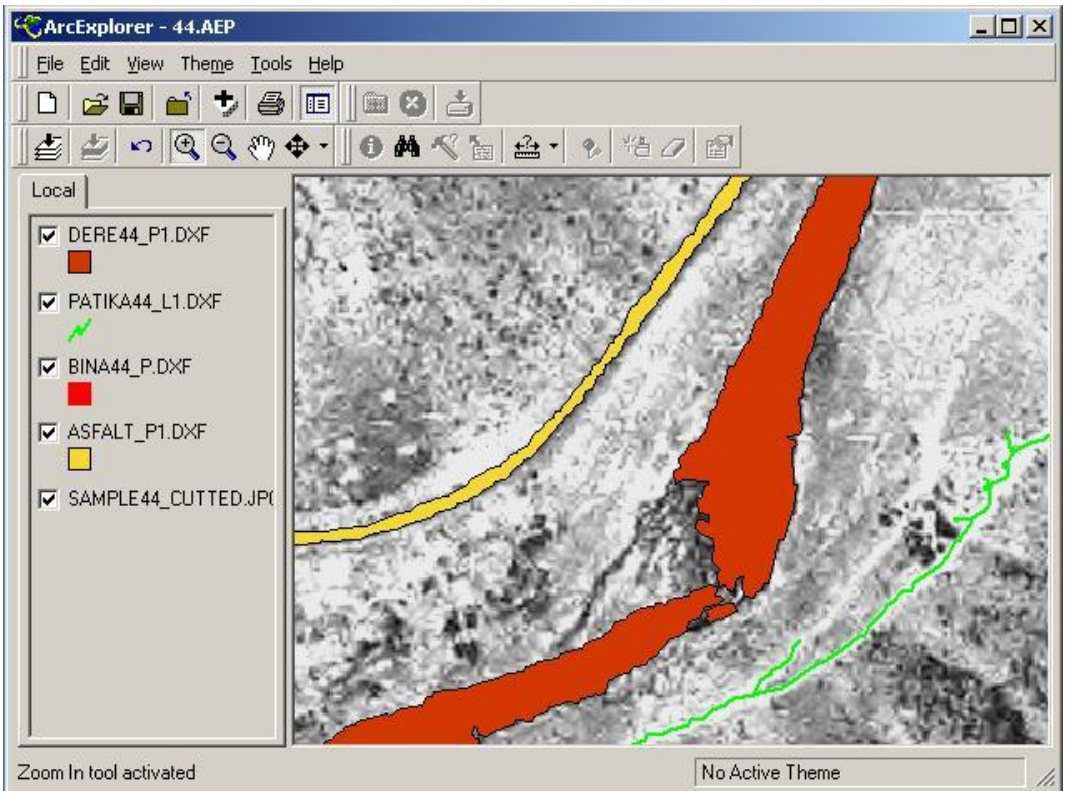
- Üçüncü test çalışmasında; 1:25000 ölçekli, 20 mikronda taranarak sayısallaştırılmış siyah/beyaz hava fotoğrafı kullanılmıştır. Bu test, bütün test verileri arasında en başarılı sonuçların alındığı test olarak ortaya çıkmıştır. Bunun nedenleri araştırıldığında en büyük etkenin görüntü kalitesinin ve kontrastlığının olduğu gözlenmiştir. Kullanılan ölçek küçüldüğünden yukarıdaki test çalışmalarından farklı olarak burada yolların sınırlarına ilave olarak merkez hatların sayısallaştırılmasına da önem verilmiş olup, elde edilen sonuçların çok tatminkar olduğu görülmüştür (Şekil 7.9).



Şekil 7.9 1:25000 ölçekli hava fotoğrafından elde edilen vektör veriler.

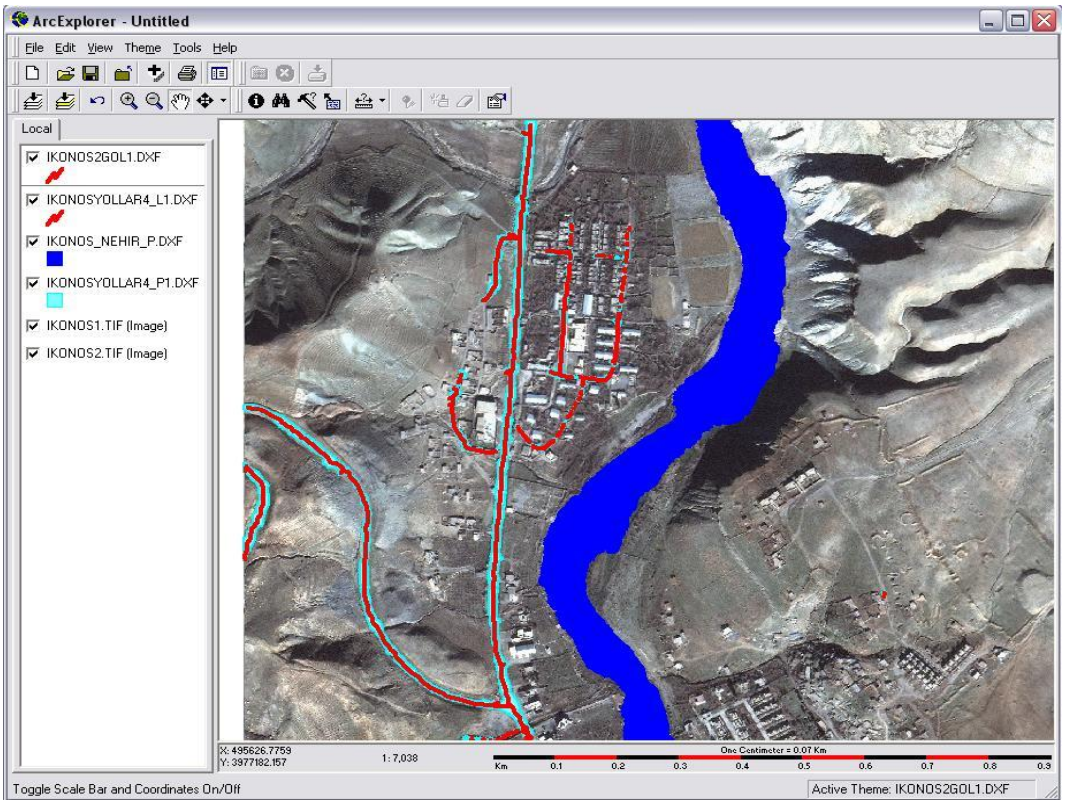
- 1:35000 ölçekli, 20 mikronda taranarak sayısallaştırılmış siyah/beyaz hava fotoğrafındaki 3 ayrı çizgisel detayın çıkarılmasının araştırıldığı dördüncü test çalışmasının sonuçları da oldukça önemlidir. Bu çalışmada, fotoğraf ölçeğinin küçülmesiyle çizgisel detayları örten engellerin (ağaç, araba vb.) etkilerinin azaldığı tespit edilmiştir. Ölçekle birlikte bu engellerin yer aldığı piksel sayısı azaldığından kapsadıkları alanda azalmaktadır. Böylece engellerden kaynaklanan boşluk miktarları daha küçük değerler almaktadır. Bu test çalışmasında, yüzeyi asfalt kaplı bir yol, sulu geniş bir dere ve bir patika sayısallaştırılmaya çalışılmıştır. Asfalt yol ile sulu dere sınırlarından alan şeklinde, patika ise merkez hattından çizgisel olarak

vektöre dönüştürülmüştür (Şekil 7.10). Asfalt yol ve sulu derenin işaretlenmesi sırasında sadece birkaç noktada işaretleme yapılmışken, patikanın işaretlenmesi sırasında tolerans değeri küçük tutulmuş ve küçük küçük adımlarla ilerlenmiştir çünkü patikanın arazi dokusundan ayırt edilmesi oldukça zordur.

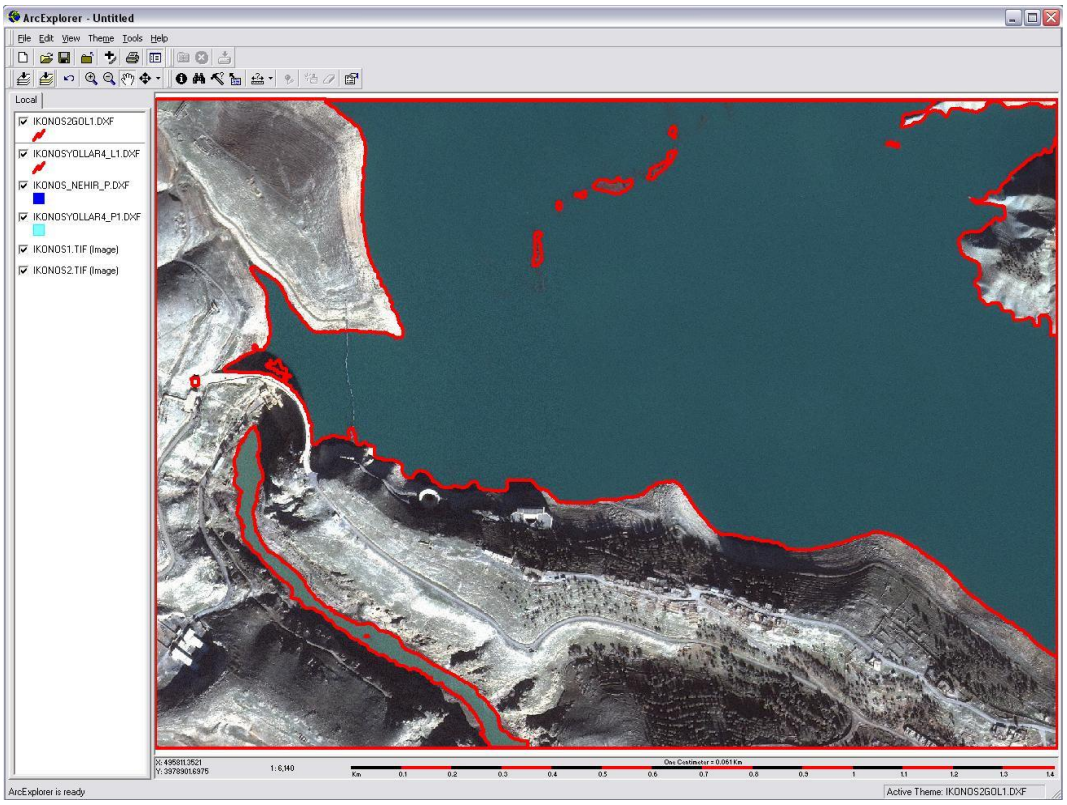


Şekil 7.10 1:35000 ölçekli hava fotoğrafından elde edilen vektör veriler.

- Bu araştırma çalışmasının kapsamına girmemekle beraber uydu görüntülerini kullanarak da test çalışmaları gerçekleştirilmiştir. İlk olarak; 1 m mekansal ayırma gücüne sahip renkli (pan-sharpened yapılmış) IKONOS uydu görüntüsü kullanılmış olup, söz konusu uydu görüntüsünün farklı iki bölgesinde detay belirleme ve çizim işlemleri gerçekleştirilmiştir. Kullanılan uydu görüntüsü ortorektifiye edilmiş bir görüntü olduğundan elde edilen vektör veriler koordinatlı olarak kaydedilebilmiştir. Birinci bölgede yollar, binalar ve sulu dere ile ilgilenilmiş ve çok iyi sonuçlar elde edilmiştir (Şekil 7.11 (a)). İkinci bölgede ise bir göl ve ona bağlı bir sulu dere çıkarılmış ve vektöre dönüştürülmüştür. Bu bölgedeki çalışmada önemli olan nokta, göl ve sulu dere gibi büyük alan detayların sınırlarının, tolerans değeri yüksek tutularak sadece birkaç kez işaretlenmesi suretiyle çok kısa bir sürede sayısallaştırılmasının gerçekleştirilmesidir (Şekil 7.11 (b)).



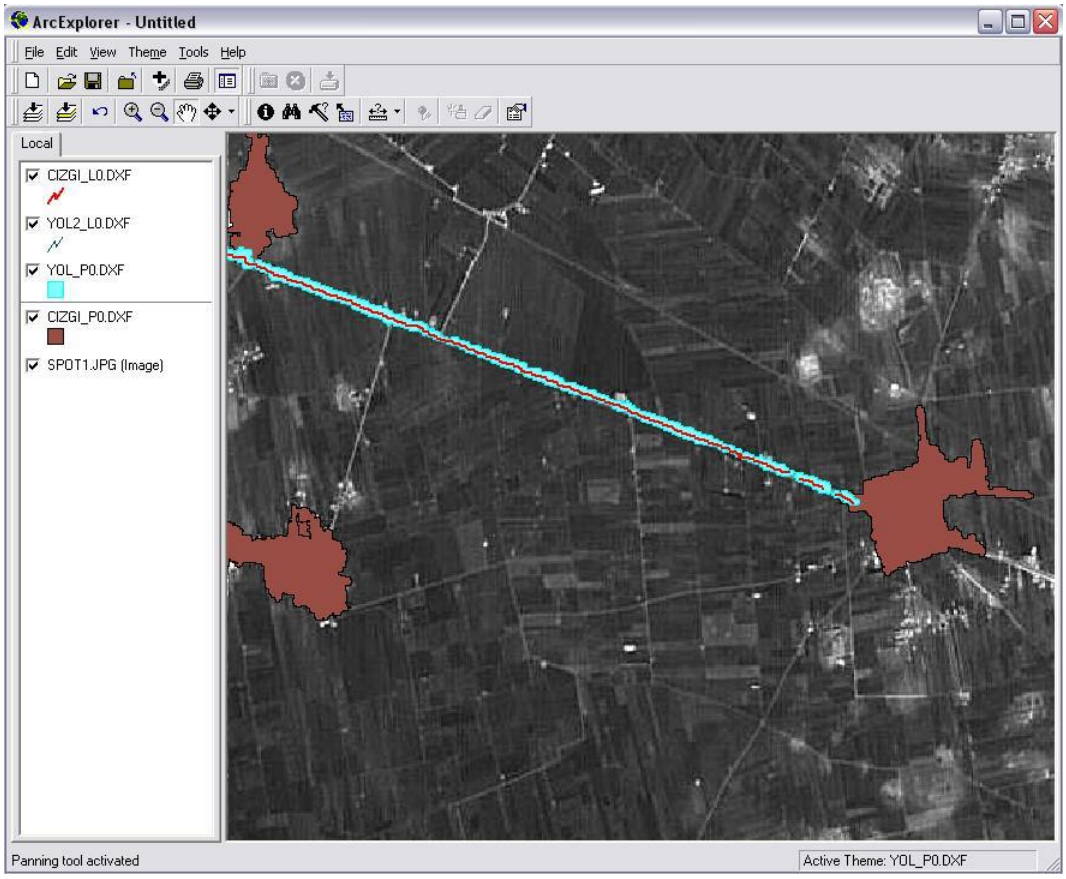
(a)



(b)

Şekil 7.11 (a) IKONOS görüntüsünden elde edilen koordinatlı vektör veriler (yollar, binalar ve dere) (c) Kırmızıyla gösterilen göl ve dere sınırları.

- İkinci olarak pankromatik ve 10 m mekansal ayırma gücüne sahip bir pankromatik SPOT uydu görüntüsü kullanılarak bazı detaylar üzerinde çalışmalar gerçekleştirilmiştir. Bu çalışmada yollara ilave olarak yerleşim yerleri sınırlarıda değerlendirmeye alınmıştır. Oldukça iyi sonuçlar elde edilmiştir. Özellikle yollar konusunda şüpheyle yaklaşılmasına rağmen kaliteli yolların rahatlıkla seçilebildiği görülmüştür (Şekil 7.12).



Şekil 7.12 SPOT görüntüsünden elde edilen vektör veriler (yollar, yerleşim yeri sınırları).

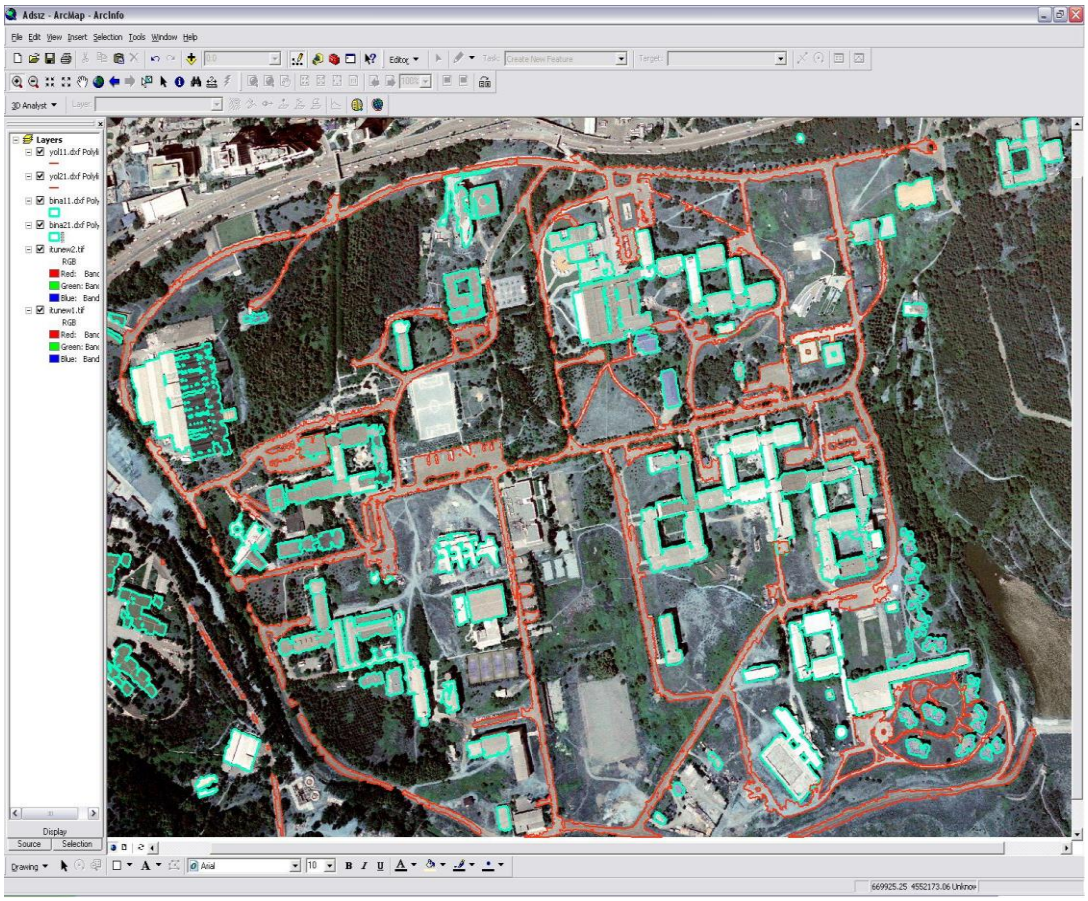
7.3.2 İ.T.Ü ayazağa kampüsünde gerçekleştirilen test çalışması ve yazılımın doğruluk analizi

İstanbul Teknik Üniversitesi, Ayazağa Kampüsünün bir bölümünü içeren 1:16 000 ölçekli renkli hava fotoğraflarından oluşturulmuş 0.5 m mekansal ayırma gücüne sahip, iki adet 1:5000 ölçekli ortofoto görüntüden koordinatlı olarak yol ve bina detayları geliştirilen yazılımla yarı otomatik olarak çizilmiştir. Elde edilen verilerin geometrik doğruluklarının tespit edilmesi maksadıyla aynı detaylar bir uzman operatör yardımıyla tamamen elle çizilmiş ve bu veriler doğru olarak kabul edilerek,

yarı otomatik toplanan verilerle karşılaştırılarak bir doğruluk araştırması yapılarak geliştirilen yöntemin hata kriterleri belirlenmeye çalışılmıştır.

Yarı otomatik veri toplama işlemine başlamadan önce TIFF formatındaki ortofoto görüntüler JPEG görüntü formatına dönüştürülmüştür çünkü geliştirilen yazılım TIFF formatındaki görüntüleri okuyamamaktadır. Görüntüler üzerinde öncelikle yol detayları çıkarılmaya çalışılmıştır. Yol detaylarının çıkartılması tamamlandıktan sonra elde edilen maske görüntü yol.BMP dosyası olarak kaydedilmiştir. Daha sonra bina detayları üzerinde durulmuş ve aynı yöntemle bina detayları da çıkartıldıktan sonra elde edilen maske görüntü bina.BMP dosyası adıyla kaydedilmiştir.

Çeşitli tolerans değerleri uygulanarak çıkartılan ve maske raster görüntüler şeklinde kaydedilen görüntüler, ana arayüzde orto görüntülerin sol alt köşe koordinatlarının girilmesi ve çözünürlüklerin 0.5m ye ayarlanmasından sonra vektör işlemleri penceresinde kırıklık toleransı 0 olarak ayarlanıp ayrı ayrı yol.DXF ve bina.DXF dosyaları şeklinde vektöre dönüştürülmüşlerdir (Şekil 7.13).

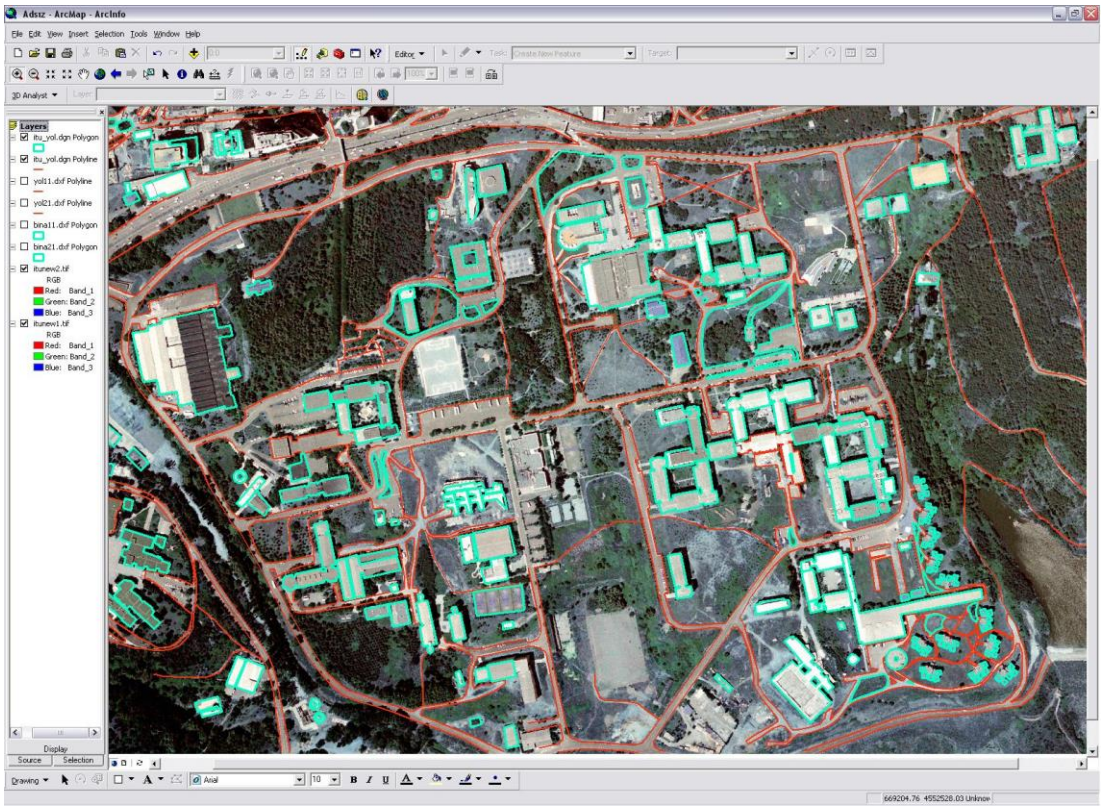


Şekil 7.13 Yarı otomatik olarak elde edilen koordinatlı vektör veriler.

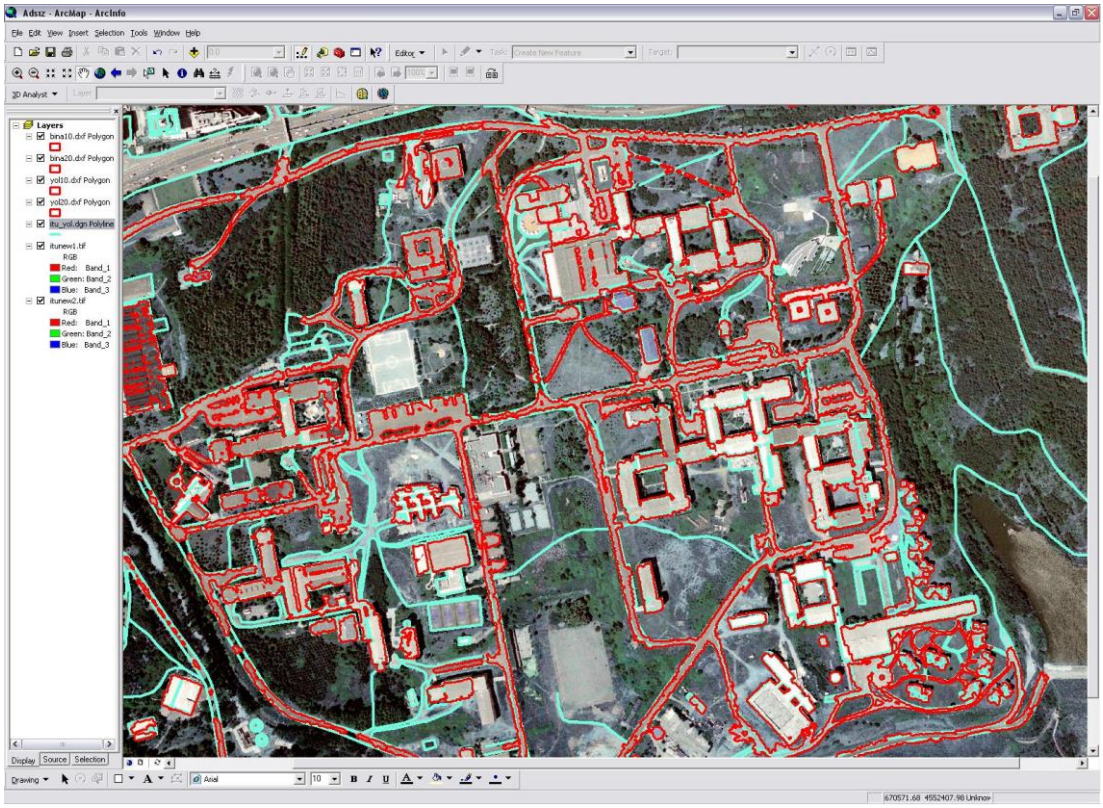
Kırıklık toleransının 0 olarak ayarlanmasının nedeni, bu çalışma sonucunda yöntemin hata kriterlerinin belirleneceğinden kırıklık toleransından kaynaklanacak hataların bu kriterleri etkilemesinin önüne geçmektir. Yarı otomatik detay belirleme ve çizimişlemi yaklaşık yarımgün sürmüştür ve bu veriler kesinlikle hiçbir şekilde düzeltilmemiş olup hamhallerinde doğruluk araştırmasına tabi tutulmuştur.

Yarı otomatik olarak detay belirleme ve çizimsırasında karşılaşılan en büyük sorun, bina çatılarının homojen olmayıp, bina çatılarında farklı yapılaşmaların olması ve resim ölçeğinin büyük olmasından dolayı yolların ağaç, otomobil gibi engeller tarafından kapatılmasıdır.

Yarı otomatik olarak çıkarılan detaylar, uzman bir fotogrametri operatörü tarafından yaklaşık yarım gün içinde tekrar kıymetlendirilmiştir (Şekil 7.14). Her iki veri toplama işlemide yaklaşık aynı süreler içerisinde gerçekleştirilebilmiştir. Doğruluk araştırması amacıyla, uzman operatör tarafından gerçekleştirilen kıymetlendirme sonuçları referans olarak alınmış ve yarı otomatik yöntemile elde edilen vektör veriler karşılaştırılmıştır (Şekil 7.15). Bu karşılaştırma bina ve yol detayı olmak üzere iki ayrı grupta yapılmıştır.



Şekil 7.14 Uzman operatör tarafından kıymetlendirilen vektör veriler.

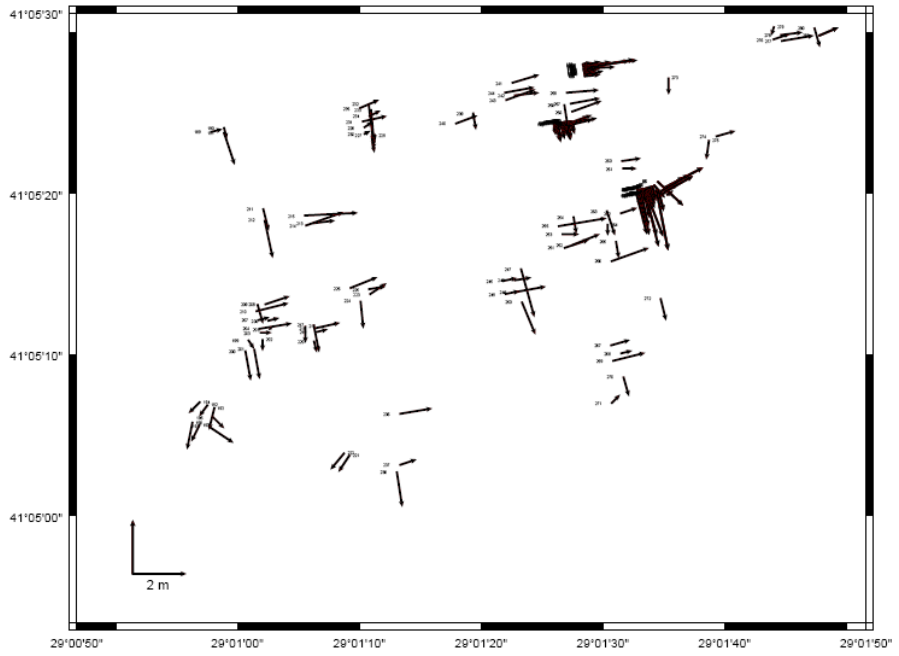


Şekil 7.15 Yarı otomatik olarak yazılımla elde edilen vektör veriler (kırmızı renkli) ile uzman operatör tarafından çizilmiş vektör veriler (cyan renkli).

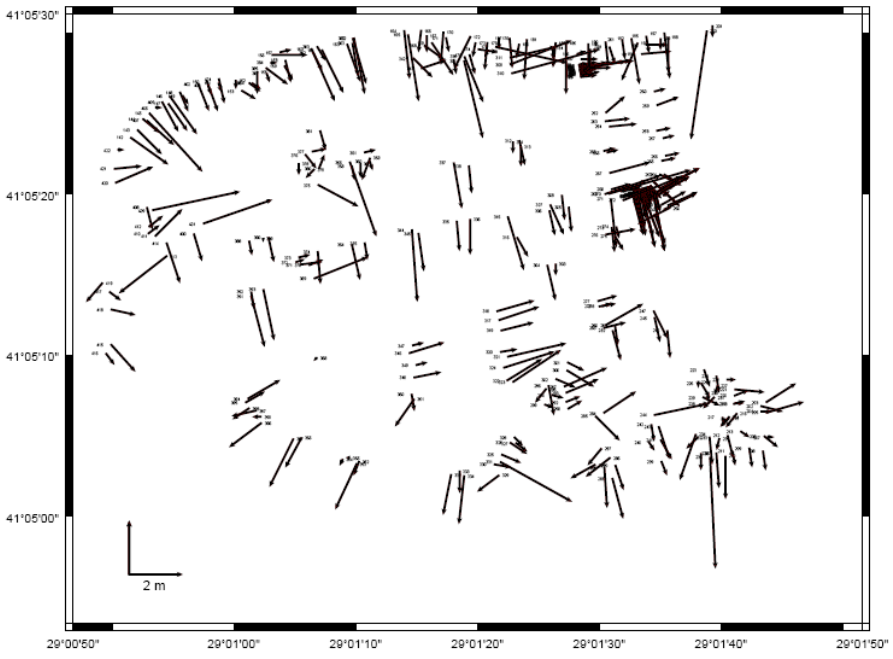
Doğruluk araştırması amacıyla, her iki vektör veri ArcGIS programında üst üste açılmış ve detaylar üzerinde belirlenen noktalarda hızları gösteren dik çizgiler çizilmiş ve bu çizgilerin uzunlukları ölçülmüştür. Aslında bu çizgiler hata vektörlerini temsil etmektedirler. Yol detaylarına ilişkin 422 noktada, bina detaylarına ilişkin 281 noktada ölçüm yapılmıştır. Yapılan ölçümlere ve çizilen hata vektörlerine ilişkin örnekler Şekil 7.16' de gösterilmiştir. Burada kırmızı çizgiler yarı otomatik yöntemle elde edilen vektör verileri, sarı çizgiler ise uzman operatör tarafından kıymetlendirilen vektör verileri göstermektedir. Hata vektörleride sarı ve dik çizgilerle gösterilmiştir.

Şekil 7.16 Gerçekleştirilen hata ölçümleri.

Hata vektörlerinin boylarının ölçülmesi sonucunda elde edilen hata miktarlarının her iki grup (yol ve bina) için ayrı ayrı ortalama ve karesel ortalama hataları hesaplanmış ve binalara ait hataların vektör hızları Şekil 7.17'de, yollara ait vektör hızlar ise Şekil 7.18'de ayrı ayrı gösterilmiştir. Hata analizi sonucunda elde edilen ortalama ve karesel ortalama hata sonuçları Tablo 7.2'de sunulmuştur.



Şekil 7.17 Binalara ait hataların vektör hızları.



Şekil 7.18 Yollara ait hataların vektör hızları.

Elde edilen sonuçlar ışığında, binalarda yollara göre daha iyi sonuçlar elde edildiği görülmüştür. Yollarda, yol kenarlarındaki ağaçlar ve gölgelerin, resim ölçeğinin büyük olmasından dolayı, hataların artmasına neden olduğu görülmüştür. Binaların kenarlarının daha düzgün olması nedeniyle daha düzgün vektörler elde edilmiştir ve bunun sonucunda hataların daha küçük olduğu gözlenmiştir.

Tablo 7.2. Hesaplanan ortalama ve karesel ortalama hata miktarları.

	ORTALAMA HATA	**KARESEL ORTALAMA HATA**	**MİNİMUM – MAKSİMUM HATA**
YOLLAR (422 Nokta)	0.930 m	±0.663 m	0.099 m - 4.183 m
BİNALAR (281 Nokta)	0.710 m	±0.463 m	0.031 m - 2.292 m

Karesel ortalama hata miktarlarına baktığımızda yaklaşık ±0.5m civarında olduğu görülmektedir. Bu da kullanılan orto görüntülerin mekansal ayırma gücü değerlerine denk düşmektedir. Başka bir deyişle bu yöntemin karesel ortalama hatasının, kullanılan raster görüntünün bir pikselinin boyutuna eşit olduğu söylenebilir. Buradan şu sonuca varılabilir: Geliştirilen yöntemin hata kriteri kullanılan dijital hava fotoğrafının ±1 pikselinin boyutuyla sınırlıdır.

Bu test araştırması sonucunda; geliştirilen yarı otomatik yöntemle elde edilen vektör veriler bir operatör tarafından editlenerek düzeltilirse hata miktarlarının daha da düşürülebileceği fakat editlemeyle birlikte yeni bir iş gücü ve ilave zaman gerekeceğinden maliyetlerin de artacağı söylenebilir.

8. SONUÇLAR VE ÖNERİLER

Bu çalışmanın amaç ve hedeflerine paralel olarak, renk farkları ile düzey kümesi yöntemlerini baz alarak geliştirilen yarı otomatik görüntü bölümlemesi ve buna bağlı olarak çalışan rasterdan vektöre dönüştürme yazılımlarının uygulanabilirliğinin, işlevselliğinin ve etkinliğinin test edilmesi amacıyla farklı ölçek ve türlerdeki hava fotoğrafları ile uydu görüntüleri üzerinde yarı otomatik vektör toplama işlemleri gerçekleştirilmiştir ve geliştirilen yöntemin hata kriterlerinin belirlenebilmesi için bir doğruluk araştırması gerçekleştirilmiştir.

Yapılan doğruluk araştırması sonucunda, geliştirilen yöntemin, kullanılan dijital hava fotoğrafının, uydu görüntüsünün veya raster görüntünün ±1 pikselinin boyutuna eşit olan bir hata kriterine sahip olduğu bulunmuştur.

Bununla birlikte; bu yöntemin fotogrametrik harita üretiminde ve CBS için fotogrametrik veri sağlanmasında yeni bir yöntem olarak kullanılabileceği düşünülmektedir. Yapılan test çalışmaları sonucunda bu yöntemle veri toplama işlemlerinde diğer yöntemlere göre bazı avantajlar elde edilebileceği görülmüştür:

- Bu yöntem, özellikle göller, sulu dereler ve binalar gibi homojen yapıdaki detayların sınırlarına ait vektör verilerin toplanmasında çok başarılı ve etkili bir şekilde kullanılabilecektir.
- İstenildiği takdirde, tolerans değerinin uygun olarak belirlenmesi suretiyle, söz konusu detaylar üzerinde gözle ayırt edilemeyen sınıflandırmalar ve bölümlemeler gerçekleştirilebilecektir.
- Kaliteli yolların sınırları ve/veya merkez hatları (kullanılan fotoğrafın ölçeğine bağlı olarak) etkili ve hızlı bir şekilde çıkartılabilecektir.
- Kırıklık toleransı değerleri ayarlanarak istenilen kırıklıkta vektör veriler elde edilebilecektir.
- Rasterdan vektöre dönüşümde hem sınırların hem de merkez hatların kullanılabilmesi etkinliğe çok katkı sağlayacaktır.

Bu avantajlarla birlikte programın hiçbir kütüphane veya paket program dosyası kullanmaması ve başka hiçbir yazılıma bağlı olmaması, yazılımın pratikliğini ve kullanışlılığını arttırdığı düşünülmektedir.

Test çalışmaları sırasında yazılımın olumsuz olarak etkilendiği ve eksik kaldığı bazı faktörler de tespit edilmiş olup, bunlar da aşağıda sunulmuştur:

- Özellikle büyük ölçekli görüntülerde detaylar üzerinde bulunan engeller (ağaç, araba, gölge gibi) detay belirleme ve çizimleşmeyi olumsuz olarak etkilemektedir. Bu etki ölçek küçüldükçe azalmaktadır.

- Tolerans değerleri uygun olarak belirlenmediği takdirde yanlış detay çizimleri gerçekleşebilmektedir. Tolerans yüksek tutulduğunda ilgilenilen detaylar atlanabilmekte, küçük tutulduğunda ise çok fazla data vererek küçük küçük adımlarla ilerleme sağlandığından yazılımın etkinliği düşürülmektedir.

- Çok büyük boyutlu görüntü dosyaları kullanıldığında, tasarlanan algoritmada piksellerin değerleri bilgisayar hafızasına kayıt edildiğinden, çok fazla hafızaya ihtiyaç duyulacağından bazı donanımsal hatalarla karşılaşılabilmektedir.

- Görüntülerin kalitesi, kontrastlığı ve gürültü oranları algoritmanın başarısını önemli ölçüde etkilemektedir.

- Çizgisel detayların yüzey ve kaplama özellikleri ile kontrastlıkları da algoritmanın başarısını etkilemektedir.

Söz konusu problemlerin giderilmesi ve yazılımın daha etkin bir hale getirilebilmesi için; görüntülerdeki kontrastlığın arttırılması, gürültü oranlarının azaltılması için görüntü işleme algoritmalarından faydalanılarak anisotropik difüzyon gibi filtreler ile kenar zenginleştirme algoritmalarının uygulanmasının, engellerden kaynaklanan boşlukların doldurulması için farklı enterpolasyon yöntemlerinin kullanılmasının, bölümleme işlemlerinde renk farkı yerine daha gelişmiş, AKM gibi, algoritmaların kullanılmasının ve büyük boyutlu görüntülerin bilgisayar ortamlarında daha kolay elde alınabilmesi için piramit seviyelerinin kullanılmasının faydalı olacağı değerlendirilmektedir.

Son olarak; harita üretiminde ve CBS için fotogrametrik olarak veri toplama işlemlerinde, tamotomatik yöntemlerin başarısının uzak görünmesinden dolayı, yarı

otomatik çözümlere daha fazla ağırlık verilmesiyle ve bu konulardaki çalışmaların motive edilmesiyle, operatörün zekası ve değerlendirme yeteneğiyle birlikte bilgisayarların hesaplama hızı birleştirilerek önemli ilerlemelerin katedilebileceği değerlendirilmektedir.

KAYNAKLAR

[1] **Bajcsy, R, Tavakoli, M.** 1976. Computer Recognition of Roads from Satellite Pictures, IEEE *Transactions on Systems, Man, and Cybernetics*, **6(9)**, 623-637.

[2] **Baumgartner, A, Steger, C, Mayer, H, Eckstein, W** 1977. Multi-Resolution, Semantic Objects, and Context for Road Extraction, *Akzeptiert für: Semantic Modelling and Topographic Information*.

[3] **Fischler, M, Tenenbaum, J., Wolf, H** 1981. Detection of Roads and Linear Structures in Low Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique, *Computer Graphics and Image Processing*, **15**, 201-223.

[4] **Fua, P., Leclerc, Y.,** 1990. Model Driven Edge Detection, *Machine Vision and Applications*, **3**, 45-56.

[5] **Grün, A, Li, H,** 1995. Road Extraction from Aerial and Satellite Images by Dynamic Programming, *ISPRS Journal of Photogrammetry and Remote Sensing*, **50(4)**, 11-20.

[6] **Grün, A, Li, H,** 1996. Linear Feature Extraction with LSB-Snakes from Multiple Images, *International Archives of Photogrammetry and Remote Sensing*, **31**, 266-272.

[7] **Kass, M, Witkin, A, Terzopoulos, D,** 1987. Snakes: Active Contour Models, *International Journal of Computer Vision*, **1(4)**, 321-331.

[8] **Mayer, H, Steger, C,** 1996. A New Approach for Line Extraction and its Integration in a Multi-Scale, Multi-Abstraction-Level Road Extraction System, *IAPR TC-7 Workshop: Mapping Buildings, Roads and other Man-Made Structures from Images*, Oldenbourg Verlag, Wien, Österreich, 331-348.

[9] **McKeown, D, Denlinger, J.,** 1988. Cooperative Methods For Road Tracking In Aerial Imagery, *Computer Vision and Pattern Recognition*, 662-672.

[10] **Merlet, N, Zerubia, J.,** 1996. New Prospects in Line Detection by Dynamic Programming, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18(4)**, 570-585.

[11] **Neuenschwander, W,** 1996. Elastic Deformable Contour and Surface Models for 2-D and 3-D Image Segmentation, Hartung-Gorre Verlag, Konstanz.

[12] **Nevatia, R, Babu, K,** 1980. Linear Feature Extraction and Description, *Computer Graphics and Image Processing*, **13**, 257-269.

[13] **Steger, C,** 1996. Extracting Curvilinear Structures: A Differential Geometric Approach, *Fourth European Conference on Computer Vision*, Band I, Cambridge, UK, April 14-18, 630-641.

[14] **Trinder, J., Li, H,** 1996. Extraction of Man-Made Features by 3-D Active Contour Models, *International Archives of Photogrammetry and Remote Sensing*, **31**, 874-879.

[15] **Quam, Lynn H,** 1978. Road Tracking and Anomaly Detection in Aerial Imagery, *Image Understanding Workshop*, Cambridge, UK, May 9-11, 51-55.

[16] **Courant R and Hilbert D,** 1989. Methods of Mathematical Physics, Wiley Press, NewYork.

[17] **Laptev, I.,** 1997. Road extraction based on line extraction and snakes, *Master Thesis*, KTH, Stockholm, Sweden.

[18] **Wismüller, A, Vietze F., Dersch, D R,** 2000. Segmentation with Neural Networks, *Handbook of Medical Imaging*, Bankman, I. N (ed), 107-126, Academic Press, San Diego.

[19] **Maraş, H,** 2005. PC tabanlı üç boyutlu tıbbi görüntü işleme uygulaması, *Yüksek Lisans Tezi,* S.Ü Fen Bilimleri Enstitüsü, Konya.

[20] **Liang, Z,** 1993. Tissue Classification and Segmentation of MR Images, *IEEE in Medicine and Biology*, **12**, 81-85.

[21] **Bezdek, J. C., Hall, L. O., Clarke, L. P.,** 1993. Review of MR Image Segmentation Techniques Using Pattern Recognition, *Med. Phys*, **20**, 1033-1048.

[22] **Gonzales, R. C., Woods, R. E.,** 1993. Digital Image Processing, Reading, MA: Addison-Wesley Publishing Company.

[23] **Clarke, L. P., Velthuizen, R. P., Camacho, M. A., Heine, J. J., Vaidyanathan, M., Hall, L. O., Thatcher, R. W., Silbiger, M. L.,** 1995. MRI Segmentation: Methods and Applications, *Mag. Res. Imag.*, **13**, 334-368.

[24] **Castleman, K. R.,** 1996. Digital Image Processing, Upper Saddle River: Prentice Hall.

[25] **Sonka, M., Hlavac, V., Boyle, R.,** 1999. Image Processing, Analysis and Machine Vision. CA: PWS Publishing, Pacific Grove.

[26] **Suetens, P. Bellon, E., Vandermeulen, D., Smet, M., Marchal, G., Nuyts, J., Mortelman, L.,** 1993. Image Segmentation: Methods and Applications in Diagnostic Radiology and Nuclear Medicine, *European Journal of Radiology*, **17**, 14-21.

[27] **Goldszal, A. F., Pham, D. L.,** 2000. Volumetric Segmentation, *Handbook of Medical Imaging*, Bankman, I. N(ed), 185-194, Academic Press, San Diego.

[28] **Shareef, N., Wang, D. L., Yagel, R.,** 1999. Segmentation of Medical Images Using LEGION, *IEEE Trans. Med. Imag.*, **18**, 74-91.

[29] **Awcock, G. J., Thomas, R.,** 1996. Applied Image Processing, New York: Mc Graw-Hill, Inc.

[30] **Rajapakse, J. C., Giedd, J. N., Rapoport, J. L.,** 1997. Statistical Approach to Segmentation of Single-Channel Cerebral MR Images, *IEEE Trans. Med. Imag.*, **16**, 176-186.

[31] **Höhne, K. H., Bernstein, R.,** 1986. Shading 3D-Images from CT Using Gray-Level Gradients. *IEEE Transactions on Medical Imaging*, **M- 5(1)**, 45-47.

[32] **Sobel, I.,** 1996. An Isotropic 3x3x3 Volume Gradient Operator, Hewlett-Packard's Voxelator V-3 CDROM.

[33] **Sethian, J. A,** 1997. Level Set Methods: An Act of Violence, *American Scientist*, **85(3)**, 12-35.

[34] **Sethian, J. A,** 1998. Fast Marching Methods and Level Set Methods for Propogating Interfaces, *von Karman Institute Lecture Series*, Computational Fluid Mechanics, Belgium.

[35] **Adalsteinsson, D, and Sethian, J. A,** 1995. Fast Level Set Method for Propagating Interfaces, *Journal of Comp. Phys.*, **118**, 269-277.

[36] **Chopp, D L,** 1993. Computing Minimal Surfaces via Level Set Curvature Flow, *Journal of Comp. Phys.*, **106**, 77-91.

[37] **Crandall, M G, and Lions, P-L,** 1983. Viscosity Solutions of Hamilton Jacobi Equations, *Tran. AMS*, **277**, 1-43.

[38] **Malladi, R, Sethian, J A, and Vemuri, B C,** 1994. Evolutionary Fronts for Topology-Independent Shape Modeling and Recovery, *Proceedings of Third European Conference on Computer Vision, Stockholm, Sweden, Lecture Notes in Computer Science*, **800**, 3-13.

[39] **Osher, S., and Sethian, J. A,** 1988. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulation, *Journal of Comp. Phys.*, **79**, 12-49.

[40] **Rouy, E and Tourin, A,** 1992. A Viscosity Solutions Approach to Shape-From-Shading, *SIAM J. Num Anal*, **29**, 867-884.

[41] **Sethian, J. A,** 1996, Applications of Level Set Methods for Propogating Interfaces, Acta Numerica.

[42] **Sethian, J. A,** 1982. An analysis of flame propagation, *Ph. D Dissertation*, Dept. of Mathematics, University of California, Berkeley, CA.

[43] **Sethian, J. A,** 1985. Curvature and the Evolution of Fronts, *Comm Math. Phys.*, **101**, 487-499.

[44] **Sethian, J. A,** 1987. Numerical Methods for Propagating Fronts, in *Variational Methods for Free Surface Interfaces*, Eds. P. Concus and R. Finn, Springer-Verlag, NY.

[45] **Sethian, J. A,** 1996. Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, *Computer Vision and Material Science*, Cambridge University Press.

[46] **Maraş, H,** 1998. Coğrafi veri tabanı güncelleştirmesine yönelik coğrafi bilgi sistemi tasarımı ve uygulaması, *Doktora Tezi*, İ.T.Ü Fen Bilimleri Enstitüsü, İstanbul.

[47] **ESRI,** 1997. ARC/INFO User's Guide Cell-Based Modelling With GRID, USA.

[48] **Şehsuvaroğlu, S.,** 2001. Coğrafi bilgi sistemlerinde etkileşimli olarak renkli raster görüntüden vektör veriye dönüşümi için bir arayüz tasarımı ve uygulaması, *Yüksek Lisans Tezi*, İ.T.Ü Fen Bilimleri Enstitüsü, İstanbul.

[49] **Bank, E,** 1994. Coğrafi Bilgi Sistemleri Ders Notları, Ankara.

[50] **Sarbanoğlu, H,** 1991. Coğrafi Bilgi Sistemleri için Veri Toplama Yöntemleri, (1'nci Bölüm), *Harita Dergisi*, Ankara, **106**, 40-65.

[51] **Özbalmumcu, M,** 1999. Coğrafi bilgi sistemlerinin oluşturulmasında kullanılan veri kaynakları, veri toplama sistemleri ve konumsal veri toplama yöntemlerinin araştırılması, *Doktara Tezi*, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[52] **Taştan, H,** 1991. Coğrafi bilgi sistemleri, bir coğrafi bilgi sisteminin (Akbis) tasarımı ve gerçekleştirilmesi, *Yüksek Lisans Tezi*, İ.T.Ü Fen Bilimleri Enstitüsü, İstanbul.

[53] **Sarbanoğlu, H,** 1991. Coğrafi Bilgi Sistemleri için Veri Toplama Yöntemleri, (2'nci Bölüm), *Harita Dergisi*, Ankara, **107**, 51-81.

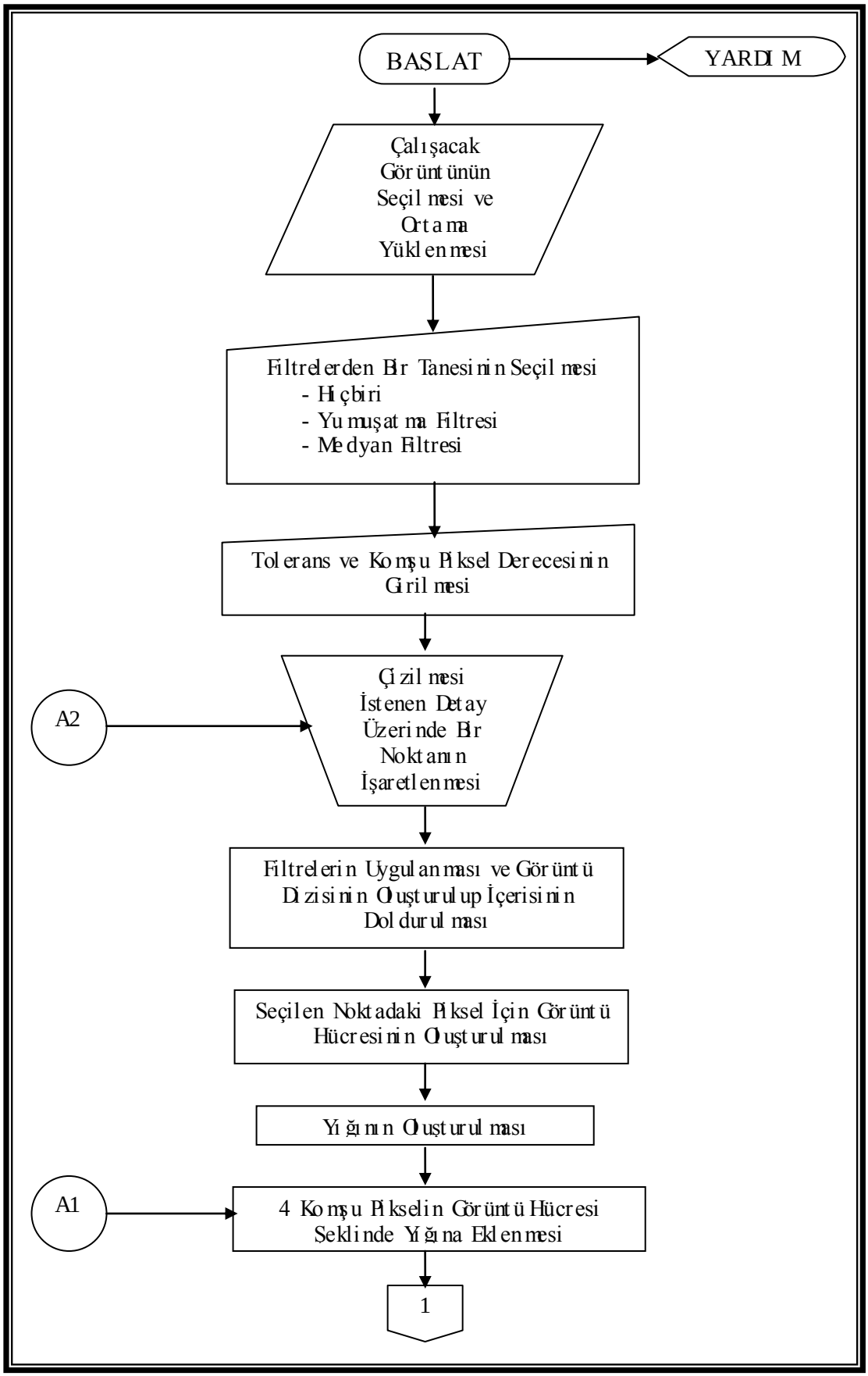
[54] **Karagülle, İ. ve Pala, Z.,** 2002., Borland C++ Builder 6, Türkmen Kitabevi, İstanbul.

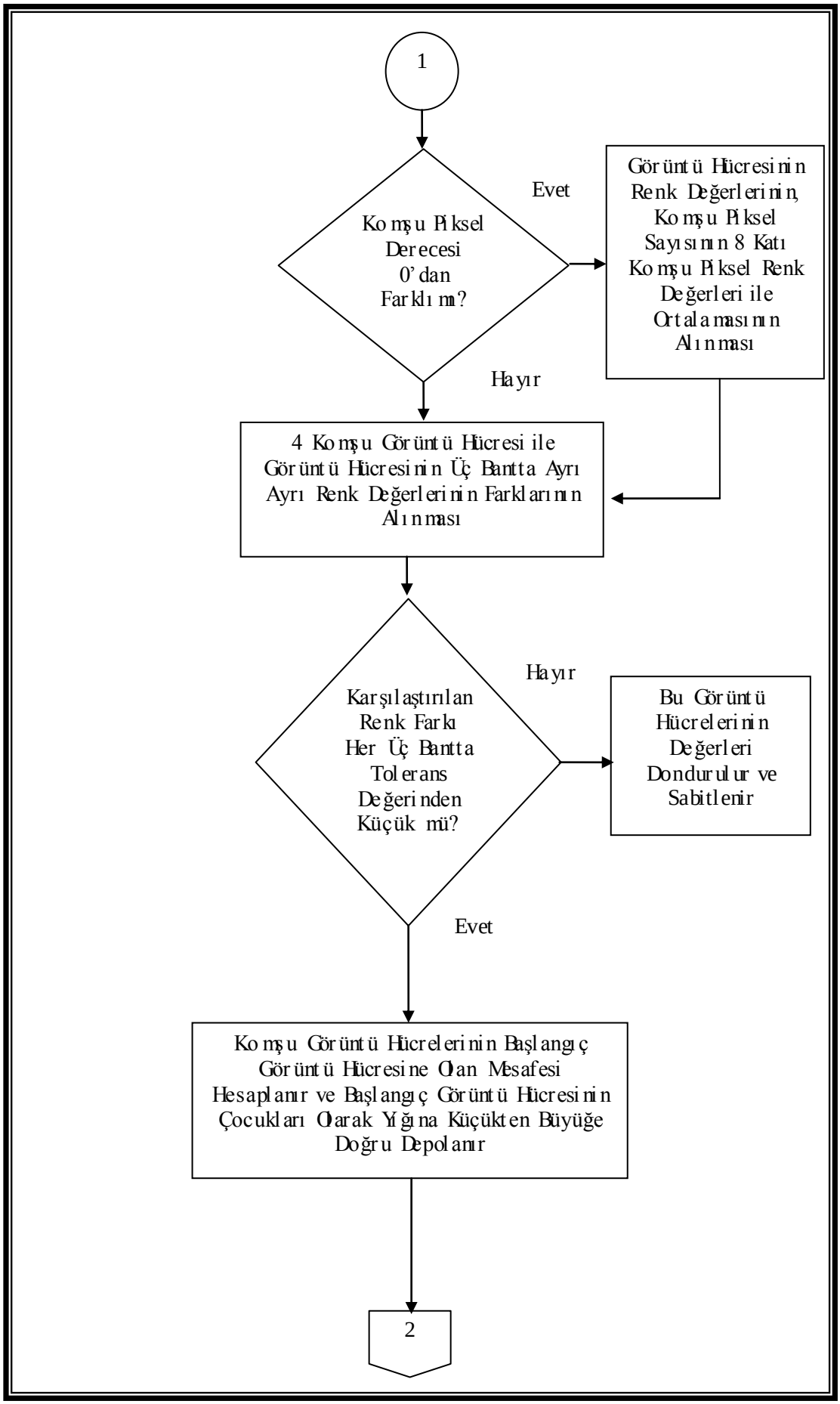
[55] //xmailserver.org/davide.html.

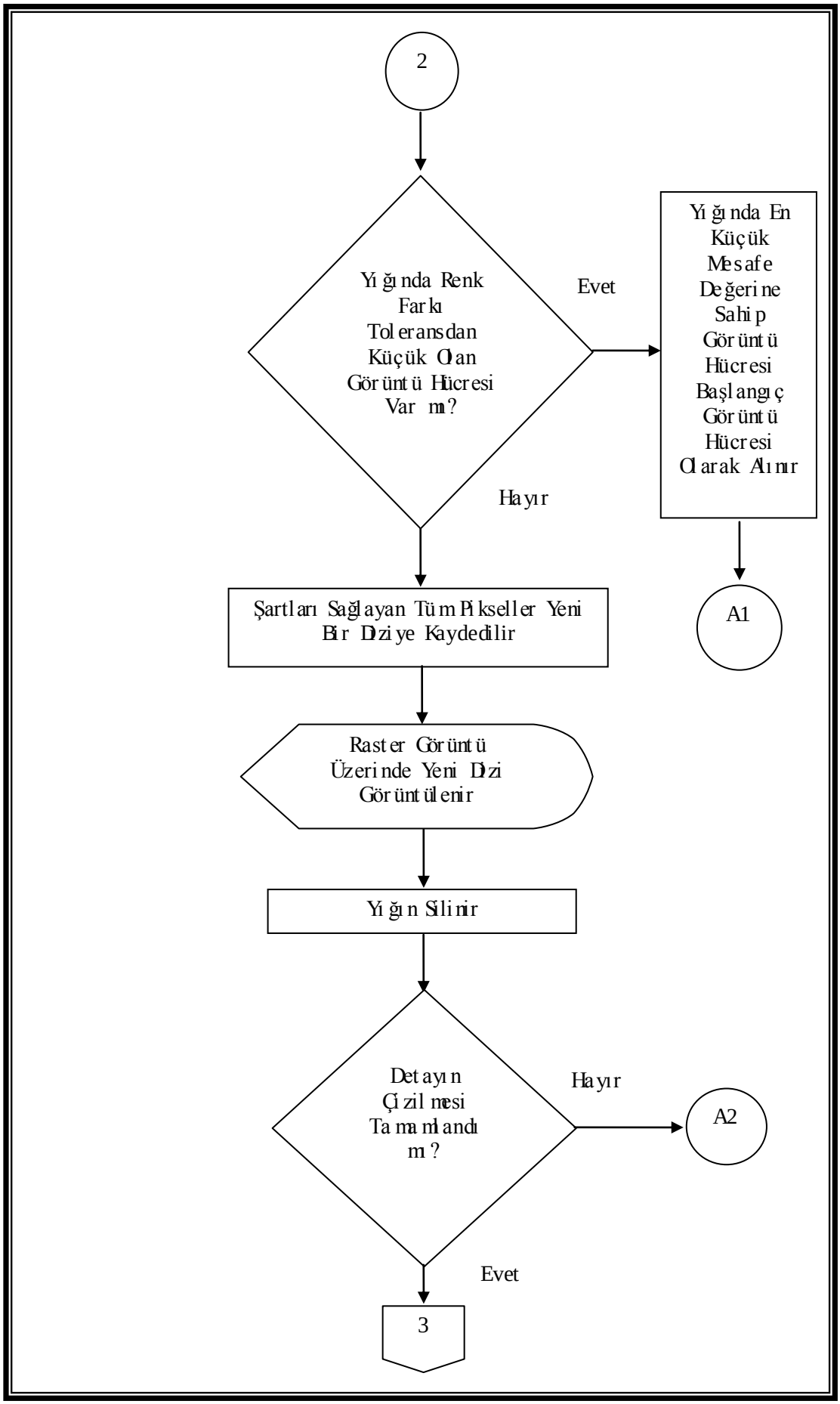
[56] //math.berkeley.edu/~sethian.

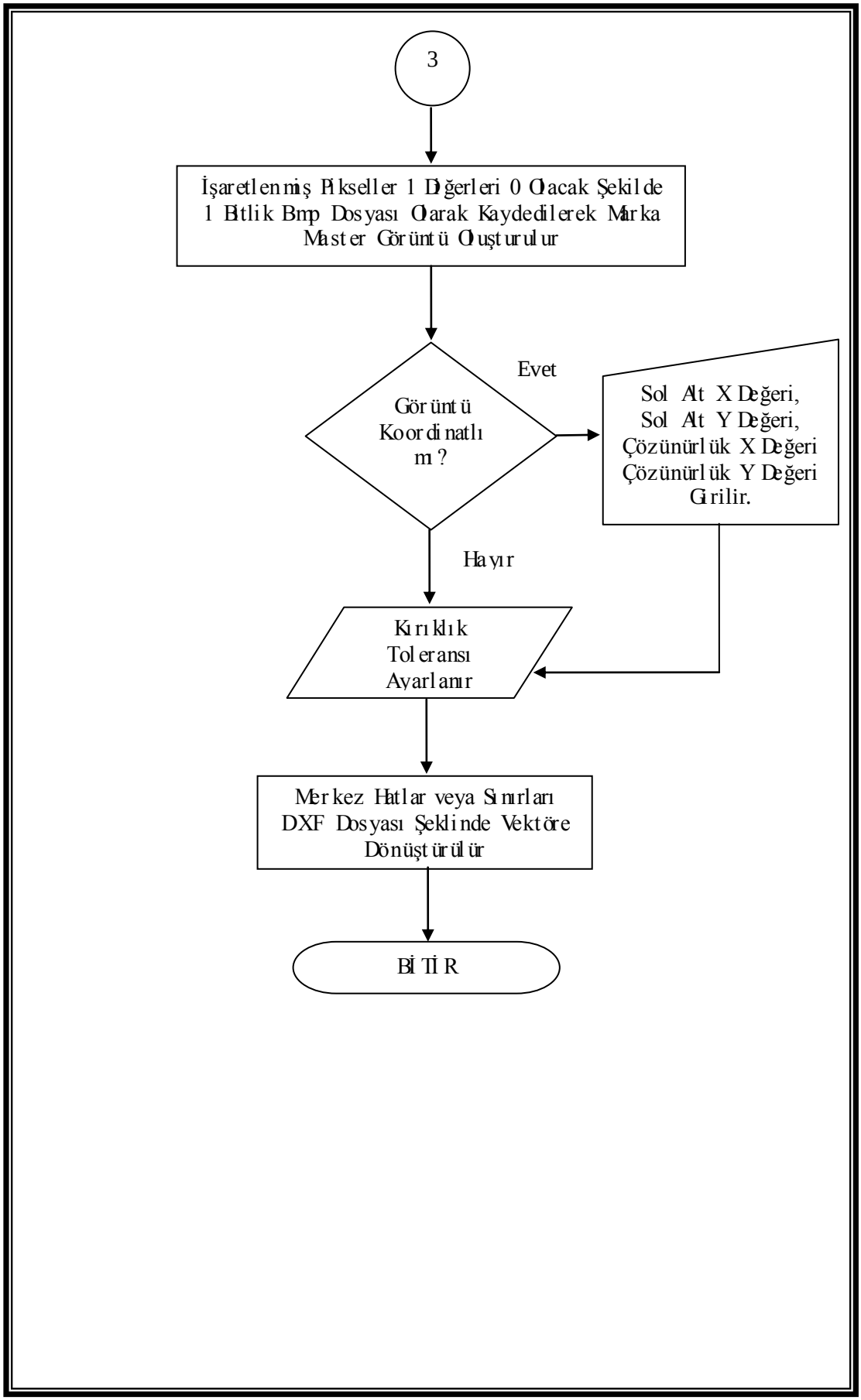
EKLER

EK-A GELİŞTİRİLEN PROGRAMIN AKIŞ DİYAGRAMI




1
Komşu Piksel Derecesi 0'dan Farklı mı?
Evet
Görüntü Hücresinin Renk Değerlerinin, Komşu Piksel Sayısının 8 Katı Komşu Piksel Renk Değerleri ile Ortalamasının Alınması
Hayır
4 Komşu Görüntü Hücresi ile Görüntü Hücresinin Üç Bantta Ayrı Ayrı Renk Değerlerinin Farklarının Alınması
Karşılaştırılan Renk Farkı Her Üç Bantta Tolerans Değerinden Küçük mü?
Hayır
Bu Görüntü Hücrelerinin Değerleri Dondurulur ve Sabitlenir
Evet
Komşu Görüntü Hücrelerinin Başlangıç Görüntü Hücresine Olan Mesafesi Hesaplanır ve Başlangıç Görüntü Hücresinin Çocukları Olarak Yığına Küçükten Büyüğe Doğru Depolanır
2


2
Yığında Renk Farkı Toleranstan Küçük Olan Görüntü Hücresi Var mı?
Evet
Yığında En Küçük Mesafe Değerine Sahip Görüntü Hücresi Başlangıç Görüntü Hücresi Olarak Alınır
Hayır
A1
Şartları Sağlayan Tüm Pikseller Yeni Bir Diziye Kaydedilir
Raster Görüntü Üzerinde Yeni Dizi Görüntülenir
Yığın Silinir
Detayın Çizilmesi Tamamlandı mı?
Hayır
A2
Evet
3


3
İşaretlenmiş Pikseller 1 Diğerleri 0 Olacak Şekilde 1 Bitlik Bmp Dosyası Olarak Kaydedilerek Marka Master Görüntü Oluşturulur
Görüntü Koordinatlı mı ?
Evet
Sol Alt X Değeri, Sol Alt Y Değeri, Çözünürlük X Değeri Çözünürlük Y Değeri Girilir.
Hayır
Kırıklık Toleransı Ayarlanır
Merkez Hatlar veya Sınırları DXF Dosyası Şeklinde Vektöre Dönüştürülür
BİTİR

EK-B ANA PROGRAM

```
void __fastcall TDuzeyKumesiTestForm: GoruntuDblClick(TObject *Sender)
{
  genislik = Goruntu->Picture->Width;
  yukseklik = Goruntu->Picture->Height;
  tolerans = ToleransDegeri->Value;
  Komsu = KomsulukDegeri->Value;
  EsikDegeri = 2.0 * genislik * yukseklik;
  TCursor Save_Cursor = Screen->Cursor;
  Screen->Cursor = crHourGlass;
  if (goruntu_dizisi == NULL)
  {
  if (sbitmap == NULL)
   {
    sbitmap = new Graphics::TBitmap();
    sbitmap->PixelFormat = pf24bit;
    sbitmap->Width = genislik;
    sbitmap->Height = yukseklik;
    sbitmap->Canvas->Draw(0, 0, Goruntu->Picture->Graphic);
   }
  if (maske == NULL)
  {
    maske = new unsigned char *[yukseklik];
   for (int y=0; y<yukseklik; y++)
   {
     maske[y] = new unsigned char[genislik];
     memset(maske[y], 0, genislik);
   }
  }
  Goruntu->Picture->Bitmap->PixelFormat = pf24bit;
  Goruntu->Picture->Bitmap->Width = genislik;
  Goruntu->Picture->Bitmap->Height = yukseklik;
  Goruntu->Picture->Bitmap->Canvas->Draw(0, 0, sbitmap);

/***** goruntu_dizisinin yaratılması **/

  goruntu_dizisi = new GoruntuHucresi**[yukseklik];
  for (int y=0; y<yukseklik; y++)
    goruntu_dizisi[y] = new GoruntuHucresi*[genislik];

/***** goruntu_dizisinin yaratılması son**/

/**** goruntu_dizisinin içerisinin doldurulması ***/

  switch (cmbFiltreYontemi->ItemIndex)
  {
   case 0: // Hiçbiri
   {
    for (int y=0; y<yukseklik; y++)
    {
      unsigned char *sptr = (unsigned char *)sbitmap->ScanLine[y];
      for (int x=0; x<genislik; x++)
      {
        goruntu_dizisi[y][x] = new GoruntuHucresi(x, y, *(sptr++), *(sptr++), *(sptr++));
```

```
          Goruntu->Canvas->Pixels[x][y] = TColor(RGB(goruntu_dizisi[y][x]->kirmizi,
                                        goruntu_dizisi[y][x]->yesil,
                                        goruntu_dizisi[y][x]->mavi));
        }
       }
       break;
      }
      case 1: // Yumuşatma
      {
       for(int y=0; y<yukseklik; y++)
       {
        unsigned char *sptru = (y > 0 ? (unsigned char *)sbit map->ScanLine[y-1] : NULL);
        unsigned char *sptr = (unsigned char *)sbit map->ScanLine[y];
        unsigned char *sptrd = (y < yukseklik - 1 ? (unsigned char *)sbit map->ScanLine[y+1] : NULL);
        for(int x=0; x<genislik; x++)
        {
         int kirmizil = (x>0 ? *(sptr - 3) : 0);
         int yesill = (x>0 ? *(sptr - 2) : 0);
         int mavil = (x>0 ? *(sptr - 1) : 0);
         int kirmizir = (x<genislik-1 ? *(sptr + 3) : 0);
         int yesilr = (x<genislik-1 ? *(sptr + 4) : 0);
         int mavir = (x<genislik-1 ? *(sptr + 5) : 0);
         int kirmiziu = (y>0 ? *(sptru + 0) : 0);
         int yesilu = (y>0 ? *(sptru + 1) : 0);
         int maviu = (y>0 ? *(sptru + 2) : 0);
         int kirmizid = (y<yukseklik-1 ? *(sptrd + 0) : 0);
         int yesild = (y<yukseklik-1 ? *(sptrd + 1) : 0);
         int mavid = (y<yukseklik-1 ? *(sptrd + 2) : 0);
         goruntu_dizisi[y][x] = new GoruntuHucresi(x, y,
                              (kirmizil + kirmizir + kirmiziu + kirmizid)/4,
                              (yesill + yesilr + yesilu + yesild)/4,
                              (mavil + mavir + maviu + mavid)/4);
         Goruntu->Canvas->Pixels[x][y] = TColor(RGB(goruntu_dizisi[y][x]->mavi,
                                        goruntu_dizisi[y][x]->yesil,
                                        goruntu_dizisi[y][x]->kirmizi));
         sptru += 3;
         sptr += 3;
         sptrd += 3;
        }
       }
       break;
      }
      case 2: // Median Filtresi
      {
       int rgb3x3[3][3][3];
       for(int y=0; y<yukseklik; y++)
       {
        unsigned char *sptru = (y > 0 ? (unsigned char *)sbit map->ScanLine[y-1] : NULL);
        unsigned char *sptr = (unsigned char *)sbit map->ScanLine[y];
        unsigned char *sptrd = (y < yukseklik - 1 ? (unsigned char *)sbit map->ScanLine[y+1] : NULL);
        for(int x=0; x<genislik; x++)
        {
```

```
          for (int xi=x-1,i=0; i<3; i++, xi++)
          {
           if (xi>=0  && xi<genislik)
           {
            if (y > 0)
            {
             rgb3x3[0][i][0] = *(sptru + i * 3 - 3);
             rgb3x3[0][i][1] = *(sptru + i * 3 - 2);
             rgb3x3[0][i][2] = *(sptru + i * 3 - 1);
            }
             rgb3x3[1][i][0] = *(sptr + i * 3 - 3);
             rgb3x3[1][i][1] = *(sptr + i * 3 - 2);
             rgb3x3[1][i][2] = *(sptr + i * 3 - 1);
            if (y < yukseklik - 1)
            {
             rgb3x3[2][i][0] = *(sptrd + i * 3 - 3);
             rgb3x3[2][i][1] = *(sptrd + i * 3 - 2);
             rgb3x3[2][i][2] = *(sptrd + i * 3 - 1);
            }
           }
           else
           {
            for (int j=0; j<3; j++)
            {
             rgb3x3[j][i][0] = 0;
             rgb3x3[j][i][1] = 0;
             rgb3x3[j][i][2] = 0;
            }
           }
          }
          for (int i=0; i<3; i++)
          {
           for (int j=0; j<3; j++)
           {
            if (rgb3x3[i][0][j]>rgb3x3[i][1][j]) swap(rgb3x3[i][0][j], rgb3x3[i][1][j]);
            if (rgb3x3[i][1][j]>rgb3x3[i][2][j]) swap(rgb3x3[i][1][j], rgb3x3[i][2][j]);
           }
          }
          goruntu_dizisi[y][x] = new GoruntuHucresi(x, y, rgb3x3[1][1][0],
                                      rgb3x3[1][1][1],
                                      rgb3x3[1][1][2]);
          Goruntu->Canvas->Pixels[x][y] = TColor(RGB(goruntu_dizisi[y][x]->mavi,
                                        goruntu_dizisi[y][x]->yesil,
                                        goruntu_dizisi[y][x]->kirmizi));
          sptru += 3;
          sptr += 3;
          sptrd += 3;
         }
        }
        break;
       }
       default : return;
      }
     }
/***** goruntu_dizisinin içerisinin doldurulması son***/
```

```
/**** Düzey Kümesi ile Bölümleme ***/

  HucreYenile();
  Yayilma(x_konumu, y_konumu);

/**** Düzey Kümesi ile Bölümleme Son***/

/**** Düzey Kümesi ile Bölümleme Sonucunda Piksellerin İşaretlenmesi ***/
 for (int yy=1; yy<3; yy++)
 {
 for (int y=0; y<yukseklik; y++)
 {
   for (int x=0; x<genislik; x++)
   {
     double d = goruntu_dizisi[y][x]->mesafe - EsikDegeri;
     if(d != 0.0)
        goruntu_dizisi[y][x]->mesafe = d;
     else
        goruntu_dizisi[y][x]->mesafe = 1.0;
     int deger =((int)(fabs(goruntu_dizisi[y][x]->mesafe/512.0))) %256;
     if (deger > 1)
     {
       Goruntu->Canvas->Pixels[x][y] = TColor(RGB(deger,0,0));
       maske[y][x] = 255;
     }
   }
 }
}
Screen->Cursor = crDefault;
}
/**** Düzey Kümesi ile Bölümleme Sonucunda Piksellerin İşaretlenmesi Son***/

/**** Düzey Kümesi ile Bölümleme ve Bunları İşaretleme Bitti***/
```

EK-C RENK FARKI ALGORİTMASININ KODLANMASI

```
double TDuzeyKumesiTestForm::RenkFarki(GoruntuHucresi *goruntu_hucresi)
{
  int x = goruntu_hucresi->x;
  int y = goruntu_hucresi->y;
  int kirmizi1 = goruntu_hucresi->kirmizi;
  int yesil1 = goruntu_hucresi->yesil;
  int mavi1 = goruntu_hucresi->mavi;
  int k =1;
  if (Komsu > 0)
   {
   for (int yy=1; yy<=Komsu; yy++)
    for (int xx=1; xx<=Komsu; xx++)
      {
           if(y+yy >= 0 && y+yy < yukseklik && x-xx>=0 && x-xx<genislik)
             {
              kirmizi1 = goruntu_dizisi[y + yy][x-xx]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y + yy][x-xx]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y + yy][x-xx]->yesil + yesil1;
              k = k+1;
             }
           if(x-xx>=0 && x-xx<genislik)
            {
              kirmizi1 = goruntu_dizisi[y][x-xx]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y][x-xx]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y][x-xx]->yesil + yesil1;
              k = k+1;
            }
           if(y-yy >= 0 && y-yy < yukseklik && x-xx>=0 && x-xx<genislik)
            {
              kirmizi1 = goruntu_dizisi[y-yy][x-xx]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y-yy][x-xx]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y-yy][x-xx]->yesil + yesil1;
              k = k+1;
            }
           if(y+yy >= 0 && y+yy < yukseklik)
            {
              kirmizi1 = goruntu_dizisi[y+yy][x]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y+yy][x]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y+yy][x]->yesil + yesil1;
              k = k+1;
            }
           if(y-yy >= 0 && y-yy < yukseklik)
           {
              kirmizi1 = goruntu_dizisi[y-yy][x]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y-yy][x]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y-yy][x]->yesil + yesil1;
              k = k+1;
           }
           if(y+yy >= 0 && y+yy < yukseklik && x+xx>=0 && x+xx<genislik)
           {
              kirmizi1 = goruntu_dizisi[y+yy][x+xx]->kirmizi + kirmizi1;
              mavi1 = goruntu_dizisi[y+yy][x+xx]->mavi + mavi1;
              yesil1 = goruntu_dizisi[y+yy][x+xx]->yesil + yesil1;
```

```
                k = k+1;
            }
            if(x+xx>=0 && x+xx<genislik)
            {
                kirmizi1 = goruntu_dizisi[y][x+xx]->kirmizi + kirmizi1;
                mavi1 = goruntu_dizisi[y][x+xx]->mavi + mavi1;
                yesil1 = goruntu_dizisi[y][x+xx]->yesil + yesil1;
                k = k+1;
            }
            if(y-yy >= 0 && y-yy < yukseklik && x+xx>=0 && x+xx<genislik)
            {
                kirmizi1 = goruntu_dizisi[y-yy][x+xx]->kirmizi + kirmizi1;
                mavi1 = goruntu_dizisi[y-yy][x+xx]->mavi + mavi1;
                yesil1 = goruntu_dizisi[y-yy][x+xx]->yesil + yesil1;
                k = k+1;
            }
        }
    }
    kirmizi1 = kirmizi1/k;
    mavi1 = mavi1/k;
    yesil1 = yesil1/k;
   if(abs(kirmizi1 - kirmizi) <= tolerans && abs(yesil1 - yesil) <= tolerans && abs(mavi1 -
mavi) <= tolerans)
        return 0.0;
    else
        return EsikDegeri;
}
//*************** Renk Farkını Hesaplayan Program Bitti ****************-
```

EK-Ç YAYILMA ALGORİTMASININ KODLANMASI

```
void TDuzeyKumesiTestForm::Yayilma(int i, int j)
{
    Yigin *yigin = new Yigin(genislik + yukseklik);
    GoruntuHucresi *goruntu_hucresi4 = goruntu_dizisi[j][i];
    goruntu_hucresi4->mesafe = 0.0;
    goruntu_hucresi4->isaretlenmis = true;
    yigin->ekle(goruntu_hucresi4);
   int k = 1;
    kirmizi = goruntu_hucresi4->kirmizi;
    yesil = goruntu_hucresi4->yesil;
    mavi = goruntu_hucresi4->mavi;
    GoruntuHucresi *goruntu_hucresi5 = yigin->MinCikar();
    do
    {
      goruntu_hucresi5->isaretlenmis = true;
      i = goruntu_hucresi5->x;
      j = goruntu_hucresi5->y;
      if(i - 1 >= 0)
      {
        GoruntuHucresi *goruntu_hucresi = goruntu_dizisi[j][i - 1];
       if(!goruntu_hucresi->isaretlenmis)
        {
          HucreGuncelle(goruntu_hucresi);
         if(goruntu_hucresi->konumu < 0)
            yigin->ekle(goruntu_hucresi);
         else
            yigin->sabitle(goruntu_hucresi);
        }
      }
      if(i + 1 < genislik)
      {
        GoruntuHucresi *goruntu_hucresi1 = goruntu_dizisi[j][i + 1];
       if(!goruntu_hucresi1->isaretlenmis)
        {
          HucreGuncelle(goruntu_hucresi1);
         if(goruntu_hucresi1->konumu < 0)
            yigin->ekle(goruntu_hucresi1);
         else
            yigin->sabitle(goruntu_hucresi1);
        }
      }
      if(j - 1 >= 0)
      {
        GoruntuHucresi *goruntu_hucresi2 = goruntu_dizisi[j - 1][i];
       if(!goruntu_hucresi2->isaretlenmis)
        {
          HucreGuncelle(goruntu_hucresi2);
         if(goruntu_hucresi2->konumu < 0)
            yigin->ekle(goruntu_hucresi2);
         else
            yigin->sabitle(goruntu_hucresi2);
        }
      }
```

```
        if(j + 1 < yukseklik)
        {
            GoruntuHucresi *goruntu_hucresi3 = goruntu_dizisi[j + 1][i];
            if(!goruntu_hucresi3->isaretlenmis)
            {
                HucreGuncelle(goruntu_hucresi3);
                if(goruntu_hucresi3->konumu < 0)
                    yigin->ekle(goruntu_hucresi3);
                else
                    yigin->sabitle(goruntu_hucresi3);
            }
        }
        goruntu_hucresi5 = yigin->MinCikar();
        if (goruntu_hucresi5 == NULL)
            break;
        if (goruntu_hucresi5->mesafe > EsikDegeri)
            return;
        if(RenkFarki(goruntu_hucresi5) < 1.0)
        {
            int l = k + 1;
            kirmizi = round((float)(kirmizi * k + goruntu_hucresi5->kirmizi) / (float)l);
            yesil = round((float)(yesil * k + goruntu_hucresi5->yesil) / (float)l);
            mavi = round((float)(mavi * k + goruntu_hucresi5->mavi) / (float)l);
            k = l;
        }
    }
    while(true);
    delete yigin;
}
//***************Düzey Kümesi ile Yayilma Algoritmasi Son*************
```

EK-D GÜNCELLEME, YENİLEME ve KARESLLEŞTİRME ALGORİTMALARININ KODLANMASI

```
void TDuzeyKumesiTestForm::HucreGuncelle(GoruntuHucresi *goruntu_hucresi)
{
   int i = goruntu_hucresi->x;
   int j = goruntu_hucresi->y;
    double l = RenkFarki(goruntu_hucresi) + 1;
    double l1 = l * l;
    double d = (1.0 / 0.0);
    double d1 = (1.0 / 0.0);
   if(i - 1 >= 0)
      d = goruntu_dizisi[j][i - 1]->mesafe;
   if(i + 1 < genislik && d > goruntu_dizisi[j][i + 1]->mesafe)
      d = goruntu_dizisi[j][i + 1]->mesafe;
   if(j - 1 >= 0)
      d1 = goruntu_dizisi[j - 1][i]->mesafe;
   if(j + 1 < yukseklik && d1 > goruntu_dizisi[j + 1][i]->mesafe)
      d1 = goruntu_dizisi[j + 1][i]->mesafe;
    double d2;
    double d3;
   if(d <= d1)
   {
      d2 = d;
      d3 = d1;
   }
   else
   {
      d2 = d1;
      d3 = d;
   }
    double d10 = (1.0 / 0.0);
   if(d3 != (1.0 / 0.0))
   {
      double d4 = 2.0;
      double d6 = -2.0 * (d3 + d2);
      double d8 = (d3 * d3 + d2 * d2) - l1;
      d10 = karesellestirme(d4, d6, d8);
   }
   if(d10 == (1.0 / 0.0) || d10 < d3)
   {
      double d5 = 1.0;
      double d7 = -2.0 * d2;
      double d9 = d2 * d2 - l1;
      d10 = karesellestirme(d5, d7, d9);
   }
    double d11 = d2 + l;
   if (d10 > d11)
     goruntu_hucresi->mesafe = d11;
   else goruntu_hucresi->mesafe = d10;
   return;
}
//---------------------------------- Hücrelerin Güncellenmesi Son----------------------------------
```

```
//---------------------------------- Hücrelerin Yenilenmesi----------------------------------
void __fastcall TDuzeyKumesiTestForm::HucreYenile()
{
  for (int y=0; y<yukseklik; y++)
    for (int x=0; x<genislik; x++)
       goruntu_dizisi[y][x]->yenile();
}
//---------------------------------- Hücrelerin Yenilenmesi Son----------------------------------

//---------------------------------- Karesel leştirme----------------------------------
double karesellestirme(double d, double d1, double d2)
{
    double d3 = d1 * d1 - 4.0 * d * d2;
   if(d3 < 0.0)
      return (1.0 / 0.0);
   else
      return (sqrt(d3) - d1) / (2.0 * d);
}
//---------------------------------- Karesel leştirme Son----------------------------------
```

EK-E YIĞIN ALGORİTMASININ ve FONKSİYONLARININ KODLANMASI

```
#include "Yigin.h"
Yigin::Yigin(int i)
{
   indeks = -1;
   yigin = new GoruntuHucresi*[i];
   boyut = i;
}
Yigin::~Yigin()
{
   if (yigin != NULL)
      delete [] yigin;
}
//---------------------------------- Yığına Eleman Ekleme ----------------------------------
void Yigin::ekle( GoruntuHucresi *goruntuhucresi)
{
   indeks++;
   if(indeks == boyut)
   {
      GoruntuHucresi **agoruntuhucresi = new GoruntuHucresi*[2 * boyut];
      for(int i = 0; i < boyut; i++)
         agoruntuhucresi[i] = yigin[i];

      delete [] yigin;
      yigin = agoruntuhucresi;
      boyut *= 2;
   }

   yigin[indeks] = goruntuhucresi;
   goruntuhucresi->konumu = indeks;
   sabitle(goruntuhucresi);
}
//---------------------------------- Yığına Eleman Ekleme Son ----------------------------------
//---------------------------------- Yığında Elemanı Sabitleme ----------------------------------
void Yigin::sabitle( GoruntuHucresi *goruntuhucresi)
{
   do
   {
      if(goruntuhucresi->konumu == 0)
         return;

      int i = (goruntuhucresi->konumu - 1) / 2;
      GoruntuHucresi *goruntuhucresi1 = yigin[i];

      if( DahaKucuk(goruntuhucresi1, goruntuhucresi))
         return;

      yigin[goruntuhucresi->konumu] = goruntuhucresi1;
      goruntuhucresi1->konumu = goruntuhucresi->konumu;
      yigin[i] = goruntuhucresi;
      goruntuhucresi->konumu = i;
```

```
    } while(true);
}
//---------------------------------- Yığında Elemanı Sabitleme Son----------------------------------
//---------------------------------- Yığından En Küçük Elemanı Çıkarma ----------------------------
GoruntuHucresi *Yigin::MinCikar()
{
   if (indeks == -1)
      return NULL;
    GoruntuHucresi *goruntuhucresi = yigin[0];
    GoruntuHucresi *goruntuhucresi1 = yigin[indeks];
   yigin[0] = goruntuhucresi1;
   goruntuhucresi1->konumu = 0;
   indeks--;
    do
    {
      int i = goruntuhucresi1->konumu * 2 + 1;
      int j = i + 1;

      if(i > indeks)
         return goruntuhucresi;

       GoruntuHucresi *goruntuhucresi2 = yigin[i];
       GoruntuHucresi *goruntuhucresi3;

      if(j > indeks)
         goruntuhucresi3 = NULL;
      else
         goruntuhucresi3 = yigin[j];

      if(goruntuhucresi3 == NULL || DahaKucuk(goruntuhucresi2, goruntuhucresi3))
      {
         if( DahaKucuk(goruntuhucresi1, goruntuhucresi2))
            return goruntuhucresi;

         yigin[goruntuhucresi1->konumu] = goruntuhucresi2;
         goruntuhucresi2->konumu = goruntuhucresi1->konumu;
         yigin[i] = goruntuhucresi1;
         goruntuhucresi1->konumu = i;
      }
      else
      {
         if( DahaKucuk(goruntuhucresi1, goruntuhucresi3))
            return goruntuhucresi;

         yigin[goruntuhucresi1->konumu] = goruntuhucresi3;
         goruntuhucresi3->konumu = goruntuhucresi1->konumu;
         yigin[j] = goruntuhucresi1;
         goruntuhucresi1->konumu = j;
      }
   } while(true);
}

bool Yigin::DahaKucuk(GoruntuHucresi *goruntuhucresi, GoruntuHucresi
*goruntuhucresi1)
{
```

```
    return goruntuhucresi->mesafe <= goruntuhucresi1->mesafe;
}
double Yigin::min()
{
    return yigin[0]->mesafe;
}

//---------------------------------- Yığından En Küçük Elemanı Çıkarma Son----------------------------
```

EK-F. ETKİLEŞİMLİ VEKTÖR PENCERESİ PROGRAMLARI

```
void __fastcall TForm11::MerkezHatlarDXF1Click(TObject *Sender)
{
        double strtox = StrToFloat(DuzeyKumesiTestForm->Edit3->Text)-
                        (0.0*StrToFloat(DuzeyKumesiTestForm->Edit1->Text));
        double strtoy = StrToFloat(DuzeyKumesiTestForm->Edit2->Text)-
                        (0.0*StrToFloat(DuzeyKumesiTestForm->Edit4->Text));
        AnsiString xtostr = FloatToStr(strtox);
        AnsiString ytostr = FloatToStr(strtoy);
        AnsiString FName = ExtractFilePath(Application->ExeName)+"ras2vek "+"-c "
                           +"-x "+"-X"+DuzeyKumesiTestForm->Edit1->Text+" -Y
                          "+DuzeyKumesiTestForm->Edit4->Text +
                          "-e "+Edit1->Text +
                          "-t "+xtostr.c_str()+" -u "+ytostr.c_str()+" "
                           +ExtractFilePath(Application->ExeName)+"cizgi_l.bmp";
        char *FName2=FName.c_str();
        WinExec(FName2,SW_SHOWNORMAL);
        Form11->Show();
}
//---------- Merkez Hatların Vektöre Dönüştürülmesi Son----------
//----------Sınırların Vektöre Dönüştürülmesi ----------
void __fastcall TForm11::SnrlarDXF1Click(TObject *Sender)
{

        double strtox = StrToFloat(DuzeyKumesiTestForm->Edit3->Text)-
                        (1.5*StrToFloat(DuzeyKumesiTestForm->Edit1->Text));
        double strtoy = StrToFloat(DuzeyKumesiTestForm->Edit2->Text)-
                        (0.5*StrToFloat(DuzeyKumesiTestForm->Edit4->Text));
        AnsiString xtostr = FloatToStr(strtox);
        AnsiString ytostr = FloatToStr(strtoy);
        AnsiString FName = ExtractFilePath(Application->ExeName)+"ras2vek "+"-d "
                           +"-x "+"-X"+DuzeyKumesiTestForm->Edit1->Text+" -Y
                          "+DuzeyKumesiTestForm->Edit4->Text +
                          "-e "+Edit1->Text +
                          "-t "+xtostr.c_str()+" -u "+ytostr.c_str()+" "
                          +ExtractFilePath(Application->ExeName)+"cizgi_p.bmp";
        char *FName2=FName.c_str();
        system(FName2);
        Form11->Show();
}
//----------Sınırların Vektöre Dönüştürülmesi Son----------
```

EK-G DAVIDE LIBENZI'NIN RASTERDAN VEKTÖRE DÖNÜŞTÜRME PROGRAMININ GÜNCELLENMİŞ KODU

```
//---------- RAS2VEK ----------------------------------------------------------
#include<windows.h>
#include<windowsx.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<malloc.h>
#include<math.h>
#include<float.h>
#include<time.h>
#include<sys/types.h>
#include<sys/stat.h>
#include"ras2vec.h"
#include"dbll_list.h"
#include"bmputil.h"
#include"filescan.h"
#include"himage.h"
#include"vectorize.h"
#include"thinner.h"
#include"log.h"
#include"util.h"
#include"tiffio.h"
#include"tiffext.h"

#define VERSION_HI              1
#define VERSION_LO              1
#define MAX_DEMO_SIZE           202
#define CENTER_LINE             1
#define DOUBLE_LINE             2
#define DXF_OUTPUT              1
#define HPGL_OUTPUT             2
#define POLYLINE_OUTPUT         3
#define EMF_OUTPUT              4
#define MAX_TOLLERANCE          10.0

#define bound_value(v,mn,mx)    (min((mx),max((mn),(v))))
//-----------------------------------------------------------------------------
static int make_conversion_cmtline(char *img_file_name, double pnt_error_tollerance,
                double poly_lenght_factor, char *out_poly_file_name)
{

   int          use_a_copy = TRUE;
   time_t       start_time,
                end_time;
   char         start_time_str[128],
                end_time_str[128],
                file_name_ext[MAX_PATH],
                temp_img_file_name[ MAX_PATH] = "";

   _splitpath(img_file_name, NULL, NULL, NULL, file_name_ext);
```

```
   if (stricmp(file_name_ext, ".tif") == 0)
   {
      GetTempFileName(".", "img", 0, temp_img_file_name);
      scr_printf(img_file_name,"- tiff dosyasindan %s",temp_img_file_name," bmp
dosyasina donusturuldu %s\n");
      if (!tiff_file_to_bmp_file(img_file_name, temp_img_file_name))
      {
         remove(temp_img_file_name);
         log_printf("- tiff %s bmp ye donusturme hatasi %s\n", img_file_name,
               temp_img_file_name);
         return (FALSE);
      }
      img_file_name = temp_img_file_name;
      use_a_copy = FALSE;
   }

   time( &start_time);
   scr_printf("- vektore donusturuluyor %s\n", img_file_name);
   if (!cntline_convert_file(img_file_name, use_a_copy, out_poly_file_name,
            pnt_error_tollerance, poly_lenght_factor))
   {
      if (strlen(temp_img_file_name) > 0)
         remove(temp_img_file_name);
      log_printf(img_file_name, "- dosyasindan vektorler cizilirken hatayla karsilasildi
%s\n");
      return (FALSE);
   }
   time( &end_time);
   strcpy(start_time_str, ctime( &start_time));
   strcpy(end_time_str, ctime( &end_time));
   scr_printf("- vektore donusturme basariyla tamamlandi\n"
         "- baslangic zamani ...: %s"
         "- bitis zamani .....: %s"
         "- gecen hesap suresi .....: %lf sec\n", start_time_str,
         end_time_str, difftime(end_time, start_time));
   if (strlen(temp_img_file_name) > 0)
      remove(temp_img_file_name);
   return (TRUE);

}
//-----------------------------------------------------------------------------------------------
static int  make_conversion_dblline(char *img_file_name, double pnt_error_tollerance,
               double poly_lenght_factor, char *out_poly_file_name)
{

   time_t      start_time,
            end_time;
   char        start_time_str[128],
            end_time_str[128],
            file_name_ext[MAX_PATH],
            temp_img_file_name[ MAX_PATH] = "";

   _splitpath(img_file_name, NULL, NULL, NULL, file_name_ext);
   if (stricmp(file_name_ext, ".tif") == 0)
   {
```

```
        GetTempFileName(".", "img", 0, temp_img_file_name);
        scr_printf(img_file_name,"- tiff dosyasindan %s",temp_img_file_name," bmp
dosyasina donusturuldu %s\n");
        if (!tiff_file_to_bmp_file(img_file_name, temp_img_file_name))
        {
          remove(temp_img_file_name);
          log_printf("- tiff %s bmp ye donusturme hatasi %s\n", img_file_name,
                temp_img_file_name);
          return (FALSE);
        }
        img_file_name =temp_img_file_name;
    }

    time( &start_time);
    scr_printf(img_file_name, "- dosyasından vektorler olusturuluyor %s\n");
    if (!dbline_convert_file(img_file_name, out_poly_file_name, pnt_error_tollerance,
                poly_lenght_factor))
    {
        if (strlen(temp_img_file_name) > 0)
          remove(temp_img_file_name);
        log_printf(img_file_name, "- dosyasindan vektorler cizilirken hatayla karsilasildi
%s\n");
        return (FALSE);
    }
    time( &end_time);
    strcpy(start_time_str, ctime( &start_time));
    strcpy(end_time_str, ctime( &end_time));
    scr_printf("- vektore donusturme basariyla tamamlandi\n"
          "- baslangic zamani ...: %s"
          "- bitis zamani .....: %s"
          "- gecen hesap suresi .....: %lf sec\n", start_time_str,
          end_time_str, difftime(end_time, start_time));
    if (strlen(temp_img_file_name) > 0)
        remove(temp_img_file_name);
    return (TRUE);

}
//-----------------------------------------------------------------------------------------
int    main(int argc, char *argv[])
{

    int          arg,
                 color = 0,
                 conversion_mode = CENTER_LINE,
                 output_mode = DXF_OUTPUT;
    double       x_scale = 1.0,
                 y_scale = 1.0,
                 pnt_error_tollerance = CNT_DEF_MAX_PNT_ERROR,
                 poly_lenght_factor = -1.0,
                                        xxx = 0.0,
                                        yyy = 0.0;
    char         layer_name[64] = "RAS2VEC",
                 out_directory[MAX_PATH] = ".";

    init_logging(log_proc, err_proc, log_proc);
```

```
    TIFFSetErrorHandler((TIFFErrorHandler) tiff_error_handler);
    TIFFSetWarningHandler((TIFFErrorHandler) tiff_warning_handler);

    if (argc < 2)
    {
      log_printf("- kullanim: ras2vec [-cdhxpXYeo] img_file ...\n"
            "      -c      = merkez hat (default)\n"
            "      -d      = poligon (alan)\n"
            "      -h      = hpgl cikti\n"
            "      -m      = emf cikti\n"
            "      -x      = dxf cikti (default)\n"
            "      -p      = polyline cikti\n"
            "      -Xsf    = X olcek faktoru (default 1.0)\n"
            "      -Ysf    = Y olcek faktoru (default 1.0)\n"
            "      -etf    =toleransi ayarla [0..10] (default %f)\n"
            "      -odir   = cikti klasorunu ayarla\n"
            "      -flf    = minimum cizgi uzunlugunu ayarla max(x, y) / plf (default -1.0 =
none)\n"
            "      img_file  =kullanilabilecek goruntu formatlari {bmp | tif}\n"
            "                siyah & beyaz, 1 bit x piksel\n"
            "                bmp uncompressed, tif tum tipler\n", pnt_error_tollerance,
            out_directory);
      return (__LINE__);
    }
    for (arg = 1; (arg < argc) &&(argv[arg][0] == '-'); arg++)
    {
      switch (argv[arg][1])
      {
        case ('c'):
          conversion_mode = CENTER_LINE;
          break;

        case ('d'):
          conversion_mode = DOUBLE_LINE;
          break;

        case ('h'):
          output_mode = HPGL_OUTPUT;
          break;

        case ('x'):
          output_mode = DXF_OUTPUT;
          break;

        case ('m'):
          output_mode = EMF_OUTPUT;
          break;

        case ('p'):
          output_mode = POLYLINE_OUTPUT;
          break;

        case ('o'):
          if (++arg < argc)
          {
```

```
            int   ii,
                  dir_length = strlen(argv[arg]);

            strcpy(out_directory, argv[arg]);

            for (ii = (dir_length - 1); out_directory[ii] == '\\'; ii--)
               out_directory[ii] = '\0';
          }
          break;

        case ('X'):
          if (++arg < argc)
            x_scale = atof(argv[arg]);
          break;

        case ('Y'):
          if (++arg < argc)
            y_scale = atof(argv[arg]);
          break;

        case ('e'):
          if (++arg < argc)
            pnt_error_tollerance = atof(argv[arg]);
          pnt_error_tollerance = bound_value(pnt_error_tollerance, 0.0,
MAX_TOLLERANCE);
          break;

        case('f'):
          if (++arg < argc)
            poly_lenght_factor = atof(argv[arg]);
          break;

                    case ('t'):
          if (++arg < argc)
                        xxx = atof(argv[arg]);
          break;

                    case ('u'):
                        if (++arg < argc)
          yyy = atof(argv[arg]);
          break;


      }
    }
    for (; arg < argc; arg++)
    {
      HANDLE        find_handle;
      char          *p_bkslsh = strrchr(argv[arg], '\\');
      WIN32_FIND_DATA wfd;
      char          base_file_path[MAX_PATH] = "";

      if (p_bkslsh != NULL)
      {
        int           base_path_length = (int) (p_bkslsh - argv[arg]) + 1;
```

```
    strncpy(base_file_path, argv[arg], base_path_lengtn);
    base_file_path[base_path_lengtn] ='\0';
  }

 if ((find_handle = FindFirstFile(argv[arg], &wfd)) == INVALID_HANDLE_VALUE)
 {
    log_printf("- unable to find %s\n", argv[arg]);
    continue;
 }
 do
 {
    int          conversion_result;
    char         input_file_name[ MAX_PATH],
                 file_name_base[ MAX_PATH],
                 file_name_ext[ MAX_PATH],
                 out_poly_file_name[ MAX_PATH];

    sprintf(input_file_name, "%s %s", base_file_path, wfd.cFileName);
    _splitpath( wfd.cFileName, NULL, NULL, file_name_base, file_name_ext);
    sprintf(out_poly_file_name, "%s\\ %s.ply", out_directory, file_name_base);
    switch (conversion_mode)
    {
       case (DOUBLE_LINE):
          conversion_result = make_conversion_dbline(input_file_name,
                pnt_error_tollerance, poly_lenght_factor * 4.0,
                out_poly_file_name);
          break;

       case (CENTER_LINE):
          conversion_result = make_conversion_cntline(input_file_name,
                pnt_error_tollerance, poly_lenght_factor,
                out_poly_file_name);
          break;
    }
    if (conversion_result)
    {
       char          out_file_name[ MAX_PATH];

       switch (output_mode)
       {
          case (EMF_OUTPUT):
             {
                sprintf(out_file_name, "%s\\ %s.emf", out_directory, file_name_base);
                scr_printf("- emf dosyasini olusturuyor %s\n", out_file_name);
                if (!emf_poly_dump(out_file_name, out_poly_file_name, x_scale, y_scale,
                          RGB( 0, 0, 0)))
                   log_printf("- emf dosyasini yazarken hata %s\n", out_file_name);
                remove(out_poly_file_name);
             }
             break;

          case (HPGL_OUTPUT):
             {
                sprintf(out_file_name, "%s\\ %s.hgl", out_directory, file_name_base);
```

```
                    scr_printf("- hpgl dosyasini olusturuyor %s\n", out_file_name);
                    if (!hpgl_poly_dump(out_file_name, out_poly_file_name, x_scale, y_scale,
                                color))
                      log_printf("- hpgl dosyasini yazarken hata %s\n", out_file_name);
                    remove(out_poly_file_name);
                  }
                  break;

                case (DXF_OUTPUT):
                  {
                    sprintf(out_file_name, "%s\\%s.dxf", out_directory, file_name_base);
                    scr_printf("- dxf dosyasini olusturuyor %s\n", out_file_name);
                    if (!dxf_poly_dump(out_file_name, out_poly_file_name, x_scale, y_scale,
xxx,
                                                                        yyy, layer_name,
color))
                      log_printf("- dxf dosyasini yazarken hata %s\n", out_file_name);
                    remove(out_poly_file_name);
                  }
                  break;

                case (POLYLINE_OUTPUT):
                  break;
              }
            }
            else
              remove(out_poly_file_name);
            scr_printf("- islemler basariyla sonuclandirildi \n\n");
        } while (FindNextFile(find_handle, &wfd));
        FindClose(find_handle);
    }
          scanf ("%d");
    return (0);

}
//---------------RAS2VEK SON----------------------------------------------------------
```

ÖZGEÇMİŞ

Oktay EKER, 1972 yılında Eskişehir'de doğdu. İlk öğrenimini Mehmetçik İlkokulu'nda, orta öğrenimini Eskişehir Anadolu Lisesi'nde 1987 yılında tamamladı. Aynı yıl Kuleli Askeri Lisesi'nde öğrenimine başladı ve 1990 yılında buradan mezun olarak Kara Harp Okulu'nda öğrenimine devam etti. 1994 yılında Kara Harp Okulu'ndan Harita Subayı olarak mezun olarak M.S.B. Harita Yüksek Teknik Okulu'nda iki yıl süreyle mühendislik öğrenimi gördü ve Harita Mühendisi ünvanıyla mezun oldu. 2000 yılında İstanbul Teknik Üniversitesi, Jeodezi ve Fotogrametri Mühendisliği bölümünde Yüksek Lisansını tamamladı.

1996 yılında Harita Yüksek Teknik Okulu'ndan mezun olduktan sonra Harita Genel Komutanlığı, Fotogrametri Dairesi'nde göreve başladı, Fotogrametri Dairesi'nin farklı kademelerinde görev yaptıktan sonra halen aynı dairede Sayısal Kıymetlendirme Kısım Amiri olarak görevine devam etmektedir. Evli ve bir kız çocuk babasıdır.