

MEKANSAL VERİ ANALİZİNDE POINT IN POLYGON TESTİ

İ. Öztuğ BİLDİRİCİ

Selçuk Üniversitesi Mühendislik Mimarlık Fakültesi Jeodezi ve Fotogrametri Mühendisliği
Bölümü , 42031 Kampüs KONYA, email: bildirici@selcuk.edu.tr

Özet: Coğrafi bilgi sistemlerinde mekansal veri analizinde point in polygon testi oldukça önemli bir yere sahiptir. Söz konusu test ile bir noktasal objenin bir alansal objenin (poligonun) içinde olup olmadığı test edilir. Point in polygon testi, kapalı poligomu oluşturan doğru parçaları ile, test edilen nokta ve poligonun dışındaki bir noktanın oluşturduğu doğru parçasının (test doğru parçası) kesişimine dayanır. Test doğru parçası ile poligonun kesişim sayısı tek ise, nokta poligonun içinde, değilse nokta poligonun dışındadır.

1. GİRİŞ

Coğrafi Bilgi Sistemlerinde mekansal veri analizinde point in polygon testi oldukça önemli bir yere sahiptir. Söz konusu test ile bir noktasal objenin bir alansal objenin (poligonun) içinde olup olmadığı belirlenir. Point in polygon testi, poligonu (alansal objeyi) oluşturan kenarlar ile, test edilen nokta ve poligonun dışındaki bir noktanın oluşturduğu doğru parçasının (test doğru parçası) kesişimine dayanır. Test doğru parçası ile poligonun kesişim sayısı tek ise, nokta poligonun içinde, değilse nokta poligonun dışındadır. Poligonun şekli ne kadar karmaşık olursa olsun, bu basit test yöntemi doğru sonuç verir. Poligonun şekline bağlı olarak testte iki doğru parçasının kesişim problemi önem kazanır. Kesişim problemi, programlama açısından ilginç bir problemdir. Test işleminde aranan sadece iki doğru parçasının kesişip kesişmediğidir. Kesişim noktasının koordinatları ise test açısından gerekli değildir. Kesişme hesaplamaları box testi olarak adlandırılan bir programlama tekniği ile hızlandırılabilir.

Bu çalışmada, box kavramı ve box testi, iki doğrunun kesişim problemi ve point in polygon testi teorik ve programlama tekniği açısından incelenecektir. Önerilen yaklaşımların uygulandığı FORTRAN dilinde yazılmış program parçacıkları (subroutine) da ek olarak verilmiştir.

2. İKİ DOĞRU PARÇASININ KESİŞİM PROBLEMİ

Geometrik olarak iki doğru birbirine paralel değilse kesişir. Ancak CBS, bilgisayar destekli çizim ve genel olarak grafik programlamada iki doğrunun değil iki doğru parçasının kesişimi önemlidir. İki doğrunun kesişiminin genel ifadesi,

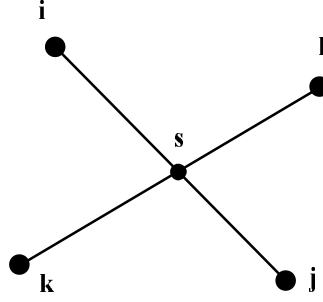
Birinci doğru $Ax + By + C = 0$

İkinci doğru $Ex + Fy + G = 0$ olmak üzere,

$$\begin{aligned} x_s &= \frac{(GB - FG)(FA - EB)}{(CE - AG)(FA - EB)} \\ y_s &= \frac{(CE - AG)(FA - EB)}{(FA - EB)} \end{aligned} \quad (1)$$

İki doğru paralel ise, $FA - EB = 0$ olur. Bu durumda kesişim mevcut değildir.

Yukarıdaki gibi lineer denklem formu ile çözüm yapılırsa önce kesişim noktası hesaplanır, sonra kesişim noktasının iki doğru üzerinde olup olmadığı araştırılır. Çeşitli özel durumların dikkate alınmasını gerektiren bu yaklaşım yerine, programlama açısından daha kullanışlı olan ve nokta koordinatlarına dayanan aşağıdaki yöntem izlenebilir. Birinci doğru parçasının başlangıç ve bitim noktaları $i(x_i, y_i)$ ve $j(x_j, y_j)$, ikinci doğru parçasının başlangıç ve bitim noktaları $k(x_k, y_k)$ ve $l(x_l, y_l)$ olsun (Şekil 1).



Şekil 1: İki doğru parçasının kesişimi

Doğru parçalarının paralel olup olmadığını belirlemek için d parametresi hesaplanır.

$$d = (x_j - x_i)(y_k - y_l) - (x_k - x_l)(y_j - y_i) \quad (2)$$

$d=0$ ise doğru parçaları paralel olup kesişim hesaplanamaz. $d \neq 0$ ise p_1 ve p_2 parametreleri hesaplanır.

$$p_1 = \frac{(x_k - x_l)(y_i - y_k) - (x_i - x_k)(y_l - y_i)}{d} \quad (3)$$

$$p_2 = \frac{(x_i - x_k)(y_j - y_i) - (x_j - x_i)(y_l - y_k)}{d}$$

$0 \leq p_1 \leq 1$ ve $0 \leq p_2 \leq 1$ ise kesişim noktası doğru parçalarının üzerindedir. Kesişim noktasının koordinatları,

$$x_s = x_i + p_1(x_j - x_i) \quad (4)$$

$$y_s = y_i + p_1(y_j - y_i)$$

Kesişim problemi programlama tekniği açısından düşünülürse reel sayılar olan nokta koordinatlarının hangi değişken tipi ile tanımlandığı önemlidir. Double precision (8 byte) değişken tipi kullanıldığında ondalık noktanın yerinden bağımsız olarak 14 rakam anlamlıdır. Sekiz değişik değerden hesaplanan d parametresinin (2) doğruların paralel olması durumunda tam olarak sıfır olması beklenemez. Daha da açık olarak d parametresinin nadiren sıfır çıkacağı, genellikle sıfıra çok yakın bir değer alacağı söylenebilir.

Doğru parçalarının paralellliğini belirlemek için bir hesap hassasiyeti belirlemek gerekir. Örneğin d parametresi altıncı basamağına kadar sıfır ise sıfır kabul edilebilir. Bu durumda yazılacak kod aşağıdaki gibi olabilir:

```
if abs(d)<1e-6 then ...
```

Bazı hallerde sadece kesişimin var olup olmadığını belirlemek gerekir. Bu durumda önce paralellik incelenir, sonra p_1 ve p_2 parametreleri hesaplanarak kesişimin doğru parçaları üzerinde olup olmadığı belirlenir. Kesişim noktasının koordinatları hesaplanmaz. Örneğin point in polygon testinde sadece kesişimin var olup olmadığını bilmek önemlidir.

p_1 ve p_2 parametrelerinin geometrik anlamı:

- $p_1=0$ ise, kesişim noktası i noktası ile çakışık
- $p_1=1$ ise, kesişim noktası j noktası ile çakışık
- $p_2=0$ ise, kesişim noktası k noktası ile çakışık
- $p_2=1$ ise, kesişim noktası l noktası ile çakışık

Yine programlama tekniği açısından p_1 ve p_2 parametrelerinin tam olarak sıfır ya da bir olması beklenemez. Bunun yanında sayısallaştırma hataları vb gibi nedenlerle parametrelerin sıfırdan çok az küçük ya da birden çok az büyük olmaları durumunda kesişim noktasının doğru parçalarının ilgili uç noktaları ile çakışık olarak kabul edilmesi gerekebilir. Uygun seçilecek bir ε tolerans değerine göre parametrelerin irdelenmesi,

$$-\varepsilon \leq p_1 \leq 1 + \varepsilon \quad \text{ve} \quad -\varepsilon \leq p_2 \leq 1 + \varepsilon \quad (5)$$

biçiminde yapılabilir. Ancak p_1 ve p_2 parametreleri birer orantı olup bir uzaklık ifade etmezler. Bu bakımdan ε tolerans değerinin seçimi çok kolay değildir. Bu noktada şöyle bir yaklaşım önerilebilir. Tolerans değeri kabul edilebilir bir yuvarlatma hatası olarak alınır, daha sonra koordinatları hesaplanan kesişim noktasının uç noktalarına olan uzaklıklarına bakılıp bu noktalarla çakışık kabul edilip edilemeyeceğine bakılır.

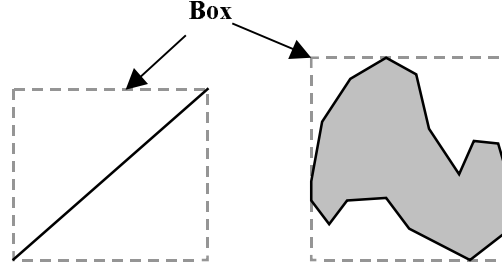
3. BOX TESTİ

Grafik programlama tekniğinde box testi olarak bilinen yöntemle, çizgisel ve alansal objelerin birbiri ile olan yakınlığı irdelenir. Kutu olarak da Türkçeleştirilebilecek box kavramı çizgisel ya da alansal objenin dışına çizilen, objeyi kapsayan, koordinat eksenlerine paralel bir dikdörtgen objeyi ifade eder. Box yardımıyla sadece mantıksal karşılaştırma yaparak iki çizgisel objenin kesişme olasılığı olup olmadığı, iki alansal objenin ise birbirini örtme olasılığının olup olmadığı belirlenebilir.

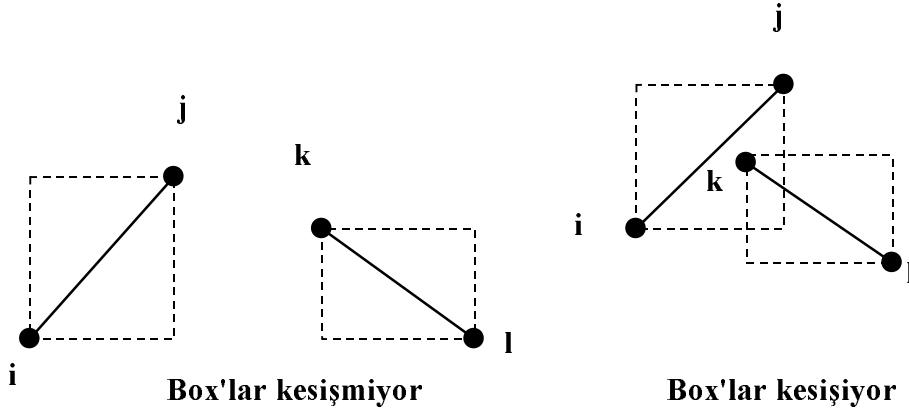
Şekil 2'de de görüldüğü gibi, box koordinatları ilgili objenin maksimum ve minimum koordinat değerlerinden elde edilebilir. i-j ve k-l doğru parçalarının kesişme olasılığını irdeme durumunda box testi (Şekil 3),

```
if max{yi, yj} < min{yk, yl} AND max{xi, xj} < min{xk, xl} then
>> Kesişme mümkün değil
else
>> Kesişme mümkün
end if
```

şeklinde kodlanabilir. Yapılan bu karşılaştırmalar sonucu kesişme mümkün değilse kesişimle ilgili hiç bir hesaplama gerek yoktur [Cromley, 1992; Bildirici, 2000]. Benzer şekilde alansal objelerin de birbirini örtme olasılığının olup olmadığı test edilebilir.



Şekil 2: Box kavramı



Şekil 3: Box testinin geometrik anlamı

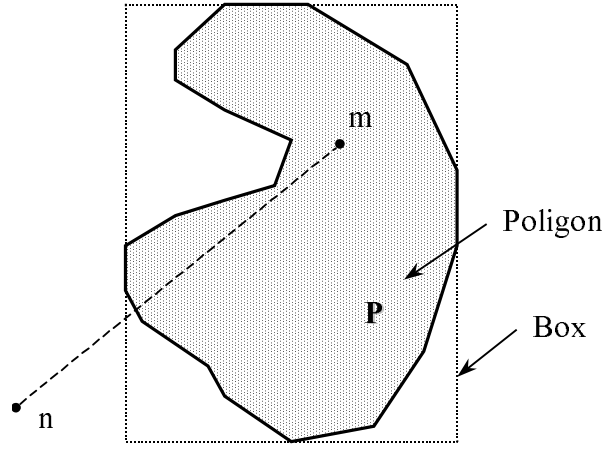
4. POINT IN POLYGON TESTİ

"Point in Polygon" testi, bir noktanın bir alansal objenin içinde olup olmadığını belirlemesi problemidir. Bir noktanın bir alansal objenin (kapalı şeklin) içinde olup olmadığını belirlemek için kapalı şeklin kesinlikle dışında olan bir yardımcı noktadan yararlanılır. Araştırılan nokta m , yardımcı nokta n olmak üzere $m-n$ doğru parçasının, şekli kaç defa kestiği belirlenir (Şekil 4). Kesişim sayısı tek ise nokta şeklin içinde, çift ise dışındadır [Cromley, 1992]. Point in polygon testi kendi içinde son derece basittir. Ancak arkasında programlama açısından çok da basit olmayan kesişim problemi vardır. Burada kesişim noktasının koordinatları ile değil yalnızca kesişimin var olup olmadığı ile ilgilenilmektedir.

Kesişim sayısının araştırılmasından önce, m noktasının P poligonu içinde olup olmayacağını incelemek gerekir (Şekil 4). m noktası poligonu oluşturan box'ın içinde değilse poligonun da içinde olamaz. Box'ın sol alt köşe koordinatları poligonu oluşturan nokta koordinatlarının en küçükleri, sağ üst köşe koordinatları ise şekli oluşturan koordinatların en büyükleri olduğuna göre;

$$\text{if } \min\{x_i\} > x_m > \max\{x_i\} \text{ AND } \min\{y_i\} > y_m > \max\{y_i\}$$

şartı sağlanmıyorsa, nokta poligonun içinde olamaz. Şart sağlanıyorsa kesişim hesaplarına geçilebilir [Bildirici, 2000].



Şekil 4: Poligon içinde olup olmadığı araştırılan nokta (p), yardımcı nokta (n) ve box

Yardımcı n noktasının koordinatları, poligonu oluşturan box'dan yararlanarak aşağıdaki gibi hesaplanabilir:

$$\begin{aligned} x_n &= \min(x) - (\max(x) - \min(x)) \\ y_n &= \min(y) - (\max(y) - \min(y)) \end{aligned} \quad (6)$$

(7) eşitliği ile n noktasının kesin olarak poligonun dışında olması sağlanabilir. Uzakta bir n noktası seçme yerine eksenlerden birine paralel bir yardımcı doğru da seçilebilir [Cromley, 1992].

Poligonu oluşturan kenar sayısı kadar kesişim hesabı yapılarak, kesişme sayısı belirlenmek zorundadır. Bu aşamada da bölüm 3 de değinildiği gibi kesişim öncesi box testi uygulanarak (Şekil 3), m-n doğrusu ile ilgili kenarın kesişme olasılığı var ise d (2), parametresi ile paralellik araştırılır. Paralel olma durumu yok ise p_1 ve p_2 (3) parametreleri hesaplanır. Eğer,

$$0 > p_1 > 1 \quad \text{VE} \quad 0 > p_2 > 1 \quad (7)$$

şartı sağlanıyorsa, kesişim var, aksi halde yoktur. Kesişim noktasının koordinatlarına ihtiyaç olmadığından hesaplama bu noktada kesilir, kesişim sayısı bir artırılır ve bir sonraki kenara geçilir.

Poligonu oluşturan tüm kenarlar için kesişim olup olmadığı araştırıldıktan sonra, toplam kesişim sayısı tek ise nokta poligonun içinde, çift ise değildir. Bu noktada bir üçüncü olasılık noktanın poligonu oluşturan kenarların birinin üzerinde olmasıdır. (7) bağıntısına göre nokta herhangi bir kenar üzerinde ise ya da nokta poligonu oluşturan noktalardan biri ile çakışık ise noktanın poligon dışında olduğu kararı verilmektedir. Çünkü nokta kenarlardan biri üzerinde ise p_1 ya da p_2 sifıra ya da bire eşit olur. Nokta poligonu oluşturan noktalardan biri ile çakışık ise hem p_1 hem de p_2 sifıra ya da bire eşit olur. Bu durumlarda nokta poligonun içinde kabul edilecekse (7) bağıntısı aşağıdaki gibi olmalıdır.

$$0 \geq p_1 \geq 1 \quad \text{VE} \quad 0 \geq p_2 \geq 1 \quad (8)$$

p_1 , p_2 ve d parametreleri reel sayılar olduğundan, değişken tipi olarak double precision kullanılması gerekir. Ancak bu şekilde tanımlanan değişkenlerin tam olarak sıfıra ya da bire eşit olmaları beklenemez. Sıfıra ya da bire eşit olma (5) bağıntısındaki mantık ile yapılmalıdır. Buradaki ε parametresi dikkatli seçilmelidir. Buradaki parametreler uzunluk değil, orantıdır. Örneğin p_1 parametresinin birden farkı, kesişim noktasının kenarlardan birine ne kadar yaklaştığı (uzaklık olarak) hakkında bir fikir vermez.

p_1 ve p_2 parametreleri uygun şekilde test edilerek point in polygon testi sonucu, nokta içerde, nokta poligon kenarı üzerinde ve nokta dışarıda olmak üzere üç değişik sonuç da elde edilebilir.

Box, kesişim ve point in polygon algoritmasının uygulanması konusunda Fortran dilinde kodlanmış üç program parçasığı ek olarak verilmiştir.

5. SONUÇ

Point in polygon testi, Coğrafi Bilgi Sistemlerinde mekansal sorgulama ve analizde önemlidir. Algoritma kendi içinde programlama olarak basittir. Ancak algoritma, programlama açısından ilginç özellikleri olan iki doğru parçasının kesişimine dayanmaktadır. Bu çalışmada önce kesişim problemi irdelenmiş, daha sonra testin dayandığı algoritma programlama tekniği açısından tartışılmıştır. Ek olarak her iki problemin çözümü için Fortran dilinde kodlanmış program parçacıkları verilmiştir.

6. KAYNAKLAR

Cromley, R.G., 1992, Digital Cartography, Prentice Hall, New Jersey, 316p.

Bildirici, İ.Ö., 2000, 1: 1000-1: 25 000 Ölçek Aralığında Bina ve Yol Objelerinin Sayısal Ortamda Kartografik Genelleştirmesi, Doktora Tezi, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

7. EK: Program Kodları

```
c*****BOX BELIRLEME*****
      subroutine cg_box(xmin,ymin,xmax,ymax,x,y,pkt_anz,rnull)
c*****Decalarations*****
      implicit none
      integer*4    i,pkt_anz
      real*8       xmin,ymin,xmax,ymax,rnull
      real*8       x(pkt_anz),y(pkt_anz)
c*****Program*****
      xmin=1.d09
      ymin=1.d09
      xmax=0.d0
      ymax=0.d0

c
      do i=1,pkt_anz
         if(x(i).gt.xmax) xmax=x(i)
         if(x(i).lt.xmin) xmin=x(i)
         if(y(i).gt.ymax) ymax=y(i)
         if(y(i).lt.ymin) ymin=y(i)
      end do
      xmin=xmin-rnull
      ymin=ymin-rnull
      xmax=xmax+rnull
```

```

        ymax=ymax+rnull
100    end
c*****KESISIM*****
      logical function cg_int2(xi,yi,xj,yj,xk,yk,xl,yl,p1,p2)
c*****
c 1. dogru i,j
c 2. dogru k,l
c p1 0 ile 1 arasinda ise kesisim 1. dogru parcasi uzerinde
c p2 0 ile 1 arasinda ise kesisim 2. dogru parcasi uzerinde
c kesim varsa true yoksa false (kesim uzantida ise yine false)
c kesim noktası noktalarından biri ile çakışık ya da kenarlardan biri
c uzerinde olsa da true
c
c Oztug Bildirici 12/2002
c
c*****Declarations*****
      implicit none
      real*8      xi,yi,xj,yj,xk,yk,xl,yl
      real*8      a1, a2,b1, b2,c1, c2,d,p1, p2
      real*8      r_null
      parameter (r_null=1.d-4)
c*****Program*****
      a1 = xj - xi
      a2 = yj - yi
      b1 = xk - xl
      b2 = yk - yl
      c1 = xi - xk
      c2 = yi - yk
      d = a1*b2 - a2*b1
c ... paralel mi?
      if( dabs(d) .lt. r_null) then
         cg_int2=.false.
         goto 100
      end if
c kesim noktası varmi?
      p1 = ( b1*c2 - b2*c1 ) / d
      p2 = ( c1*a2 - c2*a1 ) / d
      if(dabs(p2).le.1.d-8) p2=0.d0
      cg_int2=p1.ge.0.d0.and.p1.le.1.d0
      * .and.p2.ge.0.d0.and.p2.le.1.d0
100    end
c*****POINT IN POLYGON*****
      logical function cg_pip(x,y,n,xp,yp)
c*****Declarations*****
      implicit none
      integer*4   i,j,n,schnitte
      real*8      x(n),y(n),xp,yp,xr,yr
      real*8      xmin,ymin,xmax,ymax,rnull,p1,p2
      parameter   (rnull=1.d-4)
c External fuct
      logical     cg_int2
c*****Program*****
      schnitte=0
c box hesapla..
      call cg_box(xmin,ymin,xmax,ymax,x,y,n,rnull)
      xr=2.d0*xmax-xmin
      yr=2.d0*ymax-ymin
      if(xp.ge.xmin.and.yp.ge.ymin.and.
      * xp.le.xmax.and.yp.le.ymax) then
         do i=1,n
            if(i.eq.n)then
               j=1
            else
               j=i+1
            end if
            if(dabs(x(i)-xp).le.rnull.and.dabs(y(i)-yp).le.rnull.or.
            * dabs(x(j)-xp).le.rnull.and.dabs(y(j)-yp).le.rnull) then
               cg_pip=.true.
               goto 100
            end if
            if(cg_int2(x(i),y(i),x(j),y(j),xp,yp,xr,yr,p1,p2)) then
               if(p1.ge.0.d0.and.p1.le.1.d0.and.p2.eq.0.d0) then
                  cg_pip=.true.
                  goto 100
               end if
               schnitte=schnitte+1
            end if
         end do
      end if

```

```
        end do
        cg_pip=(schnitte.gt.0.and.mod(schnitte,2).ne.0)
        else
        cg_pip=.false.
    end if
100  end
```