

**KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

**UYDU MOZAIK GÖRÜNTÜLERİN ÖRÜLMESİ VE NESNE
ÇIKARIMI İÇİN DONANIMA BAĞLI ADAPTİF YAKLAŞIM**

ÜMİT MERT

KOCAELİ 2016

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

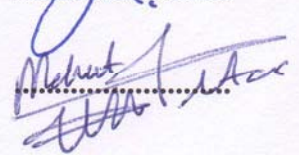
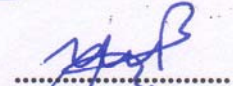
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

UYDU MOZAIK GÖRÜNTÜLERİN ÖRÜLMESİ VE NESNE
ÇIKARIMI İÇİN DONANIMA BAĞLI ADAPTİF YAKLAŞIM

ÜMİT MERT

Doç.Dr. Ahmet SAYAR
Danışman, Kocaeli Üniversitesi
Prof.Dr. Yaşar BECERİKLİ
Jüri Üyesi, Kocaeli Üniversitesi
Yrd.Doç.Dr. Mehmet Sıddık AKTAŞ
Jüri Üyesi, Yıldız Teknik Üniversitesi



Tezin Savunulduğu Tarih: 27.01.2016

ÖNSÖZ VE TEŞEKKÜR

Yüksek lisans eğitimim süresince değerli birikimlerini benimle paylaşan, tezimin her aşamasında sorunlarımı dinleyerek, çalışmalarına yön veren ve yoğun akademik yaşamında değerli zamanını her türlü problemimi çözmeye ayıran tez danışmanım saygıdeğer hocam Doç. Dr. Ahmet SAYAR'a ve Arş. Gör. Süleyman EKEN'e teşekkürlerimi sunarım.

Bugünlere gelmemi sağlayan anneme ve babama, maddi ve manevi desteklerinden dolayı saygı, sevgi ve sonsuz teşekkür ederim. Ayrıca tez projemi destekleyerek bana tezim üzerinde çalışma olanağı sağlayan ve değerli vakitlerini tezimde karşılaştığım sorunlar için ayıran TÜBİTAK BİLGEM BTE Hava Yolu Ulaşım Bölümü yönetici ve çalışma arkadaşlarıma teşekkürlerimi sunarım.

Ocak – 2016

Ümit MERT

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR	i
İÇİNDEKİLER	ii
ŞEKİLLER DİZİNİ.....	iv
TABLolar DİZİNİ	v
SİMGELER VE KISALTMALAR DİZİNİ	vi
ÖZET.....	vii
ABSTRACT	viii
GİRİŞ	1
1. GENEL BİLGİLER.....	5
1.1. Tez Çalışmasının Amacı ve Başlatılma Sebepleri	5
1.2. Tez Çalışmasının Katkıları	6
1.3. Tez Düzeni	7
2. İLGİLİ ÇALIŞMALAR	8
2.1. Görüntülerin Birleştirilmesi	8
2.2. Görüntülerden Nesne Çıkarımının Yapılması.....	10
2.3. Raster Görüntünün Vektörel Veriye Dönüştürülmesi	10
2.4. Görüntünün Mekansal Veritabanına Kaydedilmesi ve Sorgu Yapılması.....	11
3. UZAKTAN ALGILAMA	12
3.1. Uzaktan Algılamaya Genel Bakış	12
3.2. Uzaktan Algılamanın Tarihçesi.....	13
3.3. Uzaktan Algılamada Çözünürlük Kavramı	14
3.3.1. Konumsal çözünürlük.....	14
3.3.2. Radyometrik çözünürlük	15
3.3.3. Spektral çözünürlük.....	15
3.3.4. Zamansal çözünürlük	17
4. METOT	19
4.1. Donanıma Bağlı Görüntü Birleştirme Mimarisi.....	19
4.1.1. Belleğin etkin kullanılması.....	21
4.1.2. Uydu görüntülerinde bellek (cache) mekanizması.....	25
4.1.3. Uydu görüntülerinin birleştirilmesi	27
4.1.4. Hibrit görüntü birleştirme algoritması.....	29
4.1.4.1. Özellik tabanlı birleştirme yaklaşımı	30
4.1.4.2. Koordinat tabanlı birleştirme yaklaşımı	32
4.2. Birleştirilmiş Görüntülerden Nesne Çıkarımı ve Vektörel Modelleme	36
4.2.1. Sezgisel yaklaşımla yarı otomatik nesne çıkarım mimarisi	36
4.2.2. Vektörel modelleme ve zaman-mekansal analizler.....	39
5. UYGULAMA.....	41
5.1. Geliştirilen Uygulama	41
SONUÇLAR VE ÖNERİLER	51
KAYNAKLAR	54
EKLER.....	59
KİŞİSEL YAYIN VE ESERLER	75

ÖZGEÇMİŞ	76
----------------	----

ŞEKİLLER DİZİNİ

Şekil 1.1.	Temsili mozaik uydu görüntüsü	5
Şekil 2.1.	Çakışan uydu görüntülerinin durumu	9
Şekil 3.1.	Farklı mekansal çözünürlüğe sahip aynı uydu görüntüsü.....	14
Şekil 3.2.	Farklı radyometrik çözünürlüğe sahip aynı görüntü.....	15
Şekil 3.3.	Elektromanyetik spektruma göre görüntülerin durumu.....	16
Şekil 3.4.	Spektral yansıma dalgaları	16
Şekil 3.5.	Uydu görüntülerinde zamansal çözünürlük oluşumu	17
Şekil 4.1.	Donanımına bağlı uydu görüntüsü birleştirme mimarisi	19
Şekil 4.2.	Mozaik uydu görüntülerinin genel görünümü	20
Şekil 4.3.	7681x7831 piksel çözünürlüğe sahip gerçek uydu görüntüsü	22
Şekil 4.4.	Ehcache mimarisi.....	26
Şekil 4.5.	Ehcache konfigürasyon kodu	26
Şekil 4.6.	World file (wld) parametrelerinin farklı durumları	28
Şekil 4.7.	Özellik tabanlı birleştirme algoritma şeması	30
Şekil 4.8.	LandSat 8 OLI uydu mozaik görüntüler	31
Şekil 4.9.	SURF ile özellik çıkarım ve eşleştirme	31
Şekil 4.10.	RANSAC'ın uygulanması ve gürültülerin elimine edilmesi	31
Şekil 4.11.	Aşamalı harmanlama (blending) ile birleştirilmiş görüntü.....	31
Şekil 4.12.	Lighten metodu kullanılarak birleştirilmiş uydu görüntüsü	34
Şekil 4.13.	Lighten metodu kullanılmadan birleştirilmiş uydu görüntüsü.....	35
Şekil 4.14.	Birleştirilmiş uydu görüntüsünü oluşturan parça mozaik görüntüler	35
Şekil 4.15.	Yarı otomatik nesne çıkarım mimarisi akış diyagramı	38
Şekil 4.16.	Nesne çıkarım mimarisi alt adımlarının ara çıktıları	39
Şekil 5.1.	Uygulama anasayfasının arayüzü	41
Şekil 5.2.	Mozaiklerin sisteme yüklenmesini sağlayan arayüz.....	42
Şekil 5.3.	Mozaiklerin seçilmesini sağlayan arayüz	42
Şekil 5.4.	Kullanıcının seçtiği mozaiklerin gösterildiği arayüz	43
Şekil 5.5.	Mozaiklerin birleştirildikten sonra gösterildiği arayüz.....	44
Şekil 5.6.	İlgili mozağin ön tarafa getirilmesi.....	45
Şekil 5.7.	Çıkarılacak nesnenin belirlendiği arayüz.....	46
Şekil 5.8.	Çıkarılacak nesnenin vektörleştirildiği arayüz.....	46
Şekil 5.9.	Çıkarılacak diğer nesnenin belirlendiği arayüz.....	47
Şekil 5.10.	Çıkarılacak diğer nesnenin vektörleştirildiği arayüz	47
Şekil 5.11.	Vektörleştirilen nesnenin farklı pencerede gösterilişi.....	48
Şekil 5.12.	Yüklenmiş mozaiklerin silindiği arayüz	48
Şekil 5.13.	Bellek kullanım bilgisinin gösterildiği arayüz	49
Şekil 5.14.	Logların gösterildiği arayüz	49
Şekil 5.15.	Postgis veritabanının ait alan ve çevre sorgusunun yapıldığı arayüz.....	50

TABLolar DİZİNİ

Tablo 4.1. SIFT ve SURF performans sonuçları	32
Tablo 4.2. Çözünürlük Oranının Çıkarılan Nesne Alanına Etkisi	40

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

BRIEF	: Binary Robust Independent Elementary Features (İkili Sağlam Bağımsız Temel Özellikler)
CBS	: Coğrafi Bilgi Sistemleri
CPU	: Central Processing Unit (Merkezi İşlem Ünitesi)
EMS	: Elektromanyetik Spektrum
FAST	: Features Accelerated Segment Test (Hızlandırılmış Özellikli Bölüm Test)
GB	: Gigabyte (Gigabayt)
GIS	: Geographic Information Systems (Coğrafi Bilgi Sistemleri)
JPEG	: Joint Photographic Experts Group (Birleşik Fotoğraf Uzmanları Grubu)
KM	: Kilometer (Kilometre)
M	: Meter (Metre)
MB	: Megabyte (Megabayt)
NE	: North East (Kuzeydoğu)
NM	: Nanometer (Nanometre)
NW	: North West (Kuzeybatı)
OGC	: Open Geospatial Consortium (Açık Coğrafi Konsorsiyum)
OLI	: Operational Land Imager (Operasyonel Arazi Görüntüleyici)
ORB	: Oriented FAST and Rotated BRIEF (Yönlü FAST ve Döndürülmüş BRIEF)
PCA-SIFT	: Principal Component Analysis - Scale-Invariant Feature Transform (Temel Bileşen Analizi – Değişmeyen Ölçekli Özellik Dönüşümü)
RANSAC	: Random Sample Consensus (Rastgele Örnek Seçimi)
RGBA	: Red Green Blue Alpha (Kırmızı Yeşil Mavi Alfa)
SE	: South East (Güneydoğu)
SIFT	: Scale-Invariant Feature Transform (Değişmeyen Ölçekli Özellik Dönüşümü)
SQL	: Structured Query Language (Yapılandırılmış Sorgu Dili)
SURF	: Speeded Up Robust Features (Hızlandırılmış Sağlam Özellikler)
TIFF	: Tagged Image File Format (Etiketlenmiş Görüntü Formatı)
USGS	: United States Geological Survey (Amerika Birleşik Devletleri Jeolojik Araştırmalar)
WLD	: World File (Dünya Dosyası)

UYDU MOZAİK GÖRÜNTÜLERİN ÖRÜLMESİ VE NESNE ÇIKARIMI İÇİN DONANIMA BAĞLI ADAPTİF YAKLAŞIM

ÖZET

Sistem, LandSat-7 ve LandSat-8 uydularından alınan mozaik görüntülerin birleştirilerek tek parça ada uydu görüntüsünün oluşturulması ve bu görüntülerin renk uzayı dönüşümü segmentasyonu, bölge gruplama-etiketleme ve sınır piksellerinin takibine dayalı vektörleştirme işlemine tabi tutularak poligon-vektör modellerinin oluşturulması ve son olarak, nesne ilişkisel veritabanları üzerinden zaman-mekânsal (spatio-temporal) ve topolojik analizlerinin yapılmasına olanak sağlanmasına dayanmaktadır. Ada görüntülerinin seçilmesinin nedeni, adaların diğer mekânsal objelerden (ülke, şehir, nehir, yol gibi) görsel olarak uydu görüntüleri üzerinden ayırt edilebilmeleridir.

Uydu görüntüleri, yüksek çözünürlüğe sahip görüntülerdir. Bu sebeple birden fazla parça uydu görüntüsü birleştirilerek tek parça görüntünün elde edilmesi işlemi, yüksek boyutta bellek (memory) ve işlem gücü (cpu) gerekmektedir. Geliştirilen uygulama ile mozaik uydu görüntüleri boş bellek miktarını dikkate alan bir yaklaşımla birleştirilmiştir. Birleştirmek istenen uydu görüntüsü sayısı ve çözünürlüklerinin yanı sıra mevcut boş olan bellek miktarı dikkate alınarak, görüntülerde uygulanacak olan çözünürlük azaltma oranı hesaplanmaktadır. Ayrıca üzerinde çalışılmak istenen ada uydu görüntüsünün sezgisel olarak bir operatör aracılığıyla seçilebileceği arayüz geliştirilmiştir.

Uydu görüntüleri üzerinde yapılan zaman-mekânsal ya da topolojik analizler piksel bazlı bilgiden ziyade nokta setleriyle tanımlanan şekil koordinat bilgilerini gerektirmektedir. Geliştirilen proje ile birleştirilen görüntüden yarı otomatik nesne çıkarımı ve poligon olarak modellenerek bilgisayarların ve bilgi sistemlerinin kolayca anlayabileceği ve işleyebileceği hale getirilmesi sağlanmıştır. Nesnenin poligon temsili daha sonra nesne-ilişkisel veritabanlarında depolanabilir ve işlenebilir hale getirilmiştir. Bu çalışmada, mekansal analiz prensiplerini kullanılarak çıkarımı yapılan objenin çevresi ve alanı hesaplanmıştır. Bununla birlikte, diğer objelerin de vektör modellemesi, zaman-mekansal, topolojik analiz çalışmalarına temel teşkil edecektir.

Anahtar kelimeler: Görüntü Birleştirme, Mekânsal Veritabanları, Uydu Görüntüleri, Vektörizasyon, Veri Modelleme.

HARDWARE BASED ADAPTIVE APPROACH FOR STITCHING AND OBJECT EXTRACTION OVER SATELLITE MOSAIC IMAGES

ABSTRACT

The system is based on the derivation of an island satellite image from the mosaics, creating polygon-vector model of that island through vectorization based on color space segmentation, region grouping-labeling, contour tracing, and finally storing island object into object-relational spatial databases and making it available for spatio-temporal and topological analysis. The reason behind selecting islands is that it is a visually distinguishable object not like the others such as countries, cities and roads whose boundaries are not easy to define visually.

Satellite images have a high resolution. Therefore, in the process of extracting a single image through merging several satellite images, high memory and powerful processor are needed. The developed application efficiently merge the mosaic satellite images by considering available memory. The rate of resolution reduction which is applied to the images is calculated considering the number of satellite images to be merged with their resolutions and the amount of available memory. In addition to an interface which intended island satellite image can be selected heuristically through an operator is developed.

For spatio-temporal and topologic analysis, vectorial information gives better information than its raster counterpart. The study presented in this project proposal is based on recognition and semi-automatic extraction of an island object in a set of digital images captured by a satellite and modeling it as a polygon (vectorial) and making it easy to process and easy to understand for computers and information science applications. Polygon representation of an island image then can be stored and manipulated through object-relational spatial databases. In this study, the surface and area of the object extracted are calculated using the principle of spatial analysis. It is believed that this study will provide a basic insight for the studies of vector modelling of other objects, spatio-temporal and topologic analysis.

Keywords: Image Mosaicing, Spatial Databases, Satellite Images, Vectorization, Data Modeling.

GİRİŞ

Uzaktan algılama teknolojileri [1-3] özellikle uydu görüntüleri meteoroloji, küresel değişim algılama ve izleme, mineraller ve petrol arama, doğal kaynakların yönetimi, tarım, çevresel değerlendirme ve izleme, afet ve kurtarma, askeri gözetim dâhil olmak üzere birçok alanda geniş bir uygulama yelpazesine sahiptir [4-9]. Uydu görüntülerinden nesne çıkarımı (yol, köprü vb.) da uzaktan algılama görüntü işleme araştırma alanlarından biridir. Uydu görüntüsü üzerinde herhangi bir nesne çıkarımı yapabilmek için uydudan alınan parça (mozaik) görüntülerin birleştirilmesi gerekmektedir. Uydu sensörlerinin uzamsal, spektral ve zamansal çözünürlüklerinin hızlı bir şekilde artması ile birlikte uydu görüntülerinin büyüklüğü ve karmaşıklığı da üstel olarak artmıştır. Örneğin USGS'den alınan birleştirilecek mozaik uydu görüntülerinin her biri 8 MB civarındadır. Bu görüntüler bilgisayarda hesaplama için açıldığında yaklaşık 300-400 MB'lık tampon (buffer) kaplıyor ve 2 GB'lık bir makinede iki tane uydu görüntüsünün birleştirilmesi bile yığın büyüklüğü (heap size) problemine sebep oluyor. Birleştirilecek olan görüntü sayısı arttıkça da ihtiyaç duyulan bellek miktarı artmakta ve bellek sıkıntısına neden olmaktadır. Bir diğer problem ise uydu görüntülerinin çözünürlüğünün çok yüksek olması sebebiyle yüksek işlem gücü ve bellek gereksinimine ihtiyaç duyulmaktadır.

Uydu mozaik görüntülerinin zaman-mekânsal (spatio-temporal) veriler olmasından dolayı bilinen piksel tabanlı özellik çıkarımı ve anahtar noktalar (key points) tespitine dayalı metotlarla görüntülerin birleştirilmesi çok zor ve başarı oranı çok düşüktür. Uydu görüntüleri, aynı bölgenin farklı zamanlarında alınan görüntüler olabilmektedir. Aynı bölgeyi temsil eden görüntü parçalarının bir kısmı güneşli bir havada elde edilmiş net bir görüntü iken bir diğer parçası bulutlu bir havada elde edilmiş bulanık bir görüntü olması ihtimal dâhilindedir. Bu sebeple bu görüntüler üzerinde benzer özelliklere sahip noktaların çıkarılarak bu noktalara göre birleştirme yapılması oldukça zor görünmektedir. Bu problemi çözmek için, uydu görüntülerini meta verileri şeklinde gelen koordinat bilgilerini kullanarak görüntüleri birleştirdik.

Literatürde görüntü birleştirme teknikleri genel olarak (i) özellik tabanlı (feature based) ve (ii) alan tabanlı (area based) olmak üzere ikiye ayrılmaktadır. Alan tabanlı yöntemlerde görüntülerdeki ayırt edici belirgin özellikler bulunmadan işlem yapmaktadırlar ve bu yöntemler görüntülerin yoğunluklarından (intensity) etkilenmektedirler. Doğrusal olarak değişmeyen farklı elektromanyetik yansımalarından dolayı alan temelli yaklaşımlar multi-spektral uydu görüntülerin birleştirilmesinde iyi sonuç vermezler. Alan tabanlı yöntemler yeterince ayırt edici anahtar noktalar içermeyen medikal görüntüler gibi görüntülerin birleştirilmesinde tercih edilir. Özellik tabanlı yöntemler ise görüntünün yoğunluk değerleri ve dağılımları ile ilgilenmezler. Bu yöntemler görüntülerin baskın özelliklerini çıkararak birleştirme işlemi yaparlar, bu yüzden uzaktan algılama uydu görüntülerinin birleştirilmesinde daha çok kullanılırlar. Her iki yöntemi de kullanan yaklaşımlar da mevcuttur [10-13]. Alan tabanlı birleştirme yaklaşımları uydu görüntüleri üzerinde başarılı değilken özellik tabanlı olanlar ise yukarıda bahsedilen kısıtlamalardan dolayı istenilen sonucu vermemektedir.

Büyük boyutlu uydu görüntülerinin birleştirilmesinde bellek problemlerinin çözüm olarak bilgisayar sistem özelliklerini dikkate alan (system-aware) ve içerik-kaynak farkında (content-aware) adaptif bir birleştirme algoritması geliştirildi. Ayrıca uygulama ihtiyaçlarına göre kullanıcıya görüntü birleştirme için alternatif çözümler sağlandı. Yani kullanıcı için performans-hız mı daha çok önemli yoksa doğruluk mu, bu kriterlere bakıldı. Hız ve doğruluk böyle sistemlerde bir çeşit çelişkidir. Hızlı olan yöntem istenildiği gibi doğru çalışmayabilir. Görüntü birleştirmelerde de durum bunun gibidir. Biz de kullanıcıya alternatif çözümler sunarak güçlü bir görüntü birleştirme iskelet yapısı (framework) oluşturduk. Doğruluk derecesi yüksek algoritmalar Ölçek Bağımsız Özellik Dönüşümü -Scale Invariant Feature Transform- (SIFT) [14] ya da Hızlı Gürbüz Özellikler -Speed-up Robust Features- (SURF) [15] ifade edilen özellik (feature) tabanlı birleştirme algoritmalarıdır. Ancak uydu görüntülerinin multi-spektral ve zaman-mekansal (spatio-temporal) özelliklerinden dolayı uydu görüntülerinde ayırt edici özelliklerin doğru bulunması hem zaman almakta hem de bazen tam bulunamadığı için sonuç vermemektedir [16]. Kısaca bu yöntemlerle birleştirme yapma risklidir ve çok zaman almaktadır. Doğruluktan ziyade hız önemliyse geometrik kesme yaklaşımı daha uygundur. Bu yaklaşım, koordinatlara

göre parça görüntülerini uygun yerlere konumlandırarak ve üst üste gelen yerleri tespit ederek uygun şekilde birleştirme yapan koordinat tabanlı bir yaklaşımdır. Mozaiklerin meta-verileri üzerinde çalıştığı için ortak bir koordinatta tanımlı parça görüntüleri birleştirmede kullanılabilir [17]. Bu yaklaşımın avantajı da görüntü kirliliklerinin görüntü işlemedeki negatif etkilerinden kurtulmaktır. Görüntülerin özelliklerini çıkarmak gerekmediğinden spektral görüntülerin zamana göre değişen özelliklerinden sistem etkilenmemektedir. Yukarıda anlatılan kullandığımız görüntü birleştirme yöntemleri ile ilgili son kullanıcıların kolayca kullanabileceği kullanıcı dostu arayüzler geliştirilmiş ve gerçek uydu görüntüleri üzerinde başarımlı testleri yapılmıştır. Uydu görüntüleri olarak U.S. Geological Survey (USGS)'den alınan LandSat-7 ve LandSat-8 uydu görüntüleri kullanılmıştır.

Bilgisayarlarda görüntü; raster grafik denilen gri renk ton değerlerini içeren iki boyutlu bir dizi formatında veya vektör grafik denilen nokta, doğru, poligon gibi matematiksel olarak ifade edilen geometrik primitifler yolu ile olmak üzere iki şekilde temsil edilir. Raster görüntüde (örneğin TIFF görüntü) görüntüyü oluşturan bütün piksellerin lokasyon ve renk bilgileri tutulduğu için dosyanın boyutu çok büyük olmaktadır. Aynı görüntünün vektörel temsilde (örneğin text dosyasındaki x, y dizileri) görüntünün matematiksel tanımlaması tutulduğu için dosya boyutu orijinalinden onlarca hatta binlerce kat daha küçük boyuta sahip olabilmektedir. rasterdan vektöre dönüşüm diğer bir deyişle vektörizasyon; coğrafi bilgi sistemlerinde (CBS) ve uzaktan algılama görüntü işleme uygulamalarında, mühendislik çizimlerini analog biçimden sayısal ortama aktarmada, doküman analizi ve tanımda önemli bir işleve sahiptir.

İnternet, coğrafi bilgi sistemleri kullanıcıları için mekânsal veri (raster veya vektör veri) transfer etmek, analizler gerçekleştirmek ve coğrafi sonuçları göstermek adına önemli bir içerik yayınlama ortamı haline gelmiştir. Web teknolojilerinin gelişmesiyle coğrafi bilgiye dayalı uygulamalar artmıştır. Diğer yandan, hızla artan mekânsal veri miktarı ile ağ bant genişliği arasında bir ikilem vardır. Cevap süresinin (istemcinin sunucudan mekânsal veri istemesinden veriyi almasına kadar geçen süre) kısa olması web-CBS uygulamalarında önemlidir. Uydu görüntülerinin piksel ızgarası şeklinde (raster) tutulmasından ziyade vektörel olarak modellenmesi; aynı verinin daha az yer kaplaması, internet üzerinden daha hızlı taşınması ve daha hızlı işlenmesi ve yorumlanmasına olanak verir.

Çalışmanın ikinci bölümünü birleştirilmiş uydu görüntüsünden nesne çıkarımı mimarisi oluşturmaktadır ve bu kısım mozaik görüntülerin birleştirilmesini gerçekleyen mimari üzerine kurulmuştur. Birleştirilen görüntü üzerinde bir ada uydu görüntü olduğu gibi birden fazla da ada uydu görüntüsü de bulunabilir. Koordinatları elde edilmek istenen ada uydu görüntüsü geliştirilen arayüz aracılığıyla sezgisel olarak seçilebilmektedir. Bu seçim, ada nesnesinin etrafına noktalar bırakılarak yapılmaktadır. Bir kullanıcı aracılığıyla yapılan bu işlem sonucunda, ada nesnesinin noktaların birleştirilerek oluşturduğu poligon içerisinde olduğu varsayılmaktadır. Bu poligon üzerinde yapılan görüntü işleme teknikleriyle ada nesnesinin çıkarımı yapılmaktadır. Çıkarılan nesnelere koordinat bilgileri ile poligon, çizgi dizisi (linestring) şeklinde modellenmekte ve mekânsal veritabanlarında depolanıp zaman-mekânsal analizler için hazır hale getirilmektedir.

1. GENEL BİLGİLER

1.1. Tez Çalışmasının Amacı ve Başlatılma Sebepleri

Uzaktan algılamayla elde edilen uydu görüntüleri, çözünürlükleri çok yüksek olan görüntülerdir. Bundan dolayı yüksek çözünürlükleri uydu görüntülerini işleyebilmek için yüksek boyutta bellek ve işlem gücü gerekmektedir. Bu projeye amaçlanan, uydu görüntüsünü işleyebilmek için gerekli olan bellek miktarını azaltmak ve son kullanıcı bilgisayarlarında da çalışmasını sağlayan bir uygulama geliştirmektir. Elde edilen uydu görüntüleri Şekil 1.1’de gösterilmektedir.



Şekil 1.1. Temsili mozaik uydu görüntüsü

Bu tez çalışmasında geliştirilecek olan uygulamanın özelliklerinden biri de, uydu görüntülerinden nesne çıkarımıdır. Yukarıdaki uydu görüntüsü alanı, tam bir nesneyi barındırabileceği gibi, bir nesnenin bir parçasını da barındırabilmektedir. Elde edilen uydu görüntüsüne göre, nesne çıkarımı yapmaya çalıştığımızda, eğer bu görüntü çıkarım yapmak istediğimiz nesnenin bir parçasını barındırıyor, nesnenin diğer parçalarını barındıran uydu görüntüleriyle birleştirilmesi gerekmektedir. Nesnenin kapladığı alana göre, birleştirilmesi gereken uydu görüntüsünün sayısı da artacaktır.

Birleřtirilmek istenen uydu grnts sayısı arttıķa, iřlem gc ve bu grntleri barındıracak olan bellek boyutu da artacaktır. Bir noktadan sonra, bellek yetmeyecektir ve hata verecektir. Gnmzdeki bilgisayarların birçoęunu hedeflendięinden dolayı, uydu grntleri ve bellek arasında bir optimizasyon gerekmektedir. Bu optimizasyona, Bu tez ile uygulamanın bir dięer zellięi ise, birleřtirilen uydu grntlerinden nesne ıkarımının yapılmasıdır. Nesne ıkarımı iřleminde bařarı oranını artırabilmek iin, birleřtirme sırasında st ste gelen uydu grnt paralarının birbirine uyumlu bir renklere birleřtirilmesi gerekmektedir.

Son olarak bu teze yapılan son iřlem ise, uydu grntlerinden ıkarılan nesneyi, mekansal nesnelere ait verileri saklayabilen ve sorgulayabilen veri tabanlarından biri olan Postgis veritabanına kaydetmek ve bu veri zerinde analizlere temel oluřturmaaktır. Bu veritabanına uydu grntsnden elde edilen nesneye ait veriler, poligon, izgi veya nokta řeklinde kaydedilmektedir.

1.2. Tez alıřmasının Katkıları

Bu alıřma ile uydu grntlerinin zerinde yapılan grnt iřleme fonksiyonlarının, bellek boyutu (minimum 1GB) ve iřlem gc ok yksek olmayan bilgisayarlar zerinde de alıřtırılabilmesi saęlanmıřtır. Bu tez ile geliřtirilen uygulama ile mevcut sistem zerinde bulunan bellek optimize bir řekilde kullanılmıřtır. Ayrıca eklenen uydu grnts sayısı arttıķa, bellek tam kapasiteyle bu uygulamaya hizmet vermesi gerektięinden, dięer uygulamalar alıřamaz duruma gelebilir. Bu durumun gerekleřmemesi sebebiyle, eklenen uydu grntleri, belleęe yklendikten sonra, dosya zerinde alıřan bir nbelleęe yklenmiřtir. Bu sayede, bu tez ile gerekleřtirilen uygulama alıřıyorken, sistem zerindeki dięer uygulamalarla da rahatlıkla alıřma yapılabilir.

Uydu grntleri zerinde iřlem yapmak istendięinde, bazı durumlarda tek bir uydu grnts, ihtiyaları karřılamamaktadır. Bu durumda birden fazla uydu grntsnn birleřtirilmesi gerekmektedir. Ancak uydu grntlerinin her bir parası farklı zamanlarda alınmıř grntler olabilir. Yani bir uydu grnts tamamen bulutlarla

kapalıyken diğeri bir uydu görüntüsü güneşli bir havadayken alınmış olabilir. Bu iki uydu görüntüsünün birleştirilmesi gereken durumlarda, iki parça uydu görüntüsü arasında SIFT veya SURF algoritmaları kullanılarak benzer noktalarını bularak birleştirme işlemi gerçekleştirilemeyebilir. Bu durumdan dolayı, uydu görüntülerinin birleştirilmesi işleminde benzer özelliklerin bulunması işleminin başarı oranı düşüktür. Bu durumdan dolayı bu tez ile yapılan çalışmada, uydu görüntüleriyle birlikte elde edilen koordinat bilgileri kullanılarak birleştirme işlemi gerçekleştirilmiştir. Bu sayede farklı zamanlarda farklı iklim koşullarında alınan uydu görüntüleri birleştirilerek bütün bir karaparçası oluşturulabilir.

Uydu görüntüleri birleştirilerek elde edilen büyük görüntüde, üzerinde zaman-mekansal analizler yapabilmek için bir nesne çıkarımı başarılı bir şekilde yapılmıştır. Burada 2 seçenek bulunmaktadır. Bunlardan birincisi, birleştirilmiş uydu görüntüsü üzerindeki tüm nesnelerin çıkarımıdır. İkincisi ise, üzerinde işlem yapılmak istenen nesnenin seçilerek çıkarılmasıdır. Birinci yöntem ek olarak ikinci yöntemde seçilen noktaların, tüm uydu görüntüsündeki nesnelerin hangisine en yakın olduğunu bulacaktır ve diğer nesnelere dikkate alınmayıp bu nesnenin çıkarımı yapılacaktır.

Son olarak ise çıkarılan nesne, poligon tipinde vektörel veritabanı olan postgis'e kaydedilecek ve bunun üzerinde kaydedilen poligonun çevresi ve alanı gibi bazı sorgular yapılacaktır.

1.3. Tez Düzeni

Bu tez çalışması, 5 bölümden oluşmaktadır. 1. Bölüm'de tez çalışmasının amacı, bu tezin başlatılma sebepleri ve tez çalışmasının katkılarından bahsedilmektedir. 2. Bölüm'de görüntülerin birleştirilmesi, büyük boyutları verilerin işlenmesinde belleğin etkisi, görüntüden nesne çıkarımı ve çıkarılan nesnelerin vektörel modellenmesi ile ilgili geniş bir literatür taraması yapılmıştır. 3. Bölüm'de uzaktan algılama ve tarihçesi, uzaktan algılamadaki çözünürlük kavramı hakkında kısa bilgiler verilmiştir. 4. Bölüm'de görüntülerin birleştirmede belleğin etkin kullanımından, görüntü birleştirmede kullanılan metodların avantajları, dezavantajları, birleştirilmiş görüntüden nesne çıkarımı ve bu nesnenin vektörel modellenip, mekansal veritabanına nasıl kaydedildiğinden bahsedilmektedir. Son bölüm olan 5. Bölüm'de ise, bu tez çalışması sırasında geliştirilen uygulamanın ekran görüntüleri ve çıktıları verilmiştir.

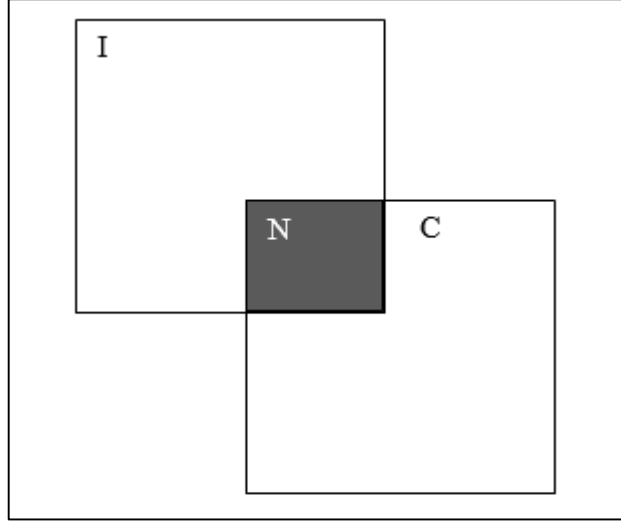
2. İLGİLİ ÇALIŞMALAR

Son yıllarda, uydu görüntülerinin kullanım alanları artmıştır ve buna paralel olarak uydu görüntüleri üzerinde yapılan bilimsel çalışmalarda da artış gözlenmektedir. Bu bölümde uydu görüntülerinin birleştirilerek bir bütün elde edilmesi, bu işlemler yapılırken belleğin etkin kullanılması, birleştirilmiş görüntü üzerinden nesne çıkarımının yapılması ve bu nesnenin vektörel veritabanına kaydedilmesiyle ilgili çözüm yöntemlerinden bahsedilecektir. Bahsedilen yöntemlerde kullanmış teknolojilerden, kullanılan yöntemlerin avantajları ve dezavantajlarıyla ilgili bilgiler verilecektir.

2.1. Görüntülerin Birleştirilmesi

Görüntü birleştirme mozaikleme işlemi, bilgisayarla görme alanında son zamanların popüler olan çalışma alanlarından biridir. Aynı zamanda görsel sahne gösterimi diye de adlandırılır [34]. Görüntü birleştirme, mozaikleme işlemi, birden fazla görüntünün, üst üste çakışan kısımlarını dikkate alarak birleştirilmesidir [22]. Birleştirilmek istenen parça görüntülerle ilgili herhangi bir bilgiye sahip olmadığımız durumlarda, üst üste çakışan kısımlarda bulunan benzer özellikler dikkate alınarak birleştirme işlemi gerçekleştirilebilir. Görüntüler arasındaki benzer özelliklerin çıkarımı işlemi elle, yarı otomatik ve tam otomatik olmak üzere üç şekilde gerçekleştirilebilir. Tam otomatik özellikler çıkarımı, elle ve yarı otomatik özellik çıkarımına göre daha hızlı daha az maliyetlidir [37]. Bu özelliklerin çıkarımı ve eşleştirilmesi işleminde SIFT, PCA-SIFT [32, 21], ORB [33], FAST [31] ve SURF algoritmaları kullanılmaktadır. SIFT, rotasyon, ölçek değişikliği, gürülü, bulanıklaştırma ve ışıklandırma ile ilgili değişimleri çözmek için Lowe tarafından 2004 yılında önerilen bir algoritmadır [26]. SURF ise kısmen SIFT algoritmasından esinlenilmiş, rotasyon ve ölçek değişimlerinden etkilenmeyen, 2006 yılında Herbert Bay tarafından ortaya atılmıştır [27]. SIFT ve SURF algoritmalarıyla özellik çıkarımı ve eşleştirilmesi yaptıktan sonra yapılacak işlem bu verilere görüntü piksellerinin birleştirilmesidir. Birleştirilme işlemi sırasında çakışan görüntülerin kenarlarının, birleştirilmiş görüntüde belirgin olmaması

gerekmektedir [23]. Bu problemin giderilmemesi durumunda, raster görüntünün



Şekil 2.1. Çakışan uydu görüntülerinin durumu

vektörel görüntüye dönüşümünde başarı oranı düşebilir. Şekil 2.1’de gösterildiği gibi üst üste çakışan görüntülerde kenarların belirginliğinin azaltılabilmesi için birkaç çözüm yöntemi vardır. Bu yöntemlerinden birinde, çakışan noktalardaki piksellerin, görüntünün kenar kısmına olan uzaklığı bir ağırlık değeri (α) olarak alınır. Alınan bu ağırlık değeri, görüntülerde Denklem (2.1)’deki gibi çakışan kısımlar piksel değerleri için;

$$N(x,y)=\alpha I(x,y)+(1-\alpha)C(x,y) \quad (2.1)$$

şeklinde uygulanır ve aşağıdaki şekilde görüldüğü gibi birleştirilmiş son görüntüde bulunan, çakışan kısımların piksel değerleri hesaplanır[24].

Kenarların belirginliğinin giderilmesinde bir diğer çözüm yöntemi ise, “lighten” [25] yöntemidir. Bu yöntemde ise her pikselin sahip olduğu RGB değerlerinin maksimum değerleri dikkate alınır. Bu çözüm yöntemi Denklem (2.2) ve (2.3)’deki gibi formüle edecek olursak;

$$I(x,y) \geq C(x,y) \text{ ise } N(x,y) = I(x,y) \quad (2.2)$$

$$I(x,y) < C(x,y) \text{ ise } N(x,y) = C(x,y) \quad (2.3)$$

şeklinde gerçekleşecektir.

2.2. Görüntülerden Nesne Çıkarımının Yapılması

Uydu görüntüleri, internet ve görüntü alma teknolojilerinin gelişmesiyle birlikte erişimin kolaylaştığı yüksek çözünürlüğe sahip görüntülerdir ve buna ek olarak teknolojideki gelişmelerin artmasıyla kompleks görüntü işleme algoritmaları da geliştirilmesine ve kullanımına olanak sağlandı. Bu gelişmelere paralel olarak, uydu görüntülerden nesne çıkarımı işlemi de popüler olmaya başladı [37, 38]. Yapılan çalışmalar uydu görüntülerinden yol çıkarımı, bina çıkarımı, uçak çıkarımı, havalimanı çıkarımı, gemi çıkarımı, ada çıkarımı ve kıyı çıkarımı üzerine yoğunlaşmıştır. Bu çalışmalar sırasında karşılaşılan bazı problemler vardır. Görüntü tiplerinin ve içeriklerinin çeşitliliğinden dolayı otomatik olarak görüntülerin belli bölümlere ayrılabilmesi bir problemdir [36]. Örneğin uydulardan elde edilen görüntünün bulutlu olması, nesne çıkarımı işleminin başarı oranı etkilemektedir. Uydu görüntülerinde bulunan bulutların yok edilmesi ve nesne çıkarımına olan etkisini azaltmak için yapılan çalışmalar vardır.

2.3. Raster Görüntünün Vektörel Veriye Dönüştürülmesi

Mekansal veri, raster ve vektör olmak üzere iki farklı formatta bulunmaktadır [39, 40]. Raster görüntüyü her bir hücresi renk değerlerini tutan ızgara şeklinde bir yapı olarak adlandırabiliriz [28]. Bu matris şeklindeki ızgaranın satır ve sütun sayıları aynıdır. Raster görüntülerde hücre büyüklüğü ile çözünürlüğü arasında ters bir orantı mevcuttur. Bir raster görüntünün hücre büyüklüğü ne kadar küçük ise, o raster görüntünün çözünürlüğü o kadar yüksektir. Bu da ilgili raster görüntüden elde edilebilecek olan detay bilgisinin daha fazla olacağını gösterir. Vektör verinin tanımı ise ilk olarak, raster görüntüde yer alan gerçek dünyanın, geometrik şekillere indirgenmesi olarak 1950'li ve 1960'lı yıllarda yapılmıştır. Geometrik şekiller olarak nokta, çizgi ve poligon belirlenmiştir [29]. Uydu görüntülerinin çözünürlüğünün yüksek olması sebebiyle veritabanında tutulması için, yüksek boyutta veritabanlarına ihtiyaç duyulmaktadır [41]. Ayrıca mekansal analizlerin, sorguların yapılabilmesi için yüksek işlem gücü ve hafızaya da ihtiyaç duyulmaktadır. Bu şartların sağlanması durumunda da yapılacak olan mekansal analizlerle ilgili sonuç alınması daha fazla vakit alacaktır.

Raster görüntüdeki herhangi bir alanın vektör veriye çevrilmesi işlemini, belirlenen alandaki piksellerin gri değerlerine göre, çeşitli sınıflandırma yöntemleriyle elde edilen katmanların sınırlarındaki piksellerin izlenmesi olarak tanımlayabiliriz [30].

2.4. Görüntünün Mekansal Veritabanına Kaydedilmesi ve Sorgu Yapılması

Literatürde mekansal veritabanıyla ilgili birçok tanım vardır [42, 43, 44, 45]. Bunları özetlemek gerekirse, mekansal veritabanı, mekansal verilerin saklanmasına, bu veriler üzerinde sorgu ve analizlerin yapılmasına olanak sağlayan özelleştirilmiş bir veritabanıdır. Mekansal veriler, geometrik veri tipleri olan nokta, çizgi ve poligon şeklinde mekansal veritabanlarında tutulur. Buna ek olarak, bu verilerle birlikte topolojileri ve tarihsel bilgi de tutulabilir [47].

Mekansal verilere erişim 1990'lı yıllara kadar çok kısıtlıydı. Bu yıllardan sonra, mekansal verilerin devlet ve ticari kuruluşlar tarafından internet üzerinden veya dijital coğrafik kütüphaneler, mekansal verihavuzları aracılığıyla dağıtmaya başlanmasıyla, mekansal verilere yönelik yapılan çalışmalar da artmaya başladı [46].

3. UZAKTAN ALGILAMA

3.1. Uzaktan Algılamaya Genel Bakış

Bir cisimle direkt temas kurulmaksızın sahip olduđu fiziksel özellikleri hakkında bilgi elde etme bilimi, uzaktan algılama olarak tanımlanmaktadır. Diđer bir deyişle yeryüzünden belirli uzaklıkta, atmosferde veya uzayda bulunan platformlara yerleştirilmiş ölçüm araçlarıyla cisimlerle fiziksel temas kurulmaksızın, yeryüzünde bulunan cisimler hakkında bilgi edinme ve bu edinen bilgileri değerlendirme yöntemidir. Uzaktan algılama işlemi iki temel aşamadan oluşmaktadır. Bunlar veri elde etme ve veri işleme aşamalarıdır. Verinin elde edilmesi aşağıdaki adımlardan oluşmaktadır;

- Enerji kaynağı: Hedefe bir kaynak aracılığıyla enerji gönderilmesi gerekmektedir. Bu kaynaktan alınan enerjiyle hedef aydınlanır veya hedefe elektromanyetik enerji gönderilmiş olur. Enerji kaynağı güneş olan uydular da vardır, bunlara optik uydular denir. Radar uyduları ise, kendi enerji kaynaklarını üzerlerinde taşıyan uydulardır. Aynı zamanda elektromanyetik enerji üretilip hedefe de gönderirler.
- Hedef ile etkileşim: Atmosferden geçen elektromanyetik dalga, hedefe vardığında hem ışınım hem de hedefin fiziksel özelliklerine bağlı olarak farklı etkileşimler oluşur.
- Işınım ve atmosfer: Enerji, kaynaktan çıkıp hedefe doğru giderken atmosferden geçer ve bu katettiği yol boyunca atmosferde bulunan bazı etkenlerin tesirinde kalır.
- Algılayıcının enerjiyi kaydetmesi: Algılayıcı hedef tarafından yayılan enerjiyi algılar, bununla ilgili olan veriyi kayıt eder.
- Verinin alınması, iletimi ve işlenmesi: Hedeften alınan toplam enerji algılayıcı tarafından kayıt edildikten sonra, işlenmesi sebebiyle uydu yer istasyonuna iletilir.

Verinin işlenmesi adımı ise aşağıdaki adımlardan oluşmaktadır;

- Yorumlama ve analiz: Algılayıcıdan elde edilen görüntü görsel, dijital ve

elektronik işleme teknikleri kullanılarak zenginleştirilir ve analiz edilir.

- Uygulama: Veriler işlendikten sonra bu veriden bilgiye ulaşılır. Bu ek olarak elde edilen sonuçlar, diğer veri kaynakları ile birleştirilerek kullanılabilir.

Günümüzde, uzaktan algılama tekniği birçok alanda etkin bir şekilde kullanılmaktadır ve temel olarak kullanılan alanlar,

- Baraj ve su yataklarının tespiti
- Otoyol, demiryolu, boru hattı koridor seçimi
- Arazi ve toprak hatlarının oluşturulması
- Doğal felaketlerde; orman yangınları ve sel felaketleri hasar tespiti
- Maden aramalarında
- Geniş topraklarda coğrafik bilgi sistemleri için veri toplanmasında
- Kaçak yapılaşma tespiti ve imar planları oluşturma

şeklinde listelenmiştir.

3.2. Uzaktan Algılamanın Tarihçesi

Uzaktan algılama kavramı ilk olarak 1960 yılında Evelyn L. Pruitt tarafından kullanılarak literatüre girmiştir. 1960' dan itibaren uzaktan algılama konusunda birçok önemli gelişmeler yaşanmıştır ve günümüzde de devam etmektedir. Havadan uzaktan algılama, fotoğrafın ve uçağın keşfiyle ortaya çıkmıştır. Uçağın keşfinden önce ise, balonlar, uçurtmalar hava fotoğrafı çekiminde etkin rol almıştır. 1960-1970 yılları arasında uzaktan algılama için uçaklar yaygın olarak kullanılmaktaydı. Ancak, o yıllardan itibaren uydular, uçaklara göre daha fazla alan etkisi altına alabildiklerinden ötürü, uçakları bertaraf ederek ilk tercih edilen platform olmuştur. Sonraki dönemlerde ise, görüntülerin sayısal olarak ifade edilebilmesi çerçevesinde bilgisayarlar kullanılarak, bu görüntüler daha kolay bir şekilde analiz edilebilir bir hal almıştır.

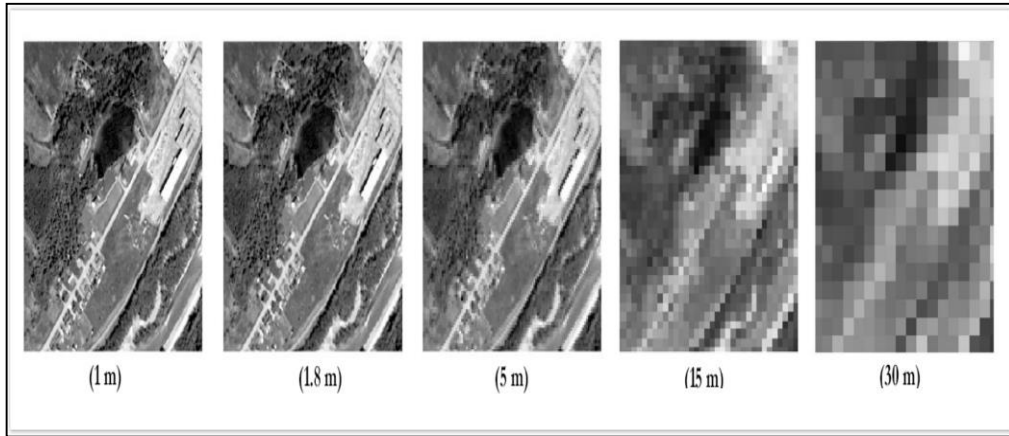
Dünyada ilk uydu 1957 yılında bugünkü adıyla Rusya olan Sovyetler Birliği tarafından Sputnik adıyla uzaya fırlatılmıştır. Bunun ardından 1959 yılında ABD tarafından yerin uzaydan fotoğraflarını çeken ilk insansız uydusu olan Explorer-6 fırlatılmıştır. Yeryüzündeki kaynakların araştırılması, incelenmesi için yine ABD tarafından 1972 yılında ERTS adında bir uydu gönderilmiştir ve bu uydunun adı daha sonra Landsat-1 olarak değiştirilmiştir.

3.3. Uzaktan Algılamada Çözünürlük Kavramı

Sayısal fotoğraf makinelerinin yaygınlaşmasıyla birlikte herkesin hakkında az çok bilgi sahibi olduğu bir kavram olarak ortaya çıkan “çözünürlük” değerinin aslında farklı çeşitleri bulunmaktadır. Sayısal görüntülerden ve özellikle uydu görüntülerinden bahsederken, çözünürlük değeri 4 farklı çözünürlük olarak çeşitlenmektedir.

3.3.1. Konumsal çözünürlük

Uydular üzerinde bulunan algılayıcının yer örnekleme aralığı(Ground Sampling Distance) olarak ifade edilebilmektedir. Yer örnekleme aralığı genellikle komşu piksellerin merkezlerinin yeryüzündeki fiziksel karşılıkları olarak tanımlayabiliriz. Genellikle santimetre (cm) veya metre (m) cinsinden ifade edilmektedir. Örneğin konumsal çözünürlüğü 1 metre olan bir uydu görüntüsünde 1 piksel, yerde 1x1 metrelik bir alana karşılık gelmektedir. Herhangi bir uydu görüntüsünde bulunan bir cisim, 1 piksel boyutunda ise oradaki cismin ayırt edilebilmesi, farkedilebilmesi için komşu piksellerden ayırt edilebilir derecede zıt bir renkte olması gerekmektedir. Bundan dolayı bir cismin ne olduğunun anlaşılabilmesi, cismin farkedilebilmesi için bir pikselden daha fazla piksellerle temsil edilmesi gerektiği, yani boyutlarının bir metreden daha fazla olması gerektiği anlaşılmaktadır.



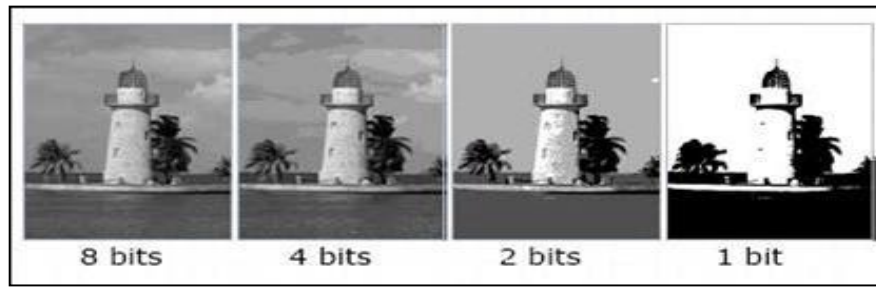
Şekil 3.1. Farklı mekansal çözünürlüğe sahip aynı uydu görüntüsü [52]

Şekil 3.1 'de farklı konumsal çözünürlük değerlerine sahip olan aynı yere ait görüntüler verilmiştir. Görüldüğü gibi konumsal çözünürlük değeri arttıkça cisimlerin temsil edildiği piksel sayısı artmakta ve dolayısıyla da cisimlerin tespiti kolaylaşmaktadır. Yani sadece büyük nesnelerin görülebildiği görüntülerin

çözünürlüğü düşüktür. Küçük nesnelere görülebildiği, ayırtılabildiği görüntülerin ise çözünürlükleri yüksektir.

3.3.2. Radyometrik çözünürlük

Bir algılayıcının elektromanyetik enerji miktarındaki büyüklüğe olan duyarlılığını ifade eder. Diğer bir ifadeyle bir algılayıcının radyometrik çözünürlüğü, enerji farklılıklarını ayırt edebilme yeteneğini gösterir. Kaydedilen enerjinin bölündüğü “bit” sayısına karşılık gelmektedir.

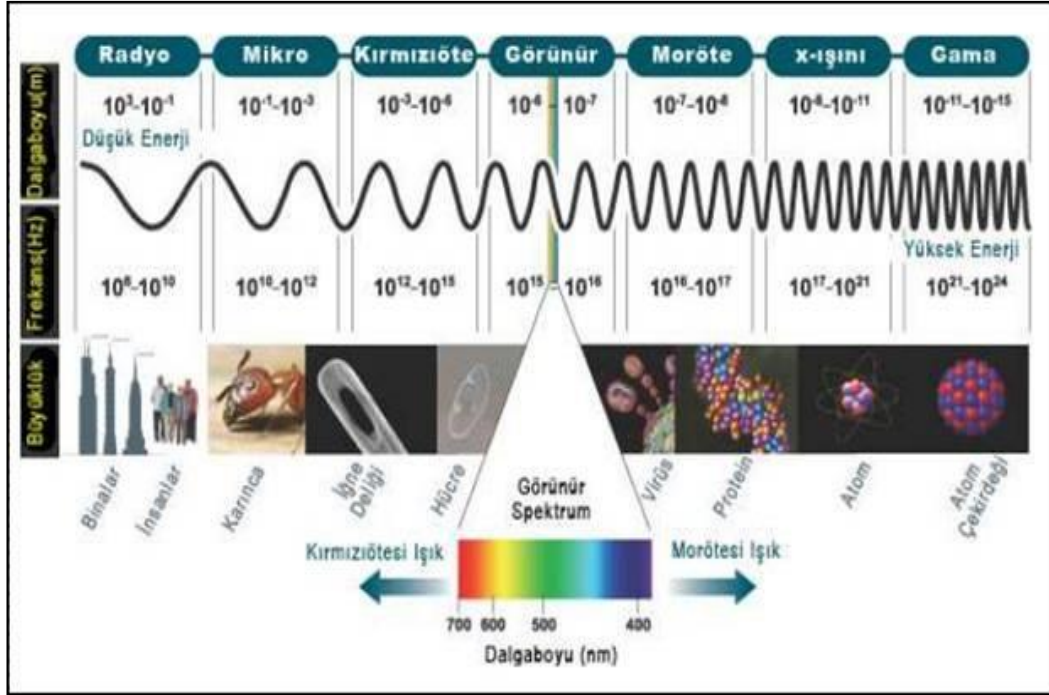


Şekil 3.2. Farklı radyometrik çözünürlüğe sahip aynı görüntü [54]

Örneğin, 8 bit veride her pikselin veri dosya değeri ikinin katları hesaba katılarak 0'dan 255'e kadar uzanırken 6-bit veride her pikselin veri dosya değeri sadece 0'dan 63'ye kadardır. Yani 8-bit veride kaydedilen enerji 256 parlaklık değerine, 6-bit veride ise 64 parlaklık değerine ayrılır. Şekil 3.2 'de aynı yerin farklı radyometrik çözünürlük değerlerinin nasıl olduğu gösterilmiştir.

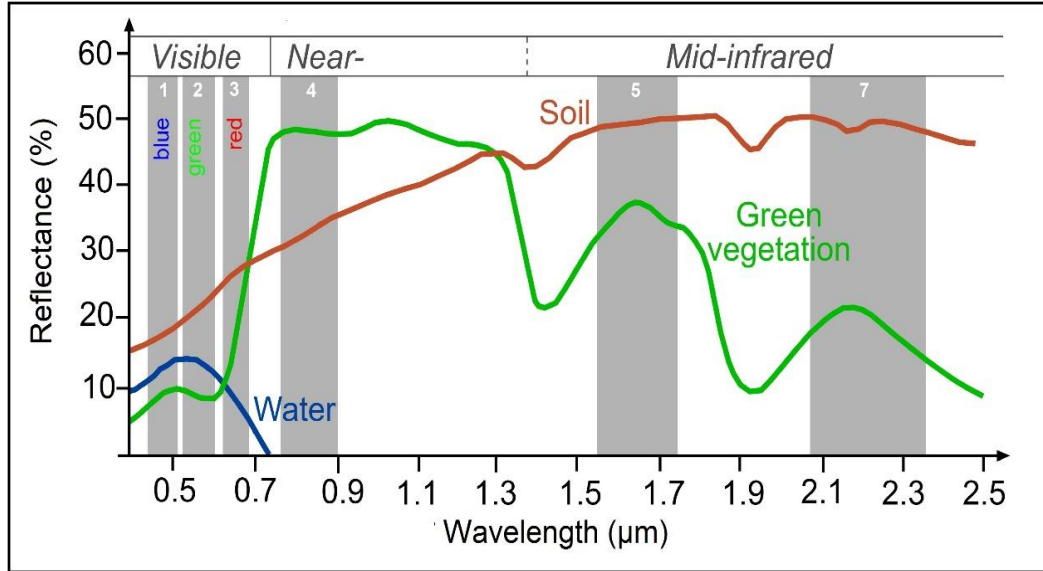
3.3.3. Spektral çözünürlük

Elektromanyetik spektrum, tüm elektromanyetik radyasyonun ve farklı ışınım türlerinin, dalga boyları veya frekanslarını dikkate alarak yerlerini belirtir. Kısaca ifade etmek gerekirse, radyasyonun dalgaboyu sırasına göre dizilişidir. Elektromanyetik spektrumda dalga boyu sıralaması; kozmik ışınlar, gamma ışınları, x ışınları, morötesi ışınlar, güneş ışınları, kızılötesi ışınlar, mikrodalga ışınlar, radio dalgaları ve çok uzun dalgalar şeklindedir. Şekil 3.3 'de elektromanyetik spektrumda, dalga boylarına ve frekanslarına göre bantların isimlendirilmesi ve anlaşılabilirliği açısından da dalga boylarının bazı nesnelere karşılaştırılması verilmiştir.



Şekil 3.3. Elektromanyetik spektruma göre görüntülerin durumu [53]

Spektral çözünürlük ise bir algılayıcının elektromanyetik spektrumda kaydedebildiği dalga boyu aralığı olarak ifade edilmektedir. Kaydedilen bu aralık küçüldükçe spektral çözünürlük artar ve aralık büyüdükçe spektral çözünürlük azalır.



Şekil 3.4. Spektral yansımaya dalgaları [54]

Herhangi bir görüntüdeki ayrıntılar ve nesnelerin farklılığı çoğunlukla dalga boyu aralıkları farklılaştıkça bu farklılığa verdikleri yanıtların karşılaştırılması ile ayırt edilebilmektedirler. Şekil 3.4 'te görüldüğü gibi toprak, su ve bitki örtüsünün dalga

boylarına karşılık yansıma yüzdeleri verilmiştir. Görüntülerin spectral çözünürlükleri, sahip oldukları bant sayısı ve bu bantların genişlikleriyle bağlantılıdır. Bu bant sayısı tek başına spectral çözünürlüğe karşılık gelmeyeceğinden, bu bantların elektromanyetik spektrumdaki durumu da dikkate alınmalıdır.

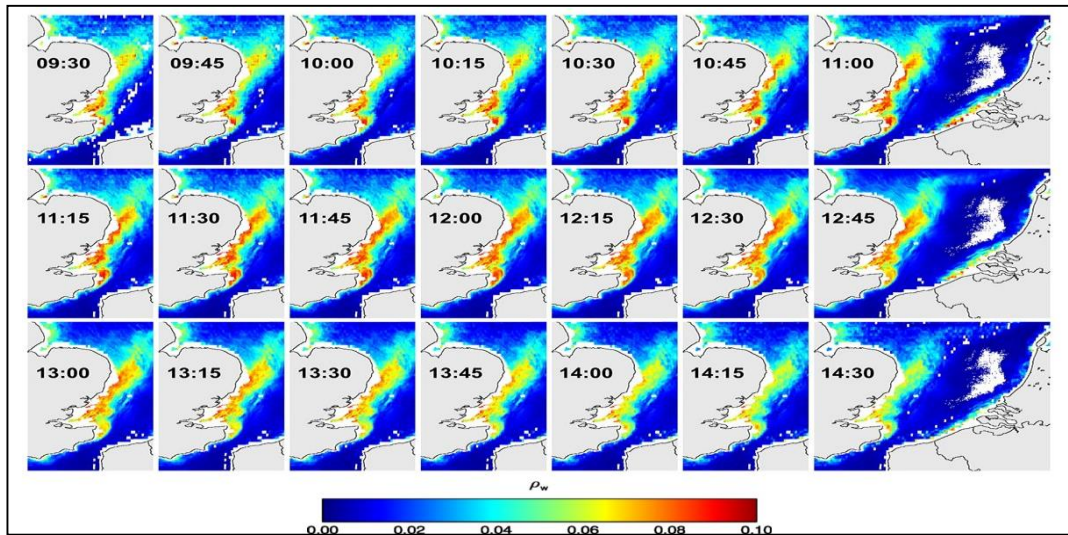
- Yüksek spektral çözünürlük: 220 bant
- Orta spektral çözünürlük: 3-15 bant
- Düşük spektral çözünürlük: 3 bant

Spektral çözünürlüğün iyi olması demek, dar bir dalga boyu demektir. Yani dalga boyu ile spectral çözünürlük ters orantılıdır.

3.3.4. Zamansal çözünürlük

Algılayıcının bir yere ait görüntüyü elde etmesiyle, yine aynı yere ait görüntüyü tekrar alması arasında geçen süre zamansal çözünürlük olarak ifade edilmektedir. Aynı yere ait görüntülerin alınması arasındaki zaman ne kadar kısaysa, zamansal çözünürlük o kadar yüksektir. Zamansal çözünürlük aşağıdaki şekilde gruplandırılabilir:

- Yüksek Zamansal Çözünürlük: < 24 saat – 3 gün
- Orta Zamansal Çözünürlük: 4 – 16 gün
- Düşük Zamansal Çözünürlük: > 16 gün



Şekil 3.5. Uydur görüntülerinde zamansal çözünürlük oluşumu [55]

Uyduların çoğu, ortalama zamansal çözünürlük olan 14 günde bir aynı yerden geçmektedir. Buna ek olarak çok yüksek zamansal çözünürlüğe sahip olan uydular ise

15 dakikada bir aynı yerden geçmektedir. Algılayıcılardaki zamansal çözünürlük, bazı konularda daha da önemli olmaktadır;

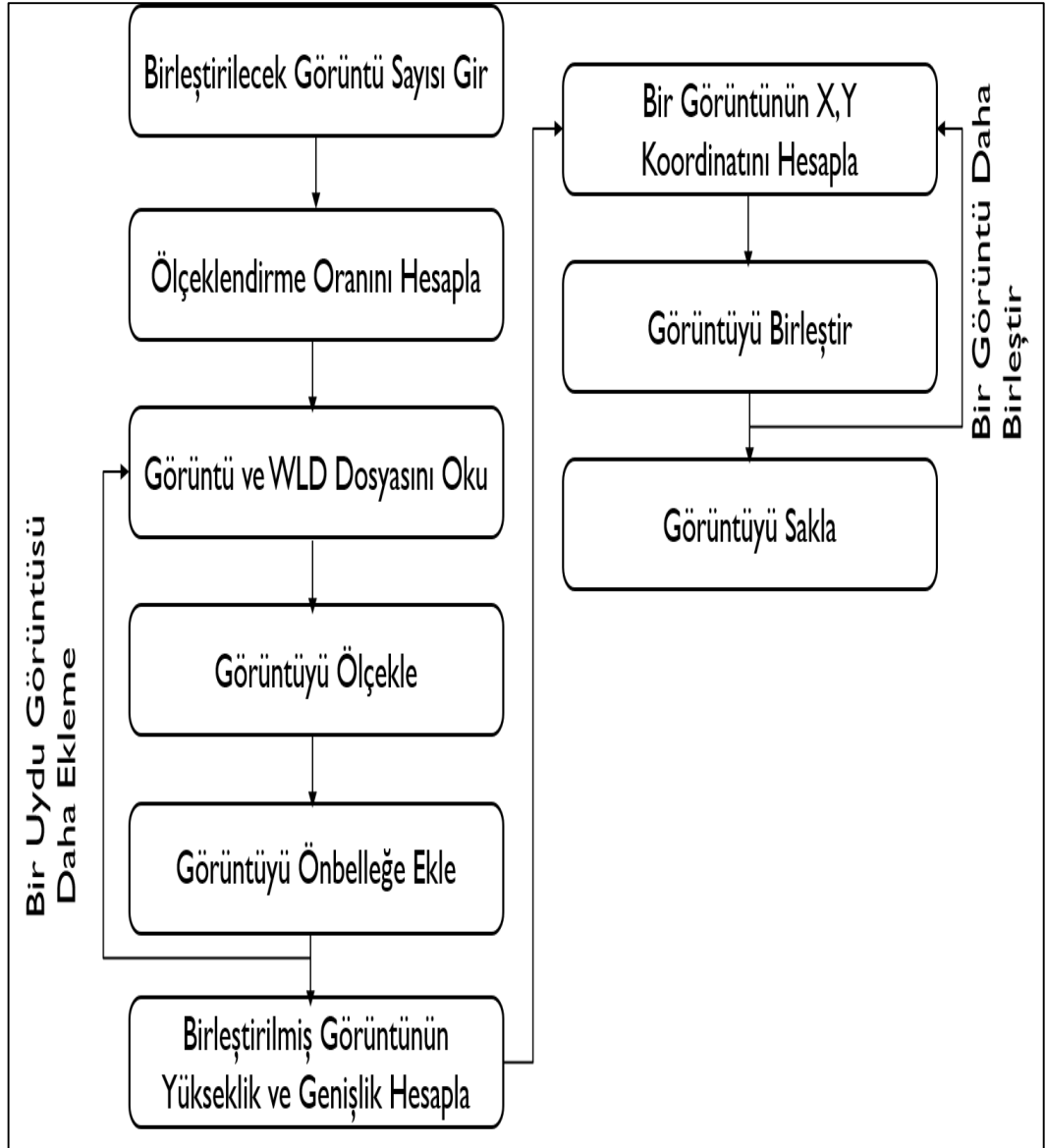
- Kalıcı bulutlardan dolayı, yeryüzünde kısıtlı görüntü alanının olması
- Sel, toprak kayması gibi kısa süreli doğal olayların görüntülenmesi
- Bir bölgedeki orman kayıplarının yıldan yıla değişiminin görüntülenmesi gibi çoklu-zamansal karşılaştırmaların gerektiği durumlarda
- Birbirine benzeyen özelliklerin zaman içindeki değişen görünümünün elde edilmesi

Görüntülerin spektral özellikleri zamanla değişebilir ve bu özellikler çoklu-zamansal görüntüler karşılaştırılarak tespit edilebilir. Örneğin mevsimlerin geçiş zamanlarında bitkilerin çoğu değişim halindedir, bu ince değişiklikleri incelemek ve ayırt etmek, algılayıcının zamansal çözünürlüğüne bağlıdır.

4. METOT

4.1. Donanıma Bağlı Görüntü Birleştirme Mimarisi

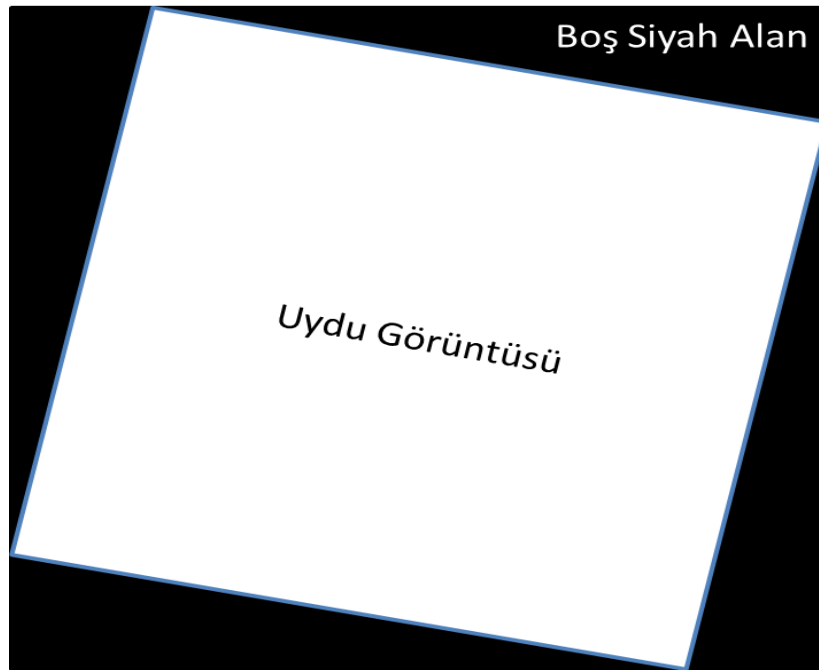
Uydu görüntülerinin çözünürlüklerinin yüksek olması, görüntünün hava şartlarından etkileniyor olması gibi görüntünün birleştirilmesini zorlaştıran problemleri dikkate



Şekil 4.1. Donanıma bağlı uydu görüntüsü birleştirme mimarisi

alan donanımına bağılı, bellek etkin bir birleştirme algoritması geliştirildi (Şekil 4.1). Sistemin bellek ve CPU kısıtları ile görüntülerin boyutları işlemin sonuçlandırılması için eşleştirildi. Bu işlem temel olarak görüntülerin çözünürlüklerinin ayarlanması ile sağlandı.

Uydu görüntüleri temsil ettikleri alanların geniş çaplı olması ve ayrıntıları da gösterebilmesi adına çözünürlüğü çok yüksek görüntülerdir. Akademik çalışma çerçevesinde elde edebildiğimiz uydu görüntüleri, Şekil 4.2 'de gösterildiği gibi, resmi sitede (<http://landsat.usgs.gov/>) “jpeg” ve “tiff” formatında bulunmaktadır.



Şekil 4.2. Mozaik uydu görüntülerinin genel görünümü

Bu formatlara sahip olan uydu görüntülerinin çözünürlükleri de birbirinden çok farklıdır. Örneğin “jpeg” formatında çözünürlük skalası (7000 – 8000) x (7000 – 8000) piksel iken tif formatındaki bir görüntünün çözünürlük skalası (40000 - 45000) x (40000 - 45000) pikseldir. Çözünürlük, uydu görüntülerini işleme konusunda doğrudan etki eden bir faktördür. Bu sebeple işlem yapacağımız formata göre algoritmanız dahil olmak üzere işlemin yapıldığı bilgisayarın sahip olduğu özellikler de değişiklik göstermek durumundadır. Bu çalışmanın amacı olan uydu görüntülerinin birleştirilmesi ve ada görüntülerin vektörel hale getirilmesi için “jpeg” tipindeki uydu görüntülerinin yeterli olduğu görülmüştür. Yine bu çalışmada “jpeg” tipindeki görüntülerin kullanılmasının diğer bir sebebi ise, uygulamanın birçok bilgisayarda

masaüstü uygulaması olarak çalıştırılmasının hedeflenmesidir. “Tiff” formatındaki uydu görüntülerinin çözünürlükleri çok yüksek olduğundan, normal masaüstü bilgisayarlarda çalıştırılması mümkün değildir. “Tiff” formatındaki bir uydu görüntüsünün sadece açabilmek için 12 GB ram’e sahip yüksek konfigürasyonlu bir bilgisayarı kullanmamıza rağmen açamadığı görülmüştür. Günümüz masaüstü bilgisayarları düşündüğümüzde birçok bilgisayar bu denemeyi yaptığımız bilgisayardan daha düşük konfigürasyonlara sahiptir. Kaldı ki “tiff” formatındaki tek bir uydu görüntüsünü açamıyorken, en az 2 görüntüyü birleştirip üzerinde, nesne çıkarımı işlemleri yapmamız beklenmektedir ve bu şartlarda bu beklentiyi karşılamak mümkün değildir.

4.1.1. Belleğin etkin kullanılması

Bu çalışmada kullanılan uydu görüntüleri Landsat uydularından elde edilmiş görüntülerdir. Bu uydudan elde edilmiş görüntüler <http://landsat.usgs.gov> sitesinden elde edilmiştir. Buradan elde edilen her bir uydu görüntüsünün çözünürlüğü en az 7000x7000 piksel civarındadır. Aynı zamanda her bir uydu görüntüsünün RGBA formatındadır, yani her piksel bellekte 32 bitlik bir yer kaplamaktadır. Örneğin aşağıdaki RGBA renk formatında ve 7681x7831 piksel çözünürlüğe sahip olan uydu görüntüsünün bellekte kapladığı alanı hesaplırsak 230 MB olur (32x7681x7831). Şekil 4.3 'te gösterilen gibi tek bir uydu görüntüsü bellekte 230 MB yer kaplayabilmektedir.



Şekil 4.3. 7681x7831 piksel çözünürlüğe sahip gerçek uydu görüntüsü

Bu çalışmada amaçladığımız karaparçalarının sınırlarının çıkarımı işlemi için öncelikle, bir karaparçasının tamamını ortaya çıkaracak kadar olan uydu görüntülerinin birleştirilmesi gerekmektedir. Yukarıdaki uydu görüntüsü Kıbrıs adasına ait olan bir parça uydu görüntüsüdür. Bir Kıbrıs haritasının oluşturabilmek için dört adet uydu görüntüsünün birleştirilmesi gerekmektedir. Örneğini verdiğimiz uydu görüntüsünün üzerinden gidersek ise 920 MB yer kaplamaktadır (4 x 230). yer kaplamaktadır. Ayrıca bu 4 adet uydu görüntüsü belleğe yüklendikten sonra bu görüntülerin de birleştirilmesi için boş bir görüntü alanı da oluşturulması gerekmektedir. Bu görüntü alanı üzerine tek tek parça görüntüler yerleştirilirken, üst üste gelen kısımlarda olabilir. Burada üst üste çakışan kısımları düşünmezsek, boş bir görüntünün boyutu, 4 adet görüntünün yan yana dizilmesidir ki bu durumda bellekten bu uygulama için ayrılması gereken alan ise 1.7 GB olduğu görülmektedir. Birleştirmek istediğimiz uydu görüntüsü sayısı arttıkça, bellekte ayrılacak olan alan da katlanarak artacaktır ve sonunda bellek bu isteklere cevap veremeyecek duruma gelecektir. Böyle bir durumda belleğin optimize kullanılması ve uydu görüntülerinin de bellekte daha az yer kaplaması sağlanmalıdır.

Belleğin optimize kullanılması için, kullandığımız ve artık ihtiyacımızın olmadığı verilerin bellekten boşaltılması gerekmektedir. Bu şekilde bellekte gereksiz yer kaplayan alanları boşaltılıp daha aktif bir şekilde kullanılması sağlanacaktır. Aslında

bu çalışmada kullandığımız dil olan Java 'nın bu işlemleri yapması için kullandığı çöp toplayıcısı (garbage collector) vardır. Ancak bu çöp toplayıcı, referansları tutulan verileri, kullanılıyor olduğunu varsayarak bellekten temizlemez. Bu sebeple bellekten temizlenmesini istediğimiz verilerin referanslarını da kaldırdıktan sonra çöp toplayıcısını kendimiz çağırarak, bellekten bu gereksiz bir şekilde tutulan verilerin boşaltılmasını sağlayabiliriz.

Diğer bir yöntem ise, uydu görüntülerinin çözünürlüklerinin yüksek olması sebebiyle, bellekte fazla yer kaplamasının engellenmesidir. Görüntülerin çözünürlükleri düşürülerek, bellekte daha az yer kaplaması sağlanabilir ve böylece birçok parça görüntü bir araya getirilerek, büyük bir görüntü oluşturulabilir. Ancak burada, problem olan kullanacağımız parça görüntülerinin çözünürlüklerinin ne kadar düşürüleceğinin bilinmemesidir. Amaçlanan minimum küçültme ile maksimum adet görüntünün birleştirilmesidir.

Bu çalışmada, parça görüntü sayısı ne kadar çok artırılsa da görüntüleri birleştiren ve aynı zamanda bellek hatasının alınmasını engelleyen bir bellek optimizasyon yapısı kurulmuştur. İlk olarak bu uygulamanın çalıştığı bilgisayardaki boş olan, kullanılmayan bellek miktarı hesaplanır. Böylece bilgisayar üzerinde çalışan uygulamanın bellekten isteyebileceği maksimum bellek miktarı da hesaplanmış olur. Şimdi hesaplanması gereken bu boş bellek kısmına, birleştirilmek istenen parça görüntü sayısı kadar görüntünün nasıl sığdırılacağıdır. Bunu sağlayabilmek için öncelikle uygulamada kullanılacak uydu görüntülerinin çözünürlükleri incelenir. Yapılan incelemeye göre okunan her bir görüntü için üst limit olarak 8000*8000 piksel çözünürlük varsayılması durumunda hemen hemen tüm uydu görüntüleri de kapsamaktadır. Daha sonra son kullanıcıdan talep edilen birleştirmek istediği parça uydu görüntü sayısı da dikkate alınarak, uydu görüntülerinin çözünürlükleri düşürülmediği durumda bellekte kapladıkları alan miktarı;

X = Birleştirilmek istenen parça uydu görüntü adeti

H1 = Her bir parça uydu görüntünün varsayılan yüksekliği

W1 = Her bir parça uydu görüntünün varsayılan genişliği

Y = Parça uydu görüntülerinin her bir pikselinin kapladığı alan

H2 = Parça uydu görüntüleri birleştirildikten sonraki tahmini yüksekliği

W2 = Parça uydu görüntüleri birleştirildikten sonraki tahmini genişliği

Z = Tüm uydu görüntülerinin bellekte kapladığı alan

X = 30 olduğunu varsayarsak;

$$Z = X \times ((H1 \times W1) + (H2 \times W2)) \times Y \quad (4.1)$$

şeklinde hesaplanır. Yukarıdaki Denklem (4.1) de anlaşıldığı üzere, 30 adet uydu görüntüsünün birleştirilmesi sırasında uygulamanın başarılı bir şekilde birleştirme işlemi için bellekten istediği miktar yaklaşık 12 GB'tır. Günümüz bilgisayarlarını dikkate aldığımızda bu miktar, birçok bilgisayarın karşılayamayacağı bir bellek boyutudur. Burada işlemi biten uydu görüntüsünün bellekten silinmesi de problemi çözmeyecektir. Çünkü görüntü adeti çok fazla ve bu görüntülerin birleştirilerek oluşturacağı görüntünün boyutu da çok yüksektir. Bu sebeple diğer bir yöntem olan uydu görüntülerinin çözünürlüklerini düşürmeliyiz ve çözünürlüklerin hangi oranda düşürüleceğini son kullanıcıya bırakmadan otomatik olarak yapmamız gerekmektedir. Yukarıda da görüldüğü üzere 30 adet parça uydu görüntüsünün bellekte kaplayacağı alan 12 GB'tır. Burada bellekteki, uygulamanın kullanabileceği boş alanın da 4 GB olduğu varsayarak çözünürlük düşürme oranını bulabiliriz. Bu oran Denklem (4.2)'deki gibi;

X = Bellekteki boş alan

Y = Tüm parça görüntülerin bellekteki kapladığı alan

Z = Çözünürlük düşürme oranı

$$Z = X / Y = 4 / 12 \cong 0,33 \quad (4.2)$$

şeklinde hesaplanır. Bu çözünürlük düşürme oranını, uygulama tarafından okunan tüm parça uydu görüntülerine uygulanır ve bir önceki görüntü bellekten boşaltılır, çözünürlüğü düşürülmüş uydu görüntüsü belleğe yüklenir. Bu işlem ise Denklem (4.3) ve Denklem (4.4)'te olduğu gibi;

W = Parça uydu görüntüsünün genişliği

H = Parça uydu görüntüsünün yüksekliği

W1 = Çözünürlüğü düşürülmüş parça uydu görüntüsünün genişliği

H1 = Çözünürlüğü düşürülmüş parça uydu görüntüsünün yüksekliği

R = Çözünürlük düşürme oranı

$$W1 = W \times R = 8000 \times 0,33 \cong 2640 \quad (4.3)$$

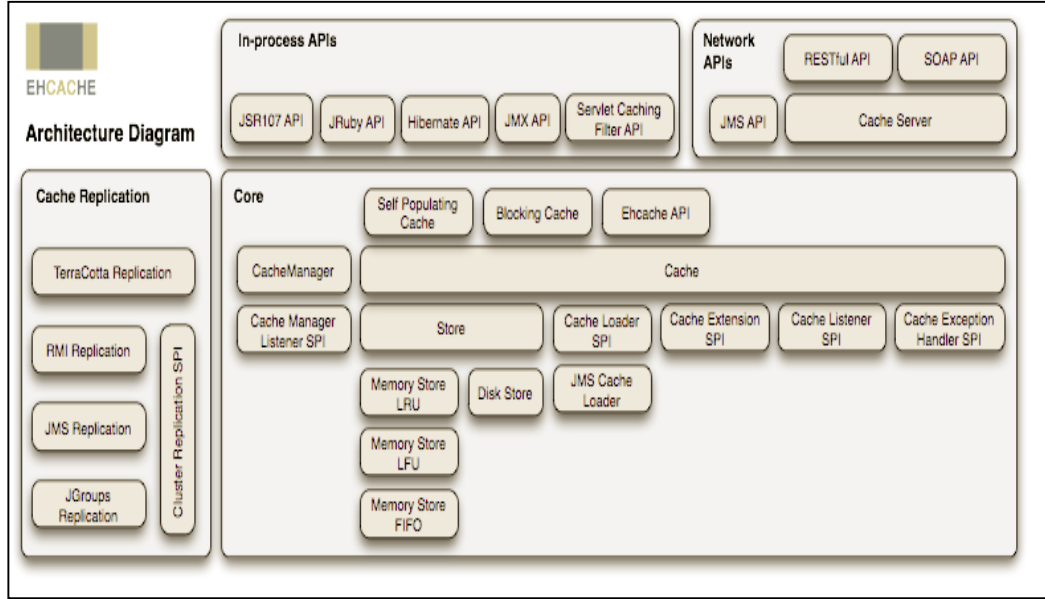
$$H1 = H \times R = 8000 \times 0,33 \cong 2640 \quad (4.4)$$

şeklinde yapılmaktadır. 8000 x 8000 çözünürlüğe sahip olan uydu görüntüsünü bellekten silinir, bunun yerine 2640 x 2640 çözünürlüğe sahip olan uydu görüntüsü yüklenir. Tek bir uydu görüntüsü için buradaki bellekten kazanım ise $4 \times ((8000 \times 8000) - (2640 \times 2640))$ 'dan 217 MB olarak hesaplanmaktadır.

Bu yöntemi kullanarak, görüntü üzerinde hiçbir işlem yapmadan yüklenebilecek uydu görüntüsünden çok daha fazla uydu görüntüsü birleştirilebilmektedir. Ayrıca oromatik bir şekilde çözünürlük küçültme oranı hesaplanmasıyla da, bellek en verimli haliyle kullanılmaktadır.

4.1.2. Uydu görüntülerinde bellek (cache) mekanizması

Caching, uygulama tarafından sıkça ihtiyaç duyulan, içeriği de çok fazla değişmeyen verilerin, her talep edildiğinde asıl kaynağına gitmeye gerek kalmadan, çabuk erişilebilmesi için bu verilerin bir kopyasının ön bellek alınması ve yönetilmesine verilen addır. Çok farklı ön bellekleme (caching) yöntemleri ve bunlara karşılık gelen caching kütüphaneleri bulunmaktadır. Bu uygulamada performansı artırmak ve ölçeklenebilirliği artırmak için java dünyasında çok yaygın bir şekilde kullanılan açık kaynak kodlu caching kütüphanesi olan ehcache [49, 50, 51] kullanılmıştır. Açık kaynak kodlu ehcache mimarisi Şekil 4.4 'te gösterilmiştir.



Şekil 4.4. Ehcache mimarisi [18]

Ehcache, uygulamada kullanılan caching mekanizmasını ve bu mekanizma ile ilgili parametreleri, “ehcache.xml” konfigürasyon dosyasına göre ayarlamaktadır. Bu konfigürasyon dosyasına göre cache tutulmasını istediğiniz veriler için liste gibi bir veri yapısı sunmaktadır. Bu çalışmada kullandığımız liste ve bu listeye ilgili ehcache konfigürasyon dosyası Şekil 4.5 ‘te gösterildiği gibidir.

```
<cache name="ImageCache"
  maxBytesLocalHeap="10m"
  maxElementsOnDisk="10000000"
  eternal="true"
  overflowToDisk="true"
  diskPersistent="true"
  diskExpiryThreadIntervalSeconds="1"
  memoryStoreEvictionPolicy="LFU">
</cache>
```

Şekil 4.5. Ehcache konfigürasyon kodu

Burada amaçlanan belleği daha verimli bir şekilde kullanmak olduğundan, bellek caching yerine disk caching kullanılmıştır. Uygulama tarafından okunan her bir parça uydu görüntüsünün üzerinde çözünürlük düşürme işlemleri yapıldıktan sonra cache’de tutulan listeye eklenmektedir. Daha sonra ise, bu uydu görüntüsü bellekten silinerek,

bellekte sonraki eklenecek parça uydu görüntüler için bellekte yer açılır. İlgili görüntüye uygulama tarafından ihtiyaç duyulduğunda bellekten değil, cache listesinden çekilerek kullanılır. Yukarıda bahsedilen adımlar, uygulama tarafından okunan her bir parça uydu görüntüsü için uygulanmaktadır.

4.1.3. Uydu görüntülerinin birleştirilmesi

Uydu görüntülerinin mozaik görüntülerden oluştuğunu daha önce bahsetmiştik. Bu parça uydu görüntülerinin birleştirilmesi için hangi görüntünün nereye konumlandırılacağı bilgisi gerekmektedir. Bunun için landsat uydu görüntülerinin elde edildiği kaynaktan, konum bilgisi de sağlanmaktadır. Konum bilgisi world file (.wld) içerisinde 6 satır olmak üzere düz metin şeklinde sunulmaktadır.

Satır 1: A: uydu görüntüsünün her bir pikselinin x yönündeki ölçüsü

Satır 2: D: uydu görüntüsünün y yönündeki rotasyon ölçüsü

Satır 3: B: uydu görüntüsünün x yönündeki rotasyon ölçüsü

Satır 4: E: uydu görüntüsünün her bir pikselinin y yönündeki ölçüsü

Satır 5: C: uydu görüntüsünün sol üst köşesinin x koordinatı

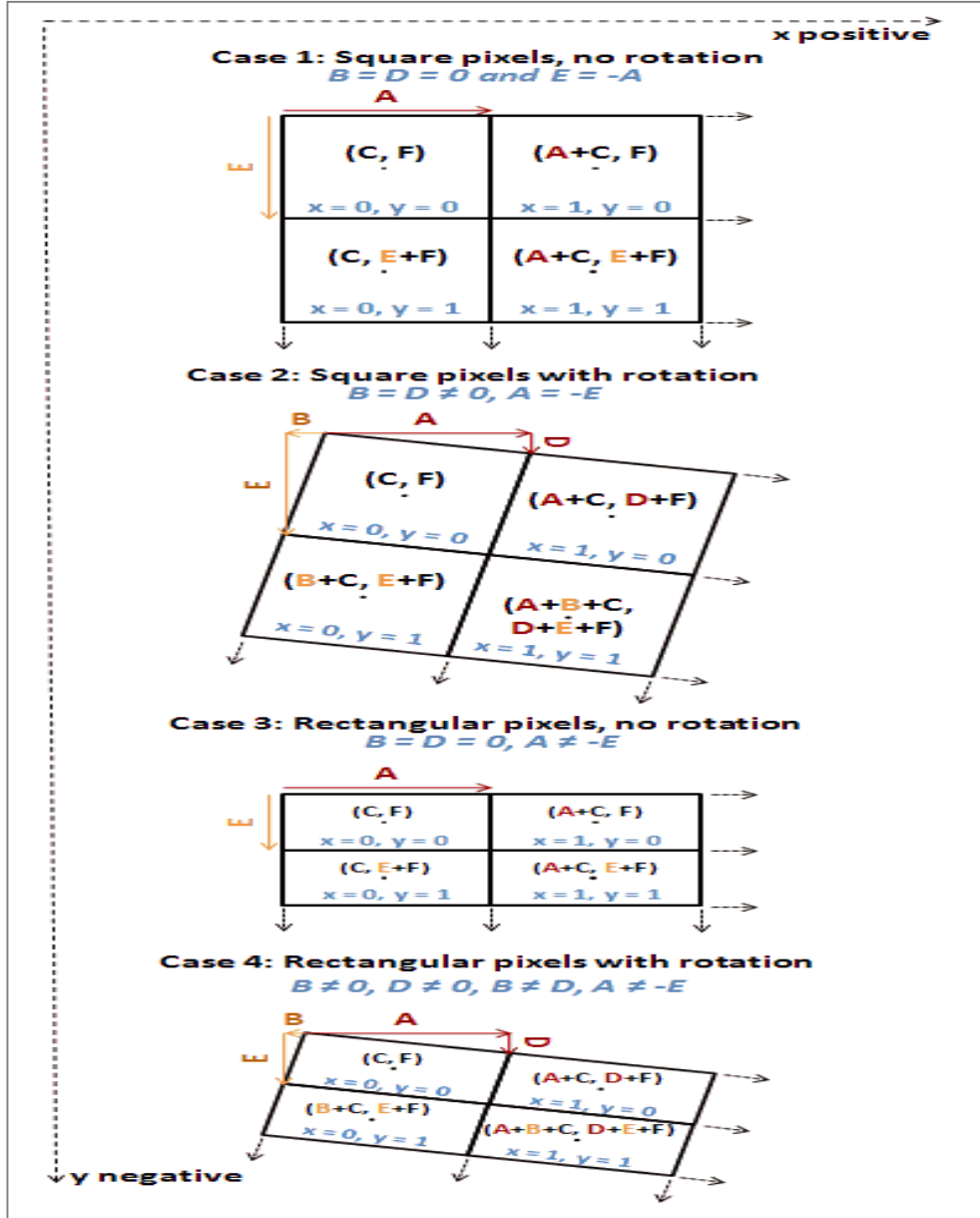
Satır 6: F: uydu görüntüsünün sol üst köşesinin y koordinatı

Bu veriler aşağıdaki şekilde kullanarak, uydu görüntülerinin, harita içerisindeki yerleri Denklem (4.5) ve Denklem (4.6)'da gibi;

$$x' = Ax + By + C \quad (4.5)$$

$$y' = Dx + Ey + F \quad (4.6)$$

şeklinde hesaplanır. Aşağıdaki şekilde de world file (wld) parametrelerinin dört farklı durumdaki davranışlarının nasıl olacağı ile ilgili bilgiler grafiksel olarak Şekil 4.6 'da belirtilmektedir.



Şekil 4.6. World file (wld) parametrelerinin farklı durumları

Ancak bu hesaplanan bu değerler dünya haritası üzerindeki yerlerini piksel cinsinden göstermektedir. Bu çalışmada birleşmiş en az iki görüntüyü dünya haritasından bir kesit şeklinde göstereceğimizden dolayı bu uydu görüntülerinden en soldaki ve en üstteki koordinatlarını baz alarak koordinatları buna göre düzenlemeliyiz. Bunun için öncelikle, okuduğumuz tüm uydu görüntülerine ait wld dosyaları satır satır okunarak, ilgili yerlere atanır. Okunan bu değerler içerisinde koordinatların en soldaki ve en üstteki koordinat değerleri bulunur ve diğer uydu görüntülerinin koordinat değerleri

göreceli olarak bu koordinat değerlerine göre düzenlenir. Formüle edecek olursak;

X = En solu gösteren x yönündeki koordinat değeri

Y = En yukarıyı gösteren y yönündeki koordinat değeri

$X1$ = A parça uydu görüntüsünün x yönündeki koordinat değeri

$Y1$ = A parça uydu görüntüsünün y yönündeki koordinat değeri

$X2$ = A parça uydu görüntüsünün x yönündeki yeni koordinat değeri

$Y2$ = A parça uydu görüntüsünün y yönündeki yeni koordinat değeri

$$X2 = X1 - X \quad (4.7)$$

$$Y2 = Y1 - Y \quad (4.8)$$

Denklem (4.7) ve Denklem (4.8)'deki gibi formüle edilebilir. Uygulama tarafından okunan her bir parça uydu görüntüsü için yukarıdaki işlem uygulanır ve wld dosyasından gelen koordinatlar, birleştirme ortamındaki koordinatlara çevrilmiş olur. Buradan sonra yapılması gereken işlem ise, tüm parça görüntülerin birleştirileceği, onları barındırabilen boş bir görüntü oluşturulmalıdır. Bu boş görüntünün genişliğinin ve yüksekliğinin ne olduğunun hesaplayabilmek için ise, parça uydu görüntülerinin uç nokta koordinatlarının hesaplanması gerekmektedir. X -yönündeki en sol ile en sağdaki koordinat bulunur ve bunların farkında boş görüntünün genişliği hesaplanır. Y -yönündeki en aşağıdaki ve en yukarıdaki koordinat bulunur ve bunların farkından boş görüntünün yüksekliği hesaplanır. Bu hesaplanan yükseklik ve genişliğe göre boş bir görüntü oluşturulur. Her bir parça görüntü boş görüntünün üzerine boyanır. Bu boyama işlemleri yaparken, çakışan kısımlar için herhangi bir harmanlama metodu kullanmazsak, uydu görüntülerinin mozaik yapısından gelen, görüntünün dışında kalan siyah renkteki kısım bir önceki uydu görüntüsünü kapatacaktır. Koordinat tabanlı birleştirme Bölüm 4.1.4.2'de detaylı olarak verilmiştir.

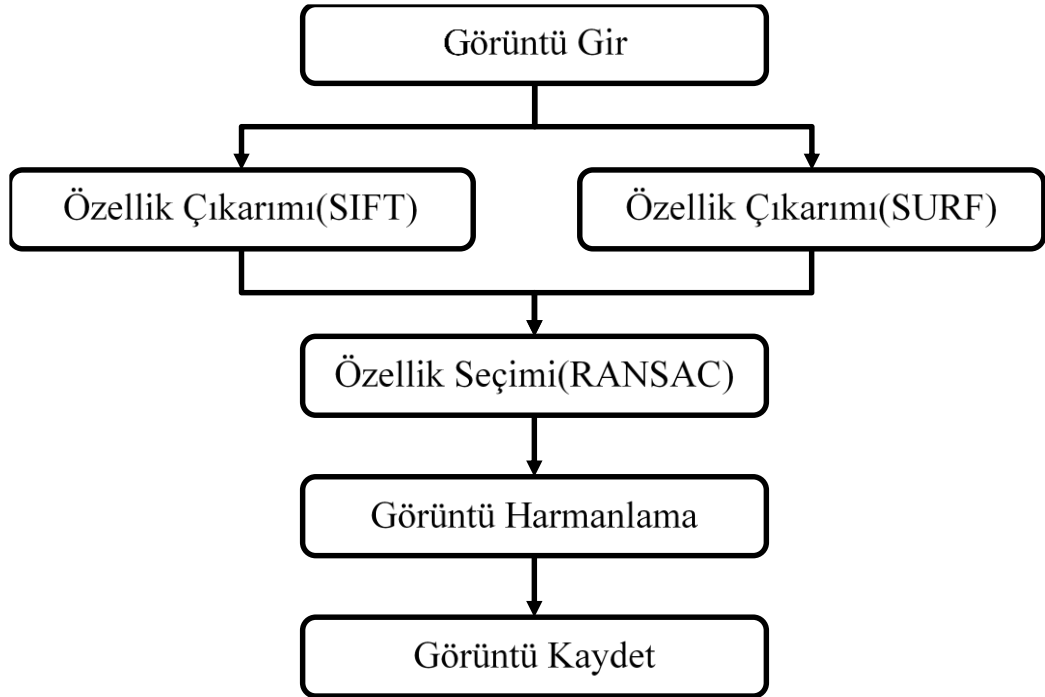
4.1.4. Hibrit görüntü birleştirme algoritması

Özellik tabanlı görüntü birleştirme algoritmaları ile geometrik kesme tabanlı görüntü birleştirme mimarisi birleştirildi.

4.1.4.1. Özellik tabanlı birleştirme yaklaşımı

Çalışmamızda, uzaktan algılama yöntemiyle elde edilen (LandSat-8 uydu görüntüleri) aynı nesneye ait kısmi çakışan multi-spektral uydu mozaik görüntülerin, özellik-tabanlı görüntü birleştirme yaklaşımıyla birleştirilmesi için bir algoritma ve yöntem geliştirilmiştir. Yöntemimiz birden fazla adımdan oluşmaktadır. Yöntemin temelini (i) SURF ve (ii) SIFT feature tanıma ve tanımlama algoritmaları oluşturmaktadır. Adımlar aşağıdaki şekilde sıralanmış ve Şekil 4.7 'de de genel olarak gösterilmiştir.

- i. Herhangi iki kısmi çakışan uydu görüntüsü girdisi,
- ii. Özellik çıkarımı (Feature extraction): Anahtar noktaların belirlenmesi (Interest-point or key-point detection),
- iii. Özellik Eşleştirme (Feature matching): Anahtar noktaların ilişkisel eşleştirilmesi (Correlation matching),
- iv. Rastgele Özellik Seçimi (RANSAC),
- v. Görüntü harmanlama (Gradient blending).



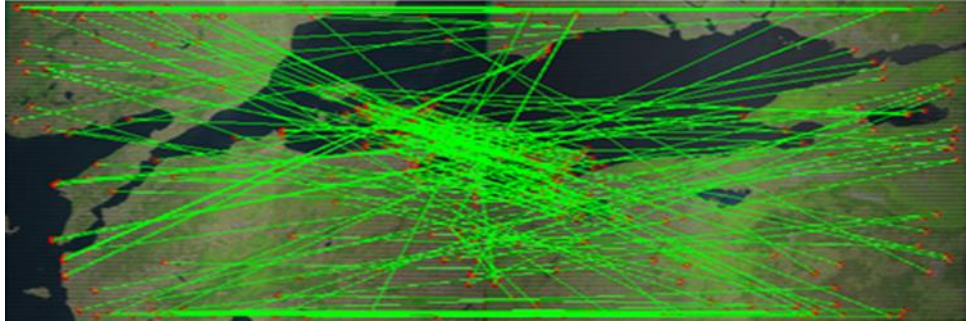
Şekil 4.7. Özellik tabanlı birleştirme algoritma şeması

Uygulama sonuçları ise, Marmara denizine ait LandSat-8 orjinal uydu görüntüleri yukarıdaki teknik ile işlenebilir hale getirildikten sonra önerilen yöntem ile birleştirilip aşağıdaki sonuçlar elde edilmiştir. Kullanılan görüntülerin özellikleri; 8 multispektral

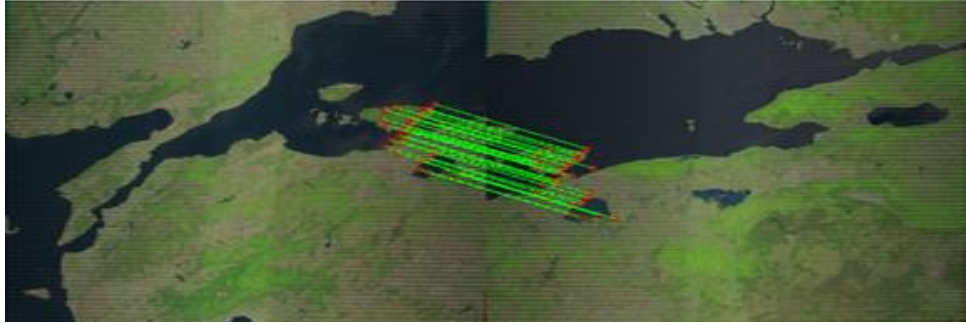
banta sahip Landsat-8 Operasyonel Arazi Görüntüleyici (Operational Land Imagery-OLI) (440 - 2200 nm), mekansal çözünürlük 30 m' dir.



Şekil 4.8. LandSat 8 OLI uydu mozaik görüntüler



Şekil 4.9. SURF ile özellik çıkarım ve eşleştirme



Şekil 4.10. RANSAC'ın uygulanması ve gürültülerin elimine edilmesi



Şekil 4.11. Aşamalı harmanlama (blending) ile birleştirilmiş görüntü

Şekil 4.7’de SIFT ve SURF yöntemleri ile Şekil 4.8, Şekil 4.9, Şekil 4.10 ve Şekil 4.11 ’de verilen iki mozaığe ait birleştirme zamanları ve standart dağılımları verilmiştir.

Tablo 4.1. SIFT ve SURF performans sonuçları

Algoritma	Ortalama Zaman	Standart sapma
SIFT	9067ms	723ms
SURF	8247ms	537ms

Yapılan testlerde özellik tabanlı birleştirme yönteminin bir takım eksikleri olduğu saptanmıştır. Bunlar:

- Haritanın köşe koordinatlarının U.S. Geological Survey (USGS)’den elde edilen görüntülerin köşe koordinatları olmaması, verilen koordinatların görüntü içinde eğik şekilde gelen haritanın köşe koordinatları olması ve boylam (longitude)’ın liner bir şekilde değişmemesi, elde mevcut bulunan görüntü içindeki haritanın gerçek enlem-boylam koordinatlarından boylam’ın tam olarak tespit edilmesini güçleştirmektedir.
- Birleştirilecek olan harita/görüntü sayısı arttıkça ihtiyaç duyulan bellek miktarı artmakta ve bellek sıkıntısına neden olmaktadır. Sanal bellek olarak SSD disklerin kullanılması ile bu problem ortadan kalkmakta ve standart disklere nazaran daha hızlı bir çalışma elde edilebilmektedir.
- Bir diğer problem ise uydu görüntülerinin çözünürlüğünün çok yüksek olması (yaklaşık 7000x7000) sebebiyle, yüksek işlem gücü ve bellek gerekmektedir.
- Ayrıca kuzeye çıkıldıkça boylam mesafesinin hızlı bir şekilde değişmesi farklı bölgelerde farklı projeksiyonların ve bölgelere özel ayrı ayrı algoritmaların kullanılmasını gerektirmektedir.

Yukarıdaki problemlere çözüm olabilecek koordinat tabanlı yeni bir birleştirme yaklaşımı geliştirilmiştir.

4.1.4.2. Koordinat tabanlı birleştirme yaklaşımı

Öncelikle yapılması gereken, enlem ve boylamın iki boyutlu düzlemdeki (x, y) koordinatlara çevrilmesidir. Bunun için her bir pikselin, enlem boylamda karşılığını bulmak gerekmektedir. Bunun için görüntünün en uç noktalarına ait enlem ve boylam değerlerinin kullanmamız gerekmektedir. X yönündeki her bir pikselin enlem karşılığı

Denklem (4.9) olduğu gibi;

$$X = \text{image_width} / (\text{NE_latitute} - \text{SW_latitute}) \quad (4.9)$$

şeklinde hesaplanırken, Y yönündeki her bir pikselin, boylam karşılığı Denklem (4.10)'da olduğu gibi ;

$$Y = \text{image_height} / (\text{NW_longitute} - \text{SE_longitute}) \quad (4.10)$$

şeklinde hesaplanmaktadır. Sistem üzerine yüklenen her bir görüntü için bu işlemleri ayrı ayrı yapıyoruz ve hata oranını en aza indirmek için en hassas (x, y) değerini bulup ona göre işlem yapmamız gerekmektedir. Bu işlemden sonra görüntüler birleştirilmeden önce, birleştirilmiş son görüntünün genişlik ve yüksekliğini buluyoruz. Bunu bulmak için sisteme yüklenen tüm görüntülerin en soldaki, en sağdaki, en üstteki ve en alttaki noktalarını bulmalıyız ve bunları piksel değerlerine çevirmeliyiz. Bu işlemi yaptıktan sonra ise görüntülerin birleştirildikten sonraki yükseklik ve genişliğini bulmuş oluyoruz.

Yukarıdaki dönüşüm işlemlerini yaptıktan sonra görüntüleri birleştirme adımına geçilir. Bunun için her bir resmin sol üst köşe koordinatının büyük görüntüde (birleştirilmiş) nereye geleceğini bulmak gerekmektedir. Her bir görüntünün sol üst köşe koordinatını (SW_latitute, NW_longitute) (x,y) ikilisine çevirdikten sonra görüntüleri birleştiriyoruz. Görüntüleri bu şekilde birleştirdikten sonra uydu görüntülerinin dışında bulunan siyah kesimler birleştirme noktalarında birleştirilen diğer resmin görüntülerini kapatmaktadır. Bu problem, tam o siyah görüntüyle üst üste gelen kısımlarda pikselin RGB değerlerini uygun bir şekilde birleştirerek çözülebilir. Her bir görüntünün her bir pikseli için elimizde bulunan bilgi, RGBA'dır (kırmızı, yeşil, mavi renklerinin karışımından oluşan renk uzayı ve alpha değerleri). Üst üste görüntülerin çakıştığı (overlap) yerlerde, siyah pikselleri değil RGB formatındaki pikselleri almamız gerekmektedir. Bunun için harmanlama türlerinden (blendig mode) olan "lighten" metodu kullanılmaktadır. Uydu görüntülerinin çakıştığı yerlerde piksel değerlerini belirleyen lighten metotunun formülasyonu;

R = Sonuç olarak elde edilen pikselin değeri

S = Alttaki görüntünün piksel değeri

Sr,Sg ve Sb = Sırası ile alttaki görüntünün RGB değerleri

D: S'nin üstüne harmanlanacak olan görüntünün piksel değeri

Dr,Dg ve Db = Sırası ile üstüne harmanlanacak olan görüntünün RGB değerleri

$$R = \max (S,D) \text{ ise } R = [\max(Sr,Dr), \max(Sg ,Dg), \max(Sb,Db)] \quad (4.11)$$

Denklem (4.11)'deki gibi gerçekleşmektedir. Bu işlemi yaptıktan sonra birleştirilmiş düzgün bir görüntü görebiliriz (Şekil 4.12 'ye bakınız).



Şekil 4.12. Lighten metodu kullanılarak birleştirilmiş uydu görüntüsü

Eğer harmanlama türlerinden olan “lighten” metodunu kullanmasaydık elde edeceğimiz görüntü Şekil 4.12 'de görüldüğü gibidir. En son eklenen görüntü, etrafındaki siyah piksellerle birlikte kendini korurken diğer parça görüntüler bir sonraki görüntülerin siyah piksel değerleri altında kalmıştır. Şekil 4.12 'deki belirlenen adayı gösteren tüm görüntü Şekil 4.14 'teki parça görüntülerinden elde edilmiştir. Gerçek dünya koordinat değerleriyle temsil edilen ada uydu görüntüsü artık bilgisayar ortamında ekran koordinat değerlerine çevirilerek tek bir görüntü haline getirilmiş olacaktır.



Şekil 4.13. Lighten metodu kullanılmadan birleştirilmiş uydu görüntüsü



Şekil 4.14. Birleştirilmiş uydu görüntüsünü oluşturan parça mozaik görüntüler

4.2. Birleştirilmiş Görüntülerden Nesne Çıkarımı ve Vektörel Modelleme

Çalışmanın bu bölümünü birleştirilmiş uydu görüntüsünden nesne çıkarımı mimarisi oluşturmaktadır ve bu kısım mozaik görüntülerin birleştirilmesini gerçekleyen mimari üzerine kurulmuştur. Çıkarılan nesnelere koordinat bilgileri ile poligon, çizgi dizisi (linestring) şeklinde modellenmekte ve mekânsal veritabanlarında depolanıp zaman-mekânsal analizler için hazır hale getirilmektedir.

4.2.1. Sezgisel yaklaşımla yarı otomatik nesne çıkarım mimarisi

Uydu görüntülerinden nesne çıkarımının otomatik yapılması çok zor ve hatta imkânsızdır. Genelde yeşil, kahverengi ve mavi tonlarından oluşan görüntülerde özelliklerin çıkarımı çok zordur. Buna ek olarak nesnenin nerede başlayıp nerede bittiğinin tespiti de otomatik olarak yapılamaz. Mesela Atatürk Bulvarı nereden nereye gitmekte keyfi olarak tanımlanmıştır. Görsel olarak çıkarımı yapılamaz.

- Nesne çıkarımının uydu görüntülerindeki bu tür problemlerden dolayı yeni bir heuristic (sezgisel) yaklaşımla bir mimari gerçekledik.
- Kullanıcı çıkarılacak nesnenin hangi mozaikleri kapsadığını biliyor ve seçiyor şeklinde varsayım yaptık. Bunu ileriki çalışmalarda hangi mozaiklere ihtiyaç olacağını otomatik belirleme şeklinde geliştireceğiz.
- Bu mozaikleri seçen kullanıcıya sistemimiz birleştirilmiş ve istenen nesneyi içeren tek parça uydu görüntüsü sağlamaktadır.
- Kullanıcı sisteme istenen nesnenin sınırlarını interaktif olarak fare aracılığıyla yaklaşık çizerek vermektedir. Buna course-grained yaklaşımı da denebilir. Fine-grained yaklaşım da ise sistem piksel bazlı ve koordinat bazlı detayda nesnenin sınırlarını vektörel olarak modellemektedir.

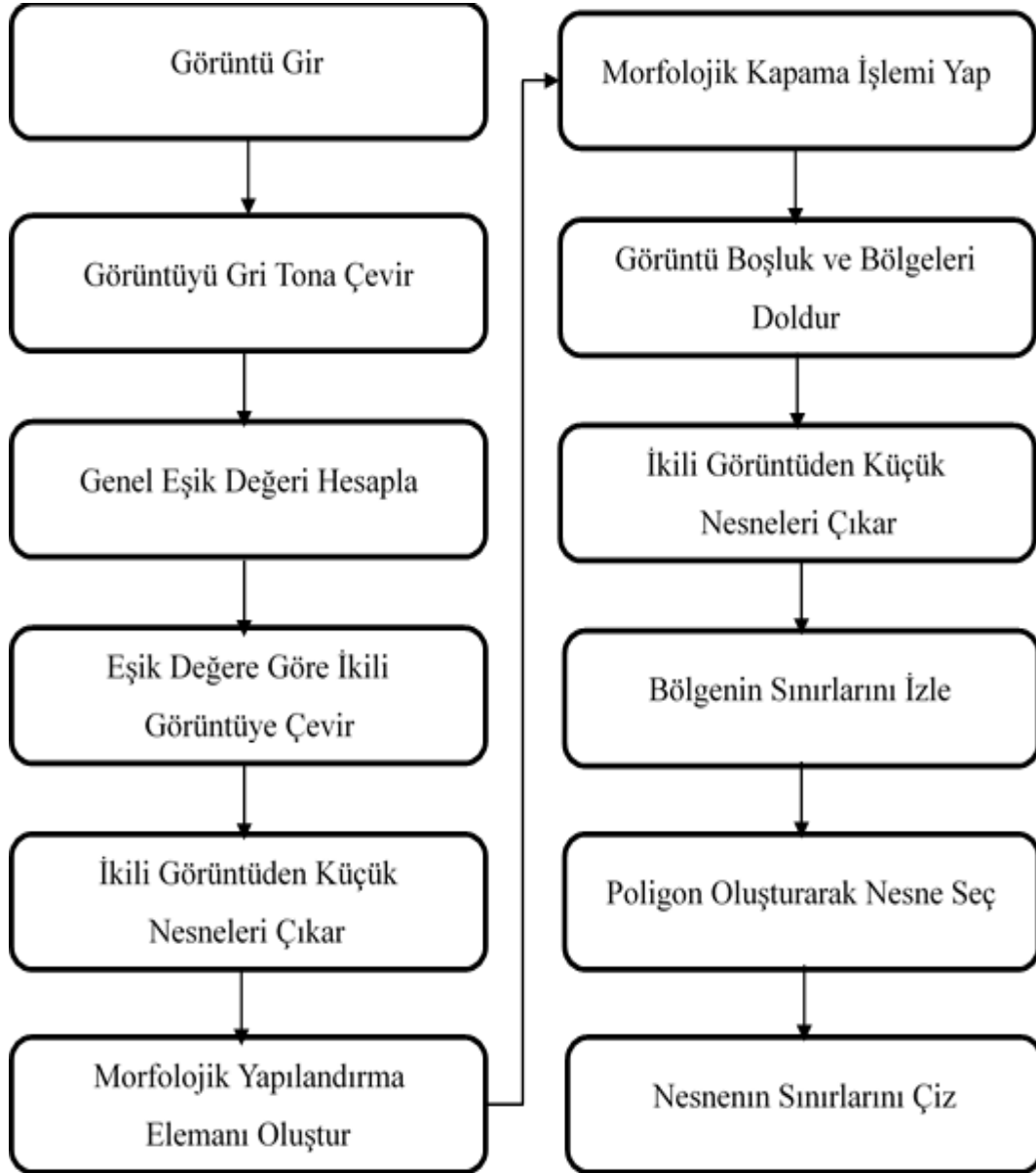
Raster mozaik uydu görüntülerinden Bölüm 2.1’de verilen yöntemle tek/bütün hale getirilen görüntü aşağıda detayları listelenen adımlarla önce yarı otomatik nesne çıkarımı sağlanmış daha sonra çıkarılan nesne vektörel olarak modellenenebilir hale getirilmiştir (Şekil 4.15’e bakınız).

- Görüntünün gri seviyeye çevrilmesi; daha sonra uygulanacak olan alt adımların sonucunu artırmak için görüntüdeki her bir piksel değerinin kırmızı (red), yeşil (green) ve mavi (blue) (RGB) değerlerinin aritmetik ortalamasının alınmasıyla gri seviye görüntü elde edilmiştir.

- Görüntünün yumuşatılması; görüntü üzerinde konvolüsozyon matrisi gezdirilerek; kabartma, kenar algılama, veya bulanıklık keskinleştirme gibi işlemler gerçekleştirilebilir. Ortanca filtre yardımı ile görüntü üzerinde meydana getirilerek görüntünün yumuşatılması sağlanmıştır.
- Görüntünün ikili seviyeye çevrilmesi; görüntüyü gri seviyeye dönüştürme işlemi ile görüntüdeki piksel değerleri her kanal için 0-255 aralığında değişen değerler almaktadır. Bilgisayar tarafından kolay algılanabilecek olan ikili hale getirilmesi için threshold/eşikleme uygulanması gerçekleştirilmektedir. Eşikleme değerinin altında kalan değerler için görüntünün piksel değeri 0 (siyah) olarak atanırken eşik değerinin üzerindeki değerler için 1 (beyaz) atanması işlemi gerçekleştirilmiştir. Eşik değerinin görüntünün ışık dağılımı, kontrast gibi özelliklere göre tüm görüntü için global bir değerden ziyade lokal bir değer olması önemlidir. Biz de görüntüleri ikili hale çevirmek için Otsu eşikleme yönteminden yararlandık [48].
- Morfolojik işlemlerle küçük objelerin elemine edilmesi; görüntü üzerinde aşınma işleminin hemen ardından genleşme işlenmesi uygulanması sonucu açma işlemi elde edilir. Görüntü içerisindeki nesnelere ve nesnelere arasındaki boşluklar yapısal elemanın büyüklüğüne göre temizlenir. Görüntü üzerinde kalan nesnelere orijinal görüntüdeki şekillerinden biraz daha küçük hale gelir. Açma işlemi ile birbirine yakın iki nesne görüntüde fazla değişime sebebiyet vermeden ayrılmış olurlar.
- Poligon oluşturulması için sınırların izlenmesi; sınırlar, sayısal görüntüdeki şeklin çevresini gösteren çizgilerdir. Raster bir ada görüntüsünde sınır çizgileri takip edilerek kapalı bir poligon elde edilebilir. Sınır değerleri elde edilirken ikili görüntüde sulu bölgeler arka plan olarak kabul edilir ve adayı temsil eden piksellerden arka planla komşu olan piksel sınır pikseli olarak işaretlenir. Bu sınır pikselleri 3x3 boyutlu bir maskenin görüntüsü üzerinde gezdirilmesi ile elde edilir. Yani sınır pikselleri 255, sınır olmayanlar ise 0 olarak kodlanır.

Bu aşamadan sonra sınır pikselleri takip edilerek oluşturulacak poligonun koordinat değerleri kaydedilir. Oluşturulacak poligon nesnesinin başlangıç ve bitiş noktaları aynı olmalıdır. Bunun için öncelikle adanın sınırı üzerinden bir nokta başlangıç noktası olarak seçilir. Amaç sınırdaki bir sonraki pikselin bulunmasıdır. Sonraki piksel, ilgilenilen piksele komşu olan bir pikseldir. Sınır çıkarma algoritması geldiği yön hariç yedi izleme yönüne izin verir. Aslında sonraki piksel, önceki piksel ve ilgilenilen

piksel tarafından tanımlı yönde bulunan pikseldir. Eğer sonraki piksel bu yönde yoksa bu yönle (referans yönü) arasında minimum açığa sahip olan piksel sonraki pikseldir. Bu pikselin koordinatı şeklin sınırını ifade eden koordinat değerleri arasına dahil edilmelidir. Şeklin sınırında bulunan bir sonraki pikseli bulma işlemi başlangıç noktası olarak seçilen piksele rastlanana kadar devam ettirilir. Alt adımlara ait ara çıktılar Şekil 4.16 'da verilmiştir.



Şekil 4.15. Yarı otomatik nesne çıkarım mimarisi akış diyagramı



Şekil 4.16. Nesne çıkarım mimarisini alt adımlarının ara çıktıları

4.2.2. Vektörel modelleme ve zaman-mekânsal analizler

Birleştirilen görüntülerden nesne çıkarım yapılmakta ve bu nesnelere koordinat bilgileri ile poligon, çizgi dizisi (linestring) şeklinde modellenmekte ve mekânsal veritabanlarında depolanıp zaman-mekânsal analizler için hazır hale getirilmektedir. Bu anlamda PostgreSQL'in PostGIS eklentisi ile geliştirilmiş ve sistemimize entegre edilmiştir. Birleştirilen uydu görüntülerinden çıkarılan nesnelere OGC standartlarında tanımlanan vektör modellere dönüştürülerek (nokta, poligon, çoklu-çizgi, çizgi, vs.) veritabanlarına nesne olarak nokta detayında koordinat bilgileri ile kaydedilmektedir. Tipik veri tabanları nümerik ve karakter veri tiplerini anlayabilirken, mekânsal veri tabanlarında mekânsal veri tiplerini (nokta, çizgi ve poligon) analiz edebilir. Bu veri tipleri geometri ya da özellik olarak bilinir. Mekânsal veriler; nokta, çizgi ve alansal olarak ifade edilebilen coğrafi haritalar, nehirler, yollar, adalar veya piksel gruplarından oluşan uydu görüntüleri, sayısal yükseklik modelleri ve hava fotoğrafları

olabilir. Mekânsal veri tabanları üzerinde Kocaeli'nin nüfusu kaç, Kocaeli'de toplam kaç tane ilçe var, A harfi ile başlayan ilçelerini listeleyin gibi tipik SQL sorgularına ek olarak mühendislik fakültesine en yakın iki restoran nerededir (yakınlık/proximity), Kocaeli'de şehirlerarası otobüs terminali hangi ilçededir (içerme/containment), Türkiye'ye komşu ülkeler hangileridir (bitişiklik/adjacency) ve Kocaeli'nin hangi ilçeleri üzerinden tren yolu geçer (kesişim-örtüşme /intersection-overlap) gibi mekânsal sorgulamalar yapılabilir.

Kullanıcı interaktif arayüzden veritabanında depolanan nesnelere üzerinden zaman-mekansal analizler yapabilmekte ve sonuçları kullanıcının ekranından sunabilmektedir. Zaman-mekansal analizler, alan ve çevre gibi sorgular olabileceği gibi; çakışma, kesişme gibi topolojik sorgular da olabilmektedir.

Uydu görüntüleri çözünürlükleri yüksek olması sebebiyle, bellekte fazla yer kaplamaktadır. Bu sebeple sisteme eklenen uydu görüntülerinin çözünürlükleri nin bir miktar azaltılması gerekmektedir. Çözünürlük azaltma oranları, sistemde birleştirilmek istenen uydu görüntü sayısı ile doğru orantılıdır. Bu oran birleştirilmiş uydu görüntüsünden çıkarılan nesnenin alan hesabını da eklemektedir.

Tablo 4.2. Çözünürlük Oranının Çıkarılan Nesne Alanına Etkisi

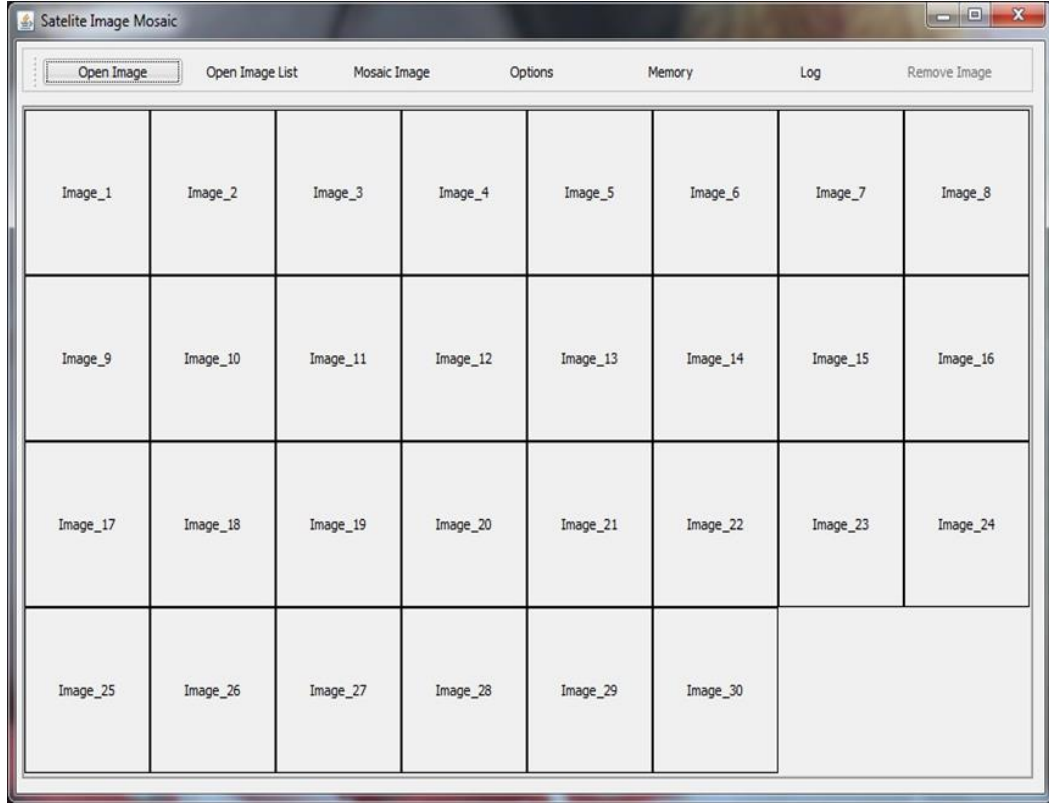
Birleştirilmek İstenen Uydu Görüntü Sayısı	Çözünürlük Oranı	Azaltma	Poligonun Alanı(km ²)
20	0,1048		9210
30	0,0699		9208
40	0,5242		9190
50	0,0419		9155
60	0,0349		9143

Yukarıdaki Tablo 4.2 'de görüldüğü gibi, birleştirilmek istenen uydu görüntüsü sayısı arttıkça, çözünürlük oranı artmakta ve Kıbrıs Adası'nın gerçek yüzölçümünden (9271 km²) uzaklaşmaktadır.

5. UYGULAMA

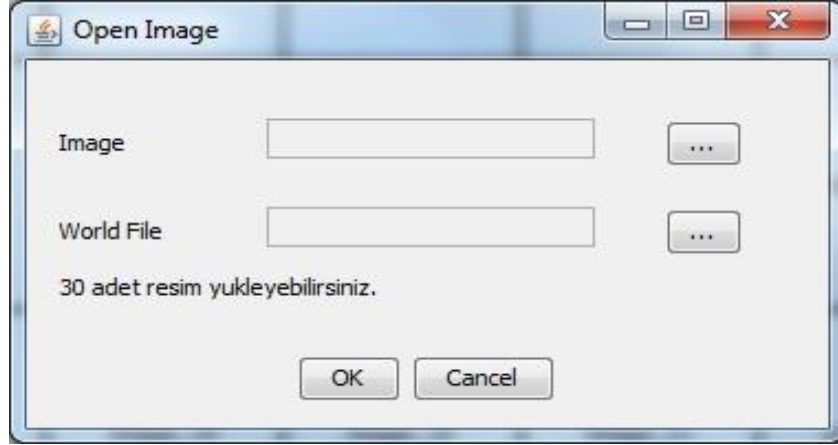
5.1. Geliştirilen Uygulama

Yukarıda anlatılan geliştirdiğimiz görüntü birleştirme ve nesne çıkarım yöntemleri ile ilgili son kullanıcıların kolayca kullanabileceği kullanıcı dostu arayüzler geliştirilmiş ve gerçek uydu görüntüleri üzerinde başarımlı testleri yapılmıştır. Kullanıcı arayüzünde var olan menülerin fonksiyonallikleri detaylı olarak bu bölümde açıklanacaktır. Tasarlanan sistemin genel kullanıcı arayüzü Şekil 5.1'deki gibidir.



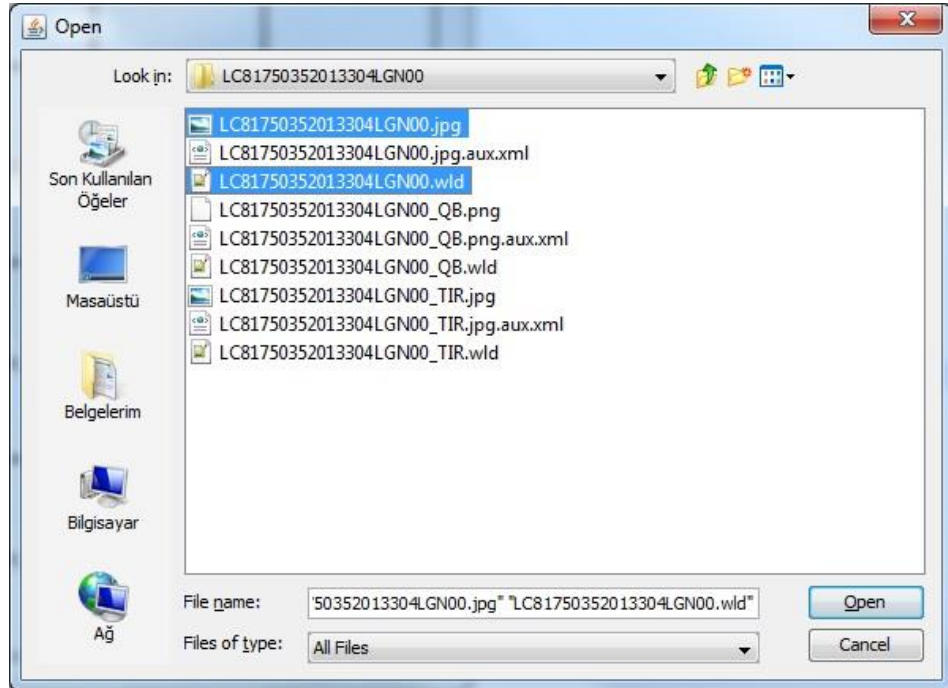
Şekil 5.1. Uygulama anasayfasının arayüzü

Kullanıcı arayüzündeki menüde yedi adet görev tanımlıdır. “Options/Seçenekler” kısmından birleştirilecek mozaik görüntü adedi belirlenmektedir. Başlangıç olarak bu sayı şekilde de görüldüğü gibi otuzdur. “Open Image/Görüntü Aç” yardımı ile birleştirilecek mozaiklere ait görüntüler ve bu görüntülerin wld (World file) dosyaları seçilir (Şekil 5.2 ve Şekil 5.3).

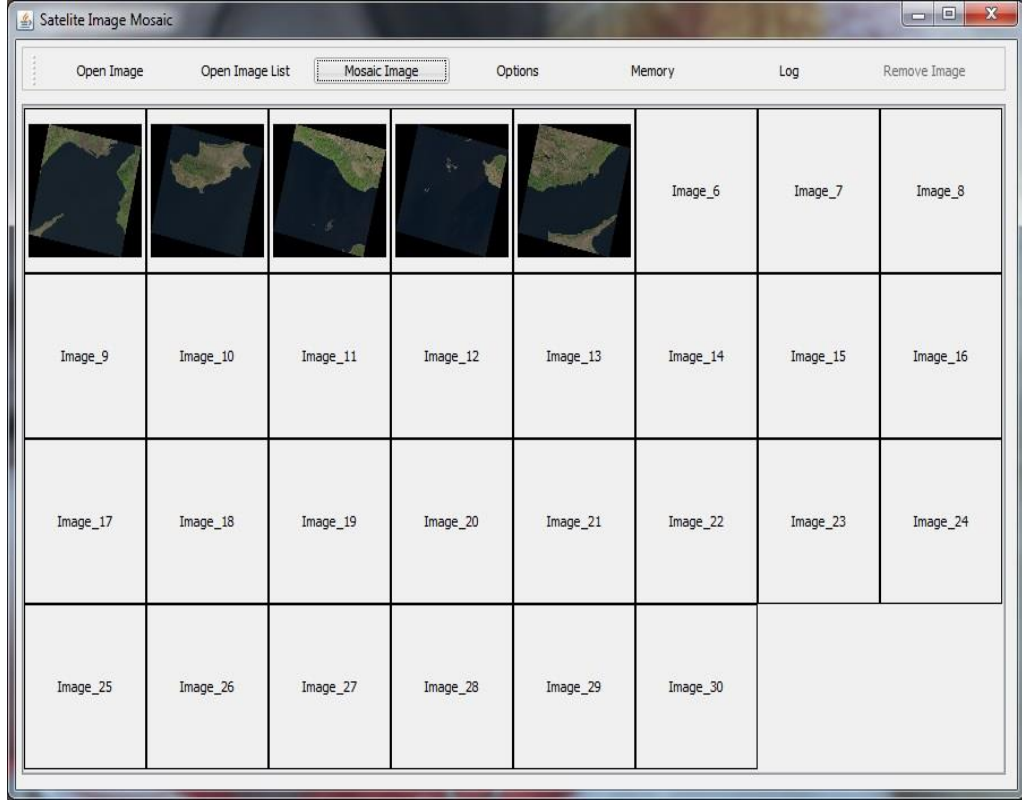


Şekil 5.2. Mozaiklerin sisteme yüklenmesini sağlayan arayüz

Bir resime ait wld dosyası, raster harita görüntülerini coğrafik olarak referanslamak için Coğrafi Bilgi Sistemleri (CBS) tarafından kullanılan düz bir metindir. Örnek bir durum çalışması olarak Kıbrıs Ada'sına ait beş mozaiği kullanıcı seçtikten sonra her bir mozaik arayüzde kullanıcıya sunulmaktadır. Böylelikle kullanıcı seçtiği mozaikleri görsel olarak yönetebilmektedir (Şekil 5.4 'e bakınız).

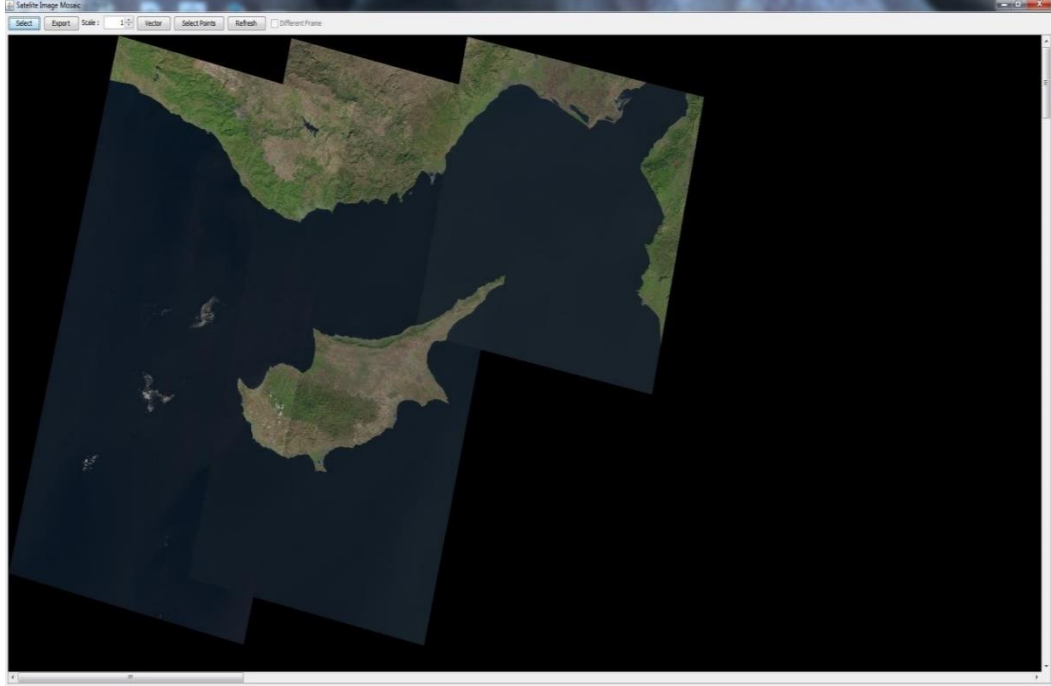


Şekil 5.3. Mozaiklerin seçilmesini sağlayan arayüz



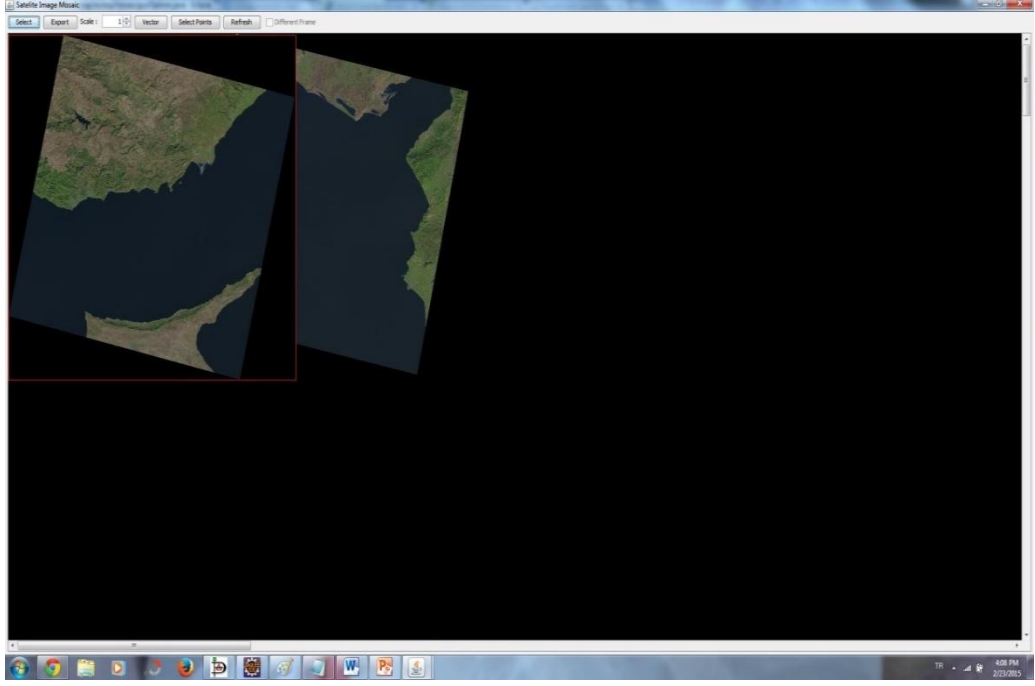
Şekil 5.4. Kullanıcının seçtiği mozaiklerin gösterildiği arayüz

“Open Image List/Görüntü Listesini Aç” menüsü ile kullanıcının mozaikleri tek tek seçmesinden ziyade bir klasör içindeki tüm mozaiklerin seçtirilmesi sağlanmıştır. Tüm mozaikler kullanıcı tarafından seçildikten sonra “Mosaic Image/Mozaikleri Birleştir” butonu yardımıyla Bölüm 4.1’de anlatılan birleştirme tekniğine göre mozaiklerin birleştirilmesi sağlanır. Şekil 5.4 ’te seçilen mozaiklerin birleştirilmiş hali Şekil 5.5 ’te sunulmuştur.



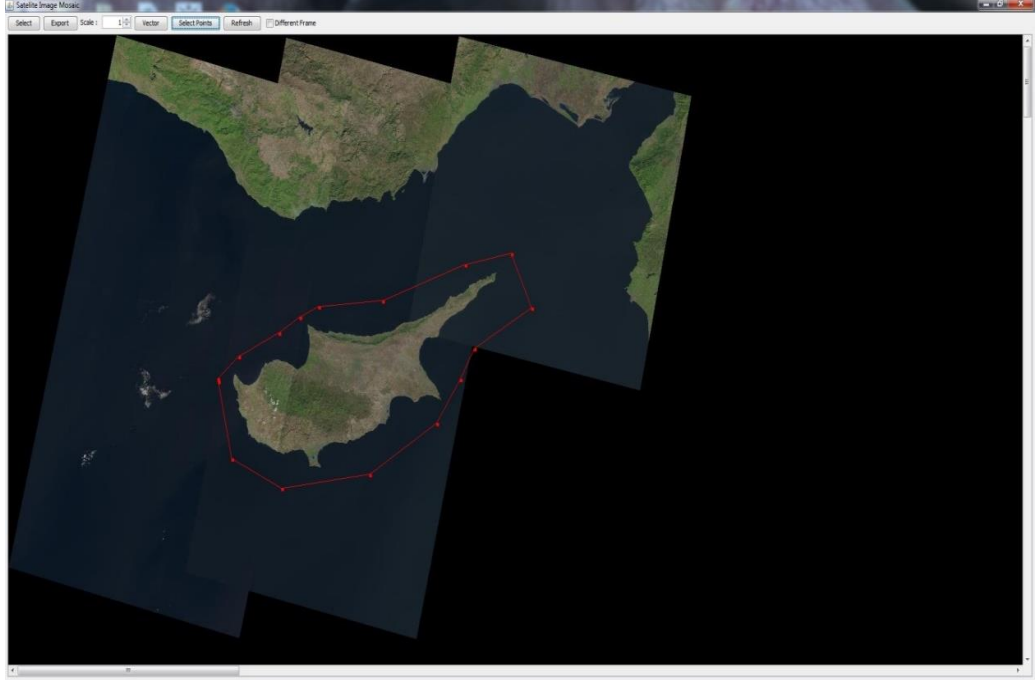
Şekil 5.5. Mozaiklerin birleştirildikten sonra gösterildiği arayüz

Birleşmiş uydu görüntüsü ayrı bir pencerede Şekil 5.5'teki gibi gösterilmektedir. Bu pencerede de birtakım menüler mevcuttur. “Select/Seç” menüsü yardımıyla kullanıcı birleşmiş görüntüdeki herhangi bir yeri seçerek ilgili mozaikin ön plana getirilmesini sağlamaktadır. Şekil 5.6'da da görüldüğü gibi birleşmeye katılan iki mozaikten birinin ön plana getirilmesi sağlanmıştır. Birleştirilen görüntü farklı bir görüntü olarak “Export/Çıkart” menüsü ile kaydedilebilmektedir. Ayrıca kullanıcı “Scale/Ölçekle” değerini 0-1 arasında değiştirerek zoom-in zoom-out yapılması sağlanmıştır.



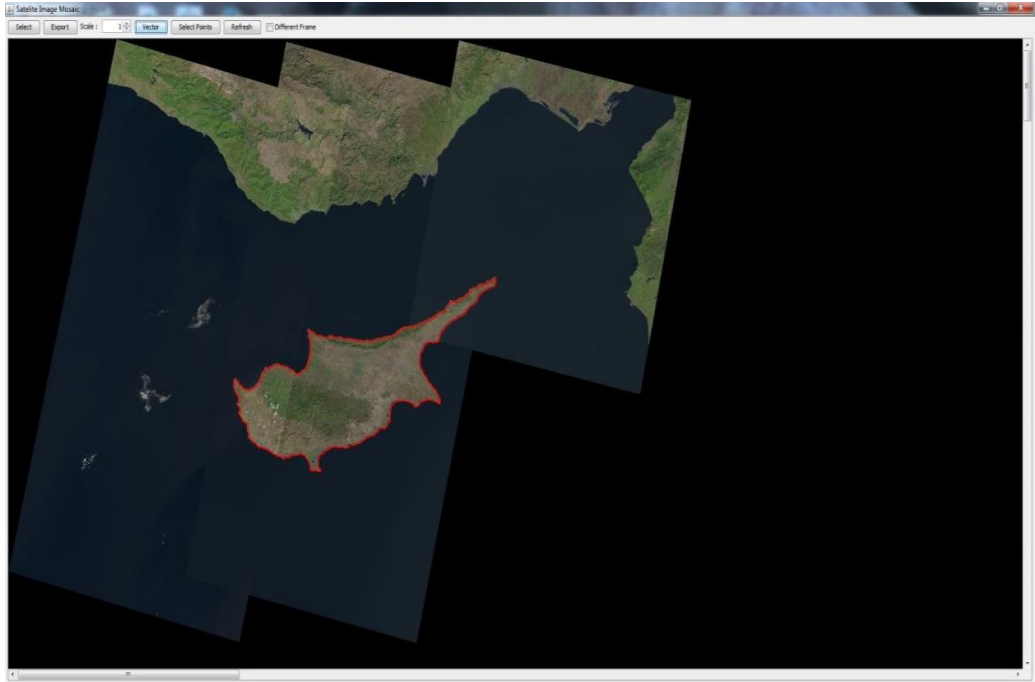
Şekil 5.6. İlgili mozağin ön tarafa getirilmesi

Birleştirilen görüntü üzerinde kullanıcı vektörleştirmek istediği nesneyi fare yardımıyla seçerek belirtmektedir. Bu işlem “Select Points/Noktaları Seç” altında tanımlanmıştır. Örneğin Şekil 5.5’teki görüntüden Kıbrıs Ada’sını çıkarmak istediğinde Şekil 5.7’teki gibi noktaları seçmelidir (kırmızı nokta ve bu noktaların birleşmesinden oluşan bölge). Bu noktalar çıkarılmak istenen nesnenin harita üzerindeki yeri hakkında bilgi vermektedir.



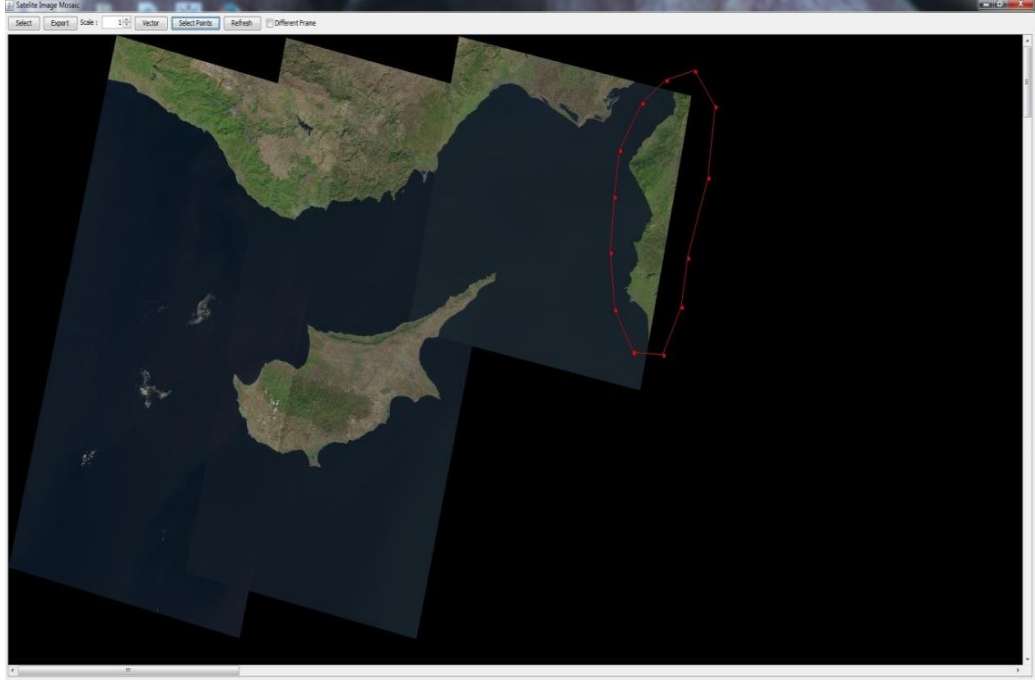
Şekil 5.7. Çıkarılacak nesnenin belirlendiği arayüz

Birleştirilen ve vektörleştirilmek istenen görüntü “Vector/Vektörleştir” menüsü yardımıyla Bölüm 4.2’de anlatılan yöntemlerle vektörleştirilmektedir. Şekil 5.7’de seçilen görüntünün vektör hali kırmızı çizgilerle Şekil 5.8’de verilmiştir.

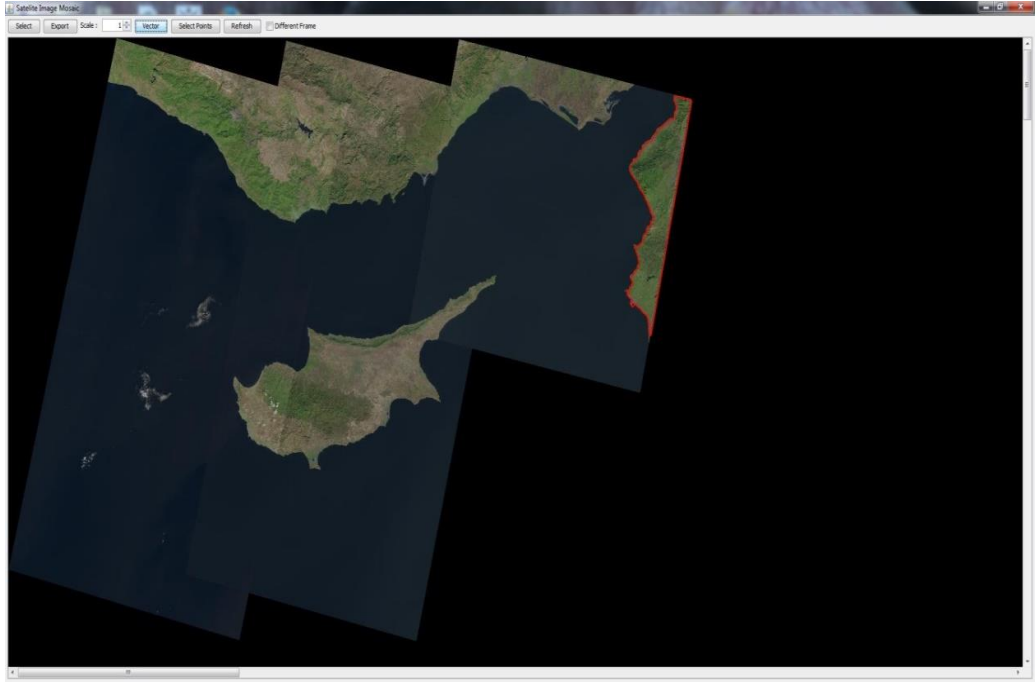


Şekil 5.8. Çıkarılacak nesnenin vektörleştirildiği arayüz

Şekil 5.9’da seçilen farklı bir nesnenin vektör hali, Şekil 5.10’da verilmiştir.

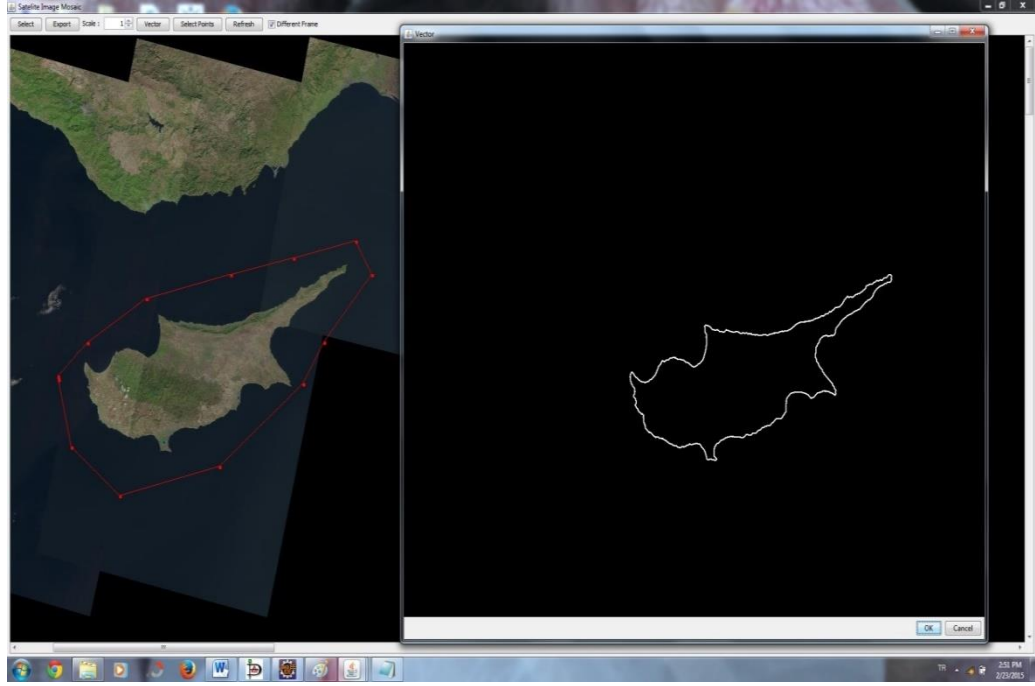


Şekil 5.9. Çıkarılacak diğer nesnenin belirlendiği arayüz



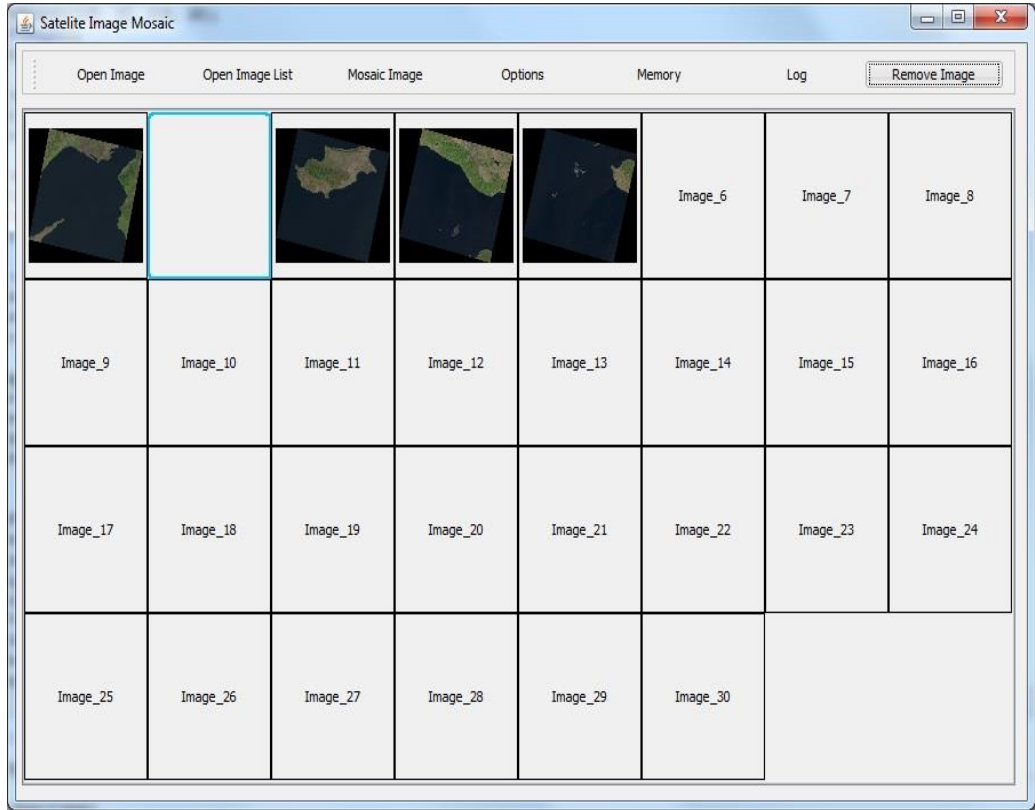
Şekil 5.10. Çıkarılacak diğer nesnenin vektörleştirildiği arayüz

Kullanıcı isterse “Different Frame/Farklı Çerçeve” seçeneğini aktif ederek vektörleşmiş resmi farklı bir çerçevede görebilir (Şekil 5.11’e bakınız).



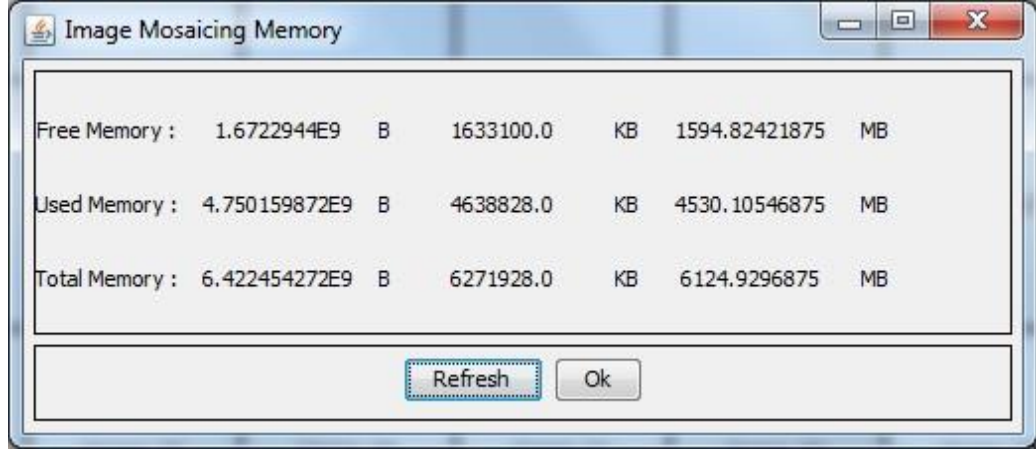
Şekil 5.11. Vektörleştirilen nesnenin farklı pencerede gösterilişi

“Remove Image/Görüntüyü Silme” seçeneği ile kullanıcı seçmiş olduğu mozaiklerden istediğini silebilir (Şekil 5.12’ye bakınız).

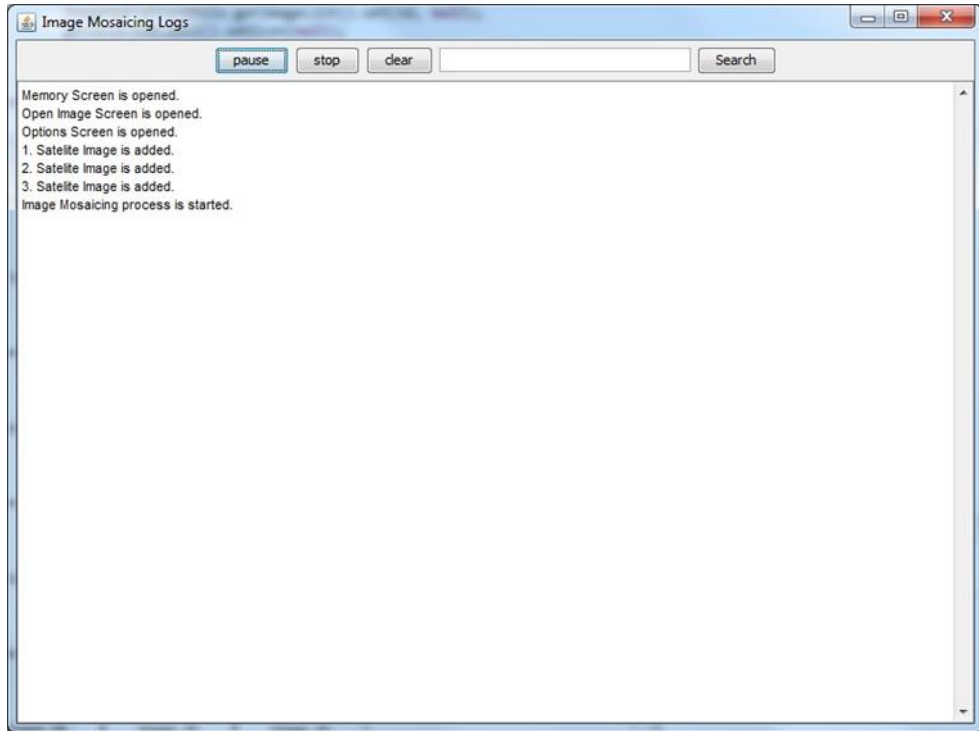


Şekil 5.12. Yüklenmiş mozaiklerin silindiği arayüz

Ayrıca kullanıcı mozaikleri birleştirirken kullanılan bellek miktarı, kalan bellek miktarı gibi bilgileri görüntüleyebilir (Şekil 5.13'e bakınız). Uygulamayı kullanırken tutulan log'lara da erişilmektedir (Şekil 5.14'e bakınız).

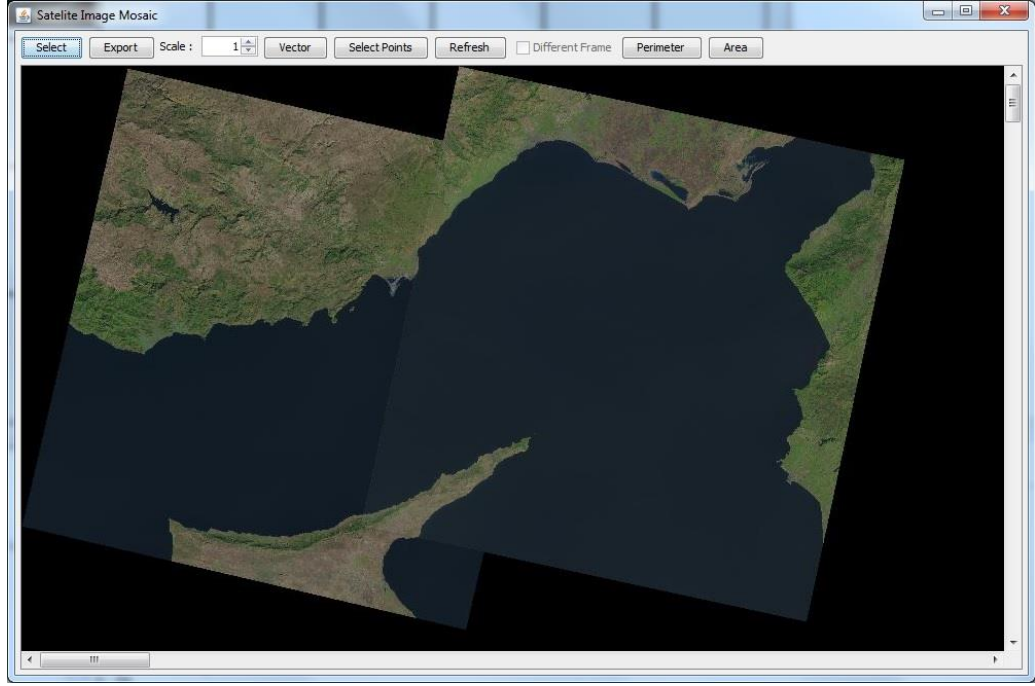


Şekil 5.13. Bellek kullanım bilgisinin gösterildiği arayüz



Şekil 5.14. Logların gösterildiği arayüz

Alan ve çevre gibi postgis veritabanı özel fonksiyon sorgularına da "Area/Alan" ve "Perimeter/Çevre" butonları aracılığı ile erişilmektedir (Şekil 5.15'e bakınız).



Şekil 5.15. Postgis veritabanının ait alan ve çevre sorgusunun yapıldığı arayüz

SONUÇLAR VE ÖNERİLER

Uydu görüntüleri, çözünürlüklerinin yüksek olan görüntülerdir. Uydu görüntüleri üzerinde yapılan çalışmalarda, bilgisayarın hafızasında çok fazla yer kaplamaktadır. Üzerinde çalışılan uydu görüntüsü sayısı arttıkça bellek gereksinimi artacak ve bir noktadan sonra bellek, ihtiyacı karşılamayacaktır ve mevcut bellek boyutunu artırılmadan yapılan çalışma devam edilemez duruma gelmektedir. Bu çalışmada geliştirilen algoritma ile, bilgisayarın mevcut bellek boyutunu artırmadan, üzerinde işlem yapılacak uydu görüntü sayısı artırılabilir. Böylelikle, çok fazla parça uydu görüntülerinin bir araya gelerek oluşturduğu büyük adaların, denize kıyısı olan büyük kara parçalarının üzerinde de zaman-mekansal veya topolojik sorgular yapılmasına imkan sağlanmaktadır. Geliştirilen algoritma ile, bellek optimize bir şekilde kullanılmaktadır.

Bu tez çalışmasıyla geliştirilen uygulamada performansın artırılması, belleğin etkin kullanılması için önbellek(caching) mekanizması olan ehcache mimarisi kullanılmaktadır. Bu mimariyle, belleğe yüklenen her bir uydu görüntüsü bir sonraki uydu görüntüsü yüklenmeden önce serilize (serialization) edilerek diskte tutulmaktadır. Böylece uygulamaya ardarda uydu görüntüsü yüklendikçe, belleğin dolması ve yeni taleplere cevap verememesi engellenmiş olmaktadır. Kullanılan ehcache'in disk önbellekleme mimarisinde, veriler diskte serilize bir şekilde tutulmaktadır ve diskin bellekten daha yavaş olduğu dikkate alındığında, işlem zamanında yavaşlama olacaktır. Birçok uydu görüntüsüyle birlikte çalışmak ve bellek artırımına gidilmek istenmediğinden dolayı bu yavaşlama gözardı edilmektedir. Bu yavaşlamayı gidermek için Terracotta firmasının ücretli ürünü olan "BigMemory" kullanılarak, mevcut bellek miktarı yazılımsal olarak artırılabilir. Böylece önbelleğe alma işlemi diskin üzerinde değil de belleğin üzerine yapılabilmektedir.

Uydu görüntüleri farklı zamanlarda, farklı iklim koşullarında elde edilen görüntülerdir. Aynı yerin uydu görüntüsünün biri çok bulutlu bir havada, bir diğeri güneşli, aydınlık bir havada olabilir. Bu gibi durumlar, benzer özelliklerin bulunarak, görüntülerin

birleştirildiği uygulamalar için dezavantaj oluşturmaktadır. Bu çalışmada ise, uydu görüntülerinin elde edildiği USGS çevrimiçi sitesinin sağladığı “wld” dosyası kullanılarak, görüntülerin mozaikleştirilmesi sırasında değişen iklim koşullarının etkisi ortadan kaldırılmaktadır. Ayrıca farklı zamanlarda, farklı iklim koşullarındaki uydu görüntülerinin mozaikleştirilmesi sırasında, üstü üste gelen kısımlarda renk farklılıkları olmaktadır. Bu renk farklılıklarının azaltılması için, “lighten” metodu kullanılmıştır. Mevcut bilgisayarın boş bellek saatini dikkate alarak yapabiliriz.

Uydu görüntüleri üzerindeki zaman-mekansal veya topolojik sorguların performansını artırmak ve bilgisayara olan yükünü azaltmak için uydu görüntülerinin vektörleştirilmesi gerekmektedir. Uydu görüntülerinin birleştirilmesiyle üzerinde işlem yapılmak istenen, ada görüntü veya denize kıyısı olan bir kara parçası oluşturulmuştur. Bu kara parçaları üzerinde yapılacak sorgular için vektörleştirilmesi için poligon şeklinde modellenmesi gerçekleştirilmiştir. Aynı zamanda bu çalışma ile, birleştirilmiş büyük bir görüntü içerisinde birden fazla ada uydu görüntüsü veya denize kıyısı olan kara parçası varsa, bunlardan herhangi biri seçilerek sadece seçilen olan kısım poligon şeklinde vektörleştirilebilmektedir. Bu sayede, uydu görüntüsünde bulunan ancak üzerinde çalışma yapılmak istenmeyen kısımlar gözardı edilerek, hem zaman hem performans kazancı sağlanmaktadır.

Vektörleştirme esnasında poligon olarak modellenen kısımlar, coğrafik veritabanlarından olan postGIS veritabanına kaydedilmektedir. Böylelikle üzerinde sorgu yapmak istediğimiz ancak belleğe bile sığdıramadığımız uydu görüntüleri, nokta setlerinden oluşan bir poligon olarak veritabanında tutulmaktadır. Bu veriler üzerinde yapılan sorgular postGIS veritabanında özel olarak tasarlanmış sorgularla hızlı bir şekilde yapılmaktadır. Bu çalışma ile;

- Mevcut bellek miktarlarıyla, yüksek çözünürlüğe sahip uydu görüntülerinin birleştirilmesi sağlanmıştır. Bu sayede bellek artırımına gitmenin, maliyeti artırmanın önüne geçilmiştir.
- Görüntü birleştirme işleminde kullanılan diğer bazı algoritmalar (SIFT, SURF, RANSAC) araştırılmış ve uydu görüntüleri üzerindeki eksiklikleri belirtilmiştir.
- Uydu görüntüleri vektörleştirilerek, bilgisayarda kapladığı yer miktarı azaltılmıştır. Ayrıca dağıtık sistemlerle, web servisleriyle uydu görüntülerinin işlenmesi de kolaylaştırılmıştır.

- Birleştirilmiş büyük uydu görüntüleri üzerinde bulunan nesnelerin (adalar, denize kıyısı olan kara parçaları) içinden çalışma yapılacak olan nesnenin seçilebilmesi sağlanmıştır.
- Sadece ada uydu görüntüsünün değil, denize kıyısı olan diğer karaparçalarının da vektörleştirilmiştir.
- Deprem, toprak kayması, dolgu gibi çeşitli olayların yeryüzüne etkisini ölçebilmek için temel oluşturan, vektörleştirilen nesnelerin alanları, postgis veritabanında sorguyla hesaplanmıştır.
- Geliştirilen uygulama ve uygulamada kullanılan kütüphaneler açık kaynak kodlu olması sebebiyle, herkes tarafından geliştirmeye, iyileştirmeye açıktır. Açık kaynak kodlu olması sebebiyle, uygulamanın geliştirme maliyeti de azaltılmıştır.

Bu çalışma, zaman-mekansal ve topolojik analizlere temel oluşturmaktadır. Bundan sonraki çalışmada elde edilen, coğrafik veritabanına kaydedilen poligon bilgileri üzerinde bu analizlerin yapılması ve sonuçların karşılaştırılması hedeflenmektedir.

KAYNAKLAR

- [1] Schowengerdt R. A., *Remote Sensing: Models and Methods for Image Processing*, 3rd ed., Academic Press, California, 2007.
- [2] Lillesand T., Kiefer R. W., Chipman J., *Remote Sensing and Image Interpretation*, 7th ed., John Wiley & Sons, New Jersey, 2015.
- [3] Campbell J. B., Wynne R. H., *Introduction to Remote Sensing*, 5th ed., The Guilford Press, New York, 2011.
- [4] Baltasvias E., Zhang C., Automated updating of road databases from aerial images, *International Journal of Applied Earth Observation and Geoinformation*, 2005, **6**(3), 199-213.
- [5] Jodouin S., Bentabet L., Ziou D., Vaillancourt J., Armenakis C., Spatial database updating using active contours for multispectral: application with Landsat 7, *ISPRS Journal of Photogrammetry & Remote Sensing*, 2003, **57**(5), 346-355.
- [6] Zhang Q., Seto K. C., Mapping urbanization dynamics at regional and global scales using multi-temporal DMSP/OLS nighttime light data, *Remote Sensing of Environment*, 2011, **115**(9), 2320-2329.
- [7] Taubenböck H., Esch T., Felbier A., Wiesner M., Roth A., Dech S., Monitoring urbanization in mega cities from space, *Remote Sensing of Environment*, 2012, **117**, 162-176.
- [8] Salovaara J. K., Thessler S., Malik N. R., Tuomisto H., Classification of Amazonian primary rain forest vegetation using Landsat ETM+ satellite imagery, *Remote Sensing of Environment*, 2005, **97**(1), 39-51.
- [9] Cicek H., Sunohara M., Wilkes G., McNairn H., Pick F., Topp E., Lapen R. D., Using vegetation indices from satellite remote sensing to assess corn and soybean response to controlled tile drainage, *Agricultural Water Management*, 2010, **98**(2), 261-270.
- [10] Yi Z., Zhiguo C., Yang X. Multi-spectral remote image registration based on SIFT, *Electronics Letters*, 2008, **44**(2), 107-108.
- [11] Song Z. L., Zhang J., Remote sensing image registration based on retrofitted SURF algorithm and trajectories generated from Lissajous figures, *Geoscience and Remote Sensing Letters*, 2010, **7**(3), 491-495.
- [12] Lee S. R., A coarse-to-fine approach for remote-sensing image registration based on a local method, *International Journal on Smart Sensing and Intelligent Systems*, 2010, **3**(4), 690-702.

- [13] Rube I. E., Sharkas M., Salman A., Salem A., Automatic Selection of Control Points for Remote Sensing Image Registration Based on Multi-Scale SIFT, *International Conference on Signal, Image Processing and Applications*, Chennai, India, 17-18 December 2011.
- [14] Lowe D. G., Distinctive image features from scale-invariant keypoints, *International journal of computer vision*, 2004, **60**(2), 91-110.
- [15] Bay H., Tuytelaars T., Van Gool L., SURF: Speeded Up Robust Features, Editors: Leonardis A., Bischof H., Pinz A., *Computer Vision–ECCV 2006*, 1st ed., Springer-Verlag Berlin Heidelberg, Berlin, 404-417, 2006.
- [16] Sayar A., Eken S., Mert U., Registering Landsat-8 Mosaic Images: A Case Study on the Marmara Sea, *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, Ankara, Türkiye, 7-9 November 2013.
- [17] Sayar A., Eken S., Mert U., Tiling of Satellite Images to Capture an Island Object, Editors: Mladenov V., Jayne C., Iliadis L., *Engineering Applications of Neural Networks*, 1st ed., Springer International Publishing, Switzerland, 195-204, 2014.
- [18] Luck G., The Role of Caching in Large Scale Architecture, DZone, https://static.dzone.com/dz1/dz-files/layered_architecture.png (Ziyaret tarihi: 12 Şubat 2014).
- [19] Eken S., Sayar A., An automated technique to determine spatiotemporal changes in satellite island images with vectorization and spatial queries, *Sadhana*, 2015, **40**(1), 121-137.
- [20] Otsu N., A threshold selection method from gray-level histograms, *IEEE Transactions on Systems, Man, and Cybernetics*, 1979, **9**(1), 62–66.
- [21] Ke Y., Sukthankar R., Huston L., Efficient Near-duplicate Detection and Sub-image Retrieval, *Proceedings of the 12th annual ACM international conference on Multimedia*, New York, 10-16 October 2004.
- [22] <http://www.turksatglobe.com/Views/Services/ImageProcessing.aspx?ContentID=1> (Ziyaret tarihi: 3 Mart 2015).
- [23] Burt P. J., Adelson E. H., A multiresolution spline with application to image mosaics, *ACM Transactions on Graphics (TOG)*, 1983, **2**(4), 217-236.
- [24] Rankov V., Locke R. J., Edens R. J., Barber P. R., Vojnovic B., An algorithm for image stitching and blending, *Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XII*, California, 22 January 2005.
- [25] Ma, Y., The mathematic magic of Photoshop blend modes for image processing, *2011 International Conference on Multimedia Technology (ICMT)*, Hangzhou, 26-28 July 2011.
- [26] Panchal P. M., Panchal S. R., Shah S. K., A comparison of SIFT and SURF, *International Journal of Innovative Research in Computer and*

Communication Engineering, 2013, **1**(2), 323-327.

- [27] Karakuş P., Karabörk H., Surf Algoritması Kullanılarak Uzaktan Algılama Görüntülerinin Geometrik Kaydı, *5. Uzaktan Algılama-CBS Sempozyumu (UZAL-CBS 2014)*, İstanbul, Türkiye, 14-17 Ekim 2014.
- [28] Cassettari S., *Introduction To Integrated Geo-Information Management*, 1st ed., Springer Science & Business Media, London, 2012.
- [29] Couclelis H., People manipulate objects (but cultivate fields): beyond the raster-vector debate in GIS, *International Conference GIS-Theories and methods of spatio-temporal reasoning in geographic space*, Pisa, Italy, 21-23 September 1992.
- [30] Alkış Z., Coğrafi Bilgi Sistemi Bileşenleri, TMMOB Harita ve Kadastro Mühendisleri Odası, http://www.hkmo.org.tr/resimler/ekler/KOTM_588e674d3f0faf9_ek.pdf (Ziyaret tarihi: 4 Nisan 2014).
- [31] Guerrero Pena M., A comparative study of three image matching algorithms: SIFT, SURF, and FAST, Master's Thesis, Utah State University, Computer Science, Utah, 2011.
- [32] Juan L., Gwun O., A comparison of sift, pca-sift and surf, *International Journal of Image Processing (IJIP)*, 2009, **3**(4), 143-152.
- [33] Rublee E., Rabaud V., Konolige K., Bradski G., ORB: an efficient alternative to SIFT or SURF, *2011 IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 6-13 November 2011.
- [34] Shum H. Y., Szeliski R., Construction of panoramic image mosaics with global and local alignment, Editors: Benosman R., Kang S. B., *Panoramic Vision: sensors, theory, and applications*, 1st ed., Springer-Verlag New York, New Jersey, 227-268, 2001.
- [35] Turner D., Lucieer A., Watson, C., An automated technique for generating georectified mosaics from ultra-high resolution unmanned aerial vehicle (UAV) imagery, based on structure from motion (SfM) point clouds, *Remote Sensing*, 2012, **4**(5), 1392-1410.
- [36] Barrett W. A., Mortensen E. N., Fast, accurate, and reproducible live-wire boundary extraction, *4th International Conference on Visualization in Biomedical Computing*, Hamburg, Germany, 22-25 September 1996.
- [37] Ozkaya M., Road extraction from high resolution satellite images, Yüksek Lisans Tezi, Orta Doğu Teknik üniversitesi, Enformatik Enstitüsü, Ankara, 2009, 255626.
- [38] Wei Y., Zhao Z., Song J., Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection, *2004 IEEE International Geoscience and Remote Sensing Symposium Proceedings*, Alaska, USA, 20-24 September 2004.

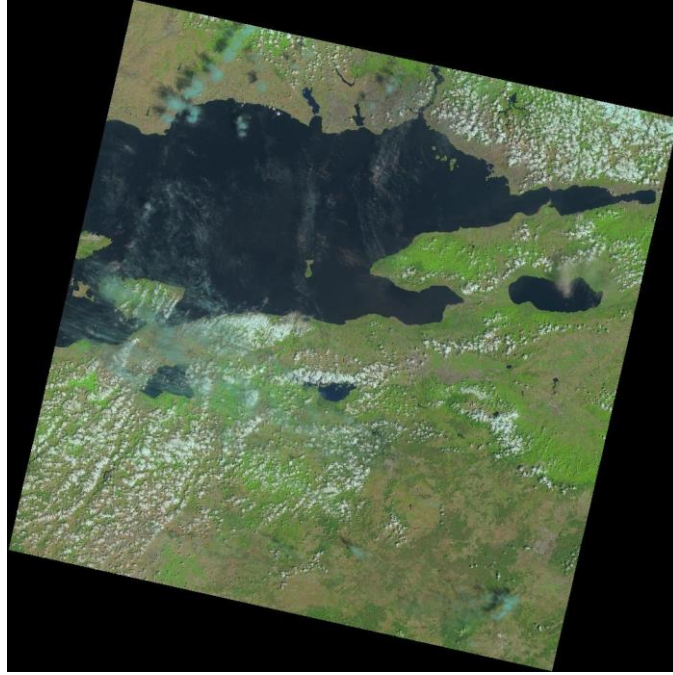
- [39] Congalton R. G., Exploring and evaluating the consequences of vector-to-raster and raster-to-vector conversion, *Photogrammetric Engineering and Remote Sensing*, 1997, **63**(4), 425-434.
- [40] Burrough P. A., Principles of geographical information systems for land resources assessment, *Geocarto International*, 1986, **1**(3), 54.
- [41] Peuquet D. J., An examination of techniques for reformatting digital cartographic data/Part 1: the raster-to-vector process, *Cartographica: The International Journal for Geographic Information and Geovisualization*, 1981, **18**(1), 34-48.
- [42] Güting R. H., An introduction to spatial database systems, *The VLDB Journal-The International Journal on Very Large Data Bases*, 1994, **3**(4), 357-399.
- [43] Paredaens J., Van den Bussche J., Van Gucht D., Towards a theory of spatial database queries, *PODS '94 Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, Minneapolis, USA, 24-27 May 1994.
- [44] Brinkhoff T., Kriegel H. P., Schneider R., Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems, *Proceedings of IEEE 9th International Conference on Data Engineering*, Vienna, Austria, 19-23 April 1993.
- [45] Brinkhoff T., Horn H., Kriegel H. P., Schneider R., A storage and access architecture for efficient query processing in spatial database systems, *The Third International Symposium on Large Spatial Databases (SSD '93)*, Singapore, 23-25 June 1993.
- [46] Yeung A. K., Hall G. B., *Spatial database systems: Design, implementation and project management*, 1st ed., Springer Science & Business Media, Dordrecht, 2007.
- [47] Van Oosterom P. J. M., Lemmen C. H. J., Spatial data management on a very large cadastral database. *Computers, Environment and Urban Systems*, 2001, **25**(4), 509-528.
- [48] Jianzhuang L., Wenqing L., Yupeng T., (1991, June). Automatic thresholding of gray-level pictures using two-dimension Otsu method. *China 1991 International Conference on Circuits and Systems*, Shenzhen, China, 16-17 June 1991.
- [49] Wind D., *Instant Effective Caching with Ehcache*, 1st ed., Packt Publishing, Birmingham, 2013.
- [50] Bräger M., Martini R., Brightwell M., Suwalska A., Koufakis E., High-Availability Monitoring and Big Data: Using Java Clustering and Caching Technologies to Meet Complex Monitoring Scenarios, *ICALPCS13 International Conference on Accelerator & Large Experimental Physics Control Systems*, San Francisco, USA, 6-11 October 2013.

- [51] Cosma D. C., Marinescu R., Distributable features view: Visualizing the structural characteristics of distributed software systems, *4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, Banff, Ontario, 24-25 June 2007.
- [52] <http://portal.netcad.com.tr/pages/viewpage.action?pageId=111477857> (Ziyaret tarihi: 10 Ağustos 2015).
- [53] <http://www.seos-project.eu/modules/classification/classification-c00-p05.html> (Ziyaret tarihi: 10 Ağustos 2015).
- [54] <http://www.hvkk.tsk.tr/tr/IcerikDetay.aspx?ID=239> (Ziyaret tarihi: 11 Eylül 2015).
- [55] <http://odnature.naturalsciences.be/geocolour/results> (Ziyaret tarihi: 12 Eylül 2015).

EKLER

EK-A

Bu ekte, USGS web sitesinden alınmış örnek bir uydu görüntüsü ve meta verileri sunulmuştur. Şekil A.1’de örnek bir uydu görüntüsü sunulmuştur.



Şekil A.1. Örnek bir uydu görüntüsü

Tablo A.1’de örnek bir uydu görüntüsüne ait meta veriler sunulmuştur.

Tablo A.1. Örnek bir uydu görüntüsü meta verileri

Veri Kümesi	Değer
Landsat Scene Identifier	LC81800322015217LGN00
WRS Path	180
WRS Row	032
Target WRS Path	180
Target WRS Row	032
Full or Partial Scene	FULL
Nadir/Off Nadir	NADIR
Data Category	NOMINAL
Bias Parameter File Name OLI	LO8BPF20150805082118_20150805091240.02
Bias Parameter File Name TIRS	LT8BPF20150805081724_20150805091441.01
Calibration Parameter File	L8CPF20150701_20150930.01
RLUT File Name	L8RLUT20150303_20431231v11.h5
Roll Angle	-.001
Station Identifier	LGN
Day/Night	DAY

Data Type Level 1	L1T
Sensor Identifier	OLI_TIRS
Date Acquired	2015/08/05
Start Time	2015:217:08:45:09.1367380
Stop Time	2015:217:08:45:40.9067340
Image Quality	9
Scene Cloud Cover	19,63
Sun Elevation	60,18752533
Sun Azimuth	135,21757112
Geometric RMSE Model X	5,125
Geometric RMSE Model Y	4,703
Ground Control Points Model	376
Ground Control Points Version	2
Browse Exists	Y
Processing Software Version	LPGS_2.5.1
Center Latitude	40°19'58.80"N
Center Longitude	28°40'09.01"E
NW Corner Lat	41°23'23.39"N
NW Corner Long	27°50'08.27"E
NE Corner Lat	40°58'48.25"N
NE Corner Long	30°02'53.27"E
SE Corner Lat	39°15'39.85"N
SE Corner Long	29°28'18.88"E
SW Corner Lat	39°40'07.03"N
SW Corner Long	27°18'52.63"E
Center Latitude dec	40,333
Center Longitude dec	28,66917
NW Corner Lat dec	41,38983
NW Corner Long dec	27,83563
NE Corner Lat dec	40,98007
NE Corner Long dec	30,04813
SE Corner Lat dec	39,26107
SE Corner Long dec	29,47191
SW Corner Lat dec	39,66862
SW Corner Long dec	27,31462

EK-B

Uydu Görüntülerinin Birleştirilmesi ve Vektörel Modellenmesi İçin Geliştirilen Sınıflara Ait Program Kodları

Bu ekte, uydu görüntülerinin birleştirilmesi ve vektörel modellenmesiyle ilgili Java ve Matlab dilinde yazılmış sınıfların kodları sunulmuştur. Bellek optimizasyonun algoritmasının gerçekleştirildiği sınıf:

```
package org.mosaic.utils;

public class MemoryOptimization {

    private final int byteAmount = 106496;
    private final int avgImageDimension = 8000;
    private final int bytePerPixel = 4;
    private int imageNumber = 30;

    private double totalMemory;
    private double jvmMemory;
    private double totalImagesMemory;
    private static MemoryOptimization instance;

    public static MemoryOptimization getInstance(){
        if(instance == null) {
            instance = new MemoryOptimization();
        }
        return instance;
    }
    public void initialize(){
        this.totalMemory = getTotalMemory();
        this.jvmMemory = getJvmMemory();
        this.totalImagesMemory = getTotalImagesMemory();
    }

    private double getTotalMemory() {
        com.sun.management.OperatingSystemMXBean bean =
        (com.sun.management.OperatingSystemMXBean)
        java.lang.management.ManagementFactory.getOperatingSystemMXBean();
        totalMemory = bean.getTotalPhysicalMemorySize();
        return totalMemory / byteAmount ;
    }
    private double getJvmMemory() {
        jvmMemory = getTotalMemory() / 4;
        System.out.println("Sum Jvm Memory : " + jvmMemory);
        return 5041;}
}
```



```

private double getTotalImagesMemory(){

        double imageMemoryInByte = avgImageDimension *
avgImageDimension * bytePerPixel;
        double totalImageMemoryInByte = imageNumber *
imageMemoryInByte;
        totalImagesMemory = totalImageMemoryInByte / byteAmount;
        System.out.println("Sum Total Images Memory : " +
totalImagesMemory);
        return totalImagesMemory;
    }

    public double getScaleRate(){
        initalize();
        double scaleRate =(jvmMemory / (totalImagesMemory));
        if(scaleRate >= 1) scaleRate = 1;
        //scaleRate = (scaleRate - 100) / 100;
        System.out.println(scaleRate);
        return scaleRate;
    }

    public int getImageNumber() {
        return imageNumber;
    }

    public void setImageNumber(int imageNumber) {
        this.imageNumber = imageNumber;
    }

}

```

Uydu görüntülerinin birleştirilme ve vektörleştirme algoritmasının gerçekleştirildiği sınıf:

```

package org.mosaic.blender;

import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Composite;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Toolkit;

```

```

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import javax.swing.JDialog;
import javax.swing.JLayeredPane;
import javax.swing.JPanel;

import net.sf.ehcache.Cache;
import net.sf.ehcache.Element;

import org.mosaic.gui.VectorFrame;
import org.mosaic.gui.controller.CompositePanelController;
import org.mosaic.image.GeoImage;
import org.mosaic.image.writer.ImageWriters;
import org.mosaic.utils.CacheConfigurator;
import org.mosaic.utils.CoordinateOperation;
import org.mosaic.utils.GeoOperationUtils;
import org.mosaic.utils.ImageOperationUtils;

import BwBoundary.BwBoundary;

import com.mathworks.toolbox.javabuilder.MWCellArray;

public class CompositePanel extends JPanel{

    private static final long serialVersionUID = 1L;
    private BufferedImage image = null;
    private BufferedImage sourceImage = null ;
    private Composite composite = BlendComposite.Lighten;
    private int compositeImageWidth;
    private int compositeImageHeight;
    private boolean repaint = false;
    private GeoOperationUtils geoOp;
    private CoordinateOperation coord;
    private double scale = 1.0;
    private Cache cache;
    private boolean isSelectedImage = false;
    private int clickedX;
    private int clickedY;
    private int lastClickedX = -1;
    private int lastClickedY = -1;
    private int screenWidth;

```

```

private int screenHeight;
private Graphics2D g2;
private GeoImage selectedImage;
private Composite defaultComposite;
private CacheConfigurator cacheConfig;
private boolean defaultPaint = true;
private boolean isVector = false;
private boolean isRefresh = false;
private String vectorImagePath;
private List<Integer> restrictedImageIds;
private List<List<Point>> vectorPoints;
private List<Point> clickedPoints;
private CompositePanelController cpController;
private List<Point> tempA;
private List<Point> tempB;
private JLayeredPane layeredPane;
private Graphics2D generalGraphics;
private JPanel panelBlue = new JPanel();

//For Crop Start Point
private Point leftPoint;
private Point topPoint;

public CompositePanel(final CompositePanelController cpController) {
    this.cpController = cpController;
    restrictedImageIds = new ArrayList<Integer>();
    vectorPoints = new ArrayList<List<Point>>();
    clickedPoints = new ArrayList<Point>();
    clickedPoints.clear();
    setOpaque(true);
    addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            clickedX = e.getX();
            clickedY = e.getY();
            if(cpController.isSelect()){
                for(GeoImage geoImage :
ImageOperationUtils.getImageList()){
                    Rectangle rect = geoImage.getRect();
                    if(rect.contains(clickedX, clickedY)){
                        System.out.println("Selected Image : "
+ geoImage.getImageID());

                        selectedImage = geoImage;
                        paintImage();
                        break;
                    }
                }
            }
            revalidate();
            paintImage();
        }
    });
}

```

```

        }else if(cpController.isSelectPoint()){
            clickedPoints.add(new Point(clickedX, clickedY));
            System.out.println(clickedX + ","+clickedY);
            paintImage();
        }
    }
});

Dimension screenDimension =
Toolkit.getDefaultToolkit().getScreenSize();
screenHeight = screenDimension.height;
screenWidth = screenDimension.width;
cacheConfig = CacheConfigurator.newInstance();
setOpaque(true);

geoOp = new GeoOperationUtils();
geoOp.calculateBounds();
coord = new
CoordinateOperation(geoOp.getLeft(),geoOp.getRight(), geoOp.getTop(),
geoOp.getDown());
    compositeImageWidth =
coord.getWidthBasedOnPixel(geoOp.getLeft(), geoOp.getRight());
    compositeImageHeight =
coord.getHeightBasedOnPixel(geoOp.getTop(), geoOp.getDown());
}

private void selectImage(Graphics2D g,int x,int y){
    for(GeoImage geoImage : ImageOperationUtils.getImageList()){
        Rectangle rect = geoImage.getRect();
        int xx = (int)(geoImage.getX()-geoOp.getLeft());
        int yy = (int)(geoOp.getTop() - geoImage.getY());
        Element element = cache.get(geoImage.getImageID());
        BufferedImage image1 =
ImageOperationUtils.byteArraytoImage((byte[])element.getObjectValue());
        image1 = GraphicsUtilities.toCompatibleImage(image1);
        if(!restrictedImageIds.contains(selectedImage.getImageID())){
            if(geoImage.getImageID() ==
selectedImage.getImageID()){
                System.out.println("geldi 1");
            }
        }
        g.setComposite((AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1f));
        g.drawImage(image1, xx, yy,null);
        g.setColor(Color.RED);
        g.drawRect(rect.x, rect.y, rect.width,
rect.height);
    }
}
}
g.dispose();

```

```

}

@Override
public Dimension getPreferredSize() {
    int w = (int)(scale * compositeImageWidth);
    int h = (int)(scale * compositeImageHeight);
    return new Dimension(w, h);
}

private void paintImage(){
    if (cpController.isSelect()) {
        refreshMosaic();
        Graphics2D g3 = (Graphics2D) image.createGraphics();
        selectImage(g3, clickedX, clickedY);
        System.out.println("Select");
    } else if(cpController.isVector()){
        isVector = false;
        if(cpController.getDiffFrameCheckBox().isSelected()){
            VectorFrame vectorFrame = new
VectorFrame(vectorPoints);

            vectorFrame.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
            vectorFrame.setVisible(true);
        } else{
            generalGraphics = (Graphics2D) this.getGraphics();
            isVector = false;
            generalGraphics.setColor(Color.RED);
            for(List<Point> pList : vectorPoints){
                if(pList != null){
                    for(Point p : pList){
                        generalGraphics.fillRect(p.x,p.y, 2,
2);
                        System.out.println("X,Y : " + p.x + " ,
"+ p.y );
                    }
                }
            }
            System.out.println("Çizdi...");
        }
    } else if(cpController.isSelectPoint()){
        System.out.println("Select Point");
        //Graphics2D g3 = (Graphics2D) image.createGraphics();
        generalGraphics = (Graphics2D) this.getGraphics();
        generalGraphics.setColor(Color.RED);
        generalGraphics.fillRect(clickedX, clickedY, 5, 5);
        if(lastClickedX != -1 && lastClickedY != -1){

```

```

        generalGraphics.drawLine(lastClickedX,
lastClickedY, clickedX, clickedY);
    }
    lastClickedX = clickedX;
    lastClickedY = clickedY;
}
}

public void paint(Graphics g) {
    super.paint(g);
    generalGraphics = (Graphics2D) g;
    Graphics2D g2 = (Graphics2D) g;
    if(cpController.isMosaicing()){
        try {
            if (image == null) {
                image = new
BufferedImage(compositeImageWidth,compositeImageHeight,
BufferedImage.TYPE_INT_RGB);
                cache =
cacheConfig.getCacheManager().getCache("ImageCache");
                byte[] imageInByte =
ImageOperationUtils.imageToArray(image);
                Element element = new
Element(ImageOperationUtils.getCounter(), imageInByte);
                cache.put(element);
            }
            Graphics2D g3 = image.createGraphics();
            g3.setComposite(AlphaComposite.Clear);
            g3.fillRect(0, 0, compositeImageWidth,
compositeImageHeight);
            g3.setComposite(BlendComposite.Lighten);
            for (GeoImage geoImage :
ImageOperationUtils.getImageList()) {
                if(geoImage != null &&
!restrictedImageIds.contains(geoImage.getImageID())){
                    int x = (int) (geoImage.getX()
- geoOp.getLeft());
                    int y = (int) (geoOp.getTop()
- geoImage.getY());
                    Element element =
cache.get(geoImage.getImageID());
                    BufferedImage image1 =
ImageOperationUtils.byteArraytoImage((byte[]) element.getObjectValue());
                    image1 =
GraphicsUtilities.toCompatibleImage(image1);
                    g3.drawImage(image1, x, y,
null);
                    image1 = null;

```

```

        System.gc();
        geoImage.setRect(new
Rectangle(x, y, geoImage.getWidth(), geoImage.getHeight()));
    }
}
} catch (Exception e) {
    e.printStackTrace();
}

cpController.setMosaicing(false);
repaint = true;
}
if(repaint){
    refreshMosaic();
}
g2.dispose();
};

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

}

public void setComposite(Composite composite) {
    if (composite != null) {
        this.composite = composite;
        this.repaint = true;
        repaint();
        revalidate();
    }
}

public Composite getComposite() {
    return this.composite;
}

public double getScale() {
    return scale;
}

public void setScale(double s)
{
    scale = s;
    revalidate();    // update the scroll pane
    repaint();
}

public BufferedImage getImage() {
    return image;
}

```

```

}
public void setImage(BufferedImage image) {
    this.image = image;
}

public GeoImage getSelectedImage() {
    return selectedImage;
}

public void setSelectedImage(GeoImage selectedImage) {
    this.selectedImage = selectedImage;
}

public void toVectorBySelectImage(){
    vectorImagePath = "C:\\temp.jpg";
    Element element = cache.get(getSelectedImage().getImageID());
    BufferedImage image =
ImageOperationUtils.byteArraytoImage((byte[])element.getObjectValue());
    image = GraphicsUtilities.toCompatibleImage(image);
    ImageWriters imageWrite = ImageWriters.getInstance();
    imageWrite.exportImage(vectorImagePath, image, "jpg");
    isVector = true;
    isSelectedImage = false;
    defaultPaint = true;
    BwBoundary m;
    try {
        m = new BwBoundary();

        int c = 1;
        Object [] result = m.bwboundary(c, vectorImagePath);
        MWCellArray mc = (MWCellArray)result[0];
        int x = (int) (selectedImage.getX() - geoOp.getLeft());
        int y = (int) (geoOp.getTop() - selectedImage.getY());
        for(int i=1;i<=mc.numberOfElements();i++){
            List<Point> tempList = new ArrayList<Point>();
            int [] index = {i,1};
            Object n = mc.get(index);
            double[][] k = (double[][]) n;
            int M = k.length;
            for(int j=0;j<M;j++){
                int xx = x + (int)k[j][1];
                int yy = y + (int)k[j][0];
                tempList.add(new Point(xx, yy));
            }
            vectorPoints.add(tempList);
        }

        } catch (Exception e) {
            System.out.println("HATA");

```



```

    }
    restrictedImageIds.add(selectedImage.getImageID());
    paintImage();
}

public void toVectorBySelectedPoints(){
    repaint = true;
    List<Point> tempList = null;
    int tempSumDistance = Integer.MAX_VALUE;
    vectorImagePath = "C:\\temp.jpg";
    Dimension dm = getSelectedPointDimension(clickedPoints);
    System.out.println("Point : " + leftPoint.x + " , " + topPoint.y + "
Dimension : " + dm.width + " , " + dm.height);
    BufferedImage croppedImage =
sourceImage.getSubimage(leftPoint.x, topPoint.y, dm.width, dm.height);
    ImageWriters imageWrite = ImageWriters.getInstance();
    imageWrite.exportImage(vectorImagePath, croppedImage, "jpg");
    BwBoundary m;
    try {
    m = new BwBoundary();
    int c = 1;
    Object [] result = m.bwboundary(c, vectorImagePath);
    MWCellArray mc = (MWCellArray)result[0];
    for(int i=1;i<=mc.numberofElements();i++){
        List<Point> tempList2 = new ArrayList<Point>();
        int [] index = {i,1};
        Object n = mc.get(index);
        double[][] k = (double[][]) n;
        int M = k.length;
        for(int j=0;j<M;j++){
            int xx = leftPoint.x + (int)k[j][1];
            int yy = topPoint.y + (int)k[j][0];
            tempList2.add(new Point(xx, yy));
        }
        vectorPoints.add(tempList2);
    }
    } catch (Exception e) {
        System.out.println("HATA");
    }

    for(List<Point> pList : vectorPoints){
        int closestDistance = Integer.MAX_VALUE;
        int sumDistance = 0;
        for(Point p : pList){
            for(Point p2 : clickedPoints){
                int a = (int) Math.sqrt(((p.x - p2.x)*(p.x -
p2.x))+((p.y-p2.y)*(p.y-p2.y)));
                if(closestDistance > a){
                    closestDistance = a;

```

```

        }
    }
    sumDistance = sumDistance + closestDistance;
}
if(tempSumDistance > sumDistance){
    tempSumDistance = sumDistance;
    tempList = pList;
}

}
vectorPoints.clear();
vectorPoints.add(tempList);
paintImage();
}

public List<List<Point>> getVectorPoints() {
    return vectorPoints;
}

public void setVectorPoints(List<List<Point>> vectorPoints) {
    this.vectorPoints = vectorPoints;
}

public void refreshMosaic(){
    int w = getWidth();
    int h = getHeight();
    generalGraphics.clearRect(0, w, 0, h);
    double screenX = (w - scale * image.getWidth()) / 2;
    double screenY = (h - scale * image.getHeight()) / 2;
    AffineTransform at =
AffineTransform.getTranslateInstance(screenX, screenY);
    at.scale(scale, scale);
    generalGraphics.drawRenderedImage(image, at);
    sourceImage = image;
    clickedPoints.clear();
}

public void clearSelectedPoints(){
    clickedPoints.clear();
    lastClickedX = -1;
    lastClickedY = -1;
}

private Dimension getSelectedPointDimension(List<Point>
selectedPoint){
    Collections.sort(selectedPoint,new PointXCompare());
    leftPoint = selectedPoint.get(0);
    int minX = selectedPoint.get(0).x;
    int maxX = selectedPoint.get(selectedPoint.size()-1).x;

```

```

        Collections.sort(selectedPoint,new PointYCompare());
        topPoint = selectedPoint.get(0);
        int minY = selectedPoint.get(0).y;
        int maxY = selectedPoint.get(selectedPoint.size()-1).y;
        Dimension dm = new Dimension(maxX-minX,maxY-minY);
        return dm;
    }
    private static class PointXCompare implements Comparator<Point>{

        @Override
        public int compare(Point p1, Point p2) {
            return p1.x - p2.x;
        }
    }

    private static class PointYCompare implements Comparator<Point>{

        @Override
        public int compare(Point p1, Point p2) {
            return p1.y - p2.y;
        }
    }
}

```

Vektörleştirme işlemlerini yapan matlab kodu:

```

function[boundary]=bwBoundaries(path)
% Reading all three images
F = imread(path);
a = rgb2gray(F);
threshold = graythresh(a);
bw = im2bw(a,threshold);
bw = bwareaopen(bw,300);
se = strel('disk',2);
bw = imclose(bw,se);
bw2 = imopen(bw,se);
bw = imfill(bw2,'holes');
alan = bwareaopen(bw,10000);
[B,L,N] = bwboundaries(alan);
boundary1 = B;
end

```

EK-C

Bu ekte, bir poligonun alanının nasıl hesaplandığından bahsedilmiştir. Bu çalışmada birleştirilmiş uydu görüntüsünden çıkarılan nesnedeki alan hesabı postGIS veritabanının sağlamış olduğu “ST_area()” fonskyionuyla sağlanmaktadır. Şekil C.1’de koordinatları verilen bir poligonun alan hesabını yapan postGIS kodu bulunmaktadır.

```
SELECT ST_Area(the_geom) As sqft, FROM (SELECT  
ST_GeomFromText('POLYGON((743238 2967416, 743238 2967450, 743265  
2967450, 743265.625 2967416, 743238 2967416))',2249) ) As foo(the_geom);
```

Şekil C.1. Bir poligonun alan hesabını yapan postGIS kodu

Matematiksel olarak bir poligonun alanı ise;

A: Poligonun Alanı

n: Poligonun Kenar Sayısı

$$(x_i, y_i), i = 1,2,3,\dots,n \quad (C.1)$$

$$A = \frac{1}{2} | x_1y_2 + x_2y_3 + x_3y_4 + \dots + x_{n-1}y_n | \quad (C.2)$$

Denklem (C.1) ve (C.2)’de görüldüğü şekilde hesaplanmaktadır.

KİŞİSEL YAYIN VE ESERLER

- [1] Arar Ö. F., Öncü A., Akpınar S., Yalçın Ö., **Mert Ü.**, Sadak M. S., Kosunalp S., A Flexible E-exam Framework and Air Traffic Controller Selection System, *3rd World Conference on Information Technology (WCIT-2012)*, Barcelona, Spain, 14-16 November 2012.
- [2] Sayar A., Eken S., **Mert U.**, Registering landsat-8 mosaic images: A case study on the Marmara Sea, *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, Ankara, Türkiye, 7-9 November 2013.
- [3] Öncü A., Arar Ö. F., Akpınar S., Yalçın Ö., **Mert Ü.**, Sadak M. S., Öztürk U., Yetenek Ölçümüne Yönelik Elektronik Sınav Sisteminin Geliştirilmesi ve Hava Trafik Kontrolör Seçimine Uygulanması, *VIII. Ulusal Yazılım Mühendisliği Sempozyumu*, Güzelyurt, KKTC, 8-10 Eylül 2014.
- [4] Sayar A., Eken S., **Mert U.**, Tiling of Satellite Images to Capture an Island Object, *15th International Conference on Engineering Applications of Neural Networks*, Sofia, Bulgaria , 5-7 September 2014.

ÖZGEÇMİŞ

1987 yılında İstanbul'un Eyüp ilçesinde doğdu. İlköğrenim, ortaokulu ve lise öğrenimini İstanbul Sultanbeyli'de tamamladı. 2005 yılında girdiği Çanakkale Onsekiz Mart Üniversitesi Bilgisayar Mühendisliği bölümünden 2011 yılında mezun oldu. 2010-2011 yılları arasında çeşitli özel sektör firmalarında yazılım mühendisi olarak çalıştıktan sonra 2011 yılından itibaren TÜBİTAK BİLGEM bünyesindeki Bilişim Teknolojileri Enstitüsü'nde Araştırmacı olarak çalışmaktadır. 2012 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'ndaki Yüksek Lisans eğitimine devam etmektedir.