YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UYDU GÖRÜNTÜLERİNDEN ZİNCİR KOD YÖNTEMİ KULLANILARAK OTOMATİK YOL ÇIKARIMI

YÜKSEK LİSANS TEZİ

Muhammed TEKİN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

HAZİRAN 2019

YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UYDU GÖRÜNTÜLERİNDEN ZİNCİR KOD YÖNTEMİ KULLANILARAK OTOMATİK YOL ÇIKARIMI

YÜKSEK LİSANS TEZİ

Muhammed TEKİN
(105105013)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müfit ÇETİN

HAZİRAN 2019

YALOVA Üniversitesi Fen Bilimleri Enstitüsü'nün 105105013 numaralı Yüksek Lisans Öğrencisi **Muhammed TEKİN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"UYDU GÖRÜNTÜLERİNDEN ZİNCİR KOD YÖNTEMİ KULLANILARAK OTOMATİK YOL ÇIKARIMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Müfit ÇETİN** Yalova Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Müfit ÇETİN** Yalova Üniversitesi	.....
	Dr. Öğr. Üyesi Osman Hilmi KOÇAL Yalova Üniversitesi	.....
	Prof. Dr. Mehmet YILDIRIM Kocaeli Üniversitesi	.....

Teslim Tarihi : **13 Mayıs 2019**
Savunma Tarihi : **13 Haziran 2019**

Eşime ve çocuklarıma,

ÖNSÖZ

Sayın hocam Prof.Dr.Müfit Çetin'e bu çalışma süresince vermiş olduğu destek ve yardımlarından ötürü çok teşekkür etmek istiyorum.

Ayrıca tez süreci boyunca bilgi ve tecrübeleriyle sürekli bana yardımcı olan Yalova Üniversitesi Bilgisayar Mühendisliği Bölümündeki tüm hocalarıma da ayrı ayrı teşekkürlerimi sunarım.

Son olarak aileme de bu süreç içerisinde desteklerini bir an olsun esirgemediklerinden dolayı müteşekkirim.

Mayıs 2019 Muhammed Tekin

İÇİNDEKİLER

Sayfa

KISALTMALAR

GPS **:** Global Positioning System
CBS **:** Coğrafik Bilgi Sistemleri
LCS **:** Longest Common Subsequence (En Uzun Ortak Alt Dizi)
RGB **:** Red-Green-Blue (Kırmızı-Yeşil-Mavi)

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

UYDU GÖRÜNTÜLERİNDEN ZİNCİR KOD YÖNTEMİ KULLANILARAK OTOMATİK YOL ÇIKARIMI

ÖZET

Gerek navigasyon gerekse de Coğrafik Bilgi Sistemlerinin doğru çalışabilmesi açısından güncel kara yolu haritalarına sahip olmak son yılların önemli araştırma konulardan birisidir. Verilerin büyüklüğünden kaynaklı iş yükü dolayısıyla da işlemlerin otomatize edilmesi büyük önem taşımaktadır.

Bundan dolayı bu çalışmada uydu görüntüsündeki kara yollarının tespitinde yeni bir yarı otomatik yöntem geliştirilmeye çalışılmıştır. Sistem beş ana aşamadan oluşmaktadır. İlk aşama, uydu görüntüsündeki kenarlıkların ortaya çıkarılma işlemidir. İkinci aşama, zincir kod yönteminin (chaincode) kenar nesnelerine uygulanarak her bir kenar nesnesine ait zincir kodun elde edilmesidir. Üçüncü aşama, en uzun ortak alt dizi (longest common subsequence) yöntemi ile zincir kod benzerlikleri karşılaştırılarak birbirine benzer kenarların otomatik tespit edilmesidir. Bu aşama sonunda benzer kenarların arasındaki alanlar olası yol pikselleri olarak işaretlenmesidir. Dördüncü aşama, önceki aşamada yol olarak belirlenmiş piksellerin ortalama gri renk değerlerinin hesaplanıp bu değerden küçük piksellerin elimine edilmesidir. Son aşama ise önceki adımda yol olması muhtemel olarak belirlenmiş piksellerin bölge büyütme algoritmasına besleme değeri olarak verilmesi ile gerçek yol piksellerinin belirlenmesidir.

Geliştirilen yöntem, yerleşim yeri dışı bölgelerdeki yolları içeren dört farklı uydu görüntüsü üzerinde uygulanmış ve bunların ortalama başarım değerleri sırasıyla duyarlılık (sensitivity) %88, özgüllük (specifity) %96, kesinlik (accuracy) %94, F1 puanı %86 olarak elde edilmiştir. Söz konusu sonuçlar aynı uydu görüntülerini kullanan literatürdeki bir başka yöntem ile karşılaştırılmış ve önerilen sistemin daha az işlem ile yakın sonuçları elde edebildiği görülmüştür.

AUTOMATIC ROAD EXTRACTION FROM SATELLITE IMAGES BY USING CHAINCODE METHOD

SUMMARY

It is one of the most important research subjects of recent years to have current road maps in terms of both navigation and Geographic Information Systems. Because of data size it takes a long time to process and it makes automating this process is very important.

Therefore, in this study, a new semi-automatic method has been tried to be developed for the detection of land roads in satellite images. The system consists of five main stages. The first stage is the edge detection in the satellite image. The second step is to obtain the chaincode for each edge by applying chaincode method. The third stage is the automatic detection of similar edges by comparing their chaincodes by longest common subsequence algorithm. At the end of this step the areas between the similar edges will be marked as possible road pixels. The fourth stage is to calculate the average gray color values of the pixels specified as the road in the previous stage and eliminate the pixels smaller than this value. The last step is to determine the actual road pixels by giving the pixels as the feed value to the region growing algorithm, which is likely to be the road in the previous step.

The developed method was applied on four different satellite images those includes roads in the non-residential areas. Their average performance values were 88%, specifity 96%, accuracy 94%, F1 score 86%, respectively. The results were compared with another method in the literature using the same satellite images, and the proposed system was able to achieve approximately the same results with fewer steps.

1. GİRİŞ

Keşfetme duygusu, yaradılışı gereği insanoğlunun her zaman sahip olduğu bir duygu olmuştur. Bu duygu sayesinde doğduğu günden itibaren çevresini tanıyıp öğrenmeye başlamış, önceleri ebeveynlerini ve evini keşfetmiş zamanla da dış dünyayı keşfetmeye çalışmıştır. Yaşadığı bölgeyi ve şehri derken her zaman daha da ötelerini merak etmiştir. Dış dünyayı keşfetmeye çalışırken ilk etapta kendi vücudundan yani ayaklarından faydalanıyorken daha öteleri için çeşitli ulaşım araçlarına ihtiyaç duymuştur[1]. Evcilleştirilen at, eşek ve deve gibi hayvanlardan sonra denizleri aşıp yeni yerlere ulaştıracak gemiler insanlığın hizmetinde olmuştur. Özellikle pusulanın icadıyla çok daha uzak mesafelere gidilebilmiştir. İlk başta keşfetme amaçlı olan bu seyahatler daha sonraları keşfedilen topraklara hükmetme, topraklarını büyütme, ticaret rotaları oluşturma ve ardından da diğer medeniyetlerin topraklarını ele geçirmeye de dönüşmüştür. Bugün ise yeryüzü dışındaki hayatı keşfetme arzusu ile uzay araçlarından istifade edilmektedir.

Bu keşif süreçlerindeki en önemli araçlardan birisi ise haritalardır. Haritalar sadece keşif amaçlı değil aynı zamanda bulunduğu bölgeyi daha iyi anlamak, sahip olduklarının sınırını belirlemek (evi, tarlası, şehri, ülkesi vb.), gideceği yere göre en uygun (kısa) rotayı çizmek ve topolojik durum bilgilerini kendinden sonrakilere de aktarmak için hep en önemli araçlardan birisi olmuşlardır.

Harita, Uluslararası Kartografya Birliği (International Cartographic Association) tarafından son olarak 1991 yılında tanımlanmıştır. Bu tanım, “Harita, belirlenmiş bir kullanım amacı için gerçek doğa (haritası yapılan bölge) ile ilişkili seçilmiş bilgilerin aktarımını yapan bütüncül yapıda görsel, dokunsal ya da sayısal kartografik üründür” biçimindedir. “Dokunsal” kavramı ile görme özürlüler yani haritadan bilgileri ancak parmaklarıyla dokunarak alabilenler için üretilmiş haritalar ifade edilmektedir. Sayısal kavramı ise bilgisayar ortamındaki haritalar için kullanılmaktadır[2].

İlk harita çizimlerinin milattan önce 24 bin ila 25 bin yıl öncesine kadar dayandığı düşünülmektedir[3]. Elbette bu haritaları günümüz haritaları ile karşılaştırmak çok doğru olmayabilir. Çünkü bu dönemdeki haritalar sadece basit mağara duvarı çizimleridir. Daha yakın tarihli kabul gören haritalardan birisi ise Çatalhöyük'te bulunmuştur. Yapım tarihi M.Ö. 6200 yıllarında olduğu düşünülmektedir[4]. Bu tarih, yazının keşfedilmesinden çok uzun zaman önceye tekabül etmektedir. Bir başka örnek ise Babillerin M.Ö. 6.yy dayanan haritasıdır[5]. Bugün kullanılan haritalara benzer çalışmalar ise ancak XVI. yüzyıla doğru bulunabilmektedir.

Diğer taraftan baskı cihazlarının gelişmesi de haritaların daha çok insanın kullanmasına olanak sağlamıştır. Özellikle XX. yüzyılla beraber hava ve uydu fotoğrafçılığının ortaya çıkması haritacılığı çok daha farklı bir noktaya taşımıştır. Önceleri uçaklardan çekilen hava fotoğrafları daha doğru haritalar oluşmasını sağlarken sonraları uzaydan çekilen uydu fotoğrafları sayesinde daha da hassas haritalar oluşturulabilmiştir. Bununla birlikte bu görüntülerin hızlı ve sürekli bir şekilde elde edilebilmesi de haritaların dahada sık güncellenebilmesini sağlamıştır. Güncel ve doğru haritalara sahip olma konusu özellikle navigasyon sistemlerinde, şehir planlamada, acil durumlara müdahalelerde önemli bir konudur[6].

Bu gelişmeler dijital haritacılık, GPS ve navigasyon sistemlerinin oluşmasını sağlamış ve zaman içerisinde araç takip sistemlerinde ve akıllı telefonlarda kullanılmaya başlanmıştır. Bugün artık haritalar sessiz birer kâğıt parçası olmaktan çıkıp bizi yönlendiren yardımcılara dönüşmüşlerdir. Daha da ötesi sayısal haritalar sayesinde coğrafik şekiller, binalar ve kara yolları üç boyutlu (3D) olarak gösterebilmekte ve trafik bilgileri de sunabilmektedir. Bu konuda Google Maps, Waze ve Bing örnek olarak verilebilinir. Günümüzde uydular ile sadece dünyanın değil diğer gezegenlerin ve uzayın da haritasının çıkarılmasına çalışılmaktadır.

İlk örnekleri 90'ların başlarında görülen araç navigasyon sistemleri[7] bugün hayatımızın vazgeçilmez birer parçası haline gelmiştir. Özellikle farklı şehirlere yapılan seyahatlerde gidilecek yönü bulma için mükemmel birer yardımcı araçtırlar. Bu sistemler, uydular sayesinde kişinin coğrafik konumunu otomatik olarak bulabilmektedir. Piyasada birçok navigasyon sistemi mevcut olmakla birlikte en yaygın olanı GPS (Global Positioning System) sistemidir. İlk adı Navstar GPS olan bu sistem, 1973 yılında Amerikan savunma bakanlığı tarafından geliştirilmiştir. Başlangıçta askeri amaçlar için geliştirilmiş ama 80'li yıllarda sivil kullanıma da açılmıştır. Sistemin çalışması için GPS sinyal alıcı cihazın en az 4 adet uyduyu görebilecek bir açık ortamda olması gerekmektedir. Alıcının telefon ya da internet bağlantısının olması gerekli değildir ancak mevcut ise konum daha da hassas elde edilebilmektedir[8]. Günümüzde hassasiyet değeri 5m civarında olup yakın gelecekte 30 cm hassasiyetle konum tespiti yapılabilmesi ön görülmektedir[9].

GPS sistemi özellikle araç takip ve navigasyon sistemlerinde sıkça kullanılmaktadır. Ancak bu sistemlerinin sağlıklı çalışabilmesi için sadece GPS sinyalinin doğru alınması yeterli değildir. Buradaki amaç kara yolu rotasını bulma olduğundan, navigasyon sistemindeki kara yollarının doğru ve güncel olması da sistemin doğru çalışması açısından önem arz etmektedir. Bu açıdan, uydu görüntülerinde ki yolların tespiti konusu 1970'lerden beri askeri operasyonlarda otomatik pilotlu araçların yönlendirilmesinde, şehir planlama ve doğal afetler gibi hem askeri hem de sivil açıdan büyük önem taşımaktadır[10].

Bunula birlikte işlenecek verilerin büyüklüğünden dolayı da bu işlemlerin otomatikleştirilmesi hem zaman hem de iş gücü açısından ciddi faydalar sağlamaktadır. Ancak henüz bu konuda tamamen insandan bağımsız ve güvenilir bir sistem geliştirilememiştir[11]. Yine de insanın iş yükünü olabildiğince azaltmak için birçok çalışma yapılmaktadır. Bu konuda tam başarımın elde edilememesine neden olan ve problemi karmaşıklaştıran bazı faktörler kısaca şöyle özetlenebilir[12]

- Uydu görüntülerinin çözünürlüklerinin farklı oluşu (46 cm'den 25 m'ye kadar farklı çözünürlükler)[13]
- Görüntülerdeki bazı bölgelerinin bulutlarla kaplı olması
- Ağaç, ev ve araçlar gibi istenmeyen nesnelerin yol görüntüsünü engellemesi
- Yolların uzunluk ve genişliklerinin farklı olması

- Kavşakların ve özellikle döner kavşakların düz yollara göre şekilsel olarak farklı olmaları
- Bazı binaların düz kenarlarının yol şeklinde yanıltıcı olması gibi

1.1 Tezin Amacı

Geliştirmiş olduğumuz yöntem, elle yapılan güncellemelerdeki hataların ve zamansal kayıpların önüne geçmeyi hedefleyen yarı otomatik bir işlemdir. Bu yöntem, kullanıcının sadece yol genişlik bilgisi girerek gerçekleştirilen çeşitli morfolojik işlemlerin ardından zincir kod ve LCS algoritmaları ile yolları ortaya çıkaracaktır. Sistemin başarımını daha da iyileştirilmesi için bölge büyütme yöntemi de kullanılacaktır. Gelinmek istenen nokta ise verilen bir uydu görüntüsündeki kara yolların başarıyla tanınabilmesidir. Ancak problemin geniş kapsamlı olmasından dolayı burada kara yolları olarak şehirlerarası ana yollar ile çalışma sınırlandırılmıştır.

1.2 Literatür Araştırması

Uydu görüntülerindeki yolların tespiti için literatürde yarı otomatik ve tam otomatik çalışmalar mevcuttur. Bu bölümde her iki tipteki sistemler için de örnekler verilecektir.

Fateme ve diğ. (2008) [14] yayınlarında bulanık C – Ortalamalar kümeleme analizi yöntemi ile spektral olarak görüntüleri verilen bir eşik değerine göre sınıflandırmışlardır. Ardından sonuçları daha iyi hale getirmek için de farklı morfolojik işlemler uygulayıp (genişleme, erozyon, açma, kapama vb.) graf teorisi ve en küçük tarama ağacı yöntemlerinin de yardımıyla kara yollarının orta noktalarını %98 doğrulukla tespit etmişlerdir.

Sirmacek ve Unsalan (2010) [15] yayınlarında Canny kenar bulma algoritmasından elde ettikleri eğim bilgisi ile bir mekânsal oylama matrisi elde etmişler. Ardından bu matrise en büyük oy değerinin yarısı olarak kabul ettikleri eşik değerini uygulamışlar ve sonrasında da bağlı bileşenler analizi yaparak olası güçlü yol bölümlerini elde etmişlerdir. Bir takip algoritması sayesinde de kopuk pikselleri birleştirmişlerdir. Ikonos uydusuna ait 6 adet uydu görüntüsü üzerinde yaptıkları uygulamalarında %80 civarı bir doğruluğu yakalayabilmişlerdir.

Ali ve diğ. (2010) [16] yayınlarında farklı morfolojik filtrelerden yararlanarak yol katmanı ile arka plan katmanlarını ayırmaya çalışmışlar ve %95.8 hassasiyet başarım değeri elde etmişlerdir.

Das ve diğ. (2011) [12] yayınlarında olasılıklı destek vektör makinesi yöntemi ile beraber konvolüsyonel yapay sinir ağı modelini birlikte kullanmışlardır. Ağın eğitimi için toplamda 12200 adet 21x21 boyutlarında görüntü kullanılmıştır. Bu görüntüler, bir uzman tarafından şehir içi, şehir dışı ve kırsal alan görüntülerinden oluşan 100 adetlik 512x512 çözünürlükteki veri seti içerisinden seçilmiştir. Görüntülerin 7200 tanesi yol olmayan, 5000 tanesi de yol olan alanlardan oluşmaktadır. Geliştirdikleri yöntemi 100 adetlik veri setine uygulayarak ortalama %89 başarım elde etmişlerdir.

Revathi ve Sharmila (2013) [6] seviye kümesi ve ortalama öteleme yöntemleri ile %90'a yakın bir başarım elde edebilmiştir.

Kirthika ve Mookambiga (2011) [17] ise yapay sinir ağlarından faydalanarak yol tespitine çalışmışlardır. Her bir pikselin farklı spektraldeki değerlerini kullanarak kontrast, enerji, entropi ve homojenlik değerlerini elde etmişler ve yapay sinir ağında öğrenme için kullanmışlardır. Bu şekilde %80 doğrulukla tespit yapabilmişlerdir.

Abraham ve Sasikumar (2013) [18] dalgacık tabanlı bir segmentasyon yöntemini bulanık mantık ile birleştirerek % 80 üzerinde doğrulukla yol tespiti yapmışlardır. Ayrıca, geliştirilen sistemde görüntüdeki gürültü etkisini azaltmak için bir önişlem aşamasından faydalanmışlardır.

Daha yeni çalışmalardan olan Alshehhi ve Marpu (2017) [19]'ün geliştirdikleri sistemde, yollar ile yol olmayan görüntülerin ayrıştırılması için önce Gabor ve morfolojik filtreleme yapılarak renk ve şekil bilgilerine göre graf tabanlı bir segmentasyon yöntemi ile %90 üzeri doğrulukla yollar elde edilebilmiştir.

Bir başka yeni çalışma da Maboudi ve diğ. (2018) [20] aittir. Geliştirdikleri sistemde karınca kolonisi optimizasyonundan ve bulanık mantık yönteminden faydalanarak %89 üzeri bir başarı elde etmeyi başarmışlardır.

Liu ve diğ. (2016) [21] ise kesme dönüşüm yöntemi ve iskeletlendirme algoritması kullanarak %90'a yakın bir başarım elde etmişlerdir.

Literatürdeki çalışmaların hemen hepsinde başlangıç aşamasında çeşitli önişlem metotlarının kullanıldığı görülmektedir[22]. Giriş kısmının sonunda bahsedilen uydu görüntülerindeki bazı istenmeyen gürültülerin yok edilerek bilgisayar öğrenmesi aşamalarının daha sağlıklı gerçekleştirilmesi için ön işlem metotları mutlaka kullanılmak zorundadır.

Bu çalışmada geliştirilen önişlem metodu literatürdeki diğer önişlem metotlarından farklı olarak zincir kod ve en uzun ortak alt dizi (longest common subsequence – LCS) yöntemlerini kullanarak söz konusu probleme çözüm oluşturmaya çalışmaktadır. Zincir kod yöntemi görüntü işlemede sıklıkla [23][24][25][26][27] kullanılan bir yöntemdir. Bu yöntem her bir yol kenar nesnesine ait zincir kodun çıkarılmasını sağlamaktadır. Bu kodların benzerliklerinin karşılaştırılması ve benzer olanlarının yol olduğunun belirlenmesi içinse LCS yöntemi tercih edilmiştir. Bu yöntem, sinyal işlemede iki sinyalin benzerliğinin bulunmasında ve metin işleme de de iki metnin benzerliğinin bulunmasında kullanılan bir yöntemdir[28][29][30][31][32]. Her iki yöntem de uygulaması basit ve hızlı çalışan yöntemler olduğundan tercih edilmişlerdir.

Bu önişlem kısmından sonra başarımın daha da iyileştirilmesi için bölge genişleme algoritması kullanılmıştır[33]. Bu yöntemin çalışması için gerekli olan başlangıç piksel değeri için ise üstteki yöntemlerin çıktısı verilmiştir. Böylelikle hem sonuçlar iyileştirilmiş hem de söz konusu yöntemin ihtiyaç duyduğu bilgi için kullanıcının değer girme zorunluluğunun da önüne geçilmiştir.

Sonuç olarak, literatürdeki benzer çalışmalara yakın başarım elde edebilen ve bunu da daha az adımda başarabilen bir sistem oluşturulmuştur.

2. GELİŞTİRİLEN SİSTEMİN YAPISI

Literatürdeki birçok çalışmadan da görüleceği üzere uydu görüntüleri üzerinde her hangi bir uygulama yapılacağı zaman öncelikle uygun bir ön işlem süreci yapılmalıdır. Aksi takdirde geliştirilmek istenen sistemin başarımı ciddi oranda düşecektir. Bu kapsamda geliştirilmek istenen önişlem yönteminin ana yapısı ise 8 adımdan oluşmaktadır.

1. Çok bantlı uydu görüntü dosyasının gri renk uzayına dönüştürülmesi
2. Yumuşatma
3. Morfolojik kapama
4. Canny [34] kenar bulma yönteminin uygulanması
5. İskelet çıkarma
6. Kenarlara ait zincir kodların (Chaincode) [35] çıkarılması
7. En Uzun Ortak Alt Dizi (LCS) [36][37] yöntemi ile kenarlara ait zincir kodların benzerliklerinin çıkarılması
8. Yol piksellerine ait dinamik gri renk eşiği bulunarak bu değerin altındaki piksellerin temizlenmesi

Yöntemin başarımının test edilmesi için de Das ve diğ. (2011) [12]'de ki yayınlarındaki açık kaynak olarak sunulan uydu görüntüleri kullanılmıştır. Bahsi geçen yayında bu görüntülerin http://wikimapia.org sitesinden alındığı ve 512x512 büyüklükte 1-m/piksel çözünürlüklü çok bantlı (multispectral, RGB) oldukları belirtilmiştir.

Aşağıda bu adımlarda belirtilen işlemler sırayla ayrıntılı bir biçimde açıklanmaktadır.

2.1 Çok Bantlı Uydu Görüntü Dosyasının Gri Renk Uzayına Dönüştürülmesi

İşlem zamanının kısaltılması ve kenar bulma algoritmasının çalışabilmesi için RGB renk uzayındaki uydu görüntüleri gri renk uzayına dönüştürülmüştür. Bu işlem için denklem (2.1)'deki ifade kullanılmıştır[38].

$$I_g = I_R * 0.2989 + I_G * 0.5870 + I_B * 0.1140 \quad \textbf{(2.1)}$$

Burada, I_R, I_G, ve I_B değerleri sırasıyla görüntüdeki kırmızı, yeşil ve mavi piksellerin değerini gösterir. I_g ise resmin gri renge dönüşmüş halini göstermektedir.

2.2 Yumuşatma

Her hangi bir görüntü içerisinde var olması istenmeyen piksellere gürültü (noise) denir[39]. Bu gürültü pikselleri, görüntünün alındığı kameradan, görüntünün iletildiği ortamdan veya çevresel faktörlerden kaynaklanmış olabilir. Hatta gerçekte bu tarz bir gürültüde olmayıp sadece yapılacak çalışmada var olması istenmeyen her hangi nesneye ait pikseller de olabilir. Literatürde, görüntü işleme çalışmalarında yumuşatma işlemi için lineer filtreler (ortalama filtreleme, gauss filtreleme) ve lineer olmayan filtreler (medyan filtreleme, k-en yakın komşu filtreleme) bulunmaktadır[39][40].

Bu çalışmada, basitliği ve hızlı çalışmasından dolayı çok tercih edilen bir yöntem olan ortalama filtreleme yöntemi kullanılmıştır. Bu yöntemde, m x n boyutlarında bir çekirdek (kernel) matris (bu çalışmada 3x3 boyutlarında tercih edilmiştir) resmin tüm pikselleri üzerinde soldan sağa ve yukarıdan aşağıya doğru birer birer gezdirilir. Her adımda, matrisin altında kalan alandaki piksellerin gri renk değerlerinin ortalaması hesaplanır ve ortadaki piksele değer olarak atanır. Kenar noktaları için konvolüsyon yapılırken ise görüntünün sınırını aşan noktalarla işlem yapılacağı zaman en yakında bulunan piksel dikkate alınır. Böylece f resmi g filtresi ile konvolüsyon işlemine tabi tutularak filtrelenmiş olur.

Konvolüsyon işlemi sonunda görüntüde, çekirdek pencerenin büyüklüğü ile doğru orantılı bir bulanıklık oluşacak ve bu da bazı gürültülerin önüne geçilebilmesini sağlayacaktır. Konvolüsyon işlemi için denklem (2.2)'deki gibi ifade kullanılmıştır[35]. Burada f filtrelenecek resmi, g filtreyi ve * operatörü de

konvolüsyon işlemini göstermektedir. Denklemdeki h ise konvolüsyon işleminin sonucunda yumuşatılmış resmi ifade etmektedir.

$$h(h,x) = f(x,y) * g(x,y) \quad \textbf{(2.2)}$$

Bu çalışmada filtre olarak [1/9, 1/9, 1/9; 1/9, 1/9, 1/9; 1/9, 1/9, 1/9] matrisi kullanılmıştır. Şekil 2.1' de örnek bir f görüntüsü için filtre uygulanmış ve sonuç görüntü ile değerleri gösterilmiştir.

f			*	g			=	f * g		
68	89	77		1/9	1/9	1/9		91	92	92
125	121	111		1/9	1/9	1/9		102	97	92
119	80	79		1/9	1/9	1/9		112	101	91

Şekil 2.1 : Konvolüsyon uygulaması.

Örneğin ilk piksel [f(1,1) => 68] için konvolüsyon işlemi yapılırken görüntü sınırı dışındaki noktalar için en yakın piksel değeri verilir ve işlem şu şekilde olur

f(i-1, j-1)	f(i, j-1)	f(i+1, j-1)
f(i-1, j)	f(i, j)	f(i+1, j)
f(i-1, j+1)	f(i, j+1)	f(i+1, j+1)

68	68	89
68	68	89
125	125	121

68 * 1/9	68 * 1/9	89 * 1/9
68 * 1/9	68 * 1/9	89 * 1/9
125 * 1/9	125 * 1/9	121 * 1/9

Şekil 2.2 : Konvolüsyon hesaplaması.

Hücre içlerindeki işlemler yapılıp değerler toplandığında f * g matrisinin ilk hücresi olan 91 sayısı elde edilmiş olur.

2.3 Morfolojik Kapama

Görüntü işlemede morfolojik kapama işlemi önce genişleme sonra aşınma diye de adlandırılır. Burada amaç, sonraki adımda yapılacak olan kenar bulma işleminde daha az gürültüye sahip bir görüntü sunarak başarımı arttırabilmektir. Morfolojik genişleme işlemi bir küme işlemi olup denklem (2.3)'teki gibi ifade edilmiştir.[41].

$$A \oplus B = \{z|(\widehat{B})_z \cap A \neq \emptyset\} \quad \textbf{(2.3)}$$

Burada, A değeri işlem yapılmak istenen resmi, B ise genişleme işleminde kullanılacak yapısal elementi gösterir. Bu elementin merkezi A resmindeki bir pikselin üzerine getirildiğinde en az bir ya da daha fazla piksel altta kalıyorsa bu piksel yeni oluşacak görüntüye dahil edilir. İşlem A resmindeki tüm pikseller için gerçekleştirilir. Morfolojik aşınma işlemi de bir küme işlemi olup denklem (2.4)'teki gibi ifade edilmiştir[41]

$$A \ominus B = \{z|(B)_z \cap A^c \neq \emptyset\} \quad \textbf{(2.4)}$$

Burada, yine A değeri işlem yapılmak istenen resmi, B ise aşınma işleminde kullanılacak yapısal elementi gösterir. Bu elementin merkezi A resmindeki bir pikselin üzerine getirildiğinde, elementin her hangi bir pikseli A'nın tümleyenine yani arka plana dokunuyorsa A görüntüsündeki o piksel yeni görüntüde olmayacaktır. Morfolojik kapama işlemi de kısaca denklem (2.5)'teki gibi yazılabilir[41]

$$A \cdot B = (A \oplus B) \ominus B \quad \textbf{(2.5)}$$

Bu iki yöntem sayesinde gri görüntü renk geçişleri dahada yumuşamış olur. Şekil 2.3'te orijinal görüntünün gri hali, 2.4'te yumuşatma uygulanmış hali ve 2.5'te de morfolojik kapama yapılmış hali verilmiştir.

Şekil 2.3 : Gri görüntü.

Şekil 2.4 : Yumuşatılmış gri görüntü.

Şekil 2.5 : Morfolojik kapama sonrası gri görüntü.

2.4 Kenar Bulma

Yol bulma işlemlerinde kenar bilgisi en önemli verilerden biridir. Kenar, görüntüde belirli bir değerin üzerinde değişimin olduğu yerdir diye tanımlanabilir[42]. Bu değişim, birinci derece türevin $(\nabla f = f')$ yerel maksimumu bulunarak elde edilebilir.

Literatürde birçok kenar bulma algoritması mevcut olmakla birlikte bu çalışmada tercihimiz en başarılı sonuçlar veren Canny kenar bulma algoritması olmuştur. Çünkü eldeki dört adet uydu görüntüsü üzerinde diğer iki kenar bulma algoritması olan

Prewitt ve Sobel ile deneme yapıldığında başarım değerlerinin ortalamada %5'e yakın düştüğü gözlemlenmiştir.

Bu algoritmayı diğerlerinden farklı kılan yönleri ise şunlardır[41];

- Düşük hata oranı,
- Gerçek kenar noktasına en yakın noktada kenar tespiti yapılabilmesi,
- Her bir gerçek kenar noktası için yalnızca tek bir kenar noktası bulabilme.

Canny kenar bulma algoritması aşağıda verilen 5 adımda çalışmaktadır:

1. Yumuşatma (gürültü giderme) : Birçok görüntü dosyası içerisinde gerçek görüntüyle ilgili olmayan ve kamera, ışık ya da benzeri sebeplerden kaynaklı istenmeyen pikseller yani gürültüler bulunabilmektedir. Bu gürültüler bazı kenarların hatalı olarak bulunmasına neden olabildiğinden öncelikle yumuşatma işlemi gerçekleştirilir.

   Canny algoritması Gauss filtresi kullanarak yumuşatma işlemini gerçekleştirir. Bu filtre denklem (2.6)'da verilmiştir.

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2}) \quad \textbf{(2.6)}$$

   Burada σ işareti standart sapmayı gösterir.

2. Gradyan büyüklüğünün ve yönünün bulunması: Görüntünün, Gauss fonksiyonun x ve y doğrultusunda birinci derece türevi ile konvolüsyon tabi tutulması işlemidir. Bu işlem denklem (2.7) ve (2.8)'de belirtilmiştir.

$$f_x(x,y) = f(x,y) * (\frac{-x}{\sigma^2}) exp(-\frac{x^2+y^2}{2\sigma^2}) \quad \textbf{(2.7)}$$

$$f_y(x,y) = f(x,y) * (\frac{-y}{\sigma^2}) exp(-\frac{x^2+y^2}{2\sigma^2}) \quad \textbf{(2.8)}$$

   Ardından büyüklük değeri için denklem (2.9) ve yön için de denklem (2.10) kullanılır.

$$|f(x,y)| = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} \quad \textbf{(2.9)}$$

$$\theta(x,y) = arctg(\frac{f_y(x,y)}{f_x(x,y)}) \quad \textbf{(2.10)}$$

3. Gradyan vektörü yönünde zayıf kenar noktalarını baskılama (non-maximum suppression): Bu işlem için önceki adımda bulunan θ açısı 4 ana bölgeye indirgenir. Bu bölgeler 8 temel yönü içermektedir. Diğer bir deyişle eldeki açı değeri en yakın 45 dereceye yuvarlanarak ait olduğu bölge belirlenmiş olur. Daha sonra geçerli pikselin değeri o bölgenin negatif yönündeki piksel ile ve pozitif yönündeki pikselle karşılaştırılır. Eğer en büyük değer geçerli piksele ait ise değişim yapılmaz aksi halde geçerli piksel silinir. Örneğin yön değeri kuzey ise, geçerli pikselin bir üstündeki ile bir altındaki piksel değerleri kontrol edilecektir.
4. İkili eşik değeri yöntemi ile kenarların birleştirilmesi: Canny kenar bulma yöntemi bir tane düşük ($t_{\text{düşük}}$) bir tane de yüksek ($t_{yüksek}$) eşik değeri kullanmaktadır (çoğunlukla $t_{yüksek} = 2 * t_{\text{düşük}}$ tercih edilir). Gradyan vektörünün büyüklük değeri yüksek eşik değerinden fazla olan pikseller güçlü (yani kesin kenar pikseli) olarak kabul edilirken gradyan vektörünün büyüklük değeri düşük eşik değerinden daha küçük olan pikseller de zayıf (yani kesin kenar olmayan piksel) olarak kabul edilir. Gradyan vektörünün büyüklük değeri iki eşik değeri arasında kalan piksellerin her biri için komşusu olan 8 yöne de bakılır. En az bir tane güçlü piksel bulunursa geçerli piksel de güçlü kabul edilir yoksa silinir. Yani kısaca güçlü piksellerin çevresinde kalanlar da güçlü olurlarken diğerleri de silinirler.

2.5 İskelet Çıkarma

Zincir kod işlemine başlamadan önce bir önceki adımda bulunan kenarların iskeletlerinin çıkarılması ve dallanma noktalarının koparılması gerekmektedir. Bu işlem yapılmadan kenarlara ait zincir kodlar çıkarılacak olursa elde edilen kodlar yanlış olacak ve istenen başarım elde edilemeyecektir.

İskelet çıkarma işlemi, bir A görüntüsünün aşınma ve yayılma işlemi olarak ifade edilebilir ve denklem (2.11)'deki gibi gösterilebilir[43].

$$S(A) = \bigcup_{k=0}^{K} [(A \ominus kB) - (A \ominus kB) \circ B] \quad \textbf{(2.11)}$$

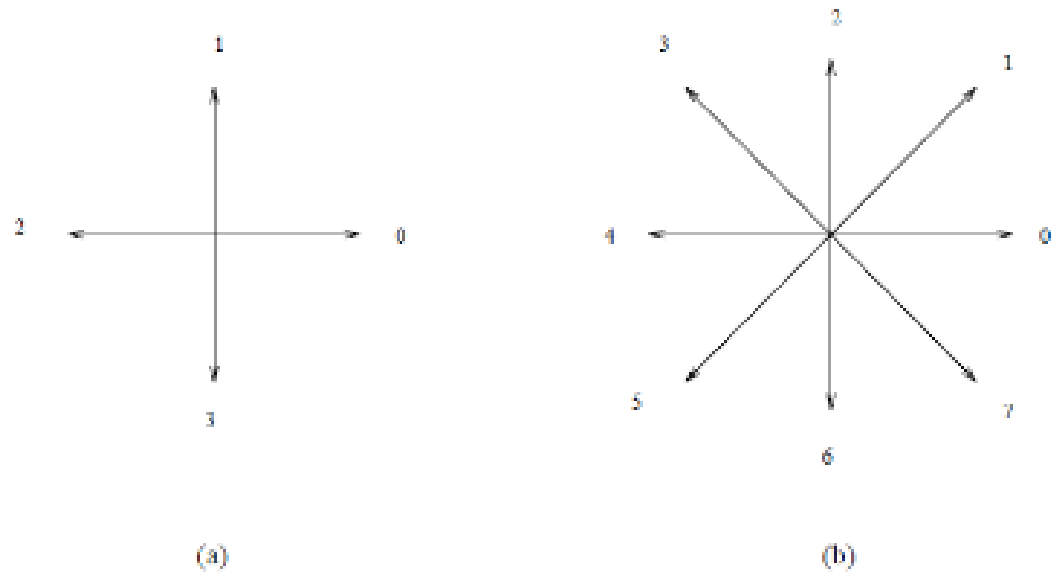
Burada B yapısal elementi gösterirken, $A \ominus kB$ işlemi ise A görüntüsünün k defa aşınmaya tabi tutulacağını ifade etmektedir. K ise A görüntüsünün boş küme olmadan önceki son adımını göstermektedir. Bu ifade denklem (2.12)'de gösterilmiştir.

$$K = max\{k | (A \ominus kB) \neq \emptyset\} \quad \textbf{(2.12)}$$

Bu işlemden sonra çatallanmış olan kenarlar dallanmanın olduğu piksel renk değeri silinerek yok edilir ve görüntüde sadece tekil kenarların kalması sağlanır.

2.6 Zincir Kod

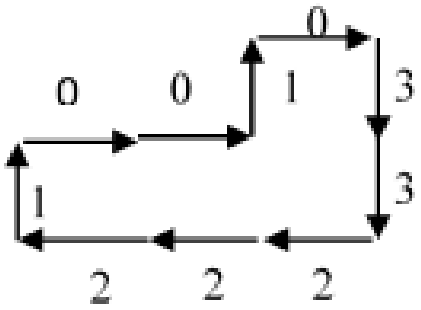
Zincir kod (chaincode) yöntemi, harflerin [23], nesnelerin [24][25][26], ya da sinyallerin[27] incelenmesinde kullanılan yöntemlerden biridir. Bu yöntemde, nesnenin sırayla kenarlarındaki tüm piksellerde dolaşılır. Her bir pikselin bir öncekine göre yön durumuna ait kod kaydedilir. Burada kod değerinin alabileceği değerler Şekil 2.6'daki yönlerden hangisi kullanılmış ise ona göre değişmektedir.



Şekil 2.6 : Zincir kod yönleri ((a)4 yön, (b)8 yön).

Bu çalışmada daha hassas ve doğru bir kod dizisi sunduğu için 8 yön kullanılması tercih edilmiştir.

Şekil 2.7'de bir nesne için zincir kod örneği verilmiştir. Bu örnek için oluşturulabilecek zincir kodlardan biri {0, 0, 1, 0, 3, 3, 2, 2, 2, 1} olacaktır. Bu zincir kod bilgisi kullanılarak aynı şekil yeniden elde edilebilir.



Şekil 2.7 : Nesnenin 4 yönlü zincir kodu.

Birden fazla zincir kod oluşturulabilmesinin nedeni ise başlangıç noktasının istenildiği yerden seçilebilmesindendir. Bundan dolayı zincir kod yöntemi uygulanırken seçilen başlangıç noktasından elde edilen kod sola doğru bir adım ötelenerek her iki kodun fark değerleri alınır. Böylece başlangıç noktasından bağımsız bir zincir kod elde edilmiş olunur. Şekil 2.7'de örnek için ötelenen kod {0, 1, 0, 3, 3, 2, 2, 2, 1, 0} ve fark zincir kodu ise {0, 1, -1, 3, 0, -1, 0, 0, -1, -1} olacaktır. Negatif değerlerin düzeltilmesi için kullanılan yön değeri üzerlerine eklenir. Örneğin 8 yön kullanıldıysa fark zincir kodu {0,1,7,3,0,7,0,0,7,7} haline gelmektedir.

Bu çalışmada yol kenarları için zincir kod çıkarılacağı için nesnenin çevresinde dönmek istediğimiz sonucu bize vermeyecektir. Çünkü yol kenarları kapalı nesneler değil açık uçlu çizgilerden oluşmaktadır. Bu yüzden bir yol kenarının herhangi bir ucundaki ilk pikselden başlanıp son piksele kadarki zincir kod çıkarılır. Yukarıda açıklanan başlangıç noktası sorunundan dolayı aynı işlem diğer uçtan başlanılarak da yapılır ve sonuçta n tane pikselden oluşan bir yol kenarı için n - 1 uzunluğunda iki adet zincir kod elde edilir.

2.7 En Uzun Ortak Alt Dizi (Longest Common Subsequence – LCS)

Zincir kod yöntemi ile yol kenarlarına ait kodları çıkardıktan sonra sıra bunların karşılaştırmasına gelmektedir. Fakat zincir kod yönteminin gürültüye ve nesnenin büyüklüğüne son derece hassas oluşundan dolayı basit bir karşılaştırma sonucu büyük

oranda benzeyen iki yol kenarı benzeşmiyor gibi gözükecektir. Bundan dolayı bu çalışmada karşılaştırma için LCS yönteminden faydalanılmıştır.

LCS yöntemi, yazım düzeltme, dosya karşılaştırma, genetik mühendisliği ve sinyal işlemede kullanılan bir yöntemdir[28][29][30][31][32]. Burada hedef, verilen iki kümeni her ikisine de soldan sağa doğru bakıldığında bulunabilecek en uzun boyutlu alt kümenin bulunmasıdır. Benzeşmenin aynı indislerde olması da gerekmez. Örneğin "abcdefg" ve "klaxcyg"şeklindeki iki kümeden "a", "c", "g", "ac", "ag" ve "acg"gibi alt kümeler çıkarılabilir. Ama burada aranan en uzun benzeşen küme olduğundan sonuç "acg" olacaktır.

Yönteme ait algoritma literatürde genellikle özyinelemeli olarak verilmektedir[44]. Denklem (2.14) bu tarz bir gösterimi ifade etmektedir. Burada karmaşıklık değeri ise $O(2^n)$ olmaktadır (n kümenin boyutudur)

$$LCS(X_i, Y_i) = \begin{cases} eğer\ i = 0\ veya\ j = 0 & \emptyset \\ eğer\ X_i = Y_i & LCS(X_{i-1}, Y_{i-1}) + 1 \\ eğer\ X_i \neq Y_i & max(LCS(X_i, Y_{i-1}), LCS(X_{i-1}, Y_i)) \end{cases} \quad \textbf{(2.14)}$$

Denklem (2.14) sözdekod(pseudocode) olarak yazılacak olursa Şekil 2.8'deki gibi yazılabilir.

```
m <- uzunluk(X)
n <- uzunluk(Y)
L <- sıfır(n+1,m+1)
b <- sıfır(n+1,m+1)
b(:,1) <- 1
b(1,:) <- 2
for i <- 2 to n+1
   for j <- 2 to m+1
        if x(i-1) = y(j-1)
           L(i,j) <- L(i-1,j-1) + 1
           b(i,j) <- 3  //sol üst
        else
           L(i,j) <- L(i-1,j-1)
        end if
        if L(i-1,j) >= L(i,j)
           L(i,j) <- L(i-1,j)
           b(i,j) <- 1  // yukarı
        end if
        if L(i,j-1) >= L(i,j)
           L(i,j) <- L(i,j-1)
           b(i,j) <- 2  // sol
        end if
   end for
end for
return L, b
```

Şekil 2.8 : LCS yöntemi için sözdekod(pseudocode).

Şekil 2.9'da da böyle bir kodun çalıştırılması sonucu X = {1, 3, 2, 4, 5, 1} ve Y = {1, 2, 3, 4, 1} zincir kodlarının benzeşim sonucu oluşan LCS(X, Y) = {1, 2, 4, 1} gösterilmiştir.

X	Y ->	1	2	3	4	1
	0	0	0	0	0	0
1	0	1	1	1	1	1
3	0	1	1	2	2	2
2	0	1	2	2	2	2
4	0	1	2	2	3	3
5	0	1	2	2	3	3
1	0	1	2	2	3	4

Şekil 2.9 : LCS yöntemi uygulama örneği.

Bu çalışmada LCS yönteminin yol kenarlarının benzerliğinin tespitinde kullanılabilmesi için öncelikle her bir kenar için kullanıcının istediği yol büyüklüğünde bir çerçeve oluşturulmuştur. Burada verilecek en az ve en çok yol genişliği değerleri geliştirilen sistemin kullanıcıdan istediği tek parametrelerdir. Referans alınan her bir kenar için oluşturulan bu çerçeveler içerisindeki kenarların zincir kodları ile referans kenarların zincir kodları karşılaştırılır. Referans kenar ile en çok benzeşen kenarın bir yol oluşturduğu belirlenir ve benzeşmenin başladığı ve bittiği alan yol olarak işaretlenir. Ancak burada dikkat edilmesi gereken nokta benzeşmenin karşılaştırılan kenarların zıt uçlarında olmaması gerektiğidir. Diğer bir deyişle bir kenar nesnesinin sol ucu diğer kenar nesnesinin sağ ucuna benziyor olabilir ama bu onların yol kenarı olabileceklerini göstermez. Çünkü yol kenarlarında benzeşmenin yakın bir bölüt içinde olması beklenir. Geliştirilen sistemde bu sorunun çözümü için öncelikle benzeşen iki kenar nesnesinin her pikselinin diğerine olan Manhattan mesafesi ölçülür. Bu hesaplama için denklem (2.15) kullanılmıştır.

$$ManhattanMesafesi = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad \textbf{(2.15)}$$

Ardından benzeşen bir piksel bulunduğunda 20 piksel öncesi ve sonrasında başka benzeşen piksel olup olmadığına da bakılır. Dolayısıyla, sadece 20 piksellik alan içinde benzeşme varsa kabul edilir. Daha uzaktaki bir pikselle benzeşiyorsa bu benzeşme geçersiz kabul edilir.

LCS yöntemi sonucunda kenar nesnelerinin zincir kodu benzeşen pikselleri arasındaki alanlar yol olarak işaretlenmektedir.

2.8 Dinamik Renk

Önceki adımda kenarlara ait zincir kodlarının benzeşmesinden dolayı yol olabilecek alanlar tespit edilmiştir. Bu adımda ise ilk olarak söz konusu alanlara ait piksellerin ortalama gri renk değerleri hesaplanmaktadır. Test için kullanılan görüntülerin hepsinde yolların gri renk değeri genel olarak koyu renkten ziyade daha çok açık renge doğru kaydığından ikinci adım olarak da görüntüde bu değerden düşük olan pikseller silinmektedir. Bu işlemin sabit ve elle verilen bir sayı ile değil de sistemin bu şekilde belirlediği o resme ait dinamik gri renk eşik değeri ile yapılmış olması söz konusu işleme ait oluşabilecek hataları en aza indirgemektedir.

Böylece kenar bulma algoritmasının ormanlık alanlarda ya da açık arazilerde tespit ettiği ve zincir kodları da benzeşen hatalı alanların temizlenmesi sağlanmış olmaktadır.

2.9 Bölge Büyütme Algoritması

Önceki adımların sonucunun daha da başarılı hale getirmek için burada ek bir adım daha bulunmaktadır. Bu adımda bir tür görüntü bölütleme yöntemi olan bölge büyütme algoritması kullanılmıştır.

Görüntü bölütleme, bilgisayarlı görme sahasındaki önemli konulardan biridir[45]. Tıbbı görüntü işleme, görüntüde nesne bulma, otomatik trafik kontrol sistemleri vb. birçok farklı sahadaki görüntülerde kullanılan yöntemler geliştirilmiştir. Bu yöntemlerin genel amacı görüntüde belirli karakteristik özellikleri ortak olan (homojen) alanlar oluşturmaktır[46]. Burada ortak özellikler (homojenlik), renk, şekil ve yüzey kaplaması gibi özelliklerden oluşabilir[47].

Literatürde bölütleme yöntemleri şu dört kategoride incelenmektedir[48].

i. Kenar bulma ile bölütleme
ii. Eşik değer ile bölütleme
iii. Bölge tabanlı bölütleme
iv. Özellik tabanlı bölütleme

Bölge büyüme (region growing) algoritması da Adams ve Bischof (1994) [33] tarafında geliştirilmiş piksel bazında yarı otomatik bir bölge tabanlı görüntü bölütleme yöntemidir.

Kolay, hızlı ve başarımı yüksek bir yöntem olduğundan birçok farklı sahadaki çalışmada[49][50][51] kullanılmıştır. Ancak yöntemin başarımın etki eden en önemli faktör başlangıç besleme (seed) değeridir. Kötü verilecek bir besleme değeri sonuçları olumsuz etkileyecektir. Bu değerin elle verilmesini engellemek için de bazı otomatize etme çalışmaları[45][47] yapılmıştır

Bu çalışmada ise bölge büyütme algoritmasının ihtiyaç duyduğu ilk besleme değeri elle değil zincir kod ve LCS yöntemlerinin sonucundan verilmektedir. Böylece hem otomatize edilmiş bir bölge büyütme yapılmış hem de önceki bölümlerde elde edilen sonuçlar daha da iyileştirilmiştir.

Burada, önceki adımda yol olarak belirlenen piksellerin ortalama gri renk değerine en yakın ilk piksel seçilerek bölge büyütme işlemi uygulanmaktadır. İşlem sonunda yeniden önceki adımdan gelen yol piksellerine bakılır ve henüz değerlendirilmemiş piksel varsa yine ortalama gri renk değerine en yakın olandan başlanarak bölge büyütme işlemi devam ettirilir. Tüm pikseller belirli bir bölgeye dahil olana kadar işleme devam edilir.

Bölge büyütme işlemi esnasında seçili pikselin dört yönüne yani üst, alt, sol ve sağ komşu piksellere bakılmaktadır. Başlangıçta ortalama gri renk değeri geçerli pikselin rengi kabul edilir. Komşu piksellerden bu değere en yakın gri değer içereni bölgenin elemanı olarak belirlenir. Eğer bu piksel bölgeye katıldığında bölgenin ortalama gri renk değerindeki değişim %10'dan az ise yeni eklenen piksel başlangıç olarak alınıp aynı işlem adımları oradan devam ettirilir ve %10'dan fazla fark oluştuğunda bölge oluştuğundan işlem sonlandırılmaktadır.

3. SONUÇLAR

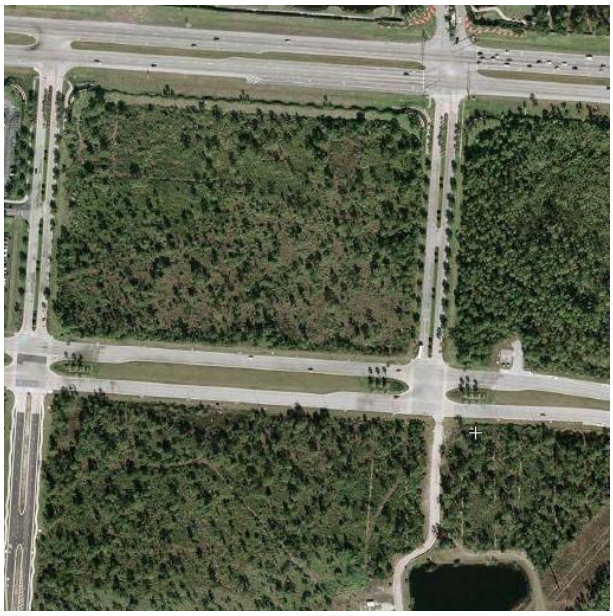
Bu bölümde geliştirilen sistemin örnek görüntüler üzerinde çalışması sunulmaktadır. Bu görüntüler, Das ve diğ. (2011) [12]'de ki yayınlarında kullanmış oldukları şehir dışı uydu görüntülerinin bazılarını içermektedir. Görüntüler, 512x512 RGB ve JPG formatında olup uygulama Matlab 2016 programı ile gerçekleştirilmiştir. Algoritmanın tamamlanması ortalama 120 sn. sürmektedir.

3.1 Yöntem Çıktıları

Şekil 3.1 – 3.20 arasında geliştirilen sistemin beş ana adımına ait çıktılar sunulmuştur. Dört farklı şehir dışı alan içeren uydu görüntüsü için verilen bu çıktıların ilki uydu görüntüsünün orijinal halini göstermektedir. İkinci çıktı Canny kenar bulma algoritması sonucunda bulunan kenarları kırmızı ile çizerek göstermektedir. Üçüncü çıktı LCS yöntemi sonucunda benzer olarak tespit edilen yol bloklarını kırmızı ile boyayarak göstermektedir. Dördüncü çıktı renk eşik değeri kontrolü sonrası elde kalan pikselleri göstermektedir. Beşinci ve son çıktı da bölge büyütme algoritması ile el çizimine ait karşılaştırmayı göstermektedir.

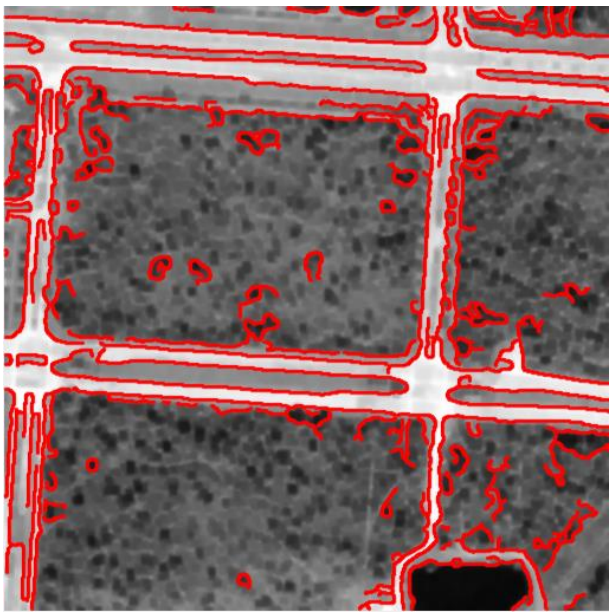
Bu karşılaştırmada yeşil renkli alanlar gerçekte yol olan ve geliştirilen sistemin de yol olarak bulduğu pikselleri, kırmızı alanlar gerçekte yol olduğu halde yol olarak tespit edilemeyen pikselleri ve sarı alanlar da gerçekte yol olmadığı halde sistemin yol olarak tespit ettiği pikselleri göstermektedir.

Karşılaştırma matrisi ise Çizelge 3.2 ve 3.3'te verilmiştir.



Şekil 3.1 : Görüntü 1 : Orijinal.

Şekil 3.1'de geliştirilen yöntemin uygulanması için kullanılacak olan ilk görüntü dosyasının orijinal hali verilmiştir. Burada kara yolları ve ağaçlık alanlar görülebilmektedir. Fakat görüntünün sağ üst kısmındaki araçlar ve kara yolunun orta kısmındaki refüj alanları yolların tespitini zorlaştıran etmenlerden biridir. Bu görüntüde yol genişliği parametresi en az 20 ve en çok 30 olarak alınmıştır.



Şekil 3.2 : Görüntü 1 : Canny kenar bulma.

Şekil 3.2'deki görüntü, öncelikle geliştirilen yöntemin ilk üç ana adımının sonucu olan bir görüntüdür. Burada orijinal görüntü önce gri renk uzayına dönüştürülmüş ardından

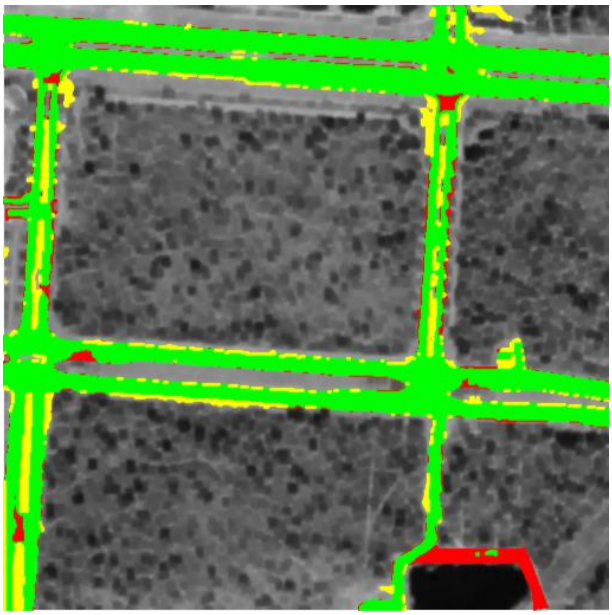
da önce yumuşatma sonra da morfolojik kapama işlemi uygulanmıştır. Ardından Canny kenar bulma algoritması eldeki görüntü üzerinde çalıştırılmış ve algoritmanın kenar olarak tespit ettiği alanlar kırmızı ile çizilmek suretiyle oluşturulmuştur (Şekil 3.2).

Şekil 3.3 : Görüntü 1 : LCS.

Şekil 3.3'de LCS yöntemi sonucu muhtemel olarak bir yola ait olabileceği belirlenen pikseller kırmızı ile boyanarak gösterilmektedir. Bu işlem için öncelikle zincir kod yöntemi Şekil 3.2'ye uygulanarak her bir kenar için zincir kodlar elde edilmiştir. Daha sonra her bir kenar nesnesi üzerine başta belirtilmiş olan yol genişliği boyutunda bir çerçeve yerleştirilmiştir. Her çerçeve içindeki kenarların zincir kodlarının, üzerine çerçeve yerleştirilen geçerli kenarın zincir kodu ile olan benzerliği LCS yöntemiyle kontrol edilmiştir. Sonuç olarak geçerli kenarla en çok benzeşen kenarın benzeştikleri pikseller ve arasında kalan alan kırmızı olarak gösterilebilmiştir.

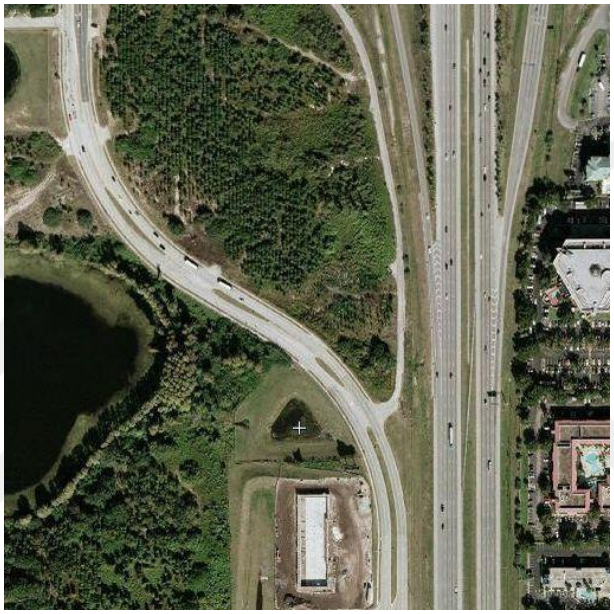
Şekil 3.4 : Görüntü 1 : Renk eşik değeri sonrası.

Şekil 3.4'te ise dinamik renk eşiği uygulanarak elde edilen yol pikselleri gösterilmektedir. Burada ormanlık, tarla vb. alanlarda aslında yol olmayan ama zincir kodları benzeşen kenarlar ve aralarındaki alanlar silinmektedir. Dinamik renk değeri bilgisi ise LCS yöntemi sonucu muhtemel yol olarak belirlenen piksellerin ortalama gri renk değeridir. Bu değerden küçük gri rengi olan pikseller silindiğinde Şekil 3.3'te ki görüntü Şekil 3.4'te ki hale gelmektedir.



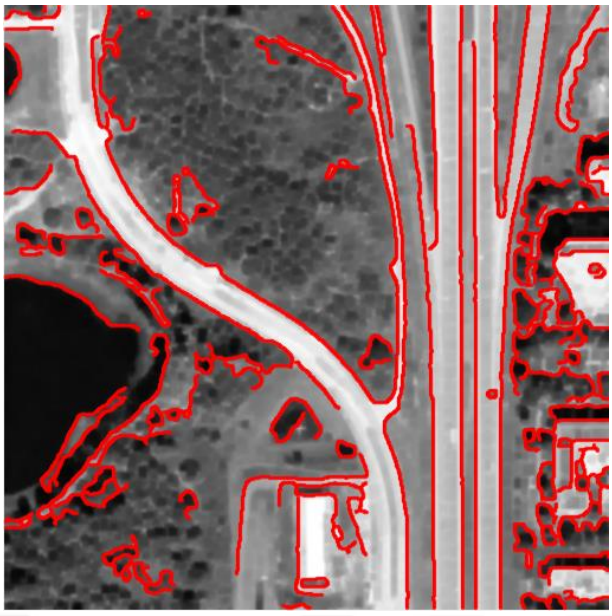
Şekil 3.5 : Görüntü 1 : Bölge büyütme sonrası el çizim ile karşılaştırma.

Son olarak Şekil 3.4 üzerinde bölge büyütme algoritması uygulanmış ve Şekil 3.5 elde edilmiştir. Söz konusu algoritmanın ihtiyaç duyduğu besleme değeri Şekil 3.4'te kırmızı olarak işaretlenmiş piksellerdir. Diğer bir deyişle muhtemel yol olabilecek alanlardan bölge büyütme işlemi yapılmış ve Şekil 3.5 elde edilmiştir. Burada yeşil ile boyanan yerler gerçekte yol olan ve geliştirilen yöntemin de yol olarak bulduğu pikselleri göstermektedir. Kırmızı alanlar ise gerçekte yol olupta geliştirilen yöntem tarafından yol olarak bulunamayan pikselleri göstermektedir. Sarı alanlar da gerçekte yol olmayan ama yol olarak tespit edilen alanları göstermektedir.



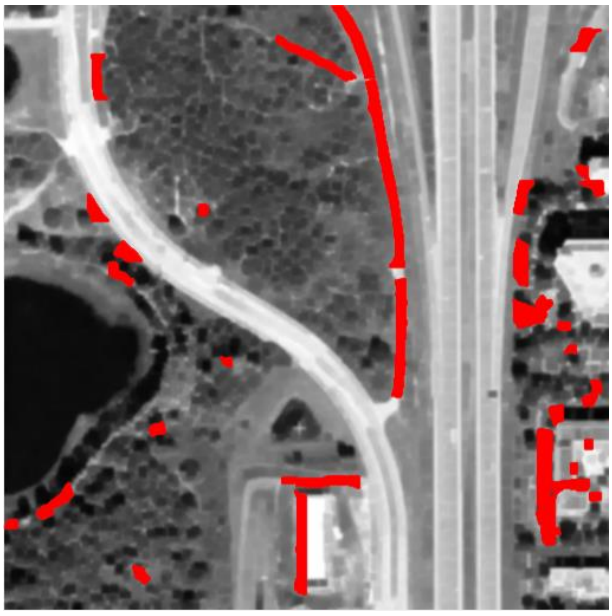
Şekil 3.6 : Görüntü 2 : Orijinal.

Şekil 3.6'da geliştirilen yöntemin uygulanması için kullanılacak olan ikinci görüntü dosyasının orijinal hali verilmiştir. Burada kara yolları, ağaçlık alanlar ve binalar görülebilmektedir. Bu görüntüde orta alt kısımdaki yapı hem şekil hem de renk olarak kara yoluna çok benzediğinden yöntemin başarımını olumsuz etkilemektedir. Bu görüntüde yol genişliği parametresi en az 5 ve en çok 15 olarak alınmıştır.



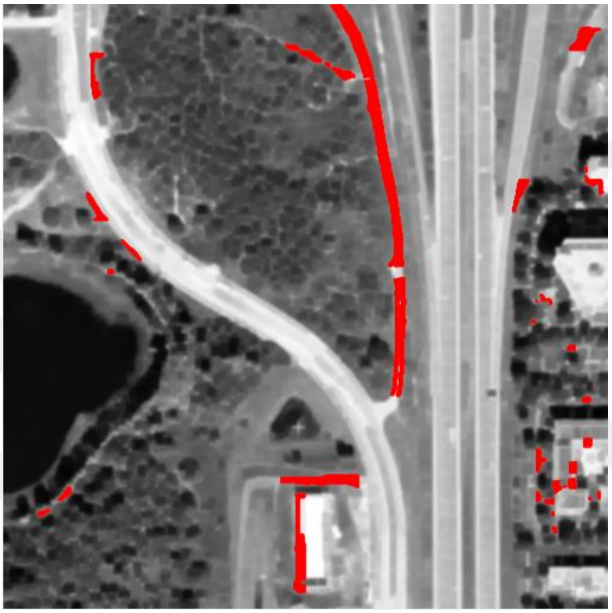
Şekil 3.7 : Görüntü 2 : Canny kenar bulma.

Şekil 3.7'deki görüntü, öncelikle geliştirilen yöntemin ilk üç ana adımının sonucunu göstermektedir. Burada orijinal görüntü önce gri renk uzayına dönüştürülmüş ardından da önce yumuşatma sonra da morfolojik kapama işlemi uygulanmıştır. Ardından Canny kenar bulma algoritması eldeki görüntü üzerinde çalıştırılmış ve algoritmanın kenar olarak tespit ettiği alanlar kırmızı ile çizilmek suretiyle Şekil 3.7 oluşturulmuştur.



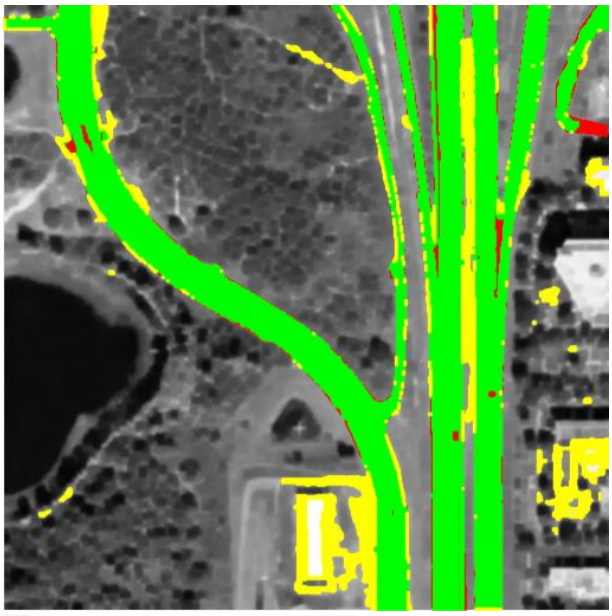
Şekil 3.8 : Görüntü 2 : LCS.

Şekil 3.8’de LCS yöntemi sonucu muhtemel olarak bir yola ait olabileceği belirlenen pikseller kırmızı ile boyanarak gösterilmektedir. Bu işlem için öncelikle zincir kod yöntemi Şekil 3.7’ye uygulanarak her bir kenar için zincir kodlar elde edilmiştir. Daha sonra her bir kenar nesnesi üzerine başta belirtilmiş olan yol genişliği boyutunda bir çerçeve yerleştirilmiştir. Her çerçeve içindeki kenarların zincir kodlarının, üzerine çerçeve yerleştirilen geçerli kenarın zincir kodu ile olan benzerliği LCS yöntemiyle kontrol edilmiştir. Sonuç olarak geçerli kenarla en çok benzeşen kenarın benzeştikleri pikseller ve arasında kalan alan kırmızı olarak gösterilebilmiştir.



Şekil 3.9 : Görüntü 2 : Renk eşik değeri sonrası.

Şekil 3.9’da ise dinamik renk eşiği uygulanarak elde edilen yol pikselleri gösterilmektedir. Burada ormanlık, tarla vb. alanlarda aslında yol olmayan ama zincir kodları benzeşen kenarlar ve aralarındaki alanlar silinmektedir. Dinamik renk değeri bilgisi ise LCS yöntemi sonucu muhtemel yol olarak belirlenen piksellerin ortalama gri renk değeridir. Bu değerden küçük gri rengi olan pikseller silindiğinde Şekil 3.8’de ki görüntü Şekil 3.9’da ki hale gelmektedir.

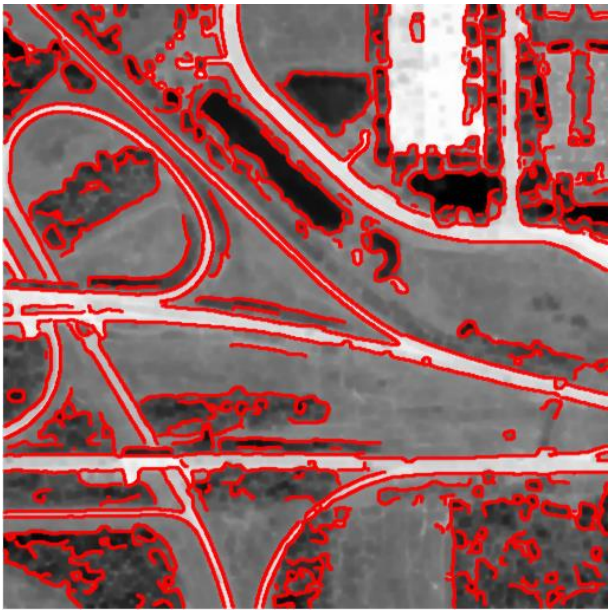


Şekil 3.10 : Görüntü 2 : Bölge büyütme sonrası el çizim ile karşılaştırma.

Son olarak Şekil 3.9 üzerinde bölge büyütme algoritması uygulanmış ve Şekil 3.10 elde edilmiştir. Söz konusu algoritmanın ihtiyaç duyduğu besleme değeri Şekil 3.9'da kırmızı olarak işaretlenmiş piksellerdir. Diğer bir deyişle muhtemel yol olabilecek alanlardan bölge büyütme işlemi ile elde edilmiştir (Şekil 3.10). Burada yeşil ile boyanan yerler gerçekte yol olan ve geliştirilen yöntemin de yol olarak bulduğu pikselleri göstermektedir. Kırmızı alanlar ise gerçekte yol olupta geliştirilen yöntem tarafından yol olarak bulunamayan pikselleri göstermektedir. Sarı alanlar da gerçekte yol olmayan ama yol olarak tespit edilen alanları göstermektedir.

Şekil 3.11 : Görüntü 3 : Orijinal.

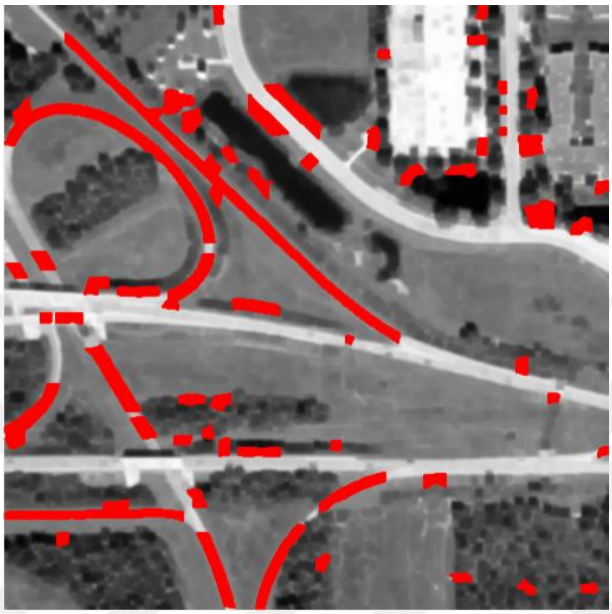
Şekil 3.11'de geliştirilen yöntemin uygulanması için kullanılacak olan üçüncü görüntü dosyasının orijinal hali verilmiştir. Burada kara yolları, ağaçlık alanlar ve binalar görülebilmektedir. Bu görüntüde üst sağ bölgedeki yapılar hem şekil hem de renk olarak kara yoluna çok benzediğinden yöntemin başarımını olumsuz etkilemektedir. Bu görüntüde yol genişliği parametresi en az 7 ve en çok 15 olarak alınmıştır.



Şekil 3.12 : Görüntü 3 : Canny kenar bulma.

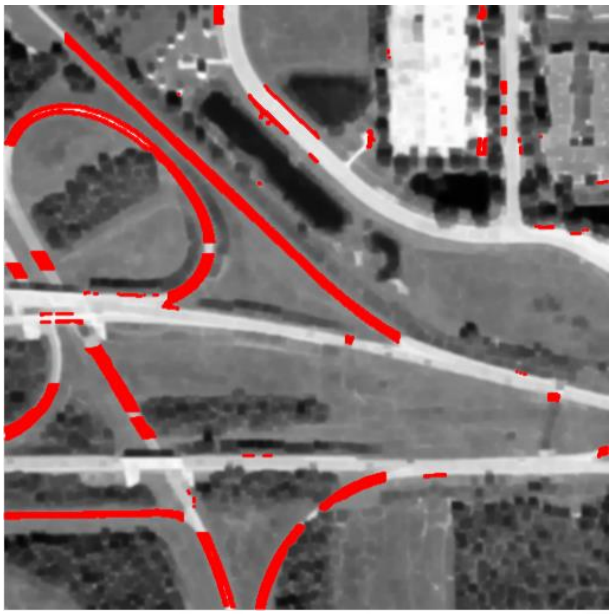
Şekil 3.12'deki görüntü, öncelikle geliştirilen yöntemin ilk üç ana adımının sonucunugöstermektedir. Burada orijinal görüntü önce gri renk uzayına

dönüştürülmüş ardından da önce yumuşatma sonra da morfolojik kapama işlemi uygulanmıştır. Ardından Canny kenar bulma algoritması eldeki görüntü üzerinde çalıştırılmış ve algoritmanın kenar olarak tespit ettiği alanlar kırmızı ile çizilmek suretiyle Şekil 3.12 oluşturulmuştur.



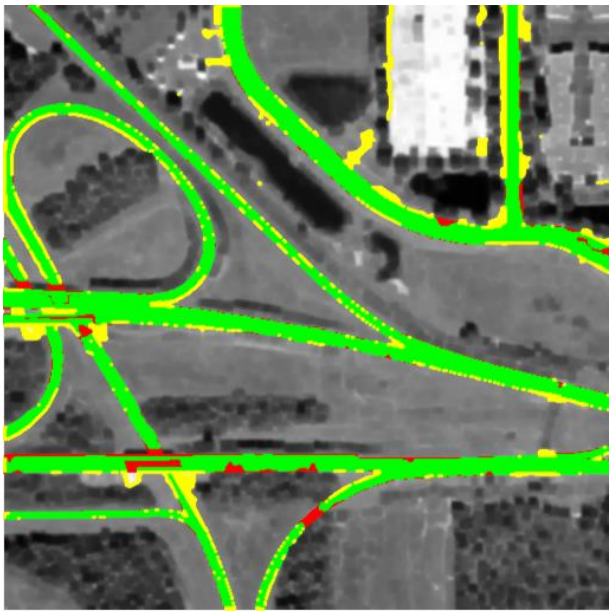
Şekil 3.13 : Görüntü 3 : LCS.

Şekil 3.13'de LCS yöntemi sonucu muhtemel olarak bir yola ait olabileceği belirlenen pikseller kırmızı ile boyanarak gösterilmektedir. Bu işlem için öncelikle zincir kod yöntemi Şekil 3.12'ye uygulanarak her bir kenar için zincir kodlar elde edilmiştir. Daha sonra her bir kenar nesnesi üzerine başta belirtilmiş olan yol genişliği boyutunda bir çerçeve yerleştirilmiştir. Her çerçeve içindeki kenarların zincir kodlarının, üzerine çerçeve yerleştirilen geçerli kenarın zincir kodu ile olan benzerliği LCS yöntemiyle kontrol edilmiştir. Sonuç olarak geçerli kenarla en çok benzeşen kenarın benzeştikleri pikseller ve arasında kalan alan kırmızı olarak gösterilebilmiştir.



Şekil 3.14 : Görüntü 3 : Renk eşik değeri sonrası.

Şekil 3.14'te ise dinamik renk eşiği uygulanarak elde edilen yol pikselleri gösterilmektedir. Burada ormanlık, tarla vb. alanlarda aslında yol olmayan ama zincir kodları benzeşen kenarlar ve aralarındaki alanlar silinmektedir. Dinamik renk değeri bilgisi ise LCS yöntemi sonucu muhtemel yol olarak belirlenen piksellerin ortalama gri renk değeridir. Bu değerden küçük gri rengi olan pikseller silindiğinde Şekil 3.13'te ki görüntü Şekil 3.14'te ki hale gelmektedir.

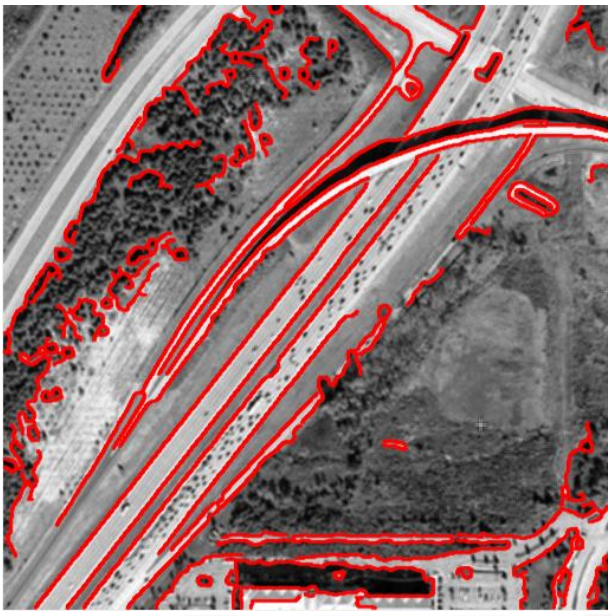


Şekil 3.15 : Görüntü 3 : Bölge büyütme sonrası el çizim ile karşılaştırma.

Son olarak Şekil 3.14 üzerinde bölge büyütme algoritması uygulanmış ve Şekil 3.15 elde edilmiştir. Söz konusu algoritmanın ihtiyaç duyduğu besleme değeri Şekil 3.14'te kırmızı olarak işaretlenmiş piksellerdir. Diğer bir deyişle muhtemel yol olabilecek alanlardan bölge büyütme işlemi yapılmış ve Şekil 3.15 elde edilmiştir. Burada yeşil ile boyanan yerler gerçekte yol olan ve geliştirilen yöntemin de yol olarak bulduğu pikselleri göstermektedir. Kırmızı alanlar ise gerçekte yol olupta geliştirilen yöntem tarafından yol olarak bulunamayan pikselleri göstermektedir. Sarı alanlar da gerçekte yol olmayan ama yol olarak tespit edilen alanları göstermektedir.

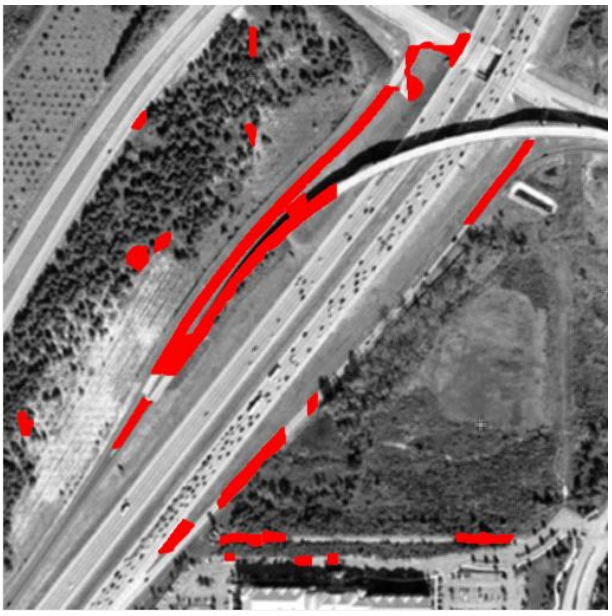
Şekil 3.16 : Görüntü 4 : Orijinal.

Şekil 3.16'da geliştirilen yöntemin uygulanması için kullanılacak olan dördüncü ve son görüntü dosyasının orijinal hali verilmiştir. Burada kara yolları, ağaçlık alanlar ve binalar görülebilmektedir. Bu görüntüde orta alt bölgedeki yapı ve otopark alanı hem şekil hem de renk olarak kara yoluna çok benzediğinden yöntemin başarımını olumsuz etkilemektedir. Aynı şekilde karayolunun sol kısmındaki kara yolu ile benzer şekil ve renkteki alanda yanlış tespit yapılmasına neden olmaktadır. Bu görüntüde yol genişliği parametresi en az 7 ve en çok 15 olarak alınmıştır.



Şekil 3.17 : Görüntü 4 : Canny kenar bulma.

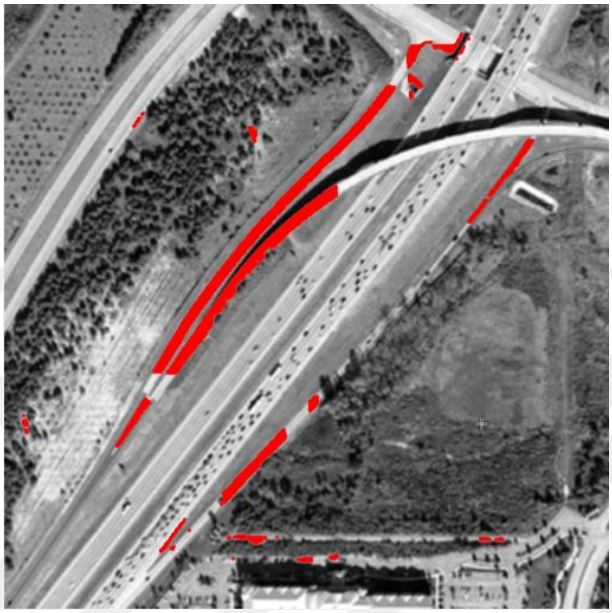
Şekil 3.17'deki görüntü, öncelikle geliştirilen yöntemin ilk üç ana adımının sonucunu göstermektedir. Burada orijinal görüntü önce gri renk uzayına dönüştürülmüş ardından da önce yumuşatma sonra da morfolojik kapama işlemi uygulanmıştır. Ardından Canny kenar bulma algoritması eldeki görüntü üzerinde çalıştırılmış ve algoritmanın kenar olarak tespit ettiği alanlar kırmızı ile çizilmek suretiyle Şekil 3.17'deki gibi oluşturulmuştur.



Şekil 3.18 : Görüntü 4 : LCS.

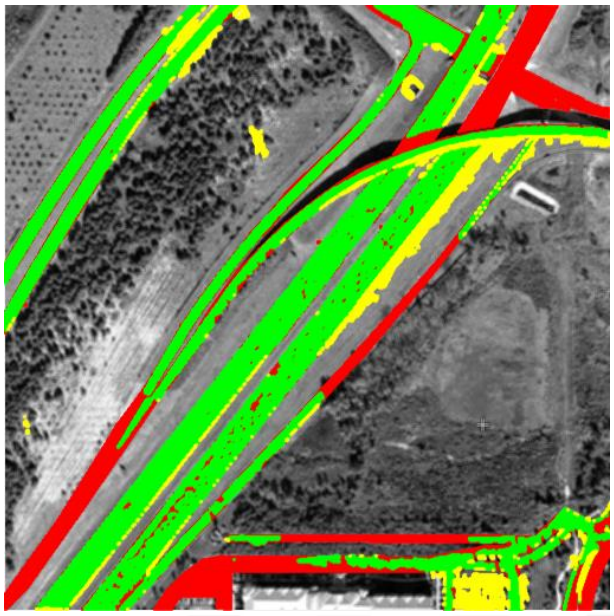
Şekil 3.18'de LCS yöntemi sonucu muhtemel olarak bir yola ait olabileceği belirlenen pikseller kırmızı ile boyanarak gösterilmektedir. Bu işlem için öncelikle zincir kod

yöntemi Şekil 3.17'ye uygulanarak her bir kenar için zincir kodlar elde edilmiştir. Daha sonra her bir kenar nesnesi üzerine başta belirtilmiş olan yol genişliği boyutunda bir çerçeve yerleştirilmiştir. Her çerçeve içindeki kenarların zincir kodlarının, üzerine çerçeve yerleştirilen geçerli kenarın zincir kodu ile olan benzerliği LCS yöntemiyle kontrol edilmiştir. Sonuç olarak geçerli kenarla en çok benzeşen kenarın benzeştikleri pikseller ve arasında kalan alan kırmızı olarak gösterilebilmiştir



Şekil 3.19 : Görüntü 4 : Renk eşik değeri sonrası.

Şekil 3.19'da ise dinamik renk eşiği uygulanarak elde edilen yol pikselleri gösterilmektedir. Burada ormanlık, tarla vb. alanlarda aslında yol olmayan ama zincir kodları benzeşen kenarlar ve aralarındaki alanlar silinmektedir. Dinamik renk değeri bilgisi ise LCS yöntemi sonucu muhtemel yol olarak belirlenen piksellerin ortalama gri renk değeridir. Bu değerden küçük gri rengi olan pikseller silindiğinde Şekil 3.18'de ki görüntü Şekil 3.19'da ki hale gelmektedir



Şekil 3.20 : Görüntü 4 : Bölge büyütme sonrası el çizim ile karşılaştırma

Son olarak Şekil 3.19 üzerinde bölge büyütme algoritması uygulanmış ve Şekil 3.20'deki sonuçlar elde edilmiştir. Söz konusu algoritmanın ihtiyaç duyduğu besleme değeri Şekil 3.19'da kırmızı olarak işaretlenmiş piksellerdir. Diğer bir deyişle muhtemel yol olabilecek alanlardan bölge büyütme işlemi yapılmış ve Şekil 3.20'deki sonuçlar elde edilmiştir. Burada yeşil ile boyanan yerler gerçekte yol olan ve geliştirilen yöntemin de yol olarak bulduğu pikselleri göstermektedir. Kırmızı alanlar ise gerçekte yol olupta geliştirilen yöntem tarafından yol olarak bulunamayan pikselleri göstermektedir. Sarı alanlar da gerçekte yol olmayan ama yol olarak tespit edilen alanları göstermektedir.

3.2 Sonuç Karşılaştırma

Geliştirilen yöntemin başarımını değerlendirebilmek için aşağıda listelenen istatistiksel kıstaslar kullanılmıştır.

Çizelge 3.1'de söz konusu kıstasların hesaplanma yöntemini gösteren karşılaştırma (confusion) matrisi verilmiştir.

Çizelge 3.1 : Karşılaştırma matrisi.

		Referans		
		Pozitif (P)	Negatif (N)	
Test Çıktısı	Pozitif	**Doğru Pozitif (DP)**	**Yanlış Pozitif (YP)**	**Hassasiyet DP / (DP + YP)**
	Negatif	**Yanlış Negatif (YN)**	**Doğru Negatif (DN)**	**Negatif Öngörü Değeri DN / (DN + YN)**
		Duyarlılık DP / (DP + YN)	**Özgüllük DN / (YP + DN)**	**Kesinlik (DP + DN) / (P + N)**

Kıstasların ifade ettikleri anlamlar ise şöyledir

- **Duyarlılık (Sensitivity)** : Yol olan pikselleri yol olarak tanımlayabilme oranıdır. Bu değer, bazı kaynaklarda [12][52] bütünlük (completeness) olarak da geçmektedir. Test verisinin çoğunluğunu pozitif kabul eden bir yöntemde bu değer çok yüksek çıkabilir.
- **Özgüllük (Specifity)** : Yol olmayan piksellerin yol olmadığını tanımlayabilme oranıdır. Bu değer, bazı kaynaklarda[12][52] doğruluk (correctness) olarak da geçmektedir. Test verisinin çoğunluğunu negatif kabul eden bir yöntemde bu değer çok yüksek çıkabilir.
- **Hassasiyet (Precision)** : Gerçekte yol olan ve yol olarak bulunan piksel sayısının tüm yol olarak bulunan piksel sayısına oranıdır. Bu değerin yüksek olması sistemin güvenilir bir pozitiflik kapasitesine sahip olduğunu gösterir. Ancak bu değer başarımın dar bir alanda mı yoksa tüm görüntüde mi olduğunu göstermez. Diğer bir deyişle az sayıda piksel hatasız olarak bulunsa da, daha tespit edilemeyen birçok piksel olduğundan tüm görüntü için bu değer güvenilir olmayabilir.
- **Negatif Öngörü Değeri (NÖD)** : Gerçekte yol olmayan ve yol değil olarak bulunan piksellerin tüm yol değil olarak bulunan piksellere oranıdır. Bu değerin yüksek olması sistemin güvenilir bir negatiflik kapasitesine sahip olduğunu gösterir. Ancak hassasiyetteki ölçek boyutunun belirsizliği sorunu burada da mevcuttur.
- **Kesinlik (Accuracy)** : Doğru olarak bulunanların (yol ve yol olmayan) sayısının toplam görüntüdeki piksel sayısına oranıdır. Bu değerin yüksek

olması sistemin hem başarımının yüksek olmasını hem de kapsamının geniş olmasını gösterir.

- **F1** : Hassasiyet ve duyarlılık değerlerinin harmonik ortalamasıdır. Bazı kaynaklarda [52] kalite diye de geçmektedir.

Çizelge 3.2' de zincir kod yönteminin sonucunun LCS ile karşılaştırması neticesinde elde edilen başarım sonuçları verilmiştir. Çizelge 3.3' de ise bu sonuçların üzerine bölge büyütme algoritması kullanılması suretiyle elde edilen son başarım değerleri verilmiştir.

Uydu görüntülerindeki kara yollarının tespiti problemi açısından bu çalışmada sunulan zincir kod ve LCS yöntemlerinin Çizelge 3.2' deki kesinlik değerlerine bakıldığında makul denebilecek bir başarım görülse de F1 skorlarının düşük olması tek başlarına söz konusu problemin çözümünde yetersiz kaldıklarını ortaya koymaktadır. Bununla birlikte söz konusu çıktıların bölge büyütme yöntemi gibi başlangıç değerinin kullanıcı tarafından el ile girilmesi gereken yerlerde kullanılması suretiyle arttırılması mümkündür. Böylece hem el ile giriş yapma gereksinimi ortadan kaldırılırken hem de genel sistemin başarımı daha da iyileştirilebilmektedir.

Çizelge 3.2 : Zincir kod ve benzerlik karşılaştırma sonrası başarım.

	Görüntü 1	**Görüntü 2**	**Görüntü 3**	**Görüntü 4**
Duyarlılık	18.37%	4.69%	23.73%	9.44%
Özgüllük	96.44%	97.04%	96.54%	97.62%
Hassasiyet	55.22%	27.72%	51.46%	52.98%
NÖD	83.19%	80.79%	89.11%	79.17%
Kesinlik	81.40%	79.04%	86.79%	78.13%
F1	27.57%	8.03%	32.48%	16.02%

Çizelge 3.3 : Bölge büyütme yöntemi sonrası başarım.

	Görüntü 1	**Görüntü 2**	**Görüntü 3**	**Görüntü 4**
Duyarlılık	90.86%	91.27%	91.09%	80.31%
Özgüllük	97.94%	94.96%	98.08%	91.20%
Hassasiyet	91.31%	81.41%	87.98%	72.15%
NÖD	97.82%	97.82%	98.61%	94.23%
Kesinlik	96.57%	94.24%	97.14%	88.80%
F1	91.08%	86.06%	89.51%	76.01%

Geliştirilen yöntemin başarımı aynı görüntüleri kullandığımız Das ve diğ. (2011) [12] makalesindeki başarımlara oldukça yakındır. Bu başarım karşılaştırması Çizelge 3.4'te verilmiştir. Söz konusu makalede duyarlılık ve özgüllük değerlerinin 25 adet görüntünün başarımının ortalaması olarak elde edildiği belirtilmiştir. Bu çalışmadaki değerler için ise kullanılan dört resmin başarım ortalamaları alınmıştır. Yukarıda çizelge kısıtlarının anlamlarında belirtildiği gibi kesinlik ve F1 değerleri başarıma ait daha güvenilir değerler sunabiliyor olsa da söz konusu makalede bu değerler verilmediğinden karşılaştırmada kullanılmamıştır. Ayrıca söz konusu makalenin çalışma süresi ile geliştirilen yöntemin çalışma süreleri de tabloda verilmiştir. Karşılaştırmanın kabul edilebilir olması açısından makalede kullanılan sistemin benzeri oluşturulmuştur. Bu sistem bir adet Pentium IV 3GHz Dört Çekirdekli işlemci içeren bir sistemdir.

Çizelge 3.4 : Literatür ile sonuç karşılaştırma.

	Duyarlılık	**Özgüllük**	**Süre (sn)**
Das ve diğ. (2011)	93%	97%	502
Geliştirilen Metot	88%	96%	120

Çizelge 3.4 de görülebileceği gibi başarım oranları oldukça yakın olduğu halde işlem süresi dörtte bir oranında daha düşüktür.

Geliştirilen yöntem her ne kadar şehirlerarası yollarda başarım sağlayabilmiş olsa da şehir içi yollar özellikle de sokaklar için yetersiz kalmaktadır. Bunun sebebi zincir kod yönteminin uzun kenarlar da daha başarılı sonuçlar vermesinden dolayıdır. Dolayısıyla, önerilen yöntemde başarımın iyileştirilmesi için Das ve diğ. (2011) [12] yöntemindeki gibi çeşitli makine öğrenmesi yöntemleri kullanılabilir.

4. DEĞERLENDİRME

Çalışmanın başında da bahsedildiği gibi uydu görüntülerindeki kara yollarının tespiti karmaşık ve tek bir yöntemle başarılamayacak bir problemdir. Güncel yol bilgisinin hızlı bir şekilde alınıp haritalara yansıtılması gerekliliği navigasyon ve CBS sistemleri açısından elzemdir. Ancak yüksek miktardaki verinin insan tarafından hızlıca işlenmesi mümkün olmadığından bilgisayarlı görme yöntemlerine ihtiyaç duyulmaktadır.

Bu çalışmada da söz konusu sorunun çözümü için literatürde birçok farklı alanda kullanılan zincir kod ve LCS yöntemleri bir ön işlem olarak önerilmiştir. Önerilen yarı otomatik sistemde sadece yol genişlik bilgisi girilmesi gerekmektedir. Girilecek bu değer sistemin başarımına direk etki eden en önemli faktördür. Bununla birlikte zincir kod yönteminin kenar bulma algoritması sonucuna göre başarım göstermesinden dolayı kenar bulma algoritmasında yapılacak iyileştirmeler de sonuçlar üzerinde olumlu etki bıraktığı görülmüştür.

Zincir kod ve LCS yöntemlerinden ortaya çıkan başarım sonuçları her ne kadar yüksek olmasa da özelliklede el ile öğrenim verisi girişi yapılması gereken adımlar için (makine öğrenmesi, bölge büyütme yöntemi vb.) bu işlemin otomatize edilmiş olması açısından önemlidir.

Bununla birlikte geliştirilen sistemin çıktıları bölge büyütme yöntemiyle daha da iyileştirilmiş ve literatürde çok daha fazla adımdan oluşan ve makine öğrenmesi içeren yöntemlere yakın başarım elde edilmiştir. Her ne kadar makine öğrenmesi üzerine yoğunlaşmış olan çalışmaların başarım değerleri daha yüksek olsa da sistemin eğitilmesi noktasında ciddi bir veri büyüklüğüne ve eğitim süresine ihtiyaç duyulmaktadır[19]. Bu çalışmada sunulan yöntem başarımı yüksek tutarken bu tarz veri miktarı gereksinimleri ile eğitim süresi gibi süreçleri önemli derecede azaltacaktır.

Geliştirilen yöntemi sadece kara yolları için değil geometrik olarak büyük çoğunlukla doğrusal biçimlerdeki nehir, damar vb. birçok farklı nesnelerin tespitinde de kullanmak mümkündür.

Yöntemin başarımının daha da arttırılması ve şehir içi yolların da tespit edilebilmesi açısından makine öğrenmesi yöntemlerinin çalışmaya eklenmesi yapılabilecek iyileştirmelerden biridir. Bir başka iyileştirme işlemi de yüksek çözünürlüklü hiperspektral görüntüleri kullanmak suretiyle makine öğrenmesi adımı için daha fazla özellik elde edilmesi ve daha yüksek doğruluklu sonuçlar elde etmektir.

KAYNAKLAR

[1] **J. K. Anderson and L. Casson**, 2010, "Travel in the Ancient World," *Class. World*, 2010.

[2] **D. UÇAR**, 2000, "Harita,". [Online]. Available: http://www.hkmo.org.tr. [Erişim: 13-Mar-2019].

[3] **A. Wolodtschenko and T. Forner**, 2007, "Prehistoric and early historic maps in Europe: Conception of Cd-Atlas," *e-Perimetron*, vol. 2, no. 2, pp. 114–6

[4] **J. Mellaart**, 2007, "Excavations at Çatal Hüyük, 1963: Third Preliminary Report," *Anatol. Stud.*

[5] **B. Museum**, "The Map of the World." [Online]. Available: https://www.britishmuseum.org/research/collection_online/collection_object_details.aspx?objectId=362000&partId=1&searchText=92687&page=1. [Erişim: 19-03-2019].

[6] **M. Rajeswari, K. S. Gurumurthy, S. N. Omkar, J. Senthilnath, and L. P. Reddy**, 2011, "Automatic road extraction using high resolution satellite images based on level set and mean shift methods," in *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*

[7] **S. Evans**, "1993 EUNOS/MAZDA COSMO CLASSIC DRIVE Uncosmopolitan: Meet the Rarest Mazda in America," 2013. [Online]. Available: https://www.motortrend.com/news/12q2-1993-eunos-mazda-cosmo-drive/. [Erişim: 19-03-2019].

[8] **National Research Council (U.S.)**, 1995, *The global positioning system: a shared national asset: recommendations for technical improvements and enhancements*. National Academies Press

[9] **S. K. Moore**, 2017, "Super-Accurate GPS Chips Coming to Smartphones in 2018 - IEEE Spectrum," *IEEE Spectr.*

[10] **R. Bajcsy and M. Tavakoli**, 1976, "Computer Recognition of Roads from Satellite Pictures," *IEEE Trans. Syst. Man Cybern.*, vol. 6, pp. 623–37

[11] **R. S. Babu, B. Radhakrishnan, and L. Padma Suresh**, 2017, "Detection and extraction of roads from satellite images based on Laplacian of Gaussian operator," in *Proceedings of IEEE International Conference on Emerging Technological Trends in Computing, Communications and Electrical Engineering, ICETT 2016*

[12] **S. Das, T. T. Mirnalinee, and K. Varghese**, 2011, "Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3906–31

[13] **C. Ünsalan and K. L. Boyer**, 2011, *Multispectral Satellite Image Understanding*

[14] **F. Ameri, M. Mobaraki, Z. Valadan, and M. Javad**, 2008, "Semi-automatic extraction of different-shaped road centerlines from MS and pan-sharpened IKONOS images," *Remote Sens. Spat. Info. Sci.*, p. 37

[15] **B. Sirmaçek and C. Ünsalan**, 2010, "Road network extraction using edge detection and spatial voting," in *Proceedings - International Conference on Pattern Recognition*, 2010, pp. 3113–16.

[16] **H. M. Ali, A. Boshir, and I. M. Ariful**, 2010, "Automatic extractions of road intersections from satellite imagery in urban areas," in *ICECE 2010 - 6th International Conference on Electrical and Computer Engineering*, pp. 686–689.

[17] **A. Kirthika and A. Mookambiga**, 2011, "Automated road network extraction using artificial neural network," in *International Conference on Recent Trends in Information Technology, ICRTIT 2011*, pp. 1061–65.

[18] **L. Abraham and M. Sasikumar**, 2013, "A fuzzy based road network extraction from degraded satellite images," in *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, pp. 2032–36.

[19] **R. Alshehhi and P. R. Marpu**, 2017, "Hierarchical graph-based segmentation for extracting road networks from high-resolution satellite images," *ISPRS J. Photogramm. Remote Sens.*, vol. 126, pp. 245–60.

[20] **M. Maboudi, J. Amini, S. Malihi, and M. Hahn**, 2018, "Integrating fuzzy object based image analysis and ant colony optimization for road extraction from remotely sensed images," *ISPRS J. Photogramm. Remote Sens.*, vol. 138, pp. 151–63.

[21] **R. Liu, J. Song, Q. Miao, P. Xu, and Q. Xue**, 2016, "Road centerlines extraction from high resolution images based on an improved directional segmentation and road probability," *Neurocomputing*, vol. 212, pp. 88–95.

[22] **X. Liu, J. Tao, X. Yu, J. J. Cheng, and L. Q. Guo**, 2012, "The rapid method for road extraction from high-resolution satellite images based on USM algorithm," in *Proceedings of 2012 International Conference on Image Analysis and Signal Processing, IASP 2012*.

[23] **H. Grailu, M. Lotfizad, and H. Sadoghi-Yazdi**, 2009, "1-D chaincode pattern matching for compression of Bi-level printed farsi and arabic textual images," *Image Vis. Comput.*, vol. 27, pp. 1615–25.

[24] **H. Sánchez-Cruz, E. Bribiesca, and R. M. Rodríguez-Dagnino**, 2007, "Efficiency of chain codes to represent binary objects," *Pattern Recognit.*, vol. 40, pp. 1660–74.

[25] **G. Q. Lu, H. G. Xu, and Y. B. Li**, 2005, "Line detection based on chain code detection," in *2005 IEEE International Conference on Vehicular Electronics and Safety Proceedings*, pp. 98–103.

[26] **E. Bribiesca**, 1999, "A new chain code," *Pattern Recognit.*, vol. 32, pp. 235–5.

[27] **B. S. Anami and V. B. Pagi**, 2013, "Acoustic signal based detection and localisation of faults in motorcycles," *IET Intell. Transp. Syst.*, vol. 8, no. 4, p. 345.

[28] **C. Gruber, T. Gruber, S. Krinninger, and B. Sick**, 2010, "Online signature verification with support vector machines based on LCSS kernel functions," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*

[29] **G. Bourque and P. A. Pevzner**, 2002, "Genome-scale evolution: Reconstructing gene orders in the ancestral species," *Genome Res*.

[30] **Q. Wang, D. Korkin, and Y. Shang**, 2011, "A fast multiple longest common subsequence (MLCS) algorithm," *IEEE Trans. Knowl. Data Eng.*

[31] **D. Frolova, H. Stern, and S. Berman**, 2013, "Most probable longest common subsequence for recognition of gesture character input," *IEEE Trans. Cybern.*

[32] **D. Jarchi, C. Wong, R. M. Kwasnicki, B. Heller, G. A. Tew, and G. Z. Yang**, 2014, "Gait parameter estimation from a miniaturized ear-worn sensor using singular spectrum analysis and longest common subsequence," *IEEE Trans. Biomed. Eng.*

[33] **R. Adams and L. Bischof**, 1994, "Seeded Region Growing," *IEEE Trans. Pattern Anal. Mach. Intell.*

[34] **J. Canny**, 1986, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 679–98.

[35] **M. Shah**, 1997, "Fundamentals Of Computer Vision." Comp. Science Dep. University Of Central Florida, Orlando.

[36] http://www.algorithmist.com/index.php/Longest_Common_Subsequence. [Erişim: 06-09-2018].

[37] https://www.ics.uci.edu/~eppstein/161/960229.html. [Erişim: 06-09-2018].

[38] **C. Poynton**, 2003, *Digital Video and HD: Algorithms and Interfaces*. Morgan Kaufmann.

[39] **O. Marques**, 2011, *Practical Image and Video Processing Using MATLAB®.*

[40] **L. C. Mai**, 2000, "Introduction to Computer Vision and Image Processing." Hanoi.

[41] **R. C. Gonzalez, R. E. Woods, and S. L. Eddins**, 2013, "Digital Image Processing Using Matlab - Gonzalez Woods & Eddins.pdf," *Education*.

[42] J**. S. Lim**, 1989, "Two-Dimensional Signal and Image Processing," *Englewood Cliffs, NJ, Prentice Hall*.

[43] **J. Serra**, 1982, *Image Analysis and Mathematical Morphology*. Academic Press Inc..

[44] **G. Kawade, S. Sahu, S. Upadhye, N. Korde, and M. Motghare**, 2017, "An analysis on computation of longest common subsequence algorithm," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 982–987.

[45] **C. Huang, Q. Liu, and X. Li**, 2010, "Color image segmentation by seeded region growing and region merging," in *Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010*.

[46] **D. Kaur and Y. Kaur**, 2014, "Various Image Segmentation Techniques : A Review," *Int. J. Comput. Sci. Mob. Comput*.

[47] **S. S. Kumar, M. Moorthi, M. Madhu, and R. Amutha**, 2010, "An improved method of segmentation using Fuzzy-Neuro logic," in *2nd International Conference on Computer Research and Development, ICCRD 2010*.

[48] **S. Kannan, G. Nalini, and V. Gurusamy**, 2014, "REVIEW ON IMAGE SEGMENTATION TECHNIQUES".

[49] **P. K. Jain and S. Susan**, 2013, "An adaptive single seed based region growing algorithm for color image segmentation," in *2013 Annual IEEE India Conference, INDICON 2013*.

[50] **J. Dehmeshki, H. Amin, M. Valdivieso, and X. Ye**, 2008, "Segmentation of pulmonary nodules in thoracic CT scans: A region growing approach," *IEEE Trans. Med. Imaging*.

[51] **P. Lu, K. Du, W. Yu, R. Wang, Y. Deng, and T. Balz**, 2014, "A new region growing-based method for road network extraction and its application on different resolution SAR images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*.

[52] **C. Heipke, H. Mayer, H. Wiedemann, and H. Jamet**, 1997, "Evaluation of automatic road extraction," *Int. Arch. Photogramm. Remote Sens*.

ÖZGEÇMİŞ

Ad Soyad: Muhammed Tekin

Doğum Yeri ve Tarihi: Bursa – 11.11.1981

Adres: Adnan Menderes Mh. 26. Sk. No:1 E Blk. D.7 Merkez-YALOVA

E-Posta: mutekin16@gmail.com

Lisans: Yakın Doğu Üniversitesi Bilgisayar Mühendisliği Bölümü

Yüksek Lisans (Varsa):

Mesleki Deneyim ve Ödüller:

Öğr. Gör. – Yalova Üniversitesi Bilgisayar Müh. Bl. – 2012

Bilgi İşlem Yöneticisi – Seçkin Onur A.Ş. – 2006-2012

Programcı – Alfabim A.Ş. – 2004-2005

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Tekin, M., Çetin M., Uydu görüntülerinden zincir kod ve en uzun ortak alt küme yöntemleri ile yarı otomatik yol bulma, *ICONDATA-2018,* 04-07 Ekim 2018, Yalova, Türkiye