KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

PARSEL KESİŞİMLERİNDE GEOMETRİK ALGORİTMALARIN İNCELENMESİ

Emirhan ÖZDEMİR

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"YÜKSEK LİSANS (HARİTA MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 26 / 03 /2018
Tezin Savunma Tarihi : 20 / 04 /2018

Tez Danışmanı : Doç. Dr. Faruk YILDIRIM

Trabzon 2018

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Harita Mühendisliği Anabilim Dalında
Emirhan ÖZDEMİR Tarafından Hazırlanan

PARSEL KESİŞİMLERİNDE GEOMETRİK ALGORİTMALARIN İNCELENMESİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 03 / 04 / 2018 gün ve 1747 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Doç. Dr. Faruk YILDIRIM .....

Üye : Doç. Dr. Mehmet ALKAN .....

Üye : Doç. Dr. Volkan YILDIRIM .....

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

ÖNSÖZ

Bu tez çalışması Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalı'nda yüksek lisans tezi olarak hazırlanmıştır.

"Parsel Kesişimlerinde Geometrik Algoritmaların İncelenmesi" isimli tez çalışmasını bana öneren, çalışmamın her aşamasında benim yanımda olan, engin bilgi ve tecrübelerinden daima yararlandığım ve beni sürekli olarak teşvik etmeyi ihmal etmeyen çok değerli Hocam Sayın Doç. Dr. Faruk YILDIRIM'a teşekkürlerimi sunmayı her zaman bir borç bilirim. Tüm bölüm hocalarıma, çalışmam boyunca benden yardımlarını esirgemeyen Arş. Gör. Süleyman ŞENGÜL'e, akademik ve normal hayattaki güçlüklere birlikte göğüs gerdiğimiz, her zaman maddi ve manevi olarak yanımda olan değerli arkadaşım Arş. Gör. Yusuf YANIK'a en içten teşekkürlerimi sunarım.

Bu günlere gelmemde ellerinden gelen tüm imkanları sağlayan, özellikle hayatımın bu önemli aşamasında maddi ve manevi desteklerini eksik etmeyen, haklarını hiçbir zaman ödeyemeyeceğim ve bana her türlü desteği sağlayan aileme müteşekkir olduğumu belirtir, bu çalışmanın ülkemize faydalı olmasını temenni ederim.

Emirhan ÖZDEMİR

Trabzon 2018

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Parsel Kesişimlerinde Geometrik Algoritmaların İncelenmesi” başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. Faruk YILDIRIM’ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.
20/ 04/ 2018

Emirhan ÖZDEMİR

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

PARSEL KESİŞİMLERİNDE GEOMETRİK ALGORİTMALARIN İNCELENMESİ

Emirhan ÖZDEMİR

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Faruk YILDIRIM
2018, 64 Sayfa

Günümüzde birçok haritacılık uygulamasında (kamulaştırma, kentsel ve kırsal düzenlemeler vb.) geometrik olarak birçok yöntem uygulanmaktadır. Bu uygulamalarda karşılaşılacak durumlardan birisi ise iki parselin birbirine göre durumlarıdır.

Tez kapsamında; öncelikle verilen bir parsel ve keyfi noktalar için nokta-parsel ilişkileri incelemiştir. Bu noktaların parselin içinde veya dışında olup olmadığı sorgulanmıştır. Sonrasında, verilen iki parselin birbirlerine göre durumları irdelenip, bu iki parselin birbirleri ile kesişen bölgeleri varsa, bu bölgenin köşe noktaları bulunmuş ve alan hesabı yapılmıştır. Alan hesabı yapılırken iki parselin birbirlerine göre özel durumları incelenmiştir. Matlab programı yardımıyla oluşturulan algoritma bu özel durumların her biri için çalıştırılıp, elde edilen sonuçlar şekiller yardımıyla verilmiştir. Elde edilen sonuçlar CAD ve CBS yazılımları ile karşılaştırılıp avantaj ve dezavantajları irdelenmiştir.

Yapılan bu algoritma, arazide iki parsel için karşılaşılacak durumlarda altlık olabilecek düzeyde olup, hesaplamalarda büyük kolaylık sağlayacağı öngörülmektedir.

Anahtar Kelimeler: Nokta-parsel ilişkileri, parsel-parsel ilişkileri, Weiler–Atherton Yöntemi, Nokta-Vektör Yöntemi.

Master Thesis

SUMMARY

INVESTIGATION OF GEOMETRIC ALGORITHMS IN PARSEL INTERSECTION

Emirhan ÖZDEMİR

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatic Engineering Graduate Program
Supervisor: Assoc. Prof. Faruk YILDIRIM
2018, 64 Pages

Today, many mapping techniques (expropriation, urban and rural regulations, etc.) are applied geometrically. One of the situations that will be encountered in these applications is the situation of two parcels relative to each other.

Within the thesis; he first examined a given parcel and point-parcel relations for arbitrary points. It has been questioned whether these points are inside or outside the parcel. Then, if the two parcels are analyzed according to each other and if these two parcels intersect with each other, the corner points of this region are found and the account of the area is made. When the area account is made, special cases of two parcels are examined according to each other. The algorithm generated by the help of Matlab program is executed for each of these special cases and the obtained results are given with formative help. The results are compared with CAD and GIS software and their advantages and disadvantages are examined.

This algorithm is designed to be a base for two parcels in the field, and it is envisaged that it will provide great convenience in calculations.

Keywords: Point-Parcel Relations, Parcel-Parcel Relations, Weiler–Atherton Methods, Point-Vector Methods.

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

M_y	$\lvert y_0 - y_i \rvert$ mutlak değer farkı en büyük değer
m_x	$\lvert x_0 - x_i \rvert$ mutlak değer farkı en küçük olan sıfırdan farklı değer

1. GENEL BİLGİLER

1.1. Giriş

1.1.1. Problemin Tanımı

Haritacılık uygulamalarında (kamulaştırma, kentsel ve kırsal düzenlemeleri, CBS sorgulama ve analiz uygulamaları vb.) iki veya daha fazla parselin kesişmesi sonucunda ortak bir alan oluşmaktadır. Ortak olarak oluşan bu alanın yüz ölçümünün ve kesişim noktalarının koordinatlarının hesabı için birçok algoritma mevcuttur. Bu algoritmaların işlem zamanlarının belirlenmesi ve özel durumlar (kesiştirilecek iki parselin kenarlarının; paralel olması, kesişmesi, dik kesişmesi vb.) için sorgulanması algoritmanın kullanılması için genel bir tercihtir. Ayrıca bu algoritmaların otomatik yapılıp kullanıcının, nokta seçimi ve alan hesabı yükünden kurtulması amaçlanması gerekmektedir. Haritacılık için kullanılan CAD ve CBS yazılımlarının bu amaçlar doğrultusunda gerçekleşip gerçekleşmediğini araştırılması gerekmektedir.

1.1.2. Tezin Amacı

Problemin tanımı kısmında bahsedilen haritacılık programlarının parsel sorgulamalarına hangi ölçüde cevap verebildiğini araştırmak için Matlab programında bir arayüz yazılması amaçlanmıştır. Bu arayüz; parsellerin çakışması sonucunda ortaya çıkan ortak parselin öncelikli olarak köşe koordinatlarını bulup, daha sonrasında bulunan bu köşe koordinatları yardımıyla ortak parselin alanını bulmayı amaçlamaktadır.

Matlab'da yapılan algoritmayla; verilen iki parselin kesişimi ve oluşacak alanın hesaplanması için, problemin tanımı kısmında bahsedilen özel durumlar ve işlem zamanı tayini farklı algoritmalar kullanılmıştır. Bu farklı algoritmalar içinden tüm özel durumlar için en kısa sürede çalışan algoritma belirlenmiştir. Ayrıca algoritmada kullanıcının kesişim noktası belirleyip, bu noktalardan alan hesabı için parselin belirlemesine gerek kalmamıştır. Bu algoritma harita mühendisliği için kullanılan CAD ve CBS yazılımları ile karşılaştırılmıştır. Böylece bu yazılımların avantaj ve dezavantajları irdelenmiştir.

1.1.3. Metodoloji

Genel bilgilerde;

- ❖ Haritacılıkta parsel ilişkileri,
- ❖ Geometrik ilişkiler,
- ❖ Nokta-parsel ilişkileri,
- ❖ Parsel oluşturma yöntemleri,

Uygulamada;

- ❖ Matlab program algoritması,
- ❖ Uygulama alanları,
- ❖ Program çıktıları,
- ❖ Çıktılar yorumlanarak bulgular ve irdelemeler kısmı, çalışılan konu başlıkları irdelenerek sonuç ve öneriler kısmı belirlenmiştir.

1.2. Haritacılıkta Parsel İlişkileri

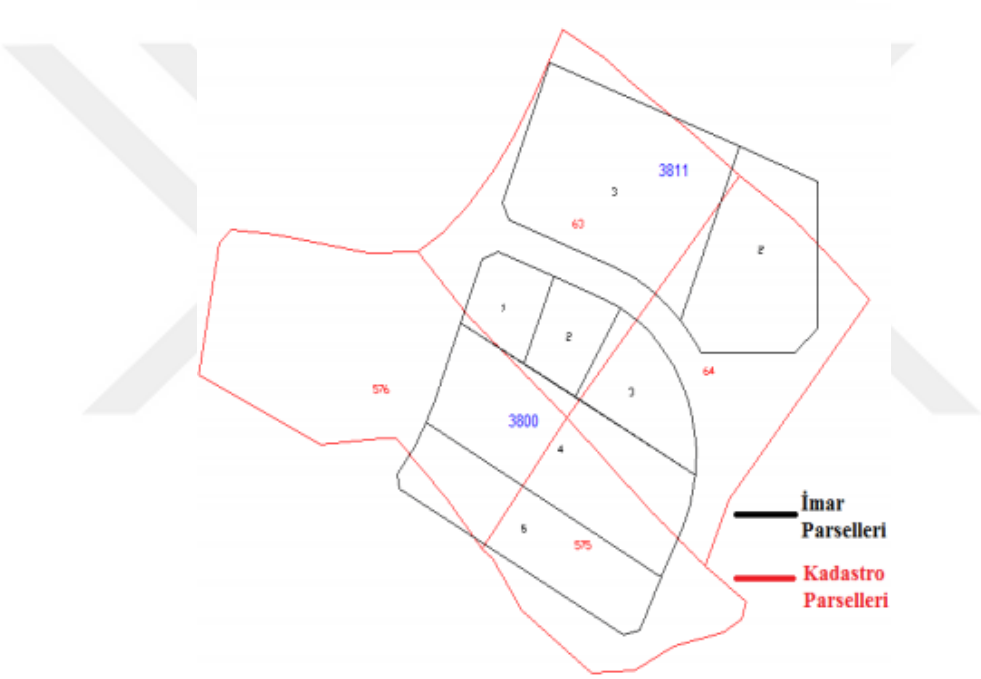
Haritacılıkta nokta, doğru ve parsel ilişkilerini sorgulayan ve bu sorgulama sonucunda koordinat ve alan bilgisine ihtiyaç duyulan birçok uygulama vardır. Bunlardan bazıları kamulaştırma, kentsel ve kırsal alan düzenlemeleri vb. gibidir. Bu uygulamaların birçoğunda geometri bilgisine ihtiyaç duyulmaktadır.

İmar planlarında yol, meydan, çocuk bahçesi, park, otopark, trafo gibi belde de kamu tesislerine, hayvan kesim yeri (mezbaha), toptancı hali, garaj, itfaiye gibi belediye hizmet binalarına ayrılmış yerler ile okul, hastane, kreş, resmi kurum hizmet alanları vardır. Tesis ve binaların planda yerleştirildiği araziler çoğu kez belediye ve kamu kurumlarının mülkiyetinde değildir. Öte yandan tesis ve binaların yapılabilmesi için bu arazilerin öncelikle kamuya mal edilmesi gerekir. İşte kamu yararının gerektirdiği durumlarda, özel mülkiyetteki taşınmazların, kamu yönetimlerince kamu mülkiyetine zorla geçirilmesine kamulaştırma (istimlâk) denir. Böyle bir zorunluluk, kanun ve yönetmeliklerle, ilgili kurum ve kuruluşlar tarafından yapılmaktadır (Kitay, 1985). Kamulaştırmayla, kadastro parsellerinden kamulaştırmaya giden alanın ve bu alanın köşe koordinatlarının belirlenip araziye aplikasyonu amaçlanmaktadır (Tırak, 2017).

Terk; Taşınmazın imar planına uygun hale getirilmesi için imar planlarında yol, otopark, yeşil alan gibi kamusal hizmetlere rastlayan kısımların, taşınmaz sahiplerinin

isteğine göre ya da reesen plandaki fonksiyonda kullanılmak üzere bedelsiz olarak kamuya bırakılması işlemidir (Çelik, 2013). Terk işleminde, kadastro parselinden imar uygulamasına göre ne kadarlık bir alanın terk edileceğinin belirlenmesi ve köşe koordinatlarının araziye aplikasyonu gerçekleştirilmektedir.

Arsa düzenlemesinde amaç; arazi ve arsaların imar planına uygun hale getirilmesi, yeni oluşan parsellerin eşitlik ilkesine göre dağıtılması, imar planında genel hizmetlere ayrılmış alanlar için gereken toprak ihtiyacının karşılanması ve bu umumi hizmet tesislerinden arsa ve arazilerin yararlanmasını sağlamaktır (Yakar, 2000).



Şekil 1.1. İmar parseli ile kadastro parselinin çakıştırılması

Kırsal alan düzenlemesi; Bakanlar Kurulunca ilan edilen proje alanlarında, gerçek ve tüzel kişiler ile kamuya ait tarım arazilerinin parsel sayısının asgariye indirilerek birleştirilmesi ve yeniden düzenlenmesi, gerektiğinde genişletilmesi, toprakların verimli bir şekilde işletilmesi ve bu durumun korunması, ucuz ve kaliteli üretimin artırılmasına imkan sağlayan önlemlerin alınması, toprakların ekonomik bir şekilde işlenmesini sağlamak için yol, sulama ve drenaj şebekelerinin kurulması, arazi tesviyesi, erozyon önleyici tesisler ve yan dere yatağı ıslahının yapılması, arazi yeteneği ve toprak özelliklerine göre çayır, mera

dâhil tüm bitkisel üretim alanları ile iskân ve sanayi yerlerinin belirlenmesi, mevcut mera alanlarının düzenli oluşturulmasını kapsar.

Yukarıda özetlenen haritacılık uygulamalarında yapılan işlemlerde ortak alanlar oluşmaktadır (Şekil 1.1.). Bu bölgelerin köşe koordinatları ve alan hesapları; nokta, doğru ve parsel ilişkilerinin geometrik bir çözümüdür.

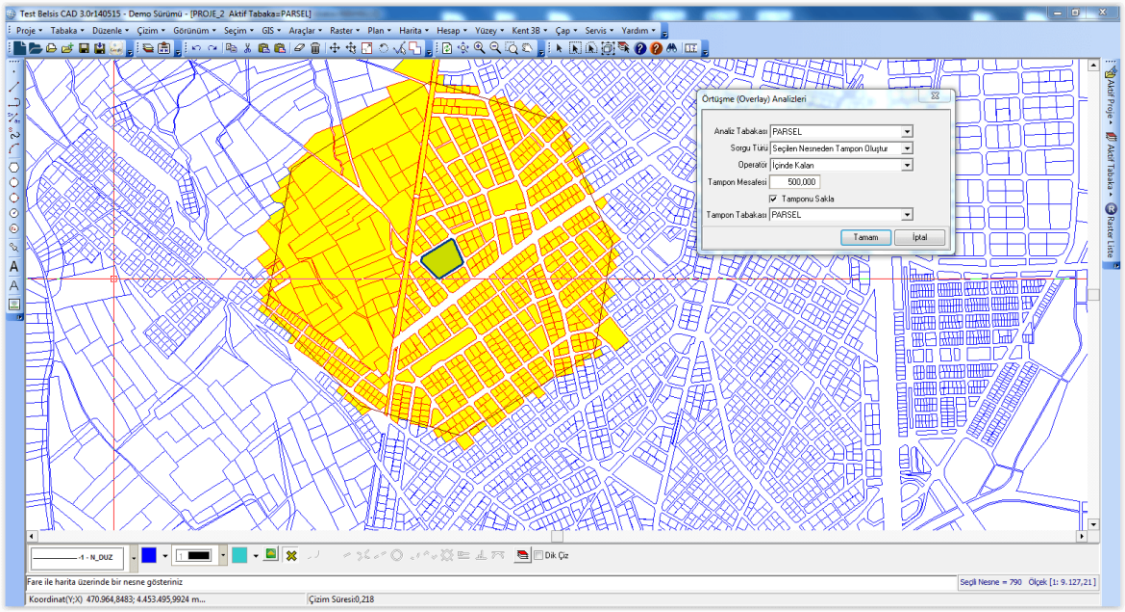
1.3. CBS' de Önemi

Coğrafi bilgi sistemlerinin gelişimi bilgi teknolojisindeki değişimlere bağlı olmakla birlikte, harita işlemlerinin daha hızlı ve doğru yapılabilmesi için bilgisayardan yararlanma isteği de bu süreci hızlandırmıştır. Haritalardaki karmaşık bilgi yapısının daha anlaşılır olabilmesi ve yoğun bilginin denetlenebilmesi için bilgisayar desteği kaçınılmaz olmuştur.

Sayısal Arazi Modelleri, üçgen ağlarından oluşan modelden yükseklik enterpolasyonuyla oluşturulmaktadır. Yeni bir noktanın bu SAM modelin içinden hangi üçgenin içinde olduğu veya bu modelin dışından olduğunun bulunması problemi CBS yazılımları için iyi bir örnektir. SAM modelinin oluşturulmasında arazi topoğrafyasının daha iyi temsil edilmesi için seçilen noktaların dağılım ve yeri önemlidir (Yanalak, M. ve İpbüker, C., 2003).

Konum analizlerinde herhangi bir coğrafik bölgenin tamamı yerine bu bölge içerisindeki belli bir alana ait bilgilerden yararlanmak gerekebilir. Örneğin bir kentte ilişkin konumsal bilgilerden sadece bir mahalleye ait olan bilgiler ayıklanarak yeni bir mahalle veri dosyası oluşturulabilir. Böylece haritanın bütünü yerine, sınırla tespit edilecek, sadece istenen bölge yeniden elde edilmiş olacaktır. Bir diğer örnek ise; bir kentin beldelerini gösteren katmandan, o kente ait yolları gösteren katmandaki grafik bilgilere isabet edecek bölgeleri ayırıp, yollar ile sınırlandırılmış beldeleri gösteren yeni bir katman elde edilir (Şekil 1.2.). Öncelikle ayırma katmanı ile bu katmandaki detaylara isabet etmesi istenen bindirme katmanı üst üste çakıştırılarak ortak alanların bulunduğu yeni katman oluşturulur (Yomralıoğlu, 2000).

Yukarıda verilen örneklerde görüleceği üzere, çakışan katmanların veya parsellerin ortak oluşturduğu bölgelerin köşe koordinatları veya alan hesabı CBS için önemli faktörlerdir.



Şekil 1.2. CAD yazılımı örneği

1.4. Geometrik Hesaplamalar

1.4.1. Doğru Denklemi

Doğrunun genel denklemi $ax + by + c = 0$ ya da $y = mx + n$ şeklindedir. Doğru denklemine bakılarak eğim kolaylıkla bulunabilmektedir. Doğru dekleminde y yalnız bırakıldığında x'in katsayısı eğimi vermektedir. Buradan

$$ax + by + c = 0 \tag{1.1}$$

$$y = -\frac{a}{b}x - \frac{c}{b} \tag{1.2}$$

$$m = -\frac{a}{b} \tag{1.3}$$

elde edilir.

Eğimi m olan ve $M(x_0, y_0)$ noktasından geçen doğru denklemi

$$y - y_0 = m(x - x_0) \tag{1.4}$$

şeklindedir. Aynı zamanda eğim,

$$m = \frac{y - y_0}{x - x_0} \tag{1.5}$$

elde edilir. Eğer doğrunun sadece geçtiği iki nokta ($A(x_1, y_1)$, $B(x_2, y_2)$) biliniyorsa, öncelikli olarak eğim,

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{1.6}$$

elde edilir. Doğru denklemi ise

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{1.7}$$

şeklindedir. Eğimin bir diğer ifade şeklide; bir doğrunun pozitif yönde, (saatin dönüş yönünün aksi yönü) x ekseni ile yaptığı açının tan değeri, bu doğrunun eğimini vermektedir. Eğer doğrunun x ekseni ile yaptığı açı θ ise eğim,

$$m = tan\theta \tag{1.8}$$

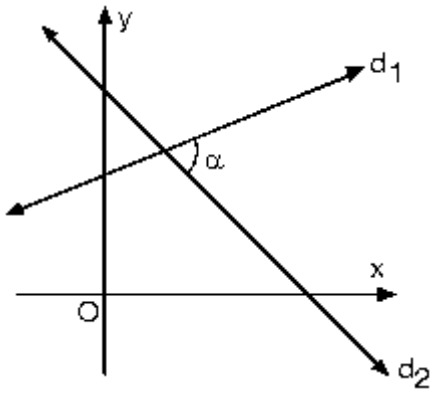
şeklinde bulunur.

1.4.2. İki Doğru Arasındaki Açı

d_1, d_2 kesişen iki doğru ve sırasıyla eğimleri m_1, m_2 olsun. Kesişen bu iki doğrunun arasındaki açı θ olmak üzere

$$tan\,\theta = \frac{m_1 - m_2}{1 + m_1 m_2} \tag{1.9}$$

elde edilir (Şekil 1.3.).

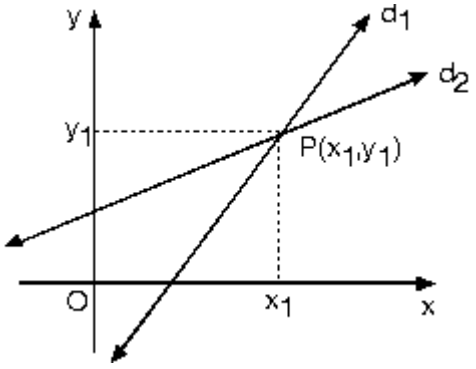


Şekil 1.3. Kesişen iki doğru arsındaki açı

Buradan birbirine paralel iki doğrunun arasındaki açı 0 olduğundan, bu iki doğrunun eğimleri eşittir. Eğer iki doğru birbirine dik ise bu iki doğrunun eğimler çarpımı -1 olmaktadır.

1.4.3. Doğruların Kesişim Noktası

İki boyutlu bir düzlemde kesişen iki doğru sadece bir noktada kesişmektedir (Şekil 1.4.). Kesiştikleri bu nokta, her iki doğruda da mevcut olduğundan her iki denklem ortak çözüm yapılarak, kesiştikleri nokta bulunur. Bu kesişim noktası aşağıdaki adımlarla bulunmaktadır. Öncelikle d_1, d_2 kesişen iki doğru, sırasıyla $(x_1, y_1), (x_2, y_2)$ geçtikleri noktalar ve m_1, m_2 bu doğruların eğimleri olsun.



Şekil 1.4. Kesişen iki doğru

Buradan

$$d_1: y = m_1(x - x_1) + y_1 \tag{1.10}$$

$$d_2: y = m_2(x - x_2) + y_2 \tag{1.11}$$

doğru denklemleri elde edilir. Öncelikle, bu doğru denklemleri karşılıklı olarak birbirlerine eşitlenirse

$$m_1(x - x_1) + y_1 = m_2(x - x_2) + y_2 \tag{1.12}$$

elde edilir. Buradan x ifadesi yalnız bırakılırsa

$$x = \frac{m_1 x_1 - y_1 - m_2 x_2 + y_2}{m_1 - m_2} \tag{1.13}$$

bulunur. Sonrasında, y ifadesini bulmak için bulunan x eşitliği, d_1 veya d_2 doğru denklemlerinin herhangi birinde yerine yazılarak

$$y = m_1 \frac{m_1 x_1 - y_1 - m_2 x_2 + y_2}{m_1 - m_2} + y_1 \tag{1.14}$$

elde edilir. Böylelikle kesişen iki doğrunun kesiştikleri nokta kolaylıkla bulunur.

1.5. Nokta Parsel İlişkileri

Nokta parsel sorgulaması; konumu belirli bir noktanın, tüm köşe koordinatları belli olan bir parselin içinde olup olmadığının incelenmesidir. Bu inceleme için birçok matematiksel ve geometrik yöntem bulunmaktadır. Bu yöntemler konveks ve konkav parsellerin şekillerine göre değişkenlik göstermektedir.

Sutherland ve Hodgman, dışbükey, düzlemsel veya düzlemsel olmayan poligonlar için geçerli olan bir algoritma geliştirmişlerdir. Bu algoritmadaki poligonlar sadece köşe

noktaları ile değil aynı zamanda kenarlarıyla birlikte tanımlanmaktadır. Bu yaklaşım, çokgenlerin kırpılması sürecini basitleştirmiştir (Sutherland ve Hodgman, 1974).

Weiler ve Atherton, birbiri içinde bulunan iki genel içbükey çokgeni kırpabilmek için bir algoritma geliştirmişlerdir (Weiler ve Atherton, 1977). Bu algoritmada işlem süresi verilen çokgenlerin karmaşıklığı ile ilgilidir.

Cyrus ve Beck, bir kırpma algoritmasını, 2D ve 3D'de nasıl uygulandığını açıklamaktadır. Bu algoritma, bir çizgi ile dışbükey düzlemsel çokyüzlü arasındaki kesişim noktasını kontrol etmektedir (Cyrus ve Beck, 1978).

Barsky ve Liang, dikdörtgen bir kırpma penceresi yardımıyla kırpma yöntemi sunmuşlardır (Barsky ve Y. D. Liang, 1984). Bu algoritma, kırpılacak çizgi parçasını parametrik bir gösterime eşleştirmektedir. Bu algoritma için öne sürülen temel avantaj, görünmez kesimleri reddetmesi, yani kırpma penceresinin sınırları dışındaki bölümleri reddetmesidir ve algoritmanın herhangi bir dışbükey çokgene karşı kırpılmak üzere genelleştirilebilmesidir. Ancak algoritmanın dezavantajı, ana sınırlama çokgeninin daima eksenlere paralel olan bir dikdörtgen olması gerektiğidir.

Nicholl ve diğ., 2D'de kırpma çizgileri için dikdörtgen bir pencere yardımıyla kırpma algoritması geliştirmiştir (Nicholl, Lee ve Nicholl, 1987). Bu algoritma, Barsky-Liang algoritması ve Cohen-Sutherland algoritması' dan daha iyi performans vermektedir. Bu algoritma, çizgi kesimi ile pencere sınırları arasındaki kesişme noktalarını x-sol ve x-sağ ile y-üst ve y-alt ile tanımlamaktadır. Bir kesişim mevcutsa, çizgi parçasının pencere ile kesişen uç noktalarının koordinatları hesaplanmaktadır.

Maillot, Cohen-Sutherland'in çizgi kırpma algoritmasını benimseyen çokgenleri kırpmak için 2D kırpma algoritmasını geliştirmiştir. Bu algoritma, uygulandığı şekle göre tamsayı ve kayan nokta hesaplamalarını işleyebildiği ve daha fazla uygulama içermektedir.

Day, kırpma işlemi için yeni bir algoritma sunmaktadır (Day, 1992). Diğer algoritmalara kıyasla bu algoritma, kırpılan çizginin tamamen kırpma penceresinde bulunduğu durumlarda daha az sayıda işlem kullanır. Algoritma ayrıca daha küçük pencere boyutlarıyla daha iyi sonuç vermektedir.

Moller T. kesişen iki üçgen için yeni bir algoritma geliştirmiştir (Möller, 1997). Bu algoritma, üçgenler için düzlem denklemlerini hesaplar ve iki üçgeninin kesiştiği veya ortak düzlemde olup olmadığını test etmek için kullanılmaktadır.

Greiner ve Hormann, 2D keyfi kapalı çokgenleri kırpmak için bir algoritma sunmuşlardır (Greiner ve Hormann, 1998). Algoritma, Weiler ve Atherton ve Vatti

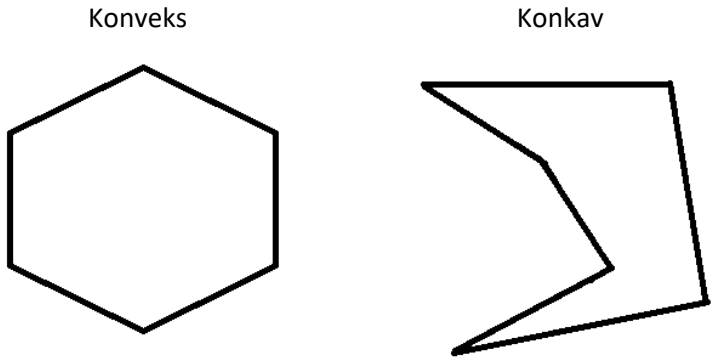
algoritmaları gibi keyfi çokgenlerin genel durumunu ele alır (Liu, Wang, Bao, Gomboši, ve Žalik, 2007).

Huang ve Liu, genel çokgenleri kırpan yeni bir çizgi kırpma algoritması tanımlamayı başarmışlardır (Huang ve Liu, 2002). Bu algoritma önceki kırpma algoritmalarına kıyasla hesap sayısının azalması nedeniyle daha hızlı ve işlemeden kaynaklanan kırpmayı basitleştirmektedir.

1.5.1. Konveks ve Konkav Parseller İçin Yöntemler

Bu başlık altında hem konveks hem de konkav (Şekil 1.5.) parseller için geçerli olan, Işın Kestirme (Ray Intersection), Açıların Toplamı (Sum of Angles) ve Şerit (Swath) yöntemleri anlatılmaktadır. Yöntemlerin oluşumu verilip, ayrıca karşılaşılan zorluklara değinilecektir.

Nokta parsel sorgulamasında; keyfi bir Q noktasının konumu ve verilen bir P parselinin köşe noktalarının koordinatları ile bellidir. P parselinin köşe noktaları (P_1,P_2,...,P_n) ve Q noktası (x, y) koordinatı olarak verilmektedir. Nokta parsel sorgulamasında; eğer Q noktası P parselinin köşeleri içinde kalıyorsa Q noktası P parselinin içindedir, aksi durumda Q noktası P parselinin dışında kalmaktadır.

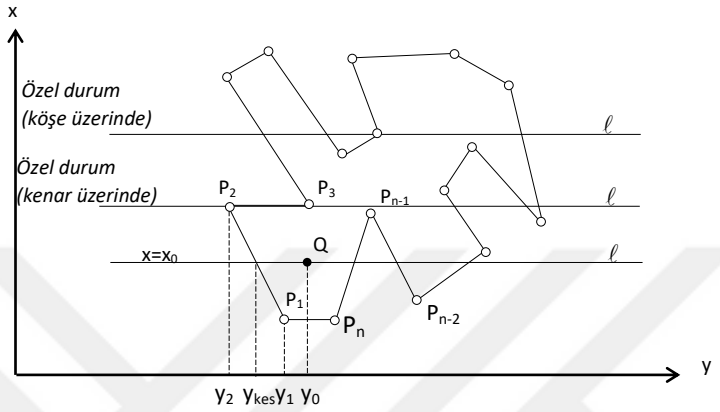


Şekil 1.5. Konveks ve konkav şekiller

1.5.1.1. Işın Kestirme Yöntemi

Işın kestirme yönteminde; Q noktasından geçen bir doğru çizilir. Bu doğru genellikle koordinat eksenlerinden birine paralel olarak seçilir (Şekil 1.6.). Çizilen bu doğru ile parsel

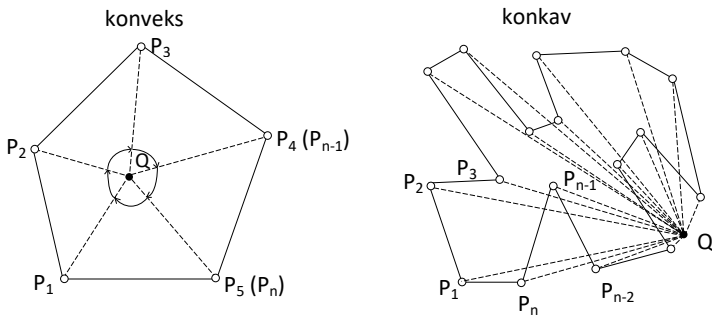
kenarlarının kesişim sayıları hesaplanır. Bu sayı tek ise Q noktası parselin içinde, çift ise Q noktası parselin dışında kalmaktadır. Bu yöntemde oluşturulan algoritmalar bazı özel durumlarda geçerli değildir. Bu özel durumlar; Q' dan geçen doğrunun parsel köşe noktalarından geçmesi veya parsel kenarlarından birinin üzerinde olması durumudur.



Şekil 1.6. Işın kestirme yöntemi

1.5.1.2. Açıların Toplamı Yöntemi

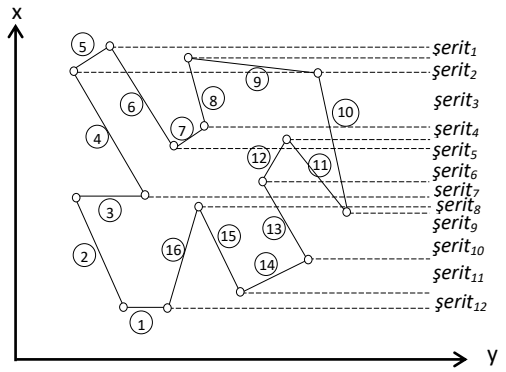
Q noktası ile parsel köşeleri birleştirilir. Böylece ardışık köşeler kullanılarak üçgenler (QP_iP_{i+1}) elde edilir. Oluşan tüm üçgenlerin Q noktasındaki iç açıları hesaplanır. Bu açıların toplamı 360° ise Q noktası parselin içindedir, aksi durumda Q noktası parselin dışındadır (Şekil 1.7.). Hesaplanan açının negatif veya 180° den büyük çıkması özel durumlardır. Açının $+180^\circ$ den büyük olması (konveks ve konkav parsellerin her ikisinde de) durumunda açı 360° den çıkartılır. Açının -180° den büyük olması durumunda; konveks parsellerde açı mutlak değerce hesaba katılır, konkav poligonlarda ise açı işaretiyle beraber hesaba katılır. Açının -180° den küçük olması (konveks ve konkav parsellerin her ikisinde de) durumunda ise açıya 360° eklenir (Kaya ve Yıldırım, 1999). Bu yöntem, Işın kestirme yöntemine göre sonuca daha uzun bir zamanda ulaşır. Fakat ışın kestirme yöntemindeki özel durumlar, bu yöntem için bir problem teşkil etmemektedir.



Şekil 1.7. Açıların Toplamı Yöntemi

1.5.1.3. Şerit Yöntemi

Şerit yönteminde; parselin tüm köşe noktalarından y eksenine paralel olarak çizilen doğrular tarafından parsel şeritlere bölünmektedir (Şekil 1.8.). Sonraki adımda parsel köşe noktalarının, x koordinatları büyükten küçüğe sıralanır ve şeritlerin alt ve üst x koordinatları bulunur. İstenilen Q noktasının x koordinatından yararlanılarak, Q noktasının hangi şeride düştüğü incelenir. Daha sonra Q noktasının bulunduğu şeritteki parsel kenarları bulunur. Bu kenarlara ışın kestirme yöntemi uygulanır ve noktanın parselin içinde olup almadığına bakılır. Bu yöntemde işlem adımlarının artmasına rağmen, ışın kestirme yönteminde problem olan özel durumlar ortaya çıkmayacaktır.



Şekil 1.8. Şerit yöntemi

1.5.1.4. Nokta-Vektör Gösterim Yöntemi

Işın kestirme yönteminin algoritmasında (Şekil 1.6.) gösterilen özel durumlar için (köşe noktasından geçme ve kenar üzerinde olma) yeterli değildir. Nokta-Vektör gösterimi ile ışın belirlenerek bu özel durumları da içeren bir algoritma elde edilmektedir (Kaya ve Yıldırım, 1999).

Düzlemde bir doğrunun birçok gösterim tarzı vardır. Baş (x_0, y_0) ve son (x_1, y_1) noktasının koordinatları ile tanımlı bir L doğru parçasının vektörel formu;

$$L = \{(x_0, y_0) + (r[x_1 - x_0], r[y_1 - y_0] | \ 0 < r < 1\} \tag{1.15}$$

denklemiyle tanımlıdır.

Düzlemde L ve L′ doğru parçaları verilmiştir. L ve L′ doğru parçaları sırasıyla P_0, P_1 ve P_0', P_1' noktaları arasında tanımlanmıştır (Şekil 1.9.). İki doğrunun kesişim problemlerinde doğru parçalarının P_0, P_1, P_0' ve P_1'noktalarının koordinatları bellidir. L ve L′ doğrularının kesişmesi için

$$P_0 + r(P_1 - P_0) = P_0' + r'(P_1' - P_0') \tag{1.16}$$

eşitliğinin sağlanması gerekmektedir. Noktaların koordinat değerlerinden

$$x_0 + r(x_1 - x_0) = x_0' + r'(x_1' - x_0') \tag{1.17}$$
$$y_0 + r(y_1 - y_0) = y_0' + r'(y_1' - y_0') \tag{1.18}$$

denklemleri yazılır. r ve r' değerleri bu denklemden bir tek çözüme sahiptir. Bu değerler aşağıdaki determinantların hesabıyla bulunabilir.

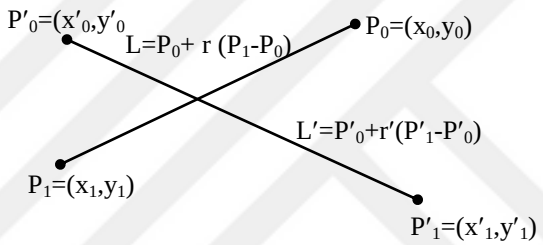
$$D = \begin{vmatrix} (y_1 - y_0) & -(y_1' - y_0') \\ (x_1 - x_0) & -(x_1' - x_0') \end{vmatrix} \tag{1.19}$$

$$D_1 = \begin{vmatrix} (y_0' - y_0) & -(y_1' - y_0') \\ (x_0' - x_0) & -(x_1' - x_0') \end{vmatrix} \tag{1.20}$$

$$D_2 = \begin{vmatrix} (y_1 - y_0) & (y_0' - y_0) \\ (x_1 - x_0) & (x_0' - x_0) \end{vmatrix} \tag{1.21}$$

Buradan $r = D_1/D$ ve $r' = D_2/D$ hesaplanır. r ve r' değerleri [0,1] arasında ise L ve L′ doğru parçaları bir noktada kesişir. Eğer D determinantı sıfıra eşitse L ve L′ doğru parçaları paraleldir. Eğer D, D_1 ve D_2 determinantları sıfıra eşitse L ve L′ doğru parçaları çakışıktır (üst üste). Programlama açısından paydanın sıfır olma ihtimali olduğundan bölme işleminden kurtulmak için sadeleştirmeye gidilebilir. Bunun için; D determinantı sıfıra eşit değil ve r sayısı [0,1] aralığında değer alıyorsa $r(1-r) = (D_1 D - D_1^2)/D^2$ değeri her zaman pozitiftir. Payda D^2 her durumda pozitif olduğundan $(D_1 D - D_1^2)$ değeri de pozitiftir.

Dolayısıyla $D_1(D - D_1)$ ve $D_2(D - D_2)$ değerleri negatif değilse doğru parçaları kesişir.



Şekil 1.9. L ve L′ doğru parçaları

Tüm parsel kenarlarıyla yapılacak kesişimleri sorgulamak için, ışınında vektörel olarak ilk ve son koordinatları belli bir doğru parçası olarak tanımlanması gerekir. Sorgulama için verilen Q noktasının (x_0, y_0) koordinatları ışının ilk noktasıdır. Işın vektörel olarak

$$R = \{(x_0, y_0) + r(m_x, 2M_y) | r > 0\} \tag{1.22}$$

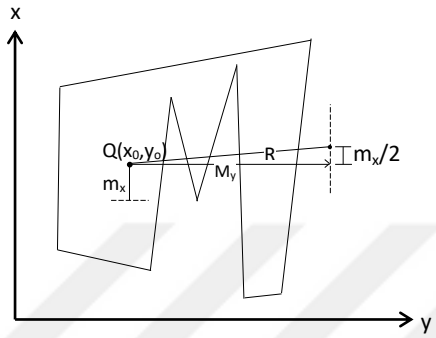
tanımlanırsa ışının son nokta koordinatları $r = 1$ olmak üzere

$$(x', y') = (x_0 + m_x, y_0 + 2M_y) \tag{1.23}$$

elde edilir. Buradaki M_y ve m_x ışına belli bir eğim vermek için kullanılır. Böylece ışın (Şekil 1.6.)'da ki özel durumların hiç biri gibi olmayacaktır. M_y için $|y_0 - y_i|$ mutlak

değer farkı en büyük değer alınır. m_x için $|x_0 - x_i|$ mutlak değer farkı en küçük olan sıfırdan farklı değer alınır (Şekil 1.10.).

Nokta parsel sorgulamasında (x_0, y_0) dan (x', y') ye kadar olan R ışını, parselin tüm S_i kenarları sorgulanarak kesişme sayısı bulunur. Kesişimlerin sayısı;



Şekil 1.10. M_y ve m_x Işınları

$$D_i'(D_i - D_i') \tag{1.24}$$

$$D_i'' D_i \tag{1.25}$$

determinant çarpımlarının hesaplanmasıyla elde edilir. Burada ki değerler;

$$D_i = \begin{vmatrix} 2M_y & -(y_{i+1} - y_i) \\ m_x & -(x_{i+1} - x_i) \end{vmatrix} \tag{1.26}$$

$$D_i' = \begin{vmatrix} 2M_y & (y_i - y_0) \\ m_x & (x_i - x_0) \end{vmatrix} \tag{1.27}$$

$$D_i'' = \begin{vmatrix} (y_i - y_0) & -(y_{i+1} - y_i) \\ (x_i - x_0) & -(x_{i+1} - x_i) \end{vmatrix} \tag{1.28}$$

determinantlarıyla hesaplanır. Eğer (1.24) ve (1.25) pozitifse R ışını parselin S_i kenarını keser. Diğer durumlar da kesmez. Böylece kesişim sayısı hesaplanır. Bu sayı tek ise Q noktası parselin içinde, çift ise parselin dışındadır.

1.5.1.5. Matlab İnpolygon Fonksiyonu

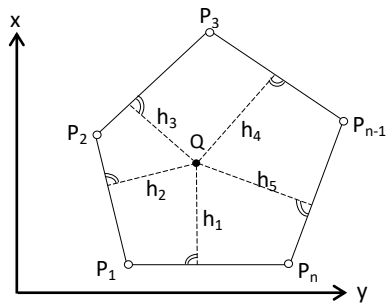
Matlab programında inpolygon fonksiyonu yardımıyla verilen noktaların, bir bölgenin içinde veya kenarları üzerinde olup olmadığını irdeleyen komuttur. Bu fonksiyon için irdelenecek bölgenin köşe koordinatları ve irdelenecek noktaların koordinatları verilmesi yeterlidir.

1.5.2. Konveks Parseller İçin Yöntemler

Bu başlık altında konveks parseller için; İşaret Karşılaştırma (Sign of Offset), Alan Toplama (Sum of Area) ve Dönüş Yönü (Orientation) yöntemleri irdelencektir. Yöntemlerin adımları verilip, ayrıca karşılaşılan zorluklara değinilecektir.

1.5.2.1. İşaret Karşılaştırma Yöntemi

Q noktasının parselin tüm ardışık (P_iP_{i+1}) kenarlarına olan mesafeleri (h) hesaplanır. Bütün bu mesafeler aynı işarete sahipse Q noktası parselin içindedir. Aksi taktirde (h mesafeleri işaret değişikliğine uğruyorsa) Q noktası parselin dışındadır (Şekil 1.11.).

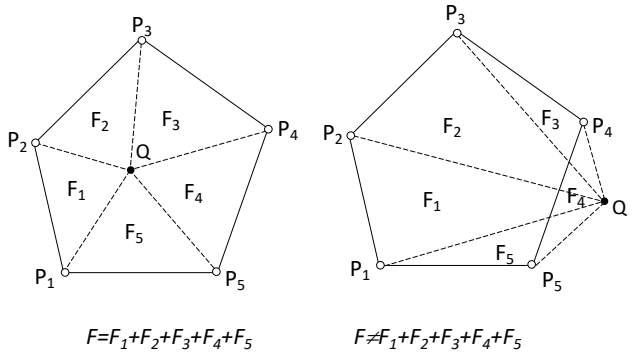


Şekil 1.11. İşaret karşılaştırma yöntemi

1.5.2.2. Alan Toplama Yöntemi

Parselin her bir köşe noktası ile Q noktası birleştirilerek üçgenler elde edilir (Şekil 1.12.). Bu üçgenlerin alanları toplamı parselin alanına eşitse, Q noktası parselin içinde aksi taktirde parselin dışındadır. Bu yöntem programlama açısından bir önceki yönteme göre

daha kullanışlıdır. Çünkü alan hesabında kullanılan matematiksel işlemler, h yüksekliğinin hesabından daha basittir.



Şekil 1.12. Alan toplama yöntemi

1.5.2.3. Dönüş Yönü Yöntemi

Q noktası, parselin her bir köşe noktası ile birleştirilerek üçgenler elde edilir. Bu üçgenlerin QP_iP_{i+1} köşelerinin dönüş yönüne (işaretine) bakılır. İşaret değişikliği yoksa Q noktası parselin içindedir, aksi durumda Q noktası parselin dışındadır.

Bu yöntem ve alan toplama yönteminin her ikisinde de alan hesaplaması yapılmasına rağmen bu yöntem daha kullanışlıdır. Çünkü alan toplama yönteminde alanlar hesaplanarak toplanmaktadır, bu yöntemde alanları toplamaya gerek duymadan sadece işarete bakılır.

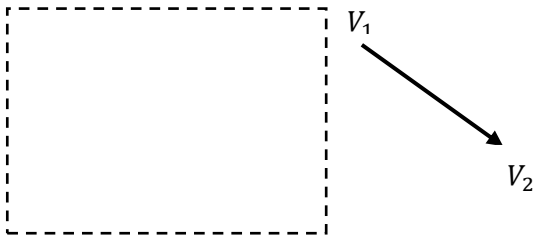
1.6. Kesişen Alanın Köşe Noktalarıyla Parsel Oluşturma Yöntemleri

İki parselin çakışması sonucu ortaya çıkan alanın, köşe noktaları karışık şekilde olup saat ibresi yönünde sıralanmamaktadır. Bu karışık noktaların kesişen alanın yüz ölçümü hesabı için saat ibresi yönünde sırayla yeniden nokta numarası verilmesi gerekmektedir. Bu işlemin geometrik algoritması için farklı yöntemler mevcut olup aşağıda incelenmiştir. Bu yöntemlerin hem konkav hem de konveks şekiller için algoritma oluşturması temel hedeftir.

1.6.1. Sutherland – Hodgman Yöntemi

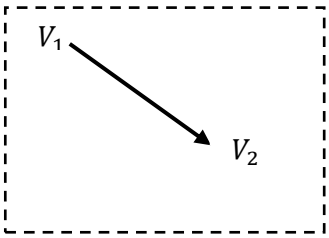
Sutherland–Hodgman yöntemi, herhangi bir çokgen şekil ile dışbükey bir çokgen şeklin çakıştırılması sonucu oluşan bölgenin köşe koordinatlarını bulmaktadır. Yani dışbükey şeklin (genelde dikdörtgen) yardımıyla kırpma işlemi yapmaktadır. Bu yöntemin dezavantajı kırpma işlemini yapacak olan şeklin sadece dışbükey bir şekil olarak seçilebilmesidir.

Sutherland–Hodgman yönteminin temelinde basit bir algoritma yatmaktadır. Algoritmayı açıklamadan önce, algoritmaya altlık olacak bazı bilgilerden bahsedilmelidir. Bu bilgilerin başında sorgulanacak noktaların birbirlerine göre durumlarıdır.



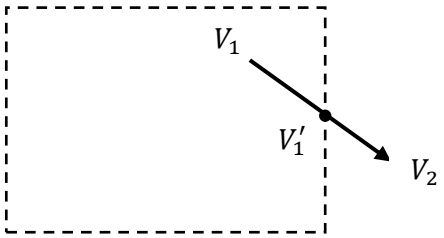
Şekil 1.13. Sutherland–Hodgman her iki noktanın dışarda olma durumu

Şekil 1.13. de görüldüğü gibi iki noktada kırpma işlemini yapacak şeklin dışında kalmaktadır. Bundan dolayı bu iki noktada kayıt altına alınmayacaktır.



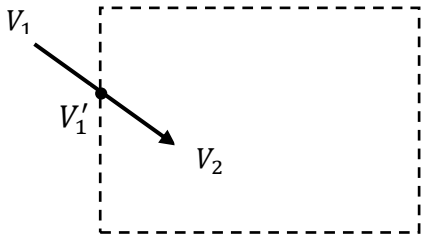
Şekil 1.14. Sutherland–Hodgman her iki noktanın içerde olma durumu

Şekil 1.14. de görüldüğü gibi iki noktada kırpma işlemini yapacak şeklin içinde kalmaktadır. Bu durumda ikinci olan nokta (V_2) kayıt altına alınmaktadır.



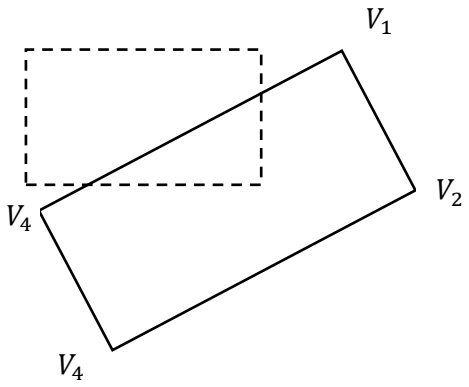
Şekil 1.15. Sutherland–Hodgman birinci noktanın içerde, ikinci noktanın dışarda olma durumu

Şekil 1.15. de görüldüğü gibi noktalardan birincisi kırpma işlemini yapacak şeklin içinde kalırken diğeri dışında kalmaktadır. Bu durumda bu iki nokta ile oluşturulan doğru ile bölgenin kenarı kesişmektedir. Kesişen bu noktaya bir isim verip (V_1'), bu nokta kayıt altına alınır.



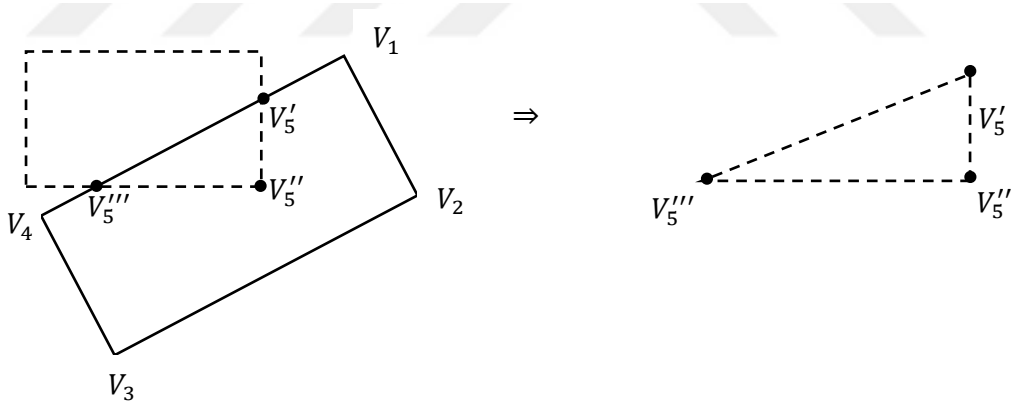
Şekil 1.16. Sutherland–Hodgman birinci noktanın dışarda, ikinci noktanın içerde olma durumu

Şekil 1.16. de görüldüğü gibi noktalardan birincisi kırpma işlemini yapacak şeklin dışında kalırken diğeri şeklin içinde kalmaktadır. Bu durumda bu iki nokta ile bölgenin kenarı kesişmektedir. Kesişen bu noktaya bir isim verilip (V_1'), bu nokta ve içerde kalan nokta kayıt altına alınır.



Şekil 1.17. Sutherland–Hodgman kesişen iki parsel

Öncelikle iki bölge çakıştırılır (Şekil 1.17.) ve dışbükey şeklin herhangi bir kenarı seçilerek başlanır. Seçilen bu kenar diğer şeklin tüm kenarları ile yukarıda belirtilen 4 durum yardımıyla incelenir. Eğer bu kenarlar üzerinde şekillerin kesiştiği nokta varsa yeni bir nokta olarak adlandırılır. Bu adımlar saat yönünde olmak üzere dışbükey şeklin tüm kenarlarına uygulanır. En sonunda oluşan yeni noktalar ve köşe noktaları yardımıyla ortak (kırpılmış) bölge elde edilmektedir (Şekil 1.18.).



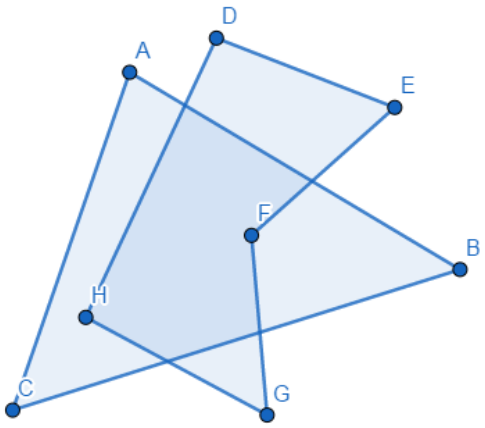
Şekil 1.18. Sutherland–Hodgman kesişme bölgesi

1.6.2. Weiler–Atherton Yöntemi

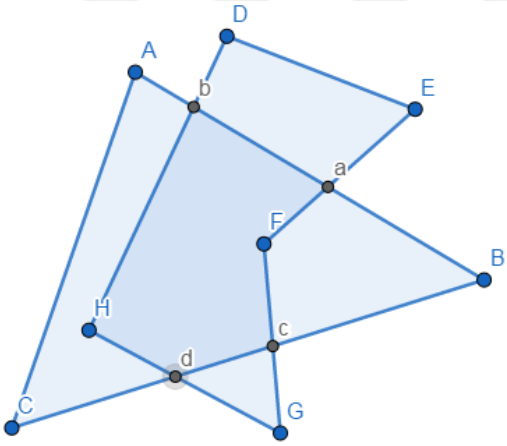
Weiler-Atherton yöntemi kesişen iki çokgenin (içbükey veya dışbükey) kesiştirilmesi sonucu oluşan ortak bölgeyi bulup, o bölge üzerinde yapılacak işlemlere (alan hesabı, nokta sorgulama, vb.) imkân sağlar. Bu yöntemin, Sutherland-Hodgman yönteminden avantajlı olmasının nedeni, alınan iki şeklinde herhangi bir çokgen olabilir olmasıdır.

Öncelikle iki kesişen bölge alınır (Şekil 1.19.). İki bölgenin de köşe noktalarının isimleri ayrı başlıklar altında saat yönündeki sıra ile yazılmalıdır. Sonrasında şekillerden herhangi birini seçerek, ardışık köşeleri yardımıyla her bir kenarının doğru denklemleri elde edilir. Aynı işlem diğer bölgeye de uygulanıp tüm kenarlarının doğru denklemleri elde edilir. Sonrasında sırası ile bu kenarların doğru denklemlerinin her birinin kesişip kesişmediği kontrol edilir. Eğer kesişen doğrular varsa, kesiştikleri noktaya yeni bir isim verilir ve bu nokta hangi kenarlar arasında ise sıralı şekilde yazılan köşe noktalarının olduğu tabloda o iki köşe noktasının arasına eklenir. Bu işlem tüm kenarlar için yapılır ve sıralama en son halini alır. Oluşturulan bu sıralamada soldan başlayarak ilk kesişen noktaya kadar ilerlenir. Bu nokta, ortak bölgenin ilk noktası olarak alınır ve diğer kesişim noktasına kadar olan aradaki tüm noktalar ortak bölgenin kenarları olarak yazılır. İkinci kesişim noktasına gelindiğinde durulur. Eğer ikinci kesişim noktası başlangıçtaki sıralamada varsa o sıraya geçer ve aynı işleme orda devam edilir. Bu süreç ortak bölgenin ilk noktası olarak alınan ilk noktaya gelene kadar devam eder. Böylece ortak bölgenin tüm köşe koordinatları bulunmuş olur. Bu anlatılanlara örnek aşağıda verildiği gibidir.

1.Bölge A → B → C, 2.Bölge D → E → F → G → H olarak tüm kenarları saat dönüş yönünde sıralanarak yazılır. 1.bölgede ilk kenar olan AB kenarı ile ikinci şeklin tüm kenarlarının kesişip kesişmediğine bakılır. AB kenarı ile ikinci bölgenin EF ve HD kenarları kesişmektedir. Kesişen o noktalara sırasıyla a ve b isimleri verilir. Sonrasında aynı adımlar BC kenarına uygulanır. Bu kenar ile FG kenarı ve GH kenarı kesişmektedir. Bulunan bu noktalara sırası ile c ve d isimleri verilir. Son olarak CA kenarına bakılır ve kesişim noktasının olmadığı görülür. Kesişim noktaları bulunma adımları sonrası, şekil en son halini (Şekil 1.20.) almıştır.



Şekil 1.19. Weiler-Atherton kesişen iki parsel



Şekil 1.20. Weiler-Atherton kesişim noktaları

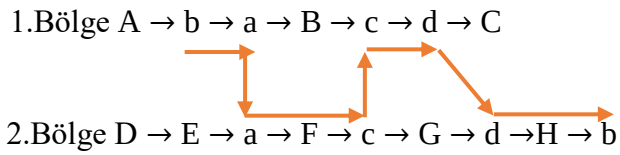
Şimdi kesişim noktaları sıralamaya dâhil edilmelidir. Buradan

1.Bölge A → b → a → B → c → d → C

2.Bölge D → E → a → F → c → G → d →H → b

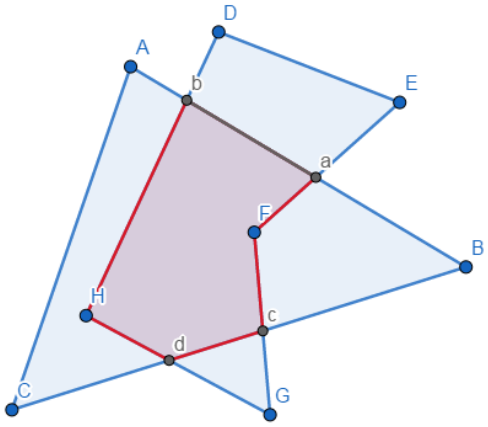
elde edilir. Sonrasında 1.bölge sıralamasında ilk kesişim noktası olan b noktasından başlanır ve ikinci kesişim noktası a ya kadar devam edilir. a noktası 2.bölgedeki sıralamada bulunur ve oradan c ye kadar devam eder. Eğer c noktası 1.bölgedeki sıralamada olmasaydı orda durulacaktı. Ancak c noktası 1. bölgedeki sıralamada mevcuttur. Devamında 1.bölgedeki c noktasına gidilip bir sonraki kesişim noktası olan d noktasına kadar devam edilir. Aynı şekilde d noktası 2.bölgedeki sıralamada mevcuttur. Son olarak

2.bölgedeki sıralamadan d noktasından sonraki kesişim noktası b noktasıdır. Bu nokta başlangıçta alınan noktadır. Yani ortak bölgenin köşe koordinatları bulunmuş olur.



Şekil 1.21. Weiler-Atherton kesişen bölge oluşturma

Bu noktalar birleştirildiği zaman ortak bölge oluşmaktadır (Şekil 1.22.).

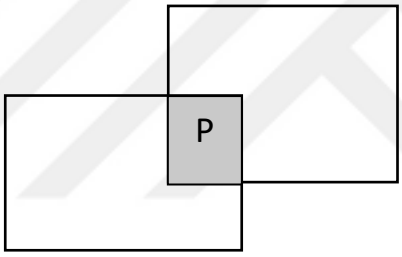


Şekil 1.22. Weiler-Atherton kesişme bölgesi en son halinin gösterimi

2. YAPILAN ÇALIŞMALAR

Bu çalışmadaki amaç, konveks veya konkav iki parselin çakışması sonucu oluşan ortak bölgenin köşe koordinatlarını ve alan hesabını Matlab programı yardımıyla elde etmektir. Oluşan bu ortak bölgenin alanı hesaplanırken, yazılan algoritmalar sayesinde sadece iki parselin koordinatları verilmesi yeterli olacaktır.

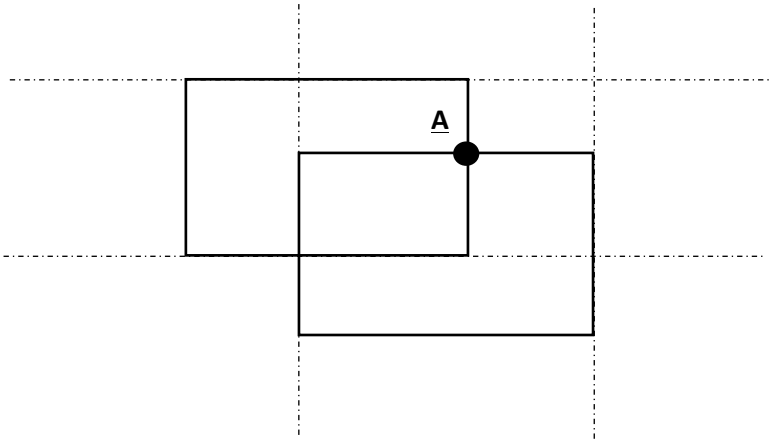
Öncelikle bu iki parselin çakışması sonucu oluşacak kesişim parselinin köşe koordinatlarının hesaplanması gereklidir. Bu parselin köşe koordinatlarını bulmak için alınan iki parselinde tüm kenarlarının doğru denklemlerini yazmak gerekmektedir. Amacımız yazılan her doğru denkleminin birbirleri ile kesişim noktalarının bulunmasıdır. Çünkü bu kesişim noktalarından bazıları, kesişim parselinin (Şekil 2.1.) köşe koordinatlarıdır.



Şekil 2.1. Kesişen iki parsel

Köşe koordinatlarını bulmak için geometrik olarak (1.13) ve (1.14) iki doğrunun kesişim noktası formülleri kullanılır. Ayıca, elde edilen kesişim noktalarının hangisinin, kesişim parselinin köşe koordinatları olduğu algoritma olarak geliştirilmesi gerekmektedir. Çünkü iki doğrunun kesişim noktası bulunurken bu iki doğrudan herhangi biri, koordinatı belli olan parselin diğer kenarları ile kesişebilir. Bu problem; hangi iki doğru kesiştiriliyorsa, o iki doğrunun koordinatları yardımıyla her bir doğru için oluşturulan dikdörtgen şeklinin dışındaki kesişim noktaları alınmamaktadır. (Şekil 2.2.)

Kesişim parselinin köşe koordinatlarının belirlenmesinde bir diğer adım ise, kesişim noktaları haricinde kesişim parselinin diğer noktalarının parsel köşe noktaları olup olmadığı sorgulanmasıdır.

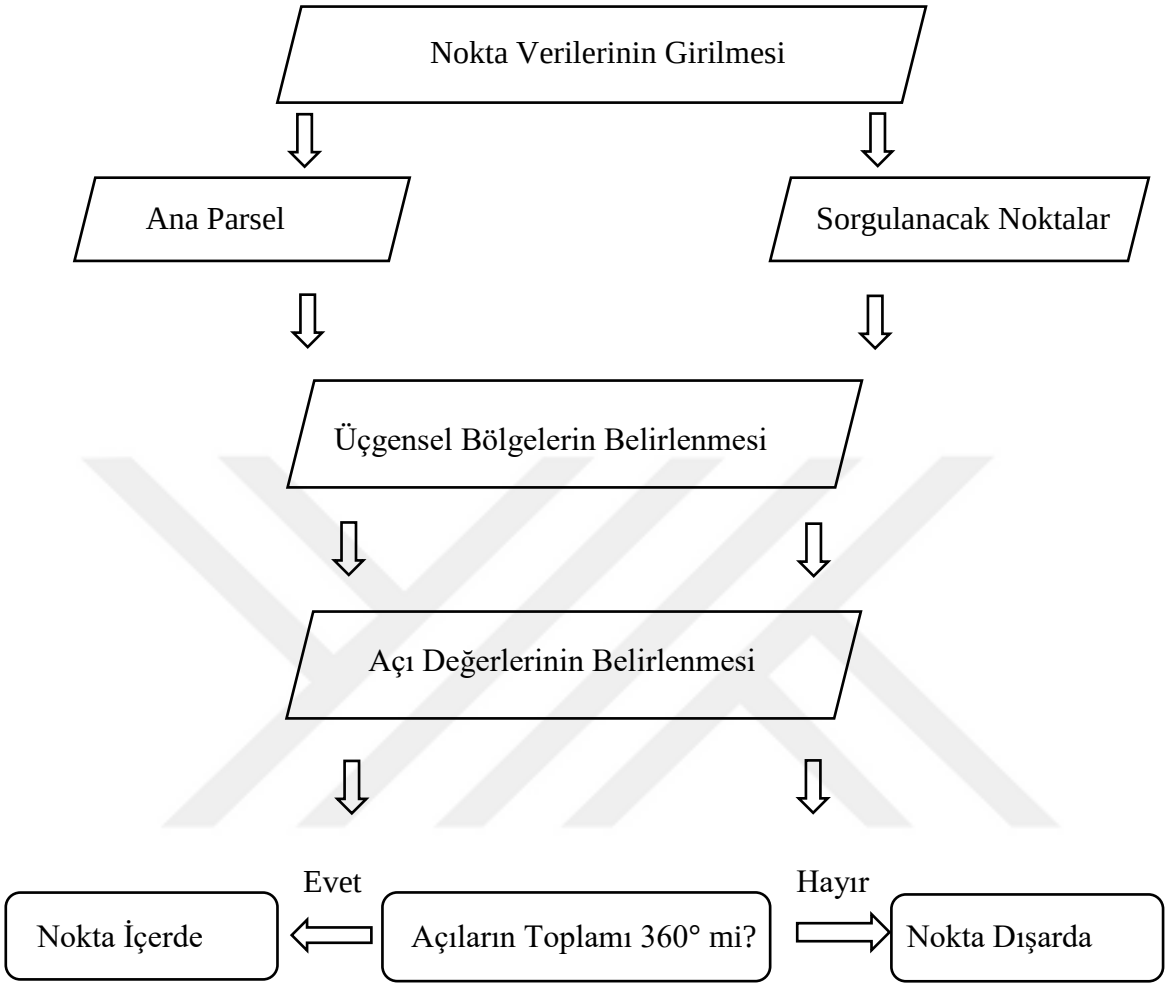


Şekil 2.2. Kesişim noktasının seçimi

2.1. Kullanılan Yöntemler

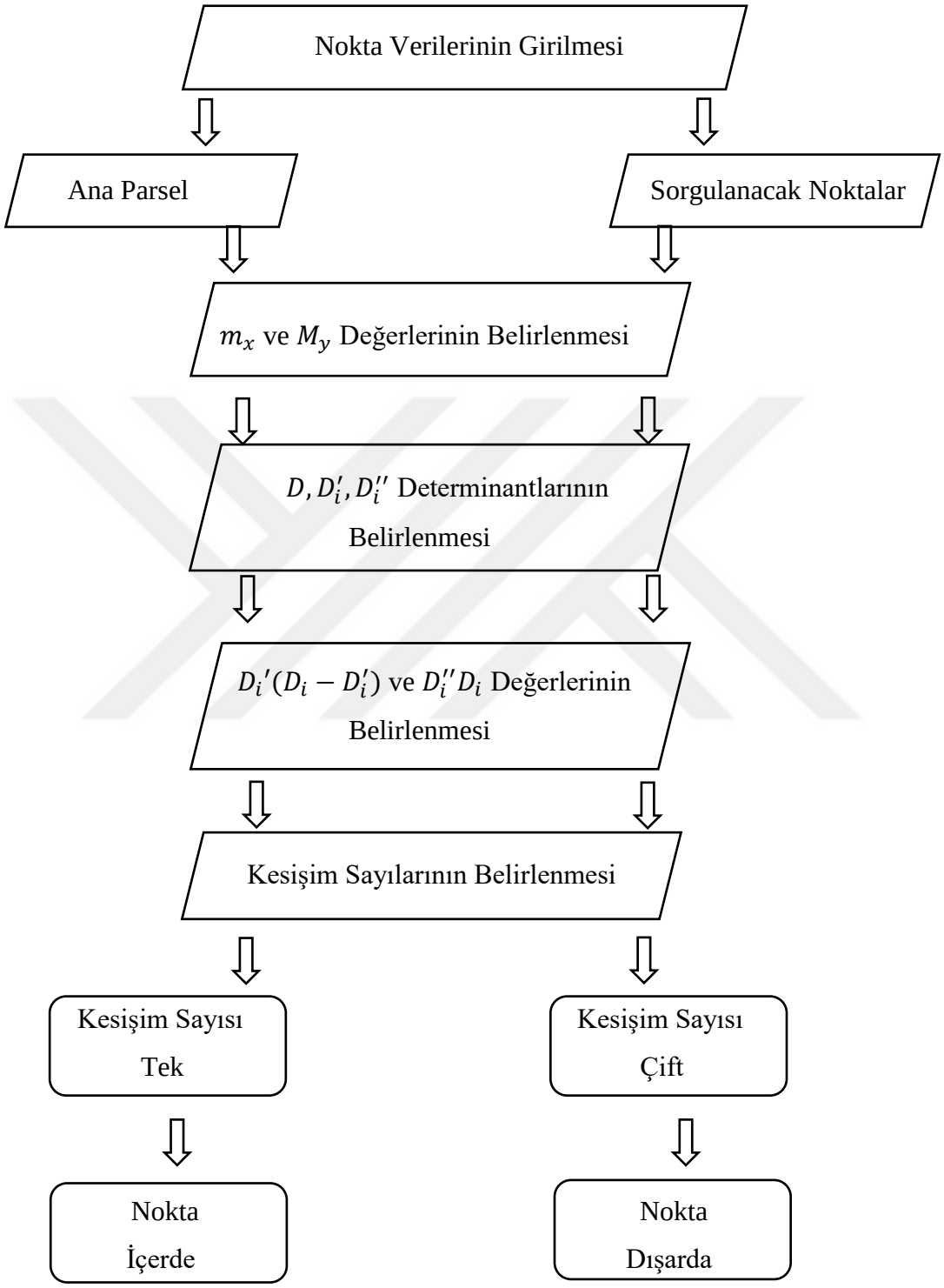
Kesişim parselinin köşe koordinatlarını ve alanını bulmak için ilk adım; parsellerin köşe noktalarının birbiri içinde olup olmadığı durumdur. Bunun için Matlab yardımıyla yazılan üç farklı algoritma oluşturulmuştur. Bu algoritmalar oluşturulurken; açıların toplamı yöntemi, nokta-vektör yöntemi ve Matlab programında bulunan inpolygon fonksiyonu kullanılmıştır. Bu üç yöntemde, parsellerin kenar noktalarının birbirlerine göre durumlarını incelemektedir. Bu yöntemlerin Matlab programı yardımıyla yazılan kodları verilmektedir (Ek 1., Ek 2., Ek 3. ve Ek 4.).

2.1.1. Açıların Toplamı Yöntemi Algoritması



Şekil 2.3. Açıların toplamı yöntemi iş akış şeması

2.1.2. Nokta - Vektör Gösterim Yöntemi Algoritması



Şekil 2.4. Nokta-Vektör yöntemi iş akış şeması

Yukarıda bahsedilen algoritmalar sonucunda aşağıdaki örnek verilmiştir. Bu örnekte ana parsel ve sorgulanacak noktalar alınmıştır. Sorgulanacak noktaların, ana parselin içinde olup olmadığı yukarıda verilen algoritmalar yardımıyla irdelenecektir.

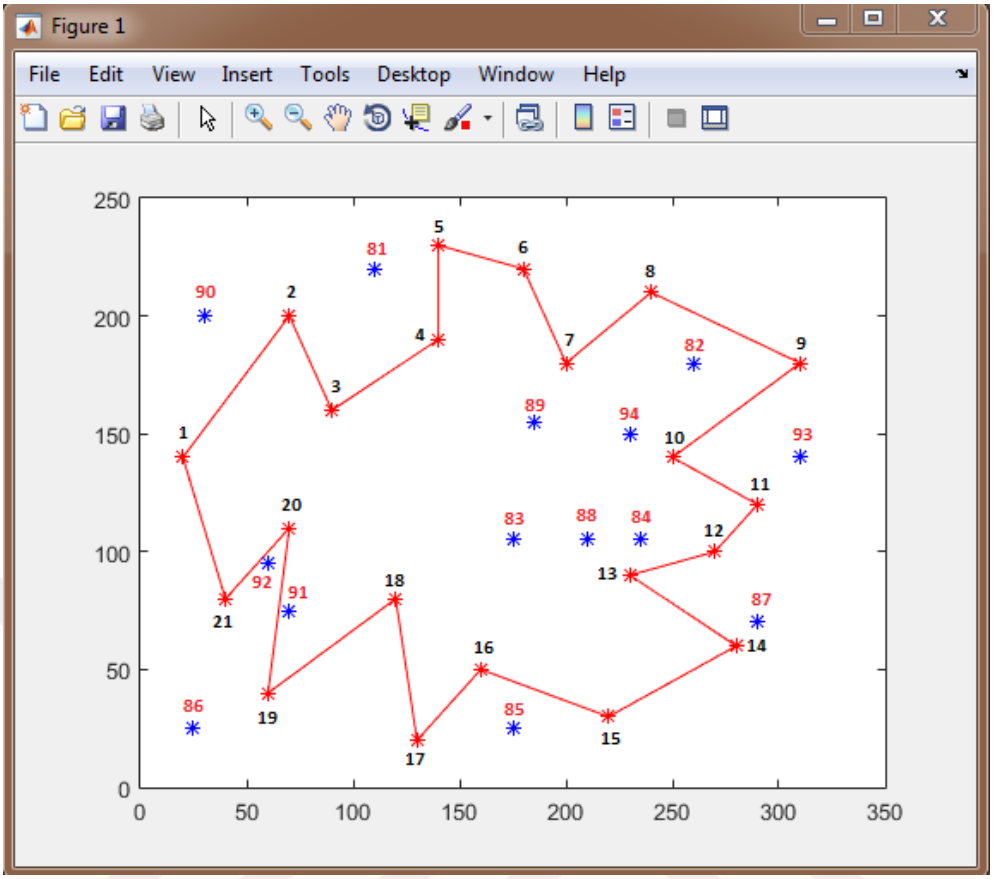
Tablo 2.1. Ana parselin köşe noktalarının koordinatları

ANA PARSEL					
N.N	**Y(m)**	**X(m)**	**N.N**	**Y(m)**	**X(m)**
1	20	140	12	270	100
2	70	200	13	230	90
3	90	160	14	280	60
4	140	190	15	220	30
5	140	230	16	160	50
6	180	220	17	130	20
7	200	180	18	120	80
8	240	210	19	60	40
9	310	180	20	70	110
10	250	140	21	40	80
11	290	120			

Tablo 2.2. Sorgulanacak noktaların koordinatları

SORGULANACAK NOKTALAR		
N.N	Y(m)	X(m)
81	110	220
82	260	180
83	175	105
84	235	105
85	175	25
86	25	25
87	290	70
88	210	105
89	185	155
90	30	200
91	70	75
92	60	95
93	310	140
94	230	150

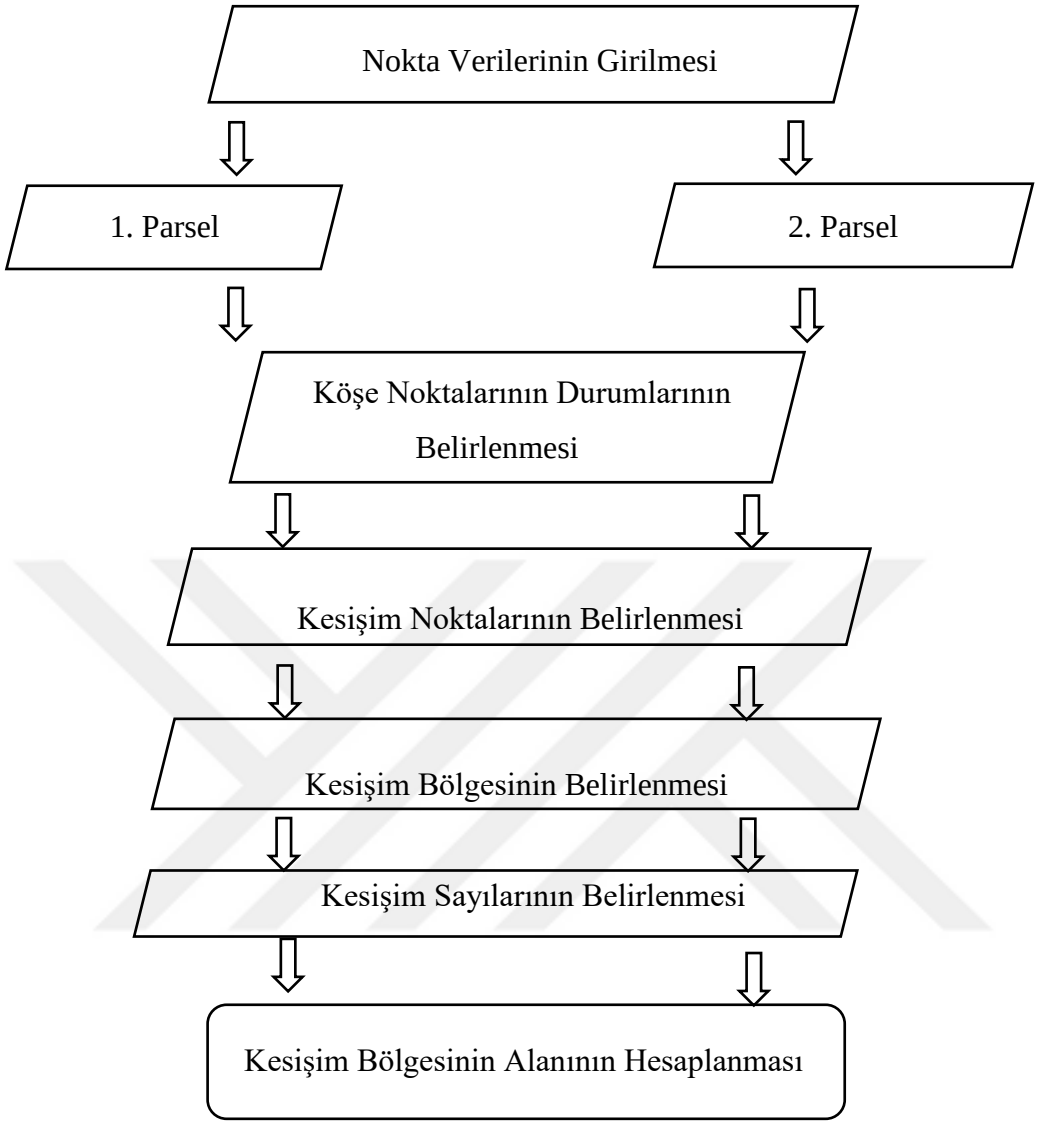
Yukarıdaki verilen ana parselin ve sorgulanacak noktalar Şekil 2.4.'de gösterilmiştir.



Şekil 2.5. Ana parsel ve sorgulanacak noktaları

Sorgulanacak noktaların, ana parselin içinde olup olmadığını yukarıda bahsedilen farklı üç yöntem ile irdelenmiştir. Bu sorgulama yöntemlerin işlem zamanlarıyla yapılmıştır.

Bu algoritmayla kesişim parselinin saat ibresi yönünde köşe koordinatlarının belirlenmiş ve alanı hesaplanmıştır. Köşe koordinatları ve alan hesabında birden fazla sorgulanacak parseller seçilmiştir. Bunun amacı özel durumların irdelenmesidir. Bu özel durumlar aşağıda detaylıca irdelenmiştir. Bulunan alanlar ve köşe koordinatları CBS yazılımlarıyla kontrol edilerek avantaj ve dezavantajları irdelenmiştir.

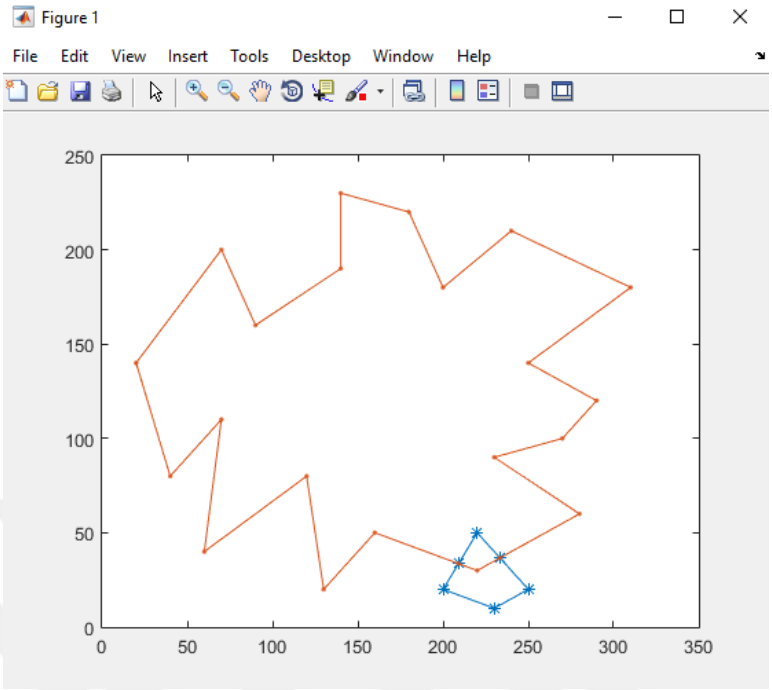


Şekil 2.6. Alan hesabı iş akış şeması

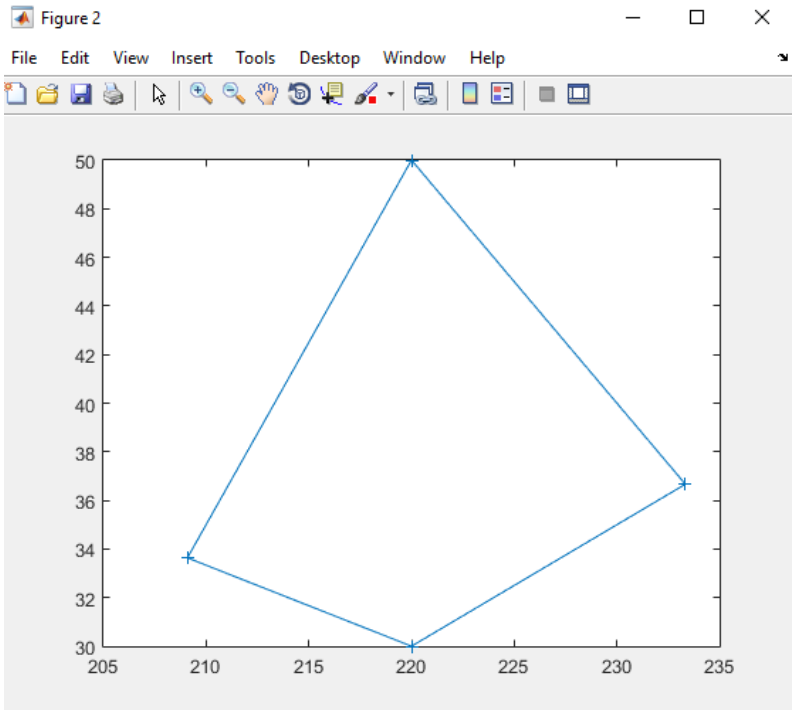
Durum 1 Ana parseli iki farklı noktada kesen bir parsel alınmıştır. Şekillerde sırasıyla iki parselin birbirlerine göre durumları ve bu parsellerin kesişmesi sonuncunda oluşan ortak bölge ayrıntılı olarak köşe koordinatları ile gösterilmektedir.

Tablo 2.3. Durum 1 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	**Y (m)**	**X(m)**
A	220	50
B	250	20
C	230	10
D	200	20



Şekil 2.7. Durum 1 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.8. Durum 1 iki parselin kesişim bölgesi

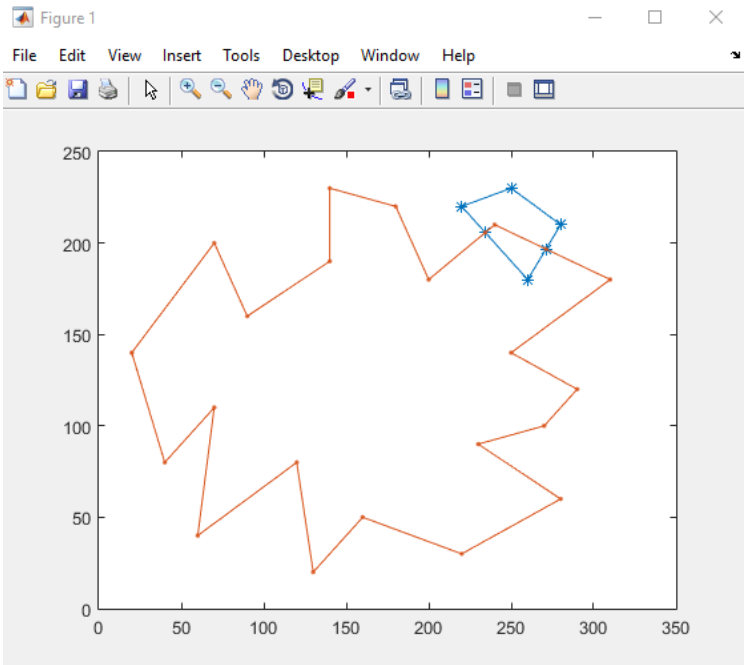
Tablo 2.4. Durum 1 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	233,334	36,667
	209,090	33,637

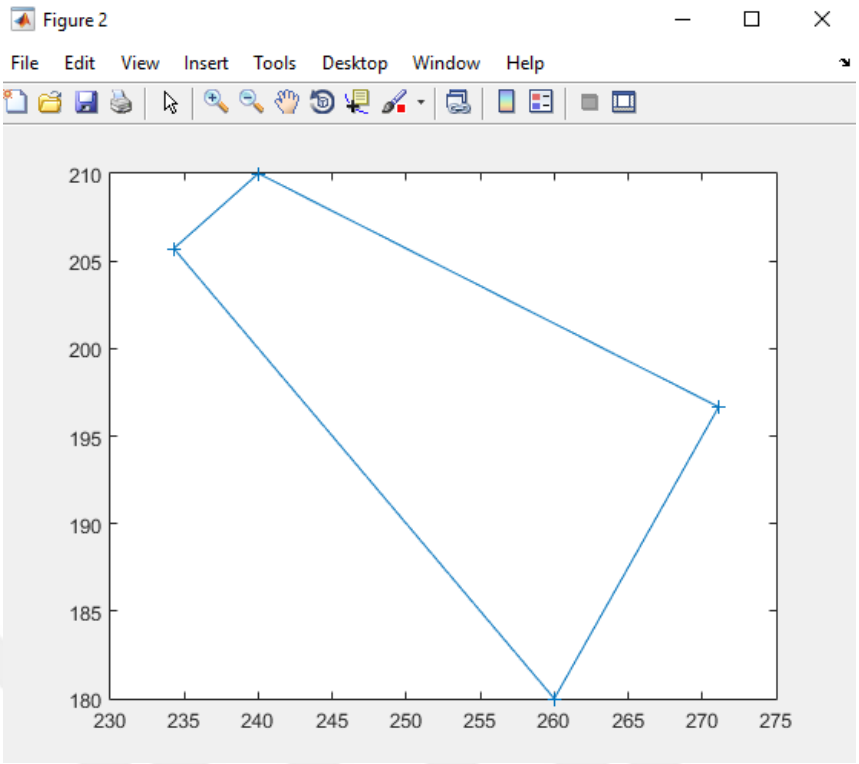
Durum 2 Ana parseli iki farklı noktada kesen bir parsel alınmıştır ve sonrasındaki şekilde her ikisinin kesiştiği bölge ayrıntılı bir şekilde gösterilmiştir.

Tablo 2.5. Durum 2 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X (m)
A	220	220
B	250	230
C	280	210
D	260	180



Şekil 2.9. Durum 2 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.10. Durum 2 iki parselin kesişim bölgesi

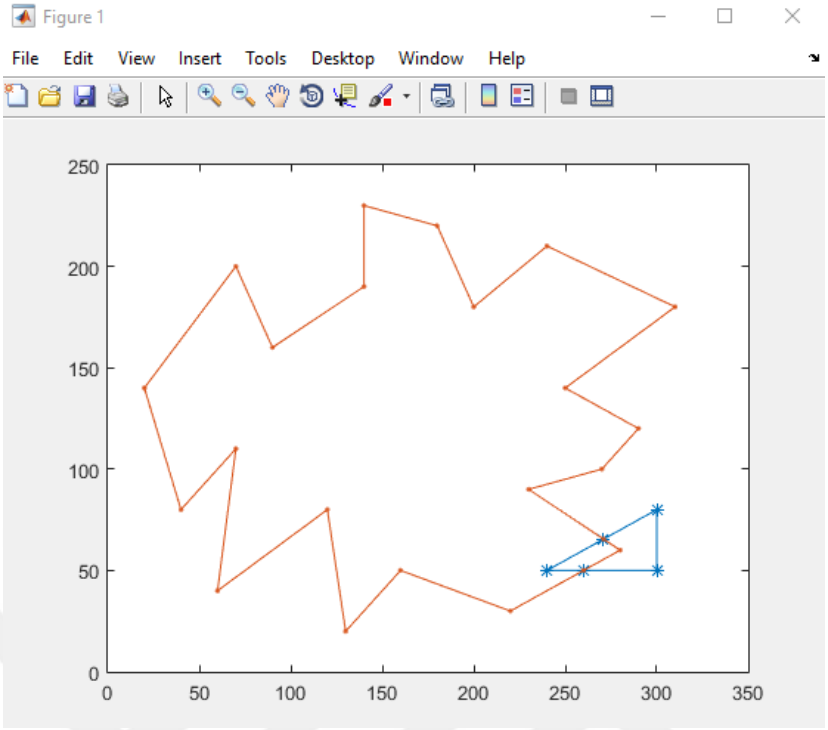
Tablo 2.6. Durum 2 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	234,286	205,714
	271,112	196,667

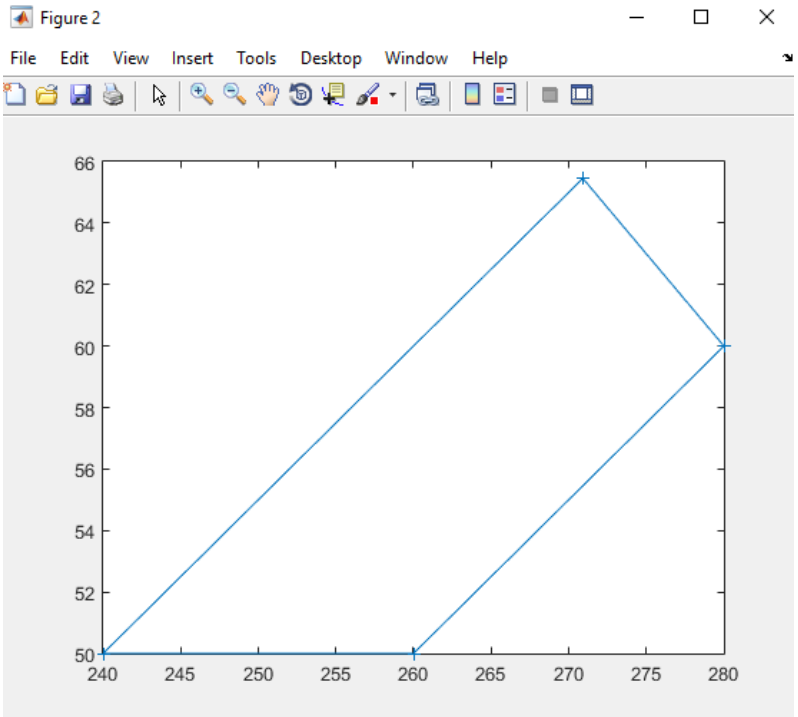
Durum 3 Ana parselin sağ tarafında iki farklı noktada kesişen bir parsel alınmıştır. Ana parseli iki farklı noktada kesen bu parsellerin, ana parselin farklı yerlerinde seçilmesinin sebebi, Weiler-Atherton algoritmasının doğru bir şekilde çalışıp çalışmadığını teyit etmek içindir.

Tablo 2.7. Durum 3 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	300	80
B	300	50
C	240	50



Şekil 2.11. Durum 3 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.12. Durum 3 iki parselin kesişim bölgesi

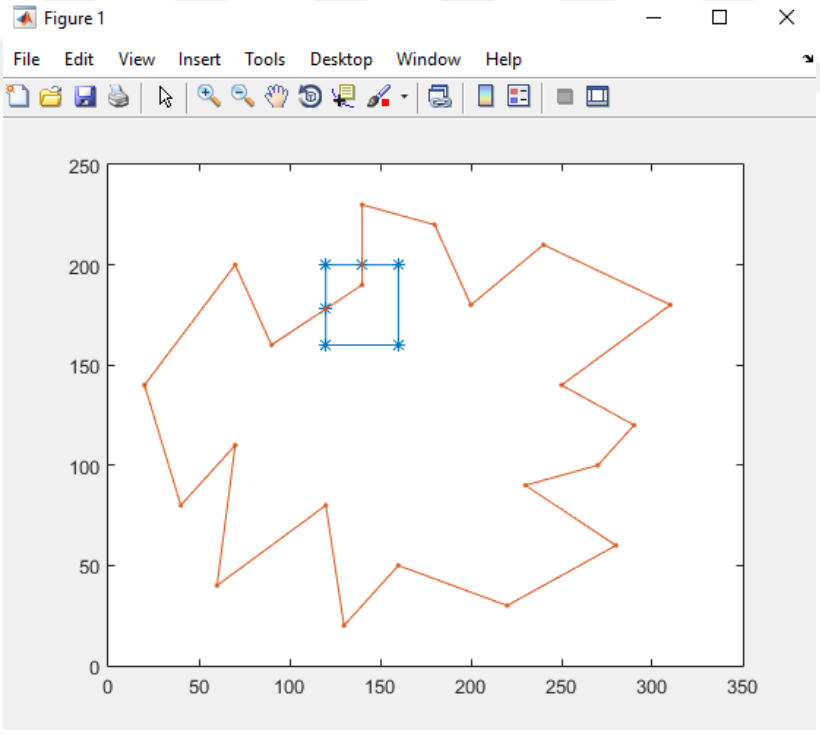
Tablo 2.8. Durum 3 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	270,910	65,455
	260,000	50,000

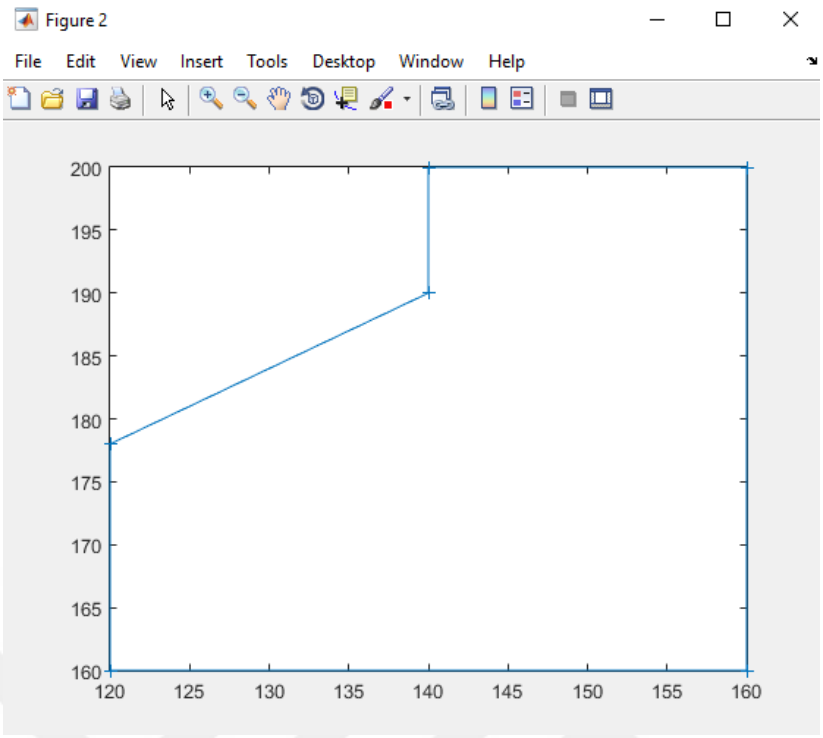
Durum 4 Ana parselin bir kenarı ile sorgulanacak parselin bir kenarı dik kesişmektedir. Bu durum özel bir durum olarak incelenmektedir.

Tablo 2.9. Durum 4 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	120	200
B	160	200
C	160	160
D	120	160



Şekil 2.13. Durum 4 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.14. Durum 4 iki parselin kesişim bölgesi

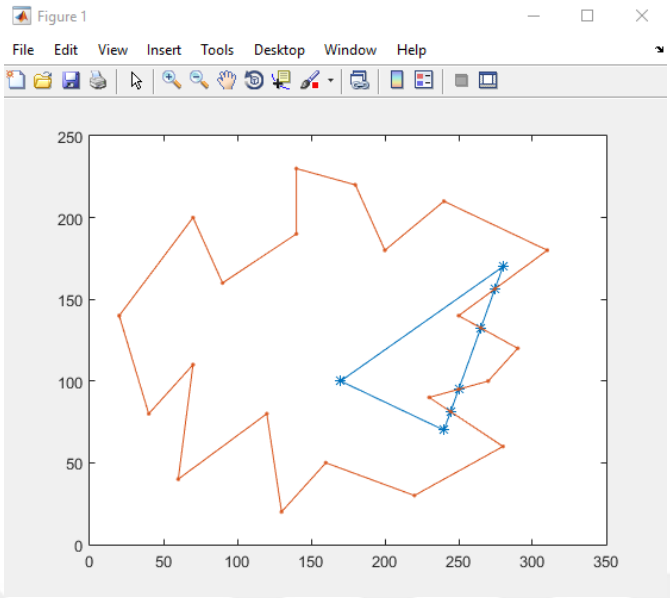
Tablo 2.10. Durum 4 kesişim Noktaları

Kesişim Noktaları	**Y(m)**	**X(m)**
	120	178
	140	200

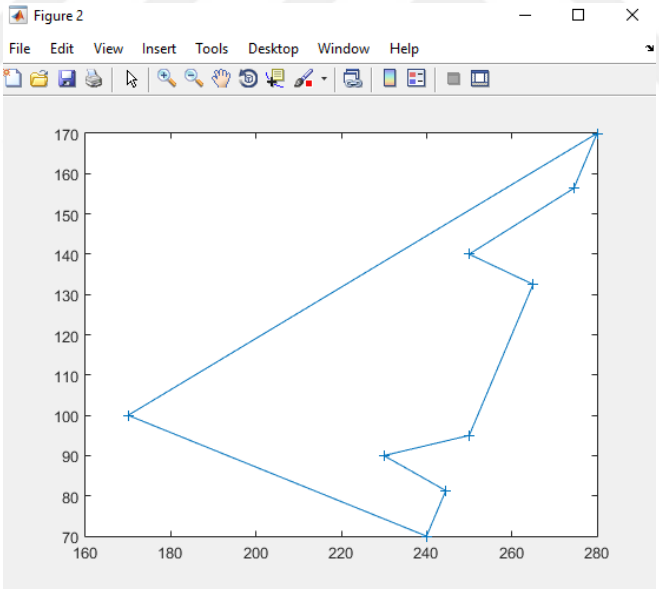
Durum 5 Bu durumun diğer şekillerden farklı olan kısmı, ana parsel ile sorgulanan parselin ikiden daha fazla kesişme noktası olmasıdır. Bu durum yazılan algoritmanın, kesişim sayısının ikiden daha fazla olduğu durumlarda da tam olarak çalıştığını göstermektedir.

Tablo 2.11. Durum 5 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	**Y (m)**	**X(m)**
A	280	170
B	270	100
C	170	100



Şekil 2.15. Durum 5 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.16. Durum 5 iki parselin kesişim bölgesi

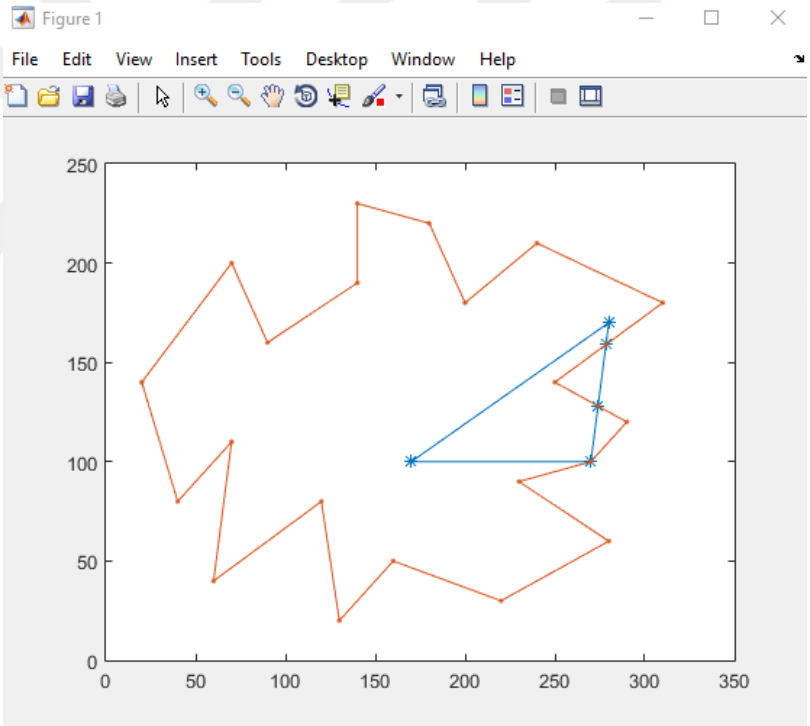
Tablo 2.12. Durum 5 kesişim Noktaları

Kesişim Noktaları	Y(m)	X(m)
	274,545	156,364
	265,000	132,500
	250,000	95,000
	244,516	81,290

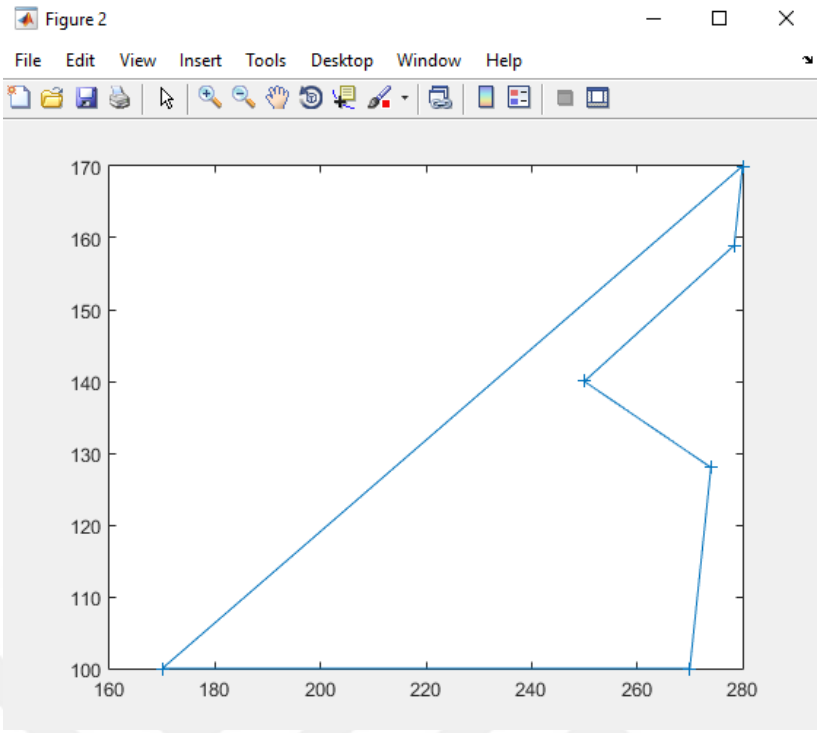
Durum 6 Ana parsel ile sorgulanan parselin bir köşe noktasının aynı olmasıdır. Bu durum ana parselin ve diğer parselin köşe noktalarının aynı olduğu durumda da hatasız çalıştığını göstermektedir.

Tablo 2.13. Durum 6 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	280	170
B	270	100
C	170	100



Şekil 2.17. Durum 6 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.18. Durum 6 iki parselin kesişim bölgesi

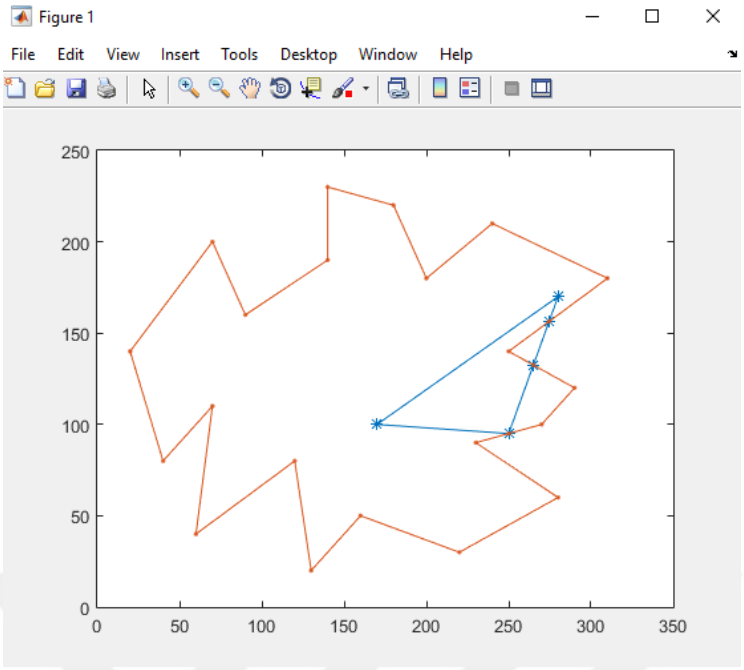
Tablo 2.14. Durum 6 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	158,947	278,421
	128,000	274,000
	100,000	270,000

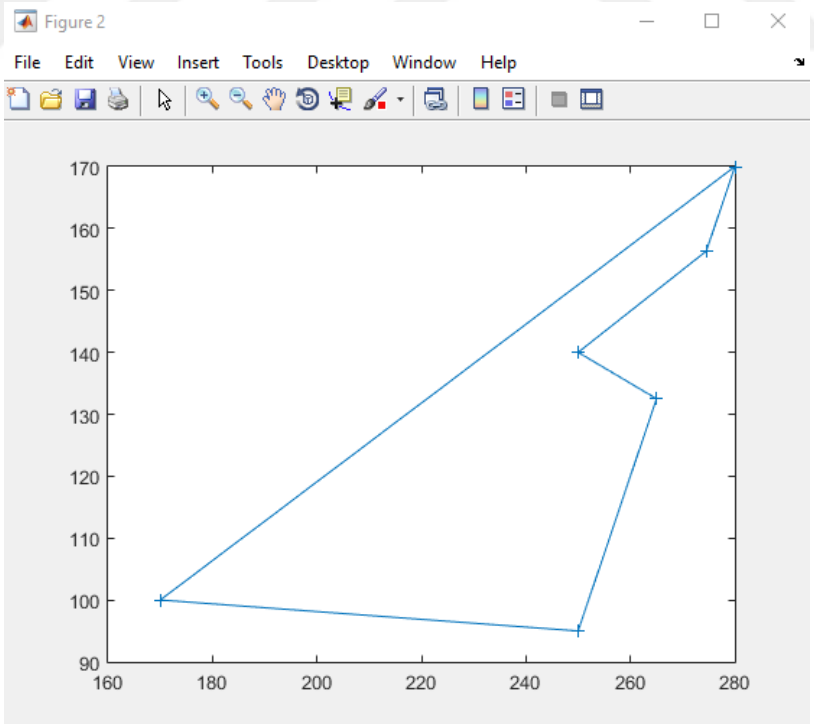
Durum 7 Ana parsel ile sorgulanan parselin kesiştikleri nokta, aynı zamanda sorgulanan parselin köşe noktası olmasıdır.

Tablo 2.15. Durum 7 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	280	170
B	250	95
C	170	100



Şekil 2.19. Durum 7 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.20. Durum 7 iki parselin kesişim bölgesi

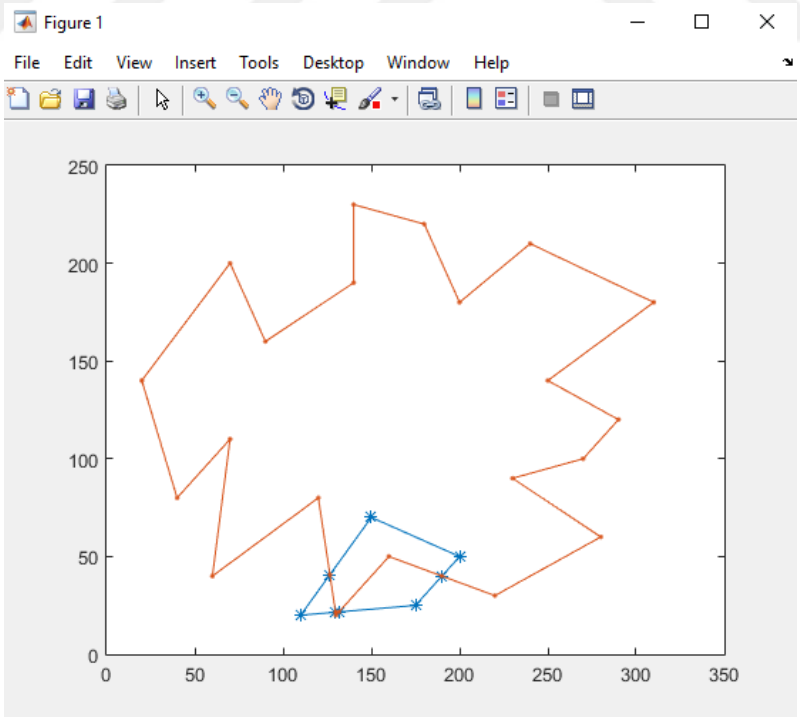
Tablo 2.16. Durum 7 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	274,545	156,364
	265,000	132,500
	250,000	95,000

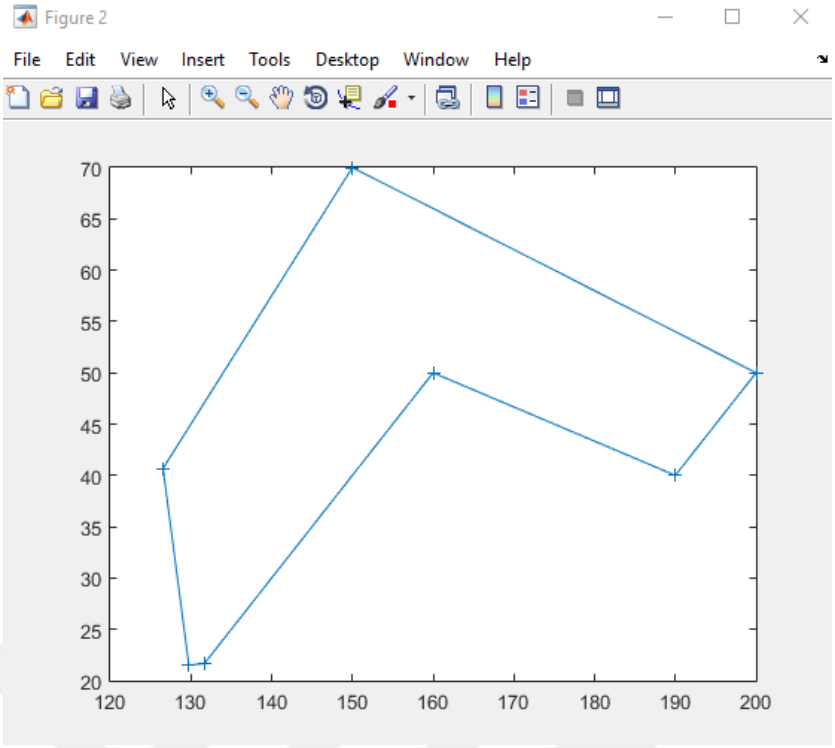
Durum 8 Ana parsel ile sorgulanan parselin en az bir kenarının birbirine paralel olmasıdır.

Tablo 2.17. Durum 8 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	150	70
B	200	50
C	175	25
D	110	20



Şekil 2.21. Durum 8 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.22. Durum 8 iki parselin kesişim bölgesi

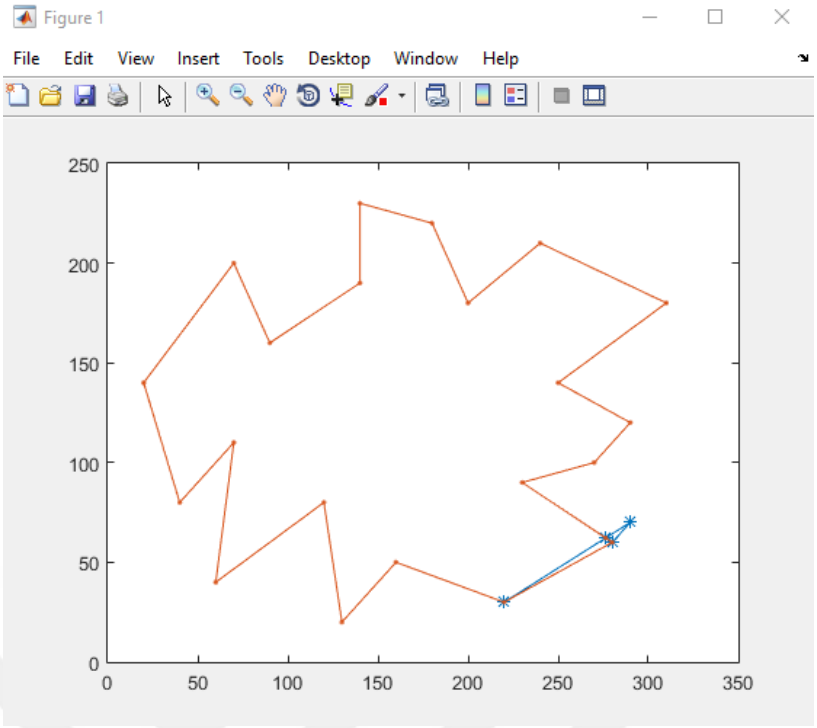
Tablo 2.18. Durum 8 kesişim noktaları

Kesişim Noktaları	Y(m)	X(m)
	40,000	190,000
	21,667	131,667
	21,519	129,747
	40,689	126,552

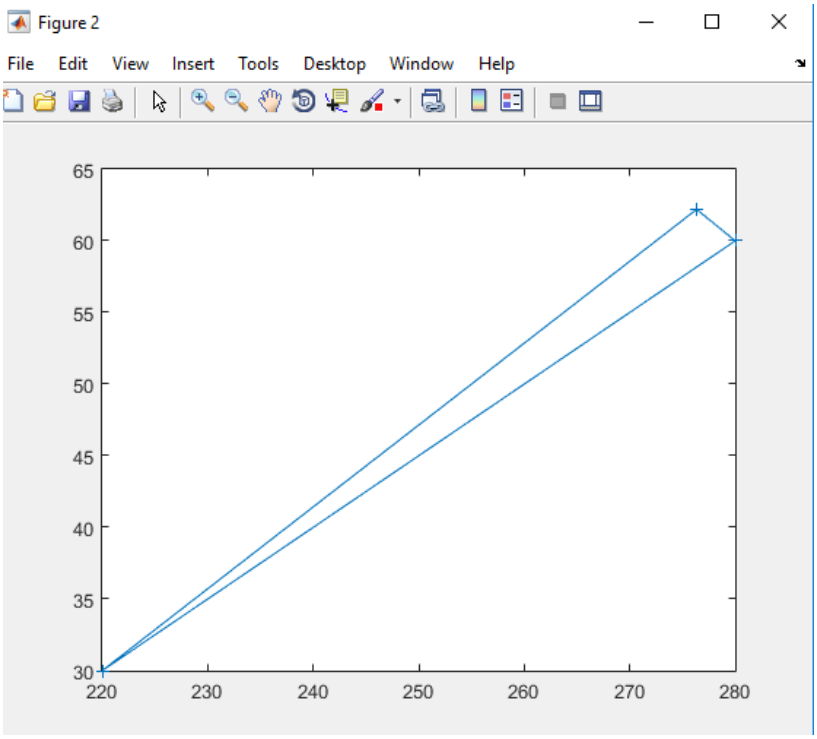
Durum 9 Ana parsel ile sorgulanan parselin bir kenarının ortak olmasıdır.

Tablo 2.19. Durum 9 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	290	70
B	280	60
C	220	30



Şekil 2.23. Durum 9 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.24. Durum 9 iki parselin kesişim bölgesi

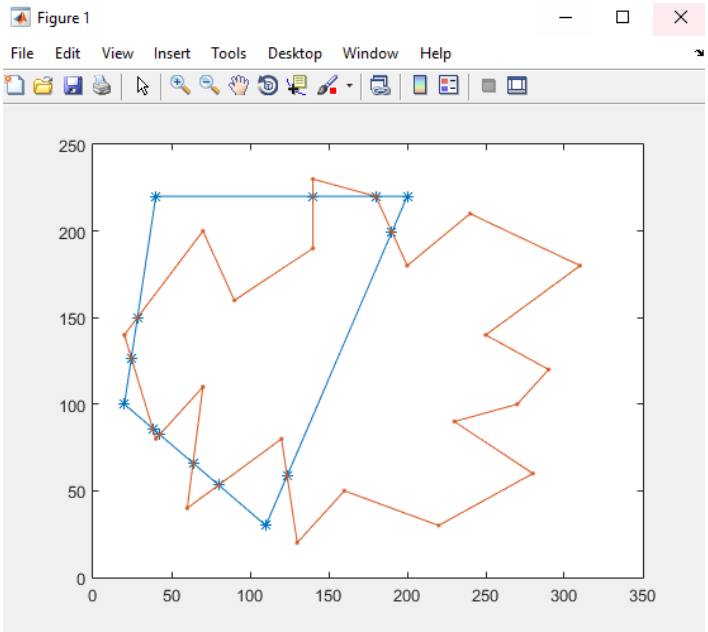
Tablo 2.20. Durum 9 kesişim Noktaları

Kesişim Noktaları	Y(m)	X(m)
	276,341	62,195
	280,000	60,000
	220,000	30,000

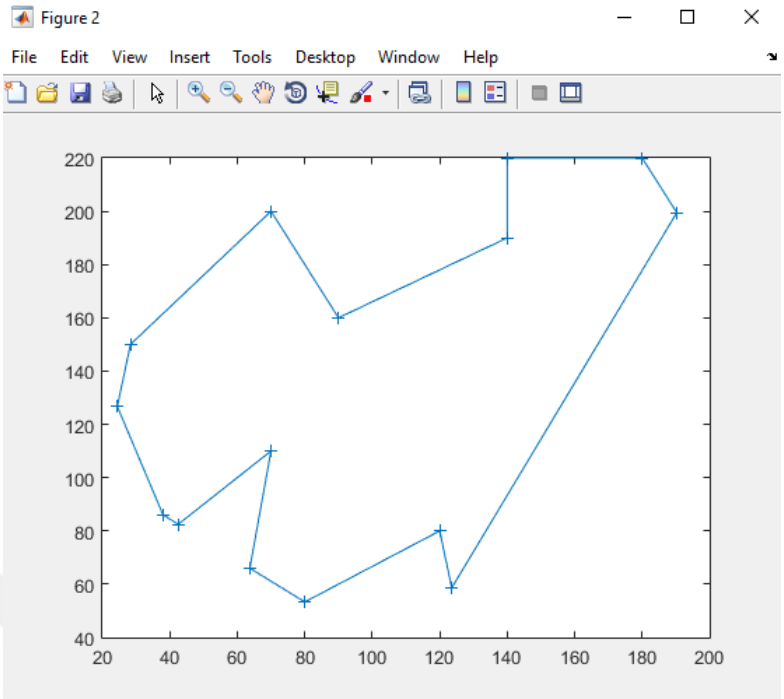
Durum 10 Bu durumda birçok özel durum aynı anda bulunmaktadır. Bu özel durumlardan biri iki parselin köşe noktalarının aynı olması, biri bu parsellerin kenarlarının birbirlerine dik olma durumu ve son durum ise kenarların kesişme sayılarının aynı kenar üzerinde ikiden daha fazla olma durumudur. Bu parsel arazideki çoğu duruma örnek olmaktadır. Bahsedilen algoritmaya göre hatasız bir şekilde çalışmaktadır.

Tablo 2.21. Durum 10 sorgulanacak parselin köşe koordinatları

Sorgulanacak parsel	Y (m)	X(m)
A	40	220
B	200	220
C	110	30
D	20	100



Şekil 2.25. Durum 10 ana parsel ile sorgulanacak parselin birbirlerine göre durumları



Şekil 2.26. Durum 10 iki parselin kesişim bölgesi

Tablo 2.22. Durum 10 kesişim noktaları koordinatları

	Y(m)	X(m)
Kesişim Noktaları	28,334	150,000
	140,000	220,000
	180,000	220,000
	190,270	199,460
	123,562	58,630
	80,000	53,334
	63,714	66,000
	42,500	82,500
	38,000	86,000
	24,445	126,667

3. BULGULAR VE İRDELEMELER

3.1. Yöntemlerin Karşılaştırılması

Nokta sorgulama yöntemleri olarak yukarıda anlatılan üç yöntemin zamansal olarak karşılaştırılması tablo 3.1. de verilmektedir.

Tablo 3.1. Zamansal karşılaştırma

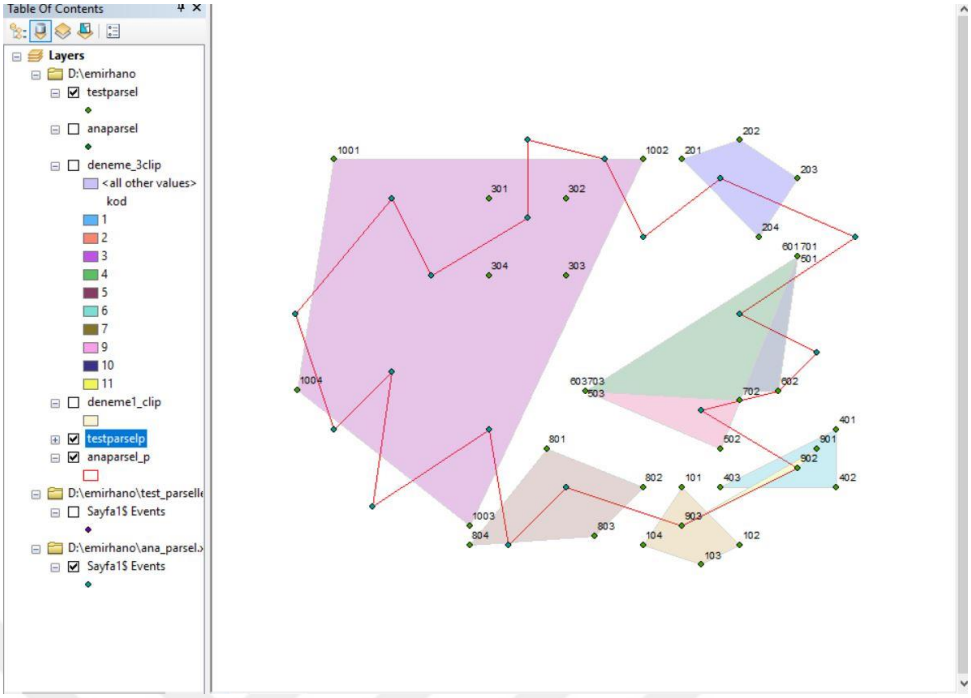
Yöntemler	Zaman (saniye)
Açıların Toplamı Yöntemi	0.915
Nokta-Vektör Yöntemi	1.075
Matlab İnpolygon Fonsiyonu	1.187

Yapılan bu zamansal karşılaştırmaya göre, açıların toplamı yöntemi, bu üç yöntem içinde en hızlı sonuca ulaşan yöntemdir. Nokta – Vektör yöntemi ise açıların toplamı yöntemine göre daha geç sürede sonuca ulaşmıştır. Son olarak Matlab programının inpolygon fonksiyonu, bahsedilen bu üç yöntem arasında sonuca en uzun sürede ulaşan yöntemdir. Bu fonksiyon Matlab programının hazır bir komutu olmasına rağmen, diğer iki algoritmaya göre daha yavaş sonuca ulaşmaktadır.

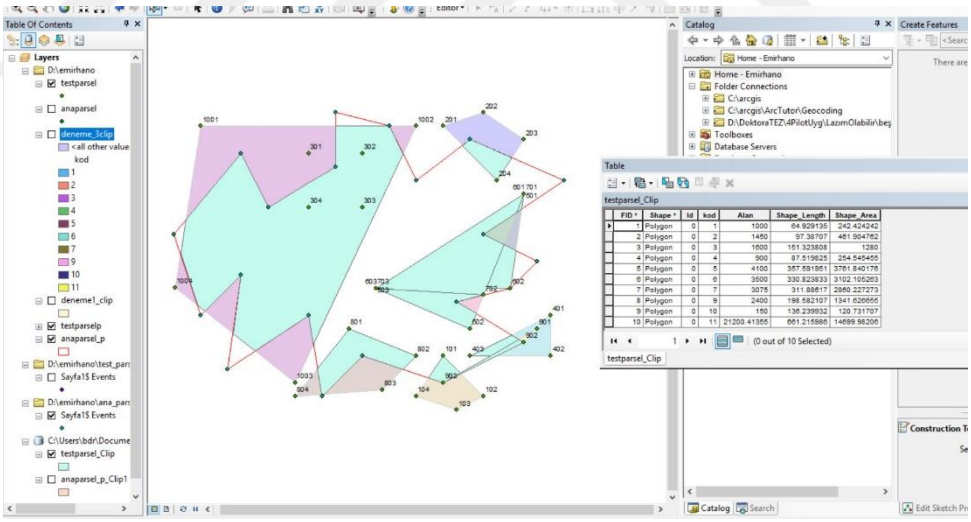
3.2. ArcGIS Karşılaştırması

Bu tez çalışmasında, iki parselin çakışması sonucu oluşan ortak bölgenin alan hesabı için yapılan adımlar ArcGıs programında da yapılmıştır. Yukarıda açıklanan özel durumlardaki sorgulanan parseller tekrardan ArcGıs programında irdelenmiştir. Elde edilen sonuçlar aşağıda açıklanmıştır.

Öncelikli olarak iki parselin köşe koordinatları ArcGıs programına tanıtılmıştır. Programında içerisinde bulunan hazır komut (clip) yardımıyla ana parsel ile sorgulanan parseller elde edilmiştir. Sonrasında programın bünyesinde bulunan alan hesaplama yöntemi ile bu bölgelerin alanları hesaplanmıştır.



Şekil 3.1. ArcGIS programında sorgulanan parseller



Şekil 3. 2. ArcGIS programında sorgulanan parsellerin alan gösterimi

Bu hesaplamalar, Matlab programı yardımıyla yazılan kod ile bulunan alan hesapları ile aynıdır (Tablo 3.2.). Her ikisi de verilen özel durumdaki parseller için hatasız bir şekilde alan hesaplamalarını yapmışlardır. Ancak ArcGıs programında bu adımlar yapılırken, ana parsel ile sorgulanan parsellerin kesişim noktaları doğrudan hesaplanamamıştır. Program yardımıyla birkaç adım yardımıyla kesişen noktalar bulunmuş

ve o noktalara koordinat değerleri verilmiştir. Bu tez çalışmasında yazılan algoritmada ise sadece iki parselin köşe koordinatları verilmesi yeterlidir. Algoritmanın içerindeki adımlar yardımıyla ana parsel ile sorgulanan parselin kesişim noktaları direk bir şekilde elde edilip kullanıcıya verilmektedir.

Tablo 3.2. Sorgulanan parsellerin alan hesabı

Sorgulanan Parseller	**Alan Hesabı (m^2)**
Durum 1	242,424
Durum 2	461,905
Durum 3	254.546
Durum 4	1280,000
Durum 5	3761,841
Durum 6	3102,107
Durum 7	2860,229
Durum 8	1341,628
Durum 9	120,730
Durum 10	14726,000

4. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında, çakışan iki parselin oluşturduğu ortak bölgeyi bulup, bulunan bu bölgenin köşe koordinatları ve alan hesabını yapmaktır. Bu amaç doğrultusunda literatürde mevcut birçok yöntem incelenip bunların içinden hem konkav hem de konveks poligonlar için işlem yapabilen algoritmalar tercih edilmiştir. Bu algoritmaların her biri ayrı ayrı on farklı özel durum için ayrıntılı bir şekilde incelenip test edilmiştir. Bu yöntemler içinden açıların toplama yöntemi; işlem zamanı, konkav ve konveks parseller ve özel durumlar için en uygun yöntem olarak belirlenmiştir.

Parsellerin kesişimi ile oluşan ortak parselin köşe noktalarının belirlenmesi için kullanılan açıların toplama yönteminden sonra bu parselin alan hesabı için ortak parselin köşe noktalarının ardışık sırada numaralandırılması gerekmektedir. Bu uygulama CAD ve GIS yazılımlarından kullanıcı tarafından belirlenmektedir. Kullanıcıya gerek kalmadan noktaların ardışık sıralanması için birkaç farklı yöntem incelenmiş olup, bu yöntemler içinden Weiler–Atherton Yöntemi test edilerek seçilmiştir. Bu yöntemin tercih sebebi hem konkav hem de konveks parseller için sonuç üretebilmesidir.

Açıların toplama ve Weiler–Atherton yöntemleri kullanılarak oluşturulan Matlab programıyla örnek bir ana parselle farklı özel durumlar için belirlenen kesiştirilecek parseller test edilmiştir. Uygulamalar sonucunda kullanıcıya gerek kalmadan sadece kesiştirilecek iki parselin köşe koordinatları girilerek kesişimden oluşan ortak parselin köşe koordinatları ve alanı hesaplanmıştır. Hesaplanan ortak alanlar ve köşe koordinatları ArcGIS yazılımıyla test edilmiştir. Alanlar ve köşe koordinatları hem bu tezde yazılan algoritma hem de ArcGIS yazılımında birbiriyle mm hassasiyetinde tutarlı sonuçlar vermiştir.

Haritacılık sektöründe kullanılan CAD ve CBS yazılımlarının dinamik yapıya sahip olmaları kullanıcılara, bu programlara görülen bir eksikliği veya sonradan ortaya çıkan bir hatayı gidermek için yama (patch) program yapma imkânı sunarlar. Bu çalışmada Matlabda yapılan algoritmayla; haritacılıkta sıklıkla kullanılan pek çok uygulamada kullanıcıya, zaman kazandırmakla beraber farklı disiplinlerle olan çalışmalarda farkındalık oluşturmayla kendilerini genişletmeleri teşvik etmeyi hedeflemiştir. Bu nedenle, haritacılık ve farklı disiplinlerle yapılan ortak uygulamalara ilişkin geometrik problemleri çözme bilgi ve becerisinin kazandırılması önem kazanmaktadır. Bu alanlarda yaygın olarak kullanılan

nokta(konum), doğru ve parsel ilişkilerinin geometrik disiplinlerinin mühendislerimiz tarafından tanınması ve ilgi alanı olarak görülmesi, gelecekte karşılaşacakları geometrik problemlerin çözümüne büyük katkılar sağlayacaktır.

5. KAYNAKLAR

Barsky, B., A. and Liang, Y., D., 1984. A New Concept and Method for Line Clipping, ACM Transactions on Graphics, 3, 1, 1–22.

Cyrus, M. and Beck, J., 1978. Generalized Two- and Three-Dimensional Clipping, Computers & Graphics, 3, 1, 23–28.

Çelik, N., 2013. İptale Konu İmar Planı Uygulamalarında Geri Dönüş İşleminin İrdelenmesi ve Çözüm Önerileri Yaklaşımı, Yüksek Lisans Tezi, K.T.Ü., Fen Bilimleri Enstitüsü, Trabzon.

Day, J., D., 1992. A New Two-Dimensional Line Clipping Algorithm for Small Windows, Computer Graphics Forum, 11, 4, 241–245.

Foley, V., D., Feiner, S., K. and Huges, J., F., 1990. Computer Grpahics Principles and Practice, 2nd ed. Reading, Mass: Addison- Wesley.

Greiner, G. and Hormann, K., 1998. Efficient Clipping of Arbitrary Polygons, ACM Transactions on Graphics, 17, 2, 71–83.

Huang, Y., Q. and Liu, Y., K., 2002. An Algorithm for Line Clipping against a Polygon Based on Shearing Transformation, Computer Graphics Forum, vol. 21, no. 4, pp. 683–688.

Kaya, A. ve Yıldırım, F., 1999. Nokta- Parsel Sorgulaması, Yerel Yönetimlerde Kent Bilgi Sistemi Uygulamaları Sempozyumu, Nisan, Trabzon, Bildiriler Kitabı: 1-10.

Kitay, M., G., 1985. Land Acquisitionin Developing Countries, Lincolh İnstitue, Boston, Usa.

Liu, Y., K., Wang, X., Q., Bao, S., Z., Gomboši, M. and Žalik, B., 2007. An Algorithm for Polygon Clipping, and for Determining Polygon Intersections and Unions, Computers & Geosciences, 33, 5, 589–598.

Maillot, P., G., 1992. A New, Fast Method for 2D Polygon Clipping: Analysis and Software Implementation, ACM Transactions on Graphics, 11, 3, 276–290.

Möller, T., 1997. A Fast Triangle-Triangle Intersection Test, Journal of Graphics Tools, 2, 2, 25–30.

Nicholl, T., M., Lee, D., T. and Nicholl, R., A., 1987. An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis, ACM SIGGRAPH Computer Graphics, 21, 4, 253–262.

Tırak, O., 2017. Kentsel Gelişme Alanlarının Çok Yönlü Düzenlenmesinde İmar Uygulamaları: Van Örneği, Yüksek Lisans Tezi, K.T.Ü., Fen Bilimleri Enstitüsü, Trabzon.

Sproull, R., F. and Sutherland, I., E., 1968. A Clipping Divider," Proceedings of the December, New York, USA, fall joint computer conference, part I on – AFIPS'68 (Fall, part I).

Sutherland, I., E. and Hodgman, G., W., 1974. Reentrant Polygon Clipping, Communications of the ACM, 17, 1, 32–42.

Weiler, K. and Atherton, P., 1977. Hidden surface removal using polygon area sorting, ACM SIGGRAPH Computer Graphics, 11, 2, 214–222.

Yakar, A., 2000. 18. Madde Uygulamasının Eleştirisi ve Öneriler, Mülkiyet Dergisi, 25-31.

Yanalak, M. ve İpbüker, C., 2003. Hesaplamalı Geometri, Harita Dergisi,129, 50-62.

Yomralıoğlu, T., 2000. Coğrafi Bilgi Sistemleri: Temel Kavramlar ve Uygulamaları, 5.Baskı (2009), 480. ISBN 975-97369-0-X, İstanbul.

6. EKLER

Ek 1. Açıların toplamı yöntemi Matlab algoritması

```
function acilarin_toplami
clc
clear all
A = xlsread('noktalarkose.xls');
B = xlsread('yeni_noktalar.xls');%irdenilecek noktalar
epsilon=1e-4;
dongu_sayisi=length(B(:,1));
A_eleman_sayisi=length(A(:,1));

for i=1:dongu_sayisi
   sumang=0;
   for j=1:A_eleman_sayisi-2
      a=B(i,3);
      b=B(i,2);

      x1=A(j,3);
      y1=A(j,2);

      x2=A(j+1,3);
      y2=A(j+1,2);

      dx1=x1-a;
      dy1=y1-b;

      dx2=x2-a;
      dy2=y2-b;
      teta=arctanprog(dy2,dx2)-arctanprog(dy1,dx1);
      if (teta<0)&(teta<-pi)
         sumang=sumang+(2*pi+teta);
      elseif (teta<0)&(teta>-pi)
         sumang=sumang+teta;
      elseif teta>pi
         sumang=sumang+(2*pi-teta);
      else
         sumang=sumang+teta;
      end
   end
   t=sumang*180/pi;
   i
   if (t>360-epsilon)&(t<360+epsilon)
     'nokta içeride'
   else
      'nokta dışarıda'
   end
end
```

```
function arct=arctanprog(pay,payda)
if (pay<0)&(payda==0)
   arct=3*pi/2;
elseif (pay>0)&(payda==0)
   arct=pi/2;
else
   tt=atan(pay/payda);
   if (pay>0)&(payda>0)
      arct=tt;
   elseif (pay>0)&(payda<0)
      arct=tt+pi;
   elseif (pay<0)&(payda<0)
      arct=tt+pi;
   elseif (pay==0)&(payda<0)
      arct=tt+pi;
   elseif (pay==0)&(payda>0)
      arct=2*pi;
   else
      arct=tt+2*pi;
   end
end
end
end
```

Ek 2. Nokta-Vektör yöntemi Matlab algoritması

```
clc
clear all
A = xlsread('noktalarkose.xls');
B = xlsread('yeni_noktalar.xls');%irdenilecek noktalar
dongu_sayisi=length(B(:,1));
A_eleman_sayisi=length(A(:,1));
for i=1:dongu_sayisi
   sayac=0;
   nokta=B(i,2:3);
   mm=abs(nokta(2)-A(1:end-2,3));
   MM=sort(mm);
   [bb,cc]=find(MM>0);
   mx=MM(bb(1));
   My=max(abs(nokta(1)-A(1:end-2,2)));
   for j=1:A_eleman_sayisi-2
      Di_matris=[2*My -(A(j+1,2)-A(j,2));mx -(A(j+1,3)-A(j,3))];
      Di_ussu_matris=[2*My A(j,2)-nokta(1);mx A(j,3)-nokta(2)];
      Di_2_ussu_matris=[A(j,2)-nokta(1) -(A(j+1,2)-A(j,2));A(j,3)-nokta(2) -(A(j+1,3)-
A(j,3))];

      Di=det(Di_matris);
      Di_ussu=det(Di_ussu_matris);
      Di_2_ussu=det(Di_2_ussu_matris);

      hesap_1=Di_ussu*(Di-Di_ussu);
      hesap_2=Di_2_ussu*Di;
      if (hesap_1>0)&(hesap_2>0)
         sayac=sayac+1;
      end
   end
   i
   if mod(sayac,2)==0
      'nokta dışarıda'
   else
      'nokta içeride'
   end
end
```

Ek 3. Matlab inpolygon fonksiyonu

```
clc
clear all
A = xlsread('noktalarkose.xls');
B = xlsread('yeni_noktalar.xls');%irdenilecek noktalar

dongu_sayisi=length(B(:,1));
A_eleman_sayisi=length(A(:,1));
A_koordinatlari=[A(:,2) A(:,3)];
for i=1:dongu_sayisi
   deney_noktasi(i,:)=[B(i,2) B(i,3)];
   [in,on] =
inpolygon(deney_noktasi(i,2),deney_noktasi(i,1),A_koordinatlari(:,2),A_koordinatlari(:,1);
   nokta_ic_dis(i)=in;

   if in==1|on==1
      'nokta icerde'
   else
      'nokta dısarda'
   end
end
```

Ek 4. Alan hesabı Matlab algoritması

```
%iki noktada kesişme durumunda alan hesabı
clc
clear all
%A:ana parselin köşe koordinatları
%A'nın elemanları [y x] şeklindedir
%A matrisinin son iki elemanı baştaki 1. ve 2. elemandır
%Çünkü eğer diğer parselin noktalarının içeride olup olmadığını belirlemek
%için açıların toplamı yöntemi kullanılması düşünülürse bu iki noktaya
%ihtiyaç vardır
%A'nın noktaları bir excel dosyasından çekilmektedir:
%Excel dosyasında 1. sütun sıra numaralarını 2. sütun köşe noktalarının y
%koordinatlarını 3. sütunu ise x koordinatlarını göstermektedir. Dolayısı
%ile A matrisi 3. sütuna sahiptir.
A = xlsread('noktalarkose.xls');
%B:irdelenen parselin köşe noktaları
%B'nin elemanları [y x] şeklindedir:
%        B = [220 50;250 20;230 10;200 20];
%      B = [220 220;250 230;280 210;260 180];
%         B=[120 200;160 200;160 160;120 160]; %kensrlsrın dik olma durumu
%       B=[300 80;300 50;240 50]; %üçgensel bölge
%      B=[280 170;240 70;170 100];% sağ taraftaki 4 noktadan kesişen
%      B=[280 170;270 100;170 100]; %ortak köşe noktası
%      B=[280 170;250 95;170 100];  %kesiştikleri noktayı köşe kabul eden
%        B=[150 70;200 50;175 25;110 20]; %paralellik
%     B=[290 70;280 60;220 30]; %ortak kenarlar
%    B=[20 220;200 220;110 30;50 100];
         B=[40 220;200 220;110 30;20 100]; %en büyük bölge
%    B=[110 220;310 220;250 20;70 75];
%  B=[40 220;240 240;110 30;10 60];

%A_alan_noktaları, B_alan_noktaları: A parseli ile B parselinin ortak
%bölgesini oluşturan noktaları belirlemek için A_alan_noktaları ve
%B_alan_noktaları diye yeni matrisler oluşturuluyor
%Başlangıçta bu matrisler A ve B matrislerinin y ve x koordinatları olarak
%seçilmekte ve sonrasında sadece alanı oluşturan bölgenin köşe noktalarına
%indirgenecektir.
%A_alan_noktaları(başlangıçta):A'nın [y x] noktaları olarak alınıyor:
A_alan_noktalari=[A(1:end-1,2) A(1:end-1,3)];
%B_alan_noktaları(başlangıçta):B'nin [y x] noktaları olarak alınıyor ve
%başlangıç noktasına dönmesi için(kapalı bir şekil oluşması için) ilk nokta
%sisteme ilave ediliyor:
B_alan_noktalari=[B;B(1,:)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ilk adım: B parselinin köşe noktalarının hangilerinin A parselinin iç
%noktası olduğu belirleniyor
```

```
%dongu sayısı: B nin her bir köşe noktasının A'nın içerisinde kalıp
%kalmadığı belirleneceğinden döngü sayısı B'nin satır sayısına eşittir:
dongu_sayisi=length(B(:,1));

%A_koordinatları: A'nın 1. sütunu sıra sayısını gösterdiği için A'nın koordinatları A'nın 2.
%ve 3. sütunundan çekilmelidir:
A_koordinatlari=[A(:,2) A(:,3)];
%parsel_2_ic: Bu vektör B'nin içerde kalan noktalarını tutan vektördür.
%Başlangıçta [0 0] alınıyor ve bulunan her bir iç nokta bu vektöre ilave
%ediliyor
%Bu programda iç noktalar matlab inpolygon fonksiyonu ile bulunmuştur.
%Diğer yöntemler de sisteme ilave edilebilir.
parsel_2_ic=[0 0];
%B'nin A parselinin içerisinde kalan noktalar belirleniyor:
for i=1:dongu_sayisi
    %deney_noktası: B'nin i.köşe noktası alınarak içerde olup olmadığı
    %inceleniyor:
    deney_noktasi=[B(i,1) B(i,2)];
    % function acilarin_toplami komutu ile noktanın içeride olup olmadığı inceliyor:
    %Eğer in değişkeninin değeri 1'e eşit olursa nokta içeridedir. Dolayısı
    %ile eğer in=1 ise iç noktaları tutan vektöre bu nokta ilave
    %edilmelidir:
    if in==1
       parsel_2_ic=[parsel_2_ic;deney_noktasi]; %parsel 2 deki iç noktalar vektörü [y x]
şeklinde
    end
end

%B parselinin köşe noktalarının hangileri A parselinin içerisinde kaldığı
%belirlendi. Bu noktalar ortak alanın köşe noktalarıdır. Alan hesabında
%tekrar kullanılacaktır.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%B parseli ile A parselinin kesişme noktaları belirleniyor:
%Yöntem: A parselinin ardışık köşeleri ile B parselinin ardışık köşeleri
%kullanılarak doğrular oluşturuluyor. Bu doğruların kesişme noktaları
%belirleniyor. Eğer bu nokta A ve B 'nin ilgili köşe noktalarının
%belirlediği dikdörtgensel bölgenin içerisinde kalıyorsa kesişme noktası
%aranılan noktadır. Aranılan noktalar sisteme ilave edilir. Ekstra durumlar
%aşağıda belirtilecektir.


%Dikdörtgensel bölgenin belirlenmesi stratejisi: A parselinin i. köşe
%noktası ile (i+1). köşe noktalarının birşeltiren doğru ile B parselinin j.
%köşe noktası ile (j+1). köşe noktasını birleştiren doğrularının kesişme
%noktası (y0,x0) olsun. Bu A parselinin ilgili köşe noktalarının  x ve y noktalarına
%göre minimum ve maksimumu(A_min,A_mak) hesaplanır. Benzer şekilde B parselinin
ilgili köşe noktalarının
```

```
%x ve y noktalarına göre minimum ve maksimumu(B_min,B_mak) hesaplanır. Eğer x0 ve
y0 bu
%A_min_x<=x0<=A_mak_x ve B_min_x<=x0<=B_mak_x; A_min_y<=y0<=A_mak_y
ve
%B_min_y<=y0<=B_mak_y ise (y0,x0) kesişme noktası aranılan alanın köşe
%noktasıdır.

%kesisme_sayısı: A ve B parsellerinin kaç noktada kesiştiğini belirleyen
%değişken
kesisme_sayisi=0;

%A_eleman_sayisi: A parselinin köşe sayısını belirleyen değişken. Bu A
%matrisinin sutün sayısına eşittir. A parselinin ardışık köşe noktaları ile
%B parselinin ardışık köşe noktaları kesiştireceğinden A_eleman_sayisi
%döngünün kaç kere devam edeceğini belirler
A_eleman_sayisi=length(A(:,1));

%B_koordinatlari: B koordinatlarına B'nin 1. elamanı eklenmesi ile elde
%edilir.
B_koordinatlari=[B;B(1,1) B(1,2)];

%kesisme_noktalari: Aranılan kesişme noktalarını tutan matris. [x y]
%şeklinde tutuluyor. Başlangıçta [0 0] olarak alınıyor. Bulunan kesişme noktaları
%bu matrise ilave edilecek
kesisme_noktalari=[0 0];
%A parselinin ardışık köşeleri ile B parselinin ardışık köşelerinden geçen
%doğruların kesişme noktaları hesaplanıyor
for j=1:A_eleman_sayisi-2
   for i=1:dongu_sayisi
      %[a_1,b_1] A parselinin j. köşesinin [x,y] koordinatı
      a_1=A_koordinatlari(j,2);
      b_1=A_koordinatlari(j,1);

      %[a_2,b_2] A parselinin (j+1). köşesinin [x,y] koordinatı
      a_2=A_koordinatlari(j+1,2);
      b_2=A_koordinatlari(j+1,1);

      %[x_1 y_1] B parselinin i. köşesinin [x,y] koordinatı
      x_1=B_koordinatlari(i,2);
      y_1=B_koordinatlari(i,1);

      %[x_2 y_2] B parselinin (i+1). köşesinin [x,y] koordinatı
      x_2=B_koordinatlari(i+1,2);
      y_2=B_koordinatlari(i+1,1);


      %Özel durumlara göre doğruların kesişme noktaları belirleniyor:
      %Durum 1: B parselinin i. ve (i+1). köşesinden geçen doğru x
      %eksenine dik olması durumu. Yani x_1=x_2 olması
```

```
        if x_1==x_2
            %m2: A parselinin j. ve (j+1). noktalarından geçen doğrunun
            %eğimi
            m2=(b_2-b_1)/(a_2-a_1);

            %Durum 1'de oluşacak kesişme noktaları:
            kesisme_x=x_1;
            kesisme_y=b_1+m2*(kesisme_x-a_1);

        %Durum 2: A parselinin j. ve (j+1). köşesinden geçen doğru x
        %eksenine dik olması durumu. Yani a_1=a_2 olması
        elseif a_1==a_2
            %m1: B parselinin i. ve (i+1). noktalarından geçen doğrunun
            %eğimi
             m1=(y_2-y_1)/(x_2-c_1);

            %Durum 2'de oluşacak kesişme noktaları:
            kesisme_x=a_1;
            kesisme_y=y_1+m1*(kesisme_x-x_1);
        %Durum 3: Durum 1 ve Durum 2'nin olmadığı durum
        else
            %m1: B parselinin i. ve (i+1). noktalarından geçen doğrunun
            %eğimi
            m1=(y_2-y_1)/(x_2-x_1);
            %m2: A parselinin j. ve (j+1). noktalarından geçen doğrunun
            %eğimi
            m2=(b_2-b_1)/(a_2-a_1);

            %Durum 3'de oluşacak kesişme noktaları:
            kesisme_x=(-m2*a_1-y_1+b_1+m1*x_1)/(m1-m2);
            kesisme_y=y_1+m1*(kesisme_x-x_1);
        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Kesişme noktası aranılan kesişme noktası olup olmadığı kontrol ediliyor:
        x_dik1_min=min(a_1,a_2);%1. dikdörtgende x lere göre kıyaslama(sağda mı solda
mı)
        x_dik1_mak=max(a_1,a_2);%1. dikdörtgende x lere göre kıyaslama(sağda mı solda
mı)
        test_dik1_x=(kesisme_x>=x_dik1_min)&(kesisme_x<=x_dik1_mak);%1.
dikdörtgenin x koordinatının arasında kalıyor mu?

        y_dik1_min=min(b_1,b_2);%1. dikdörtgende y lere göre kıyaslama(sağda mı solda
mı)
        y_dik1_mak=max(b_1,b_2);%1. dikdörtgende y lere göre kıyaslama(sağda mı solda
mı)
        test_dik1_y=(kesisme_y>=y_dik1_min)&(kesisme_y<=y_dik1_mak);%1.
dikdörtgenin y koordinatının arasında kalıyor mu?
```

```
        x_dik2_min=min(x_1,x_2);%2. dikdörtgende x lere göre kıyaslama(sağda mı solda
mı)
        x_dik2_mak=max(x_1,x_2);%2. dikdörtgende x lere göre kıyaslama(sağda mı solda
mı)
        test_dik2_x=(kesisme_x>=x_dik2_min)&(kesisme_x<=x_dik2_mak);%2.
dikdörtgenin x koordinatının arasında kalıyor mu?

        y_dik2_min=min(y_1,y_2);%2. dikdörtgende y lere göre kıyaslama(sağda mı solda
mı)
        y_dik2_mak=max(y_1,y_2);%2. dikdörtgende y lere göre kıyaslama(sağda mı solda
mı)
        test_dik2_y=(kesisme_y>=y_dik2_min)&(kesisme_y<=y_dik2_mak);%2.
dikdörtgenin y koordinatının arasında kalıyor mu?


         test=test_dik1_x&test_dik1_y&test_dik2_x&test_dik2_y;
         %Eğer kesişme noktası oluşturulan dkdörtgenin içerisinde ise bu
         %nokta aranılan keissme noktası olup alan noktasının bir köşe
         %noktasını temsil eder. Dolayısı ile bu nokta hem
         %kesisme_noktaları hem de alan noktalarına eklenmelidir.
         if test==1
            %Özel durum: Farklı doğruların ortak kesişme durumu olması
            %Eğer son elde edilen kesişme noktası yeni elde edilen ile
            %aynı ise kesişme  noktalarına eklenmemelidir.
            if
(kesisme_y~=kesisme_noktalari(end,2))|(kesisme_x~=kesisme_noktalari(end,1))

               kesisme_sayisi=kesisme_sayisi+1;
               %kesişme noktalarına yeni bulunan noktalar ekleniyor
               %[x y] şeklinde
               kesisme_noktalari=[kesisme_noktalari;kesisme_x kesisme_y];

               %Bulunan kesisme noktası alan noktalarında ilgili yere
               %atanıyor. Noktanın eklenme yeri A ve B parselinin hangi
               %noktalar arasında kaldığına bakılarak karar verilir.

               %Özel durum: A parselinin j. köşe noktası (j+1). köşe noktası arasında 1'den
fazla
               %kesişme noktası içermesi durumu veya benzer şekilde B parselinin i.
               %köşesi ile (i+1). köşesi kesişme noktası içermesi durumu

               % A parseli için: Bu durumda kesişme noktası A parselinin
               % j. köşe noktası ile (j+1). köşe noktası arasına
               % eklenmelidir. Fakat önceki kesişme noktasının öncesine
               % mi sonrasına mı eklenmesi gerektiğine karar verilmeli.


               if kesisme_sayisi>1
                  %Eğer j ve i'ler j_eski ve i_eski'ye eşit oluyorsa bu
```

```
                %durumda önceki kesişme noktasının hemen ardına ilgili
                %noktalar eklenmelidir. Bu yer ise önceden tanımlanan
                %kesisme_yeri değişkenin hemen ardı olmalıdır.
                if j==j_eski
                  A_alan_noktalari=[A_alan_noktalari(1:kesisme_yeri_A,:);kesisme_y
kesisme_x;A_alan_noktalari(kesisme_yeri_A+1:end,:)];
                  kesisme_yeri_A=kesisme_yeri_A+1;
                  j_eski=j;
                else
                  A_alan_noktalari=[A_alan_noktalari(1:j+kesisme_sayisi-1,:);kesisme_y
kesisme_x;A_alan_noktalari(j+kesisme_sayisi:end,:)];
                  kesisme_yeri_A=j+kesisme_sayisi;
                  j_eski=j;
                end

                if i==i_eski
                  B_alan_noktalari=[B_alan_noktalari(1:kesisme_yeri_B,:);kesisme_y
kesisme_x;B_alan_noktalari(kesisme_yeri_B+1:end,:)];
                  kesisme_yeri_B=kesisme_yeri_B+1;
                  i_eski=i;
                else
                  XX=find(B_alan_noktalari(:,1)==y_1&B_alan_noktalari(:,2)==x_1);
                  B_alan_noktalari=[B_alan_noktalari(1:XX(1),:);kesisme_y
kesisme_x;B_alan_noktalari(XX(1)+1:end,:)];
                  kesisme_yeri_B=XX(1)+1;
                  i_eski=i;
                end
              else
                %eğer kesişme sayısı 1'e eşit oluyorsa
                %i_eski, j_eski: Bu değişkenler ilerideki kesişme
                %noktasının önceki kesişme noktasının ardına gelip
                %gelmeyeceğinin belirlemede kullanılacaktır. Örneğin,
                %A parseli için j. ve (j+1). noktalar arasında 2 tane
                %kesişme noktası varsa i_eski i'ye eşit olacağınfdan
                %yeni kesişme noktası önceki kesişme noktasının ardına
                %eklenmesi için kullanılacaktır. Ekleme yeri burada
                %tanımlanan kesisme_yeri_A değişkeninin hemen ardı
                %olacak şekilde belirlenecektir. Benzer durum B
                %matrisi için de geçerlidir.
                i_eski=i;
                j_eski=j;
              A_alan_noktalari=[A_alan_noktalari(1:j,:);kesisme_y
kesisme_x;A_alan_noktalari(j+1:end,:)];
              kesisme_yeri_A=j+1;

              B_alan_noktalari=[B_alan_noktalari(1:i,:);kesisme_y
kesisme_x;B_alan_noktalari(i+1:end,:)];
              kesisme_yeri_B=i+1;
              end
```

```
            end
        end
    end
end

kesisme_sayisi
parsel_2_ic=parsel_2_ic(2:end,:);
 [X_uzunluk_satir,X_uzunluk_sutun]=size(A_alan_noktalari);
 [Y_uzunluk_satir,Y_uzunluk_sutun]=size(B_alan_noktalari);

 kesisme_noktalari=kesisme_noktalari(2:end,:)

plot(B_alan_noktalari(:,1),B_alan_noktalari(:,2),'*-')
hold on
plot(A_alan_noktalari(:,1),A_alan_noktalari(:,2),'.-')
% B_alan_noktalari
% A_alan_noktalari
alan_koor=[0 0];
for i=1:kesisme_sayisi
    i
    if i~=kesisme_sayisi
        if mod(i,2)~=0

X_k1=find(kesisme_noktalari(i,1)==A_alan_noktalari(:,2)&kesisme_noktalari(i,2)==A_al
an_noktalari(:,1))

X_k2=find(kesisme_noktalari(i+1,1)==A_alan_noktalari(:,2)&kesisme_noktalari(i+1,2)==
A_alan_noktalari(:,1))

            indis1=X_k1:X_k2

          alan_koor=[alan_koor;A_alan_noktalari(indis1,:)];
        else

Y_k1=find(kesisme_noktalari(i,1)==B_alan_noktalari(:,2)&kesisme_noktalari(i,2)==B_al
an_noktalari(:,1))

Y_k2=find(kesisme_noktalari(i+1,1)==B_alan_noktalari(:,2)&kesisme_noktalari(i+1,2)==
B_alan_noktalari(:,1))
            if Y_k1>Y_k2
                indis2=[Y_k1:Y_uzunluk_satir 2:Y_k2]
            else
                indis2=Y_k1:Y_k2
            end
          alan_koor=[alan_koor;B_alan_noktalari(indis2,:)];
        end
    else
```

```
Y_k1=find(kesisme_noktalari(i,1)==B_alan_noktalari(:,2)&kesisme_noktalari(i,2)==B_al
an_noktalari(:,1))

Y_k2=find(kesisme_noktalari(1,1)==B_alan_noktalari(:,2)&kesisme_noktalari(1,2)==B_a
lan_noktalari(:,1))
            if Y_k1>Y_k2
               indis2=[Y_k1:Y_uzunluk_satir 2:Y_k2]
            else
               indis2=Y_k1:Y_k2
            end
         alan_koor=[alan_koor;B_alan_noktalari(indis2,:)];
   end
end


alan_koor=alan_koor(2:end,:)
 X_koor=alan_koor(:,2);
 Y_koor=alan_koor(:,1);

 figure
 plot(Y_koor,X_koor,'+-')
  polyarea(X_koor,Y_koor)
```

ÖZGEÇMİŞ

Emirhan ÖZDEMİR 1992 yılında İstanbul ilinin Bakırköy ilçesinde doğdu. İlköğrenimini Mustafa Kemal Paşa İlköğretim Okulunda ve ortaöğrenimini 50.Yıl İnsa Lisesi'nde tamamladı. 2010 yılında Karadeniz Teknik Üniversitesi, Fen Fakültesi, Matematik Bölümü'nde "Lisans" eğitimi almaya hak kazandı. Lisans eğitimini 07/06/2014 tarihinde tamamladı. Aynı yıl Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalı'nda "Yüksek Lisans" eğitimine başladı ve 2015 yılında Recep Tayyip Erdoğan Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Anabilim Dalı'nda başlamış olduğu "Yüksek Lisans" eğitimini 27/11/2017 tarihinde tamamladı. Şuan özel bir kurumda matematik öğretmeni olarak çalışmaktadır.