OFSETLEME VE CEP FREZELEME İŞLEMLERİNDE ANALİTİK YAKLAŞIM

Mustafa GÖKTAŞ

YÜKSEK LİSANS TEZİ

MAKİNE EĞİTİMİ

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2010

ANKARA

Mustafa GÖKTAŞ tarafından hazırlanan OFSETLEME VE CEP FREZELEME İŞLEMLERİNDE ANALİTİK YAKLAŞIM adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Abdulmecit GÜLDAŞ ……..
Tez Danışmanı, Makine Eğitimi Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Makine Eğitimi Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. Mahmut GÜLESİN ……..……
(Makine Eğitimi, Gazi Üniversitesi)
Prof. Dr. Ahmet ÖZDEMİR ……..……
(Makine Eğitimi, Gazi Üniversitesi)
Doç. Dr. Bülent BOSTAN ……..……
(Metal Eğitimi, Gazi Üniversitesi)
Yrd. Doç. Dr. Hakan DİLİPAK ……..……
(Makine Eğitimi, Gazi Üniversitesi)
Yrd. Doç. Dr. Abdulmecit GÜLDAŞ ……..
(Makine Eğitimi, Gazi Üniversitesi)

Tarih: 16/07/2010

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Bilal TOKLU ……
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Mustafa GÖKTAŞ

OFSETLEME VE CEP FREZELEME İŞLEMLERİNDE ANALİTİK YAKLAŞIM
(Yüksek Lisans Tezi)

Mustafa GÖKTAŞ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Temmuz 2010

ÖZET

Bu çalışmada CNC freze tezgâhlarında cep frezeleme operasyonları için cep profiline uygun ve hatasız takım yollarının üretilmesi amacıyla yeni bir ofsetleme algoritması geliştirilmiştir. Bu algoritmada, DXF dosya yapısından cep profiline ait veriler okunmuştur. Ofset çizgileri ve ofset yayları profili oluşturan unsurların analitik denklemleri kullanılarak hesaplanmıştır. Daha sonra hatalı kısımların atılması işlemini kolaylaştırmak için ofset çizgileri kesişim noktalarından kırılmıştır. Hatalı takım yollarının oluşmasına sebep olacak hatalı ofset kısımları atılarak ofsetleme operasyonu tamamlanmıştır. Algoritma, AutoCAD 2009 programında hazırlanan DXF verilerine dayalı olarak Delphi ortamında bilgisayar programında geliştirilmiştir. Böylece, farklı örnekler için ofsetleme algoritmasının doğruluğu kanıtlanmıştır.

Bilim Kodu : **708.3.028**
Anahtar Kelimeler : **Ofsetleme, Unsur tanıma, Cep işleme, DXF**
Sayfa Adedi : **112**
Tez Yöneticisi : **Yrd. Doç. Dr. Abdulmecit GÜLDAŞ**

ANALYTICAL APPROACH TO OFFSETTING AND POCKET MILLING
(M.Sc. Thesis)

Mustafa GÖKTAŞ

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
July 2010

ABSTRACT

In this study, a new offsetting algorithm has been developed for pocket milling operations on CNC milling machines in order to generate tool paths without error and suitable for the pocket profile. In this algorithm, pocket milling properties are read from DXF database. Offset lines and arcs are calculated by analytical equations of items that formed the profile. Then, all of the items are trimmed at the intersection points of offset for simplifying unnecessary items. Inaccurate offset items are thrown away, which would cause incorrect tool paths and then offsetting operation is ended. The algorithm works with DXF files obtained from AutoCAD 2009 and is applied using Delphi 7 program. The developed offsetting algorithm has been approved in the different applications.

Science Code : 708.3.028
Key Words : Offset, Feature recognition, Pocket milling, DXF
Page Number: 112
Adviser : Assist. Prof. Abdulmecit GÜLDAŞ

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışmanım Yrd. Doç. Dr. Abdulmecit GÜLDAŞ'a yine tecrübelerinden faydalandığım kıymetli hocam Yrd. Doç. Dr. Hakan DİLİPAK'a, ayrıca çalışmamda bana yardımcı olan tüm çalışma arkadaşlarıma, manevi destekleriyle beni hiçbir zaman yalnız bırakmayan çok değerli aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	**Açıklama**
SD	Sayısal Denetim
BSD	Bilgisayarlı Sayısal Denetim
BDT	Bilgisayar Destekli Tasarım
BDİ	Bilgisayar Destekli İmalat
NC	Numerical Control
CNC	Computer Numerical Control
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing

Semboller	**Açıklama**
x1	Başlangıç noktasının x eksenindeki koordinatı
y1	Başlangıç noktasının y eksenindeki koordinatı
x2	Bitiş noktasının x eksenindeki koordinatı
y2	Bitiş noktasının y eksenindeki koordinatı
m1	Birinci çizginin eğimi
m2	İkinci çizginin eğimi
r1	Birinci yayın yarıçapı
r2	İkinci yayın yarıçapı
a1	Birinci yayın merkezinin x eksenindeki koordinatı
b1	Birinci yayın merkezinin y eksenindeki koordinatı

a2	İkinci yayın merkezinin x eksenindeki koordinatı
b2	İkinci yayın merkezinin y eksenindeki koordinatı
d	Ofsetleme mesafesi

1. GİRİŞ

Tasarım ve imalatta ulaşılmak istenen ana hedef, en kısa zamanda, en düşük üretim maliyeti ve istenilen kalitede ürün üretimini gerçekleştirmektir. Bu amaç doğrultusunda 1950'li yıllarda NC (Numerical Control – Sayısal Denetim - SD) ve daha sonra, CNC (Computer Numerical Control – Bilgisayarlı Sayısal Denetim - BSD) tezgâhları imalat sektöründe yer almıştır. BSD tezgâhlarının imalat sektöründe kullanımıyla birlikte düşük maliyetli üretim ve ürün kalitesinde olumlu artışlar sağlanmıştır. Özellikle ölçü hassasiyeti, ürün kalitesi, üretim kabiliyeti ve zaman tasarrufu bunların başında gelmektedir. Böylelikle üretim maliyetlerinin azalması nedeni ile verimlilik artmaktadır.

Bilgisayarlı tezgâhların hızlı gelişimiyle BSD programlama konularında yapılan çalışmalar ve araştırmalar artmış ve bilgisayar teknolojisinin de bu sürece katılımı ile çeşitli CAD/CAM (Computer Aided Design / Computer Aided Manufacturing – Bilgisayar Destekli Tasarım / Bilgisayar Destekli İmalat – BDT/BDİ) sistemleri ve yazılımları geliştirilmiştir. Bu alanda BDT yazılımları ile üretilmek istenen parçanın iki boyutlu çizimleri veya üç boyutlu modelleri oluşturulurken, BDİ yazılım ve sistemleri ile de ürünün BSD takım tezgâhlarında imalatı için gerekli takım yolları türetilmektedir. Oluşturulan bu takım yolları ve belirlenen işleme parametreleri, BDT/BDİ sistemlerindeki grafiksel simülasyon yardımı ile üretim aşamasına geçilmeden önce hata ve eksikliklerin tespit edilip giderilmesine imkan sağlamaktadır.

Günümüz üretim teknolojilerinin vazgeçilmez unsuru haline gelmiş olan BSD teknolojisi insan üzerindeki kontrol yükünü büyük ölçüde azaltmıştır. Buna karşın BSD tezgâhlar için üretilecek her bir ürüne ait farklı kod yazma uğraşısı ortaya çıkmıştır. Karmaşık geometriye sahip parçaların BSD kodlarının, elle yazılması hem zaman almaktadır hem de zordur ve de hata yapmaya elverişlidir.

BSD tezgâhlar ile yapılan üretim işlemlerinde sıkça uygulanan işleme operasyonlarından biri de cep frezeleme işlemidir. Yapılan araştırmalar sonucunda karmaşık profile sahip cepler için takım yollarının türetilmesinde, cep profilinin ofsetlenmesinin, önemli bir rol oynadığı görülmüştür. Cebi işleyecek kesicinin temas verilerinin, konum verilerine dönüştürülmesi için ofsetleme işleminin hatasız bir şekilde gerçekleştirilmesi zorunluluk arz etmektedir. Ofsetleme işleminin güvenilirliği, elde edilecek takım yolları ile doğrudan ilişkili olduğu için doğru ofsetlemeyi sağlayacak başarılı bir ofsetleme algoritmasına ihtiyaç duyulmaktadır. Bu çalışmada da ofsetlemeyi başarılı bir şekilde gerçekleştirecek yeni bir ofsetleme algoritmasının geliştirilmesi amaçlanmıştır.

Geliştirilen çalışmada, iki boyutlu olarak çizilen bir cebin BSD dik işleme merkezinde boşaltılması amacıyla BSD kodlarının türetilmesi için ofsetleme yöntemi geliştirilmiştir. Cep profilinin üzerinde bulunduğu iş parçasının geometrik sınırları dikkate alınmamıştır. Kullanıcı tarafından oluşturulan ve DXF dosyasına kaydedilen cep profilinde profilde açıklığın ve üst üste binmiş unsurların olmadığı var sayılmış, kullanıcının oluşturduğu profilin ideal bir kapalı profil olduğu kabul edilerek çalışma gerçekleştirilmiştir. Yapılan çalışmada işlenecek cebi tanımlayan cep ve ada profilleri öncelikle ofsetleme hataları göz ardı edilerek kabaca ofsetlenmiştir. Daha sonra hataların giderilmesi amacı ile birbiriyle kesişen ofset unsurları kırpılmak yerine, kesişim noktalarından kırılarak her bir parça ayrı ofset unsuru olarak kabul edilmiştir. Ofset unsurlarının kesişim noktaları bulunurken ofset unsurlarının analitik denklemleri kullanılmıştır. Geçersiz ofset kısımlarının atılmasında ise diğer çalışmalardan farklı olarak, geçerli bir ofset unsurunun taşıması gereken özellikler göz önünde tutularak gerçekleştirilmiştir. Belirlenen bu özelliklere göre de tüm ofset unsurları değerlendirilerek hatalı unsurların atılıp atılmamasına karar verilmiştir. Çalışmanın aşamaları, aşağıda belirtilmiştir.

- BDT ortamında cep profilinin çizilip DXF olarak kaydedilmesi,
- DXF dosyası okutturularak unsurların algılanması,
- Unsurların sıralatılması,

- Sıralama yönünün belirlenmesi,
- Kaba ofsetleme işleminin gerçekleştirilmesi,
- Kaba ofset unsurlarının kesişim noktalarından kırılması,
- Geçersiz ofset unsurlarının atılarak ofsetlemenin tamamlanması.

2. LİTERATÜR ARAŞTIRMASI

İmalatta ulaşılmak istenen asıl amaç, en kısa sürede, en düşük maliyet ile istenilen kalitedeki ürünü üretebilmektir. Hassas, hızlı ve kaliteli imalat yöntemlerinden birisi de BSD tezgâhlar ile üretim yapılabilmektedir. BSD tezgâhlar ile üretim artık günümüzde vazgeçilmez bir üretim unsuru haline gelmiştir.

BSD tezgâhların yaygınlaşması, genel olarak endüstride, her bir parça için ayrı BSD kodlarını yazma iş yükünü doğurmuştur. BSD tezgâhların daha etkin kullanılabilmesi ve karmaşık geometriye sahip iş parçalarının imalatı için, bilgisayar ile BSD kodlarını türetilmesini sağlayan BDT/BDİ teknolojisi geliştirilmiştir [1].

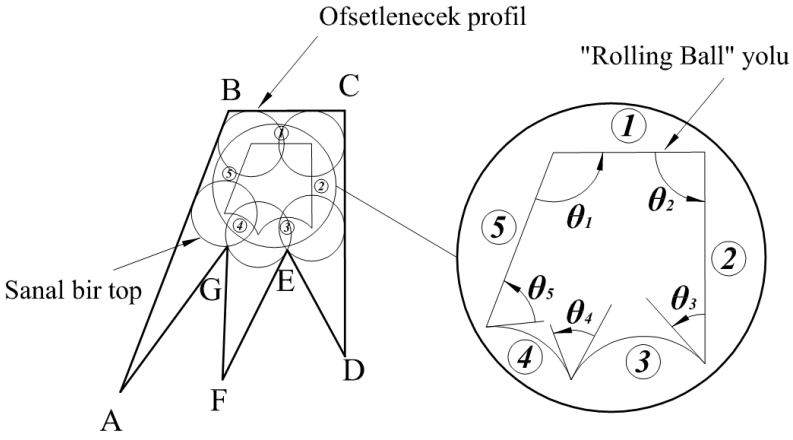
BSD tezgâhlar ile yapılan işlemlerden bir tanesi de cep frezeleme operasyonlarıdır. Her profil için farklı takım yoları değişik yazılabilmektedir. Bunlar genel olarak, tek yönlü doğrusal takım yolları, zigzag takım yolları, profil tekrarlayan takım yolları, spiral takım yolları, karma takım yolları vb. dir.

Cep profilini işleyecek takım yolu tipi, kesici takım üzerine gelen kesme kuvveti ve gerilimlerdeki değişimi etkilemektedir. Tek yönlü doğrusal takım yollarında ve zigzag takım yollarında, kesme işlemi, profil tekrarlayan takım yollarındakine göre daha kesikli ve kararsızdır. Bu durum kesici takım üzerine gelen kesme kuvveti ve gerilim değişimlerini olumsuz olarak etkilemektedir. Kesici takım üzerine gelen kesme kuvveti ve gerilim değişimleri de takımın dayanımını düşüreceği için takım ömrü de kısalacaktır. Daha uzun bir takım ömrü için profil tekrarlayan takım yolu tipi endüstride daha çok tercih edilen bir yöntemdir [2].

Profil tekrarlayan takım yolları oluşturulurken, işlenen profilin, işleme yönündeki ofseti kullanılmaktadır. Kesici takım ile profil arasındaki kesici temas verilerinin kesici konum verilerine dönüştürülmesi işleminde profilin ofsetlenmesi gerekmekte ve ofsetlemeyi gerçekleştirecek bir yönteme ihtiyaç duyulmaktadır.

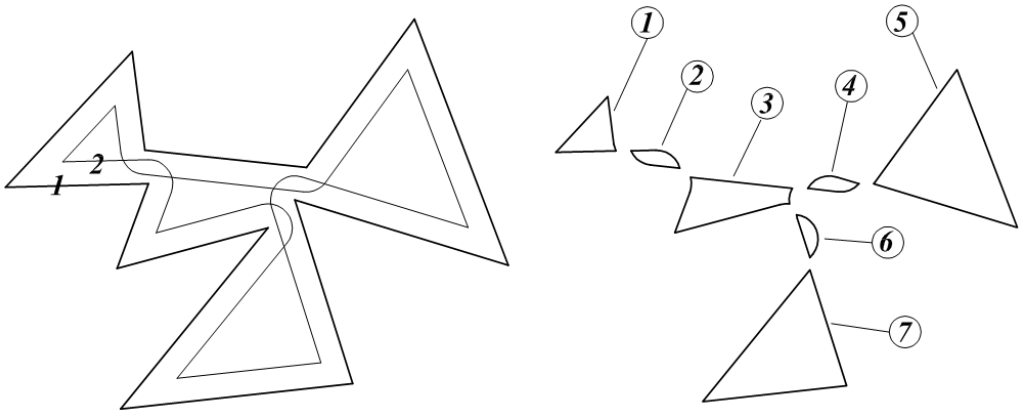
Cep profilini işleyecek takım yollarındaki hatalar iş parçasında dalma ve kalıntılara neden olmaktadır. Bu nedenle, takım yollarının elde edilmesini sağlayan ofsetleme yönteminin güvenilir olması büyük önem taşımaktadır. Ofsetleme işlemi oldukça karmaşık bir süreç olduğu için ofsetleme sırasındaki hatalar doğrudan takım yollarına yansımakta ve hatalı bir işlemeye neden olmaktadır.

Literatür araştırmalarına göre güvenilir takım yollarını üretebilecek doğru çalışan bir ofsetleme algoritmasının geliştirilmesine yönelik birçok çalışma yapıldığı gözlenmiştir. Literatürde ofsetleme problemine yönelik değişik yöntemler sunulmakla birlikte genel olarak üç farklı yöntem göze çarpmaktadır. Önerilen genel yöntemlerden birisi "Top Yuvarlama" ("Rolling Ball") yöntemidir. Bu yöntemde ofsetleme mesafesi yarıçapındaki sanal bir topun ofsetlenecek profile temas edecek şekilde, profil boyunca yuvarlandığı kabul edilmiş ve bu sanal topun merkezinin çizdiği yörünge ofsetleme sürecinde kullanılmıştır (Şekil 2.1).



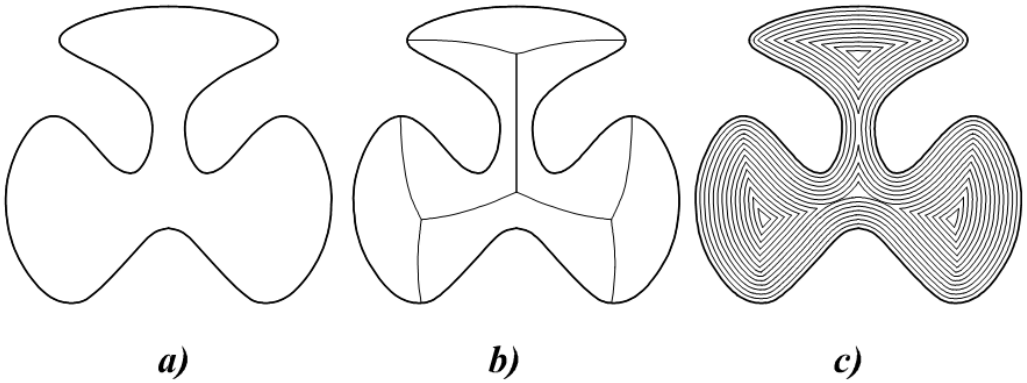
Şekil 2.1. "Rolling Ball" Yöntemi ile ofsetleme süreci [2]

Genel olarak kullanılan bir diğer yöntem, geçersiz kapalı profillerin atılmasına dayalı yöntemlerdir [3,4]. Cep profilinin ofsetlenmesi ile oluşan kapalı profillerden hangisinin geçerli hangisinin geçersiz olduğu geliştirilen algoritmalarla tespit edilerek atılmaktadır (Şekil 2.2).



Şekil 2.2. Ofsetleme sonucu oluşan geçerli ve geçersiz kapalı profiller [3]

Ofsetleme işlemi için genel olarak kullanılan 3. yöntem ise "Voronoi" eğrilerinin kullanıldığı yöntemdir. Bu yöntemde ofsetlenecek profil, doğru ve yaylar yerine parametrik eğrilerden oluşmaktadır. Bu yöntemde profil "Voronoi" eğrileri ile parçalara ayrılmış ve her bir parça ayrı ayrı ofsetlenmesi ile profilin ofsetlenerek gerçekleştirilmiştir [5-7]. Ofsetlenecek profil eğimin yüksek olduğu noktalardan kırılmış ve profilin ofsetinin, kendi kendisini kesmesi önlenmiştir (Şekil 2.3).



Şekil 2.3. 'Voronoi' eğrileri ile gerçekleştirilen Ofsetleme işlemi a) Ofsetlenecek profil b) 'Voronoi' eğrilerinin eklenmesi c) Profilin ofsetlenmesi [5]

Choi ve Park (1999) tarafından gerçekleştirilen çalışmada, ofsetleme sürecine, ofsetleme işlemini engelleyecek darlıktaki aralıkları oluşturan unsurların atılmasıyla başlanmıştır. Sonraki adımda ise kaba ofsetleme ve bunun sonucunda ortaya çıkan

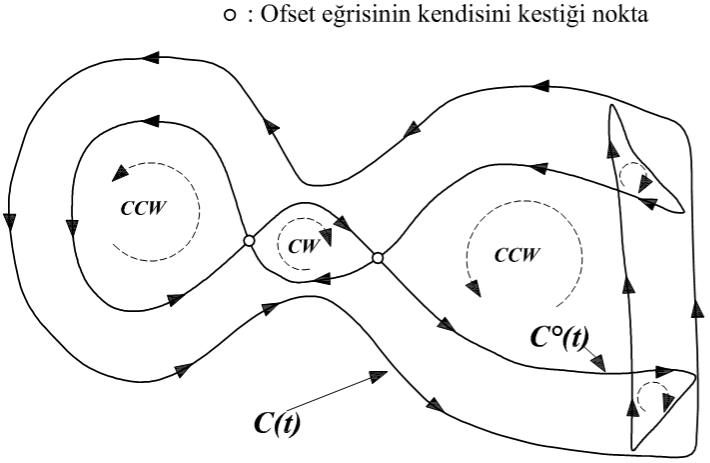
geçersiz kapalı alanları oluşturan ofset unsurlarının atılmasıyla ofsetleme işlemi tamamlanmaktadır [8].

Liu, Yong, Zheng ve Sun (2007) tarafından yapılan çalışmada ofsetlenecek unsurlarının kapalı profil oluşturması istenmemiş ve girdinin "polyline" (çoklu çizgi) olduğu varsayılmıştır. Kapalı profil oluşturmayan, açık eğrilerin ofsetlenmesini de sağlayan bu yöntem, eğri elemanlarının ofsetlenmesi, bu ofsetlemelerin duruma göre budanması veya birleştirilmesi ve bir kırpma algoritmasıyla geçersiz ofset unsurlarının kırpılması olmak üzere üç aşamadan oluşmaktadır [9].

Choy ve Chan (2001) tarafından yapılan çalışmada, ofsetleme mesafesinin takımın yarıçapından daha fazla olduğu durumlarda takımın profilin dışbükey köşelerine tam olarak yanaşamamasından kaynaklanan kalıntıları gidermek için ek takım yolları ekleyen bir yöntem geliştirmişlerdir [10].

Kim, Lee ve Yang (2006) isimli araştırmacılar tarafından yapılan çalışmada işleme esnasında takıma gelen kesme kuvvetini daha düzgün, keskin dönüş hareketlerindeki titreşimleri azaltmak ve talaş kaldırma oranını artırmak için genel yöntemler ile elde edilen takım yollarına eklemeler yapmışlardır [11]. Bu yöntemle kesici takım için daha büyük yana kayma mesafeleri kullanılabilmekte ve bu sayede toplam takım yolu uzunluğu kısaltılmış olmaktadır.

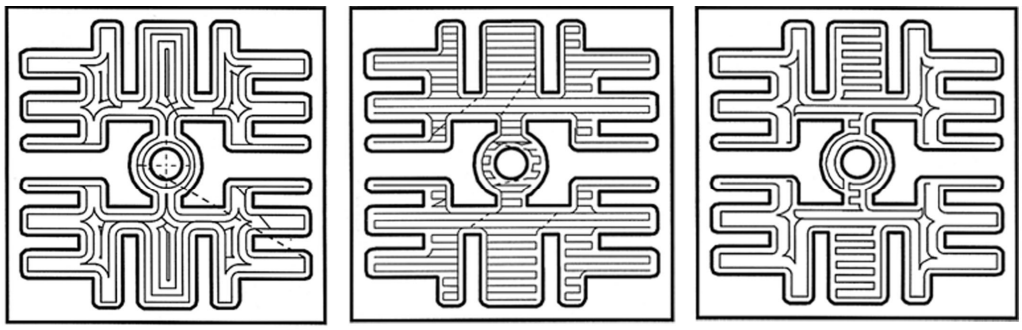
Lee'nin (2003) yaptığı çalışmada bir cep profilinin ofsetlenmesi ile oluşan kapalı alanlardan hangisinin geçerli hangisinin geçersiz olacağına yönelik olarak bir yöntem geliştirilmiştir [12]. Bu yönteme göre ofsetlenen cep profili için saat yönü ya da tersinde bir yön tayin edilmiştir. Ofsetleme ile oluşan kapalı profillerden sıralama yönü ofsetlenen profilin yönü ile aynı olan kapalı alanların geçerli, farklı olan kapalı alanların da geçersiz olduğu belirtilmektedir (Şekil 2.4).



Saat yönünde (CW) kapalı alan: içiçe geçmiş kapalı alan (geçersiz kapalı alan)
Saat tersi yönünde (CCW) kapalı alan: ofset eğrisinin parçası olan kapalı alan (geçeli kapalı alan)

Şekil 2.4. Ofsetleme ile oluşan profillerin yönleri [12]

Park, Chung ve Choi (2003) tarafından yapılan çalışmada takım yollarını optimize etmek amacıyla işlem esnasındaki takım geri çekilmelerini en aza indirmek için ofsetleme ile elde edilen takım yollarına eklemeler yaparak bu takım yollarını bir birlerine bağlamışlardır. Bu çalışmada takımın iş parçasına yanaşarak bir defada cep frezeleme işlemini tamamlaması amaçlanmıştır [13].



Şekil 2.5. Takımyolu tipleri a) Profil tekrarlayan, b) Doğrusal, c) Karma takım yolları [14]

Yao ve Gupta (2004) yaptıkları çalışmada, zig-zag takım yolları ve profil tekrarlayan takım yollarının her ikisini de kullanarak cep profilini işleyebilen, karma bir takım

yolu tipi önermişlerdir [14]. Bu yöntemle elde edilen takım yolu uzunluğunun diğer yöntemlere göre daha kısa olduğu belirtilmektedir (Şekil 2.5).

Zhao, Wang, Zhou ve Qin (2007) tarafından yapılan çalışmada işlenen cep profilinin keskin köşelerinde oluşan malzeme kalıntılarını temizlemek ve takımın köşelere daha iyi yanaşabilmesi için genel yöntemler ile elde edilen takımyollarına eklemeler yapılmıştır. Önerilen yöntemin doğruluğu yapılan deneysel uygulama ile kanıtlanmıştır [15].

Sheen ve You'nun (2006) gerçekleştirdikleri çalışmada profilin ofsetlenmesi ile oluşan hatalı kesişimlerin giderilebilmesi için dinamik bir yöntem sunulmuştur. Bu yöntemde ofset unsurları oluşturulurken bir başka ofset unsuru ile veya ana profil ile kesişip kesişmediği göz önünde tutularak hatalı kesişmelerin oluşmadan önüne geçilmesi amaçlanmıştır. Ayrıca bu çalışmada cep profili içerisinde yer alan adaların da dalmalara yol açmadan işlenebilmesi için ada ve cep profillerinin ofset grupları arasında köprüler oluşturularak takım yolları bir birine bağlanmıştır [16].

Chuang ve Lin (1997) tarafından yapılan çalışmada serbest formlara sahip eğrilerden oluşturulmuş cep profilleri ilk önce parametrik olan 'B-spline' veya 'Bezier' eğrilerine dönüştürülmüş daha sonra bu eğrilerin ofsetlenmesi ile oluşan çıktıları kullanarak cebi işleyecek doğrusal takım yollarını türetmişlerdir [17].

Yan, Shuilai ve Shuiguang (2000) isimli araştırmacılar tarafından yapılan çalışmada genel yöntemler ile oluşturulan takım yollarında yana kayma mesafesinin takım yarıçapından büyük olduğu durumlarda kalıntıların oluştuğuna dikkat çekilmiştir. Bu problemin giderilmesi için de "Düzgün olmayan ofsetleme" tekniğini kullanmışlardır. Buna göre köşeler ofsetlenirken yana kayma mesafesinin takım yarıçapından büyük olması durumunda köşelere daha fazla yanaşan ek yollar kullanarak köşelerdeki kalıntıları önlemişlerdir. Bu yöntemde de yana kayma mesafesi artırılarak toplam takım yolunu kısaltmışlardır [18].

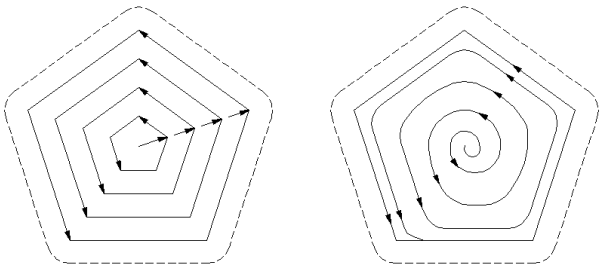
Lambregts ve arkadaşları (1995) tarafından yapılan çalışmada, sabit derinlikli karmaşık forma sahip ceplerin işlenmesi için gerekli olan BSD takım yollarını otomatik olarak oluşturabilen, etkili takım yolu oluşturma yöntemi sunulmaktadır. Cepler, doğru ve yay parçalarından oluşabilir ve işlenmeyecek adalara sahip olabilirler. Gerçekleştirilen cep profiline paralel takım yolu oluşturmak için algoritmalar kullanılmıştır [19].

Wong, T. ve Wong K. (1996) tarafından yapılan çalışmada, birden fazla adaya sahip karmaşık forma sahip ceplerin frezelenmesi için takım yolları oluşturmada kullanılabilecek bir yöntem önerilmiştir. Bu yöntemde cep ve ada profilleri doğru ve yaylardan oluşan dışbükey ve içbükey geometrilerden oluşmaktadır. Diğer cep frezeleme çalışmalarına göre, bu çalışmanın en belirgin farkı, cep ve ada profillerinin, algoritmanın ilk basamağında işlemlerinin yapılmış olmasıdır. Kesici takımın geri çekilme hareketi yapmadan işlemesini sağlayan takım yolları oluşturmak için dış hat köprülerinden yararlanılarak, cep ve ada profilleri bir birine bağlanmıştır [20].

Korkut ve Tekiner (2000), Delphi programında hazırladıkları bir yazılım ile IGES dosya yapısından üretilecek parçaya ait verileri almışlardır. BDT ortamında tasarlanan silindirik iş parçalarının, tasarım aşamasından üretim aşamasına kadar olan süreci planlamışlardır. Öncelikle, IGES veri yapısından, iş parçasının unsurlarına ve boyutlarına ilişkin veriler okunduktan sonra bulunan unsurlara göre kesici takım seçimini yapmışlar ve kesme parametrelerini belirlemişlerdir. Sonraki adımda Fanuc işletim sistemine uyumlu BSD kodlarını türetmişlerdir. Ayrıca türetilen BSD parça programlarının işlem sıralamasını ve parçaya ait takım yolu simülasyonunu oluşturmuşlardır [21].

Bieterman ve Sandstrom, düzlemsel ceplerin işlenmesi için yeni bir takım yolu oluşturma metodu geliştirmişlerdir. Geliştirdikleri yöntemde bir cep profili içerisinde tanımlanan eliptik kısmi diferansiyel denklem sınır değeri probleminin çözümünü kullanmışlardır. Önerdikleri bu matematiksel fonksiyon cep profilinin formuna

uyumlu ve cep içinde iç içe düz eğimli spiral yollar şeklinde ortaya çıkmaktadır. Yapılan bu çalışma ile tüm metal malzemelerin işleme zamanında ve kesici takım aşınmasında önemli kazanç sağlamışlardır. Ayrıca geliştirdikleri yöntemde ayrıca değişken bir ilerleme optimizasyonu prosedürü yapılmıştır. Bu işlem, takım yolunu ve makine kısıtlamalarını içine almaktadır ve her takım yolu için makine performansını en yüksek yapmak için uygulanabilmektedir. Şekil 2.6'da cep boşaltma işlemi için Klasik ve geliştirilen Curvelinear takım yolları görülmektedir [22,23].



Şekil 2.6. Cep boşaltma işleminde takım yolunun curvelinear yöntemi ile optimizasyonu a) klasik ve b) curvelinear takım yolları [22,23]

Bieterman ve Sandstrom, geliştirdikleri bu yeni takım yolu metodunu klasik takım yolu ile karşılaştırmak amacı ile 5 adet farklı poligonal cep geometrisini iki farklı kesici takımla işleyerek işleme zamanındaki ve takım aşınmasındaki azalmaları 14 farklı kesme hızında denemişlerdir. Sonuç olarak Bieterman ve Sandstrom geliştirdikleri bu yeni yöntem ile yapılan denemelerde, işleme zamanlarını %10-%30 değerleri arasında azaltabilmişlerdir. Ayrıca sert malzemelerin işlenmesinde takım ömrünü klasik takım yoluna göre bu yöntem ile yaklaşık olarak iki kat kadar artırabilmişlerdir [22,23].

Yang, Joneja ve Zhu'nun yaptığı iki aşamalı çalışmada, işlem planlama ve unsur tanımada işleme zamanı optimizayonu için genellenmiş ceplerin tanımlanması amacıyla yeni bir yöntem geliştirmişlerdir. Geliştirdikleri bu yöntemi, daha önce yapılan çalışmalardaki dikdörtgen şeklindeki cepler, kanallar ve delikler gibi basit şekilli unsurların çıkarılması ve işlem planlaması yönünden dolayı ayrı tutmuşlardır.

Ayrıca, daha karmaşık şekilli işlenebilir unsurları tanımaya ve aynı zamanda akıllı işlem planlama yöntemi ile toplam işleme zamanını azaltmaya çalışmışlardır. Çalışmalarının birinci kısmında işleme unsurlarının çıkarılmasındaki yöntemleri ele almışlardır. Geliştirdikleri algoritmaları örneklerle desteklemişler ve tanıma sistemini, birçoğu kalite testi içeren, 70'den fazla endüstriyel parçada başarılı bir şekilde denemişlerdir. Çalışmalarının ikinci kısmında ise çoklu bir takım işleme planlama tekniği tanımlamışlardır. Bu amaçla işleme planlayıcı tabanlı bir strateji geliştirmişler ve bunu önceki unsur tanıma sistemi ile birleştirmişlerdir [23- 25].

Göktaş, Dilipak ve Güldaş (2009) tarafından yapılan bir çalışmada kapalı profilin ofsetlenmesi ile oluşan geçersiz ofset unsurlarının ayıklanabilmesi için öncelikle tüm ofset unsurları kesişim noktalarından kırılmıştır. Geçerli ofset unsurlarının ortak özellikleri tespit edilmiş ve bu özellikleri taşımayan ofset unsurları atılmıştır. Geçersiz ofset unsurlarının atılmasıyla geriye kalan geçerli ofset unsurlarıyla ofsetleme tamamlanmıştır [26].

Göktaş, Dilipak ve Güldaş (2010) tarafından yapılan bir diğer çalışmada ise kapalı profili oluşturan unsurların analitik denklemleri kullanılarak profilin ofset unsurları elde edilmiştir. Daha sonra dalma ve kalıntılara neden olan hataları gidermek için geçerli bir ofset unsurunun taşıması gereken ortak özellikler belirlenmiş ve bu özellikleri taşımayan unsurlar atılarak hatalar giderilmiştir. Geliştirilen bu yöntem ile sunulan diğer çalışmalara göre daha pratik ve kararlı bir hata giderme uygulaması gerçekleştirilmiştir [27].

Pateloup, Duc ve Ray (2010) tarafından yapılan çalışmada iki boyutlu düzlemsel cepler için yeni bir yöntem sunulmuştur. Bu yöntemde takım yolları oluşturulurken doğru ve yayların yerine uniform olmayan kübik B-spline eğrilerini kullanmışlardır. Geliştirdikleri denklemler ile B-spline eğrilerinin kontrol noktalarının koordinatlarını hesaplatarak takım yolu eğrisini oluşturmuşlardır. Önerdikleri bu yöntem ada içermeyen cep profilleri için uygun olup adalı cepler de kullanılmamaktadır. Yapılan

bu çalışmada değişik örnek uygulamalarla geliştirilen yöntemin başarısı ortaya konmuştur [28].

Held ve Spielberger (2009) tarafından yapılan bir başka çalışmada ise yine ada içermeyen, sınır hatları doğru ve yaylardan oluşan cepler için spiral takım yollarının türetildiği bir metot sunulmuştur. Takım yolu, kullanıcının belirleyebildiği cep profili içinde yer alan herhangi bir noktadan başlayan ve profilin dışında biten bir eğri şeklinde oluşmaktadır. Bu yöntemde takım yollarının kesinlikle birbirini kesmemesi ve takımın geri çekilme hareketinin olmadığı belirtilmektedir [29].

Yukarıda yapılan literatür araştırmalarında görüldüğü gibi bir iş parçası üzerindeki adalı veya adasız cep profillerini işlenebilmesi için BSD kodlarını türetebilecek yöntemler sunulmuştur. Yapılan bu çalışmada ise profil tekrarlayan takım yollarının türetilebilmesi için bir ofsetleme algoritması geliştirilmiştir. Geliştirilen ofsetleme algoritmasında ofsetleme ile oluşan hataların giderilmesi için bir hata ayıklama yöntemi kullanılmıştır. Ofsetleme işlemi sonucu elde edilen çıktılar ile cep profilini işleyecek takım yolları ve BSD kodları türetilmiştir. Bu çalışmada önerilen yöntem ada içiren cep profilleri için de takım yollarının türetilmesini sağlamaktadır.

3. KAVRAMSAL TEMELLER

3.1. Freze Tezgahları ve Frezeleme İşlemleri

Makina imalat sektöründe üretim için kullanılan en yaygın takım tezgâhlarından birisi de freze takım tezgâhlarıdır. Üzerinde birden fazla kesicinin bulunduğu kesiciler ile malzeme üzerinden talaş kaldırmak suretiyle iş parçalarının üretildiği takım tezgâhları freze takım tezgâhı olarak bilinmektedir.

Freze tezgâhlarında, düzlem yüzey frezeleme, kanal frezeleme, cep frezeleme gibi iş parçaları üzerinde bulunan girinti ve çıkıntıların oluşturulmasını sağlayan çok çeşitli talaş kaldırma işlemleri geçekleştirilebilmektedir [30].

3.2. Bilgisayarlı Sayısal Denetim (CNC- ComputerNumerical Control)

Bilgisayar teknolojisinin gelişmesi ile SD'li tezgâhların bilgisayar adaptasyonu sağlanarak yeni bir tezgâh komut işleme sistemi ortaya çıkmıştır. Bilgisayarlar SD'li tezgâhlara şu özellikleri kazandırmışlardır:

1. Bilgilerin saklanması
2. Bilgiler üzerindeki değişiklik ve düzeltmelerin daha kolay yapılabilmesi
3. Yapılan programların simülasyonlarının (benzetimlerinin) yapılabilmesi
4. Otomasyon sistemleri ile birlikte çalışabilme
5. Bilgisayar destekli imalattaki gelişmelerden sonra diğer program yazılımlarında oluşturulan verileri işleyebilme [31].

3.2.1. BSD 'li takım tezgâhlarının avantajları

BSD'li takım tezgâhlarının endüstride yaygınlaşması ile kalite, maliyet, üretim kolaylığı ve kabiliyetleri gibi birçok yönde önemli avantajlar kazanılmıştır. Bunu yanında BSD'li tezgâhların sağladığı diğer avantajlar aşağıda verilmiştir.

1. Klasik tezgâhlarda kullanılan bazı bağlama kalıbı, mastar vb. gibi elemanlara göre tezgâhın ayarlama süresi çok kısadır.
2. Ayarlama, ölçü kontrolü, el kontrolü vb. için oluşan zaman kayıpları en aza indirilmiştir.
3. İnsan faktörünün imalat sürecindeki etkisi azaldığı için seri, hassas ve özdeş parça imalatı mümkündür.
4. El becerisi gerektirmemektedir.
5. Tezgâhın çalışma temposu her zaman yüksek ve aynıdır.
6. Parça imalatına geçiş daha hızlıdır.
7. Parça üzerinde yapılacak değişiklikler sadece programın ilgili bölümünde ve tamamı değiştirilmeden seri olarak yapılabilmektedir.
8. Daha dar tolerans arlığında üretim yapılabilmektedir.
9. Çalışma devrinin istenilen aralıklarda seçilebilmesi.
10. Sabit kesme hızı ve değişken devir sayısı ile işleyebilme.
11. Hata ve problemlerin uyarıcılar sayesinde daha çabuk ve kolay çözümü.
12. Daha güvenli bir çalışma imkânı sunabilme.
13. Daha yüksek kesme hızlarında çalışabilme.

Günümüzde BSD tezgâhlar endüstrinin hemen hemen her alanında kullanılmakla birlikte en yaygın olarak talaşlı imalat sektöründe kullanılmaktadır. Üç eksenli bir freze tezgâhı ilk kez 1952 yılında çalıştırıldığında bu tezgâh o günkü bazı imalat problemlerinin çözümünü sağladığı için çok mükemmeldi. Freze tezgâhına uygulanan bu sistemler daha sonra torna, taşlama vb. takım tezgâhlarına da uygulanmıştır. Günümüzde imalatın yapıldığı hemen hemen her alanda CNC kullanılmaktadır [32].

3.2.2. CNC tezgâhlarda koordinat sistemleri

CNC tezgâh ve sistemlerinde takım yolları kartezyen koordinat sistemi referans alınarak matematiksel bağıntılarla ifade edilir. Bu sistemde X, Y, ve Z ile gösterilen üç doğrusal hareketin yanı sıra A, B, C ile simgelenen üç dönme hareketi olmak

üzere toplam altı eksen vardır. Koordinat sistemi tek bir düzlemi ifade eden iki eksenli veya üç düzlemi gösteren üç eksenli olabilir. Iki eksenli koordinat sisteminin eksenleri X-Y, Y-Z, veya X-Z, üç eksenli sistemin eksenleri X, Y, Z şeklinde ifade edilir. Bunun yanı sıra iki düzlemde, nokta konumunu uzunluk ve açı ile veren polar koordinat sistemi de kullanılır [23].

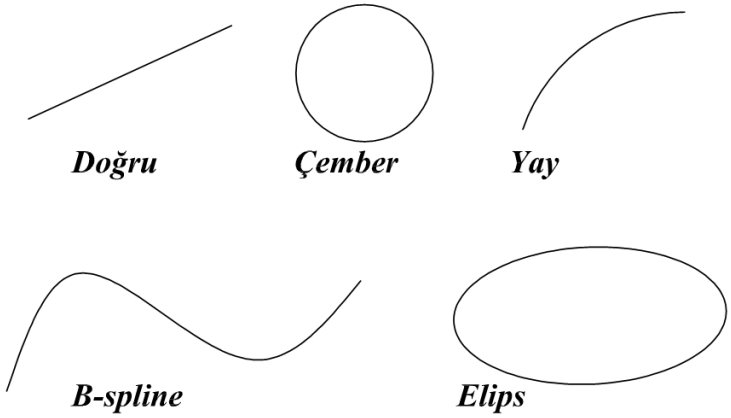
CNC tezgâh ve sistemlerde tezgâh, parça ve takım olmak üzere üç ayrı koordinat sistemi vardır. Bu koordinat sisteminin orijinlerine; tezgâha ait olanına tezgâh sıfır noktası, parçaya ait olanına parça sıfır veya program referans noktası, takıma ait olanına takım sıfır noktası denilir. Tezgâh eksenlerinin orijini yani tezgâhın sıfır noktası tezgâhın imalatı sırasında tayin edilir. Dolayısıyla sabittir ve değiştirilemez. Bir de CNC freze tezgâhlarının imalatında belirlenen takım değiştirme noktası vardır. Takım sıfır noktasının yeri de tezgâhın imalatı sırasında belirlenir, ancak konumu, takım hareket ettiği için değişkendir [33].

3.3. Bilgisayar Destekli Tasarım -BDT (Computer Aided Design - CAD)

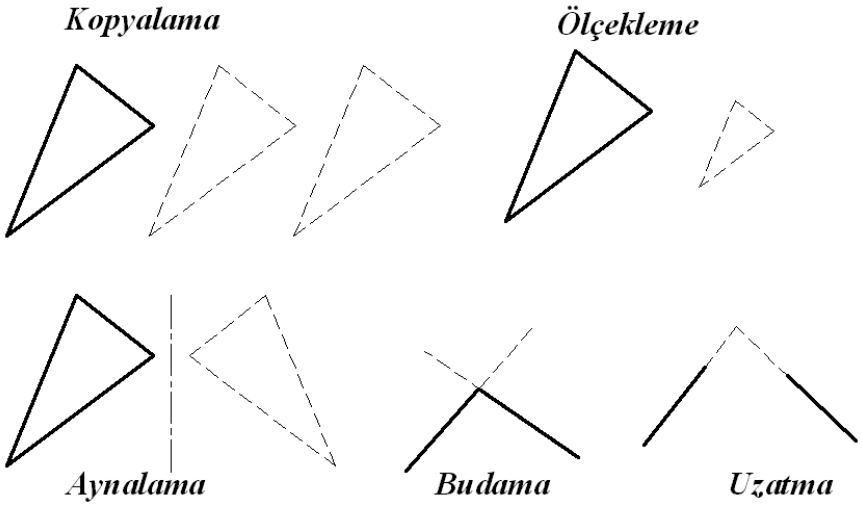
Bilgisayar destekli tasarım, tasarım sürecinde bilgisayarın kullanılması demektir. Klasik mantığın olumsuzluklarını ve yetersizliklerini gidermek için BDT mantığı geliştirilmiştir. Klasik bir teknik resim çizimi, teknik resim alet ve çizim takımı gibi nesneler kullanılarak elle yapılırken bilgisayar destekli tasarım da ise bilgisayar yazılımları kullanılarak bilgisayar ortamında yapılmaktadır.

3.3.1. İki boyutlu çizim

Bütün bilgisayar destekli tasarım sistemlerinin temelini iki boyutlu çizim oluşturmaktadır. İki boyutlu açıklanması yeterli olan tasarımlarda, çizimin oluşturulması, tekrarlanması, kopyalanması ve üzerinde değişiklikler yapılması işlemleri çok kolay bir şekilde gerçekleştirilebilmektedir. BDT yazılımları ile oluşturulabilen bazı nesneler ve gerçekleştirilebilen bazı uygulamalar Şekil 3.1'de Şekil 3.2'de görülmektedir.



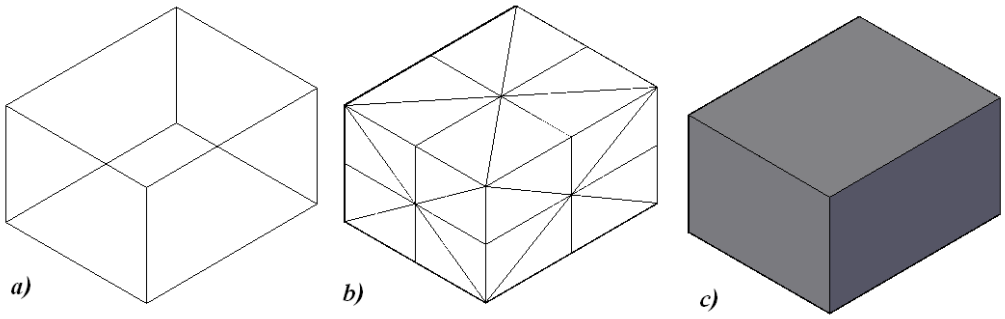
Şekil 3.1. CAD yazılımları ile oluşturulabilen bazı nesneler



Şekil 3.2. BDT yazılımları ile gerçekleştirilebilen bazı işlemler

3.3.3. Üç boyutlu modelleme

İki boyutlu çizimlerden yola çıkılarak cisimlerin gerçeğe yakın görüntüleri olan üç boyutlu görünüşleri BDT yazılımlarıyla oluşturulabilmektedir. Bu sayede iki boyutlu çiziminde fark edilmeyen tasarım hatalarının üretimden önce giderilmesi mümkün olmaktadır. Üç boyutlu modellemede genellikle tel kafes modelleme, yüzey modelleme ve katı modelleme olmak üzere üç farklı yöntem kullanılmaktadır (Şekil 3.3).



Şekil 3.3. Üç boyutlu modellerler a) Tel kafes model, b) Yüzey model, c) Katı model

3.3.4. Bilgisayar destekli tasarımın avantajları

Bilgisayar destekli tasarımın genel olarak sağladığı avantajlar:

1. İki boyutlu ve üç boyutlu çizimler çok daha kısa bir sürede çizilebilmektedir.
2. BDT yazılımlarının görüntüyü büyütme özelliği sayesinde çizimlerin ince detayları çok daha kolay ve hassas çizilebilmektedir.
3. Ölçülendirme çok daha kolay ve hızlıdır.
4. Bazı BDT yazılımları standart makine elemanlarının çizimlerini hazır olarak sunmaktadır.
5. BDT ortamında oluşturulan çizimlerin saklanması daha kolaydır.

3.4. Bilgisayar Destekli İmalat - BDİ (Computer Aided Manufacturing - CAM)

Bilgisayar destekli imalat, bir iş parçasını üretim sürecinde bilgisayar ve bilgisayarlı sayısal kontrollü tezgâhların koordineli olarak kullanılmasıdır. BDİ mantığının temelinde BDT teknolojisi ile çizimi yapılan iş parçalarının BSD tezgâhlarda üretilebilmesi için gerekli SD kodların yine bilgisayar yardımı ile türetilmesi yatmaktadır. BDİ teknolojisinin sağladığı en büyük kolaylık BSD tezgâhlarda üretimi yapılacak karmaşık geometriye sahip iş parçaları için uygun SD kodlarının çok kısa bir sürede ve hatasız olarak türetilmesidir. BDİ teknolojisinin gelişmesi ile

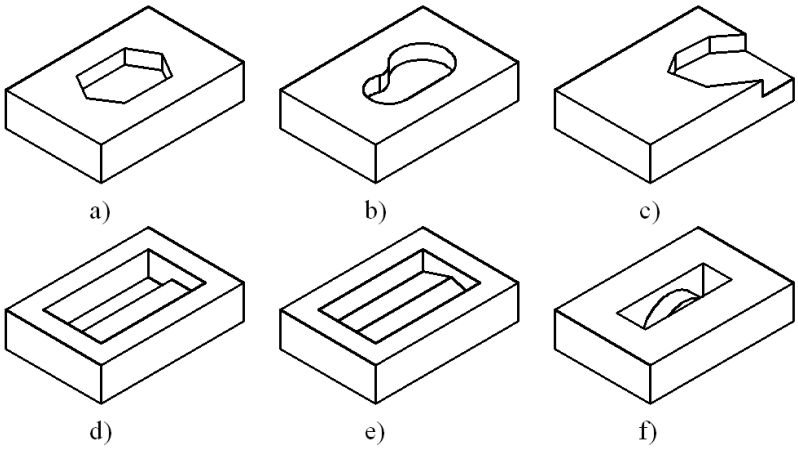
BSD tezgâhlarda en çok kullanılan işleme tipleri, düzlem yüzey işleme, çevresel işleme, cep işleme, delik delme, üç boyutlu yüzey işleme, vida çekme vb. dir.

Mekanik parçaların ve kalıpların imalatında 2.5D şekillerin işlenmesi büyük bir öneme sahiptir. Mekanik parçaların işlenmesi için frezeleme operasyonlarının yaklaşık %80'i sadece 2.5 eksenli frezeleme işlemleri kullanılarak yapılabilmektedir [34]. 2.5 eksenli işlemlerde kesiciler eşzamanlı olarak sadece ana kartezyen düzlemleri olan XY, YZ ve ZX koordinatlarında eşzamanlı olarak hareket ettiği iki eksende interpolasyon yapabilirler. 2.5 eksenli operasyonlar ile noktadan noktaya çevresel frezeleme, cep-ada operasyonları ve delik delme gibi frezeleme operasyonlarının önemli bir kısmı gerçekleştirilebilmektedir. CNC operatörü için delik delme, kılavuz çekme ve kanal açma gibi noktadan noktaya tanımlanan operasyonları programlamak kolay olmaktadır. Çünkü CNC çevrimleri ile makro olarak programlanabilmekte ve geometrik problemler oluşmaktadır. Çevresel frezeleme düzlemsel bir operasyondur ve genelde tanımlanan bir parça etrafında sürekli kesme hareketidir. Tanımlanan dış hatların tamamını veya alanı temizlemeyi amaçlayan işleme yöntemi cep işleme olarak tanımlanır. Bazı cepler dış ve iç profilleri ile tanımlanırlar. İç profiller, adalar, delikler veya boşluklar olabilir. Cep işleme genellikle zor ve karmaşık bir işlem olarak bilinir bundan dolayı çok basit şekillerin dışında nadiren elle programlanırlar.

Her ne kadar modern imalatta CNC kontrolörlerinin fonksiyonları çok gelişmiş olsa da programlanması zordur ve elle programlama kullanıldığında programcı için hata yapma olasılığı fazla ve zaman almaktadır [35]. Bundan dolayı otomatik takım yolu oluşturma ve programlama, daha kısa programlama zamanı, optimum takım yolları, uygun kesme parametreleri ve hatasız işleme kodları sağlayarak verimliliği artırabilmektedir. Böylece, programcının uzun ve sıkıcı programlama çabasını da rahatlatabilmektedir. Daha fazla otomatikleştirilmiş imalata ve artan karmaşık CAD sistemlerinin kullanımına olan ihtiyaçtan dolayı son yıllardaki araştırmalar delikler, kanallar ve cepler gibi tanımlı şekilleri işlenebilir yapmak için CAD verilerinden şekillerin çıkarılması ve tanımlanması üzerine yoğunlaşmıştır [36].

3.4.1. Ceplerin geometrik sınıflandırması

Cepler bir veya daha fazla profil veya yüzey ile tanımlanan geometrik yapılardır. İç profiller tabanlarının, işleme seviyesinin aşağısında veya yukarısında olup olmadığına bağlı olarak adalar gibi çıkıntı veya delik ve boşluk gibi negatif olabilirler. Cep şekilleri, parçanın tamamındaki pozisyonu ve amacına bağlı olarak bir parçadan diğerine değişebilir. Bundan dolayı matematiksel gösterimleri çok basit veya çok karmaşık olabilir. Cep şekilleri Şekil 3.4'de gösterildiği gibi altı ana sınıfta gruplandırılabilir. Şekil 3.4.(a)'da ki düzlem cepler, genellikle düzlemsel tabanlı ve dikey duvarlı doğru ve yay parçaları ile tanımlanırlar. Şekil 3.4.(b)'deki serbest şekil duvarlı cepler düzlem cep sınıfının bir uzantısını temsil etmektedirler çünkü serbest yapılı eğriler serbest şekil duvarlı ceplerin tanımlamalarına dahil edilebilirler. Şekil 3.4.(c)'deki asıl veya köşe cep şekilleri kapalı değillerdir çünkü genellikle cep işleme olarak farz edilirler. Şekil 3.4.(d)'deki gibi çok girişli yüzey cepler tabanın birden fazla yatay yüzeylerden meydana geldiği bir cebi tanımlarlar. Şekil 3.4.(e)'deki çok yüzeyli taban cepler, cebin giriş yüzeylerinin eğimli olduğu ceplerdir. Şekil 3.4.(f)'deki gibi bazen düzlemsel profillerin olduğu oyuklarda kullanılan serbest şekil tabanlı cepler serbest bir şekil üzerine tasarlanırlar. Bu altı sınıflandırma adaları da içerebilir veya içermez. Adalar farklı yüksekliklerde olabilirler ama basit bir ada içinde yükseklik genellikle aynıdır. Adaların varlığı takım yolu hesabı işlemini büyük oranda etkiler. Aynı zamanda bir veya daha fazla küçük cepler başka bir cep içerisinde bulunabilirler. Böyle bir durumda cebin tamamı yuvalanmış cep olarak adlandırılırlar.



Şekil 3.4. Cep şekilleri

Pratikte CAD sistemleri belirgin olarak imalata özgü otomatik işlem planlama için gerekli verileri sağlamazlar [37]. Örnek olarak cep işleme durumunda cep işleme sistemlerinin çoğu, parçalar tasarımcı tarafından yüzeyler ve hacimler açısından tanımlanırken, topolojik olarak belirsiz olmayan 2D şekiller gerektirirler. Bundan dolayı kullanıcı etkileşimli olarak dış hatları bir noktalama aracı ile seçerek veya kapalı bir düzlemsel alanı belirterek tanımlamalıdır. Bununla beraber böyle bir durumda sistem Şekil 2.1.(c)'de gösterildiği gibi tamamen belirtilmemiş bir cebi tanıyamaz [38].

İşleme unsurlarının otomatik çıkarımı CAD ve CAM arasındaki etkileşimin otomatikleştirilmesiyle CAPP sistemlerinde önemlidir. Geniş çaplı olarak cep tanımlamaları hem özel bir sistem tanımı veya hem de katı bir modelin veya bir düzlem ve bir yüzeyin kesişiminden veya bir takım yüzeylerin tarafsız dosyalarından elde edilebilirler. Wang ve diğerleri cep geometrisini bir B-rep temsili kullanarak IGES (Initial Graphics Exchange Specificaiton) dosylarından çıkarırken Liang ve diğerleri bu iş için STEP (Standard for The Transfer and Exchange of Product model data) dosyalarını kullanmışlardır [39,40].

Kesme derinliği ve bölümleme katman kesişiminin dış hatları parça ve malzeme yüzeyi ile çıkarılacak hacmi verir. Boşaltılmış yüzeylerin kaba işleminde katmanlı

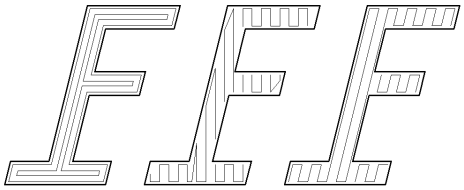
algoritma için bölümleme Z eksenine dik olan bir takım düzlemler ile yapılacağı farz edilir. Malzeme alt düzleminin Z koordinatı belirlenir (Z_{min}). Malzeme üst düzleminin Z koordinatı girilir (Z_{max}). Bölümleme yüksekliği veya kesme derinliği verilir (h). Z_{max}-h, z düzlemini verecektir. z>Z_{min} durumunda, malzeme ve istenilen bitmiş parça ile z = h yatay düzleminin kesişimi ile oluşturulan dış hatlar hesaplanır. Bu elemanlar bir cebi tanımlamak için ayrılırlar. Böylece 2.5D cep işleme operasyonu gerçekleştirilir (z = z-h). Kesme derinliği h genellikle sabit olarak hesaplanır ve işleme kataloglarından elde edilebilirler. Dong ve Vickers böyle bir varsayımın geçerliliğini göstermişlerdir ve kesme derinliği dağılımının kaba işleme zamanının azaltılmasına etkisinin çok az olduğunu belirtmişlerdir [41]. Bununla beraber kesme katmanlarının uygun bir seçimi işleme zamanını önemli bir şekilde azaltabilir. Dong ve arkadaşları müsaade edilen en büyük kesme torku veya kesme gücü ile birlikte aynı zamanda en etkili kesme hızı, ilerleme ve optimizasyon teknikleri ile elde edilen kesme derinliklerini kullanmışlardır [41].

3.4.2. Takım yolu oluşturma

Cep frezeleme için takım yolu modelleri altı stratejide sınıflandırılabilirler.

1. Spiral tip olarak da bilinen çevresel paralel takım yolu modelinde kesici takım spiral hareketle cep elemanlarının ofsetlerini içeriye doğru dış hatlar için azaltarak ve dışarıya doğru adalar için artırarak takip etmektedir (Şekil 3.5.a).

2. Paralel yön, merdiven, tarama veya süpürme frezeleme olarak da bilinen zigzag frezelemede kesici takım bir başlangıç noktasından gösterilen yöne doğru cep alanını tarayarak referans bir doğruya paralel eşit mesafeli yollarla hareket etmektedir (Şekil 3.5.b,c).



a) b) c)

Şekil 3.5. Cep işleme stratejileri

3. Parça ofset modelinde takım yolları, ada dış hatlarından birbirini izleyen dış ofsetlerle oluşturulurlar. Malzeme dış hatlarının ötesindeki fazladan takım yolları, hızlı bir ilerleme hareketi veya üzerinden hızlı geçme ile değiştirilirler.

4. Dış hatlar-ada ofset modelinde dış hat ofsetleri ada ofsetine ulaşana kadar birbirini izleyerek oluşturulurlar. Daha sonra ada ofseti durma pozisyonu tatmin edici değilken oluşturulur.

5. Orantılı karıştırma ofset modelinde takım yolu kesici takımın dış çevre ve ada arasındaki doğrusal interpolasyon yolların bir serisi boyunca hareket etmesi ile oluşturulur.

6. En büyük - en küçük ofset modelinde takım yolu dış çevre ve adanın en uzun kenarının ve alternatif olarak en kısa kenarının paralel ofsetlerinin bir serisi ile oluşturulur.

Çevre ölçeklendirme de bir cep işleme metodu olarak kullanılabilir. Ama bu yaklaşım sadece dikdörtgen, üçgen ve daire gibi düzenli şekiller için kullanışlıdır. Genellikle iki cep işleme stratejisi yaygın olarak kullanılmaktadır. Bunlar zigzag işleme (Şekil 3.5.b,c) ve çevresel paralel işlemedir (Şekil 3.5.a).

Zigzag cep işleme

Zigzag cep işlemede kesici takım işlenecek alanı zigzag bir yolda verilen bir yöne paralel bir süpürme çizgisini takip ederek çıkarmaktadır. Burada süpürme çizgisi ve yatay eksen arasındaki açı süpürme açısı olarak adlandırılır. Zigzag strateji için takım yolu hesaplama aşağıdaki gibi özetlenebilir.

1. Cep çevrelerinin ofsetlenmesi.
2. Ofset çevreler arasında üst üste binmeler varsa olası bağları ayırma.
3. Referans çizgi yani açıyı ve süpürme yönünü seçme.
4. Referans çizgiyi cebe doğru sabit aralıkta süpürme ve kesişimlerini ofset çevreleri ile hesaplama.
5. Süpürülen çizgileri kesişim noktalarında ayırma ve parçaları sürekli olarak işlenebilecek takım yolu gruplarına ayırma.
6. Takım yolu grubu parçalarını uygun bir şekilde bağlama.
7. Son çevre pasosunu dış hat ve ada etrafında başarma.

Zigzag hareketler takımın kesme için değişimli olarak devir yönünde ve zıt yönde işleme hareketlerine sebep olur. Kesme tipinin bu değişimi kesme hızında bir farklılık ve sonuç olarak tek biçimli olmayan bir yüzey kalitesi oluşturur. Bundan dolayı bitirme işleme gibi aynı yüzey kalitesine sahip olması tavsiye edilen aşamalar için kesme hızı tüm yörünge boyunca sadece bir kesme yönü kullanılarak sabit tutulur. Zıt yön hareketleri bir takım hızlı geri çekme hareketleri ile değiştirilirler. Böyle bir değişken zigzag veya tek yön işlemeyi kapsamaktadır.

Çevresel paralel cep işleme

Bu strateji ile oluşturulan kesme yolları spiral gibidir, genelde iç çevre sınırı ve ada dış çevresinin ofsetlenmesiyle elde edilirler. Bir çevrenin iç ofsetlemesi dışa doğru uzaklaşmaya başladığında veya önceden tanımlanan miktardan daha küçük olmaya

başladığında durdurulur. Genellikle etkili işleme içeriden dışarıya doğru ofsetlemenin ters yönünde oluşur.

Zigzag ve çevresel paralel işlemenin karşılaştırılması

İki ana cep işleme stratejisi olan zigzag ve çevresel paralel işlemenin karşılaştırılması uygulanabilirliğinin kolaylığını, takım yolu uzunluğunun, işlenmemiş alan ve yüzey kalitesi sonucunu göz önünde bulundurarak yapılabilir. Zigzag strateji ile eğrisel bir elemanın bir çizgi veya daha ziyade başka bir eğri ile kesişim noktalarını hesaplamak çevresel paralel stratejiye göre çok daha kolaydır. Bundan dolayı takım yolu doğrulama işlemleri zigzag yaklaşımda çok daha fazla hızlı ve güvenlidir. Bundan dolayı takım yollarının grafiksel doğrulaması bile gereksiz olabilir. Pratik bir rol olarak çevresel paralel strateji hata eğilimli, hesaplaması zor ve pahalı olarak düşünülebilir. Bu noktadan zigzag yaklaşım daha sık tercih edilir ve zigzag yaklaşım genelde son çevresel paso hesaba katılmadığında daha kısa takım yolları verir [39]. Bununla birlikte karmaşık şekiller için çevresel işleme yaklaşımı çok daha etkilidir. Diğer taraftan karmaşık şekillerde ofset cep işleme zigzag işleme metodu ile karşılaştırıldığında daha iyi bir yüzey kalitesi, daha az işleme zamanı vermekte ve daha az oranda boşluk operasyonu gerektirmektedir. Örnek olarak zigzag işleme metodunda eğer cep içbükeyse takım bir ada veya son bir kenarla karşılaştığında etrafında hareket etmeli ve üzerinden geçmelidir. Her iki durumda da daha fazla işleme zamanına veya yüzey kalitesi problemlerinin önemli bir kaynağı olan takımın daha çok çıkış ve girişlerine sebep olacaktır. Bunun yanında çevresel paralel işleme yaklaşımında çevresel kesme daha uygun görülmektedir. Bundan dolayı daha kısa işleme zamanı beklenebilir. Genel bir gözlem olarak deliklerin hazırlanması, teğet hareketler ve kesicilerin yeterliliğine bağlı olarak çok fazla geri çıkmalar ve kesme dışı hareketler zigzag yaklaşımla ilişkilendirilirler. Buna ek olarak işlenmiş yüzeyde çizgiler bırakan zigzag işlemenin aksine çevresel paralel işleme metodu ile oluşturulan takım yolları işlenmiş parçada yönlendirilmiş takım izleri bırakmazlar. Bu da tüm yönlerde aynı yüzey kalitesine sebep olur.

İki ana cep işleme stratejisi olan zigzag ve çevresel paralel işlemenin karşılaştırılması uygulanabilirliğinin kolaylığını, takım yolu uzunluğunun, işlenmemiş alan ve yüzey kalitesi sonucunu göz önünde bulundurarak yapılabilir. Zigzag strateji ile eğrisel bir elemanın bir çizgi veya daha ziyade başka bir eğri ile kesişim noktalarını hesaplamak çevresel paralel stratejiye göre çok daha kolaydır. Bundan dolayı takım yolu doğrulama işlemleri zigzag yaklaşımda çok daha fazla hızlı ve güvenlidir. Bundan dolayı takım yollarının grafiksel doğrulaması bile gereksiz olabilir. Pratik bir rol olarak çevresel paralel strateji hata eğilimli, hesaplaması zor ve pahalı olarak düşünülebilir. Bu noktadan zigzag yaklaşım daha sık tercih edilir ve zigzag yaklaşım genelde son çevresel paso hesaba katılmadığında daha kısa takım yolları verir [36]. Bununla birlikte karmaşık şekiller için çevresel işleme yaklaşımı çok daha etkilidir. Diğer taraftan karmaşık şekillerde ofset cep işleme zigzag işleme metodu ile karşılaştırıldığında daha iyi bir yüzey kalitesi, daha az işleme zamanı vermekte ve daha az oranda boşluk operasyonu gerektirmektedir. Örnek olarak zigzag işleme metodunda eğer cep içbükeyse takım bir ada veya son bir kenarla karşılaştığında etrafında hareket etmeli ve üzerinden geçmelidir. Her iki durumda da daha fazla işleme zamanına veya yüzey kalitesi problemlerinin önemli bir kaynağı olan takımın daha çok çıkış ve girişlerine sebep olacaktır. Bunun yanında çevresel paralel işleme yaklaşımında çevresel kesme daha uygun görülmektedir. Bundan dolayı daha kısa işleme zamanı beklenebilir. Genel bir gözlem olarak deliklerin hazırlanması, teğet hareketler ve kesicilerin yeterliliğine bağlı olarak çok fazla geri çıkmalar ve kesme dışı hareketler zigzag yaklaşımla ilişkilendirilirler. Buna ek olarak işlenmiş yüzeyde çizgiler bırakan zigzag işlemenin aksine çevresel paralel işleme metodu ile oluşturulan takım yolları işlenmiş parçada yönlendirilmiş takım izleri bırakmazlar. Bu da tüm yönlerde aynı yüzey kalitesine sebep olur.

4. OFSETLEME ALGORİTMASI

Bu çalışmada sunulan yöntem iki ana aşamadan oluşmaktadır.

- Hazırlık aşaması: Bu aşamada ofsetleme aşaması için gerekli olan, profil verilerinin elde edilmesi, ada ve cep profillerinin algılanması, referans noktalarının bulunması gibi hazırlık adımları bulunmaktadır.

- Ofsetleme aşaması: Ofsetleme aşamasında profile ait verilerin işlenerek kaba ofsetin elde edilmesi, geçerli ve geçersiz ofset unsurlarını ayıracak özelliklerin belirlenmesi ve hataların giderilerek doğru bir ofsetlemenin oluşturulmasına ilişkin adımlar bulunmaktadır.

4.1. Hazırlık Aşaması

Hazırlık aşamasında sırasıyla aşağıdaki işlemler yapılmaktadır.

- DXF dosyasından unsurların okunması

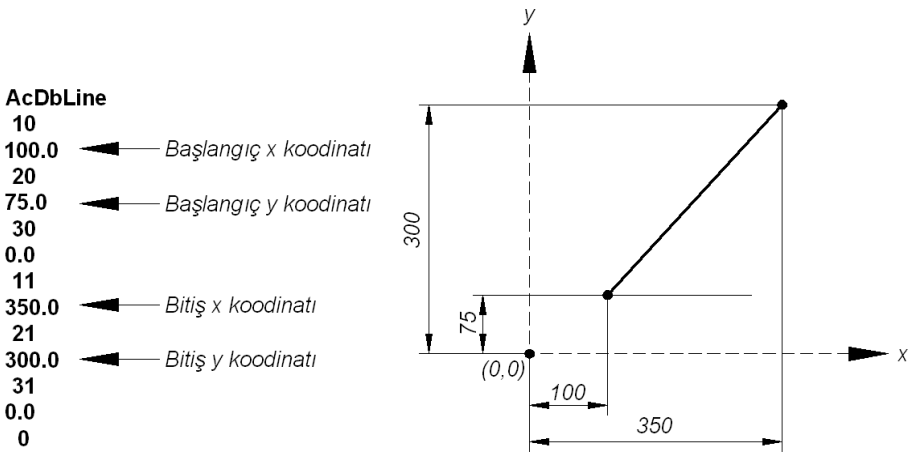
- Doğru ve yay profillerinin belirlenmesi ve unsurların sıralanması

- Profili oluşturan unsurların sıralama yönlerinin bulunması

4.1.1. DXF dosyasından unsurların okunması

Bu çalışmada, günümüzde birçok BDT/BDİ yazılımı tarafından okunup yazabilen, iki boyutlu veriler için ortak bir veri aktarım dili haline gelmiş olan DXF dosya yapısı, cep ve ada profillerine ait verilerin alınması için kullanılmıştır. Öncelikle kullanıcı tarafından, DXF dosyasını okuyup yazabilen herhangi bir BDT/BDİ programı ortamında cep ve ada profilleri tasarlanmıştır. Daha sonra oluşturulmuş cep ve ada profillerinin çizildiği doküman DXF dosya uzantısı ile kaydedilmiştir. Delphi

ortamında oluşturulan programın DXF dosyasının okunması, bu dosya içinde bulunan cep ve ada profillerine ait verilerin ilgili satır ve başlıklar altında yer alan koordinat bilgilerinin programın kendi ortamına alması şeklinde gerçekleştirilmiştir. Dosya içinde bulunan unsurlardan doğruya ait başlangıç ve bitiş noktalarının x ve y koordinatları'AcdbLine' başlığı altında yer alırken, yaylara ait merkez noktasının x ve y koordinatları, yarıçap ve yayın başlangıç ve bitiş açıları 'AcdbCircle' başlığı altında yer almaktadır. Çember de 360 derecelik bir yay olarak kabul edilmekte ve çembere ilişkin veriler de AcdbCircle' başlığı altında yer almaktadır. İlgili başlıklar altında bulunan veriler, Delphi ortamında oluşturulan program tarafından okunarak program içerisindeki bir dizi değişkene aktarılmış ve DXF dosyasını okuma işlemi tamamlanmıştır. Cep ve ada profiline ait verilerin bulunduğu bir DXF dosyası herhangi bir yazı editörü ile açıldığında ilgili başlıklar ve koordinat bilgileri görülebilmektedir. Şekil 4.1. 'de bir doğruya ait verilerin DXF dosyasında yer alış biçimi görülmektedir.

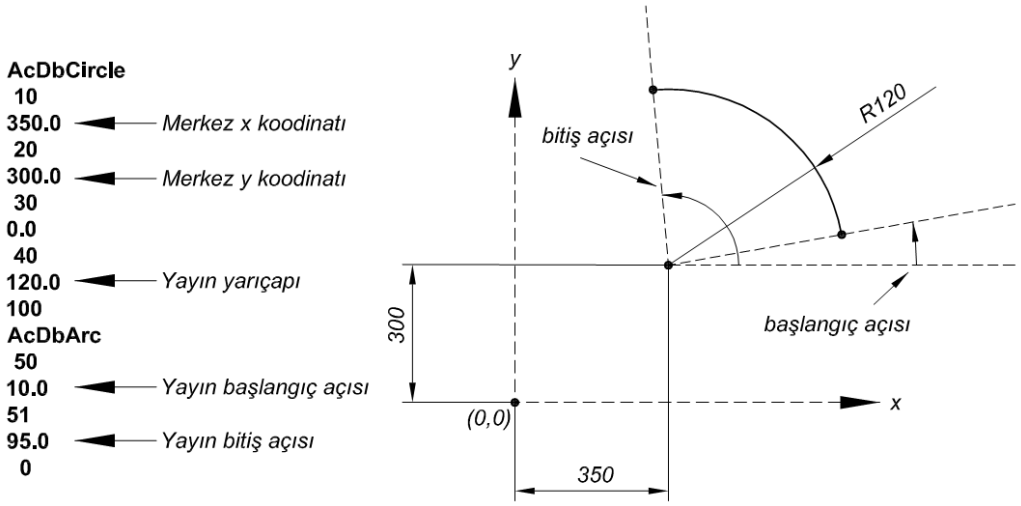


Şekil 4.1. DXF dosyasında doğruya ait verilerin gösterimi

Aşağıda verilen program kodları ile DXF dosya yapısından doğru tipindeki unsurlara ait veriler okumuştur. Bu veriler Delphi ortamında kullanılan 'hamunsurlar' isimli dizi değişkene atanmıştır.

```
…
reset(dxf);
 while not eof (dxf) do
   begin
     readln(dxf,satir);
     if (satir='AcDbLine') then
      begin
        hamunsurlar[i,0]:=inttostr(i+1);
        hamunsurlar[i,1]:='Çizgi';
        readln(dxf,satir);readln(dxf,satir);
        hamunsurlar[i,2]:=floattostr(roundto(strtofloat(satir),-tol));
        readln(dxf,satir);readln(dxf,satir);
        hamunsurlar[i,3]:=floattostr(roundto(strtofloat(satir),-tol));
        readln(dxf,satir);readln(dxf,satir);
        readln(dxf,satir);readln(dxf,satir);
        hamunsurlar[i,4]:=floattostr(roundto(strtofloat(satir),-tol));
        readln(dxf,satir);readln(dxf,satir);
        hamunsurlar[i,5]:=floattostr(roundto(strtofloat(satir),-tol));
        hamunsurlar[i,6]:='';
        hamunsurlar[i,10]:='1';
        hamunsurlar[i,11]:='0';
      end;
   end;
 closefile(dxf);
…
```

Şekil 4.2. ‘de bir doğruya ait verilerin DXF dosyasında yer alış biçimi görülmektedir.



Şekil 4.2. DXF dosyasında yaya ait verilerin gösterimi

Aşağıda verilen program kodları ile DXF dosya yapısından yay tipindeki unsurlara ait veriler okunmaktadır. Bu veriler Delphi ortamında kullanılan ‘hamunsurlar’ isimli dizi değişkene atanmıştır.

```
…
 reset(dxf);
 while not eof (dxf) do
  begin
    readln(dxf,satir);
    if (satir='AcDbCircle') then
     begin
       readln(dxf,satir);readln(dxf,satir);
       mx:=strtofloat(satir);
       readln(dxf,satir);readln(dxf,satir);
       my:=strtofloat(satir);
       readln(dxf,satir);readln(dxf,satir);
       readln(dxf,satir);readln(dxf,satir);
       r:=strtofloat(satir);;
       readln(dxf,satir);
```

```
          if satir='100' then
           begin
             readln(dxf,satir);
             readln(dxf,satir);readln(dxf,satir);
             basaci:=strtofloat(satir);
             readln(dxf,satir);readln(dxf,satir);
             bitaci:=strtofloat(satir);
           end;
         if satir='  0' then
           begin
             basaci:=0;
             bitaci:=360;
           end;
          hamunsurlar[i,0]:=inttostr(i+1);
          hamunsurlar[i,1]:='Yay';
          hamunsurlar[i,2]:=floattostr(roundto(r*cos(basaci*pi/180)+mx,-tol));
          hamunsurlar[i,3]:=floattostr(roundto(r*sin(basaci*pi/180)+my,-tol));
          hamunsurlar[i,4]:=floattostr(roundto(r*cos(bitaci*pi/180)+mx,-tol));
          hamunsurlar[i,5]:=floattostr(roundto(r*sin(bitaci*pi/180)+my,-tol));
          hamunsurlar[i,6]:=floattostr(roundto(mx,-tol));
          hamunsurlar[i,7]:=floattostr(roundto(my,-tol));
          hamunsurlar[i,8]:=floattostr(r);
          hamunsurlar[i,9]:='sag';
          hamunsurlar[i,10]:='1';
          hamunsurlar[i,11]:='0';
        end;
    end;
  closefile(dxf);
…
```

Delphi ortamında hazırlanan program ile DXF dosyasından okunarak elde edilen veriler Çizelge 4.1 ve Çizelge 4.2'te görülen yapıya dönüştürülmüştür.

Çizelge 4.1. Program ortamında doğru unsurunu tanımlayan yapı

1	Unsur numarası
Doğru	Unsur tipi
100	Doğrunun başlangıç noktasının x koordinatı
75	Doğrunun başlangıç noktasının y koordinatı
350	Doğrunun bitiş noktasının x koordinatı
300	Doğrunun bitiş noktasının y koordinatı
47	Doğrunun açısı

Çizelge 4.2. Program ortamında yay unsurunu tanımlayan yapı

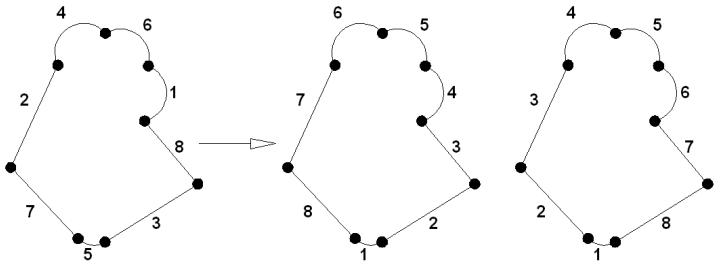
2	Unsur numarası
Yay	Unsur tipi
371.85	Yayın başlangıç noktasının x koordinatı
303.85	Yayın başlangıç noktasının y koordinatı
298.07	Yayın bitiş noktasının x koordinatı
322.10	Yayın bitiş noktasının y koordinatı
350	Yayın merkez noktasının x koordinatı
300	Yayın merkez noktasının y koordinatı
120	Yayın yarıçapı

Doğru unsurunun tanılanabilmesi için yeterli olan veriler DXF dosyasından alınmış ve program ortamında kullanılan daha sade bir yapıya dönüştürülmüştür (Çizelge 4.1). Yay unsuru için ise yayı tanımlayan veriler DXF dosyasından okunarak alınmış ve program ortamında kullanılan özel yapıya dönüştürülmüştür. Ancak DXF dosyasından farklı olarak yayın başlangıç ve bitiş açıları yerine trigonometrik oranlar kullanılarak yayın başlangıç ve bitiş koordinatları hesaplanmış ve yapıda bu değerler kullanılmıştır (Çizelge 4.2).

DXF dosyasından verilerin okunmasından sonra oluşturulan yazılım ile okunan verilerin yorumlanması işlemine geçilmiştir.

4.1.2. Doğru ve yay profillerinin belirlenmesi ve unsurların sıralanması

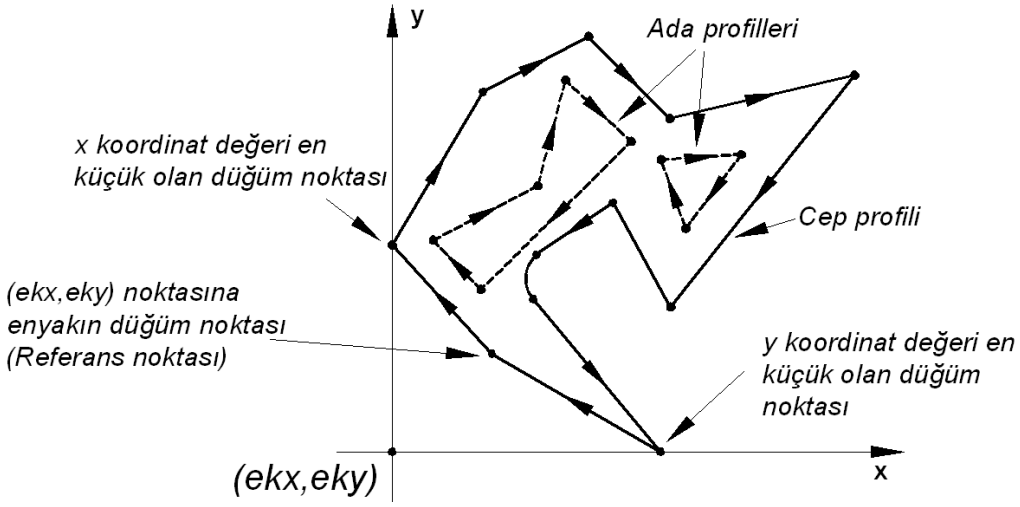
DXF dosyasından profillere ait verilerin okunup Çizelge 4.1 ve Çizelge 4.2'teki formata dönüştürüldükten sonra verilerin yorumlanarak profil haline getirilmesi işlemine geçilmiştir. DXF dosyasında adasız, bir adalı veya birden fazla adaya sahip cebe ait bilgiler bulunabileceğinden, hangi unsurun (doğru ya da yay) hangi profile (cep profili ya da ada profili) ait olduğunun belirlenmesi gerekmektedir. Ayrıca DXF dosyasını oluşturan kullanıcının profili tasarlarken doğru ve yayları bir zincir oluşturacak şekilde belli bir sıra gözetmeksizin çizdiği varsayılmıştır (Şekil 4.3. a). Bu nedenle Delphi ortamında uygulanan algoritmanın DXF dosyasındaki verileri yorumlayabilmesi için unsurların düzgün bir sıralamada olması gerekmekte ve bu unsurları sıralayacak bir işleme ihtiyaç duyulmaktadır (Şekil 4.3. b).



Şekil 4.3. Rastgele oluşturulmuş ve sıralanmış profil unsurları

Sıralama işlemini gerçekleştirebilmek için ilk önce referans noktası diye anılan bir başlama noktası belirlenmiştir. Referans noktası, profilleri oluşturan unsurların en küçük x (ekx) ve en küçük y (eky) koordinat değerlerinin bulunmasıyla elde edilen (ekx,eky) noktasına en yakın düğüm noktasıdır (Şekil 4.4.). Referans noktası en dışta bulunan profil üzerinde yer alacağı için referans noktasını üzerinde bulunduran kapalı profil cep profili ve bundan sonra belirlenecek bütün profiller de ada profili

olacaktır. Referans noktasının bulunmasından sonra bu noktadan başlayıp yine bu noktada biten kapalı profili oluşturan unsurlar aranacaktır.



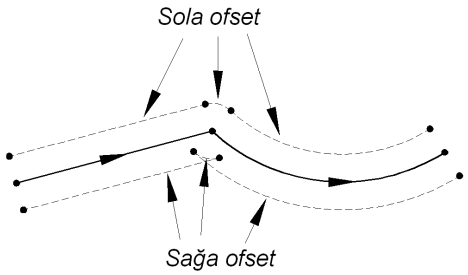
Şekil 4.4. Referans noktasının bulunması ve unsurların sıralanması

Başlangıç noktasının koordinatları, referans noktasının koordinatlarına eşit olan ilk unsurun bulunmasından sonra başlangıç noktası ilk unsurun bitiş koordinatına eşit olan bir sonraki unsur bulunur. Cep profilini oluşturan unsurların aranması işlemi bulunan son unsurun bitiş noktasının koordinatları yine referans noktasına eşit oluncaya kadar devam eder. Bulunan her unsurun verileri, Delphi ortamında geliştirilen yazılım tarafından bir dizi değişkene aktarılır (EK-1).

İlk profilin bulunmasından sonra ilk profil üzerinde yer almayan ve referans noktasına en yakın düğüm noktası bulunur ve bu nokta yeni referans noktası kabul edilerek sonraki profilin unsurlarının aranması işlemine devam edilir. Unsur arama işleminin bitmesiyle cep ve ada profilleri belirlenmiş, profilleri oluşturan unsurlar sıralanmış ve profillere ait veriler geliştirilen yazılımın kullanabileceği bir şekilde dizi değişkene aktarılmıştır.

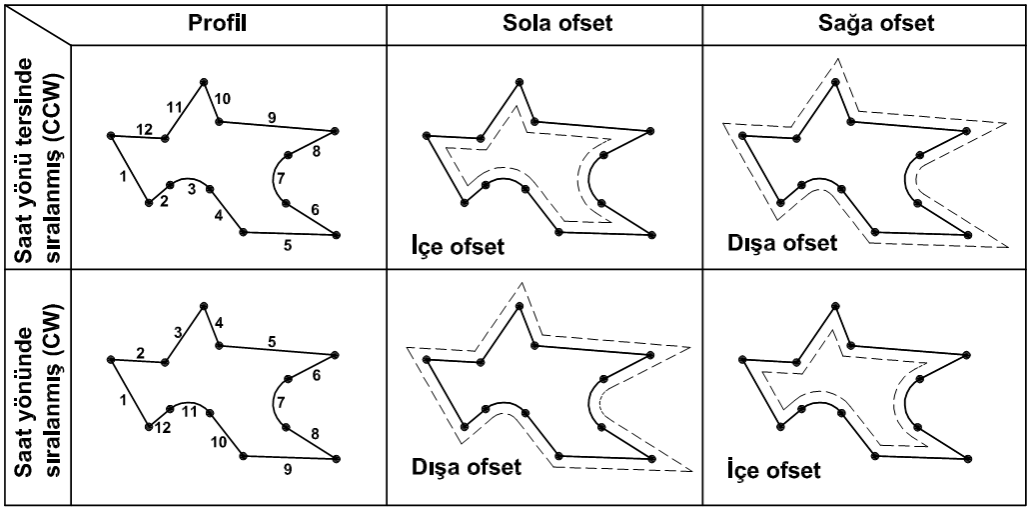
4.1.3. Profilleri oluşturan unsurların sıralama yönlerinin bulunması

Cep profilini işleyecek takım yollarını oluşturabilmek için cep profilinin içe ada profilinin de dışa doğru ofsetlenmesi gerekmektedir. Bir profili içe ya da dışa ofsetleyebilmek için profili oluşturan unsurların her birinin ya sağa ya da sola doğru ofsetlenmelidir (Şekil 4.5).



Şekil 4.5. Unsurların sağa ve sola ofseti

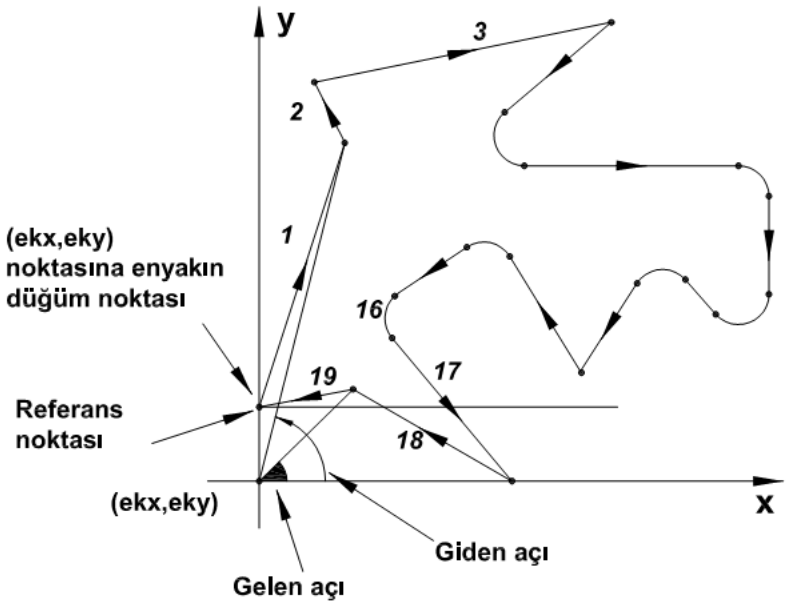
Bir unsurun ofsetleme yönünün sağa doğru mu yoksa sola doğru mu olacağının tayini için unsurların oluşturduğu profilin sıralama yönünün bilinmesi gerekmektedir. Çünkü, saat yönünde sıralanmış bir profilin unsurlarını sağa doğru ofsetlediğimizde profil içe ofsetlenmiş olurken saat yönü tersinde sıralanmış bir profilin unsurlarını yine sağa doğru ofsetlediğimizde profil dışa ofsetlenmiş olmaktadır (Şekil 4.6).



Şekil 4.6. Profilin içe ve dışa ofsetlenmesi

Profili oluşturan unsurların sıralama yönünü bulmak için öncelikle profilin en küçük x (ekx) ve en küçük y (eky) koordinat değerleri bulunur. Daha sonra (ekx,eky) noktasına, iki unsurun bağlandığı en yakın düğüm noktası yani referans noktası bulunur. Referans noktasından başlayan unsurun bitiş noktası ile (ekx,eky) noktasını birleştiren doğru parçasının yatay ile yaptığı açı 'giden açı' olarak adlandırılmıştır. Referans noktasında biten unsurun başlangıç noktası ile (ekx,eky) noktasını birleştiren doğru parçasının yatay ile yaptığı açı 'gelen açı' diye adlandırılmıştır. Gelen açı giden açıdan büyük ise sıralama yönü saat yönü tersindedir. Gelen açı giden açıdan küçük ise sıralama yönü saat yönündedir (Şekil 4.7). Referans noktasında bağlanan unsurlardan herhangi birinin veya ikisinin de yay olması durumunda unsurların bitiş/başlangıç noktası yerine yayın tepe noktası ile (ekx,eky)'yi birleştiren doğru parçasının yatay ile yaptığı açı kullanılır. Sıralama işlemini gerçekleştiren program kodları EK-2'de verilmiştir.

Aynı metot ada profillerinin sıralama yönlerinin bulunması için de kullanılır.



Şekil 4.7. Profilin sıralama yönünün bulunması

Çizelge 4.3'te Şekil 4.7'de yapılan sıralama işlemi sonucunda elde edilen veriler görülmektedir. 1. unsur referans noktasından başlamaktadır. En son unsur olan 19. doğru de yine referans noktasında bitmektedir. Ardı ardına gelen unsurlar incelendiğinde bir sonraki unsur bir önceki unsurun bitiş noktası ile başlamaktadır. Birinci unsurun başlangıç noktası, aynı zamanda sonuncu unsurun bitiş noktasıdır.

Çizelge 4.3. Delphi ortamında sıralama işlemi sonucu elde edilen veriler

1	2	3		18	19
Doğru	Doğru	Doğru		Doğru	Doğru
40.49	54.59	49.57		82.75	55.96
48.48	92.00	101.98		36.27	51.39
54.59	49.57	98.49	...	55.96	**40.49**
92.00	101.98	111.85		51.39	**48.48**
72.04	116.69	11.40		129.95	190.65
Referans noktası (40.49 , 48.48) ekx: 36.27 eky: 40,49					

Cep profili ve ada profillerinin sıralama yönlerinin de bulunması ile hazırlık aşaması tamamlanmıştır.

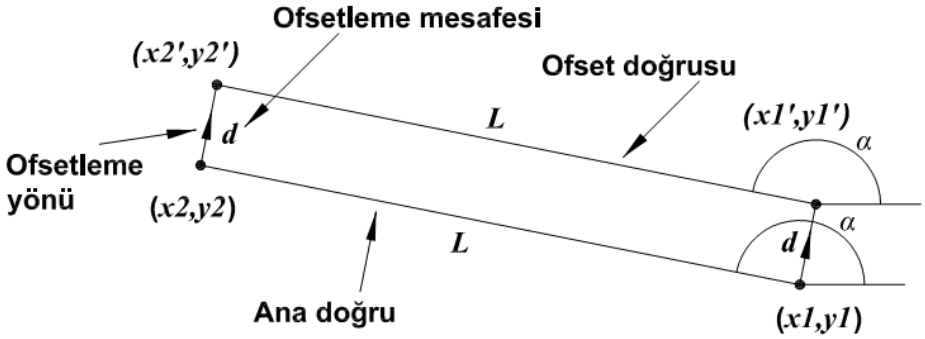
4.2. Ofsetleme Aşaması

Ofsetleme aşamasında öncelikle kaba ofsetleme işlemi gerçekleştirilmektedir. Kaba ofsetleme işleminde cep ve ada profillerini oluşturan unsurlar bire bir ölçüsünde sağ tarafa ve ya sol tarafa kopyalanır (Şekil 4.5). Bu işlem gerçekleştirilirken cep ve ada profilini oluşturan doğrular, yaylar ve buları bir birine bağlandığı düğüm noktaları ayrı ayrı ofsetlenir. Kaba ofsetleme işleminde ofsetleme ile birlikte gelen, dalma ve kalıntılara neden olabilecek hatalar, göz ardı edilmiştir. Cep profilleri içe, ada

profilleri de dışa ofsetlenmiş ve hataların giderilmesi ileriki adımlarda uygulanan hata ayıklama işleminde gerçekleştirilmiştir.

4.2.1. Doğruların ofsetlenmesi

Bir doğrunun ofseti bulunurken doğrunun başlangıç ve bitiş noktalarının, ofsetleme yönünde ve ofsetleme mesafesi kadar uzağında bulunan noktaların yani ofset doğrusunun uç noktalarının koordinatları hesaplanır.



Şekil 4.8. Doğrunun ofsetlenmesi

Şekil 4.8'de görüldüğü gibi doğrunun ofsetlenmesi sonucu ofsetlenen doğru ile aynı boyda, aynı açıda ve ofset mesafesi kadar ötede yeni bir doğru elde edilir.

Ofsetleme sola doğru yapılıyorsa ofset doğrusunun uç noktalarının hesaplanması için kullanılacak denklemler (Eş. 4.1 - 4.4) 'de verilmiştir.

$$x1' = x1 + \cos(90 + a).d \tag{4.1}$$

$$y1' = y1 + \sin(90 + a).d \tag{4.2}$$

$$x2' = x2 + \cos(90 + a).d \tag{4.3}$$

$$y2' = y2 + \sin(90 + a).d \tag{4.4}$$

$x1' = x1 + \cos(270 + a).d$ (4.5)

$y1' = y1 + \sin(270 + a).d$ (4.6)

$x2' = x2 + \cos(270 + a).d$ (4.7)

$y2' = y2 + \sin(270 + a).d$ (4.8)

Ofsetleme sağa doğru yapılıyorsa ofset doğrusunun uç noktalarının hesaplanması için (Eş. 4.5 - 4.8) 'de verilen denklemler kullanılmıştır.

Denklemlerde kullanılan parametreler aşağıda açıklanmıştır.

x1 : Ofsetlenen doğrunun başlangıç noktasının x eksenindeki koordinatı
x1' : Ofset doğrusunun başlangıç noktasının x eksenindeki koordinatı
y1 : Ofsetlenen doğrunun başlangıç noktasının y eksenindeki koordinatı
y1' : Ofset doğrusunun başlangıç noktasının y eksenindeki koordinatı
x2 : Ofsetlenen doğrunun bitiş noktasının x eksenindeki koordinatı
x2' : Ofset doğrusunun bitiş noktasının x eksenindeki koordinatı
y2 : Ofsetlenen doğrunun bitiş noktasının y eksenindeki koordinatı
y2' : Ofset doğrusunun bitiş noktasının y eksenindeki koordinatı
a : Ofsetlenen doğrunun x ekseni ile yaptığı açı
d : Ofsetleme mesafesi

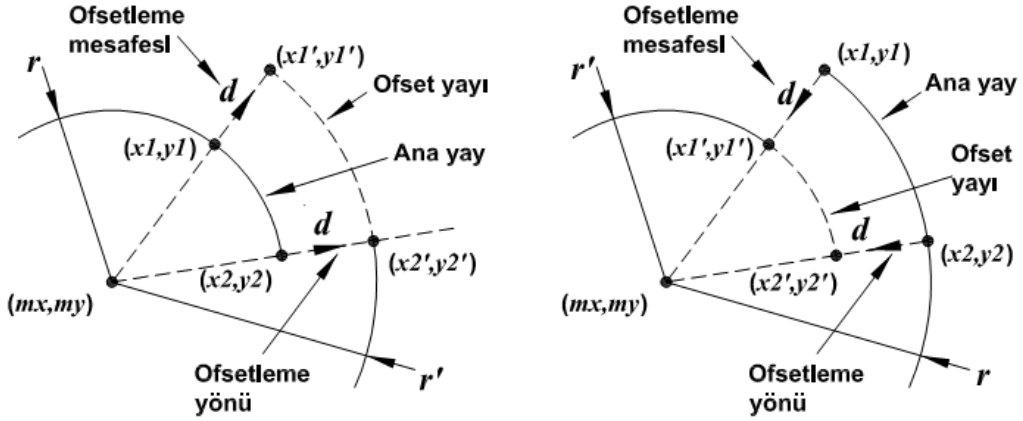
Aşağıda verilen program kodları ile doğrunun ofseti hesaplanmış ve program ortamındaki 'kabaofsetler' isimli dizi değişkene atanmıştır.

```
…
        if sirunsurlar[n+k-2,1]='Çizgi' then
          begin
            x1:=strtofloat(sirunsurlar[n+k-2,2]);
            y1:=strtofloat(sirunsurlar[n+k-2,3]);
```

```
            x2:=strtofloat(sirunsurlar[n+k-2,4]);
            y2:=strtofloat(sirunsurlar[n+k-2,5]);
            aci:=acibul(x1,y1,x2,y2);
            ox1:=x1+(d*cos((aci+ofaci)*pi/180));
            oy1:=y1+(d*sin((aci+ofaci)*pi/180));
            ox2:=x2+(d*cos((aci+ofaci)*pi/180));
            oy2:=y2+(d*sin((aci+ofaci)*pi/180));
            kabaofsetler[(n+k-2)*2,0]:=inttostr((n+k-2)*2);
            kabaofsetler[(n+k-2)*2,1]:='Çizgi';
            kabaofsetler[(n+k-2)*2,2]:=floattostr(ox1);
            kabaofsetler[(n+k-2)*2,3]:=floattostr(oy1);
            kabaofsetler[(n+k-2)*2,4]:=floattostr(ox2);
            kabaofsetler[(n+k-2)*2,5]:=floattostr(oy2);
            kabaofsetler[(n+k-2)*2,6]:=sirunsurlar[n+k-2,6];
            kabaofsetler[(n+k-2)*2,10]:=inttostr(i);
            kabaofsetler[(n+k-2)*2,11]:='0';
          end;
…
```

4.2.2. Yayların ofsetlenmesi

Bir yayın ofseti bulunurken değişen tek parametresi yayın yarıçapıdır. Yayın diğer parametreleri bulunan ofset yayındaki parametreler ile eşittir. Ofsetleme yayın merkezinden dışına doğru yapılıyorsa yeni ofset yayının yarıçapı ofsetleme mesafesi kadar artmakta, ofsetleme yayın merkezine doğru yapılıyorsa yeni ofset yayının yarıçapı ofsetleme mesafesi kadar azalmaktadır (Şekil 4.9).



Şekil 4.9. Yayın içe ve dışa ofsetlenmesi

Ofset yayının yarıçapının ofsetlenmesinde kullanılan denklemler (Eş. 4.9), (Eş. 4.10) numaralı eşitliklerde verilmiştir.

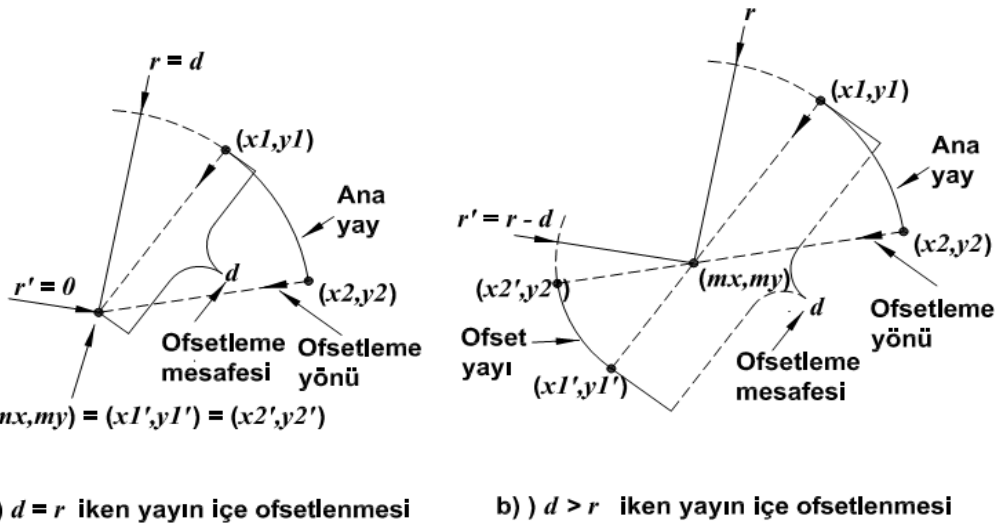
Ofsetleme merkeze doğru yapılıyorsa;

$$r' = r - d \tag{4.9}$$

Ofsetleme merkezden dışa doğru yapılıyorsa;

$$r' = r + d \tag{4.10}$$

olur.



a) $d = r$ iken yayın içe ofsetlenmesi b)) $d > r$ iken yayın içe ofsetlenmesi

Şekil 4.10. d = r ve d > r olması durumunda yayın içe ofsetlenmesi

Ofsetleme mesafesi ofsetlenen yayın yarıçapına eşit olması durumunda, merkeze doğru yapılan bir ofsetleme işleminde yayın ofseti merkez koordinatında bulunan bir nokta şeklinde oluşacaktır (Şekil 4.10.a). Ofsetleme mesafesi ofsetlenen yayın yarıçapından büyük olması durumunda, merkeze doğru yapılan bir ofsetleme işleminde ise yayın ofseti, ofsetleme mesafesi ile yayın yarıçapının farkı kadar yarıçaplı ve başlangıç ve bitiş açıları ofsetleme yönüne göre ters istikamette olan bir yay şeklinde oluşacaktır (Şekil 4.10. b).

Ofsetleme mesafesinin yayın yarıçapından büyük olması durumunda, ofsetleme işlemi merkeze doğru yapılıyorsa, ofset yayının yarıçapının hesaplanması için kullanılacak denklem;

$$r' = d - r \tag{4.11}$$

olur.

Ofset yaylarının yarıçaplarının hesaplanması için kullanılan denklemlerdeki parametreler aşağıda açıklanmıştır.

r : Ofsetlenen yayın yarıçapı

r' : Oluşacak ofset yayının yarıçapı

d : Ofsetleme mesafesi

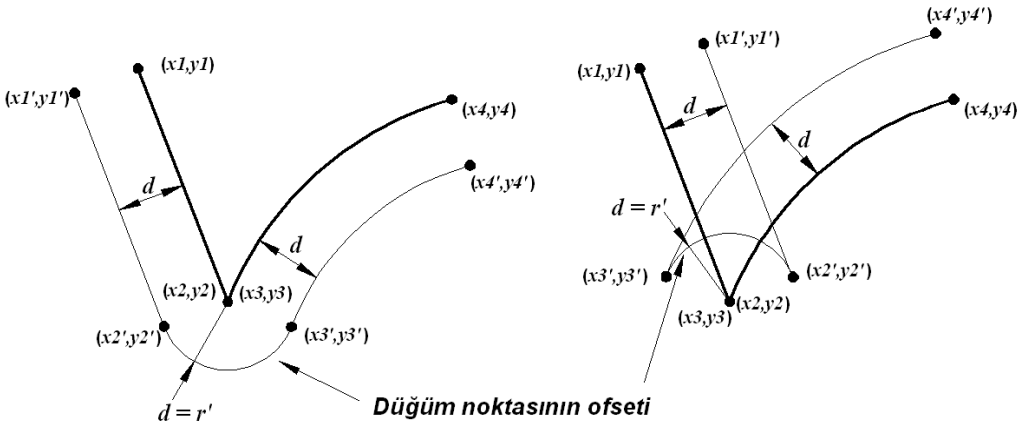
Yayın ofsetini bulduran program kodları aşağıda verilmiştir.

```
…
        if sirunsurlar[n+k-2,1]='Yay' then
         begin
           x1:=strtofloat(sirunsurlar[n+k-2,2]);
           y1:=strtofloat(sirunsurlar[n+k-2,3]);
           x2:=strtofloat(sirunsurlar[n+k-2,4]);
           y2:=strtofloat(sirunsurlar[n+k-2,5]);
           mx:=strtofloat(sirunsurlar[n+k-2,6]);
           my:=strtofloat(sirunsurlar[n+k-2,7]);
           r:=strtofloat(sirunsurlar[n+k-2,8]);
           if r>d then
            begin
              basaci:=acibul(mx,my,x1,y1);
              bitaci:=acibul(mx,my,x2,y2);
            end;
           if r<d then
            begin
              basaci:=acibul(mx,my,x2,y2);
              bitaci:=acibul(mx,my,x1,y1);
            end;
           if (siralamayonu='sy') and (sirunsurlar[n+k-2,9]='sag') then ofr:=r+d;
           if (siralamayonu='sy') and (sirunsurlar[n+k-2,9]='sol') then ofr:=r-d;
           if (siralamayonu='sty') and (sirunsurlar[n+k-2,9]='sag') then ofr:=r-d;
           if (siralamayonu='sty') and (sirunsurlar[n+k-2,9]='sol') then ofr:=r+d;
           if r<d then ofr:=abs(r-d);
           kabaofsetler[(n+k-2)*2,0]:=inttostr((n+k-2)*2);
```

```
  kabaofsetler[(n+k-2)*2,1]:='Yay';
  kabaofsetler[(n+k-2)*2,2]:=floattostr(mx+cos(basaci*pi/180)*ofr);
  kabaofsetler[(n+k-2)*2,3]:=floattostr(my+sin(basaci*pi/180)*ofr);
  kabaofsetler[(n+k-2)*2,4]:=floattostr(mx+cos(bitaci*pi/180)*ofr);
  kabaofsetler[(n+k-2)*2,5]:=floattostr(my+sin(bitaci*pi/180)*ofr);
  kabaofsetler[(n+k-2)*2,6]:=sirunsurlar[n+k-2,6];
  kabaofsetler[(n+k-2)*2,7]:=sirunsurlar[n+k-2,7];
  kabaofsetler[(n+k-2)*2,8]:=floattostr(ofr);
  kabaofsetler[(n+k-2)*2,9]:=sirunsurlar[n+k-2,9];
  kabaofsetler[(n+k-2)*2,10]:=inttostr(i);
  kabaofsetler[(n+k-2)*2,11]:='0';
end;
```

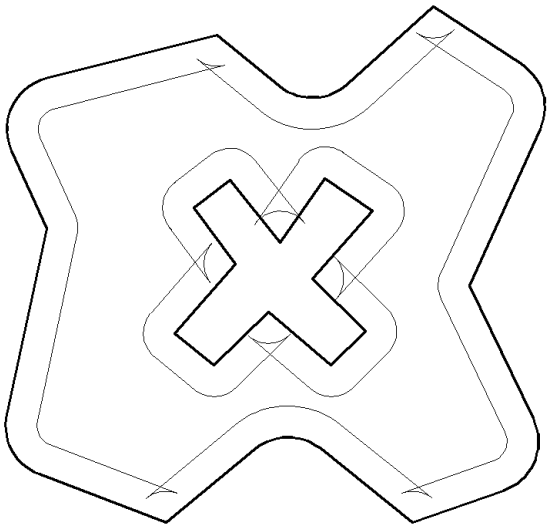
4.2.3. Noktaların ofsetlenmesi

İki unsurun birleştiği düğüm noktasının ofseti, ofsetleme yönünde, ofsetlenen düğüm noktası ile aynı koordinatlı merkezindeki yeni oluşan bir yaydır. Oluşan bu yayın yarıçapı ofsetleme mesafesi kadardır. Başlangıç ve bitiş açıları ise ofsetlenen düğüm noktasında birleşen unsurların sırlandırdığı açı aralığındadır (Şekil 4.11).



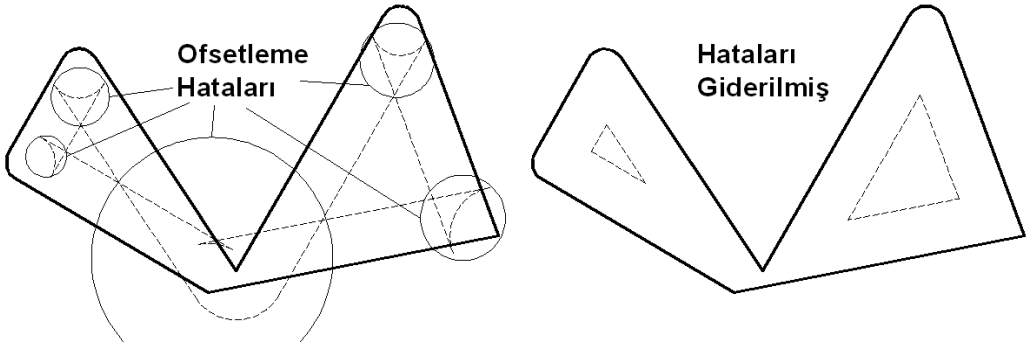
Şekil 4.11. Düğüm noktasının ofsetlenmesi

Profili oluşturan doğru, yay ve düğüm noktalarının ofsetlenmesi ile profil kabaca ofsetlenmiş olur. Cep profili içe doğru, ada profili ise dışa doğru ofsetlenmiştir (Şekil 4.12.). Kaba ofsetleme ile oluşan hatalı fazlalıklar bir hata ayıklama uygulaması ile atılacaktır.



Şekil 4.12. Kaba ofsetlenmiş cep ve ada profili

Ofset unsurları kullanılarak takım yolları oluşturulacağı için dalma ve kalıntılara neden olacak ofset kısımları hatalı kabul edilmiştir. Genel olarak ofset hataları ofset unsurlarının birbirlerini ya da ofsetlenen ana profili kesmesi durumunda ortaya çıkmaktadır (Şekil 4.13).



Şekil 4.13. Ofsetleme hataları

4.3. Ofset Doğrularının ve Yaylarının Kırılması

Kaba ofsetleme ile oluşan ofsetleme hatalarının giderilmesi için bir hata ayıklama işlemi uygulanmaktadır. Ancak, hatalı ofset kısımları atılırken geçerli ofset kısımlarının da atılmaması için birbirleri ile kesişen tüm ofset doğrularının ve yaylarının kesişim noktalarından kırılması yani parçalara ayrılması gerekmektedir. Bu sayede geçersiz ofset kısımlarını atmak oldukça kolaylaşmaktadır. Kırma işlemi gerçekleştirilirken Delphi ortamında hazırlanan program her bir unsuru ayrı arı ele alarak o unsurun kesişim durumlarını değerlendirmektedir. Kesişim durumunda olan her unsur atılarak yerine şekil olarak aynı unsuru oluşturan birden fazla unsur türetilmektedir.

Ofset doğru ve yaylarının birbirleri ile kesişip kesişmediklerini bulabilmek için,

- İki doğrunun kesişimi,
- Doğru ve yayın kesişimi,
- İki yayın kesişimi

durumları ayrı ayrı incelenmiştir. Kesişip kesişmediği kontrol edilen her iki unsurun (doğru veya yay) kesişim durumu kontrol edilmiş ve kesişiyorlar ise kesişim noktaları hesaplanmıştır. Daha sonra bu kesişim noktaları unsurun başlangıç noktasına olan uzaklıklarına göre sıralanmış ve bir sonraki adımda da yeni unsurlar oluşturulmuştur.

4.3.1. İki doğrunun kesişimi

Doğruları kesişim noktalarından kırabilmek için bir doğruyu yine bir doğrunun kesmesi durumunda bu doğruların kesişim noktalarını bilinmesi gerekmektedir. Kırılıp kırılmaması konusunda irdelenen doğru ile karşılaştırılan diğer doğrunun kesişim noktası hesaplanmıştır. İki doğrunum kesişim noktası bulunurken her iki doğrunun analitik denklemleri (Eş. 4.12, - 4.15) oluşturulmuştur.

$$m1 = (y2-y1)/(x2-x1) \tag{4.12}$$

$$m2 = (y4-y3)/(x4-x3) \tag{4.13}$$

$$y-y1=[(y2-y1)/(x2-x1')](x-x1) \tag{4.14}$$

$$y-y3=[(y4-y3)/(x4-x3)].(x-x3) \tag{4.15}$$

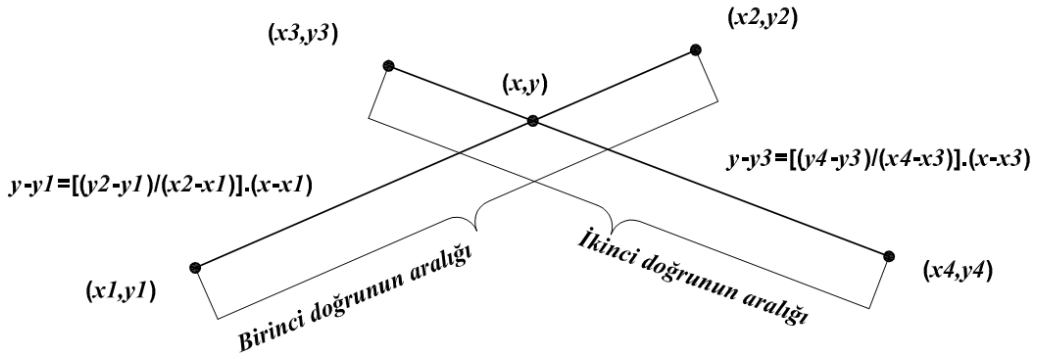
Oluşturulan denklemlerin ortak çözümü ile Eş. 4.16 ve Eş. 4.17 verilen denklemler elde edilmiştir.

$$x=([(y2-y1)/(x2-x1)].x1-[(y4-y3)/(x4-x3)].x3+y3-y1)/([(y2-y1)/(x2-x1)] -[(y4-y3)/(x4-x3)]) \tag{4.16}$$

$$y=[(y2-y1')/(x2-x1)].(x-x1)+y1 \tag{4.17}$$

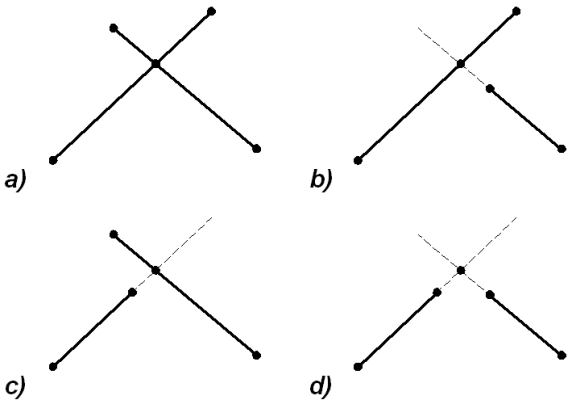
Elde edilen yeni denklemler ile iki doğrunun kesişim noktası hesaplanmıştır. Kesişim noktasını hesaplama işlemine gidilmeden önce doğruların eğimleri (Eş. 4.12 ve Eş. 4.13) hesaplanarak paralel olup olmadıkları kontrol edilmiştir. İki doğrunun eğimi eşit ise bu doğrular paralel demektir. Paralel iki doğrunun kesişmediği kabul edilmiş ve kesişim noktasının hesaplanmasına gerek kalmamıştır. Doğruların eğimleri birbirine eşit değil ise doğruların doğrultuları bir noktada kesişiyor demektir. Bu kontrol işleminden sonra doğruların kesiştikleri nokta hesaplanmıştır.

Şekil 4.14.'de görüldüğü gibi ortak çözüm denkleminin kullanılmasıyla kesişim noktası bulunmakadır.



Şekil 4.14. Doğruların kesişim noktasının bulunması

Doğru denklemlerinin ortak çözüm kümesinin bulunması doğruların kesiştiği anlamına gelmemektedir. Çünkü kesişim noktasının iki doğrunun da başlangıç ve bitiş noktası aralığında olması gerekmektedir. Şekil 4.15'te görüldüğü gibi dört durumda da doğru uzantıları kesişmektir. Sadece Şekil 4.15.a'da ki durumda kesişim noktası her iki doğrunun da aralığında bulunduğu için bu iki doğru kesişmektedir. Şekil 4.15.b, Şekil 4.15. c ve Şekil 4.15. d'de ki durumlarda ise kesişim noktası her iki unsurun da aralığında bulunmadığı için söz konusu unsurlar kesişmemektedir.



Şekil 4.15. İki doğrunun kesişim durumları

İki doğrunun kesişim noktalarını bulduran program kodları EK-3'te verilmiştir.

Ugulanan bu işlemler ile iki doğrunun kesişim noktaları bulunmuştur. Hesaplanan kesişim noktası ve kırılacak doğrunun başlangıç ve bitiş noktaları kullanılarak kırma sonucu elde edilen yeni unsurlar oluşturulacaktır.

4.3.2. Doğru ve yayın kesişimi

Kırılacak unsur doğru iken bu unsuru kesen unsurun yay olması durumunda veya kırılacak unsurun yay ve bu unsuru kesen unsurun doğru olması durumunda bu doğru ve yayın kesişim noktasının bulunması gerekmektedir. Daha sonra kırılacak unsur bulunan kesişim noktasından kırılmakta ve yeni unsurlar oluşturulmaktadır. Bir doğru ve yayın kesişim noktası bulunurken doğru ve yayın ikisinin de ayrı ayrı analitik denklemleri, Eş. 4.14, Eş. 4.18 ve Eş. 4.19 oluşturulmuştur.

$$y-y1=[(y2-y1)/(x2-x1')](x-x1) \tag{4.14}$$

$$m=(y2-y1)/(x2-x1) \tag{4.18}$$

$$r^2=(x-a)^2+(y-b)^2 \tag{4.19}$$

Oluşturulan bu analitik denklemlerinin ortak çözümü ile Eş. 4.20 ve Eş. 4.21 'de verilen denklemler elde edilmiştir.

$$x^2.(1+m^2)+x.(-2a+2m.(-m.x1+y1-b))+(a^2+(-m.x1+y1-b)^2-r^2)=0 \tag{4.20}$$

$$y=[(y2-y1)/(x2-x1')].(x-x1')+y1 \tag{4.21}$$

Elde edilen ortak çözüm denklemlerinin çözüm kümesi doğru ve yayın kesişim noktalarını vermektedir.

Denklemlerde kullanılan parametreler;

m : Doğrunun eğimi

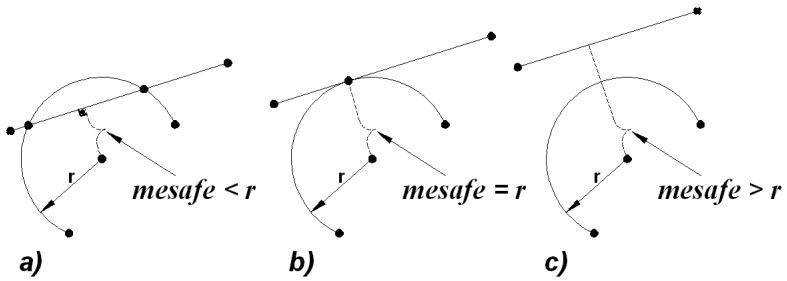
r : Yayın yarıçapı

a : Yayın merkezinin x eksenindeki koordinatı

b : Yayın merkezinin y eksenindeki koordinatı

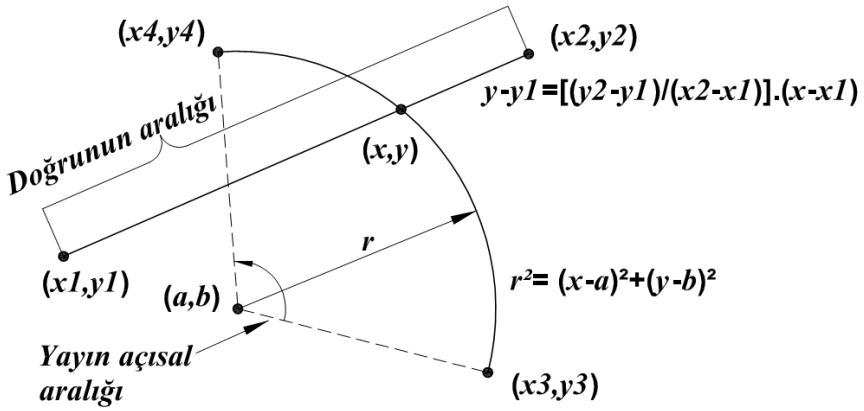
olarak tanımlanmıştır.

Kesişim noktasının hesaplanması işlemine gidilmeden önce doğru ve yayın kesişip kesişmediğinin ön kontrolü için doğrunun yayın merkezine dik uzaklığı hesaplanmıştır (Şekil 4.16). Bulunan uzaklık yayın yarıçapından küçükse doğrunun doğrultusu ve yayın üzerinde bulunduğu çember iki noktada kesişiyor demektir. Bulunan mesafe yayın yarıçapına eşit ise doğrunun doğrultusu yayın üzerinde bulunduğu çemberi tek noktada kesiyor ve çembere teğet demektir. Aynı şekilde bulunan mesafe yayın yarıçapından büyük ise doğrunun ve çember kesişmiyor demektir ve kesişim noktasının bulunması işlemine gidilmeyecektir.



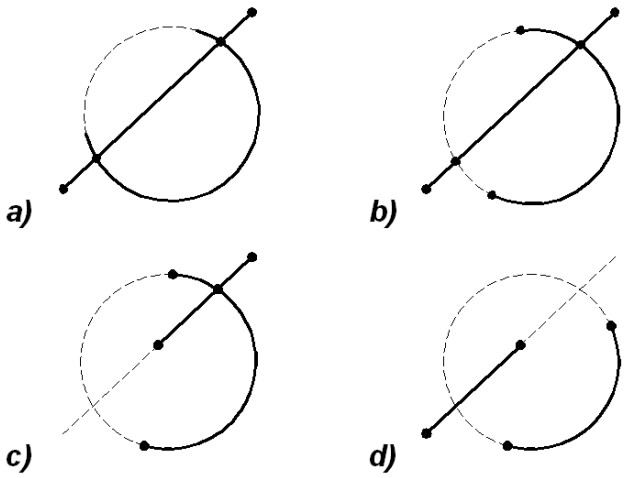
Şekil 4.16. Yay merkezlerinin doğruya dik uzaklıkları

Şekil 4.17.'de görüldüğü gibi ortak çözüm denkleminin kullanılmasıyla kesişim noktası bulunur.



Şekil 4.17. Doğru ve yayın kesişim noktasının bulunması

Ortak çözüm denkleminin çözüm kümesinin bulunmasından sonra bulunan kesişim noktalarının doğru ve yayın her ikisinin de aralığında olup olmadığı kontrol edilmiştir. Şekil 4.18.'de görüldüğü gibi kesişim noktalarından; Şekil 4.18.a 'da iki kesişim noktası, Şekil 4.18.b 'de ve Şekil 4.18.c'de bir kesişim noktası doğru ve yayın aralığındadır. Şekil 4.18.d'de ki durumda ise kesişim noktalarının ikisi de aralıkların dışında olduğu için doğru ve yay kesişmemektedirler.



Şekil 4.18. Doğru ve yayın kesişim durumları

Bir doğru ve yayın kesişim noktalarını tespit eden program kodları EK-4 'te verilmiştir.

Uygulanan bu işlemler ile doğru ve yayın kesişim noktaları hesaplanarak kırılma noktaları belirlenmiştir. Daha sonra kırılacak unsur atılmakta, kırılan unsurun uç noktaları ve hesaplanan kırılma notaları ile sınırlandırılmış yeni ofset unsurları oluşturulmaktadır.

4.3.3. İki yayın kesişimi

Kırılacak unsurun ve onu kesen unsurun yay olması durumunda kırma işleminin uygulanabilmesi için kırılma noktalarının hesaplanması gerekmektedir. Kesişim noktası hesaplandıktan sonra yay unsuru bu noktadan kırılarak parçalara ayrılmaktadır. İki yayın kesişim noktası bulunurken yayların ikisinin de ayrı ayrı analitik denklemleri Eş. 4.22 ve Eş. 4.23 oluşturulmuştur.

$$r1^2=(x-a1)^2+(y-b1)^2 \tag{4.22}$$

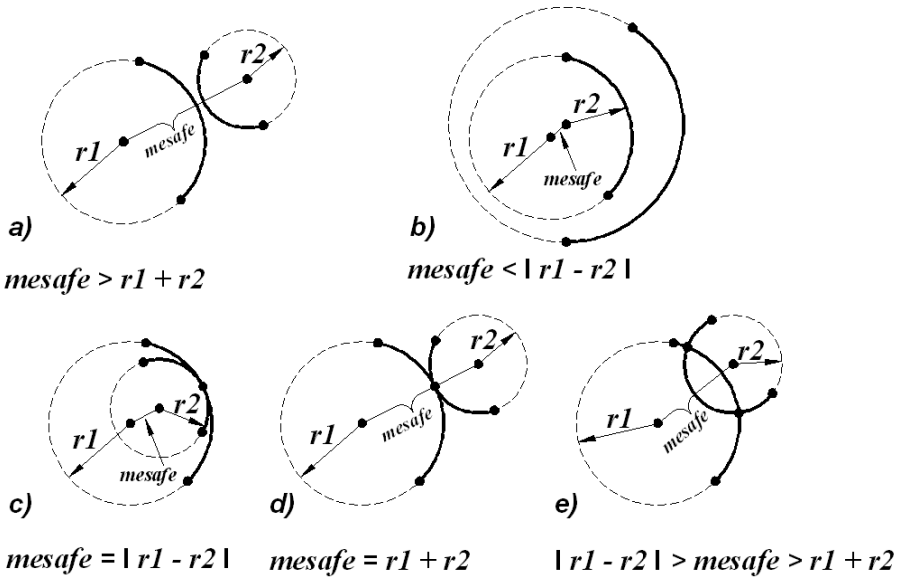
$$r2^2=(x-a2)^2+(y-b2)^2 \tag{4.23}$$

Oluşturulan bu analitik denklemlerinin ortak çözümü ile Eş. 4.24 ve Eş. 4.25'de verilen denklemler elde edilmiştir.

$$4.x^2[(a1-a2)^2+(b1-b2)^2]+x.[4.(a1-a2).(a2^2-a1^2+r1^2-r2^2+(b1-b2)^2)-8.a1.(b1-b2)^2] + (a2^2-a1^2+r1^2-r2^2+(b1-b2)^2)^2-4.(b1-b2)^2.(r1^2-a1^2)=0 \tag{4.24}$$

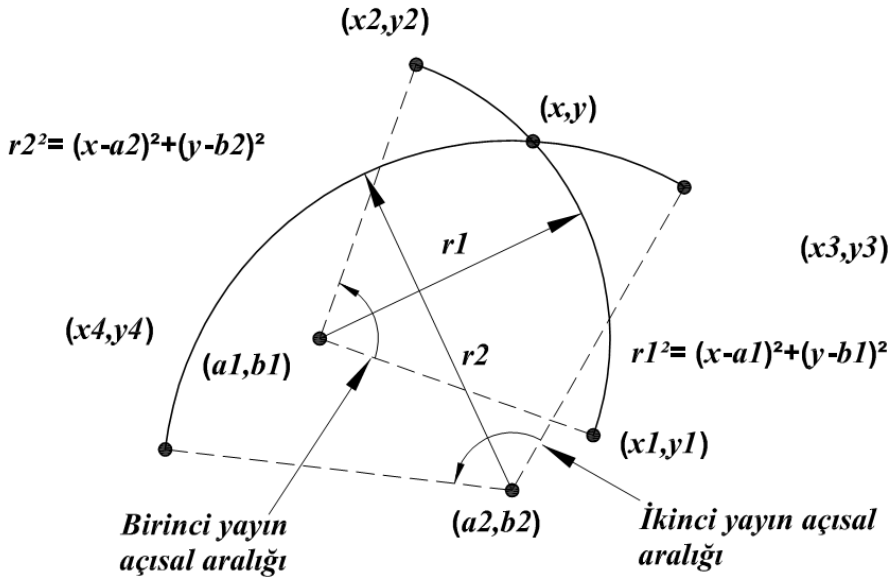
$$y=(r1^2-(x-a1)^2)^{0.5}-b1 \tag{4.25}$$

Elde edilen ortak çözüm denklemlerinin çözüm kümesi iki yayın kesişim noktalarını vermektedir.



Şekil 4.19. Yay merkezleri arasındaki mesafeler

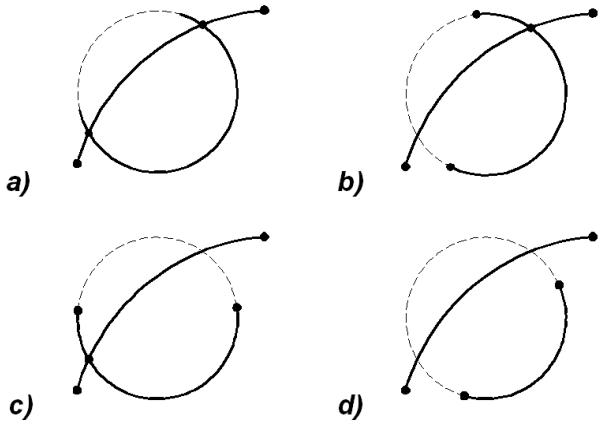
Kesişim noktalarının hesaplanması işlemine gidilmeden önce bu iki yayın kesişip kesişmediğinin ön kontrolü için yayların merkezleri arasındaki mesafe hesaplanmıştır. Bulunan bu mesafe iki yayın yarıçapları toplamından büyük (Şekil 4.19.a) veya yayların yarıçaplarının farkından küçük (Şekil 4.19.b) ise bu iki yay kesişmiyor demektir. Bulunan mesafe yayların yarıçapları toplamına veya yarıçapları farkına eşit ise (Şekil 4.19.c) ve Şekil 4.19.d) bu iki yay tek noktada kesişiyor yani teğettir. Bulunan, merkezler arası mesafe her iki yayın yarıçapları toplamından küçük ve yarıçaplarının farkından büyük ise (Şekil 4.19.e) bu iki yay iki noktada kesişiyor demektir.



Şekil 4.20. İki yayın kesişim noktasının bulunması

Şekil 4.20.'de görüldüğü gibi ortak çözüm denkleminin kullanılmasıyla kesişim noktası bulunur.

Yayların kesişip kesişmediğini tespit etmek için ortak çözüm denkleminin çözüm kümesinin bulunmasından sonra bulunan kesişim noktalarının her iki yayın da açısal aralığında olup olmadığı kontrol edilmiştir. Şekil 4.21'de görüldüğü gibi kesişim noktalarından; Şekil 4.21.a'da iki kesişim noktası, Şekil 4.21.b'de ve Şekil 4.21.c'de bir kesişim noktası doğru ve yayın aralığındadır. Şekil 4.21.d'de ki durumda ise kesişim noktalarının ikisi de aralıkların dışında olduğu için doğru ve yay kesişmemektedirler.



Şekil 4.21. İki yayın kesişim durumları

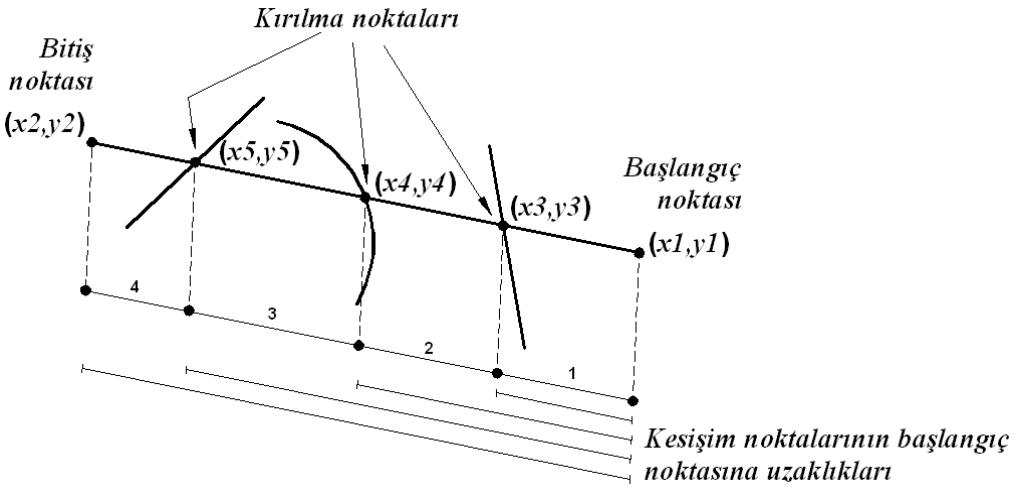
İki yayın kesişim noktalarını tespit eden program kodları EK-5 ‘te verilmiştir.

Uygulanan bu işlemler ile iki yayın kesişim noktaları hesaplanarak kırılma noktaları hesaplanmıştır. Daha sonra kırılacak unsur atılmakta, kırılan unsurun uç noktaları ve hesaplanan kırılma notaları ile sınırlandırılmış yeni ofset unsurları oluşturulmaktadır.

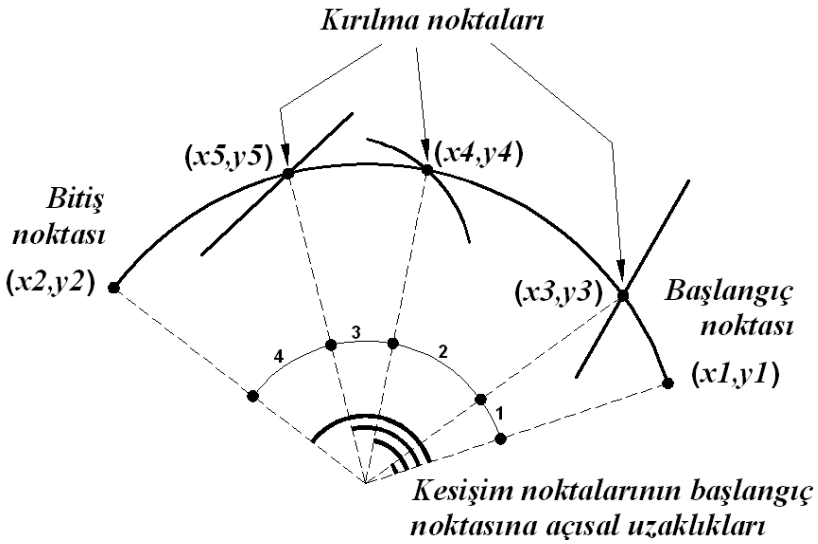
4.3.4. Kesişim noktalarının sıralanması ve yeni unsurların oluşturulması

Her hangi bir doğru veya yayın birden fazla noktada kesişmesi durumunda unsurun kesişim noktalarından kırılabilmesi için kesişim noktalarının belirli bir sıra dâhilinde olmaları gerekmektedir. Kamaşık bir sıralama ile oluşturulan yeni kırık parçalar bir biri üzerine bineceğinden hatalı ofset kısımlarının atılmasına imkân vermeyecektir. Bu nedenle kesişim noktaları unsurun her hangi bir ucuna göre olan uzaklıklarına göre sıralı olmak zorundadırlar. Sıralama işlemi için uzaklıklar unsurun başlangıç noktası referans alınarak unsurların bu referansa göre olan uzalıkları değerlendirilmiştir. Doğru ve yayların kesişim noktalarının bulunmasından sonra bu kırılma noktaları doğru veya yayın başlangıç noktasına olan uzaklıklarına göre sıralanmıştır. Doğrularda kesişme noktasıyla doğrunun başlangıç noktası arasındaki dik uzaklık kullanılırken, yaylarda kesişim noktası ile yayın başlangıç noktasının açısal aralığı kullanılmıştır. Sıralama işleminden sonra doğrular ve yaylar başlangıç noktasından başlayan ve kendisine en yakın kırılma noktasında biten, en yakın

kırılma noktasında başlayıp sonraki en yakın kırılma noktasında biten şeklinde ve en sonunda en uzak kesişme noktasında başlayıp bitiş noktasında biten unsur parçalarına bölünürler (Şekil 4.22 ve Şekil 4.23).



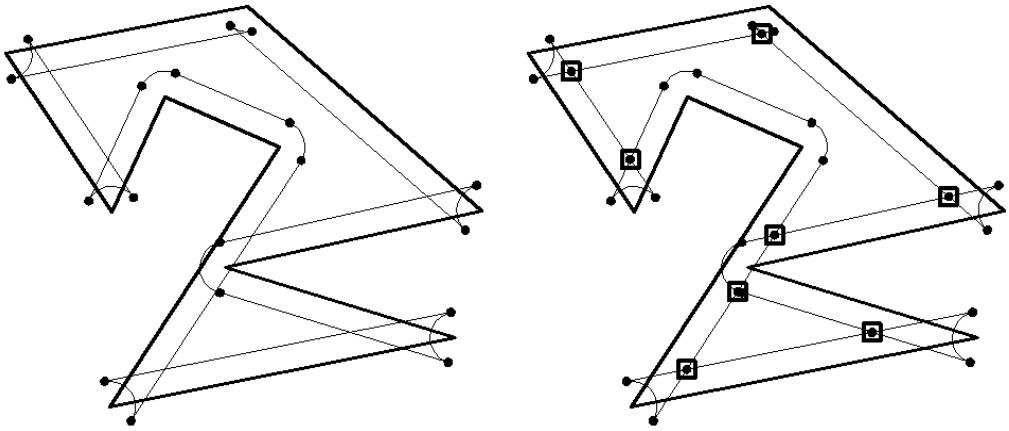
Şekil 4.22. Doğrularda kesişim noktalarının sıralanması



Şekil 4.23. Yaylarda kesişim noktalarının sıralanması

Kırma işlemi ile herhangi bir ofset unsuru ile kesişen her unsur kesişim noktasından bölünerek farklı unsurlar şeklinde yeniden oluşturulmuştur. ‘n’ sayıda kesişim noktası olan bir unsur yerine onu oluşturan ‘n + 1’ sayıda yeni unsur kullanılmıştır.

Doğru ve yayların kırılması işleminden sonra kaba ofsetleme ile elde edilen ofset unsur grubu bir birini kesmeyen müstakil unsurlardan oluşan yeni bir ofset unsur grubu haline gelir (Şekil 4.24.).



a) Kırma işleminden önce **b) Kırma işleminden sonra**

Şekil 4.24. Kırılmamış ve kırılmış ofset unsur grupları

Kırma işlemi için unsurların kesişim noktalarını bulduran, bulunan kesişim noktalarını sıralayan ve yeni unsurları oluşturan program kodları EK-6'da verilmiştir.

4.4. Hataların Giderilmesi

Kaba ofsetleme işleminde ofsetleme hataları göz ardı edilerek sadece unsurlar ofsetlenmişti. Hata ayıklama uygulamasında ise ofsetleme ile gelen hatalı fazlalıklar tespit edilerek atılmıştır. Hatalı fazlalıkların atılmasıyla da geriye geçerli ofset kısımları kalmış ve ofsetleme işlemi tamamlanmıştır.

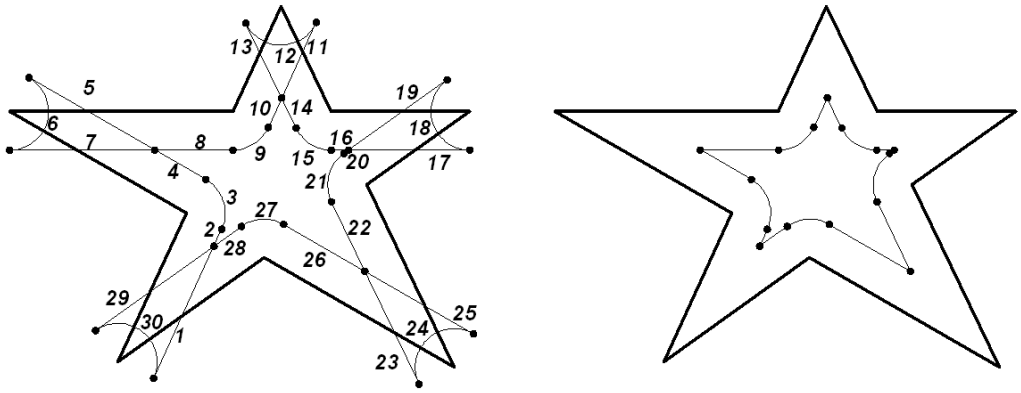
Ofsetleme işlemi ile oluşan hatalı fazlalıkları tespit edebilmek için öncelikle geçerli bir ofset unsurunun taşıması gereken ortak özellikler belirlenmiştir. Bu ortak özellikleri taşımayan ofset kısımları atılarak hatalar giderilmiştir. Bir ofset unsurunun geçerli sayılabilmesi için taşıması gereken şartlar aşağıda maddeler halinde sıralanmıştır.

1) Geçerli bir ofset unsuru ofsetlenen ana profili kesmez. Yani bir ofset unsurunun geçerli olabilmesi için ana profili oluşturan yay ve doğrulardan hiç birisi ile kesişmiyor olması gerekmektedir.

2) Geçerli bir ofset unsuru ofsetlenen ana profile ofset mesafesinden daha yakın olamaz. Yani bir ofset unsurunun geçerli olabilmesi için ana profili oluşturan yay ve doğrularla arasındaki mesafenin, ofset mesafesinden daha küçük olmaması gerekmektedir.

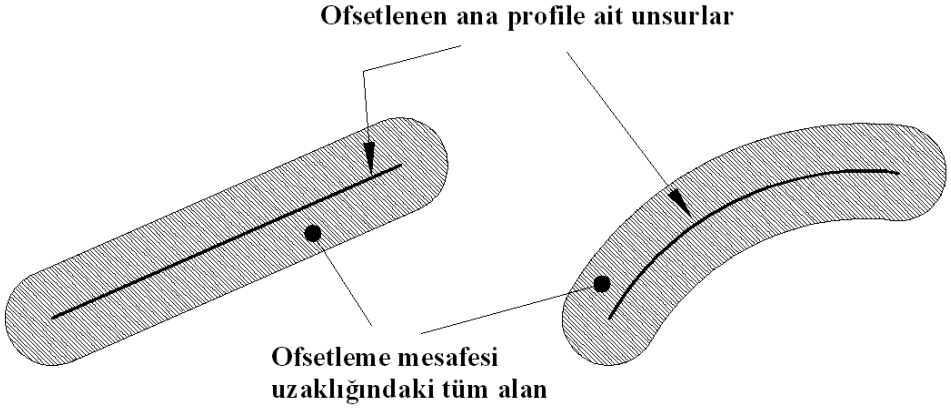
Bir ofset unsurunun, 1 numaralı geçerlilik şartını sağlayıp sağlamadığı kontrol etmek için, doğru ve yayların kırılması işleminde kullanılan denklemler kullanılır. Kontrol edilen ofset unsurunun analitik denklemi ile ana profili oluşturan doğru ve yayların analitik denklemlerinin ayrı ayrı ortak çözümleri bulunarak kesişme durumları tespit edilir. Ana profili oluşturan unsurlardan herhangi birisi ile bir kesişme durumu var ise bu ofset unsuru geçersiz kabul edilir ve atılır. Herhangi bir kesişme söz konusu değil ise o ofset unsuru geçerli sayılır ve atılmaz.

Şekil 2.25.'de görüldüğü gibi 1) numaralı şartı sağlamayan 1.,5.,6.,7., 11., 12., 13., 17., 18., 19., 23., 24., 25., 29. ve 30. ofset unsurları atılmıştır.



Şekil 4.25. Hatalı ofset unsurlarının atılması

Bir ofset unsurunun, 2 numaralı geçerlilik şartını sağlayıp sağlamadığı kontrol edilirken, ana profili oluşturan unsurlardan her birinin ofsetleme mesafesi kadar uzağındaki tüm alan belirlenir (Şekil 4.26).



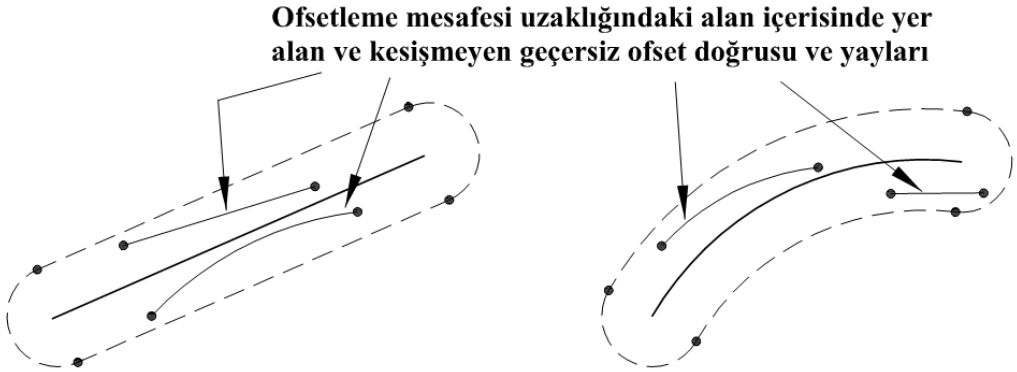
Şekil 4.26. Ofsetleme mesafesi uzaklığındaki tüm alan

Bu alanı sınırlandıran doğru ve/veya yaylar ile geçerliliği aranan ofset unsurunun kesişip kesişmediği kontrol edilir (Şekil 4.27.). Kesişim var ise o ofset unsuru kesişimi olan ilgili ana profil unsuruna ofsetleme mesafesinden daha yakın demektir.

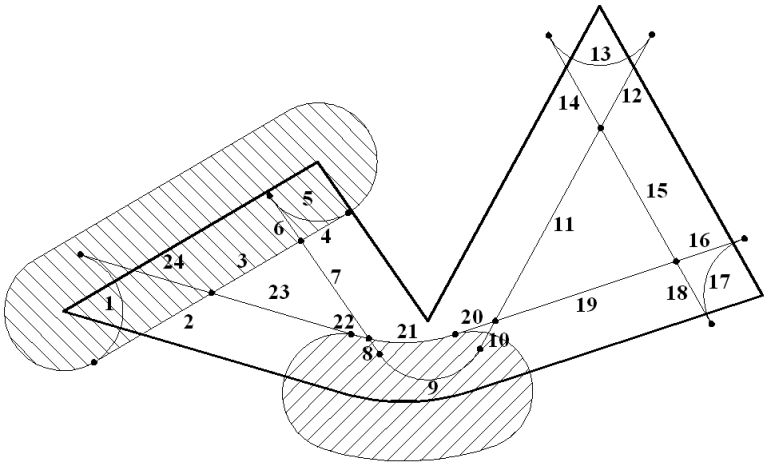
Şekil 4.27. Sınır doğruları ve yayları

Ofsetleme mesafesi uzaklığındaki doğru ve/veya yaylardan herhangi birisi ile kesişen ofset unsurları geçersiz kabul edilip atılacaktır. Bu sınır doğruları ve yayları ile kesişmediği halde ana profil unsuruna ofset mesafesinden daha yakın olduğu ayrı bir durum vardır (Şekil 4.28). Bu durumda ofset unsuru tamamen bu alan içinde yer

alırken, hem sınır unsurları ile hem de ana profil unsuru ile kesişmemektedir. Böyle durumlarda ofset unsurunun başlangıç ve bitiş noktalarının ana profil unsuruna olan uzaklıklarının da ana profil unsuruna ofset mesafesinden daha yakın olup olmadığı kontrol edilir.

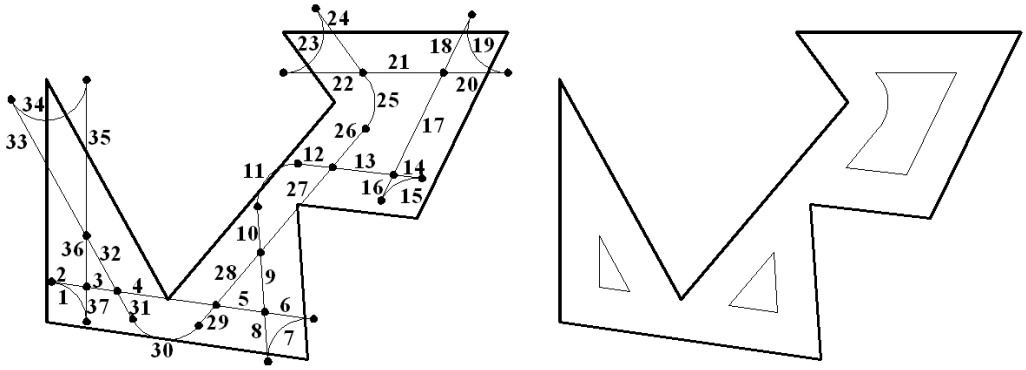


Şekil 4.28. Kesişmeyen ofset unsurları



Şekil 4.29. Ofset mesafesi alanı içinde kalan ofset unsurları

Şekil 4.29'da 4.,5. ve 6., Şekil 4.30'da 1., 2., 3., 5., 9., 10., 12., 14., 15., 16., 29., 31. ve 37. ofset unsurları 2 numaralı şartı sağlamadıkları için atılmışlardır.



Şekil 5.30. Ana profile ofset mesafesinden daha yakın ofset unsurlarının atılması

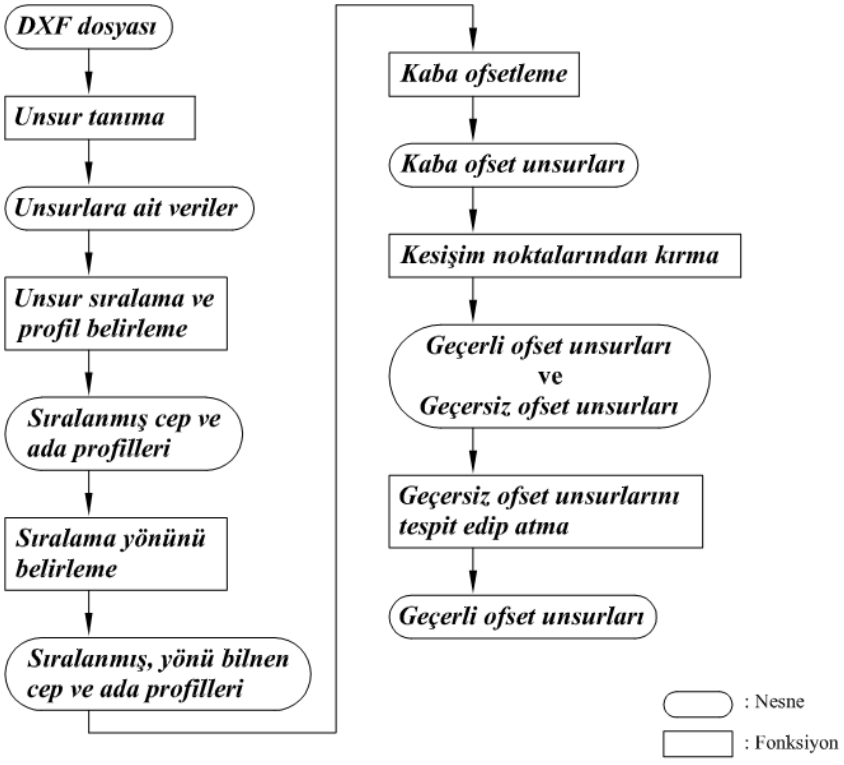
Geçerli bir ofset unsuru şartlarını taşımayan ofset unsurlarının atılmasıyla geriye geçerli ofset unsurları kalmakta ve ofsetleme işlemi başarıyla gerçekleştirilmektedir (Şekil 5.30).

5. PROGRAMIN GELİŞTİRİLMESİ VE UYGULAMALAR

Bu bölümde sunulan ofsetleme yönteminin uygulanabilmesi için bir program algoritması geliştirilerek yazılım oluşturulmuştur. Gerçekleştirilen örnek uygulamalar ile önerilen yöntemin doğruluğu sınanmıştır.

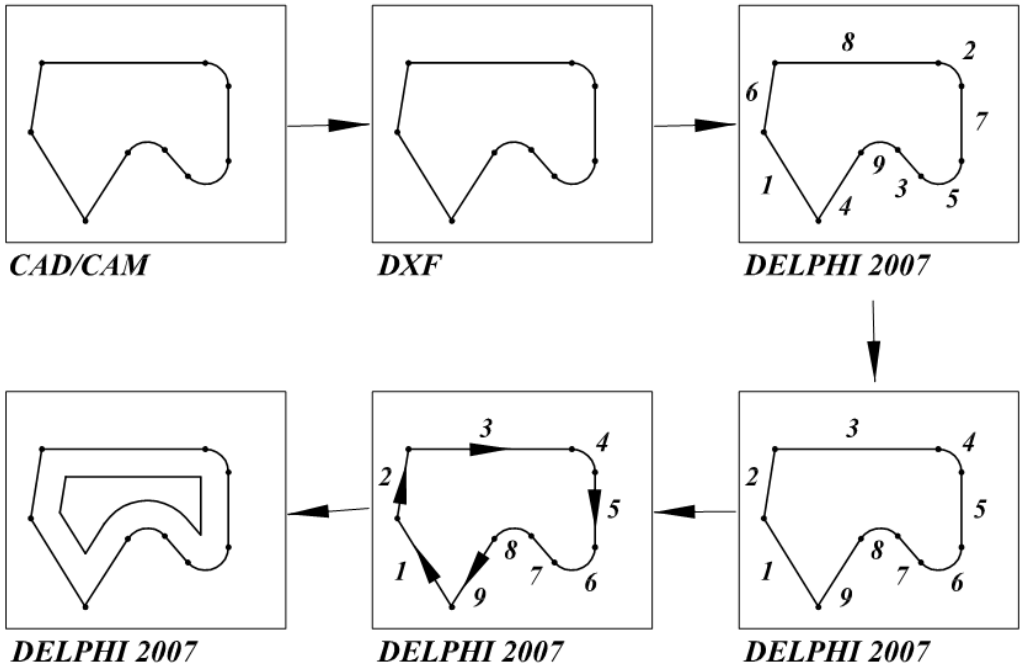
5.1. Programda Kullanılan Algoritma

Ofsetleme algoritmasının doğruluğunun denenmesi için Delphi 2007 ortamında bir program geliştirilmiştir. Geliştirilen programda, cep işleme için gerekli çizimin herhangi bir CAD ortamında tasarlandıktan sonra DXF formatına dönüştürülen dosyasının açılması ile birlikte bir takım işlemler belirli bir hiyerarşiye göre otomatik olarak gerçekleştirilmektedir. Şekil 5.1'de programda kullanılan algoritmanın şematik yapısı görülmektedir.



Şekil 5.1. Ofsetleme algoritmasının şematik yapısı

Kullanıcı tarafından DXF dosyasını okuyabilen bir BDT/BDİ yazılımı ile tasarlanan cep profili DXF dosya formatı şeklinde kaydedilmiştir. Oluşturulan bu dosya programda girdi olarak kullanılmıştır. DXF dosyasında kayıtlı olan cep profiline ait veriler program tarafından okunarak program ortamına alınmıştır (Şekil 5.2). Profile ait tasarım verilerinin program ortamına alınmasından sonra yapılan ilk işlem unsurların tanınması işlemidir. Cep profilini oluşturan doğruları ve yaylara ait tanımlama bilgileri okunarak programın kendi ortamındaki dizi değişkenlere atanmıştır. Bu aşamada DXF verilerinden direkt olarak elde edilemeyen tasarımdaki yay unsurlarının başlangıç ve bitiş noktaları da trigonometrik oranlarla bulunmuş ve yine ilgili değişkenlere atanmıştır.



Şekil 5.2. Tasarım verisinin ofset unsurlarına dönüşümü

Cep profilinin içinde adaların olabileceğinden ada profilini oluşturan doğru ve yaylara ait veriler de DXF dosyasından okunarak program ortamına alınmıştır. Program ortamına alınan unsurlar dizi değişkenlere atanmadan önce DXF dosyasında kaç tane unsurun bulunduğu program tarafından sayılmış ve dizi değişken bu unsur sayısına göre boyutlandırılmıştır. Ancak tanımlanan unsurların cep veya ada

profilinden hangisine ait olduğunun bilgisi DXF dosyasında bulunmamaktadır. Bu nedenle bu bilgiler program ortamında trigonometrik oranlarla elde edilmiştir.

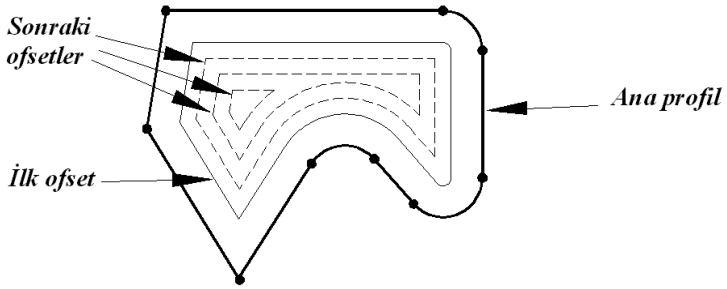
Takım yollarının düzgün bir şekilde oluşturulabilmesi için kullanıcının gelişi güzel bir sıralamayla çizdiği profil unsurlarını zincir şeklinde devam eden ardışık bir sıralamaya sokmak için program ortamında bir alt program (prosedür) oluşturulmuştur. Oluşturulan bu alt program, sıralama işlemine başlamadan önce sıralamaya başlayacağı ilk nokta olan referans noktasını bulabilmek için profillerin üzerindeki en küçük x ve en küçük y kartezyen koordinat değerlerini bulmuştur. Daha sonra (en küçük x,en küçük y) koordinatına en yakın olan unsurun uç noktasını (referans noktası) bulmuştur. Alt program bir noktaya en yakın uç noktasını bulurken o nokta ile diğer bütün noktaların aralarındaki uzaklıkları tek tek hesaplanarak en küçüğünü belirlemiştir. Takım yolları oluşturulurken cep profili içe, ada profili de dışa doğru ofsetlendiğinden hangi unsurun hangi profile ait olduğunun bilinmesi önemlidir. Oluşturulan bu alt program ile unsurlar hem sıralanmış hem de cep profili içindeki adalar ve hangi unsurun hangi adaya ait olduğu belirlenmiştir. Ayrıca her unsur için kendinden önceki ve sonraki komşu unsurlar tespit edilmiştir.

Profilleri oluşturan unsurların sıralanmasından sonra bu sıralamanın hangi yönde yapıldığının bulunması için program içerisinde ayrı bir alt program daha geliştirilmiştir. Cep ve ada profillerinin içe ya da dışa doğru ofsetlenebilmesi için sıralama yönünün bilinmesi gerekmektedir. Geliştirilen alt program profillerin üzerindeki belirli açıları karşılaştırarak sıralama yönünü tayin etmiştir.

Program ortamında sıkça kullanılan açı bulma, iki nokta arasındaki uzaklığı bulma gibi işlemler için fonksiyonlar oluşturularak hesaplama işlemleri daha basit bir yapıya indirgenmiştir.

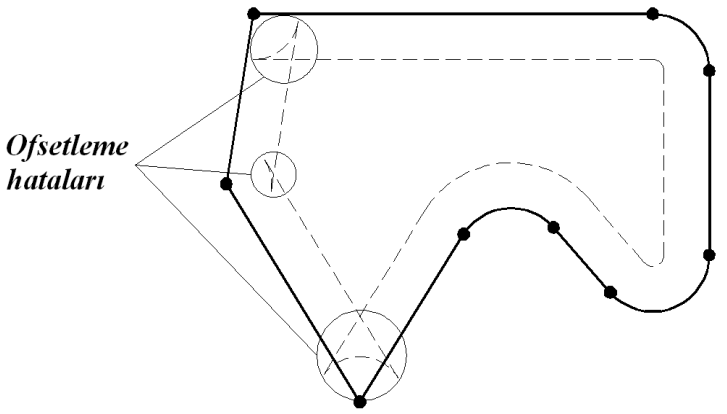
Sıralama işleminin tamamlanmasından sonra kaba ofsetleme ofsetleme alt programı cep ve ada profillerine ait doğruları, yayların ve bunların birleştiği düğüm noktalarının ofsetlerinin koordinatlarını hesaplamaktadır. Bulunan ofset unsurlarına

ait veriler de ayrı bir dizi değişkene atanmaktadır. Ofsetleme işlemi gerçekleştirirken kullanılan ofsetleme mesafesi ilk ofsetlemede takımın yarıçapı kadar alınırken sonraki ofsetleme işlemlerinde ise ofsetleme mesafesi, kesici takım için belirlenen yana kayma mesafesi kadar alınmaktadır (Şekil 5.3).



Şekil 5.3. Ofsetleme mesafeleri

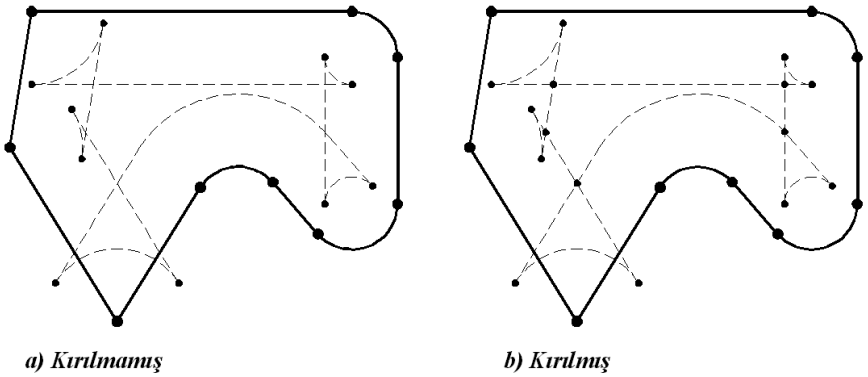
Kaba ofsetleme prosedürü cep profilini içe ada profillerini de dışa doğru ofsetlemektedir. Kaba ofsetleme işlemi gerçekleştirilirken ofsetleme hataları göz ardı edilmiştir (Şekil5.4).



Şekil 5.4. Kaba ofsetleme ve ofsetleme hataları

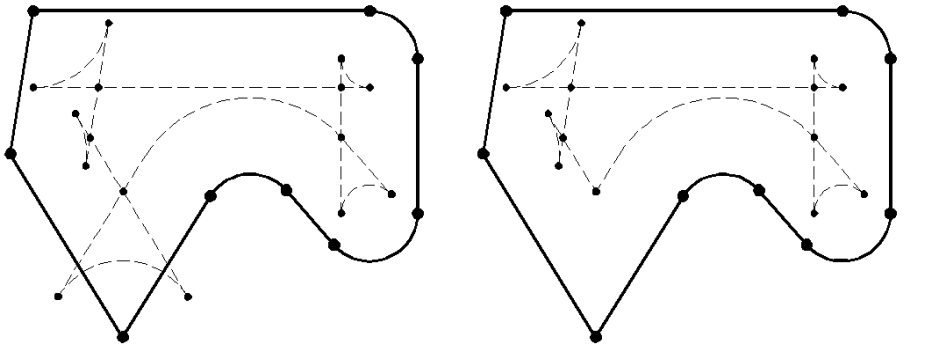
Kaba ofsetleme işleminden sonra ofsetleme ile oluşan hatalı fazlalıkların atılmasını kolaylaştırmak için ofset doğrularını kesişim noktalarından kıran bir kırma alt programı geliştirilmiştir. Bu alt programda kesişim denklemleri kullanılarak her bir

ofset unsurunun (ofset doğrusu veya ofset yayı) diğer tüm ofset unsurları ile kesişim durumları kontrol edilmiştir. Her bir ofset unsuru için kendi üzerindeki bütün kesişim noktaları bulunarak ayrı bir dizi değişkene aktarılmıştır. Bu kesişim noktaları ofset unsurunun başlangıç noktasına olan uzaklıklarına göre sıralanmıştır. Daha sonra ofset unsurları, başlangıç ve bitiş noktaları da dâhil olmak üzere her iki kesişim noktası arasındaki kısımlar ayrı bir parça olacak şekilde kırılarak yeni ofset unsurları haline getirilmiştir (Şekil 5.5).



Şekil 5.5. a) Kırılmamış ofset unsurları, b) Kırılmış ofset unsurları

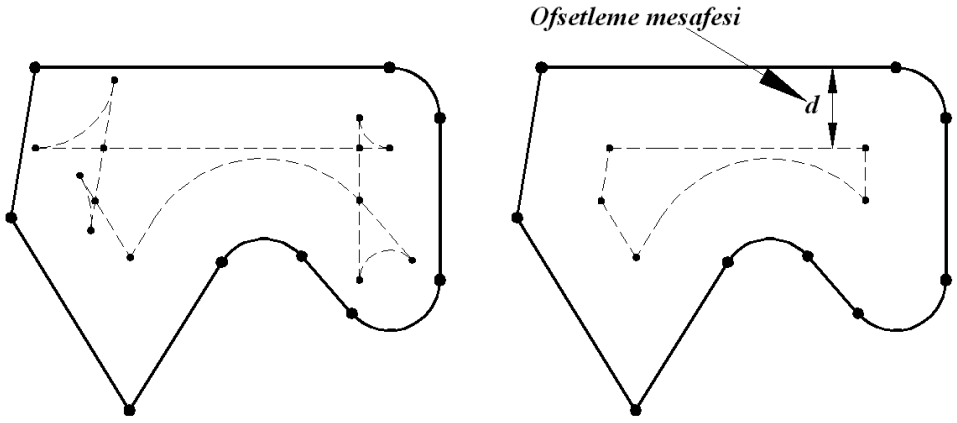
Kırma işleminden sonra ofsetleme hatalarının giderilmesi için bir hata ayıklama alt programı oluşturulmuştur. Bu prosedür, geçerli bir ofset unsuru için belirlenen şartları taşımayan ofset unsurlarını tespit ederek atmaktadır. Bu işlem iki adımda gerçekleştirilmiştir. İlk adımda cep profilini veya ada profilini kesen ofset unsurları kesişim denklemleri kullanılarak tespit edilmiş ve tespit edilen hatalı unsurlar atılmıştır (Şekil 5.6).



a) Geçerli ve geçersiz ofset unsurları *b) Ana profili kesen ofset unsurları atılmış*

Şekil 5.6. Hata ayıklama prosedürünün birinci adımı

İkinci adımda ise cep ve ada profiline ofsetleme mesafesinden daha yakın olan ofset unsurları yine kesişim denklemleri ve iki nokta arasındaki uzaklığı bulan fonksiyonların kullanımı ile tespit edilmiş ve tespit edilen bu hatalı unsurlar da atılmıştır (Şekil 5.7).



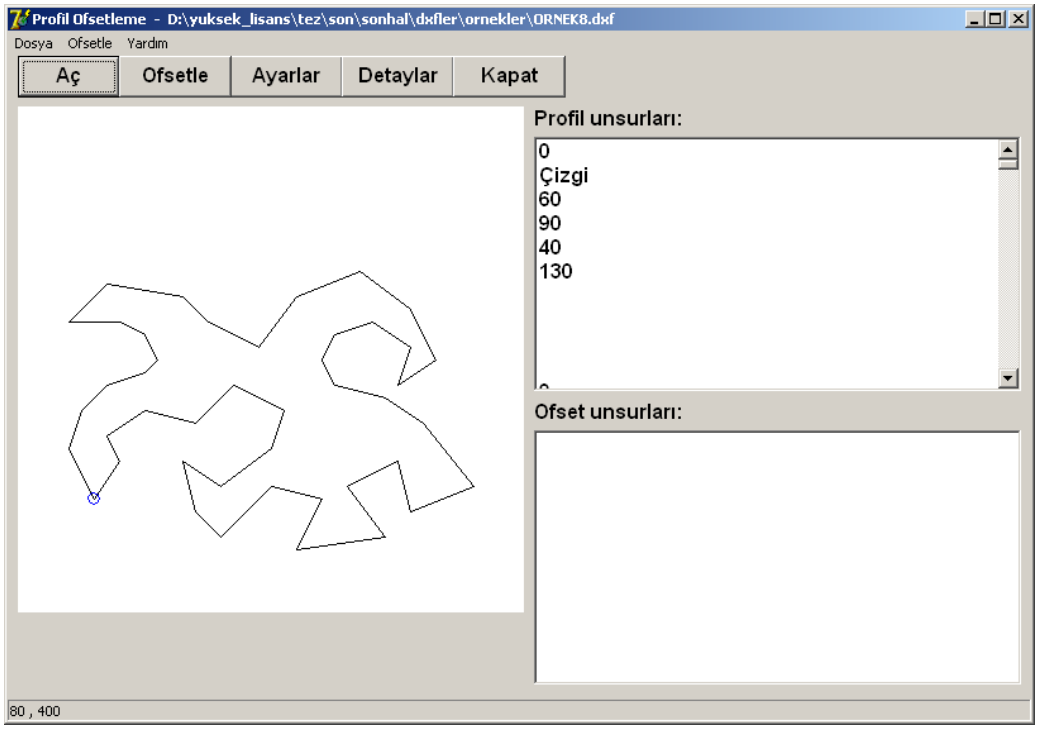
a) Ana profili kesen ofset unsurları atılmış *b) Ana profile ofset mesafesinden daha yakın olan ofset unsurları atılmış*

Şekil 5.7. Hata ayıklama prosedürünün ikinci adımı

Ofsetleme hatalarının atılmasıyla doğru bir ofsetleme işlemi gerçekleştirilmiştir.

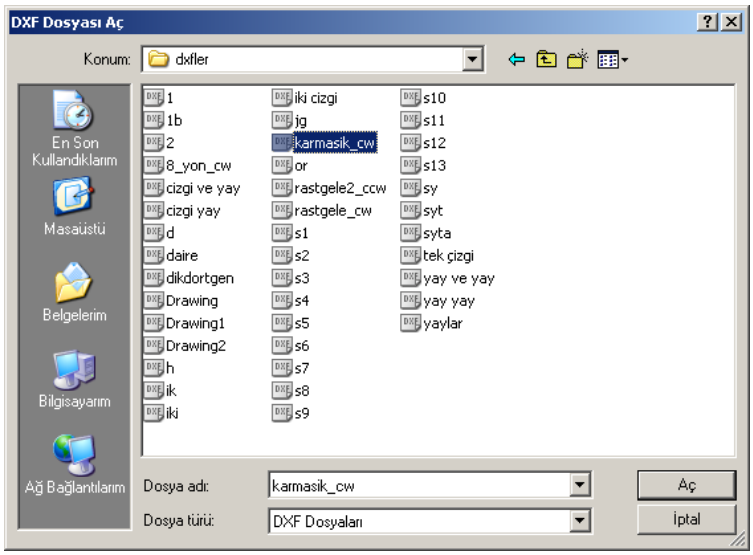
5.2. Program ve Örnek Uygulamalar

Bir önceki bölümde sunulan program algoritması Delphi 2007 ortamında bir program haline dönüştürülmüştür. Geliştirilen programın ara yüzü Şekil 5.8'de görülmektedir.



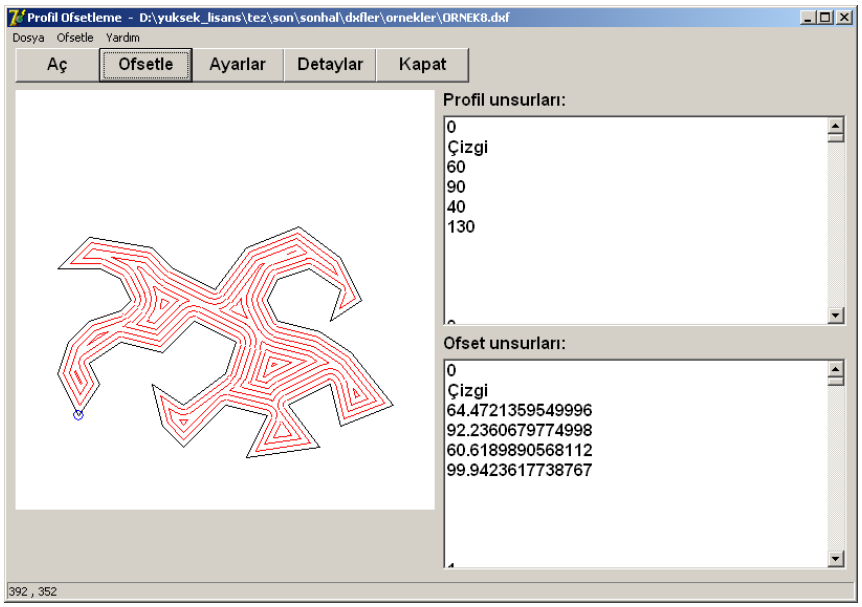
Şekil 5.8. Geliştirilen programın ara yüzü

Program kullanılırken ilk önce, daha önceden oluşturulmuş, cep profiline ait verileri bulunduran DXF dosyası okunmaktadır. DXF dosyasının okunması ile dosyada bulunan unsurlara ait veriler değerlendirilerek ofsetlenecek profil programın ekranına çizdirilmektedir (Şekil 5.8). Bir DXF dosyasının çağrılarak program ortamına alınması Şekil 5.9'da görülmektedir.



Şekil 5.9. DXF dosyasının çağrılması

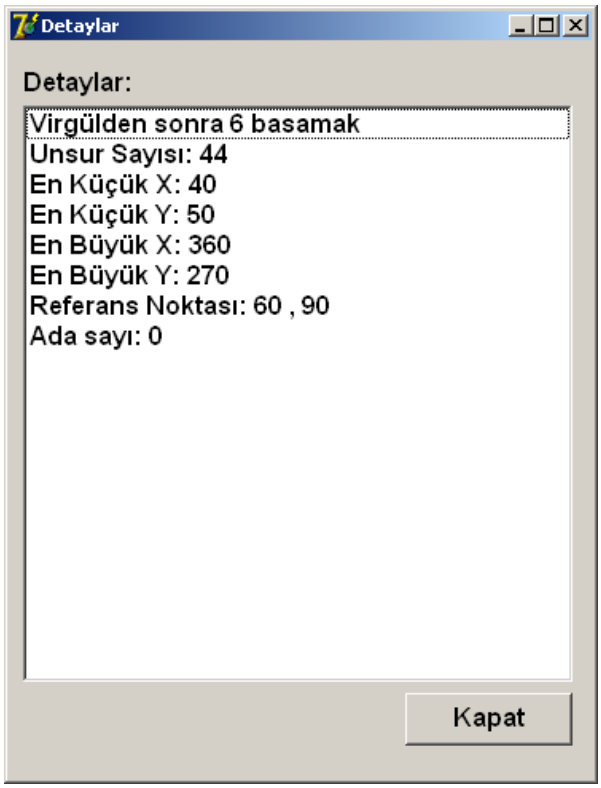
DXF dosyasını okunup verilerin program ortamına alınmasında sonra programın ara yüzünde yer alan veri kutusuna uygulanacak ofsetleme mesafesi girilmektedir. Ofsetleme mesafesinin girilmesinden sonra yine ara yüzde yeralan ofset düğmesine basılarak ofsetleme işlemi icra edilmekte ve ofset doğruları programın ekranına çizdirilmektedir.



Şekil 5.10. Program ile gerçekleştirilen ofsetleme işlemi

Program ile gerçekleştirilmiş bir ofsetleme işlemi Şekil 5.10'da görülmektedir. Ofsetleme işlemi veri kusuna girilen ofsetleme miktarı her seferinde bir kat daha artırılarak ofsetleme işlemi tekrarlanmaktadır. Ofsetleme işleminin tekrarlanması her seferinde profilin artan ofsetleme mesafesi değerlerinde yapılan yeni bir ofsetleme işlemi şeklinde gerçekleşmketedir. Ofsetleme işlemi, ofsetlenen profilin içerisinde ofsetleme işleminin gerçekleştirilemeyecek kadar daralması durumuna kadar devam edip sona ermektedir.

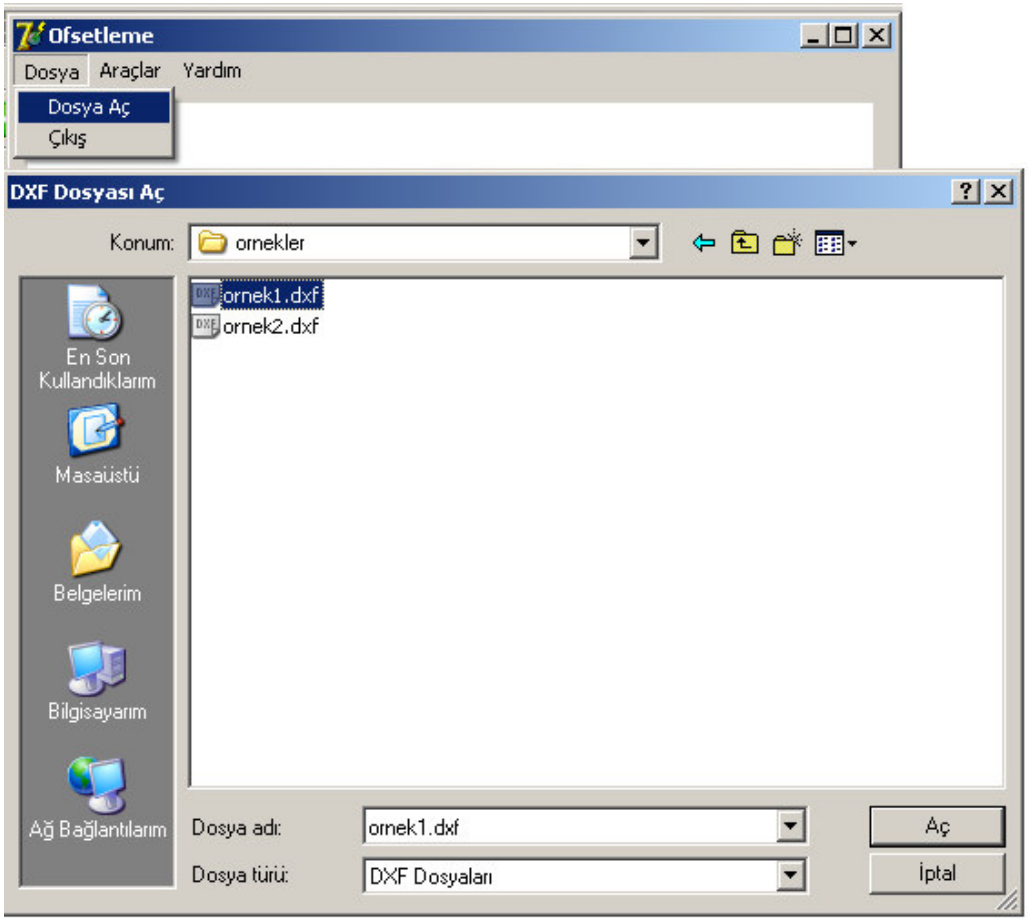
Hazırlık aşaması ve ofsetleme işlemi sırasında kullanılan parameteler, hesaplanan sıralama yönleri ve elde edilen ofset unsurlarına ait veriler, programın detay penceresindeki listelerden izlenebilmektedir. Şekil 5.11'de ise Ofsetleme parametreleri, ofsetlenen unsurlar ve ofset unsurlarının listesi görülmektedir.



Şekil 5.11. Ofsetleme parametreleri, ana unsurlar ve ofset unsurları

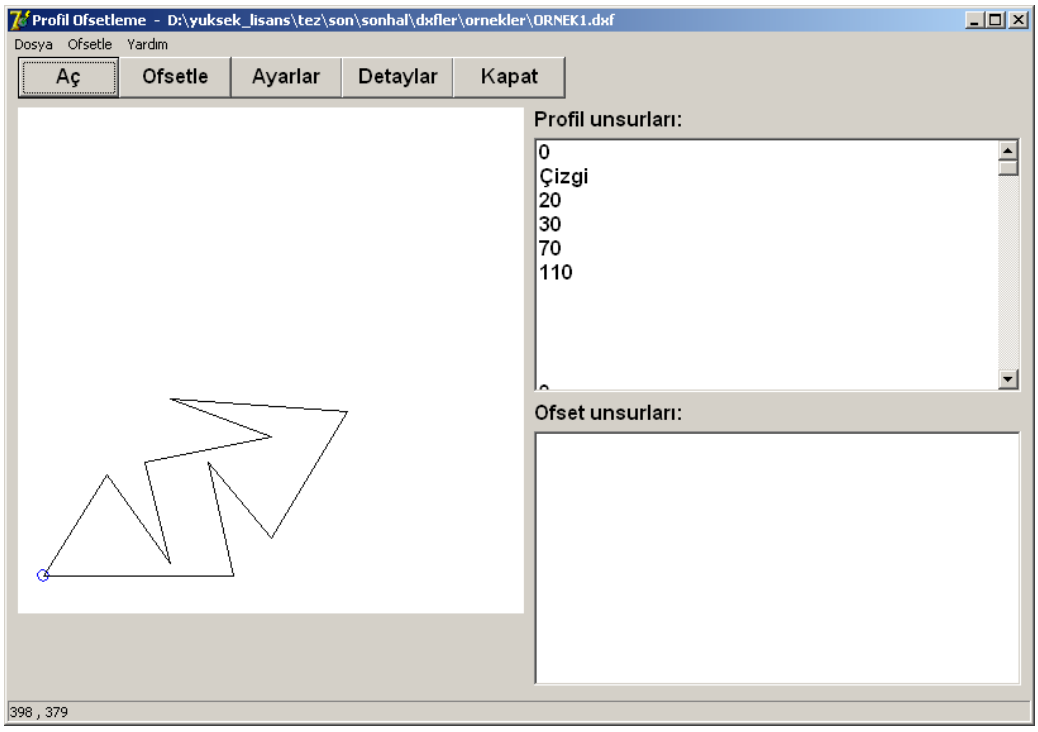
5.2.1. Örnek uygulama 1

Programın çalıştırılmasımasından sonra 'Dosya' ve 'Aç' komutları sırası ile çalıştırılarak 'ornek1.dxf' adlı DFX dosyası program tarafından açılmıştır (Şekil 5.12).



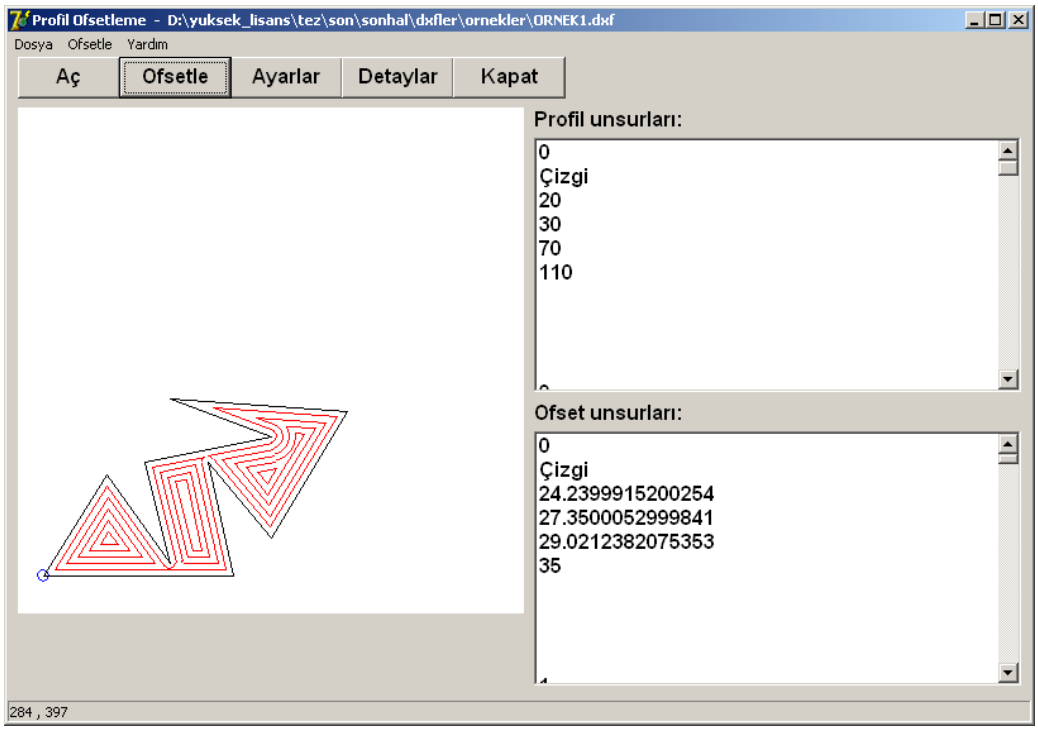
Şekil 5.12. 'ornek1.dxf' dosyasının açılması

'ornek1.dxf ' dosyasının program tarafından açılarak okunmasıyla dosya içinde bulunan unsur verileri alınmış ve ofsetlenecek cep profili programın ekranına çizdirilmiştir (Şekil 5.13).



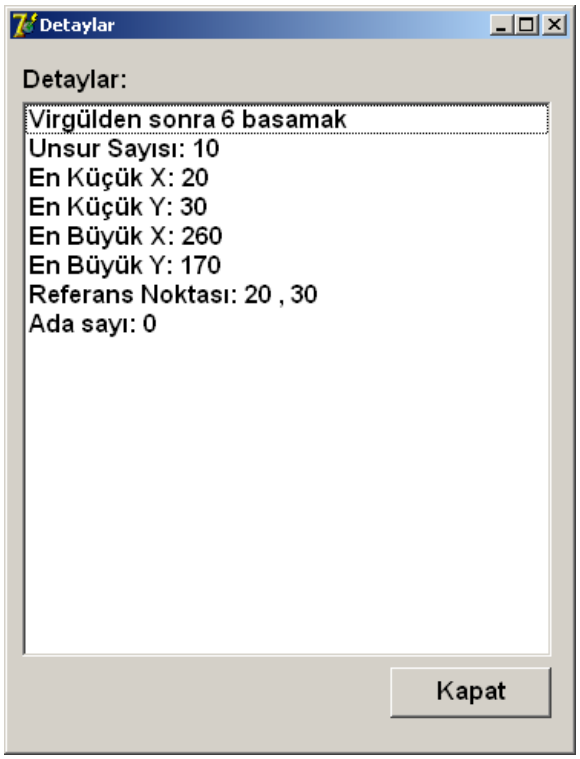
Şekil 5.13. ‘ornek1.dxf’ dosyasındaki profilin ekrana çizdirilmesi

Daha sonra veri kutusuna ‘5mm’lik ofsetleme mesafesi girilerek ‘ofsetle’ düğmesine basılmış ve ofsetleme işlemi gerçekleştirilmiştir. Ofsetleme işlemi 5 mm, 10 mm, 15 mm, 20 mm ve 25 mm’lik ofsetleme mesafelerinde beş defa tekrarlanmıştır (Şekil 5.14).



Şekil 5.14. Ofsetleme işleminin gerçekleştirilmesi

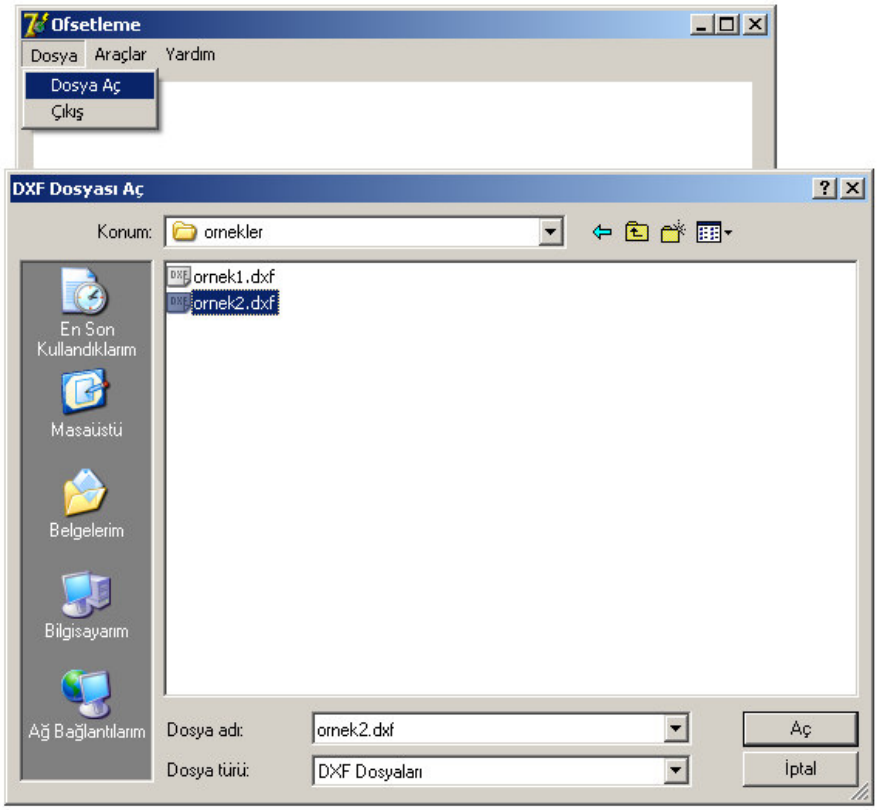
Elde edilen ofset doğrularına ait veriler Şekil 5.15'de 'Detaylar' penceresinde görülmektedir.



Şekil 5.15. Ana profile ve ofset doğrusuna ait veriler

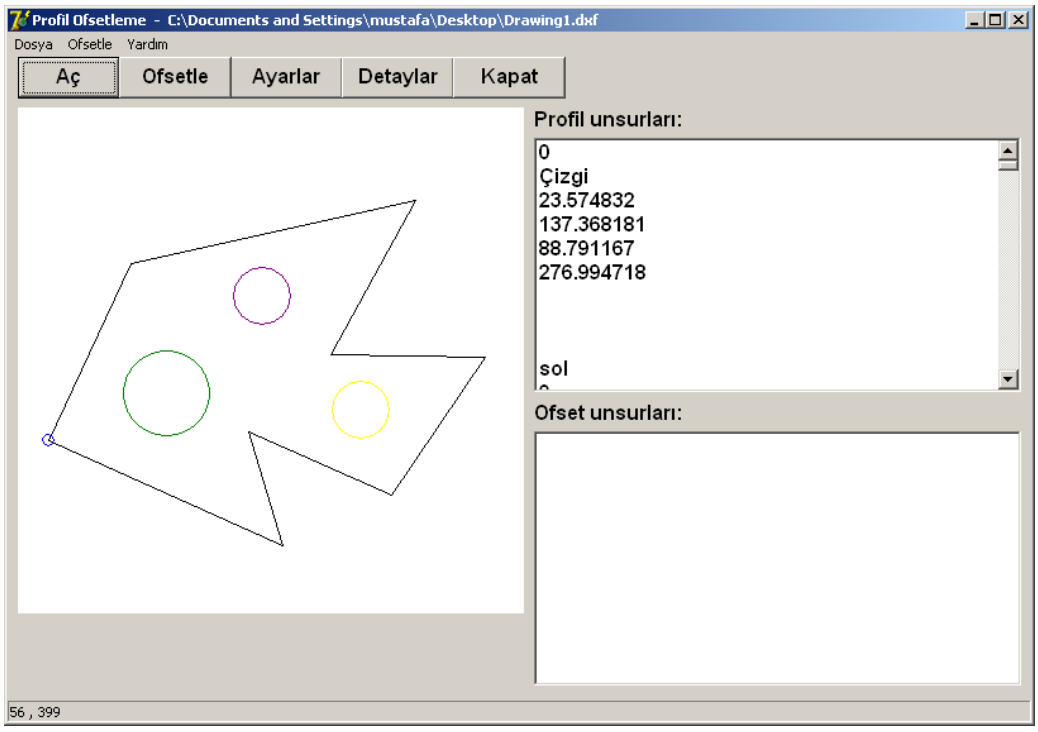
5.2.1. Örnek uygulama 2

Programın çalıştırılmasımasından sonra 'Dosya' ve 'Aç' komutları sırası ile çalıştırılarak 'ornek1.dxf' adlı DFX dosyası program tarafından açılmıştır (Şekil 5.16).



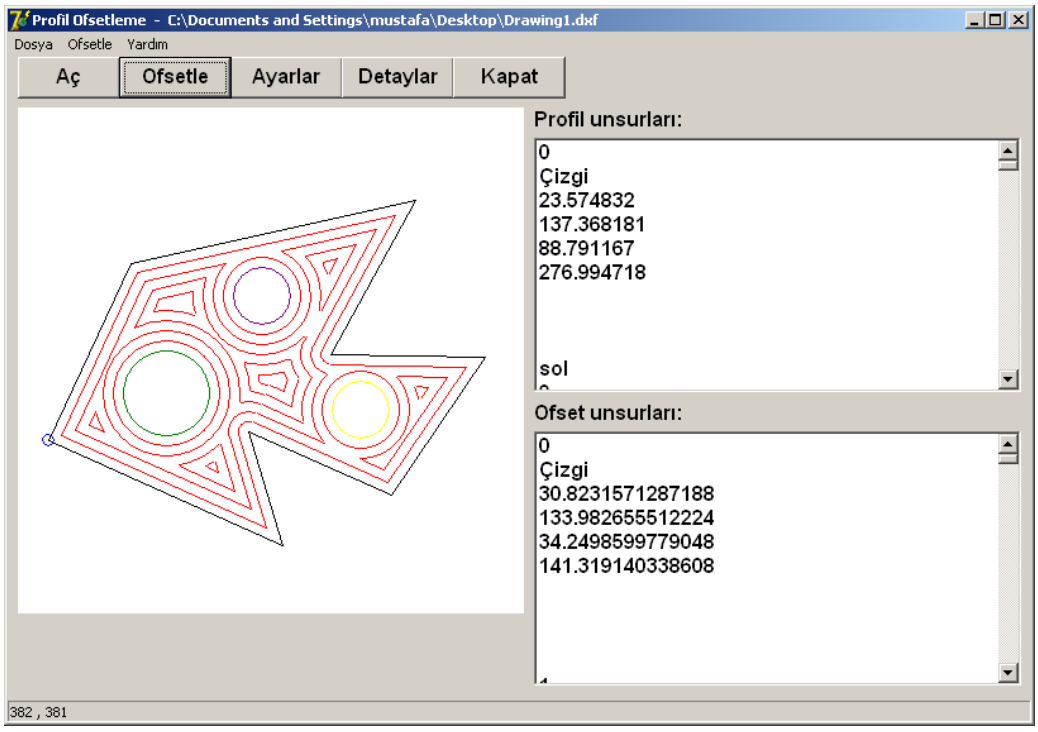
Şekil 5.16. 'ornek1.dxf' dosyasının açılması

'ornek1.dxf ' dosyasının program tarafından açılarak okunmasıyla dosya içinde bulunan unsur verileri alınmış ve ofsetlenecek cep profili programın ekranına çizdirilmiştir (Şekil 5.17).



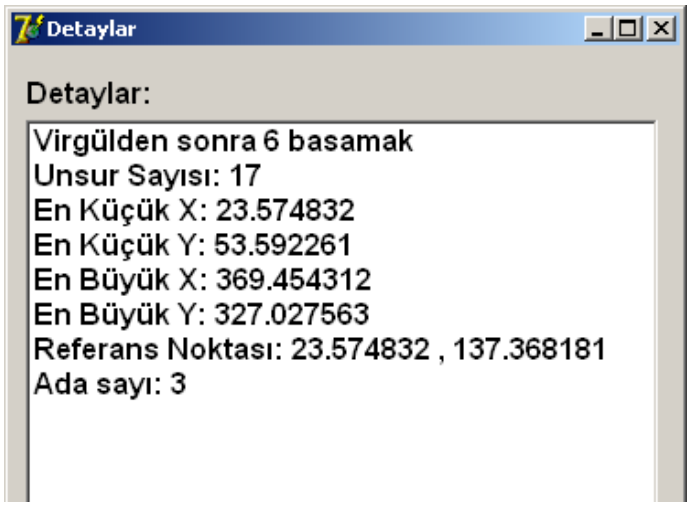
Şekil 5.17. 'ornek2.dxf' dosyasındaki profilin ekrana çizdirilmesi

Daha sonra veri kutusuna '8mm'lik ofsetleme mesafesi girilerek 'ofsetle' düğmesine basılmış ve ofsetleme işlemi gerçekleştirilmiştir. Ofsetleme işlemi 8 mm, 16 mm, 24 mm, 32 mm, 40 mm'lik ofsetleme mesafelerinde 5 defa tekrarlanmıştır (Şekil 5.18).



Şekil 5.18. Ofsetleme işleminin gerçekleştirilmesi

Elde edilen ofset doğrularına ait veriler Şekil 5.19'de 'Detaylar' penceresinde görülmektedir.



Şekil 5.19. Ana profile ve ofset doğrusuna ait veriler

6. SONUÇ VE ÖNERİLER

6.1. Sonuç

CNC takım yollarının türetilmesinde ofsetleme işleminin oynadığı rol göz önüne alındığında ofsetlerdeki geçersiz unsurların işleme esnasında dalmalara ve istenmeyen kalıntılara neden olduğu bilinmektedir. Bu çalışmada sunulan yöntem de literatürdeki diğer çalışmalarda olduğu gibi ofsetleme işlemini hatasız olarak gerçekleştirmektedir. Ancak, geliştirilen hata ayıklama yöntemi ofsetleme problemine yeni bir bakış açısı getirmiştir. Bu çalışmada ofsetleme hatalarını giderilmesi matematiksel denklemlerden farklı olarak daha pratik bir yöntemle gerçekleştirilmiştir. Geliştirilen bu yeni yöntem literatürdeki diğer yöntemlere nazaran daha pratik ve uygulanabilir bir modeldir [26].

Yapılan bu çalışma ile karmaşık geometriye sahip cep profilleri doğru bir şekilde ofsetlenmiştir. Ofsetleme sonuçları geliştirilen program ekranında çizilerek doğrudan izlenebilmektedir. Ayrıca ofset unsurlarına ait koordinat verileri liste halinde görülebilmektedir.

Bu çalışmada literatürdeki diğer çalışmalardan farklı olarak parametrik denklemler ya da vektörler yerine analitik denklemler kullanılmıştır. Literatürde sunulan bazı çalışmalarda [2] yay içeren profiller göz ardı edilirken gerçekleştirilen bu çalışmada sunulan yöntem, yay içeren profilleri de ofsetleyebilmektedir.

Voronoi eğrileri kullanılarak gerçekleştirilen birçok uygulamalarda ofsetleme işlemi dışa doğru yapılamamaktadır [6]. Bir çok uygulamada dışa doğru ofsetleme yapılamaması, içerisinde ada bulunan ceplerin işlenmesinde önemli bir eksiklik olarak görülmektedir. Sunduğumuz çalışma ise içe ve dışa doğru ofsetlemeyi gerçekleştirebilmektedir. Voronoi eğrilerini kullanan yöntemler de dahil olmak üzere literatürde önerilen bir çok yöntem profili oluşturan unsurların parametrik denklemleri olan eğrilere ihtiyaç duymaktadır. İşlenecek bir cep profili

oluşturulurken parametrik eğriler yerine daha kolay olduğu için doğru ve yay gibi unsurlardan oluştuğu dikkate alınarak çalışma gerçekleştirilmiştir.

Bu çalışmada sunulan yöntemde, literatürde sunulan bazı metodlarda olduğu gibi kapalı profili oluşturan unsurların "polyline" ya da sıralanmış olarak üretilmesine ihtiyaç duyulmamaktadır [8]. Bu çalışmada sunulan yöntemde kapalı profilin sıralı olup olmaması yada 'polyline' olarak oluşturulup oluşturulmamasına bağlı değildir. Unsurlar oluşturulduktan sonra sıralamasını geliştirilen program yapmaktadır.

Rohmfeld'in önerdiği yöntem ofsetleme ile oluşan teğet noktalarında yanılgıya düşebilmektedir [14]. Burada sunulan yöntemde ki hata giderme algoritması teğet noktalarında hataları başarıyla giderebilmektedir.

Çizelge 6.1. Geliştirilen programın karşılaştırılması

	Profil	***20mm ofset***	***35mm ofset***	***50mm ofset***
AutoCAD 2009			 Sonuç yok	 Sonuç yok
SolidWorks 2009		 Sonuç yok	 Sonuç yok	 Sonuç yok
MasterCam X2				
Delphi'de geliştirilen program				

Yapılan bu çalışmada önerilen ofsetleme yöntemi kullanılarak, Delphi ortamında geliştirilmiş program ile endüstride yaygın olarak kullanılan CAD/CAM programları karşılaştırılmıştır (Çizelge 6.1). Buna göre bazı yazılımlar örnek profil için belirli ofsetleme mesafelerinde ofsetleme sonucu üretebilirken, bazıları da hiçbir sonuç vermemektedir. Bu tez kapsamında geliştirilen program da başarılı bir ofset sonucu üretebilen programlar gibi doğru bir sonuç vermektedir.

6.2. Öneriler

Yapılan bu çalışmanın devamı olarak elde edilen ofset unsurları cebi işleyecek takım yollarını türetmek için kullanılabilir. Geliştirilecek yeni bir yazılım ile doğru ve yaylardan oluşan ofset grupları bir birleri ile ilişkilendirilip, her bir unsur için uygun takım hareketlerini gerçekleştirecek CNC kodları, çok hızlı ve hatasız bir şekilde türetilebilir.

Ek olarak parametrik eğrilerden oluşan cep profillerinin ofsetlenmesi için de eğrilerin kendi denklemleri kullanılarak hata ayıklama işlemi uygulanabilir. Böylece eğrilerden oluşan cepler için de takım yolları türetilebilir.

Ayrıca ofsetleme için sunulan bu yöntem kullanılarak cep profillerini işlemek için tek yönlü doğrusal, çift yönlü doğrusal ve karma takım yolları da türetilebilir.

Bu çalışmada ofsetleme işleminin gerçekleştirilmesi için başlama noktası olarak referans noktası tercih edilmiştir. Elde edilen ofset unsurlarından takım yolları türetilirken kesici için başlama noktası belirlenirken kesicinin parçaya rampa yaparak dalmasını kolaylaştırmak için daha büyük olan doğru veya yaylar tercih edilebilir.

Üretim süresini ve toplam takım yolu uzunluğunu kısaltmak için işleme esnasında oluşacak kesici yarıçapından küçük adacıklar için düz doğru şeklinde takım yolları eklenebilir.

KAYNAKLAR

1. Pinar, A.M., CNC Frezelemede Alternatif Başlama Noktaları ve İşlem Basamaklarına Göre Zaman Analizi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi***, Ankara, (2000)

2. You, C.F., Sheen, B.T., Lin, T., "Robust Spiral Tool-Path Generation for Arbitrary Pockets", ***International Journal of Advanced Manufacturing Technology***, 17:181–188, (2001)

3. Lai, Y., Wu, J., Hung, J., Chen, J., "A Simple Method for Invalid Loops Removal of Planar Offset Curves", ***International Journal of Advanced Manufacturing Technology***, 27:1153–1162, (2006)

4. Rhomfeld R.F., "IGB-offset for Plane Curves-loop Removal by Scanning of Interval Sequences", ***Computer Aided Geometric Design,*** 15:339-375, (1997)

5. Jeong, J., Kim, K., "Generating Tool Paths for Free-Form Pocket Machining Using Z-Buffer-Based Voronoi Diagrams", ***International Journal of Advanced Manufacturing Technology,*** 15:182–187, (1999)

6. Held, M., "Voronoi Diagrams and Offset Curves of Curvilinear Polygons ", ***Computer-Aided Design***, 30(4):287-300, (1998)

7. Jeong, J., Kim, K., "Generation of Tool Paths for Machining Free-Form Pockets with Islands Using Distance Maps", ***International Journal of Advanced Manufacturing Technology***, 15:311–316, (1999)

8. Choi, B., Park, S., "A Pair-Wise Offset Algorithm for 2D Point-Sequence Curve", ***Computer-Aided Design,*** 31:735–745, (1999)

9. Liu, X., Yong, J., Zheng, G., Sun, J., "An Offset Algorithm for Polyline Curves", ***Computers in Industry,*** 58:240–254, (2007)

10. Choy, H.S., Chan, K.W., "A Corner-Looping Based Tool Path for Pocket Milling", ***Computer-Aided Design,*** 35:155–166, (2003)

11. Kim, H.C., Lee. S.G., Yang, M.Y., "An Optimized Contour Parallel Tool Path for 2D Milling with Flat Endmill", ***International Journal of Advanced Manufacturing Technology***, 31:567-573, (2006)

12. Lee, E., "Counter Ofset Approch to Spiral Toolpath Generation with Constant Scallop Height", ***Computer-Aided Design,*** 35:511-518, (2003)

13. Park, S.C., Chung, Y.C., Byoung, K.C., "Contour-Parallel Offset Machining Without Tool-Retractions", ***Computer-Aided Design,*** 35:841-849, (2003)

14. Yao, Z., Gupta, S.K., "Cutter Path Generation for 2.5D Milling By Combining Multiple Different Cutter Path Patterns", ***International Journal of Production Research***, 42(11):2141–2161, (2004)

15. Zhao, Z.Y., Wang, C.Y., Zhou, H.M., Qin, Z., "Pocketing Toolpath Optimization for Sharp Corners", ***Journals of Materials Processing Technology***, 192-193:175-180, (2007)

16. Sheen, B.T., You, C.F., "Tool Path Generation For Arbitrary Pockets With Islands", ***Journal of Intelligent Manufacturing,*** 17:275–283, (2006)

17. Chuang S.H., Lin, W.S., "Tool-Path Generation for Pockets with Freeform Curves Using Bezier Convex Hulls", ***The International Journal of Advanced Manufacturing Technology***, 13:109-115, (1997)

18. Yan, S., Shuilai, W., Shuiguang, T., "Uneven Offset Method of NC Tool Path Generation for Free-Form Pocket Machining", ***Computers in Industry,*** 43:97-103, (2000)

19. Lambregts, C.A.H., "Efficient Automatic Toolpath Generator for 2[½]D Free Form Pockets", ***International Journal of Production Research,*** 33(6):1683-1697, (1995)

20. Wong, T.N., Wong, K.W., "NC Tollpath Generation for Abitrary Pockets with Islands", ***The International Journal of Advanced Manufacturing Technology***, 12:174-179, (1996)

21. Tekiner, Z., Korkut, İ., "Iges Veri Yapısı Kullanılarak Dönel Parçalarda Süreç Planlama", ***Makina Tasarım ve İmalat Dergisi***, 3(5):232-241, (2000)

22. Bieterman, M.B., Sandstrom, D.R., "A Curvilinear Tool-Path Method For Pocket Machining", ***Journal of Manufacturing Science and Engineering***, 125:709-715, (2003)

23. Yıldız, Y., "Dönel Parçalar İçin Unsur Tabanlı Bir Cam Programı Hazırlanması ve Diğer Programlama Yöntemleri ile Karşılaştırılması", ***Gazi Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi***, Ankara, (2004)

24. Yang, Z., Joneja, A., Zhu, S., "Recognizing Generalized Pockets for Optimizing Machining Time in Process Planning – Part 1", ***International Journal of Production Research***, 39(15):3377-3397, (2001)

25. Yang, Z., Joneja, A., Zhu, S., "Recognizing Generalized Pockets for Optimizing Machining Time in Process Planning – Part 2", ***International Journal of Production Research***, 39(16):3601-3621, (2001)

26. Göktaş, M., Dilipak, H., Güldaş, A., "Ofsetleme ve Cep Frezeleme İşlemlerinde Analitik Yaklaşım", ***5th International Advanced Technologies Symposium,*** *Karabük,* (2009)

27. Göktaş, M., Dilipak, H., Güldaş, A., "İki Boyutlu Profillerin İşlenmesinde Takım Yolu ve Ofsetleme için Yeni Bir Algoritma Geliştirilmesi", ***Gazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi,*** 25(1):179-187, (2010)

28. Pateloup, V., Duc, E., Ray, P., "Bspline Approximation of Circle Arc and Straight Line for Pocket Machining", ***Computer-Aided Design***, S0010-4485(10):00084-9, (2010)

29. Held, M., Spielberger, C., "A Smooth Spiral Tool Path for High Speed Machining of 2D Pockets", ***Computer-Aided Design***, 41:539-550, (2009)

30. İpekçioğlu, N., "Frezecilik", ***Milli Eğitim Basımevi***, İstanbul, (1984)

31. Gülesin, M., Güllü, A., Avcı, Ö., Akdoğan, G., "CNC Torna ve Freze Tezgahlarının Programlanması", ***Asil Yayın Dağıtım***, Ankara, (2005)

32. Gülesin, M., Güllü, A., Buldum, B.B., "Makine Teknolojileri için Birimler Formüller, Çizelgeler", ***Seçkin Yayıncılık***, Ankara, (2003)

33. Gökkaya, H., "Otomat Torna Tezgahlarında Bilgisayar Destekli İşlem Planlaması", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü*****,** Ankara, (1998)

34. Held, M., "A Geometry-Based Investigation of The Tool Path Generation for Zig-Zag Pocket Machining", ***Visual Computer***, 7:296-308, (1991)

35. Puttre, M., "Computer Aided Manufacturing-Sculpturing Parts from Stored Patterns", ***Mechanical Engineering***, 114:66-70, (1992)

36. Lallande JB., Walc, A., "A Super Pocket, Numerical Control Society", ***Annual Meeting and Technical Conf.***, 18-29, (1984)

37. Dong, J., Vijan, S. "Manufacturing Feature Determination and Extraction-Part 1: Optimal Volume Segmentation", ***CAD***, 29:493-496, (1997)

38. Joo, J., Cho, H., Yun, W., "Efficient Feature-Based Process Planning for Sculptured Pocket Machining", ***Computer Industrial Engineering***, 33:493-496, (1997)

39. Wang, H., Chang, H., RA., Chandawarkar, A. "On The Efficiency of NC Tool Path Planning for Face Milling Operation", ***J. Eng. Ind. Trans. ASME***, 109:370-376, (1987)

40. Southwood, JS., Chang, TC., “Asingle Feture Approach to Automatic NC Code Generation for 2.5 Dimensional Prismatic Features”, ***J. Design. Manuf.***, 4:115-127, (1994)

41. Dong, Z., Li, H., Vickers, GW., “Optimal Rough Machining of Curved Surfaces”, ***Proc. CSME Forum Transport***, 2:593-597, (1993)

EKLER

EK-1. ekx, eky ve raferans noktalarını bulduran program kodları

```
…
begin
  ekx:=0;eky:=0;
  ebx:=0;eby:=0;
  u:=false;
  for i:=1 to adet do
   begin
     if i=1 then
      begin
        if hamunsurlar[0,1]='Çizgi' then
         begin
           x:=strtofloat(hamunsurlar[0,2]);
           y:=strtofloat(hamunsurlar[0,3]);
           ekx:=x;eky:=y;
           ebx:=x;eby:=y;
           x:=strtofloat(hamunsurlar[0,4]);
           y:=strtofloat(hamunsurlar[0,5]);
           if x<ekx then ekx:=x;
           if y<eky then eky:=y;
           if x>ebx then ebx:=x;
           if y>eby then eby:=y;
         end;
       if hamunsurlar[0,1]='Yay' then
         begin
           basaci:=acibul(strtofloat(hamunsurlar[0,6]),strtofloat(hamunsurlar[0,7]),
           strtofloat(hamunsurlar[0,2]),strtofloat(hamunsurlar[0,3]));
           bitaci:=acibul(strtofloat(hamunsurlar[0,6]),strtofloat(hamunsurlar[0,7]),
           strtofloat(hamunsurlar[0,4]),strtofloat(hamunsurlar[0,5]));
           if (basaci<bitaci) then
             begin
               if (basaci<=180) and (bitaci>=180) then
                 begin
                   x:=strtofloat(hamunsurlar[0,6])-strtofloat(hamunsurlar[0,8]);
                 end
               else
                 begin
                   x:=strtofloat(hamunsurlar[0,2]);
                 end;
               if (basaci<=270) and (bitaci>=270) then
                 begin
                   y:=strtofloat(hamunsurlar[0,7])-strtofloat(hamunsurlar[0,8]);
                 end
               else
                 begin
                   y:=strtofloat(hamunsurlar[0,3]);
                 end;
             end;
           if (bitaci<basaci) then
             begin
               if ((basaci<=180) and (bitaci<=180)) or ((basaci>=180) and
               (bitaci>=180)) then
                 begin
                   x:=strtofloat(hamunsurlar[0,6])-strtofloat(hamunsurlar[0,8]);
                 end
```

EK-1 (Devam). ekx, eky ve raferans noktalarını bulduran program kodları

```
  else
    begin
      x:=strtofloat(hamunsurlar[0,2]);
    end;
   if ((basaci<=270) and (bitaci<=270)) or ((basaci>=270) and
   (bitaci>=270)) then
    begin
      y:=strtofloat(hamunsurlar[0,7])-strtofloat(hamunsurlar[0,8]);
    end
   else
    begin
      y:=strtofloat(hamunsurlar[0,3]);
    end;
  end;
ekx:=x;eky:=y;
ebx:=x;eby:=y;
if (basaci<bitaci) then
  begin
   if (basaci<=180) and (bitaci>=180) then
    begin
      x:=strtofloat(hamunsurlar[0,6])-strtofloat(hamunsurlar[0,8]);
    end
   else
    begin
      x:=strtofloat(hamunsurlar[0,2]);
    end;
   if (basaci<=270) and (bitaci>=270) then
    begin
      y:=strtofloat(hamunsurlar[0,7])-strtofloat(hamunsurlar[0,8]);
    end
   else
    begin
      y:=strtofloat(hamunsurlar[0,3]);
    end;
  end;
if (bitaci<basaci) then
  begin
   if ((basaci<=180) and (bitaci<=180)) or ((basaci>=180) and
   (bitaci>=180)) then
    begin
      x:=strtofloat(hamunsurlar[0,6])-strtofloat(hamunsurlar[0,8]);
    end
   else
    begin
      x:=strtofloat(hamunsurlar[0,2]);
    end;
   if ((basaci<=270) and (bitaci<=270)) or ((basaci>=270) and
   (bitaci>=270)) then
    begin
      y:=strtofloat(hamunsurlar[0,7])-strtofloat(hamunsurlar[0,8]);
    end
   else
    begin
      y:=strtofloat(hamunsurlar[0,3]);
```

EK-1 (Devam). ekx, eky ve raferans noktalarını bulduran program kodları

```
            end;
          end;
        if x<ekx then ekx:=x;
        if y<eky then eky:=y;
        if x>ebx then ebx:=x;
        if y>eby then eby:=y;
      end;
  end;
 if i<>1 then
  begin
    if hamunsurlar[i-1,1]='Çizgi' then
     begin
       x:=strtofloat(hamunsurlar[i-1,2]);
       y:=strtofloat(hamunsurlar[i-1,3]);
       if x<ekx then ekx:=x;
       if y<eky then eky:=y;
       if x>ebx then ebx:=x;
       if y>eby then eby:=y;
       x:=strtofloat(hamunsurlar[i-1,4]);
       y:=strtofloat(hamunsurlar[i-1,5]);
       if x<ekx then ekx:=x;
       if y<eky then eky:=y;
       if x>ebx then ebx:=x;
       if y>eby then eby:=y;
     end;
    if hamunsurlar[i-1,1]='Yay' then
     begin
       basaci:=acibul(strtofloat(hamunsurlar[i-1,6]),strtofloat(hamunsurlar
       [i-1,7]),strtofloat(hamunsurlar[i-1,2]),strtofloat(hamunsurlar[i-1,3]));
       bitaci:=acibul(strtofloat(hamunsurlar[i-1,6]),strtofloat(hamunsurlar
       [i-1,7]),strtofloat(hamunsurlar[i-1,4]),strtofloat(hamunsurlar[i-1,5]));
       if (basaci<bitaci) then
         begin
          if (basaci<=180) and (bitaci>=180) then
            begin
              x:=strtofloat(hamunsurlar[i-1,6])-strtofloat(hamunsurlar[i-1,8]);
            end
          else
            begin
              x:=strtofloat(hamunsurlar[i-1,2]);
            end;
          if (basaci<=270) and (bitaci>=270) then
            begin
              y:=strtofloat(hamunsurlar[i-1,7])-strtofloat(hamunsurlar[i-1,8]);
            end
          else
            begin
              y:=strtofloat(hamunsurlar[i-1,3]);
            end;
         end;
       if (bitaci<basaci) then
         begin
          if ((basaci<=180) and (bitaci<=180)) or ((basaci>=180) and (bitaci>=180)) then
            begin
```

EK-1 (Devam). ekx, eky ve raferans noktalarını bulduran program kodları

```
          x:=strtofloat(hamunsurlar[i-1,6])-strtofloat(hamunsurlar[i-1,8]);
        end
      else
        begin
          x:=strtofloat(hamunsurlar[i-1,2]);
        end;
      if ((basaci<=270) and (bitaci<=270)) or ((basaci>=270) and
      (bitaci>=270)) then
        begin
          y:=strtofloat(hamunsurlar[i-1,7])-strtofloat(hamunsurlar[i-1,8]);
        end
      else
        begin
          y:=strtofloat(hamunsurlar[i-1,3]);
        end;
    end;
  if x<ekx then ekx:=x;
  if y<eky then eky:=y;
  if x>ebx then ebx:=x;
  if y>eby then eby:=y;
  if (basaci<bitaci) then
    begin
      if (basaci<=180) and (bitaci>=180) then
        begin
          x:=strtofloat(hamunsurlar[i-1,6])-strtofloat(hamunsurlar[i-1,8]);
        end
      else
        begin
          x:=strtofloat(hamunsurlar[i-1,2]);
        end;
      if (basaci<=270) and (bitaci>=270) then
        begin
          y:=strtofloat(hamunsurlar[i-1,7])-strtofloat(hamunsurlar[i-1,8]);
        end
      else
        begin
          y:=strtofloat(hamunsurlar[i-1,3]);
        end;
    end;
  if (bitaci<basaci) then
    begin
      if ((basaci<=180) and (bitaci<=180)) or ((basaci>=180) and
      (bitaci>=180)) then
        begin
          x:=strtofloat(hamunsurlar[i-1,6])-strtofloat(hamunsurlar[i-1,8]);
        end
      else
        begin
          x:=strtofloat(hamunsurlar[i-1,2]);
        end;
      if ((basaci<=270) and (bitaci<=270)) or ((basaci>=270) and
      (bitaci>=270)) then
        begin
          y:=strtofloat(hamunsurlar[i-1,7])-strtofloat(hamunsurlar[i-1,8]);
```

EK-1 (Devam). ekx, eky ve raferans noktalarını bulduran program kodları

```
              end
             else
               begin
                 y:=strtofloat(hamunsurlar[i-1,3]);
               end;
            end;
          if x<ekx then ekx:=x;
          if y<eky then eky:=y;
          if x>ebx then ebx:=x;
          if y>eby then eby:=y;
        end;
    end;
end;
for j:=1 to adet do
 begin
   if j=1 then
    begin
      if hamunsurlar[0,1]='Çizgi' then
       begin
         x:=strtofloat(hamunsurlar[0,2]);
         y:=strtofloat(hamunsurlar[0,3]);
         hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
         ref:=hip;
         refx:=x;
         refy:=y;
         x:=strtofloat(hamunsurlar[0,4]);
         y:=strtofloat(hamunsurlar[0,5]);
         hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
         if hip<ref then
          begin
            ref:=hip;
            refx:=x;
            refy:=y;
          end;
       end;
      if hamunsurlar[0,1]='Yay' then
       begin
         x:=strtofloat(hamunsurlar[0,2]);
         y:=strtofloat(hamunsurlar[0,3]);
         hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
         ref:=hip;
         refx:=x;
         refy:=y;
         x:=strtofloat(hamunsurlar[0,4]);
         y:=strtofloat(hamunsurlar[0,5]);
         hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
         if hip<ref then
          begin
            ref:=hip;
            refx:=x;
            refy:=y;
          end;
       end;
    end;
```

EK-1 (Devam). ekx, eky ve raferans noktalarını bulduran program kodları

```
    if j< >1 then
     begin
       if hamunsurlar[j-1,1]='Çizgi' then
        begin
          x:=strtofloat(hamunsurlar[j-1,2]);
          y:=strtofloat(hamunsurlar[j-1,3]);
          hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
          if hip<ref then
           begin
             ref:=hip;
             refx:=x;
             refy:=y;
           end;
          x:=strtofloat(hamunsurlar[j-1,4]);
          y:=strtofloat(hamunsurlar[j-1,5]);
          hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
          if hip<ref then
           begin
             ref:=hip;
             refx:=x;
             refy:=y;
           end;
        end;
       if hamunsurlar[j-1,1]='Yay' then
        begin
          x:=strtofloat(hamunsurlar[j-1,2]);
          y:=strtofloat(hamunsurlar[j-1,3]);
          hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
          if hip<ref then
           begin
             ref:=hip;
             refx:=x;
             refy:=y;
           end;
          x:=strtofloat(hamunsurlar[j-1,4]);
          y:=strtofloat(hamunsurlar[j-1,5]);
          hip:=sqrt((x-ekx)*(x-ekx)+(y-eky)*(y-eky));
          if hip<ref then
           begin
             ref:=hip;
             refx:=x;
             refy:=y;
           end;
        end;
     end;
  end;
end;
```

EK-2. Sıralama yönünü bulduran program kodları

```
procedure yonbul();
var
  gidenaci,gelenaci,mx,my,r,basaci,bitaci,tamaci,oraci,orx,ory,x1,y1,x2,y2,aci1,aci2:real;
  i,j,n,t:integer;
begin
 setlength(kapalanlar,kaladet,4);
 for i:=1 to kaladet do
  begin
   kapalanlar[i-1,1]:='0';
  end;
 i:=1;
 while i<>adet+1 do
  begin
   kapalanlar[strtoint(sirunsurlar[i-1,10]),1]:=
   inttostr(strtoint(kapalanlar[strtoint(sirunsurlar[i-1,10]),1])+1);
   i:=i+1;
  end;
 for i:=1 to kaladet do
  begin
   n:=0;
   t:=0;
   for j:=1 to i do
    begin
     t:=strtoint(kapalanlar[j-1,1])+t;
     n:=t-strtoint(kapalanlar[j-1,1])+1;
    end;
   if sirunsurlar[n-1,1]='Çizgi' then
    begin
     gidenaci:=acibul(ekx,eky,
     strtofloat(sirunsurlar[n-1,4]),strtofloat(sirunsurlar[n-1,5]));
    end;
   if sirunsurlar[t-1,1]='Çizgi' then
    begin
     gelenaci:=acibul(ekx,eky,
     strtofloat(sirunsurlar[t-1,2]),strtofloat(sirunsurlar[t-1,3]));
    end;
   if sirunsurlar[n-1,1]='Yay' then
    begin
     x1:=strtofloat(sirunsurlar[n-1,2]);
     y1:=strtofloat(sirunsurlar[n-1,3]);
     x2:=strtofloat(sirunsurlar[n-1,4]);
     y2:=strtofloat(sirunsurlar[n-1,5]);
     mx:=strtofloat(sirunsurlar[n-1,6]);
     my:=strtofloat(sirunsurlar[n-1,7]);
     r:=strtofloat(sirunsurlar[n-1,8]);
     if sirunsurlar[n-1,9]='sag' then
      begin
       basaci:=acibul(mx,my,x1,y1);
       bitaci:=acibul(mx,my,x2,y2);
      end;
     if sirunsurlar[n-1,9]='sol' then
      begin
       basaci:=acibul(mx,my,x2,y2);
```

EK-2 (Devam). Sıralama yönünü bulduran program kodları

```
          bitaci:=acibul(mx,my,x1,y1);
         end;
       if basaci<bitaci then
         begin
          tamaci:=bitaci-basaci;
          oraci:=basaci+tamaci/2;
         end;
       if basaci>bitaci then
         begin
          tamaci:=bitaci+360-basaci;
          oraci:=basaci+tamaci/2;
         end;
       orx:=mx+r*cos(oraci*pi/180);
       ory:=my+r*sin(oraci*pi/180);
       gidenaci:=acibul(ekx,eky,orx,ory);
      end;
    if sirunsurlar[t-1,1]='Yay' then
     begin
      x1:=strtofloat(sirunsurlar[t-1,2]);
      y1:=strtofloat(sirunsurlar[t-1,3]);
      x2:=strtofloat(sirunsurlar[t-1,4]);
      y2:=strtofloat(sirunsurlar[t-1,5]);
      mx:=strtofloat(sirunsurlar[t-1,6]);
      my:=strtofloat(sirunsurlar[t-1,7]);
      r:=strtofloat(sirunsurlar[t-1,8]);
      if sirunsurlar[t-1,9]='sag' then
        begin
         basaci:=acibul(mx,my,x1,y1);
         bitaci:=acibul(mx,my,x2,y2);
        end;
      if sirunsurlar[t-1,9]='sol' then
        begin
         basaci:=acibul(mx,my,x2,y2);
         bitaci:=acibul(mx,my,x1,y1);
        end;
      if basaci<bitaci then     // yay tek parça ise...
        begin
         tamaci:=bitaci-basaci;
         oraci:=basaci+tamaci/2;
        end;
      if basaci>bitaci then      // yay iki parça ise...
        begin
         tamaci:=bitaci+360-basaci;
         oraci:=basaci+tamaci/2;
        end;
      orx:=mx+r*cos(oraci*pi/180);
      ory:=my+r*sin(oraci*pi/180);
      gelenaci:=acibul(ekx,eky,orx,ory);
     end;
    if gelenaci<gidenaci then kapalanlar[i-1,2]:='sy';
    if gelenaci>gidenaci then kapalanlar[i-1,2]:='sty';
  end;
```

EK-3. İki doğrunun kesişimini bulduran program kodları

```
begin
  aci1:=acibul(x1,y1,x2,y2);
  aci2:=acibul(x3,y3,x4,y4);
  if (abs(aci1-aci2)=0) or (abs(aci1-aci2)=180) then
    begin
      if (aci1<>0) and  (aci1<>90) and  (aci1<>180) and  (aci1<>270) then
        begin
          m1:=(y2-y1)/(x2-x1);
          m2:=(y4-y3)/(x4-x3);
          if round((m1*(x3-x1)+y1))=y3 then dogrultu:='3';
          if round((m1*(x3-x1)+y1))<>y3 then dogrultu:='0';
        end;
      if (aci1=90) or (aci1=270) then
        begin
          if x1<>x3 then dogrultu:='0';
          if x1=x3 then dogrultu:='3';
        end;
      if (aci1=0) or (aci1=180) then
        begin
          if y1<>y3 then dogrultu:='0';
          if y1=y3 then dogrultu:='3';
        end;
      if dogrultu='3' then
        begin
          if
          ((roundto(x1,-3)<roundto(x3,-3)) and (roundto(x1,-3)<roundto(x4,-3)) and
          (roundto(x2,-3)<roundto(x3,-3)) and (roundto(x2,-3)<roundto(x4,-3)))
          or
          ((roundto(x1,-3)>roundto(x3,-3)) and (roundto(x1,-3)>roundto(x4,-3)) and
          (roundto(x2,-3)>roundto(x3,-3)) and (roundto(x2,-3)>roundto(x4,-3)))
          or
          ((roundto(y1,-3)<roundto(y3,-3)) and (roundto(y1,-3)<roundto(y4,-3)) and
          (roundto(y2,-3)<roundto(y3,-3)) and (roundto(y2,-3)<roundto(y4,-3)))
          or
          ((roundto(y1,-3)>roundto(y3,-3)) and (roundto(y1,-3)>roundto(y4,-3)) and
          (roundto(y2,-3)>roundto(y3,-3)) and (roundto(y2,-3)>roundto(y4,-3)))
          then
            begin
              araliktami:='0';
            end
          else
            begin
              araliktami:='1';
            end
        end;
    end;
if (abs(aci1-aci2)<>0) and (abs(aci1-aci2)<>180) and (abs(aci1-aci2)<>360) then
    begin
      if (aci1=0) or (aci1=180) then
        begin
          y5:=y1;
          if (aci2=90) or (aci2=270) then
            begin
```

EK-3 (Devam). İki doğrunun kesişimini bulduran program kodları

```
          x5:=x3;
         end;
       if (aci2<>0) and (aci2<>90) and (aci2<>180) and (aci2<>270) then
         begin
           x5:=(y5-y3)*((x4-x3)/(y4-y3))+x3;
         end;
      end;
    if (aci1=90) or (aci1=270) then
      begin
       x5:=x1;
       if (aci2=0) or (aci2=180) then
         begin
           y5:=y3;
         end;
       if (aci2<>0) and (aci2<>90) and (aci2<>180) and (aci2<>270) then
         begin
           y5:=((y4-y3)/(x4-x3))*(x5-x3)+y3;
         end;
      end;
if (aci1<>0) and (aci1<>90) and (aci1<>180) and (aci1<>270) then
      begin
       if (aci2=0) or (aci2=180) then
         begin
           y5:=y3;
           x5:=(y5-y1)*((x2-x1)/(y2-y1))+x1;
         end;
       if (aci2=90) or (aci2=270) then
         begin
           x5:=x3;
           y5:=((y2-y1)/(x2-x1))*(x5-x1)+y1;
         end;
       if (aci2<>0) and (aci2<>90) and (aci2<>180) and (aci2<>270) then
         begin
           m1:=(y2-y1)/(x2-x1);
           m2:=(y4-y3)/(x4-x3);
           if m1<>m2 then
             begin
               x5:=(m1*x1-m2*x3+y3-y1)/(m1-m2);
               y5:=m1*(x5-x1)+y1;
             end;
         end;
      end;
     if (round(x5)<round(x1)) and (round(x5)<round(x2))
    or (round(x5)>round(x1)) and (round(x5)>round(x2))
    or (round(x5)<round(x3)) and (round(x5)<round(x4))
    or (round(x5)>round(x3)) and (round(x5)>round(x4))
    or (round(y5)<round(y1)) and (round(y5)<round(y2))
    or (round(y5)>round(y1)) and (round(y5)>round(y2))
    or (round(y5)<round(y3)) and (round(y5)<round(y4))
    or (round(y5)>round(y3)) and (round(y5)>round(y4))
    then
      begin
       dogrultu:='1';
```

EK-3 (Devam). İki doğrunun kesişimini bulduran program kodları

```
        araliktami:='0';
       end
     else
       begin
         dogrultu:='1';
         araliktami:='1';
       end
   end;
 if (dogrultu='1') and (araliktami='1') then durum:='1';
 if (dogrultu<>'1') or (araliktami<>'1') then durum:='0';
end;
```

EK-4. Doğru ve yayın kesişimi

```
begin
  alfa:=acibul(x1,y1,x2,y2);
  h:=nokdoguz(x1,y1,x2,y2,mx,my);
  h:=roundto(h,-2);
  if h-r<0 then       //iki noktada kesişir...
    begin
      hx:=x5;
      hy:=y5;
      a:=sqrt(r*r-h*h);
      x5:=roundto(a*cos(alfa*pi/180)+hx,-tol);
      y5:=roundto(a*sin(alfa*pi/180)+hy,-tol);
      x6:=roundto(a*cos((alfa+180)*pi/180)+hx,-5);
      y6:=roundto(a*sin((alfa+180)*pi/180)+hy,-5);
      dogrultu:='2';
    end;
  if h-r<0.01 then       //tek noktada kesişir...
    begin
      hx:=x5;
      hy:=y5;
      a:=0;
      x5:=roundto(a*cos(alfa*pi/180)+hx,-tol);
      y5:=roundto(a*sin(alfa*pi/180)+hy,-tol);
      x6:=roundto(a*cos((alfa+180)*pi/180)+hx,-5);
      y6:=roundto(a*sin((alfa+180)*pi/180)+hy,-5);
      dogrultu:='1';
    end;
  if h-r>0 then dogrultu:='0';
  if (dogrultu='1') or (dogrultu='2') then
    begin
      if yon='sag' then
        begin
          bas:=roundto(basaci,-2);
          bit:=roundto(bitaci,-2);
        end;
      if yon='sol' then
        begin
          bas:=roundto(bitaci,-2);
          bit:=roundto(basaci,-2);
        end;
      aci1:=roundto(acibul(mx,my,x5,y5),-2);
      aci2:=roundto(acibul(mx,my,x6,y6),-2);
      if bas<bit then
        begin
          if ((aci1<bas) and (aci1<bit)) or ((aci1>bas) and (aci1>bit))  then
            begin
              d1:='0';
            end
          else
            begin
              d1:='1';
            end;
          if ((aci2<bas) and (aci2<bit)) or ((aci2>bas) and (aci2>bit))  then
            begin
```

EK-4 (Devam). Doğru ve yayın kesişimi

```
          d3:='0';
         end
       else
         begin
           d3:='1';
         end;
     end;
   if bas>bit then
     begin
      if ((aci1<bas) and (aci1>bit)) then
        begin
          d1:='0';
        end
      else
        begin
          d1:='1';
        end;
      if ((aci2<bas) and (aci2>bit)) then
        begin
          d3:='0';
        end
      else
        begin
          d3:='1';
        end;
     end;
   if (round(x5)<round(x1)) and (round(x5)<round(x2))
   or (round(x5)>round(x1)) and (round(x5)>round(x2))
   or (round(y5)<round(y1)) and (round(y5)<round(y2))
   or (round(y5)>round(y1)) and (round(y5)>round(y2))
   then
     begin
       d2:='0';
     end
   else
     begin
       d2:='1';
     end;
   if (round(x6)<round(x1)) and (round(x6)<round(x2))
   or (round(x6)>round(x1)) and (round(x6)>round(x2))
   or (round(y6)<round(y1)) and (round(y6)<round(y2))
   or (round(y6)>round(y1)) and (round(y6)>round(y2))
   then
     begin
       d4:='0';
     end
   else
     begin
       d4:='1';
     end;
  end;
durum:='0';
araliktami1:='0';
```

EK-4 (Devam). Doğru ve yayın kesişimi

```
 araliktami2:='0';
  if (d1='1') and (d2='1') then araliktami1:='1';
  if (d3='1') and (d4='1') then araliktami2:='1';
  if (dogrultu='1') and (araliktami1='1') then durum:='1';
  if (dogrultu='2') and (araliktami1='1') and (araliktami2='0') then durum:='20';
  if (dogrultu='2') and (araliktami1='0') and (araliktami2='1') then durum:='02';
  if (dogrultu='2') and (araliktami1='1') and (araliktami2='1') then durum:='22';
end;
```

EK-5. İki yayın kesişimini bulduran program kodları

```
begin
 msf:=roundto(sqrt((mx1-mx2)*(mx1-mx2)+(my1-my2)*(my1-my2)),-3);
 basaci1:=roundto(basaci1,-3);
 bitaci1:=roundto(bitaci1,-3);
 basaci2:=roundto(basaci2,-3);
 bitaci2:=roundto(bitaci2,-3);
 if (msf>roundto((r1+r2),-3)) or (msf<roundto(abs(r1-r2),-3)) then
  begin
   dogrultu:='0'
  end;
 if msf<roundto(r1+r2,-3) then
  begin
   alfa:=acibul(mx1,my1,mx2,my2);
   x5:=r1*cos((alfa)*pi/180)+mx1;
   y5:=r1*sin((alfa)*pi/180)+my1;
   x6:=r1*cos((alfa)*pi/180)+mx1;
   y6:=r1*sin((alfa)*pi/180)+my1;
   dogrultu:='1';
  end;
 if msf<roundto(abs(r1-r2),-3) then
  begin
   if r1<r2 then
    begin
     alfa:=acibul(mx1,my1,mx2,my2);
    end;
   if r1>r2 then
    begin
     alfa:=acibul(mx2,my2,mx1,my1);
    end;
   x5:=r1*cos((alfa+180)*pi/180)+mx1;
   y5:=r1*sin((alfa+180)*pi/180)+my1;
   x6:=r1*cos((alfa+180)*pi/180)+mx1;
   y6:=r1*sin((alfa+180)*pi/180)+my1;
   dogrultu:='1';
  end;
 if (msf<roundto((r1+r2),-3)) and (msf>roundto(abs(r1-r2),-3)) then
  begin
   aa:=4*sqr(mx1-mx2)+4*sqr(my1-my2);
   bb:=4*(mx1-mx2)*(sqr(mx2)-sqr(mx1)+sqr(r1)-sqr(r2)+sqr(my1-my2))-
   8*mx1*sqr(my1-my2);
   cc:=sqr(sqr(mx2)-sqr(mx1)+sqr(r1)-sqr(r2)+sqr(my1-my2))-4*sqr(my1-
   my2)*(sqr(r1)-sqr(mx1));
   disk:=(bb*bb)-(4*aa*cc);
   x5:=(-bb-sqrt(disk))/(2*aa);
   x6:=((-bb+sqrt(disk))/(2*aa));
   aa:=4*sqr(mx1-mx2)+4*sqr(my1-my2);
   bb:=4*(my1-my2)*(sqr(my2)-sqr(my1)+sqr(r1)-sqr(r2)+sqr(mx1-mx2))-8*my1*sqr(mx1-mx2);
   cc:=sqr(sqr(my2)-sqr(my1)+sqr(r1)-sqr(r2)+sqr(mx1-mx2))-4*sqr(mx1-mx2)*(sqr(r1)-sqr(my1));
   disk:=(bb*bb)-(4*aa*cc);
   y5:=(-bb-sqrt(disk))/(2*aa);
   y6:=((-bb+sqrt(disk))/(2*aa));
   if (roundto(sqrt(sqr(mx1-x5)+sqr(my1-y6)),-5)-roundto(r1,-5)<tle) and
   (roundto(sqrt(sqr(mx2-x5)+sqr(my2-y6)),-5)-roundto(r2,-5)<tle) then
```

EK-5 (Devam). İki yayın kesişimini bulduran program kodları

```
      begin
        fy:=y5;
        y5:=y6;
        y6:=fy;
      end;
    dogrultu:='2';
  end;
 if (dogrultu='1') or (dogrultu='2') then
  begin
    if yon1='sag' then
      begin
        bas1:=basaci1;
        bit1:=bitaci1;
      end;
    if yon1='sol' then
      begin
        bas1:=bitaci1;
        bit1:=basaci1;
      end;
    if yon2='sag' then
      begin
        bas2:=basaci2;
        bit2:=bitaci2;
      end;
    if yon2='sol' then
      begin
        bas2:=bitaci2;
        bit2:=basaci2;
      end;
    aci1:=roundto(acibul(mx1,my1,x5,y5),-3);
    aci2:=roundto(acibul(mx1,my1,x6,y6),-3);
    aci3:=roundto(acibul(mx2,my2,x5,y5),-3);
    aci4:=roundto(acibul(mx2,my2,x6,y6),-3);
    if bas1<bit1 then
      begin
        if ((aci1<bas1) and (aci1<bit1)) or ((aci1>bas1) and (aci1>bit1))  then
          begin
            d1:='0';
          end
        else
          begin
            d1:='1';
          end;
        if ((aci2<bas1) and (aci2<bit1)) or ((aci2>bas1) and (aci2>bit1))  then
          begin
            d2:='0';
          end
        else
          begin
            d2:='1';
          end;
      end;
if bas1>bit1 then
```

EK-5 (Devam). İki yayın kesişimini bulduran program kodları

```
 begin
   if ((aci1<bas1) and (aci1>bit1)) then
     begin
       d1:='0';
     end
   else
     begin
       d1:='1';
     end;
   if ((aci2<bas1) and (aci2>bit1)) then
     begin
       d2:='0';
     end
   else
     begin
       d2:='1';
     end;
 end;
if bas2<bit2 then
 begin
   if ((aci3<bas2) and (aci3<bit2)) or ((aci3>bas2) and (aci3>bit2))  then
     begin
       d3:='0';
     end
   else
     begin
       d3:='1';
     end;
   if ((aci4<bas2) and (aci4<bit2)) or ((aci4>bas2) and (aci4>bit2))  then
     begin
       d4:='0';
     end
   else
     begin
       d4:='1';
     end;
 end;

if bas2>bit2
 begin
   if ((aci3<bas2) and (aci3>bit2)) then
     begin
       d3:='0';
     end
   else
     begin
       d3:='1';
     end;
   if ((aci4<bas2) and (aci4>bit2)) then
     begin
       d4:='0';
     end
   else
```

EK-5 (Devam). İki yayın kesişimini bulduran program kodları

```
          begin
            d4:='1';
          end;
        end;
      end;
  durum:='0';
  araliktami1:='0';
  araliktami2:='0';
  if (d1='1') and (d3='1') then araliktami1:='1';
  if (d2='1') and (d4='1') then araliktami2:='1';
  if (dogrultu='1') and (araliktami1='1') then durum:='1';
  if (dogrultu='2') and (araliktami1='1') and (araliktami2='0') then durum:='20';
  if (dogrultu='2') and (araliktami1='0') and (araliktami2='1') then durum:='02';
  if (dogrultu='2') and (araliktami1='1') and (araliktami2='1') then durum:='22';
end;
```

EK-6. Kırma işlemini gerçekleştiren program kodları

```
begin
  setlength(kesnoklar,adet*2,3);
  setlength(kirunsurlar,adet*12,12);
  krs:=0;
  for i:=1 to adet*2 do
    begin
      kesadet:=0;
      for j:=1 to adet*2 do
        begin
          durum:='0';

          if i<>j then
            begin
              if (kabaofsetler[i-1,1]='Çizgi') and (kabaofsetler[j-1,1]='Çizgi') then
                begin
                  kesnokbul(
                  strtofloat(kabaofsetler[i-1,2]),
                  strtofloat(kabaofsetler[i-1,3]),
                  strtofloat(kabaofsetler[i-1,4]),
                  strtofloat(kabaofsetler[i-1,5]),
                  strtofloat(kabaofsetler[j-1,2]),
                  strtofloat(kabaofsetler[j-1,3]),
                  strtofloat(kabaofsetler[j-1,4]),
                  strtofloat(kabaofsetler[j-1,5])
                  );
                end;
              if (kabaofsetler[i-1,1]='Çizgi') and (kabaofsetler[j-1,1]='Yay') then
                begin
                  kesnokbul1(
                  strtofloat(kabaofsetler[i-1,2]),strtofloat(kabaofsetler[i-1,3]),
                  strtofloat(kabaofsetler[i-1,4]),strtofloat(kabaofsetler[i-1,5]),
                  strtofloat(kabaofsetler[j-1,6]),strtofloat(kabaofsetler[j-1,7]),
                  strtofloat(kabaofsetler[j-1,8]),
                  acibul(strtofloat(kabaofsetler[j-1,6]),strtofloat(kabaofsetler[j-1,7]),
                  strtofloat(kabaofsetler[j-1,2]),strtofloat(kabaofsetler[j-1,3])),
                  acibul(strtofloat(kabaofsetler[j-1,6]),strtofloat(kabaofsetler[j-1,7]),
                  strtofloat(kabaofsetler[j-1,4]),strtofloat(kabaofsetler[j-1,5])),
                  kabaofsetler[j-1,9]);
                end;
              if (kabaofsetler[i-1,1]='Yay') and (kabaofsetler[j-1,1]='Çizgi') then
                begin
                  kesnokbul1(
                  strtofloat(kabaofsetler[j-1,2]),
                  strtofloat(kabaofsetler[j-1,3]),
                  strtofloat(kabaofsetler[j-1,4]),
                  strtofloat(kabaofsetler[j-1,5]),
                  strtofloat(kabaofsetler[i-1,6]),
                  strtofloat(kabaofsetler[i-1,7]),
                  strtofloat(kabaofsetler[i-1,8]),
                  acibul(strtofloat(kabaofsetler[i-1,6]),strtofloat(kabaofsetler[i-1,7]),
                  strtofloat(kabaofsetler[i-1,2]),strtofloat(kabaofsetler[i-1,3])),
                  acibul(strtofloat(kabaofsetler[i-1,6]),strtofloat(kabaofsetler[i-1,7]),
                  strtofloat(kabaofsetler[i-1,4]),strtofloat(kabaofsetler[i-1,5])),
```

EK-6 (Devam). Kırma işlemini gerçekleştiren program kodları

```
          kabaofsetler[i-1,9]);
         end;
       if (kabaofsetler[i-1,1]='Yay') and (kabaofsetler[j-1,1]='Yay') then
         begin
          kesnokbul2(
          strtofloat(kabaofsetler[i-1,6]),strtofloat(kabaofsetler[i-1,7]),
          strtofloat(kabaofsetler[i-1,8]),cibul(strtofloat(kabaofsetler[i-1,6]),
          strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,2]),
          strtofloat(kabaofsetler[i-1,3])),acibul(strtofloat(kabaofsetler[i-1,6]),
          strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,4]),
          strtofloat(kabaofsetler[i-1,5])),kabaofsetler[i-1,9],
          strtofloat(kabaofsetler[j-1,6]),strtofloat(kabaofsetler[j-1,7]),
          strtofloat(kabaofsetler[j-1,8]),acibul(strtofloat(kabaofsetler[j-1,6]),
          strtofloat(kabaofsetler[j-1,7]),strtofloat(kabaofsetler[j-1,2]),
          strtofloat(kabaofsetler[j-1,3])),acibul(strtofloat(kabaofsetler[j-1,6]),
          strtofloat(kabaofsetler[j-1,7]),strtofloat(kabaofsetler[j-1,4]),
          strtofloat(kabaofsetler[j-1,5])),kabaofsetler[j-1,9]);
         end;
      end;
    if durum='1' then
      begin
        kesadet:=kesadet+1;
        kesnoklar[kesadet-1,0]:=floattostr(x5);
        kesnoklar[kesadet-1,1]:=floattostr(y5);
        kesnoklar[kesadet-1,2]:='0';
      end;
    if durum='20' then
      begin
        kesadet:=kesadet+1;
        kesnoklar[kesadet-1,0]:=floattostr(x5);
        kesnoklar[kesadet-1,1]:=floattostr(y5);
        kesnoklar[kesadet-1,2]:='0';
      end;
    if durum='02' then
      begin
        kesadet:=kesadet+1;
        kesnoklar[kesadet-1,0]:=floattostr(x6);
        kesnoklar[kesadet-1,1]:=floattostr(y6);
        kesnoklar[kesadet-1,2]:='0';
      end;
    if durum='22' then
      begin
        kesadet:=kesadet+1;
        kesnoklar[kesadet-1,0]:=floattostr(x5);
        kesnoklar[kesadet-1,1]:=floattostr(y5);
        kesnoklar[kesadet-1,2]:='0';

        kesadet:=kesadet+1;
        kesnoklar[kesadet-1,0]:=floattostr(x6);
        kesnoklar[kesadet-1,1]:=floattostr(y6);
        kesnoklar[kesadet-1,2]:='0';
      end;
  end;
```

EK-6 (Devam). Kırma işlemini gerçekleştiren program kodları

```
if kesadet>2 then
  begin
    setlength(skesnoklar,kesadet,3);
    if kabaofsetler[i-1,1]='Çizgi' then
      begin
        kx1:=strtofloat(kabaofsetler[i-1,2]);
        ky1:=strtofloat(kabaofsetler[i-1,3]);
        tamboy:=
        sqrt((strtofloat(kabaofsetler[i-1,2])-strtofloat(kabaofsetler[i-1,4]))*
        (strtofloat(kabaofsetler[i-1,2])-strtofloat(kabaofsetler[i-1,4]))+
        (strtofloat(kabaofsetler[i-1,3])-strtofloat(kabaofsetler[i-1,5]))*
        (strtofloat(kabaofsetler[i-1,3])-strtofloat(kabaofsetler[i-1,5])));
        tamboy:=roundto(tamboy,-4);
        for k:=1 to kesadet do
          begin
            cmes:=tamboy;
            for f:=1 to kesadet do
              begin
                kx2:=strtofloat(kesnoklar[f-1,0]);
                ky2:=strtofloat(kesnoklar[f-1,1]);
                mes:=sqrt((kx1-kx2)*(kx1-kx2)+(ky1-ky2)*(ky1-ky2));
                mes:=roundto(mes,-4);
                if (mes<=cmes) and (kesnoklar[f-1,2]='0') then
                  begin
                    cmes:=mes;
                    s:=f;
                  end;
              end;
            kesnoklar[s-1,2]:='1';
            skesnoklar[k-1,0]:=kesnoklar[s-1,0];
            skesnoklar[k-1,1]:=kesnoklar[s-1,1];
          end;
      end;
    if kabaofsetler[i-1,1]='Yay' then
      begin
        if kabaofsetler[i-1,9]='sag' then
          begin
            basaci1:=acibul(strtofloat(kabaofsetler[i-1,6]),
            strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,2]),
            strtofloat(kabaofsetler[i-1,3]));
            bitaci1:=acibul(strtofloat(kabaofsetler[i-1,6]),
            strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,4]),
            strtofloat(kabaofsetler[i-1,5]));
            if basaci1<bitaci1 then tamboy:=bitaci1-basaci1;
            if bitaci1<basaci1 then tamboy:=bitaci1+360-basaci1;
          end;
        if kabaofsetler[i-1,9]='sol' then
          begin
            basaci1:=acibul(strtofloat(kabaofsetler[i-1,6]),
            strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,4]),
            strtofloat(kabaofsetler[i-1,5]));
            bitaci1:=acibul(strtofloat(kabaofsetler[i-1,6]),
            strtofloat(kabaofsetler[i-1,7]),strtofloat(kabaofsetler[i-1,2]),
            strtofloat(kabaofsetler[i-1,3]));
```

EK-6 (Devam). Kırma işlemini gerçekleştiren program kodları

```
          if bitaci1<basaci1 then tamboy:=abs(bitaci1+360-basaci1);
          if basaci1<bitaci1 then tamboy:=bitaci1-basaci1;
        end;
      tamboy:=roundto(tamboy,-4);
      for k:=1 to kesadet do
        begin
          cmes:=tamboy;
          for f:=1 to kesadet do
            begin
              kesaci:=acibul(strtofloat(kabaofsetler[i-1,6]),
              strtofloat(kabaofsetler[i-1,7]),strtofloat(kesnoklar[f-1,0]),
              strtofloat(kesnoklar[f-1,1]));
              if kabaofsetler[i-1,9]='sag' then
                begin
                  if basaci1<=kesaci then mes:=kesaci-basaci1;
                  if kesaci<basaci1 then mes:=abs(kesaci+360-basaci1);
                end;
              if kabaofsetler[i-1,9]='sol' then
                begin
                  if basaci1<=kesaci then mes:=kesaci-bitaci1;
                  if kesaci<basaci1 then mes:=abs(kesaci+360-bitaci1);
                end;
              mes:=roundto(mes,-4);
              if (mes<=cmes) and (kesnoklar[f-1,2]='0') then
                begin
                  cmes:=mes;
                  s:=f;
                end;
            end;
          kesnoklar[s-1,2]:='1';
          skesnoklar[k-1,0]:=kesnoklar[s-1,0];
          skesnoklar[k-1,1]:=kesnoklar[s-1,1];
        end;
    end;
  end;
if kesadet<2 then
  begin
    showmessage('kesişim noktalarından birini bulamadı...!');
  end;
if kesadet=2 then
  begin
    krs:=krs+1;
    kirunsurlar[krs-1,0]:=inttostr(krs);
    kirunsurlar[krs-1,1]:=kabaofsetler[i-1,1];
    kirunsurlar[krs-1,2]:=kabaofsetler[i-1,2];
    kirunsurlar[krs-1,3]:=kabaofsetler[i-1,3];
    kirunsurlar[krs-1,4]:=kabaofsetler[i-1,4];
    kirunsurlar[krs-1,5]:=kabaofsetler[i-1,5];
    kirunsurlar[krs-1,6]:=kabaofsetler[i-1,6];
    kirunsurlar[krs-1,7]:=kabaofsetler[i-1,7];
    kirunsurlar[krs-1,8]:=kabaofsetler[i-1,8];
    kirunsurlar[krs-1,9]:=kabaofsetler[i-1,9];
```

EK-6 (Devam). Kırma işlemini gerçekleştiren program kodları

```
            kirunsurlar[krs-1,10]:=kabaofsetler[i-1,10];
        kirunsurlar[krs-1,11]:='0';
end;
    if kesadet>2 then
      begin
       if kabaofsetler[i-1,1]='Çizgi' then
        begin
         for k:=1 to kesadet-1 do
          begin
           krs:=krs+1;
           kirunsurlar[krs-1,0]:=inttostr(krs);
           kirunsurlar[krs-1,1]:=kabaofsetler[i-1,1];
           kirunsurlar[krs-1,2]:=skesnoklar[k-1,0];
           kirunsurlar[krs-1,3]:=skesnoklar[k-1,1];
           kirunsurlar[krs-1,4]:=skesnoklar[k,0];
           kirunsurlar[krs-1,5]:=skesnoklar[k,1];
           kirunsurlar[krs-1,10]:=kabaofsetler[i-1,10];
           kirunsurlar[krs-1,11]:='0';
          end;
       if kabaofsetler[i-1,1]='Yay' then
        begin
         for k:=1 to kesadet-1 do
          begin
           krs:=krs+1;
           if kabaofsetler[i-1,9]='sağ' then
            begin
             kirunsurlar[krs-1,0]:=inttostr(krs);
             kirunsurlar[krs-1,1]:=kabaofsetler[i-1,1];
             kirunsurlar[krs-1,2]:=skesnoklar[k-1,0];
             kirunsurlar[krs-1,3]:=skesnoklar[k-1,1];
             kirunsurlar[krs-1,4]:=skesnoklar[k,0];
             kirunsurlar[krs-1,5]:=skesnoklar[k,1];
             kirunsurlar[krs-1,6]:=kabaofsetler[i-1,6];
             kirunsurlar[krs-1,7]:=kabaofsetler[i-1,7];
             kirunsurlar[krs-1,8]:=kabaofsetler[i-1,8];
             kirunsurlar[krs-1,9]:=kabaofsetler[i-1,9];
             kirunsurlar[krs-1,10]:=kabaofsetler[i-1,10];
             kirunsurlar[krs-1,11]:='0';
           if kabaofsetler[i-1,9]='sol' then
            begin
             kirunsurlar[krs-1,0]:=inttostr(krs);
             kirunsurlar[krs-1,1]:=kabaofsetler[i-1,1];
             kirunsurlar[krs-1,2]:=skesnoklar[k,0];
             kirunsurlar[krs-1,3]:=skesnoklar[k,1];
             kirunsurlar[krs-1,4]:=skesnoklar[k-1,0];
             kirunsurlar[krs-1,5]:=skesnoklar[k-1,1];
             kirunsurlar[krs-1,6]:=kabaofsetler[i-1,6];
             kirunsurlar[krs-1,7]:=kabaofsetler[i-1,7];
             kirunsurlar[krs-1,8]:=kabaofsetler[i-1,8];
             kirunsurlar[krs-1,9]:=kabaofsetler[i-1,9];
             kirunsurlar[krs-1,10]:=kabaofsetler[i-1,10];
             kirunsurlar[krs-1,11]:='0';
            end;
```

EK-6 (Devam). Kırma işlemini gerçekleştiren program kodları

```
            end;
          end;
        end;
    end;
  setlength(kirunsurlar,krs,12);
end;
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : GÖKTAŞ, Mustafa

Uyruğu : T.C.

Doğum tarihi ve yeri : 1984 Niğde

Medeni hali : Evli

Telefon : 0 (505) 230 55 84

e-mail : mgoktas84@gmail.com

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Gazi Üniv./ Makine Eğitimi	2005
Lise	Seyyid Burhaneddin A.M.L.	1996

İş Deneyimi

Yıl	**Yer**	**Görev**
2009-	Gazi Üniversitesi	Öğretim Görevlisi
2004-2009	O.D.T.Ü.	Teknisyen

Yabancı Dil

İngilizce

Yayınlar

1) Göktaş, M., Dilipak, H., Güldaş, A., "Ofsetleme ve Cep Frezeleme İşlemlerinde Analitik Yaklaşım", ***5th International Advanced Technologies Symposium,*** Karabük, (2009)

2) Göktaş, M., Dilipak, H., Güldaş, A., "İki Boyutlu Profillerin İşlenmesinde Takım Yolu ve Ofsetleme için Yeni Bir Algoritma Geliştirilmesi", ***Gazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi,*** 25(1):179-187, (2010)

3) H. Dilipak, A. Güldaş, M. Göktaş, "Yaylar İçin Ofsetleme Algoritması", ***e-Journal of New World Sciences Academy-Technological Applied Sciences***, 5(3):212-219, (2010)

Hobiler

Bilgisayar teknolojileri