KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

PANORAMA İLE ÜRETİLEN PLEVRAL EFÜZYON SİTOPATOLOJİ GÖRÜNTÜLERİ ÜZERİNDE YOLOV3 İLE OTOMATİK ÇEKİRDEK ALGILAMA

YÜKSEK LİSANS TEZİ

Bilgisayar Müh. Büşranur KILIÇ

OCAK 2020
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

PANORAMA İLE ÜRETİLEN PLEVRAL EFÜZYON SİTOPATOLOJİ GÖRÜNTÜLERİ ÜZERİNDE YOLOV3 İLE OTOMATİK ÇEKİRDEK ALGILAMA

Bilgisayar Müh. Büşranur KILIÇ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 20 / 12 /2019
Tezin Savunma Tarihi : 15 / 01 /2020

Tez Danışmanı : Prof. Dr. Murat EKİNCİ

Trabzon 2020

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalında
Büşranur KILIÇ Tarafından Hazırlanan

PANORAMA İLE ÜRETİLEN PLEVRAL EFÜZYON SİTOPATOLOJİ GÖRÜNTÜLERİ ÜZERİNDE YOLOV3 İLE OTOMATİK ÇEKİRDEK ALGILAMA

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 31 / 12 / 2019 gün ve 1834 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Murat EKİNCİ .....

Üye : Prof. Dr. Ünal ÇAKIROĞLU .....

Üye : Dr. Öğr. Üyesi Selen AYAS .....

Prof. Dr. Asim KADIOĞLU
Enstitü Müdürü

ÖNSÖZ

Bu çalışmada plevral efüzyon sitopatolojik görüntülerin derin öğrenme yöntemiyle otomatik algılanması sağlanmıştır. Veri tabanı hazırlama sürecinde ise görüntüler mikroskoptan okunarak panorama yöntemiyle birleştirilmiştir ve bu görüntüden belirli boyutlarda görüntüler kesilerek hazırlanması sağlanmıştır. Bu çalışmada danışmanlığımı üstlenen değerli hocam Prof. Dr. Murat EKİNCİ'ye ilgi, destek ve tecrübelerinden dolayı teşekkürü bir borç bilirim. Çalışmam boyunca bana her türlü desteği sağlayan hocam Arş. Gör. Elif BAYKAL'a ve yüksek lisans eğitimim boyunca sabır, destek ve sevgileriyle yanımda olan aileme ve dostlarıma çok teşekkür ederim. Ayrıca bu yüksek lisans tezi kapsamında yapılan tüm faaliyetler TÜBİTAK 117E961 nolu projedeki burs ve akademik destekleri ile gerçekleştiğinden dolayı TÜBİTAK'a teşekkürlerimi sunarım.

Büşranur KILIÇ

Trabzon 2020

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum "Panorama ile Üretilen Plevral Efüzyon Sitopatoloji Görüntülerinde Yolov3 ile Otomatik Çekirdek Algılama" başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Murat EKİNCİ'nin sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 15/01/2020

Büşranur KILIÇ

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

PANORAMA İLE ÜRETİLEN PLEVRAL EFÜZYON SİTOPATOLOJİ GÖRÜNTÜLERİ ÜZERİNDE YOLOV3 İLE OTOMATİK ÇEKİRDEK ALGILAMA

Büşranur KILIÇ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Murat EKİNCİ
2020, 50 Sayfa, 3 Sayfa Ek

Plevral efüzyon akciğer dış yüzünü ve göğüs duvarı iç yüzünü saran zarlar arasında kalan boşlukta çeşitli içeriklerde su birikmesine denir. Sitopatolojik değerlendirmede bu duruma çok rastlanmaktadır. Bu nedenle çekirdek algılama, plevral efüzyon teşhisi için sitopatolojik değerlendirmede önemli bir adımdır. Son yıllarda derin öğrenme algoritmaları nesne algılamada önemli başarılar elde etmiştir. Bu çalışmada modern konvolüsyonel nesne algılayıcı, Yolov3, plevral efüzyon sitopatolojik görüntüleri üzerinde çekirdek algılama amacıyla önerilmiştir. Mikroskoptan alınan sitopatolojik görüntülerin panorama yöntemiyle birleştirilmesiyle tanıma için kullanılacak veritabanı elde edilmiştir. Deneyler 11157 çekirdek içeren 80 görüntü üzerinde gerçekleştirilmiştir. Bu çalışmada önerilen yöntem %94,10 kesinlik, %98,98 duyarlılık ve %96,48 F-ölçütü elde etmiştir. Literatürdeki benzer durumlar için kullanılan diğer yöntemlerle karşılaştırıldığında bu yöntemin 10 kat daha hızlı olduğu sonucu ortaya çıkmıştır. Bu hızlanma dijital patolojideki gerçek zamanlı bilgisayar-destekli tanı (Computer-Aided Diagnosis-CAD) uygulamaları için önemli bir avantaj sağlamaktadır. Dolayısıyla önerilen yöntem dijital patolojide patologlar tarafından tanı aracı olarak kullanılabilecektir.

Anahtar Kelimeler: Sitopatoloji, Plevral Efüzyon, Bilgisayar-destekli tanı, Yolov3, Panorama, Çekirdek Algılama, Konvolüsyonel Sinir Ağları

Master Thesis

SUMMARY

AUTOMATIC NUCLEI DETECTION WITH YOLOV3 ALGORITHM ON PLEURAL EFFUSION CYTOPATOLOGY IMAGES PRODUCED BY PANORAMA METHOD

Büşranur KILIÇ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Assoc. Prof. Dr. Murat EKİNCİ
2020, 50 Pages, 3 Pages Appendix

Pleural effusion is the accumulation of water in various contents in the gap between the membranes surrounding the outer face of the lung and the inner wall of the chest. This case is commonly encountered in cytopathological evaluation. Therefore, the nuclei detection is an important step in the cytopathological evaluation for the diagnosis of pleural effusion. In recent years, deep learning algorithms have achieved significant success in object detection. In this study, the modern convolutional object detection algorithm, Yolov3, was proposed for the purpose of nuclei detection on cytopathological images of pleural effusion.

For this study by combining the cytopathological images taken from the microscope with panorama method, the database to be used for detection was obtained. Experiments were performed on 80 images containing 11157 nuclei. The proposed method in this study achieved 94.10% accuracy, 98.98% sensitivity and 96.48% F-measure. Compared to other methods used for the similar cases in the literature, this method was found to be 10 times faster. This acceleration provides a significant advantage for real-time computer-aided diagnostic (CAD) applications in digital pathology. Thus, the proposed method can be used as a diagnostic tool by pathologists in digital pathology.

Key Words: Cytopathology, Pleural effusion, Computer-aided diagnosis, Yolov3, Panorama, Nuclei detection, Convolutional Neural Networks

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

CAD	Bilgisayar-Destekli tanı (Computer-Aided Diagnosis)
FAST	Hızlandırılmış Segment Testinden Öznitelikler (Features From Accelerated Segment Test)
FPN	Özellik Piramit Ağı (Feature Pyramid Network)
NMS	Maksimum Olmayan Basturma Eşiği (Non-maximal Suppression)
RANSAC	Rastgele Örnek Onaylaşımı (Random Sample Consensus)
SVD	Tekil Değer Ayrışımı (Singular Value Decompositon)

1. GENEL BİLGİLER

1.1. Giriş

Plevra göğüs duvarı ile akciğer arasında yer alan seröz zarla kaplı bir boşluktur [1,2]. Plevra boşluğu göğüs duvarını içten örten visseral yüzey ve akciğeri dıştan örten paryetal yüzey arasında yer alır. Visseral yüzel plevral boşluğa az miktarda sıvı salgılanmasını sağlarken, paryatel yüzey de bu sıvının emilimini gerçekleştirmektedir. Plevra boşluğunda normalde çok az miktarda sıvı bulunmaktadır. Visseral yüzeyin plevral boşluğa çok miktarda sıvı salgılaması veya paryetal yüzeyin yeterli miktarda emilim yapmaması gibi bazı patolojik durumlarda plevral boşlukta fazla sıvı birikimi meydana gelmektedir. Bu sıvı birikimine plevral efüzyon denmektedir. Ülkemizde plevral efüzyonun en sık sebeplerinden birincisi tüberküloz iken diğerleri kanser, kalp yetmezliği ve enfeksiyonlardır [3,4].

Göğüs hastalıkları kliniklerinin çalışma yükünün önemli bir kısmını plevra hastalıkları oluşturmaktadır. Tanının yapılabilmesi için plevra zarları arasında biriken sıvıdan iğne yardımıyla örnek sıvı alınır. Daha sonra bu sıvı laboratuvarda uzman patologlar tarafından incelenir. El-göz koordinasyonu ile geçekleştirilen bu sitopatolojik değerlendirmede numune üzerindeki tüm hücrelerin morfolojik özelliklerinin ayrıntılı bir şekilde incelenmesi gerekmektedir. Bu inceleme oldukça yorucu ve zaman alıcı bir işlemdir. Ayrıca patologlar arası tanı farklılıklarına da yol açabilmektedir. Bu nedenle Bilgisayar-destekli tanı (Computer-Aided Diagnosis- CAD) sistemlerine ihtiyaç duyulmaktadır.

Bu tez çalışmasında derin öğrenme yöntemlerinden biri olan Yolov3 nesne algılama algoritması kullanılarak hücre içindeki en baskın yapı olan çekirdeğin algılanabilmesi amaçlanmıştır [5]. Öncelikle eğitim ve test için kullanılacak görüntülerin elde edilmesi gerekmektedir. Bu amaçla mikroskop kullanılarak lamelden görüntüler elde edilmiştir. Bu görüntüler panorama yöntemiyle birleştirilerek tek bir görüntü haline getirilmiştir.

Panoramik görüntünün elde edilmesi aşamasında öncelikle birleşecek görüntülerin köşe noktalarının ve bu noktaların özniteliklerinin bulunması gerekmektedir. Bu aşamada Hızlandırılmış Segment Testinden Öznitelikler (FAST) algoritması kullanılmıştır [6]. Çıkarılan

bu özniteliklerin benzerlik oranına bakılarak en yakın eşleştirmeler bulunmuştur. Eşleşen bu noktalardaki yanlış eşleştirmeleri minimize etmek ve dönüşüm (homografi) matrisini bulmak için Rastgele Örnek Onaylaşımı (RANSAC) algoritması kullanılmıştır [7]. Daha sonra Laplace piramiti kullanılarak görüntüler birleştirilmiştir [8].

Daha sonra da elde edilen panoramik görüntüden belirli boyutlarda kesitler alınarak Yolov3 algoritmasında kullanılacak olan eğitim ve test veri setleri oluşturulmuştur. Yolov3 algoritmasını diğer yöntemlerden ayıran en önemli fark gerçek zamanlı nesne algılamada elde ettiği başarıdır. Yolov3 özellik çıkarıcı olarak Darknet-53 ağını ve nesneleri tahmin etmek için de Özellik Piramit Ağı (Feature Pyramid Network, FPN)'na benzer bir yapı kullanmaktadır. Bu çalışmada Yolov3 ağı kullanılarak plevral efüzyon sitopatoloji görüntülerindeki çekirdeklerin hızlı ve doğru bir şekilde algılanması hedeflenmiştir.

1.2. Literatür Araştırması

Çekirdek algılama dijital patoloji uygulamalarında önemli ve zor bir işlemdir. Çekirdeklerin farklı morfolojik yapıda olması, gürültü, boya, yağ artefaktları içermesi, üst üste gelmeleri gibi problemler algılama yöntemlerindeki başarıyı düşürebilmektedir. Ayrıca mikroskobik görüntüleme sistemlerinde bir alan derinliği (depth of field) mevcuttur. Bu aralığın içerisine düşen hücreler net görünürken, bu aralığın arka veya ön bölgesine düşen hücreler bulanık görülmektedir [9]. Bu durum da yine başarıyı olumsuz etkilemektedir.

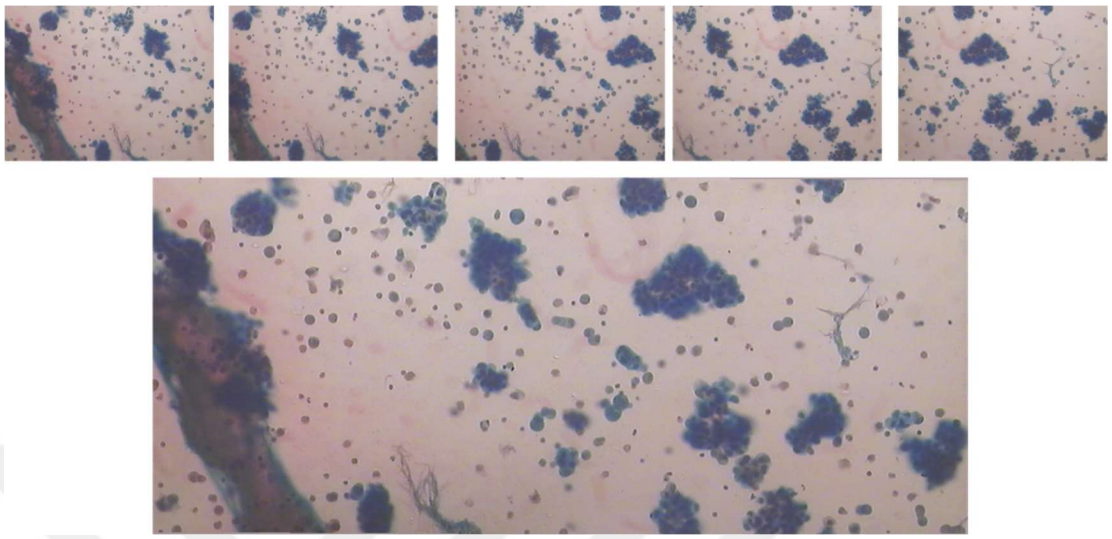
Literatürde, çekirdek algılama üzerine birçok çalışma mevcuttur. Bu çalışmaların birçoğu renk, kenar ve parlaklık bilgisine bağlı olarak elle çıkarılan düşük seviyeli özellikleri kullanmaktadır. Bu çalışmaların bir alandan başka bir alana adapte edilmesi zordur. Son zamanlarda derin öğrenme yöntemleri nesne tanıma, nesne algılama gibi çeşitli bilgisayarla görme problemlerinde yaygın şekilde kullanılmaktadır. Derin öğrenme yöntemleri yüksek seviyeli öznitelikleri otomatik öğrenebilme kabiliyetine sahiptir. Literatürde, plevral efüzyon sitopatolojisinde çekirdek algılama amacıyla önerilen birkaç çalışma mevcuttur. Viola-Jones nesne algılama yaklaşımı birçok nesne algılama problemine başarıyla uygulanmıştır. Bu yaklaşım hız ve doğruluk açısından yüksek sonuçlar vermiştir. Baykal vd. [10] bu yaklaşımı

plevral efüzyon sitoloji görüntülerinde çekirdek algılama amacıyla önermiş ve %89,32 doğruluk elde etmişlerdir. Daha sonra Yığınlanmış Seyrek Oto kodlayıcı (Stacked Sparse Autoencoders-SSAE) kullanarak başarıyı %98,30'a yükseltmişlerdir [11]. Bu yöntemlerin başarısı 178 çekirdek içeren 3 sitoloji görüntüsü üzerinde ölçülmüştür. Ancak bu yöntemler karmaşık arka plan, çekirdeklerin üst üste gelmesi ve çekirdeklerin kenarda olması gibi durumlarda düşük performans göstermişlerdir.

Literatürde mevcut olan bu küçük veri seti yeni ve daha kapsamlı bir veri seti oluşturma ihtiyacını doğurmuştur. Dolayısıyla Baykal vd. [12] çekirdek algılama üzerine yapılan çalışmaların sonuçlarının daha güvenilir olması amacıyla yeni bir plevral efüzyon sitoloji veri seti önermişlerdir. Ayrıca önerilen veri seti üzerinde üç adet konvolüsyonel nesne algılayıcıyı çekirdek algılama amacıyla önermişlerdir. Önerilen Faster R-CNN, R-FCN ve SSD yöntemleri ile elde edilen algılama sonuçları yüksek olmasına rağmen, gerçek zamanlı CAD uygulamalarında bu yöntemlerin yeterli hız sağlamadığı sonucuna varılmıştır. Önerilen yöntemin hızı CAD uygulamaları için önemlidir, çünkü patolojik numune örneği milyonlarca hücre içermektedir. Bu yüzden önerilecek yöntemin hem yüksek algılama başarısına sahip olması hem de hızlı olması gerekmektedir. YOLOv3'ü diğer yöntemlerden ayıran en önemli fark gerçek zamanlı nesne algılamada elde ettiği yüksek başarı olmuştur. Dolayısıyla bu çalışmada YOLOv3 nesne algılayıcısı Plevral Efüzyon sitopatolojisinde çekirdek algılama amacıyla önerilmiştir ve daha hızlı algılama hedeflenmiştir.

1.3. Panorama

Panorama, ortak bölgeleri olan iki veya daha fazla görüntünün birleştirilerek tek görüntü olarak elde edilmesi işlemine denmektedir. Şekil 1.1.'de buna örnek olarak 5 görüntünün birleştirilerek tek görüntü haline getirilmesi gösterilmiştir.



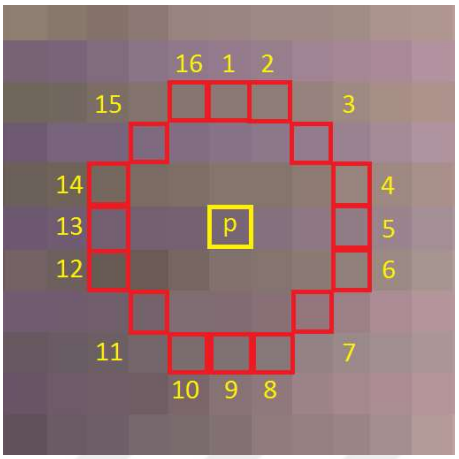
Şekil 1.1. Örnek panorama görüntüsü

Panorama görüntüsünün oluşturulmasında kullanılan işlem adımları;

- Görüntülerin köşe noktalarının tespiti yapılır.
- Bu köşe noktalarının öznitelik vektörleri oluşturulur.
- Öznitelik vektörlerine göre benzer köşe noktaları eşleştirilir.
- Ransac algoritması uygulanarak yanlış eşleştirmeler elimine edilir ve en iyi homografi matrisi bulunur.
- Görüntülerin Laplace Piramitleri oluşturulur.
- Homografi matrisi ve Laplace piramitleri kullanılarak görüntüler birleştirilir.

1.3.1. Köşe Noktalarının Tespit Edilmesi

Panoramik görüntünün oluşturulabilmesi için ilk adım köşe noktalarının bulunmadır. Köşe noktaları görüntü üzerinde hem X hem de Y eksenleri üzerinde yoğunluk olarak büyük değişimlerin olduğu koordinatlardır. Köşe noktalarının bulunması için Hızlandırılmış Segment Testinden Öznitelikler (Features From Accelerated Segment Test- FAST) algoritması kullanılmıştır [6].



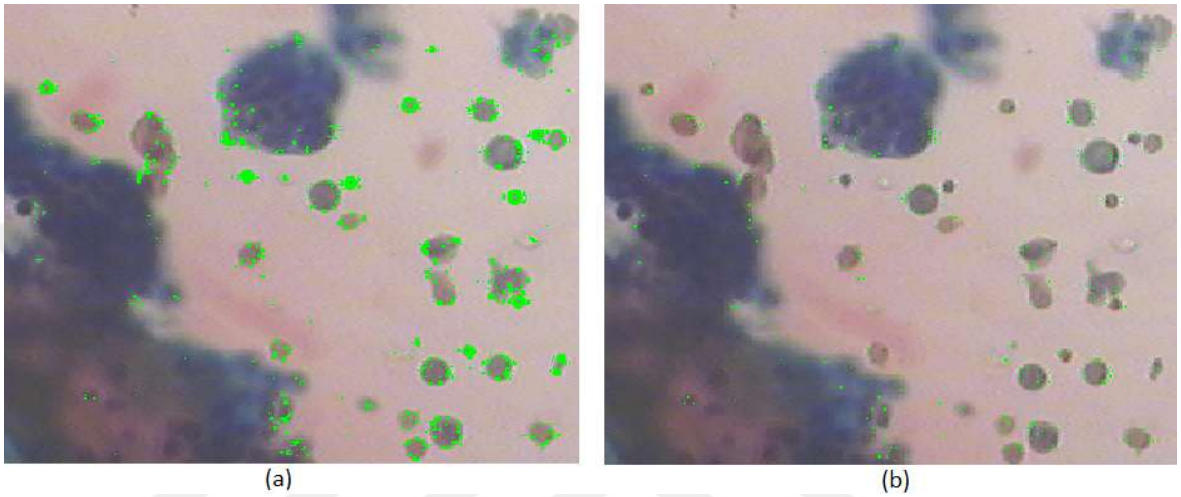
Şekil 1.2. Test edilen köşe noktası ve çevresindeki 16 piksel

FAST yönteminin algoritma adımları aşağıdaki şekildedir;

1. Öncelikle görüntüde bir p noktası seçilir. Bu nokta köşe noktası olarak tanımlanacak veya tanımlanmayacak olan noktadır. Bu p pikselin yoğunluğu I_p olarak kabul edilir.
2. Uygun eşik değeri (t) seçilir.
3. Şekil 1.2.'de görüldüğü gibi p pikselini çevreleyen 16 piksel içinde $I_p + t$ den daha parlak olan veya $I_p - t$ den daha koyu olan n tane bitişik piksel kümesi varsa p pikseli bir köşe noktası olmaktadır.
4. Algoritmanın hızlı olabilmesi için öncelikle dairedeki 1., 5., 9. ve 13.piksellerin yoğunluğu I_p ile karşılaştırılır. Şekil 1.2.'den anlaşılacağı gibi bu piksellerin en az üçünün eşik kriterini sağlaması gerekmektedir.
5. Bu dört piksel değerinden $(I_1,\ I_5,\ I_9,\ I_{13})$ en az üçü $I_p + t$ den daha büyük veya $I_p - t$ den daha küçük ise o zaman 16 pikselin tümü kontrol edilir. Ve n bitişik piksel kümesi için bu şart sağlanırsa p pikseli köşe noktası olmaktadır.
6. Bu işlemler görüntüdeki tüm pikseller için uygulanmaktadır.

Bu şekilde tüm köşe noktaları bulunmuş olur. Fakat yakın konumlarda birden fazla köşe noktası tespiti yapılmış olabilir. Bunu önlemek amacıyla Maksimum olmaya bastırma (Non-maximum suppression- NMS) eşiği uygulanır. Yani bulunan köşe noktalarının her biri için bir puan hesaplanır. Bu puan p pikseli ile çevresindeki 16 piksel değerinin arasındaki mutlak farkın

toplamı alınarak bulunur. Buna göre yakın olan köşe noktalarından düşük puana sahip olanlar elenir. Şekil 1.3'te bir görüntü üzerinde NMS uygulanmadan ve uygulandıktan sonraki bulunan köşe noktaları gösterilmiştir.



Şekil 1.3. a)NMS uygulanmadan önce b) NMS uygulandıktan sonra bulunan köşe noktaları

1.3.2. Öznitelik Vektörü Çıkarma

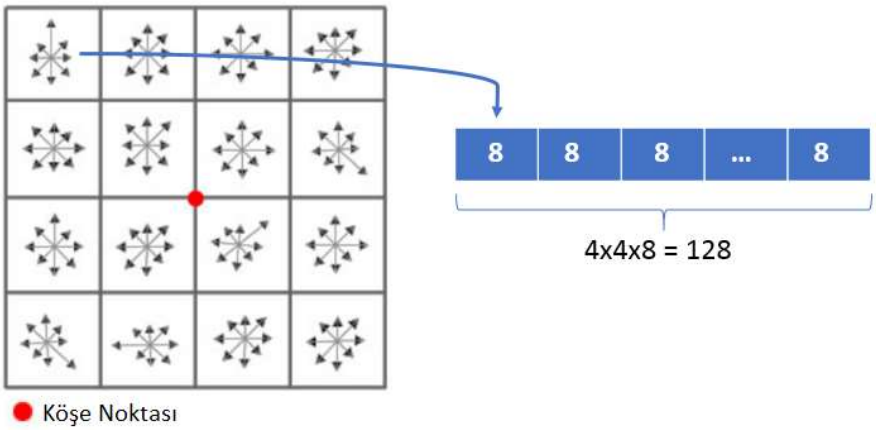
Öznitelik vektörünün oluşturulması esnasında öncelikle köşe noktasının etrafında 16x16'lık bir pencere alınır. Bu pencere içinde kalan köşe noktası etrafındaki tüm pikseller için (1.1) ve (1.2)'de belirtilen formüller kullanılarak gradyan büyüklükleri ve yönleri SIFT algoritmasına benzer şekilde hesaplanır [13].

$$m(x,y) = \sqrt{(L(x+1,y)+L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \quad (1.1)$$

$$Q(x,y) = tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \quad (1.2)$$

Daha sonra bu pencere 4x4'lük pencerelere bölünür. Bu pencerelerin her biri için 45 derecelik yönelimlere göre 8 kutudan oluşan histogramlar oluşturulur. Pikseller için bulunan

yönler bu histogramlara atanır. Böylece bir piksel için Şekil 1.4'te görüldüğü gibi 4x4x8'den 128 boyutunda öznitelik vektörü elde edilmiş olur.

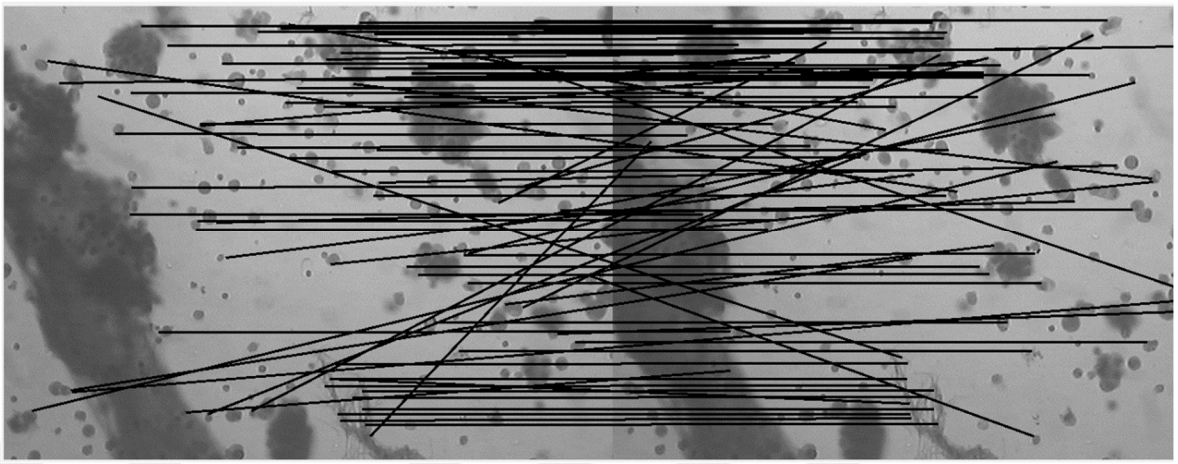


Şekil 1.4. 128 boyutlu öznitelik vektörü

1.3.3. Köşe Noktalarının Eşleştirilmesi

Köşe noktalarının eşleştirilme adımları;

- Öncelikle 1.görüntüden bir köşe noktası seçilir.
- Seçilen bu köşe noktasının öznitelik vektörü 2.görüntüdeki tüm köşe noktalarının öznitelik vektörleri ile karşılaştırılarak her biriyle aralarındaki fark alınır.
- 2.görüntüden bu farkın en az olduğu 2 köşe noktası seçilir, bunlar f1 ve f2 olsun.
- Eğer f1 < 0.95 * f2 koşulu sağlanırsa 1.görüntüden seçilen köşe noktası ile farkın en az olduğu köşe noktası (f1) arasında eşleştirme yapılır. Eğer bu koşul sağlanmazsa eşleştirme yapılmamaktadır.
- 1.görüntüdeki her bir köşe noktası için bu işlemler tekrar edilerek köşe noktaları arasında eşleştirme yapılmış olur.



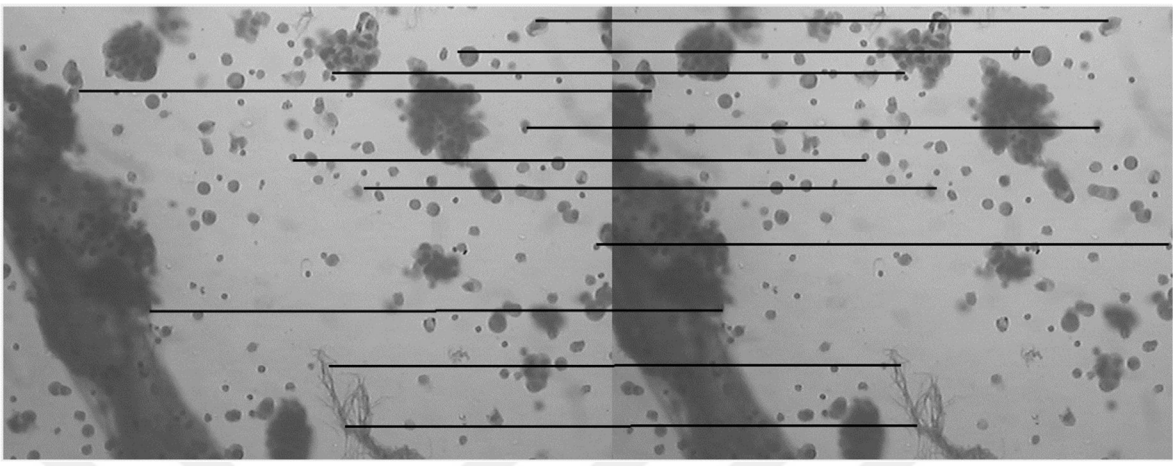
Şekil 1.5. İki görüntü arasındaki eşleştirilmiş köşe noktaları

1.3.4 RANSAC Algoritması ile Homografi Matrisinin Bulunması

RANSAC (Random Sample Concensus) algoritması [7] ile yanlış eşleşen köşe noktaları elimine edilerek en iyi homografi(dönüşüm) matrisinin bulunabilmesi amaçlanmıştır.

Ransac algoritmasının adımları şu şekildedir;

1. Rastgele 4 eşleşme noktası seçilir.
2. Seçilen bu noktaların aynı doğru üzerinde olup olmadığı kontrol edilir.
3. Aynı doğru üzerindeyse tekrar 1.adıma gidilerek rastgele 4 nokta seçilir.
4. Eğer aynı doğru üzerinde değilse bu dört nokta kullanılarak homografi matrisi hesaplanır.
5. Tüm eşleşme noktalarından bu homografi matrisine uyan kaç nokta olduğu hesaplanır.
6. En iyi homografi matrisi bulunana kadar bu işlemler tekrar edilir.



Şekil 1.6. Ransac sonrası eşleşen noktaların seyreltilmiş örnek gösterimi

Şekil 1.5.'te gösterilen eşleştirme noktaları Ransac algoritmasına verildiğinde Şekil 1.6.'daki sonuç elde edilmektedir. Şekil 1.6.'da görüldüğü gibi yanlış eşleştirmeler elenmiş ve en iyi homografi matrisi elde edilmiştir.

1.3.4.1 Homografi Matrisi

Homografi(dönüşüm) matrisi bir görüntü üzerindeki noktaların diğer görüntü üzerinde hangi noktalara karşılık geldiğini bulmak için kullanılmaktadır [14]. Homografi matrisi eşleşme yapılan köşe noktalarından dört tanesi kullanılarak aşağıdaki şekilde hesaplanır.

Image 1	Image 2
(x,y)	(wx'/w, wy'/w)=(x', y')

Şekil 1.7 Homografi matrisi için seçilen iki görüntü

Burada x, y birinci görüntünün köşe noktalarının koordinatlarını x', y' ise x ve y noktalarına denk gelen ikinci görüntüdeki köşe noktalarının koordinat değerleridir.

(Homografi Matrisi)

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1.3)$$

(1.3)'te verilen eşitlikten aşağıdaki denklemler elde edilir.

$$wx' = a_{11}.x + a_{12}.y + a_{13} \quad (1.4)$$

$$wy' = a_{21}.x + a_{22}.y + a_{23} \quad (1.5)$$

$$w = a_{31}.x + a_{32}.y + a_{33} \quad (1.6)$$

W'yi ilk iki denklemde yerine yazarak n nokta sayısı için aşağıdaki denklemler oluşturulmuş olur.

$$a_{11}x_1 + a_{12}y_1 + a_{13} - a_{31}x_1x_1' - a_{32}y_1x_1' - a_{33}x_1' = 0 \quad (1.7)$$

$$a_{21}x_1 + a_{22}y_1 + a_{23} - a_{31}x_1y_1' - a_{32}y_1y_1' - a_{33}y_1' = 0 \quad (1.8)$$

$$\vdots$$

$$a_{11}x_n + a_{12}y_n + a_{13} - a_{31}x_nx_n' - a_{32}y_nx_n' - a_{33}x_n = 0 \quad (1.9)$$

$$a_{21}x_n + a_{22}y_n + a_{23} - a_{31}x_ny_n' - a_{32}y_ny_n' - a_{33}y_n' = 0 \quad (1.10)$$

Bu denklemler de a_{33} yerine 1 yazarak denklemler yeniden düzenlenir.

$$a_{11}x_1 + a_{12}y_1 + a_{13} - a_{31}x_1x_1' - a_{32}y_1x_1' = x_1' \quad (1.11)$$

$$a_{21}x_1 + a_{22}y_1 + a_{23} - a_{31}x_1y_1' - a_{32}y_1y_1' = y_1' \quad (1.12)$$

$$\vdots$$

$$a_{11}x_n + a_{12}y_n + a_{13} - a_{31}x_nx_n' - a_{32}y_nx_n' = x_n' \quad (1.13)$$

$$a_{21}x_n + a_{22}y_n + a_{23} - a_{31}x_ny_n' - a_{32}y_ny_n' = y_n' \quad (1.14)$$

Bu denklemler eşitlik (1.15)'de gösterildiği gibi matrisle ifade edilmiştir.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -y_nx_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n' & -y_ny_n' \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ \vdots \\ x_n' \\ y_n' \end{bmatrix} \quad (1.15)$$

$$A \quad x = B$$

Burada amaç x = $(a_{11}, a_{12}, \dots, a_{32})$ homografi matrisini bulmaktır. Bunun için Tekil Değer Ayrışımı (SVD) yöntemi kullanılmıştır. Bu yöntem A matrisini UDV^T bileşenlerine ayırarak çözüm yapmaktadır. Burada U matrisi mxn boyutlarında ortogonal matris, D matrisi nxn boyutlarında köşegen bir matris ve V matrisi de nxn boyutlarında ortogonal bir matristir. V^T, V matrisinin trans pozunu ifade etmektedir. A matrisini tekil değerleri D matrisinin köşegen elemanları olmaktadır ve bu değerler AA^T matrisinin öz değerlerinin kareköklerine karşılık gelmektedir.

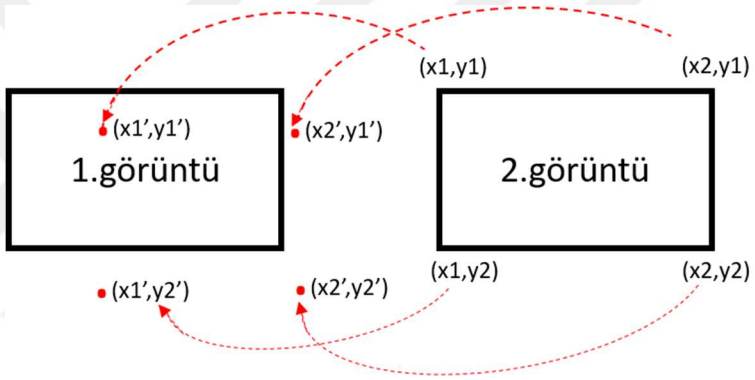
1.3.5. Panoramik Görüntünün Oluşturulması

Önceki adımlarda köşe noktaları eşleştirilip homografi matrisi bulunan iki görüntü için oluşturulacak panoromik görüntünün işlem adımları aşağıda belirtilen şekilde yapılmaktadır.

1. Öncelikle oluşturulacak olan panoromik görüntünün boyutu homografi matrisi yardımıyla hesaplanır.
2. Daha sonra 2 görüntü ayrı ayrı hesaplanan panoramik görüntünün boyutlarına genişletilir.
3. Bu görüntülerin Laplace piramitleri hesaplanır.
4. Bir maske belirlenir.
5. Bu maskeye göre görüntüler birleştirilir.

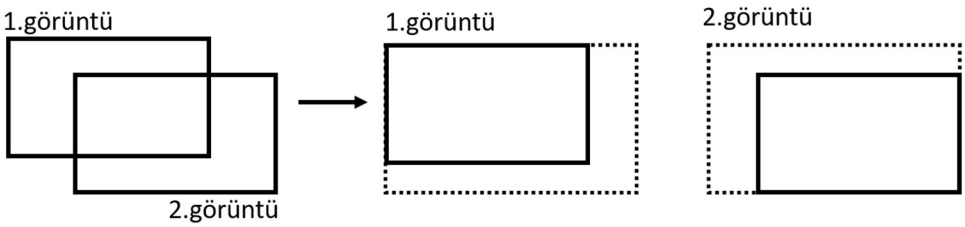
1.3.5.1. Görüntülerin Genişletilmesi

Panoramik görüntünün oluşturulmasındaki ilk adım seçilen 2 görüntünün boyutlarının oluşturulacak panoramanın boyutlarına genişletilmesi işlemidir. Şekil 1.8'e bakarsak burada 2.görüntüdeki $(x1, y1), (x2, y1), (x1, y2)$ ve $(x2, y2)$ köşe noktalarının homografi matrisiyle çarpılıp 1.görüntü üzerinde hangi noktalara karşılık geldiği gösterilmektedir. Bulunan $(x1', y1'), (x2', y1'), (x1'y2')$ ve $(x2', y2')$ noktalarından faydalanarak bu iki görüntünün birleşiminden oluşacak olan panoramik görüntünün boyutları hesaplanmış olmaktadır.



Şekil 1.8 2. görüntünün köşe noktalarının 1.görüntü üzerinde denk geldiği noktalara örnek

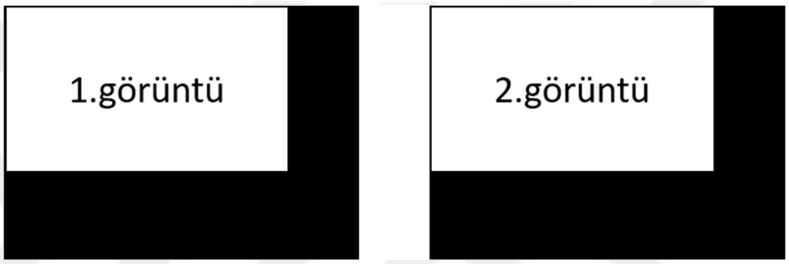
Daha sonra iki görüntüde ayrı ayrı birleşme yönlerine göre hesaplanan panoramik görüntünün boyutlarına genişletilir. Şekil 1.9'da gösterilen bu iki görüntüde 1.görüntünün sağ ve alt tarafındaki pikseller doldurularak 2.görüntünün ise sol ve üst tarafındaki pikseller doldurularak genişletme işlemi yapılır. Bu doldurma işleminde görüntülerin R, G, B kanallarının ortalama değerleri hesaplanarak bu değerlerle doldurulur.



Şekil 1.9. Görüntülerin birleşme yönlerine göre genişletilmesine örnek

1.3.5.2. Laplace Piramiti Oluşturma

Laplace piramiti görüntüler birleştiğinde geçiş bölgelerinin daha düzgün olması amacıyla kullanılmıştır [8]. Laplace piramitinin uygulanabilmesi için görüntülerin boyutlarının ($2^N + 1\ x\ 2^N + 1$) boyutlarında olması gerekmektedir. Bu nedenle görüntüler öncelikle bu boyutlara ayarlanır. Bu işlem Şekil 1.10'da gösterildiği gibi görüntülerin sağ ve alt kısımlarına 0 pikselleri eklenerek yapılmaktadır.



Şekil 1.10. Görüntülerin ($2^N + 1\ x\ 2^N + 1$) boyutlarına genişletilmesine örnek

Görüntülerin boyutları ayarlandıktan sonra Laplace Piramiti'ni oluşturabilmek için öncelikle Gauss piramitinin yapılması gerekmektedir. Gauss piramiti [15] orijinal görüntünün birkaç kopyasından oluşmaktadır. Her kopyada görüntünün kalitesi ve çözünürlüğü kademeli olarak düşürülmektedir. Piramitin 0.seviyesi (G_0) orijinal görüntü olmaktadır. 0.seviyedeki görüntü öncelikle bir alt filtreden geçirilip bulanıklaştırılır. Daha sonra alt örnekleme yapılarak çözünürlüğü düşürülür ve 1.seviye (G_1) elde edilmiş olur. Bu işleme REDUCE denmektedir ve isteğe göre birkaç adımdan oluşur. Gauss Piramiti (1.16) ve (1.17)'deki gibi ifade edilir. Eşitlik (1.16)'daki I orijinal görüntüyü temsil etmektedir.

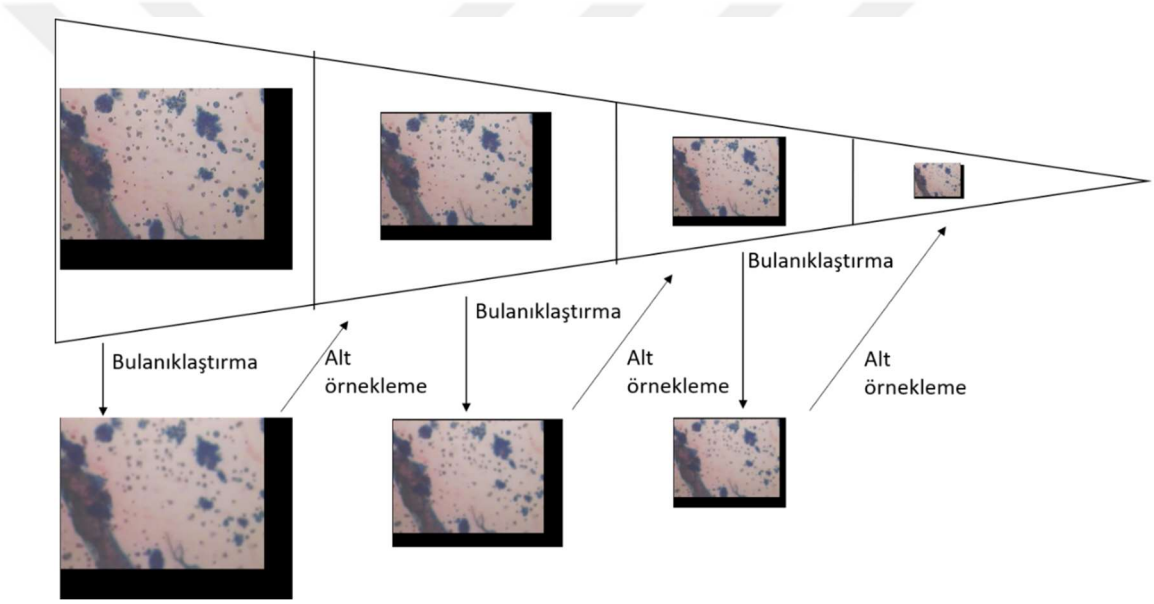
$$G_0(x, y) = I \tag{1.16}$$

$$G_{i+1}(x, y) = REDUCE(G_i(x, y)) \tag{1.17}$$

REDUCE işleminde kullanılan alçak geçirgen filtre eşitlik (1.18)'de belirtildiği gibi seçilmiştir. Filtrenin merkez pikseli komşu piksellerden daha fazla ağırlık almaktadır.

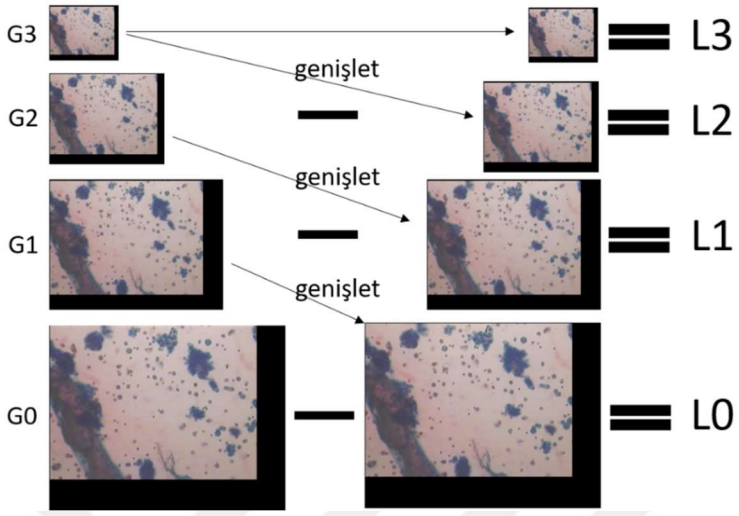
$$F(r,c) = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \tag{1.18}$$

Bu şekilde Şekil 1.11'de bir görüntü için gösterilen Gauss piramiti iki görüntü için de ayrı ayrı uygulanır.



Şekil 1.11. Bir görüntü için oluşturulan Gauss piramitine örnek

Oluşturulan bu Gauss piramitleri kullanılarak Laplace piramitleri elde edilir. Laplace piramidinin oluşturulması için gauss piramidinin alt seviyesindeki bir görüntü bir üst seviyedeki görüntünün boyutlarına genişletilerek aralarındaki fark alınır. Bu işlem tüm katmanlar için uygulanır. Laplace piramitinin son katmanı gauss piramidinin son katmanına eşit olmaktadır. Buna örnek olarak bir görüntü için oluşturulan Laplace piramiti Şekil 1.11'de gösterilmiştir.



Şekil 1.12. Bir görüntü için oluşturulan Laplace piramitine örnek

Boyut büyültme işlemi eşitlik (1.19)'da belirtildiği gibi yapılmaktadır. Bu denklemdeki $F(r,c)$ eşitlik (1.18)'de verilen 5x5'lik filtreyi belirtmektedir.

$$G_{i+1}(x,y) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} F(m,n)\, G_i\left(\frac{x-m}{2}, \frac{y-n}{2}\right) \tag{1.19}$$

Bu işlemler yapılarak iki görüntü içinde Laplace piramitleri oluşturulmuş olmaktadır.

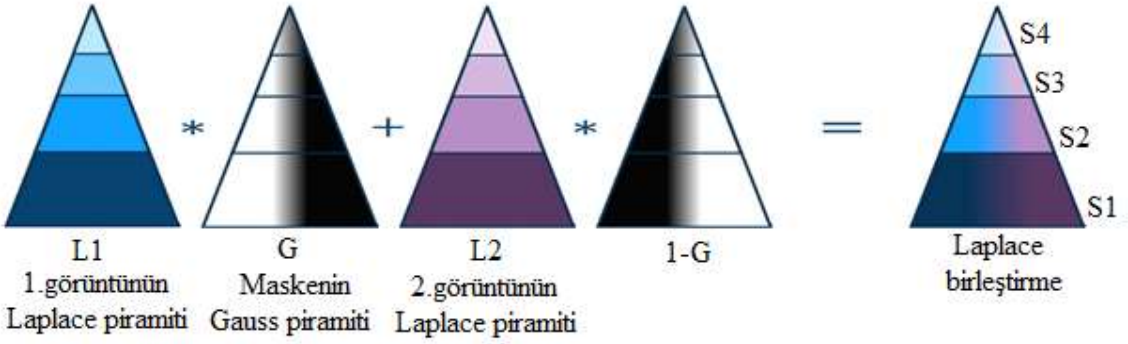
1.3.5.3. Maske Seçimi ve Görüntülerin Birleştirilmesi

Burada binary olarak oluşturulacak maske 1.ve 2.görüntülerden hangi bölgelerin alınacağı belirlenmektedir. Maske 0 ve 1 değerlerinden oluşmaktadır. Maskede 1 olan bölgeler 1.görüntüden bölge alınacağını, 0 olan bölgeler ise 2.görüntüden bölge alınacağını belirtir. Bu maskenin boyutu oluşturulacak panoramik görüntünün boyutlarıyla aynı olacak şekilde ayarlanır ve sonra $(2^N + 1 x 2^N + 1)$ boyutlarına getirilir. Bu genişletme işlemi görüntüler için yapılan işlemle aynı olarak maskenin sağ ve alt kısımları sıfır pikselleriyle doldurularak yapılmaktadır (Şekil 1.9). Daha sonra görüntülerde olduğu gibi maskenin de Gauss piramitleri oluşturulur (Şekil 1.10).

Oluşturulan maske kullanılarak eşitlik (1.20)'de belirtildiği gibi Laplace piramitleri ile her katman ayrı ayrı birleştirilir.

$$Birleştirme(i,j) = G(i,j).L1(i,j) + (1 - G(i,j).L2(i,j)) \quad (1.20)$$

Burada 1.görüntü için oluşturulan Laplace piramitlerinin her bir katmanı belirlenen maskenin Gauss piramitlerinin aynı katmanlarıyla çarpılır. 2.görüntü için oluşturulan Laplace piramiti ise maskenin tersi alınarak çarpılır. Ve bu çarpım değerleri toplanarak Şekil 1.13'te gösterildiği gibi her katman için ayrı ayrı Laplace birleştirme yapılmış olur.

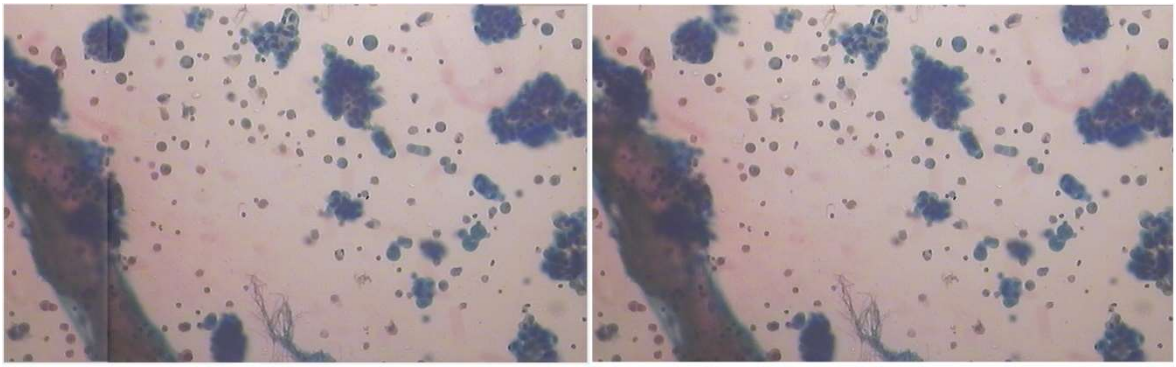


Şekil 1.13. Maskenin kullanılarak panoramik görüntünün oluşturulması [12]

Üretilen bu Laplace birleştirme piramiti ile eşitlik (1.21)'de belirtildiği gibi en küçük katmandan başlanarak genişletme yapılır ve bir üst katmanla toplanır. Böylece panorama görüntüsü oluşturulmuş olur.

$$panorama = S1 + Genislet(S2 + Genislet(S3 + Genislet(S4))) \quad (1.21)$$

Son olarak bu panorama görüntüsünden $(2^N + 1x2^N + 1)$ boyutlarına getirmek için eklediğimiz pikseller kırpılarak sonuç görüntüsü oluşturulmuş olur. Şekil 1.14'te Laplace piramiti kullanılmadan önce ve kullanıldıktan sonra birleştirilen görüntülere örnek verilmiştir.



Şekil 1.14. a) Laplace piramiti kullanılmadan yapılan b)Laplace piramiti kullanılarak yapılan birleştirmeye örnek

1.4. YoloV3 (You Only Look Once- Version3)

YoloV3(You Only Look Once-Version3) plevral efüzyon sitopatolojik görüntülerdeki çekirdeklerin algılanması amacıyla kullanılmıştır. YoloV3, Redmon(2018) ve arkadaşları tarafından geliştirilmiş gerçek zamanlı bir nesne algılayıcısıdır [5]. YoloV3 ağı hem hızlı hem de doğru sonuç üretmekte oldukça iyidir. Geleneksel bilgisayarla görme algoritmalarında farklı konum ve ölçeklerdeki nesneleri bulmak amacıyla kayan pencere yaklaşımı kullanılmaktadır. Fakat bu pahalı bir işlemdir ve nesnelerin en-boy oranının sabit olduğu düşünülmektedir. YoloV3 algoritmasında ise bu yaklaşımdan farklı olarak tüm görüntü ağ üzerinden yalnızca bir kez iletilerek nesne tespiti yapılmaktadır. Bu nedenle diğer yöntemlere göre oldukça hızlı çalışmaktadır.

Şekil 1.15'te gösterildiği gibi Yolov3 ağına bir görüntü verildiğinde çıkışta bu görüntüde bulunan nesnelerin hangi sınıfa ait olduğunu ve bu nesnelerin görüntüdeki konumlarını bize veren bir nesne algılama algoritmasıdır.

Şekil 1.15. YoloV3 ağının giriş çıkışı

1.4.1. YoloV3 Ağının Girişi

Yolov3 ağı giriş olarak bir görüntü alır. Bu görüntünün boyutları 32 sayısının katı olacak şekilde belirlenir. Bu görüntülerin derinliği ise R, G, B kanallarından dolayı 3 olmaktadır. Örneğin boyut olarak 320x320x3, 416x416x3, 608x608x3 gibi boyutlar verilebilmektedir.

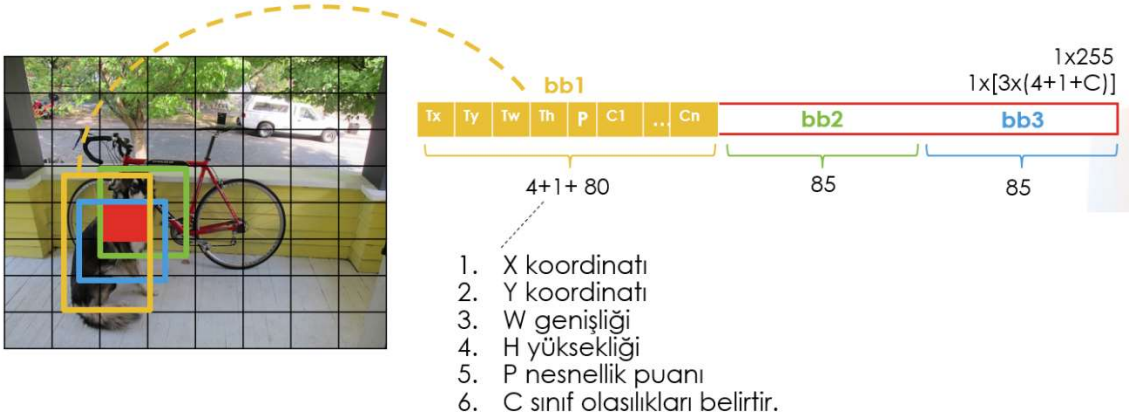
1.4.2. YoloV3 Ağının Çıkışı

Yolov3 ağının çıkışında Şekil 1.16'te gösterildiği gibi 3 farklı ölçekte çıkış alınır. Bunların ilki 13x13x255 olan 1.ölçek büyük boyutlu nesneleri algılamak için kullanılır. 26x26x255 olan 2.ölçek orta boyutlu nesneleri algılamak için ve 52x52x255 olan 3.ölçek ise küçük boyutlu nesneleri algılamak için kullanılmaktadır. Daha sonra bu 3 ölçek belirli işlemlerden geçirilerek çıkış elde edilmektedir.

Şekil 1.16. YoloV3 Ağının Çıkışı

Şekil 1.16'da gösterilen 13x13x255 çıkışını inceleyecek olursak;

Burada 13x13x255 giriş görüntüsünün 13x13'lük hücrelere bölündüğü anlamına gelmektedir. Ve bu hücrelerin her biri için 3 tane sınır kutusu tahmin edilmektedir. Şekil 1.17'de gösterildiği gibi kırmızı ile belirtilen hücre için sarı, yeşil ve mavi dikdörtgen ile belirtildiği gibi 3 tane sınır kutu tahmini yapılır. Ağın çıkışında bize bu sınır kutularının değerleri verilmektedir ve her bir sınır kutusu için çıkışta 85 değer elde edilir. Bu değerlerin ilk dördü sınır kutusunun (x, y, w, h) bilgilerini belirtmektedir. Burada (x, y) değerleri sınır kutusunun merkezi koordinatları, (w, h) değerleri ise sınır kutusunun genişlik ve yükseklik değerlerini vermektedir. Yine bu sınır kutusu için 1 tane nesnellik puanı (P) değeri tutulur. Ve 80 tane de sınıf olasılık değerleri (C) tutulmaktadır. Burada C sınıf sayısına göre değişiklik göstermektedir. Orijinal YoloV3' COCO veri seti kullanılmıştır. Bu veri setinde 80 tane sınıf olduğu için 80 tane sınıf olasılığı tahmin edilmektedir. Aynı şekilde 2.ve 3.sınır kutuları için de 85'er değer tutulur. Böylece (85 x 3)'ten her bir hücrenin derinliği 255 olmaktadır.

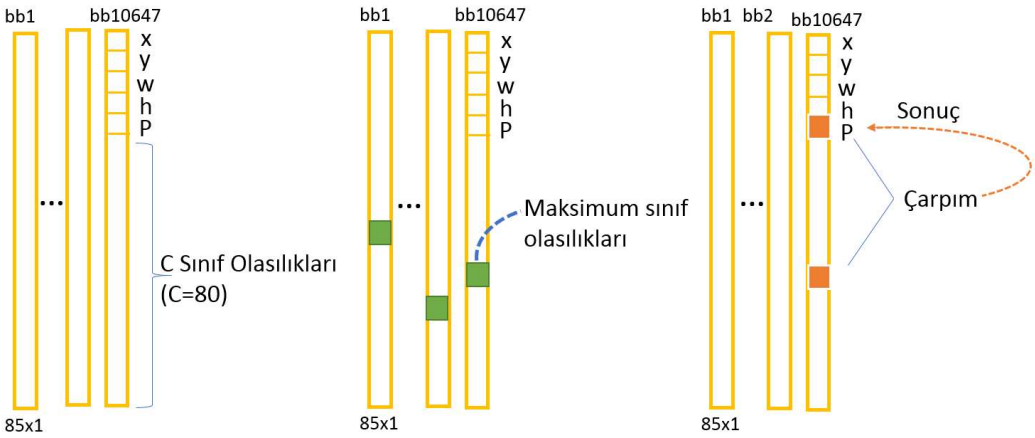


Şekil 1.17 YoloV3 ağının çıkışındaki her bir hücrenin derinliği [13]

1.4.2.1. YoloV3 Ağının Çıkışının Kullanılması

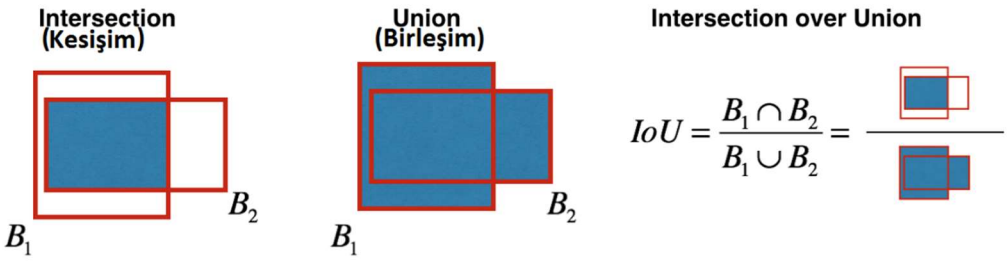
YoloV3 ağında 3 farklı ölçekte çıkış alınmaktadır. Bu 3 ölçekteki her bir hücre için 3 tane sınır kutusu tahmini yapılıyordu. Bu nedenle ağın çıkışında $3x(13^2 + 26^2 + 52^2)$'den toplam 10647 tane sınır kutusu tahmin edilmiş olmaktadır. Burada 13 değeri 1.ölçekten, 26 değeri 2.ölçekten, 52 değeri ise 3.ölçekten gelmektedir. Elde edilen bu sınır kutuları belirli işlemlerden geçirilerek algılama sonucu elde edilmektedir. Burada bu işlemlerin ne olduğu incelenecektir.

Öncelikle Şekil 1.18'de gösterildiği gibi 80 tane C sınıf olasılık değerlerinden maksimum olanlar her bir sınır kutusu için ayrı ayrı belirlenir. Bulunan bu maksimum değerlerle sınır kutularının nesnellik puanı çarpılır ve sonuç nesnellik puanının yerine yazılır. Bulunan bu yeni nesnellik puanına göre sınır kutuları belli bir eşik değerinden geçirilir. Nesnellik puanı bu eşik değerinden küçük olan sınır kutuları elenir. Elenen bu sınır kutuları nesnellik puanına göre büyükten küçüğe doğru sıralanır. Sıralanan bu sınır kutularına NMS (Non-Maximal Suppression) uygulanır.



Şekil 1.18. YoloV3 çıkış işlem adımları [17]

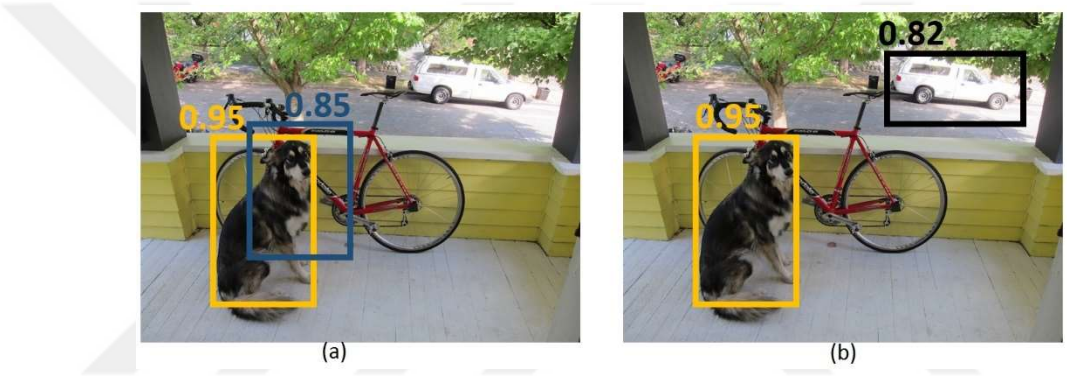
NMS algoritması bir nesneyi birden fazla sınır kutusunun algılamasını önlemek amacıyla uygulanır. Örneğin N tane sınır kutusu için nesnellik puanları (0.95, 0.85, 0.82, 0.75, …) şeklinde olsun. Burada öncelikle ilk nesnellik puanı seçilerek diğerleri arasında karşılaştırma yapılır. Şekil 1.20'de a görüntüsünde seçilen ilk iki değer için sınır kutuları gösterilmiştir. Bu değerler arasında Şekil 1.19'de gösterildiği gibi IoU(Intersection Over Union) değeri hesaplanır. Bu değer sınır kutularının konumsal kesişimlerinin konumsal birleşimlerine bölünmesiyle hesaplanmaktadır. Eğer bu değer belirlenen bir eşik değerinden (0.5) büyükse nesnellik puanı düşük olan sınır kutusu elenmektedir.



Şekil 1.19. IOU değerinin hesaplanması

Örneğin Şekil 1.20'deki a görüntüsündeki iki sınır kutusunun IoU değeri eşik değerinden büyük çıkarsa 0,85 değerini ifade eden sınır kutusu elenir. Bu durumda bir nesne için bir sınır kutusu atanmış olur. Böylece elimizde (0.95, 0.82, 0.75, …) değerleri kalmaktadır. Bir sonraki

adımda 0.95 ile 0.82 nesnellik puanlarının belirttiği sınır kutuları arasında IoU değeri hesaplanır. Bu sınır kutuları Şekil 1.20'deki b görüntüsünde gösterilmiştir. Bunlar arasındaki IoU değeri eşik değerinden küçük çıkarsa eleme yapılmaz. Bu şekilde ilk sıradaki nesnellik puanı olan 0.95'in belirttiği sınır kutusu ile diğer nesnellik puanlarının belirttiği tüm sınır kutuları karşılaştırılır. Bir sonraki adımda ikinci sıradaki nesnellik puanına geçilir ve ondan sonra gelen nesnellik puanları arasında karşılaştırma yapılır. Bu işlem dizinin sonuna gelene kadar tekrar edilir. Böylece kalan sınır kutuları ile çıkış elde edilmiş olur.



Şekil 1.20. YoloV3 Nesnellik Puanlarının ifade ettiği sınır kutularına örnek

1.4.3. YoloV3 Eğitim Aşaması

YoloV3 ağını eğitmek için aşağıda belirtilen dosyaların hazırlanması gerekmektedir.

- Train.txt
- Validation.txt
- Obj.names
- Etiket Dosyaları
- Obj.data
- Yolo-obj.cfg

1.4.3.1. Txt Uzantılı Dosyaların Hazırlanması

Train.txt ve validation.txt dosyalarında sırasıyla eğitim ve onaylama veri setlerinin yolları belirtilecek şekilde hazırlanmalıdır. Her bir bilgi yeni bir satıra yazılacak şekilde verilir. Tablo 1.1.'de bu dosyalara örnek gösterilmiştir.

Tablo 1.1 Eğitim ve onaylama verisetlerinin yollarını tutan dosyaların içeriği

Train.txt	Validation.txt
Data/obj/1.jpg Data/obj/2.jpg Data/obj/3.jpg Data/obj/4.jpg Data/obj/5.jpg	Data/obj/v1.jpg Data/obj/v2.jpg Data/obj/v3.jpg Data/obj/v4.jpg Data/obj/v5.jpg

1.4.3.2. Names Uzantılı Dosyanın Hazırlanması

Obj.names dosyasında sınıfların isimleri belirtilir. Tablo 1.2.'de gösterildiği gibi her bir sınıf ismi yeni bir satıra yazılacak şekilde hazırlanır.

Tablo 1.2 Sınıf isimlerini tutan dosyanın içeriği

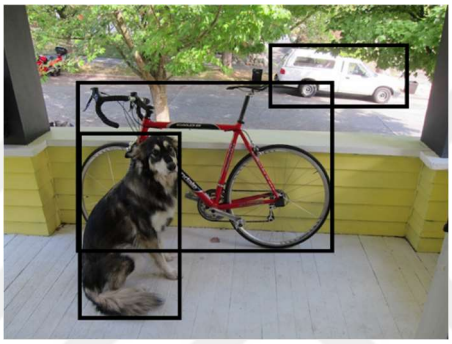
Obj.names
Araba Bisiklet Köpek

1.4.3.3. Etiket Dosyanın Hazırlanması

Öncelikle eğitim veri setinde bulunan görüntülerin Şekil 1.21'da gösterilen şekilde etiketlenmelidir. Görüntülerle aynı isimlerde txt dosyaları oluşturularak etiketlenen veriler bu dosyalara aşağıdaki biçimde yazılır.

<object_class> <x> <y> <width> <height>

Burada object_class etiketleme yapılan nesnenin hangi sınıfa ait olduğunu belirtmektedir ve [0, Sınıf Sayısı-1] arasında değer alır. Sonraki (x, y) değerleri etiketleme yapılan kutunun merkez konumlarını, (width, height) ise bu kutunun genişlik ve yükseklik değerlerini belirtmektedir. Ayrıca (x, y, width, height) değerleri görüntünün boyutlarına bölünerek normalize edilir yani (0,1] arasına düşürülür. (x/img_width, y/img_height, width/img_width, height/img_height)



Şekil 1.21 Görüntüdeki nesnelerin etiketlenmesine örnek

Tablo 1.3'te Şekil 1.19 için yapılan etiketlemenin örnek dosyası gösterilmiştir. Burada görüntünün ismi 1.jpg olduğu varsayılırsa 1.txt adında bir dosya oluşturulur. Bu dosyanın içine görüntüdeki etiketlenen nesnelerin her biri yeni bir satıra yazılacak şekilde kaydedilir. Bu nesnelerin ilk değerlerinde ki sınıf etiketleri obj.names dosyası ile ilişkilendirilir yani 0 araba sınıfını, 1 bisiklet sınıfını ve 2 de köpek sınıfını temsil etmektedir.

Tablo 1.3 Etiket dosyalarının içeriği

1.txt
0 0.8866667 0.5116667 0.165 0.063 1 0.7265668 0.7123362 0.357 0.244 2 0.6141544 0.9245688 0.051 0.269

1.4.3.4. Data Uzantılı Dosyanın Hazırlanması

Bu data uzantılı dosyada sınıf sayısı, eğitim ve onaylama veri setlerinin tutulduğu dosyanın yolu, sınıf isimlerinin tutulduğu dosyanın yolu ve sonuç ağırlıklarının nereye kaydedileceğinin yolu belirtilir ve Tablo 1.4'te gösterildiği gibi hazırlanır.

Tablo 1.4 Data uzantılı dosyanın içeriği

Obj.data
classes= 3 train = data/train.txt valid = data/test.txt names = data/obj.names backup = backup/

1.4.3.5. Cfg Uzantılı Dosyanın Hazırlanması

Bu dosya ağ mimarisinin yapısını belirtmektedir. Yolov3 bu dosyadan okuma yaparak ağı kurmaktadır. Bu dosya [net], [convolution], [shortcut], [upsample], [route], [yolo] gibi katmanların bilgileri yer almaktadır.

Tablo 1.5'te gösterildiği gibi bu dosya [net] ya da [network] yazılarak başlatılır ve sonrasında ağın yapısıyla ilgili bilgiler verilir. Eğitim esnasında batch ile belirtilen sayıda veriseti parçalara bölünür ve her iterasyonda modelin eğitimi bu parça üzerinden yapılmaktadır. Test aşamasında batch 1 olarak alınmaktadır. Subdivision değeri kullanılarak mini_batch değeri hesaplanır (mini_batch = batch / subdivision). Eğer veri boyutu GPU'ya yüklemek için çok büyükse belirtilen sayı kadar bölünerek Gpu'ya yükleme yapılır. Yani mini_batch sayısı kadar örnek aynı anda işlenmiş olmaktadır. Width, height ve channels değerleri ile ağın giriş boyutu belirtilir. Width, height değerleri 32 sayısının katı olmalıdır. Channels değeri ise R, G, B kanallarından dolayı 3 olarak alınmaktadır. Eğitim ve test sırasında her görüntü burada belirtilen boyutlara göre yeniden boyutlandırılır.

Eğer ağın girişine görüntüden farklı olarak özel bir veri girilmek istenirse $input = ağ\ boyutu$ şeklinde yazılarak belirtilmektedir. Momentum ağırlık değişiminin önceki

değerlerden ne kadar etkileneceğini belirler. Decay ise ağırlık güncellemesinin zayıflatılmasında kullanılır ve ağın dengesizliğini düzenler. Eğitim sırasında angle görüntüleri döndürmek için kullanılır. Saturation görüntülerin doygunluğunu, exposure parlaklığını hue değeri ise renk tonunu değiştirmek için kullanılır.

Tablo 1.5 Cfg dosyasındaki ağ bilgilerinin içeriği

Yolo-obj.cfg
[net] batch=64 subdivisions=32 width=608 height=608 channels=3 momentum=0.9 decay=0.0005 angle=0 saturation = 1.5 exposure = 1.5 hue=.1 learning_rate=0.001 burn_in=1000 max_batches = 10000 policy=steps steps=8000,9000 scales=.1,.1

Burn_in ile belirtilen sayıdaki ilk iterasyonlar için learning_rate eşitlik 1.22'de belirtildiği gibi hesaplanır. Burada power değeri varsayılan 4 olarak alınmaktadır.

$$currentLearningRate = learningRate * \frac{iterations^{power}}{burn\_in} \tag{1.22}$$

Toplam iterasyon sayısı max_batch ile belirtilir. Policy, (steps, sgdr, step, sig, exp, poly, random) gibi değerler alabilmektedir. Policy learning_rate'in nasıl kullanılacağını belirtir. Örneğin policy = steps olduğunda steps=8000, 9000 ile belirtilen iterasyonlarda learning_rate'in

scale=.1,.1 ile belirtilen sayılarla çarpılarak güncelleneceğini belirtmektedir. Ağın bilgileri bu şekilde verildikten sonra her katmanın bilgileri tek tek yazılır.

Tablo 1.6'da konvolüsyon katmanının nasıl belirtileceği gösterilmiştir.

Tablo 1.6 Cfg dosyasındaki konvolüsyon katman bilgilerinin içeriği

Yolo-obj.cfg
[convolutional] batch_normalize=1 filters=64 size=3 stride=2 pad=1 activation=leaky

Öncelikle [convolution] yazılarak başlanır. Devamında bu katmanda kullanılacak verilerin bilgileri verilmektedir. Batch_normalize'nin 1 olması bu katmanda normalize yapılacağını, 0 olması yapılmayacağını belirtir. Filters uygulanacak filtre sayısını, size filtrenin boyutunu, stride ise ne kadar stride yapılacağını belirtir. Pad değeri 1 ise $padding = size/2$ olarak, 0 ise $padding = 0$ olarak alınmaktadır. Pad yazmak yerine padding = istenilen padding değeri şeklinde de yazılabilmektedir. Activation kullanılacak aktivasyon fonksiyonunu belirtmektedir ve (leaky, logistic, loggy, relu, elu, selu, relie, plse, hardtan, lhtan, linear, ramp, tanh, stair) gibi değerler alabilmektedir.

Res katmanını belirtmek için Tablo 1.7'de gösterildiği gibi [shortcut] yazılır. Burada from bir önceki katman ile hangi katmanın toplanacağını belirtmektedir. -3 olması 3 önceki katmanın çıkışıyla 1 önceki katmanın çıkışının toplanacağı anlamına gelmektedir.

Tablo 1.7 Cfg dosyasındaki res katmanı bilgilerinin içeriği

Yolo-obj.cfg
[shortcut] from=-3 activation=linear

Upsample katmanında katman girişinin genişlik ve yükseklik değerleri stride ile çarpılarak genişletilir. Örneğin Tablo 1.8'de belirtilen Upsample katmanının giriş değerleri 300x300 ise bu değerler 2(stride) ile çarpılarak 600x600 boyutlarında veri elde edilir.

Tablo 1.8 Cfg dosyasındaki upsample katman bilgilerinin içeriği

Yolo-obj.cfg
[upsample] stride=2

Contacatene katmanını belirtmek Tablo 1.9'da gösterildiği gibi [route] kullanılır. Bu katman tek ya da iki değer alabilmektedir. Tek olduğunda indekslenen katmanın özellik haritasını döndürür. Çift değer alırsa indekslenen katmanların birleştirilmiş özellik haritalarını döndürür.

Tablo 1.9 Cfg dosyasındaki concatenate(birleştirme) katman bilgilerinin içeriği

Yolo-obj.cfg	
[route] layers = -1, 61	[route] layers = -4

Son olarak yolo katmanı [yolo] ile belirtilmektedir. Tablo 1.10'da belirtilen anchors(çapa) değerleri eğitim sırasında nesnelerin başlangıç boyutlarını belirtmektedir. Her bir ölçekte 3 tane sınır kutusu tahmini için toplamda 9 tane çapa kutusu belirlenir. Tablo 1.10'da belirtilen yolo katmanı 3.ölçek için yazılmıştır ve bu 3.ölçek için buradaki mask ile belirtilen indislerdeki çapa kutuları kullanılmaktadır. Yani anchor(çapa) değerlerinden 0,1 ve 2.indislerdeki (10,13, 16,30, 33,23) değerleri kullanılmaktadır. Classes ile sınıf sayısı, num ile çapa kutu sayısı belirtilir. Jitter ile en boy oranı $1 - 2 * jitter$ ve $1 + 2 * jitter$ arasında rastgele değişen görüntüleri rastgele keser ve yeniden boyutlandırılır. Ignore_thresh ve truth_thresh değerleri kayıp fonksiyonunda kullanılan eşik değerleridir. Random 1 olduğunda ise ağ rastgele yeniden boyutlandırılır.

Tablo 1.10 Cfg dosyasındaki yolo katman bilgilerinin içeriği

Yolo-obj.cfg
[yolo] mask = 0, 1, 2 anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373, 326 classes=3 num=9 jitter=.3 ignore_thresh = .7 truth_thresh = 1 random=1

1.4.4. Sınır Kutu Tahmini

Sınır kutularını doğrudan tahmin etmek yerine önceden belirlenen çapa kutuları kullanılır. Çapa kutuları ağın başlangıç değerlerini belirtmektedir. En iyi çapa kutularının değerinin bulunabilmesi için K-Ortalama algoritması (K-Means Algorithm) kullanılır[18]. Diğer bir değişle, eğitim seti hazırlığında her bir imgede el işçiliği ile seçilen tüm sınır kutularının(bounding box) boyutlarının genişlik -yükseklik uzayında oluşan dağılımın en baskın 3 küme merkezi K-Means Algoritmasıyla bulunur. Bulunan bu merkez noktalarının ifade ettiği sınır kutusu verileri(genişlik, yükseklik) çapa kutusunun genişlik, yükseklik değerleri olarak atanır. Aşağıdaki eşitlikler kullanılarak sınır kutuları tahmin edilir.

$$b_x = \sigma(t_x) + c_x \tag{1.23}$$

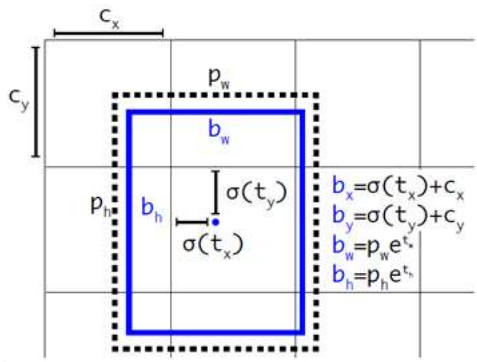
$$b_y = \sigma(t_y) + c_y \tag{1.24}$$

$$b_w = p_w e^{t_w} \tag{1.25}$$

$$b_h = p_h e^{t_h} \tag{1.26}$$

Burada t_x, t_y, t_w, t_h ağın tahmin ettiği değerlerdir. σ, sigmoid fonksiyonunu temsil eder ve t_x, t_y değerlerini 0-1 aralığına düşürür. p_w, p_h belirlenen çapa kutularının boyutlarını vermektedir. c_x, c_y sınır kutusunun konumunu belirlemek için kullanılır. Bu değerlerden

faydalanılarak b_x, b_y sınır kutusunun konumunu ve b_w, b_h sınır kutusunun boyutlarını hesaplanır. Şekil1.22.'de detaylı bir şekilde gösterilmiştir.



Şekil 1.22. YoloV3 çapa(anchor) kutuları ile sınır kutu tahmini

1.4.5. Nesnellik Puanı

Nesnellik puanı, bir nesnenin sınırlayıcı kutu içinde bulunma olasılığını belirtmektedir. YoloV3 nesnellik puanını tahmin etmek için her bir sınır kutusu için lojistik regresyon kullanır. Sınır kutuları karşılaştırıldığında hangi sınır kutusu kesin referans değer (ground truth) nesnesini daha fazla kaplıyorsa o sınır kutusunun nesnellik puanı 1 olmaktadır.

Bir sınır kutusunun ground truth nesnesi ile kesişimi belli bir eşik değerinden fazla olmasına rağmen o ground truth nesnesini başka bir sınır kutusu daha fazla kaplanırsa bu sınır kutusu ihmal edilir. Böylece bir nesne için yalnızca bir sınır kutusu atanmış olur.

1.4.6. Sınıf Tahmini

YoloV3 çok etiketli sınıflandırma kullanır ve sınır kutularının içerebileceği sınıfları tahmin eder. Nesnenin belirli bir sınıfa ait olma olasılığını belirlemek için softmax kullanmak yerine bağımsız lojistik sınıflandırıcı (independent logistic classifier) kullanır. Eğitim aşamasında kayıp fonksiyonu olarak ikili çapraz-entropi (binary cross-entropy) kullanılır.

1.4.7. Kayıp Fonksiyonu

Yolov3 ağında kayıp fonksiyonu delta değerlerinin karelerinin toplamı alınarak hesaplanmaktadır. Delta değerleri sınıf puanları için, nesnellik puanları için ve sınır kutuları için ayrı ayrı hesaplanır.

Sınıflar için delta değeri;

- Eğer sınıf doğru tahmin edilmişse delta = 1-p olarak
- Yanlış tahmin edilmişse delta = -p olarak alınmaktadır.

Burada p lojistik aktivasyonu ifade etmektedir ve $eşitlik\ (1.27)$'de belirtidiği gibi hesaplanır.

$$p = 1/(1 + e^{-x}) \tag{1.27}$$

Nesnellik için delta değeri;

- Eğer tahmin edilen sınır kutusu ile ground truth nesnesi arasındaki IoU değeri .cfg dosyasında belirtilen ignore_thresh eşik değerinden büyükse delta = 0 olarak,
- Eğer IoU değeri ignore_thresh değerinden küçük eşitse delta = -nesnelik puanı olarak,
- Eğer IoU değeri yine cfg dosyasında belirtilen truth_thresh değerinden büyükse delta = 1-nesnellik puanı şeklinde hesaplanır.

Kutular için delta değerleri aşağıda belirtilen formüllerdeki gibi hesaplanır.

$$delta_x = (truth_x - x) * (2 - truth_w * truth_h) \tag{1.28}$$

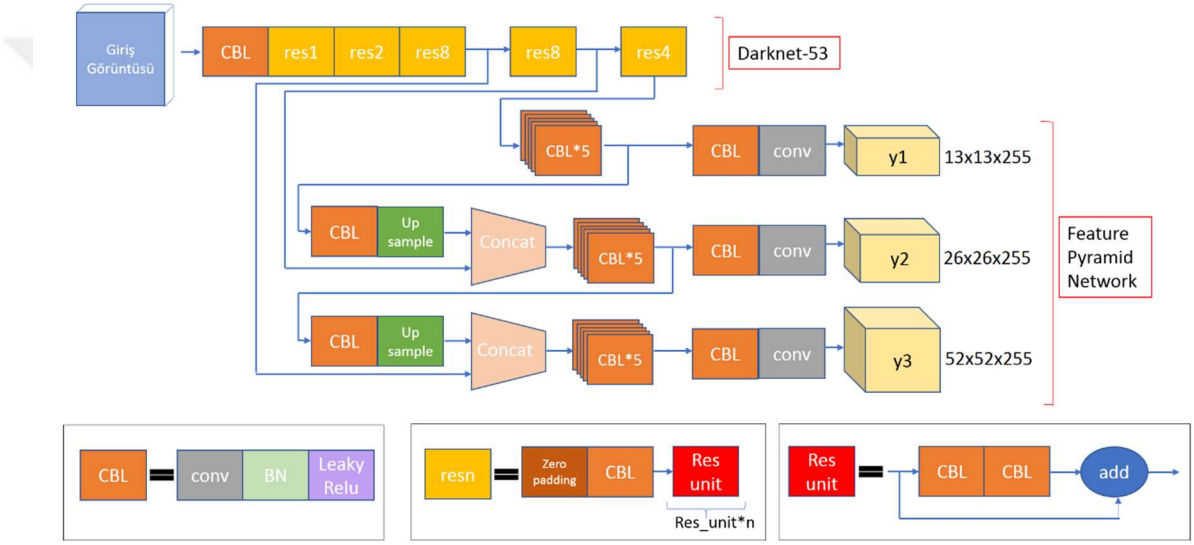
$$delta_y = (truth_y - y) * (2 - truth_w * truth_h) \tag{1.29}$$

$$delta_w = \left(\log\left(truth_w * \frac{w}{anchor_w}\right) - x\right) * (2 - truth_w * truth_h) \tag{1.30}$$

$$delta_h = \left(\log\left(truth_h * \frac{h}{anchor_h}\right) - y\right) * (2 - truth_w * truth_h) \tag{1.31}$$

1.4.8. YoloV3 Ağ Yapısı

YoloV3 ağı Darknet-53 ve Özellik Piramit Ağı (Features Pyramid Network- FPN)'na benzer bir yapıdan oluşmaktadır [20]. Darknet-53 özelliklerin çıkarılması için kullanılır. Özellik Piramit Ağı'da nesneleri tanımak amacıyla kullanılmaktadır. Şekil 1.23'de YoloV3'ün ağ yapısı gösterilmiştir.



Şekil 1.23 YoloV3 Ağ Yapısı [19]

Tüm katmanlarda kullanılan filtreler ve katmanların giriş ve çıkış boyutları Ek-2'de verilmiştir.

1.4.8.1. Darnet-53 Özellik Çıkarıcı

Darknet-53, YoloV2 de kullanılan Darknet-19 [21] ile ResNET'in [22] beraber kullanılmasıyla oluşturulmuş bir özellik çıkarıcıdır. ResNet ağının kullanılması önceki katmanların değerini sonraki katmanlara daha güçlü bir şekilde aktarılmasını sağlamıştır.

Darknet-53 75 katmandan oluşur ve bunların 53 tanesi konvolüsyon katmanı, geri kalanları ise res katmanlarıdır. Res katmanı ardışık 3x3'lük ve 1x1'lik konvolüsyon katmanlarından ve kısayol bağlantılarından oluşmaktadır. YoloV3 ağ mimarisinde çoğu

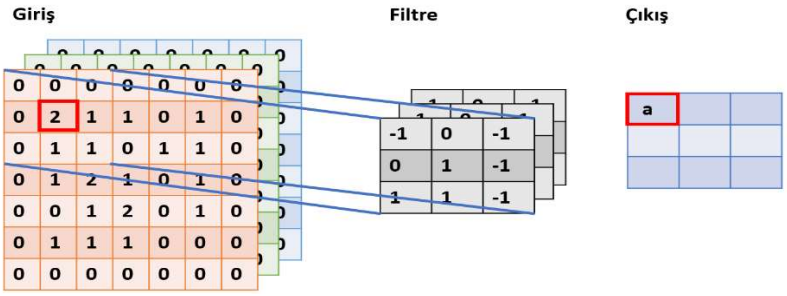
konvolüsyon katmanının ardından yığın normalizasyonu yapılmaktadır. Yığın normalizasyonu'nun ardından ise aktivasyon fonksiyonu olarak Leaky Relu kullanılmaktadır. Darknet-53 saniyede işlediği kayan nokta işlemleri yüksek olduğu için GPU'yu iyi kullanarak değerlendirmeyi hızlı ve doğru bir şekilde yapmaktadır.

1.4.8.2 Özellik Piramit Ağı

YoloV3 özellik piramit ağlarını kullanarak nesne tahmini yapar ve üç farklı ölçekte özellik çıkarmaktadır. Bu üç ölçeğin her biri farklı boyuttaki nesneleri algılamak için kullanılmıştır. 3.ölçekte giriş görüntüsü 52x52'lik ızgaralara bölünür ve bu sayede küçük boyutlu nesnelerin algılanması sağlanır. 2.ölçekte giriş görüntüsü 26x26'lık ızgaralara bölünür ve orta boyutlu görüntülerin algılanması sağlanır. 1.ölçekte ise giriş görüntüsü 13x13'lük ızgaralara bölünerek büyük boyutlu nesnelerin algılanması sağlanır.

1.4.8.3. Konvolüsyon Katmanı

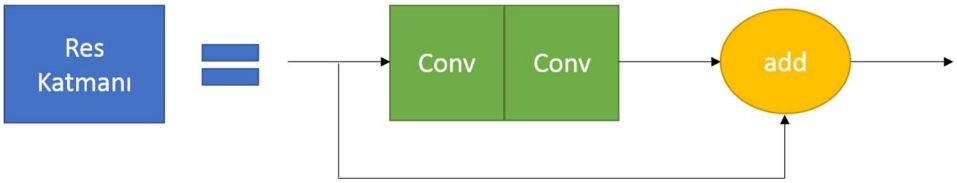
Konvolüsyon katmanı, belirlenen bir filtrenin görüntü matrisi üzerinde gezdirilmesiyle görüntünün özniteliklerinin belirginleşmesi sağlamaktadır. Şekil 1.24.'de konvolüsyon işlemine örnek gösterilmiştir. Burada giriş görüntüsündeki her bir katman filtrenin katmanları ile ayrı ayrı çarpılıp sonrasında toplanmaktadır. 1.katman; (0*-1) + (0*0) + (0*-1) + (0*0) + (2*1) + (1*-1) + (0*1) + (1*1) + (1*-1) = 1 şeklinde hesaplanır. 2.ve 3.katmanlarda bu şekilde hesaplanır en sonunda toplanarak çıkış görüntüsündeki a pikselinin değeri bulunmuş olur (a = 1.katman + 2.katman + 3.katman). Yolov3 ağında sadece 1x1 ve 3x3 boyutlarında matrisler kullanılarak konvolüsyon işlemi yapılır ve her konvolüsyon katmanında sonra Yığın (Batch) Normalizasyonu ve Leaky Relu uygulanır.



Şekil 1.24. Konvolüsyon katmanı

1.4.8.4. Res Katmanı

ResNet(Residual Network) [22] derin öğrenme ağlarında çok katmanlı bir ağ aracılığıyla önceki katmanların değerini sonraki katmanlara aktararak iyi sonuçların korunmasına yardımcı olmaktadır. Şekil1.15'te Res katmanının yapısı gösterilmiştir. Görüldüğü gibi konvolüsyon katmanlarının girişi ile çıkışı arasında bir kısayol(shortcut) bağlantısı yapılmıştır. Bu bağlantı sayesinde konvolüsyon katmanlarının çıkışına iki önceki katmandan gelen değerler eklenmiş olmaktadır. Mevcut ağırlık 0 olsa bile önceki katmandan gelen değerler ile öğrenme hatası optimize edilmiş olur.



Şekil 1.25. Res Katmanı

1.4.8.5. Yığın(Batch) Normalizasyonu

Derin sinir ağının normalize edilmesi ağın hızını, performansını ve dengesini önemli ölçüde etkilemektedir [23]. Katmana girdi olarak alınan veriler ölçeklenerek belirli bir aralıkta temsil edilirler. Bu sayede veriler daha düzenli hale gelmiş olur.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1.32}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{1.33}$$

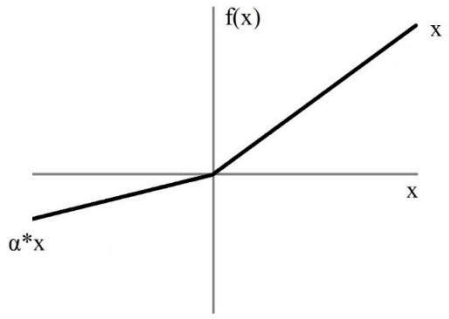
Yığın normalizasyonu yapmak için katmana gelen verilerin eşitlik (1.32) ve (1.33)'te gösterildiği gibi önce ortalaması sonra varyansı bulunmaktadır.

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \in}} \tag{1.34}$$

Denklem (1.34)'te görüldüğü gibi her bir boyutu ayrı ayrı normalleştirilir. Burada (YoloV3'de) $\in$ değeri 10^{-6} olarak alınmıştır.

1.4.8.6 Leaky(Sızıntı) ReLU Katmanı

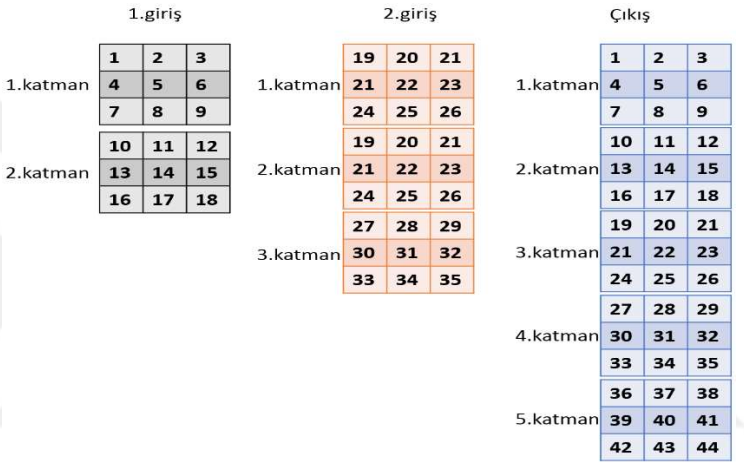
Leaky Relu bir aktivasyon katmanıdır. Bu fonksiyona göre Şekil 1.16'te görüldüğü gibi f(x), x eğer sıfırdan büyük eşitse x, sıfırdan küçükse a*x olmaktadır. Burada *a* değeri YoloV3'te 0.1 olarak alınmıştır.



Şekil 1.26. Leaky(Sızıntı) ReLU fonksiyonu

1.4.8.7. Birleştirme (Concatenation) Katmanı

Birleştirme katmanında alınan girişler belirtilen bir boyut boyunca birleştirilir. Girişler birleştirme boyutu hariç diğer boyutlarda aynı boyutta olmaktadır. Şekil 1.17'e bakıldığında 1.giriş 3x3x2 boyutunda, 2.giriş ise 3x3x3 boyutundadır. Bu girişler birleştirilerek 3x3x5 boyutlu bir çıkış elde edilmektedir.



Şekil 1.27 Birleştirme Katmanı

2. YAPILAN ÇALIŞMALAR, BULGULAR VE İRDELEME

2.1. Giriş

Bu tez çalışmasında mikroskoptan elde edilen sitopatolojik görüntülerinde bulunan çekirdeklerin gerçek zamanlı olarak otomatik bir şekilde hızlı ve doğru algılanması amaçlanmıştır. Öncelikle YoloV3 algoritmasının eğitiminde kullanılacak veri seti panorama yöntemiyle oluşturulmuştur. Panorama yönteminde görüntüler mikroskoptan sıralı bir şekilde alınmaktadır. Her bir görüntü ile bir sonraki görüntü arasında ortak noktalar bulunur ve panorama oluşturulur. Böylece elde edilen panorama kullanılarak YoloV3 nesne algılama algoritması için eğitim ve test veri setleri hazırlanır. Hazırlanan eğitim veri seti ile Yolov3 ağı eğitilerek sitopatolojik görüntülerdeki çekirdeklerin algılanması sağlanmıştır. Son olarak hazırlanan test veri setiyle bu ağın başarısı ölçülüp diğer yöntemlerle karşılaştırılmıştır [24].

2.2. Sıralı Panorama Üretimi

Mikroskoptan görüntüler sıralı bir şekilde alınıp Y_X.bmp formatında kaydedilmektedir. 0_0.bmp den 5_10'a kadar toplam 66 görüntü okunmuştur. Burada Y satır numarasını, X ise sütun numarasını belirtmektedir. Şekil 2.1'de görüldüğü gibi ilk satırda görüntüler sağa doğru ilerleyerek, sonraki satırda ise sola doğru ilerleyerek elde edilmektedir. Bu şekilde her bir satırda yön değişerek son görüntüye kadar devam edilir.

0-0.bmp →	0-1.bmp →	0-2.bmp →	…	0-10.bmp ↓
↓ 1-10.bmp	…	← 1-2.bmp	← 1-1.bmp	← 1-0.bmp
2-0.bmp →	2-1.bmp →	2-2.bmp →	…	2-10.bmp ↓
⋮	⋮	⋮	⋮	⋮
5-10.bmp	…	← 5-2.bmp	← 5-1.bmp	← 5-0.bmp

Şekil 2.1. Görüntülerin birleştirilme yönü

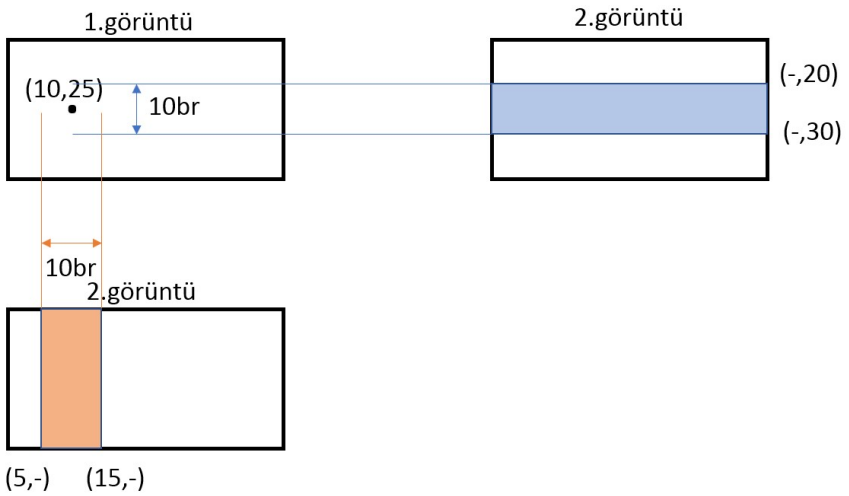
2.2.1. Sıralı Panorama İçin 1.Yöntem

Sıralı panorama üretimi için aşağıdaki algoritma adımları uygulanmıştır.

1. Öncelikle ilk iki görüntü seçilir ve bu görüntülerin köşe noktaları ve öznitelik vektörleri hesaplanır.
2. Bu özniteliklere göre köşe noktaları eşleştirilir.
3. Eşleştirilen köşe noktaları üzerinde RANSAC algoritması uygulanarak yanlış eşleştirmeler elimine edilir ve homografi matrisi bulunur.
4. Laplace piramiti uygulanarak görüntüler birleştirilerek panorama oluşturulur.
5. Burada kullanılan 2.görüntü sonraki adımlar için artık 1.görüntümüz olmaktadır. Yeni gelen görüntü de 2.görüntü olarak alınmaktadır. Yani yeni gelen görüntüyle eşleştirme yapmak için panoramanın tamamını almak yerine sadece son gelen görüntü alınır.
6. Daha önce hesaplanan homografi matrisi homog_mat_prev değişkeninde tutulur.
7. Yeni 1.ve 2.görüntülerimiz için tekrar köşe noktaları ve öznitelikler hesaplanarak eşleştirme yapılır.
8. RANSAC algoritması uygulanarak yanlış eşleştirmeler elenir.
9. 1.görüntü için bulunan eşleştirme noktalarının koordinatlarının önceki adımda oluşturulan panorama da hangi noktalara denk geldiğini bulmak için homog_mat_prev ile bu köşe noktaları çarpılır.
10. Hesaplanan bu noktalar ile 2.görüntü arasında yeni bir homografi matrisi hesaplanır.
11. Laplace piramiti kullanılarak yeni panorama görüntüsü oluşturulmuş olmaktadır.
12. Tekrar 5.adıma gidilerek bu işlemler görüntüler bitene kadar tekrar edilir.

Bu yöntemde köşe noktalarının eşleştirilme aşamasında bazı değişiklikler yapılarak homografi matrisinin daha doğru hesaplanması sağlanmıştır. Öncelikle köşe noktalarının daha düzgün eşleştirilmesi amacıyla öznitelik eşleştirme adımında bir bant aralığı ayarlanmıştır. Görüntülerin birleşme yönü bilinmektedir. Bu bilgiden faydalanarak eğer görüntünün birleşme yönü sağa veya sola ise y ekseni yönünde bir bant aralığı ayarlanır. Örneğin Şekil 2.2'ye bakıldığında ilk görüntünün köşe noktası (10,25) olsun. Bant aralığı 10 olarak ayarlandığı için 2.görüntüde bu nokta ile eşleşebilecek köşe noktaları (-,20), (-,30) aralığında olmaktadır. Aynı

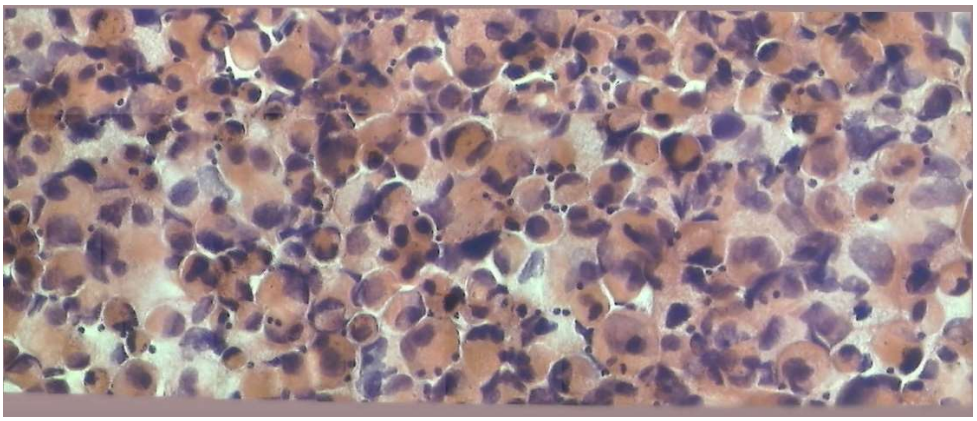
şekilde eğer görüntünün birleşme yönü aşağıya doğru ise bant aralığı x eksenine göre yapılmaktadır. Yani (10,25) için bant aralığı (5,-), (15,-) değerleri arasında olmaktadır.



Şekil 2.2. Bant Aralığı Gösterimi

Yine burada bazı durumlarda bir görüntüdeki birden fazla köşe noktası diğer görüntüdeki aynı köşe noktasıyla eşleşebilmektedir. Bu durumu engellemek için bir köşe noktası için daha önce eşleştirme yapılmışsa o köşe noktası diğer köşe noktalarıyla tekrar karşılaştırılmaz. Böylece her köşe noktası ayrı köşe noktalarıyla eşleştirilmiş olmaktadır.

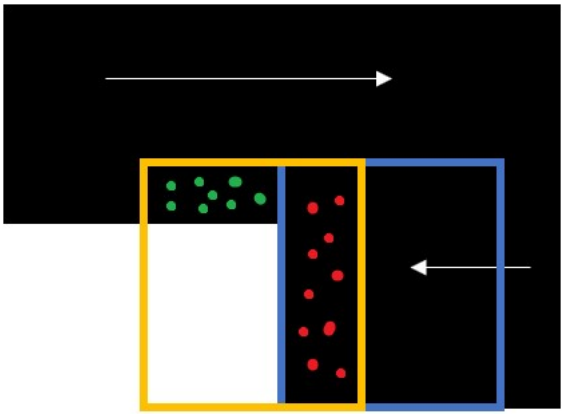
Bu şekilde yapılan değişikliklerle köşe noktaları daha düzgün eşleştirilip Ransac algoritmasına verilir. Ayrıca Ransac algoritmasında da rastgele seçilen 4 nokta için ek olarak bir düzenleme yapılmıştır. Normalde düzgün eşleşen köşe noktalarının aralarındaki mesafe birbirine eşit olmalıdır. Buna göre her bir eşleşen noktaların birbirlerine olan uzaklıkları bulunur. Eğer birleşme x yönündeyse noktaların x koordinatları arasındaki fark alınarak, eğer birleşme y yönünde ise y koordinatları arasındaki fark alınarak uzaklık hesabı yapılmıştır. Bulunan bu uzaklıkların birbirine yakın olma koşulu getirilmiştir. Eğer yakın değilse rastgele başka 4 nokta seçilerek tekrar uzaklıklar hesap edilmektedir. Bu düzenlemeler yapılarak eşleşen köşe noktaları daha düzgün olduğu için homografi matrisi daha doğru hesaplanmaktadır. Bu yönteme göre elde edilen panoramaya Şekil2.3.'de örnek verilmiştir.



Şekil 2.3. Sıralı Panorama 1.yöntem sonuç görüntüsü

2.2.2. Sıralı Panorama İçin 2.Yöntem

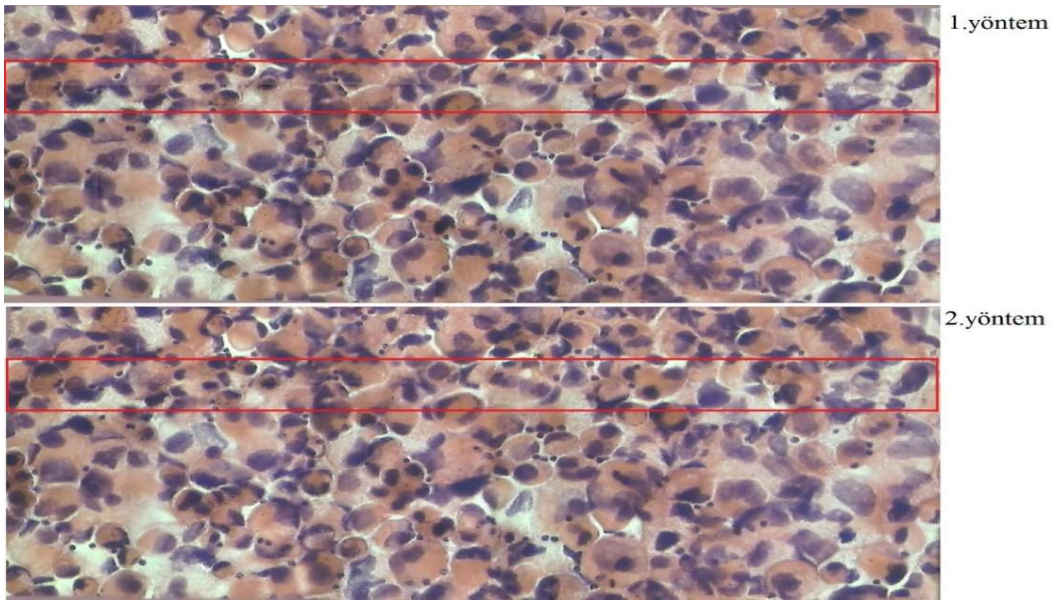
1.yöntemin bazı dezavantajları bulunmaktadır. Bu nedenle 2.yöntem geliştirilmştir. Şekil 2.4'e baktığımızda burada siyah bölge oluşturulan panorama görüntüsünü ifade etmektedir. İlk satır sağa doğru birleştirilerek panorama elde edilmiş ikinci satırda hareket sola doğru devam etmektedir. Mavi ile gösterilen bölge panoramaya son eklenen görüntüyü temsil etmektedir. Sarı ile temsil edilen bölge ise panoramaya eklenecek yeni görüntüyü temsil etmektedir. Yeni panorama oluşturulurken sadece son gelen görüntü(mavi) ile yeni gelen görüntü(sarı) arasında eşleştirme yapılmaktadır. Yani sadece kırmızı noktalarla gösterilen bölgede eşleştirme yapılmaktadır. Bu da yeni gelen görüntünün yeşil noktalarla gösterilen bölgelerde panorama ile eşleştirme yapılamadığından bu bölgelerde görüntüler tam olarak örtüşememektedir bu nedenle bazı bozulmalar meydana gelebilmektedir.



Şekil 2.4. Panorama 1.Yöntem Eşleştirme Bölgeleri

Bu yöntem 1.yöntem üzerinde bazı değişiklikler yaparak elde edilmiştir ve Şekil 2.4'te gösterilen yeşil bölgelerde de eşleştirme yapılarak panoramanın daha düzgün bir şekilde elde edilmesi amaçlanmaktadır. Bunu düzeltmek için birleştirme esnasında sadece panoramaya eklenen son görüntüyü almak yerine son eklenen görüntünün konumu kullanılarak panoramadan isteğe bağlı daha büyük bir alan seçimi yapılmıştır böylece daha düzgün bir panorama elde edilmiştir. Bu alan seçimi son gelen görüntünün genişliğinin 2 katı olacak şekilde seçilmiştir.

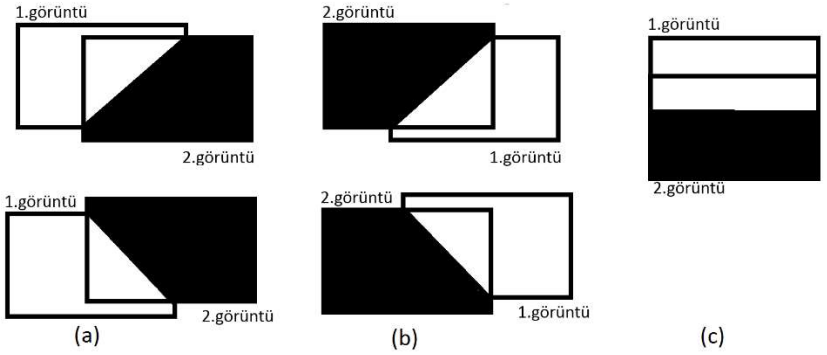
Şekil 2.5'e bakıldığında 1.yöntemde oluşan kayma problemi 2.yöntemle düzeltilmiştir.



Şekil 2.5. 1.ve 2.yöntemden elde edilen görüntüler

2.2.3. Maske Seçimi

Görüntülerin birleştirilme aşamasında maskenin belirlenmesi gerekmektedir. Bu maske 1.ve 2.görüntülerden hangi bölgelerin alınacağını belirtiyordu. Burada görüntülerin birleştirme yönüne göre maske dinamik olarak ayarlanmıştır. Şekil 2.6.'daki a görüntüsünde x yönünde sağa doğru hareket eden görüntüler için, b x yönünde sola doğru hareket eden görüntüler için c ise y yönünde aşağıya doğru hareket eden görüntüler için belirlenen maskeler. Buradaki beyaz bölgeler 1.görüntüden, siyah bölgeler ise 2.görüntüden gelmektedir.



Şekil 2.6. Yönlere göre belirlenen maskeler

2.3. YoloV3 Veri Setinin Oluşturulması

Elde edilen panorama görüntülerinden 600x600 boyutlarında kesilerek 200 görüntü elde edilmiştir. Ve her bir görüntü yaklaşık 100 çekirdek içermektedir. Veri seti eğitim ve test aşamaları için ikiye bölünmüştür. Eğitim için 80 test için 120 görüntü kullanılmıştır. Plevral efüzyon sitopatolaji görüntülerinde oluşan bu veri setleri Karadeniz Teknik Üniversitesin 'de uzman patologlar tarafından etiketlenerek eğitim için hazırlanmıştır. Bu görüntüler gürültü, boya ve yağ artefaktı gibi çeşitli problemler içermektedir.

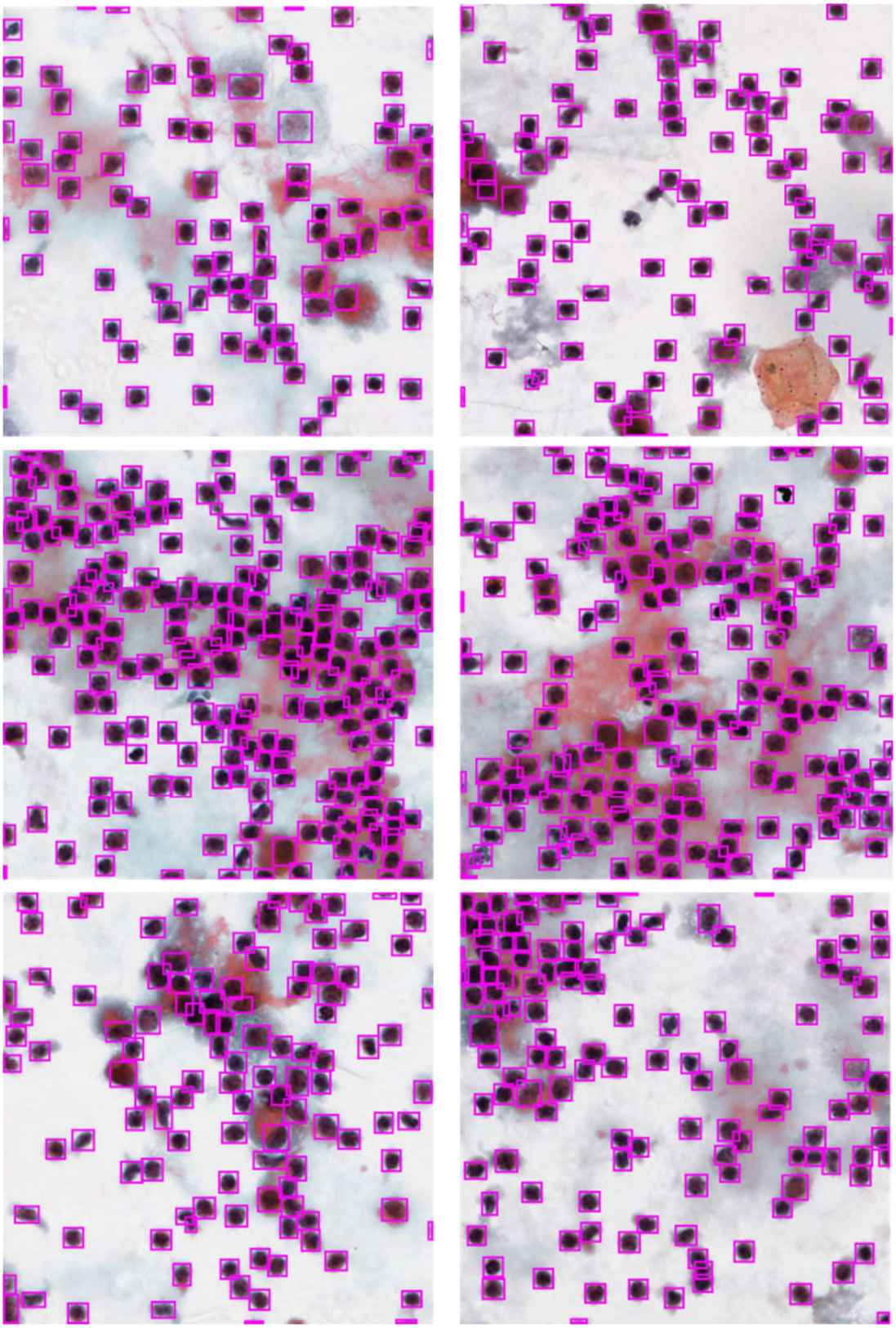
2.4. YoloV3 Ağının Eğitilmesi

Çalışmada Yolov3 mimarisi önerildiği gibi kullanılmıştır. Darknet-53 ağı giriş görüntülerini 32'nin katı olarak kabul ettiği için ağın girişi 416x416 olarak ayarlanmıştır. Ayrıca momentum 0.9, bozulma 0.0005, öğrenme oranı 0.0001 ve yığın boyutu 64 olarak ayarlanmıştır. Orjinal YoloV3'ten farklı olarak Tablo 2.1'de gösterilen boyutlarda 11 çapa kutusu kullanılmıştır. Orijinal YoloV3'te her bir ölçekte 3 sınır kutusu için toplamda 9 sınır kutu tahmini yapılır. 1.ölçek büyük boyutlu görüntüleri algılamak için kullanılır bu nedenle seçilen çapa boyutları 60x60'ın üzerinde olmalıdır. 2.ölçek orta boyutlu görüntüleri algılamak için kullanılır ve bu nedenle çapa boyutları 30x30'un üzerinde olması gerekir. 3.ölçek ise küçük boyutlu nesneleri algılamak için kullanılmaktadır. Ürettiğimiz veri tabanındaki sitopatolojik görüntülerde çekirdekler genellikle küçük boyutlu olduğu için orijinal YoloV3'te kullanılan çapa boyutları değiştirilmiştir. 3.ölçekte tahmin edilen sınır kutu sayısı 9'a çıkarılmıştır. Bu değerler K-Means Kümeleme Algoritması kullanılarak üretilmiştir. Diğer ölçeklerde ise bir tane sınır kutu tahmini yapılacak şekilde ayarlanmıştır ve boyutları minimum olabilecek şekilde yani 3.ölçek için (60,60), 2.ölçek için ise (30,30) olarak seçilmiştir.

Tablo 2.1 Her bir ölçekte kullanılan çapa kutularının boyutları

	Orijinal(YoloV3 değerleri)	Önerilen
Ölçek 1	116,90;156,198;373,326	60,60
Ölçek 2	30,61;62,45;59,119	30,30
Ölçek 3	10,13;16,30;33,23	11,18;19,12;18,19;17,23;23,19;21,22;22,26;25,23;29,29

YoloV3 bu şekilde ayarlandıktan sonra sitopatolojik görüntülerden oluşan veri setiyle eğitilmiştir ve Şekil 2.7'de çekirdek algılama sonuçları gösterilmiştir.



Şekil 2.7. Önerilen yöntemin çekirdek algılama sonuçları

Önerilen Yolov3 nesne algılayıcısının performansını diğer konvolüsyonel nesne algılayıcıları ile karşılaştırabilmek için 4 ölçüt kullanılmıştır. Bu ölçütler kesinlik, gerçek pozitif oranı(TPR), yanlış pozitif oranı (FPR) ve F-ölçütüdür. Elde edilen algılama, algılanan pencere merkezi ile kesin referans pencere merkezi arasındaki farka göre Doğru Pozitif (True Positive-TP) olarak kabul edilmiştir. Eşik değeri olarak, fark 10 pikselden düşükse, [11] 'de belirtildiği gibi TP olarak kabul edilmiştir. TN, FP ve FN sırasıyla doğru negatif, yanlış pozitif ve yanlış negatiftir. Ayrıca, gerçek zamanlı algılama için önemli olan algılama hızı/test süresi de ölçüt olarak kullanılmıştır.

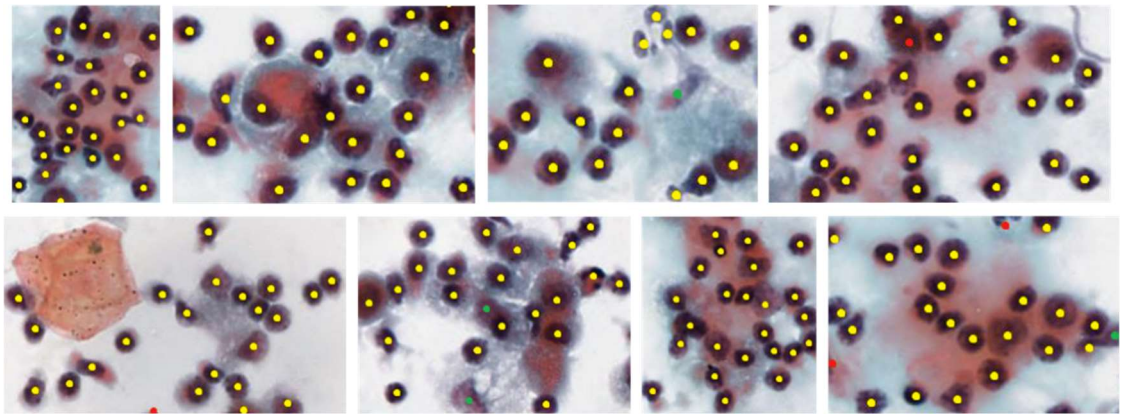
$$Kesinlik = \frac{TP}{TP + FP} \tag{2.1}$$

$$Hassasiyet = \frac{TP}{TP + FN} \tag{2.2}$$

$$Yanlış\ Pozitif\ Oranı = \frac{FP}{FP + TN} \tag{2.3}$$

$$F - \text{ölçütü} = 2x\frac{KesinlikxHassasiyet}{Kesinlik + Hassasiyet} \tag{2.4}$$

Önerilen yöntemde bu ölçütlere göre karmaşık arkaplana sahip görüntüler üzerinde elde edilen sonuçlar Şekil 2.8.'de gösterildiği gibidir. Burada sarı noktalar TP, kırmızı noktalar FP, yeşil noktalar ise FN değerlerini göstermektedir.



Şekil 2.8. Algılanan yöntemin TP, FP, FN sonuçlarına örnekler

3. SONUÇLAR VE TARTIŞMA

Bu çalışmada, önerilen yöntem daha önce aynı veri seti üzerinde uygulanan diğer konvolüsyonel nesne algılayıcılar ile (Daha Hızlı R-CNN, R-FCN ve SSD) [12] karşılaştırılmıştır. Tüm testler 8GB RAM ve bir NVIDIA GeForce GTX 1050Ti GPU barındıran Intel Core i7 işlemcili bir sistem üzerinde gerçekleştirilmiştir.

Önerilen yöntemin önceki çalışmalarla karşılaştırmalı çekirdek algılama sonuçları Tablo-3'te verilmiştir. Sonuçlar irdelendiğinde, yöntemin en önemli katkısının algılama hızı olduğu sonucuna varılmıştır. Önerilen yöntemin çekirdek algılama hızını 10 kat artırabileceği ve yüksek kesinlik ve hassasiyet sağladığı deneysel olarak kanıtlanmıştır. DarkNet-53, saniyede en yüksek kayan nokta işlemi gerçekleştirdiğinden, YOLOv3, Faster R-CNN, R-FCN ve SSD ile karşılaştırıldığında Tablo-3'te görüldüğü gibi en hızlı algılama sağlayan konvolüsyonel nesne algılayıcıdır. Önerilen yöntem sonuç olarak %94.10 kesinlik, %98.98 hassasiyet ve %96.48 F-ölçütü sağlamıştır. Diğer yöntemlerle karşılaştırıldığında YOLOv3 en yüksek hassasiyete sahiptir. Diğer sonuçlardaki düşüklük ise yöntemin yanlış algılama (FP) oranının fazla olmasından kaynaklanmaktadır.

Tablo 3.1: Önerilen yöntemin plevral efüzyon sitopatoloji veriseti üzerine yapılan çalışmalara göre çekirdek algılama performansı (en iyi sonuçlar kalın yazılmıştır)

NUCLEI DETECTION PERFORMANCE OF THE PROPOSED METHOD ACCORDING TO THE PREVIOUS STUDIES ON THE PLEURAL EFFUSION CYTOPATHOLOGY DATASET (BEST RESULTS ARE BOLDED).									
ObjectDetector	FeatureExtractor	TP	FP	FN	Precision(%)	Recall(%)	FPR(%)	F-measure(%)	Test time (sec/img)
Faster R-CNN	ResNet-101	10815	71	342	99.34	96.93	0.63	98.12	1.627
Faster R-CNN	Inceptionv2	10723	97	434	99.10	96.11	0.86	97.58	1.121
R-FCN	ResNet-101	10833	279	324	97.48	97.09	2.50	97.28	1.912
SSD	MobileNet	9747	337	1410	96.65	87.36	3.02	91.77	0.688
SSD	Inceptionv2	9739	342	1418	96.60	87.29	3.0	91.70	0.903
YOLOv3	DarkNet-53	11044	692	113	94.10	**98.98**	6.20	96.48	**0.060**

4. ÖNERİLER

Plevral efüzyon sitopatolojide sıklıkla karşılaşılan bir numunedir. Çekirdek hücre hakkında önemli bilgiler içerdiğinden çekirdek algılama ve segmentasyonu sitopatolojik değerlendirme sürecinde temel adım olarak görülmektedir. Algılama hızı gerçek zamanlı bilgisayar-destekli tanı uygulamalarında çok önemlidir. Bunun sebebi numunelerin milyonlarca hücre içermesidir. Literatürdeki yöntemler yüksek kesinlik sağlamalarına rağmen gerçek zamanlı uygulamalar için yeterli hız sağlayamamaktadır. Bu çalışmada YOLOv3 yöntemi plevral efüzyon sitopatolojisinde çekirdek algılama amacıyla önerilmiştir. Önerilen yöntem literatürdeki mevcut çalışmalardan farklı olarak 10 kat yüksek algılama hızı sağlamıştır. Gerçek zamanlı uygulamalar için bu katkı çok önemli bir katkı olmuştur. Bu çalışmada YOLOv3 önerildiği haliyle kullanılmıştır ancak gelecek çalışmalarda yöntem yanlış pozitif oranını azaltacak şekilde geliştirilecektir. Yöntemin ayrıca farklı büyütme objektiflerindeki (20x, 60x, vb.) ve farklı boyalardaki (Giemsa, vb.) başarısı test edilecektir.

5. KAYNAKLAR

1. Davidson B., Firat P. ve Michael CW., Serous effusions: etiology. Prognosis and Therapy. SpringerScience & Business Media, Diagnosis, 2011.

2. Shidham VB. ve Atkinson BF., Cytopathologic diagnosis of serous fluids e-book. Elsevier HealthSciences, 2007.

3. DeBiasi, E. M., Pisani, M. A., Murphy, T. E., Araujo, K., Kookoolis, A., Argento, A. C., ve Puchalski, J., Mortality among patients with pleural effusion undergoing thoracentesis. European Respiratory Journal, 46, 2 (2015) 495-502.

4. Marel, M., Zrtov, M., tasny, B. ve Light, R. W., The incidence of pleural effusion in a well-defined region: epidemiologic study in central Bohemia. Chest, 104, 5 (1993) 1486-1489.

5. Redmon, J. ve Farhadi, A., Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

6. Rosten, E., ve Drummond, T., Machine learning for high-speed corner detection. In European conference on computer vision, 430-443, Springer, Berlin, Heidelberg, Mayıs 2006.

7. Fischler M.A. ve Bolles R.C., Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24, 6 (1981) 381–395.

8. Burt, P. ve Adelson, E., The Laplacian pyramid as a compact image code. IEEE Transactions on communications, 31, 4 (1983) 532-540.

9. Sheppard C. ve Wilson T., Depth of field in the scanning microscope. Optics Lett 3, 3 (1978) 115–117.

10. Baykal E., Dogan H., Ekinci M., Ercin M.E. ve Ersoz S.¸ Automated nuclei detection in serous effusion cytology based on machine learning. In: Signal processing and communications applications conference (SIU), 2017 25th. IEEE 1–4.

11. Baykal E., Dogan H., Ercin M.E., Ersoz S. ve Ekinci M. Automated nuclei detection in serous effusion cytology with stacked sparse autoencoders. In: Signal processing and communications applications conference (SIU), 2018 26th. IEEE, 1–4.

12. Baykal, E., Dogan, H., Ercin, M. E., Ersoz, S. ve Ekinci, M., Modern convolutional object detectors for nuclei detection on pleural effusion cytology images. Multimedia Tools and Applications, 2019, 1-20.

13. Lowe, D. G., Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60, 2 (2004), 91-110.

14. Shah, M., Fundamentals of computer vision. University of Central Florida,1997.

15. Asirvatham A.P., Gaussian and Laplacian Pyramids. Report on Final Year Project titled Script Segmentation of Multi-script Documents, International Institute of Information Technology https://www.cs.utah.edu/~arul/report/node12.html 20 Mart 2019

16. Weighted sum for each level of the pyramid https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger /lecture_2_3_blending.pdf Accessed 20 Kasım 2019

17. Jeon T., You Only Look Once:Unified Real-Time Object Detection https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection 10 Eylül 2019

18. Hartigan, J.A. ve Wong, M.A., Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28, 1 (1979) 100-108.

19. https://www.researchgate.net/figure/YOLOv3-architecture-A-YOLOv3-pipeline-with-input-image-size-416416-and-3-types-of_fig4_334021766 YoloV3 Architecture 18 Aralık 2019

20. Lin T.Y., Dollar P., Girshick R., He K., Hariharan B. ve Belongie S., Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2, 3 (2017) 2117–2125.

21. Redmon, J. ve Farhadi, A., YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017, 7263-7271.

22. He, K., Zhang, X., Ren, S. ve Sun, J., Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 770-778.

23. Ioffe, S. ve Szegedy, C., Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

24. Kilic, B., Baykal, E., Ekinci, M., Dogan, H., Ercin, M. E. ve Ersoz, S., Automated Nuclei Detection on Pleural Effusion Cytopathology Images using YOLOv3. In 2019 4th International Conference on Computer Science and Engineering (UBMK), Eylül 2019 IEEE 1-5.

6. EKLER

Ek 1.

Katmanlar	Filtre	Boyut	Giriş		Çıkış
0 conv	32	3 x 3 /1	416 x 416 x 3	->	416 x 416 x 32 0.299 BF
1 conv	64	3 x 3 /2	416 x 416 x 32	->	208 x 208 x 64 1.595 BF
2 conv	32	1 x 1 /1	208 x 208 x 64	->	208 x 208 x 32 0.177 BF
3 conv	64	3 x 3 /1	208 x 208 x 32	->	208 x 208 x 64 1.595 BF
4 Shortcut Layer: 1					
5 conv	128	3 x 3 /2	208 x 208 x 64	->	104 x 104 x 128 1.595 BF
6 conv	64	1 x 1 /1	104 x 104 x 128	->	104 x 104 x 64 0.177 BF
7 conv	128	3 x 3 /1	104 x 104 x 64	->	104 x 104 x 128 1.595 BF
8 Shortcut Layer: 5					
9 conv	64	1 x 1 /1	104 x 104 x 128	->	104 x 104 x 64 0.177 BF
10 conv	128	3 x 3/ 1	104 x 104 x 64	->	104 x 104 x 128 1.595 BF
11 Shortcut Layer: 8					
12 conv	256	3 x 3 /2	104 x 104 x 128	->	52 x 52 x 256 1.595 BF
13 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
14 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
15 Shortcut Layer: 12					
16 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
17 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
18 Shortcut Layer: 15					
19 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
20 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
21 Shortcut Layer: 18					
22 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
23 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
24 Shortcut Layer: 21					
25 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
26 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
27 Shortcut Layer: 24					
28 conv	128	1 x 1 /1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF
29 conv	256	3 x 3 /1	52 x 52 x 128	->	52 x 52 x 256 1.595 BF
30 Shortcut Layer: 27					
31 conv	128	1 x 1 / 1	52 x 52 x 256	->	52 x 52 x 128 0.177 BF

Ek- 1'in devamı

```
32 conv     256        3 x 3 / 1     52 x  52 x 128     ->   52 x  52 x 256 1.595 BF
33 Shortcut Layer: 30
34 conv     128        1 x 1 / 1     52 x  52 x 256     ->   52 x  52 x 128 0.177 BF
35 conv     256        3 x 3 / 1     52 x  52 x 128     ->   52 x  52 x 256 1.595 BF
36 Shortcut Layer: 33
37 conv     512        3 x 3 / 2     52 x  52 x 256     ->   26 x  26 x 512 1.595 BF
38 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
39 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
40 Shortcut Layer: 37
41 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
42 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
43 Shortcut Layer: 40
44 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
45 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
46 Shortcut Layer: 43
47 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
48 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
49 Shortcut Layer: 46
50 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
51 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
52 Shortcut Layer: 49
53 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
54 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
55 Shortcut Layer: 52
56 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
57 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
58 Shortcut Layer: 55
59 conv     256        1 x 1 / 1     26 x  26 x 512     ->   26 x  26 x 256 0.177 BF
60 conv     512        3 x 3 / 1     26 x  26 x 256     ->   26 x  26 x 512 1.595 BF
61 Shortcut Layer: 58
62 conv     1024       3 x 3 / 2     26 x  26 x 512     ->   13 x  13 x 1024 1.595 BF
63 conv     512        1 x 1 / 1     13 x  13 x1024     ->   13 x  13 x 512 0.177 BF
64 conv     1024       3 x 3 / 1     13 x  13 x 512     ->   13 x  13 x 1024 1.595 BF
65 Shortcut Layer: 62
66 conv     512        1 x 1 / 1     13 x  13 x1024     ->   13 x  13 x 512 0.177 BF
67 conv     1024       3 x 3 / 1     13 x  13 x 512     ->   13 x  13 x 1024 1.595 BF
68 Shortcut Layer: 65
69 conv     512        1 x 1 / 1     13 x  13 x1024     ->   13 x  13 x 512 0.177 BF
70 conv     1024       3 x 3 / 1     13 x  13 x 512     ->   13 x  13 x 1024 1.595 BF
71 Shortcut Layer: 68
72 conv     512        1 x 1 / 1     13 x  13 x1024     ->   13 x  13 x 512 0.177 BF
73 conv     1024       3 x 3 / 1     13 x  13 x 512     ->   13 x  13 x1024 1.595 BF
74 Shortcut Layer: 71
```

Ek- 1'in devamı

```
75 conv      512       1 x 1 / 1      13 x  13 x1024   ->   13 x  13 x 512 0.177 BF
76 conv      1024      3 x 3 / 1      13 x  13 x 512   ->   13 x  13 x 1024 1.595 BF
77 conv      512       1 x 1 / 1      13 x  13 x1024   ->   13 x  13 x 512 0.177 BF
78 conv      1024      3 x 3 / 1      13 x  13 x 512   ->   13 x  13 x 1024 1.595 BF
79 conv      512       1 x 1 / 1      13 x  13 x1024   ->   13 x  13 x 512 0.177 BF
80 conv      1024      3 x 3 / 1      13 x  13 x 512   ->   13 x  13 x 1024 1.595 BF
81 conv      255       1 x 1 / 1      13 x  13 x1024   ->   13 x  13 x 255 0.006 BF
82 yolo
83 route  79
84 conv      256       1 x 1 / 1      13 x  13 x 512   ->   13 x  13 x 256 0.044 BF
85 upsample              2x           13 x  13 x 256   ->   26 x  26 x 256
86 route  85 61
87 conv      256       1 x 1 / 1      26 x  26 x 768   ->   26 x  26 x 256 0.266 BF
88 conv      512       3 x 3 / 1      26 x  26 x 256   ->   26 x  26 x 512 1.595 BF
89 conv      256       1 x 1 / 1      26 x  26 x 512   ->   26 x  26 x 256 0.177 BF
90 conv      512       3 x 3 / 1      26 x  26 x 256   ->   26 x  26 x 512 1.595 BF
91 conv      256       1 x 1 / 1      26 x  26 x 512   ->   26 x  26 x 256 0.177 BF
92 conv      512       3 x 3 / 1      26 x  26 x 256   ->   26 x  26 x 512 1.595 BF
93 conv      255       1 x 1 / 1      26 x  26 x 512   ->   26 x  26 x 255 0.012 BF
94 yolo
95 route  91
96 conv      128       1 x 1 / 1      26 x  26 x 256   ->   26 x  26 x 128 0.044 BF
97 upsample              2x           26 x  26 x 128   ->   52 x  52 x 128
98 route  97 36
99 conv      128       1 x 1 / 1      52 x  52 x 384   ->   52 x  52 x 128 0.266 BF
100 conv     256       3 x 3 / 1      52 x  52 x 128   ->   52 x  52 x 256 1.595 BF
101 conv     128       1 x 1 / 1      52 x  52 x 256   ->   52 x  52 x 128 0.177 BF
102 conv     256       3 x 3 / 1      52 x  52 x 128   ->   52 x  52 x 256 1.595 BF
103 conv     128       1 x 1 / 1      52 x  52 x 256   ->   52 x  52 x 128 0.177 BF
104 conv     256       3 x 3 / 1      52 x  52 x 128   ->   52 x  52 x 256 1.595 BF
105 conv     255       1 x 1 / 1      52 x  52 x 256   ->   52 x  52 x 255 0.025 BF
106 yolo
```

ÖZGEÇMİŞ

Büşranur KILIÇ, 1993 Trabzon doğumludur. İlkokulu ve ortaokulu Dumlupınar İlköğretim Okulu'nda ve liseyi Maçka Mehmet Akif Ersoy Anadolu Lisesi'nde tamamlamıştır. Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2015 yılında mezun olmuştur. 2015- 2016 eğitim–öğretim yılının bahar döneminde Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans programına katılmıştır. İyi derecede İngilizce bilmektedir.