Düzce Üniversitesi
Bilim ve Teknoloji Dergisi

Araştırma Makalesi

DocDig: Dijitalleştirilmiş Dokümanlarda İçerik Tabanlı Figür Arama

Burak ATAY [a], Büşra Ceren SÖNMEZ [a], Süleyman EKEN [a,*], Ahmet SAYAR [a]

[a] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Kocaeli Üniversitesi, Kocaeli, TÜRKİYE*
** Sorumlu yazarın e-post adresi: suleyman.eken@kocaeli.edu.tr*

ÖZET

Örüntü tanıma psikolojiden biyometriye, biyoenformatikten gen ifadelerinin analizine, trafikten hesaplamalı finansa kadar birçok alanda kullanılmaktadır. Optik Karakter Tanıma da bu alanlardan bir tanesidir. Kamu ve özel birçok firma, arşivlerindeki klasörlenmiş verilerini taratarak dijital hale getirmekte ve bunun için emek yoğun çalışmalar yapmaktadır. Ancak resim olarak sayısallaştırılan bu verilerin içerik olarak aranması ve işlenmesi ancak operatörlerin manuel olarak taranan resim verisine meta veri eklemesi ile kısmi olarak gerçekleşmektedir. Bu çalışmada, resim olarak taranan ve sayısal hale getirilen büyük miktarlardaki bu dokümanlar üzerinde içerik bazlı figür aramalarını mümkün kılan bir mimari geliştirdik. Kullanıcı, bazı anahtar kelimelerle arama yaparak sayısal dökümanlardaki ilgili figürleri başlıklarıyla beraber görüntüleyebilmektedir. Sistemin yapılabilirlik ve başarımı farklı veri setleri üzerinde test edilmiş, başarılı sonuçlar elde edilmiştir.

Anahtar Kelimeler: *Doküman dijitalleştirme, Figür/resim saptama, Başlık saptama, İçerik tabanlı arama, MongoDB*

DocDig: Content Based Figure Search in Digitized Documents

ABSTRACT

Pattern recognition is used in many areas, from psychology to biometrics, analysis of gene expressions from bioinformatics, from traffic to finance calculated. Optical Character Recognition is also one of these areas. Many public and private firms digitize their archived data and make labor-intensive studies for this purpose. However, the retrieval and processing of these data, which are digitized as images, is only partially realized by adding metadata to the manually scanned image data. In this work, we developed an architecture that makes content-based figure searches possible on these scanned documents in large quantities. The user can search with some keywords and display related figures in digital documents with their captions. The feasibility and performance of the system have been tested on different data sets and successful results have been obtained.

Keywords: Document digitization, Figure/picture detection, Caption detection, Content based search, MongoDB

Geliş: 21/07/2017, Düzeltme: 20/09/2017, Kabul: 02/10/2017

I. GİRİŞ

Günümüzde resmi veya gayri resmi arşivlerin sanal ortama çevrilmesi revaçta olan bir işlemdir. Kâğıt kullanılarak elde edilen belgelerden bilgisayarlı depolama ve veri alma sistemleri gibi elektronik belge yönetim sistemlerine geçiş, pek çok avantaj dolayısıyla tercih edilmektedir. Bunun yanında gün geçtikçe artarak devasa boyutlara ulaşan arşivlerin işlenebilmesi kurumlar adına büyük önem arz etmektedir. Şirketler ve kütüphaneler, Google'ın iyi bilinen bir örnek olmasıyla milyonlarca sayfa taramış bulunuyor. Arşiv verilerinin işlenmesi sorunu insan eliyle yapılamayacak kadar fazla iş gücüne ihtiyaç duymaktadır. Bilgisayar ortamı; belgeleme, belge güncelleme gibi işlemleri verimli bir şekilde gerçekleştirmeye olanak sağlamanın yanında belge işlemede de insan kriterinden çok daha sağlıklı olacaktır [1]. Bilgisayar ortamına aktarılan bu veriler üzerinde arama ve indeksleme yapmak dokümanları efektif kullanmak açısından kaçınılmaz olmaktadır.

Dijital görüntü ve videolardaki içerik temelli analiz son zamanlarda çok fazla ilgilenilen konular arasındadır. Dijital haldeki doküman içindeki figürlerin bulunması, ilgili dokümanın indekslenmesi ve içerik bazlı aranmasına olanak sağlamaktadır. Literatürde içerik olarak; renkler, objelerin şekilleri veya herhangi bir görüntüden türetilmiş bilgi ele alınmaktadır [2]. Bu çalışmada sayısal haldeki bir doküman içindeki figürlere ilişkin başlıklar ve figürlerin kendileri, içerik olarak kullanılmaktadır.
Genel olarak dijitalleştirilmiş doküman içindeki görüntüleri bulmak zor bir problemdir. Dijital haldeki bu dokümanlar basit figürler içerebildiği gibi karmaşık ve çakışan grafik tasarımlar da barındırabilmektedir. Bu makalede oldukça basit tasarımlara sahip bilimsel çalışma metinlerinden ve raporlardan figürlerin çıkarılması ile ilgilenmekteyiz. Bahsedilen türdeki dokümanları işleyebilmek için öncelikle sayfa segmentasyonu yapılması gerekmektedir. Sayfa segmentasyonunun çıktısının doğruluğu, sonraki tüm analiz ve tanıma süreçlerinin temelini oluşturduğundan dolayı önemlidir.

Yaklaşımımızda figür saptama işlemini zorlaştırmasından dolayı, belge içerisindeki metinler Optik Karakter Tanıma (Optical Character Recognition-OCR) kullanılarak algılanmakta ve belgeden temizlenmektedir. Metinden arındırılmış belgede figürlerin sınırlarının bulunması ile figür saptama gerçekleşmiş olur. Bir iyileştirme adımı olarak arka plan rengi arındırılmakta ve atık veriler temizlenmektedir. Başka bir iyileştirme adımı ise resimlerin altında bulunan başlıklara göre geliştirilen karar mekanizmasıyla atık verilerin figür olarak algılanmaması sağlanmıştır. Çalışmanın genel katkıları şu şekildedir:

- Görüntü olarak verilen dokümanlardan figür ve başlıklarının tespit edilmesi,
- Elde edilen bilgilerin indeksleme ve arama amaçlı MongoDB'de tutulması

Makalenin geri kalan kısmı şu şekilde organize edilmiştir: II. bölümde literatürde var olan figür ve metinleri birbirinden ayıran yaklaşımlardan bahsedilecektir. III. bölümde önerilen DocDig mimarisi detaylı bir şekilde anlatılacaktır. IV. bölümde ise performans değerlendirmesi yapılıp son bölümde sonuçlar tartışılmıştır.

II. İLGİLİ ÇALIŞMALAR

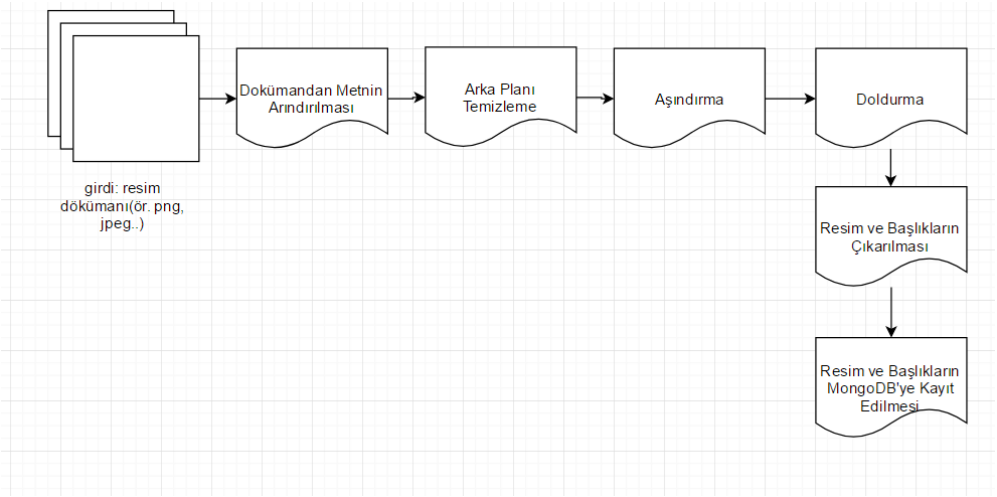
Dijitalleştirilmiş dokümanlardan figürlerin saptanması, metin ve figürlerin birbirlerinden ayrıştırılmasına yönelik literatürde çok fazla çalışma bulunmamaktadır [3-6]. Bu tür çalışmalar genellikle bağlı bileşenlerin geometrik özellikleri veya matematiksel morfoloji gibi düşük seviye

analizler gerektirir. Ayrıca nesnelerin doküman içindeki yerini/yerleşimini araştıran daha karmaşık süreçler gerektiren doküman düzen analizi de literatürde var olan çalışmalardır [7]. Ayrıca sayfa segmentasyonu ve bölge tespiti için makine öğrenmesi yöntemlerinden de yararlanılmaktadır. Burns ve Corso [8] doküman içindeki figür, metin ve boşluk gibi bölgeleri tespit etmek için sözlük öğrenmesi yaklaşımını geliştirmiştir. Chuai-Aree ve diğerleri [9] fuzzy C-Mean yöntemi ile figür tespitini sağlamışlardır. Ayrıca literatürde metin ve figürlerin birbirlerinden ayrılması ile doku ve bölge tabanlı birçok yöntem araştırmacılar tarafından önerilmiştir [10-11].

Önerilen yaklaşımda dokümandan metnin ayrılması yöntemine başvurulmuştur. Böylelikle metinden arındırılmış belgeden figürlerin saptamasının çok daha kolay yapılabilmesi sağlanmıştır. Metinden arındırma işlemi için OCR metin rengini değiştirme metodu yol gösterici olmuştur [2]. Bu metotla metin renginin arka planla aynı yapılması sağlanarak belgeden metnin yok edilmesi sağlanmıştır. Metinden temizlenmiş belge üzerinde de NCUT (Normalized Cuts) [12] kümeleme algoritması kullanılarak resimlerin algılanması amaçlanmıştır. Önerilen yaklaşımda metinden temizlenmiş belge üzerinde figür tespiti için belge x ve y koordinatlarına göre taranır. Tarama sırasında beyazdan siyah piksele geçilmesi figürün başladığını; siyah pikselden beyaz piksele geçilmesi figürün bittiğini gösterir. Bu şekilde figürün belge içerisindeki tam koordinatları tespit edilir. Bu çözümde pikseller tek tek gezilmek yerine 20 aralıklarla gezildiği için 20 piksellik alandaki olası hataları yok sayması sağlanır. Doküman içinde tespit edilen figürler ve başlıkları MongoDB'de tutularak kullanıcı etkileşimli sorgulama/arama imkânı sunulmuştur. Son zamanlarda araştırmacılar karmaşık arkaplana ait görüntülerden metin saptama ile ilgili çalışmalar yapmışlardır [13-14]. Takip eden bölümde mimari detaylı olarak açıklanacaktır.

III. DOCDİG MİMARİSİ

Şekil 1 önerilen sistemin mimarisini göstermektedir. İlk adım olarak metin kısmı figürden/görselden arındırılır. Daha sonra arka plan renginin beyazdan başka bir renk olabilme ihtimaline karşı arka plan temizlemesi yapılır. Görsellerin doğru tespiti için çöp verileri-yazılardan arta kalan ve segmentasyon işlemini engelleyecek parçacıklar- temizlemek adına sırasıyla aşındırma ve doldurma işlemleri uygulanır. Doküman x ve y koordinatlarından taranarak görseller ve görsellere ait başlıkların elde edilmesi sağlanır. Son olarak elde edilen bu görsel ve başlıklar doküman numarasıyla birlikte MongoDB'ye kayıt edilir. Geliştirilen algoritma doc, pdf gibi segmentasyonu daha kolay olan doküman tiplerinin yerine jpeg, png, bmp gibi görüntü dokümanları üzerinde çalışmaktadır. Mimarinin alt adımları ilerleyen kısımlarda daha ayrıntılı anlatılmaktadır.



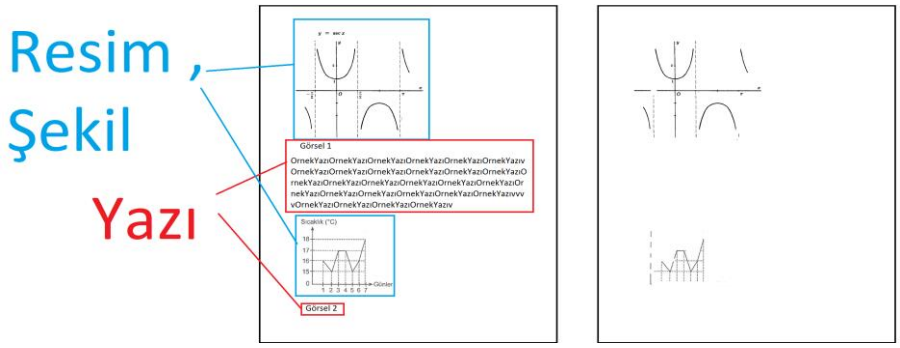
Şekil 1. *Önerilen sistem mimarisi.*

A. FİGÜR VE BAŞLIKLARIN ÇIKARILMASI

İlk olarak dijitalleştirilmiş dokümandan Tesseract OCR motorunun [15] optik karaktere benzettiği bütün verilerin arındırılması işlemi gerçekleştirilmiştir. Artakalan sayfa görüntüsü arka plan renginden arındırılıp siyah beyaz hale çevrilmektedir. Daha sonrasında aşındırma algoritması ile aşındırılan görüntü çöp verilerin temizlenmesi için üzerinde doldurma işlemi uygulanır ve ayrılmamış resim görüntüleri segmente işlemine tabi tutularak birbirlerinden ayrılırlar.
Figürlerin dijitalleştirilmiş dokümandan ayrılması aşamasında işlemi zorlaştırabilecek ve çöp veri oluşturabilecek tanımlanabilen karakter parçaları OCR motoru ile yerleri tespit edilip üzerlerine metinin uzunluğu karakterlerin yerlerine uygun olarak çerçeve içine alınır. Çerçeve içerisine alınan metin beyaza boyanarak yok edilir. Çerçeveler resim sayfa görüntüsü içerisinde bulunan grafik tablo veya resim içerisinde de oluşabilir; fakat resim içerisinde büyük bir tahribata yol açmamaktadır. Algoritma 1, metin temizleme ile ilgili yalancı kodu; Şekil 2 ise örnek bir girdi doküman üzerinde metin temizleme işlemini göstermektedir.

Algoritma 1. *Yazı Temizleme.*

Girdi: Resim
Çıktı: Yazıdan arındırılmış resim
1. **var** resim
2. **while** (yazivarmi(resim))
3. **var** yazi ← yaziyiBul(resim)
4. resim ← yazisil(resim, yazi)
5. **end**



Şekil 2. *Metin temizleme adımı.*

Algoritma 2'de gösterildiği üzere dokümanların arka plan rengi barındırması ihtimalinden dolayı belirtilen hassasiyete göre bir renk skalası oluşturup bu skala içerisinden siyah ve beyaz haricinde bulunabilecek (bulunmayabilir) rengin dokümanda beyaz hale geçirilmesini sağlamaktadır. Gerçekleşen bu işlem sayesinde arka plan renginden ve metin verisinden ayrıştırılmış olası resimler topluluğu ve gürültüden oluşan ham görüntü elde edilmiş olur.

Ham haldeki dokümanların figürler haricindeki bölgeleri segmente etmemeleri için ham doküman içerdiği kirlilikten kurtarılmalıdır. Ham dokümanın kirlilikten kurtarılmasından önce girdi siyah ve beyaza boyanır. Herhangi bir renk segmenti içeren bütün renkler siyah yapılmakta geri kalanlar beyaz olarak bırakılmaktadır. Siyah beyaz görüntü üzerinde belirtilen hassasiyet değerine göre 3X3'lük seçilen bir piksel tablosunda ortadaki piksel; etrafındaki piksellerden herhangi bir tanesi beyaz ise beyaz hale getirilerek aşındırma işlemi gerçekleştirilir. Bir alanda beyaz piksel bulunması halinde çevresinde bulunan bütün piksellerin beyaz hale getirilmesi ve aksi durumda siyah kalmasına müsaade edilerek siyah beyaz görüntü aşındırılmaktadır.

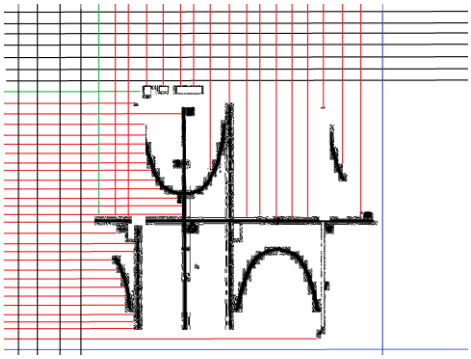
Algoritma 2. *Arkaplanı Temizleme.*

Girdi: Yazıdan arındırılmış resim
Çıktı: Arkaplanı temizlenmiş resim

```
1.  var frekanslar ← renkTekrarınıBul(resim)
2.  var sıralanmısFrekanslar ← sırala (frekans)
3.  var encokTekraredenRenk
4.  if beyaz = sıralanmısFrekanslar [1] || siyah =
    sıralanmısFrekanslar [1] than
5.      if  beyaz = sıralanmısFrekanslar [2] || siyah =
    sıralanmısFrekanslar [2] than
6.              encokTekraredenRenk ← sıralanmısFrekanslar [3]
7.      end
8.      else
9.              encokTekraredenRenk ← sıralanmısFrekanslar [2]
10.     end
11. end
12. else
13.     encokTekraredenRenk ← sıralanmısFrekanslar [1]
14.  end
15. resim ← rengiBeyazaCevir(resim, encokTekraredenRenk)
```

Aşındırılmış görüntü olası resimlerin boyutları ve parçaları arasında kopukluğa neden olabileceği için geri kalan bütün veri aşındırma fonksiyonunun tersi kullanılarak doldurma işlemi ile genişletilmektedir. 3X3'lük seçilen bir piksel tablosunda ortadaki piksel; etrafındaki piksellerden herhangi bir tanesinin siyah olması durumunda siyah hale getirilir. Böylece doldurma işlemi gerçekleştirilmekte ve ayrılmamış olası resim topluluğu elde edilmektedir.

Elde edilen ayrılmamış olası resim topluluğu için yapılacak segmente işlemi Algoritma 3'te gösterildiği gibi şu şekilde gerçekleştirilmiştir. İşlemlerden geçmiş olan görüntü üzerinde soldan sağa ve yukarıdan aşağıya doğru inilirken figürün ilk ve son satırları tespit edilir. İlk ve son satır sınırları içerisinde ilk ve son sütun tespit edilir. Sonrasında ise figür çıkarma işlemi gerçekleştirilmesi için figürün bir başlığa sahip olması durumunda kesilip alınır. Eğer bir başlığa sahip değilse bu alan resim aramasında kullanılamayacağı için dikkate alınmaz (Şekil 3'e bakınız).



Şekil 3. *Figür koordinat tespiti (siyahlar resme dokunmayan, kırmızılar dokunan, yeşil çizgiler ilk dokunmaları, lacivertler ise son dokunmaları yatay veya dikey olarak ifade eder).*

Algoritma 3. *Segmente et.*

Girdi: Doldurulmuş resim

Çıktı: Kesilmiş resim

1. **Var** klonResim ← image , ilkx←0, sonx←0, ilky←0, sony←0**,** kesilmisResimler
2. **foreach** pixel ∈ then
3. **if** pixelSiyahmi (pixel) **then**
4. **if** ilkx=0 **then** ilkx ← xKoordinatiniGetir(pixel)
5. **end if**
6. **end if**
7. **if** pixelBeyazmi(pixel)then
8. **if** (ilkx != 0) **then** sonx ← xKoordinatiniGetir (pixel)
9. **end if**
10. **end foreach**

```
11. foreach pixel ∈ then
12.       if pixelSiyahmi (pixel)
13.             if ilky=0 then  ilky ← yKoordinatiniGetir (pixel)
14.             end if
15.       end if
16.       if pixelBeyazmi (pixel)then
17.             if (ilky!=0) then sony← yKoordinatiniGetir (pixel)
18.              end if
19.  end foreach
20. Var geciciResim = resimKes(resim, ilkx, ilky, sonx, sony)
21. if altindaCaptionVarmi(geciciResim,resim) then kesilmisResimler =
    kesilmisResimler + geciciResim
22. end if
```

B. ELDE EDİLEN FİGÜR ve BAŞLIKLARIN MongoDB'de TUTULMASI

Elde edilen figürlerin uygun bir şekilde depolanabilmesi için veri tabanı seçimi önemli bir adımdır. Bu sebeple depolama işlemi için esnek veri modeli yapısına sahip, ölçeklendirilebilir ve güvenilir bir veri tabanı gerekmektedir. Bu aşamada bir NoSQL veritabanı olan MongoDB kullanılmıştır. NoSQL veritabanlarının pek çok farklı çeşidi bulunmaktadır. Bunlardan en temel olanları: (i) sütunlar şeklinde tutulanlar, (ii) anahtar-değer şeklinde depolama yapanlar, (iii) doküman tabanlılar, (iv) graf tabanlılar. Doküman tabanlı sistemler Lotus Notes'tan esinlenmişler ve anahtar-değer depolamaya oldukça benzerlik göstermektedirler. Veri tutma modeli, anahtar-değer çiftlerinin toplanmasından oluşan versiyonlanmış haldeki dokümanlardır. Yarı yapılandırılmış dokümanlar JSON (JavaScript Object Notation) formatında tutulur. Sorgulama işlemini verimli bir şekilde sağlarlar. MongoDB ölçeklenebilir, doküman tabanlı, C++ ile geliştirilmiş açık kaynak kodlu NoSQL bir veritabanıdır. Özellikle hız gerektiren ve klasik VTYS'nin çok yavaş kaldığı yerlerde kullanılmaktadır. Kullanım alanları arasında; yüksek hacim/içerikli problemler, analiz için veri saklanması, caching sistemleri, web içerik yönetim sistemleri, web yorum/etiket saklama ve yönetme yer almaktadır. 10gen şirketi (yeni adı ile MongoDB, Inc.), Google Uygulama Motoru'na benzer bir servis oluşturduğu sırada, MongoDB geliştirmesi de 2007 yılında başlamıştı. Nisan 2017 itibari ile de en son versiyonu 3.4.4'u yayınlamışlardır [16].

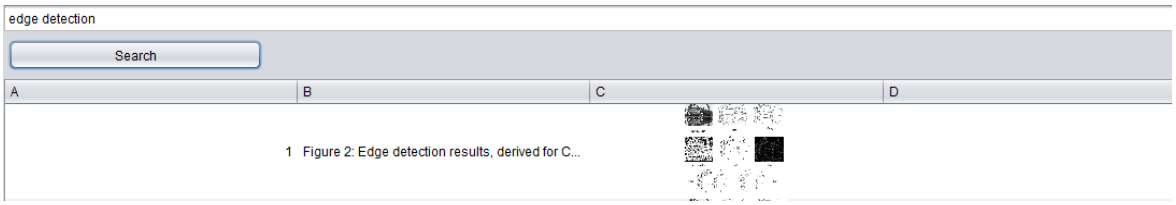
MongoDB veri kaydetmek için dahili bellek kullanır, bu veriye erişimin de hızlı olmasını sağlamaktadır. MongoDB üzerinde resim depolamak için GridFs'ten yararlanıldı. GridFS normalde çok büyük boyutlu (min 16 GB) dosyaların parçalara ayrılarak saklanmasını sağlamak amacıyla tasarlanmıştır. Ancak küçük boyutlu dosyalarda da herhangi bir sorun çıkarmamaktadır.

Figürler MongoDB'ye doküman numarası, başlık ve figür bilgilerini içerecek şekilde kaydedilmiştir. Doküman numarası figürün hangi dokümana ait olduğunu belirtmek için tutulur. Başık ise belgeden figürün elde edilme aşamasında çıkartılan figüre ait bize bilgi verebilecek olan başlığını içermektedir.

C. İÇERİK TABANLI ARAMA MEKANİZMASI

Geliştirilen arama mekanizması veritabanına kaydedilmiş olan başlık kısmı üzerinde işlem yapmaktatır. MongoDB sorgulama dilinin basitliğiyle bu aşamada büyük kolaylık sağlamaktadır. Aratılan bir kelimenin başlık kısmında geçiyor olup olmadığına bakılır. Eğer aranılan kelimeye başlıklarda rastlanırsa bu başlığı barındıran figür ve figür bilgileri ekranda listelenir. Sonuç birden fazla olabilir. Şekil 4'te yapılan bir sorguya karşılık gelen bilgiler Şekil 5'te listelenmiştir.

Şekil 4. *Sorgu sonuç ara yüzü (sorgu marie).*



Şekil 5. *Sorgu sonuç ara yüzü (sorgu edge detection).*

IV. DENEYSEL SONUÇLAR

Önerilen yaklaşımın doğruluğunu test edebilmek için farklı senaryolara göre test verileri toplanmıştır. OCR çalışma prensibinden dolayı girdi verilerin çözünürlüğünün yüksek olması gerekmektedir. Aksi halde OCR harfleri tanıyamayabilir ve sonuçlar tutarsızlaşabilir. Test edilen senaryolar şu şekildedir:

- Senaryo 1'de her görüntü sayfasında sadece bir tane figür bulunacak şekilde toplam 100 adet girdi işleme alınmıştır.
- Senaryo 2'da Senaryo 1'deki koşulları sağlayan girdi sayısı 200'e çıkarılmıştır.
- Senaryo 3'te her görüntü sayfası içerisinde birden fazla figür içerecek şekilde 100 girdi işleme alınmıştır.
- Senaryo 4'te Senaryo 3'teki koşulları sağlayacak şekilde 200 girdi verisi kullanılmıştır.

Senaryolara göre programın performans sonuçları Tablo 1'de gösterilmiştir:

Tablo 1. *Performans değerlendirilmesi.*

	İşlem zamanı	**Doğru bulunan figür sayısı**	**Başlığı doğru bulunan**	**Başlığı içeren**
Senaryo 1	27 dk 34 sn	72	13	68
Senaryo 2	68 dk 13 sn	133	25	110
Senaryo 3	38 dk 42 sn	103	20	80
Senaryo 4	73 dk 51 sn	198	43	168

Farklı adetlerde kayıtlar göz önünde bulundurularak veri tabanında yapılan arama işlemlerindeki cevap verme süresi fark edilmeyecek kadar küçük bir zaman diliminde gerçekleşmektedir. Bu zaman dilimi 0.01 ile 0.1sn olarak değişmektedir.

Yazı boyutu 7-27 arasında değişen 10 farklı Times New Roman tipinde yazı içeren dijital sayfalarda farklı çözünürlük için performans araştırması yapılmıştır:

- 595 x 841 piksel, 72 piksel/inç çözünürlüğünde yazı tespit edilememekte ve doğru sonuç üretilememektedir. Doküman başına ortalama 7 sn süre harcanır.

- 1240 x 1754 piksel, 150 piksel/inç çözünürlüğü için 10 sonuçtan 7'si için doğru sonuç üretilmektedir. Doküman başına ortalama 11 sn süre harcanır.

- 2480 x 3508 piksel, 300 piksel/inç çözünürlüğünde yapılan işlem için 10 sonuçtan 9'su için doğru şekilde yazı tespit edilmektedir. Doküman başına ortalama 17 sn süre harcanır.

V. SONUÇ VE GELECEK ÇALIŞMALAR

Bu çalışmada, görüntü dokümanlarından resim ve metin tabanlı verilerin elde edilebilmesi ele alındı. Uygulama jpg, png gibi görüntü dosyaları üzerinde çalıştırıldı ve oluşan çıktılar incelendi. Geliştirilen bu uygulama ile birlikte gittikçe artan sanal tabanlı arşivlemenin çok daha verimli bir şekilde analiz edilebilmesine büyük katkı sağlayacağı düşünülmektedir. Tablo grafik ve resimlerin hepsinin fazladan konfigürasyon gerektirmeden veriden ayrılması, uygulamanın karşılaşacağı karmaşıklığı arttırmaktadır. Bu üç birbirinden farklı veri türü de %60'dan yüksek bir oranda ve verimlilikle çalışmaktadır. Bu oran; girebilecek veri tipine, yazı boyutuna ve belge hakkında olabilecek ayrıntıların bilinmesi halinde daha da artabilir.

Uygulama, OCR karakter tanıma motorunun kabiliyeti ile sınırlıdır. Karakter tanıma motorlarının gelişmesi ile beraber uygulamanın verimi ve elde edilen verinin kalitesi artacaktır. Bu konuda en önemli sorunlardan bir tanesi çözünürlük sorunudur. Yüksek çözünürlük seviyesinde bulunan resimlerin işlenmesi çözünürlük arttıkça resim işleme algoritmalarının daha yavaş çalışmasına neden olmaktadır. Artan girdi sayısı ve girdilerin yüksek çözünürlüklerine cevap verecek şekilde önerilen mimari dağıtık çalışan bir yapıya dönüştürülecektir. Görsellerin içerisinde bulunan resim parçalarının da optik karakterlere benzemesi sorunu bulunmaktadır. Bu sorunun daha sonra dil tanıma modülü

eklenerek engellenmesi hedeflenmektedir. Bu kısımda İngilizce ve Türkçe doğal dil işleme kütüphanelerinden yararlanılarak bir karakter veya kelimeye karşılık gelip gelmediğine bakılacaktır.

IV. KAYNAKLAR

[1] K. Jung, K. I. Kim ve A. K. Jain, "Text information extraction in images and video: A survey," Pattern Recognition, c. 37, s. 5, ss. 977–997, 2004.

[2] C. Patrick, C. Francine ve D. Laurent "Picture Detection in Document Page Images," ACM Symposium on Document Engineering, Manchester, United Kingdom, 2010, ss. 211–214.

[3] S. B. Dan ve R. C. Francine, "Extraction of text-related features for condensing image documents," SPIE 2660, Document Recognition III, San Jose, CA, United States, 1996, ss. 72–88.

[4] L. A. Fletcher ve R. Kasturi "A robust algorithm for text string separation from mixed text/graphics images," IEEE TPAMI, c. 10, s. 6, ss. 910–918, 1988.

[5] C. Najwa-Maria, D. Pascal ve Y. Charles, "A Robust Algorithm for Text Extraction from Images," 39th International Conference on Telecommunications and Signal Processing, Vienna, Austria, 2016, ss. 493–497.

[6] Y. Vikas ve R. Nicolas, "Text extraction in document images: highlight on using corner points," 12th IAPR Workshop on Document Analysis Systems, Santorini, Greece, 2015, ss. 281–286.

[7] F. Shafait, D. Keysers ve T. M. Breue, "Performance evaluation and benchmarking of six page segmentation algorithms," IEEE TPAMI, c. 10, s. 6, ss. 941–954, 2008.

[8] T. J. Burns ve J. J. Corso, "Robust unsupervised segmentation of degraded document images with topic models," Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, ss. 1287–1294.

[9] S. Chuai-Aree, C. Lursinsap, P. Sophatsathit ve S. Siripant, "Fuzzy C-Mean: A Statistical Feature Classification of Text and Image Segmentation Method," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, c. 9, s. 6, ss. 661–671, 2001.

[10] A. Srivastav veJ. Kumar, "Text detection in scene images using stroke width and nearest-neighbor constraints," TENCON 2008, Hyderabad, India, 2008, ss. 1–5.

[11] M. Jaderberg, A. Vedaldi ve A. Zisserman, "Deep Features for Text Spotting," 13th European Conference on Computer Vision, Zurich, Switzerland, 2014, ss. 6–12.

[12] J. Shi ve J. Malik, "Normalized cuts and image segmentation," IEEE TPAMI, c. 22, s. 8, ss. 431–439, 2000.

[13] T. Wang, D. J. Wu, A. Coates, ve A. Y. Ng, "End-to-end text recognition with convolution neural networks," 21st International Conference on Pattern Recognition, Tsukuba, Japan, ss. 3304–3308, 2012.

[14] Y. Zhu, J. Sun ve S. Naoi, "Recognizing natural scene characters by convolutional neural network and bimodal image enhancement," International Workshop on Camera-Based Document Analysis and Recognition, Beijing, China, 2011, ss. 69–82.

[15] Tess4J, (17Haziran 2017) [Online]. Erişim: https://github.com/tesseract-ocr/tesseract

[16] E. Süleyman, K. G. Fidan, S. Ahmet ve K. Adnan, "Doküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB yatay ölçeklenebilirlik karşılaştırması," 7. Mühendislik ve Teknoloji Sempozyumu, Ankara, Türkiye, ss. 1-7, 2014.