T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

ÇİZGELERİN DÜZLEMSEL GÖSTERİMLERİNİN ELDE EDİLMESİ

Yüksek Lisans Tezi

Uğur ÖNER

Bilgisayar Mühendisliği Anabilim Dalı

İzmir
2023

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

ÇİZGELERİN DÜZLEMSEL GÖSTERİMLERİNİN ELDE EDİLMESİ

Uğur Öner

Danışman : Prof. Dr. Vecdi AYTAÇ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

İzmir
2023

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "Çizgelerin Düzlemsel Gösterilimlerinin Elde Edilmesi" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

09/02/2023

İmzası

Uğur Öner

ÖZET

ÇİZGELERİN DÜZLEMSEL GÖSTERİMLERİNİN ELDE EDİLMESİ

ÖNER, Uğur

Yüksek Lisans Tezi Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Vecdi AYTAÇ

Şubat 2023, 60 Sayfa

Evrendeki tüm nesneleri ve bunların aralarındaki ilişkileri çizgelerle modellemek mümkündür. Bu modellemeler kullanılarak analizler yapılabilir, bilimsel hesaplamalar daha hızlı çözülebilir, veriler depolanabilir. Örneğin internet, elektrik ve doğalgaz altyapıları gibi kritik sistemler bir çizge modeliyle temsil edilebilmektedir. Çizge olarak modellenen nesneler arasındaki ilişkiler üzerinde analiz yapmak ilişkileri anlaşılır ve yalın hale getirmek için çizgelerin düzlemsel gösterimleri büyük önem taşımaktadır.

Bu tez çalışmasında, karmaşık çizgeleri görüntü işleme algoritmaları kullanarak analiz etmek ve analiz sonucunda kolay herkesin kavrayabileceği bir düzlemsel çizgenin düzlemsel görünümünü sağlamak hedeflenmiştir. OpenCV kütüphanesi çizgelerin tepe noktalarını belirleme konusunda başarılı sonuçlar vermektedir fakat ayrıtları belirleme konusunda yeterli olmadığı gözlenmiştir. Bu amaç için görüntü işleme algoritmalarının hatalarının bu çalışma özelinde giderilmesi ve optimize edilmesi için bir çalışma yapılmıştır. Matplotlib gibi çizge görselleştirme yapılabilen kütüphaneler graphml gibi veri formatlarını görüntü haline getirmek konusunda başarılıdır fakat Png veri formatındaki dosyayı işlemek için uygun değildir. Tez çalışmasının ikinci fazında optimize edilen ayrıt ve tepe noktalarının, matplotlib kütüphanesinin işleyebileceği uygun bir veri formatı getirilmesi için çalışılmıştır. Çalışmanın sonunda sisteme iletilen resim dosyasının işlenerek çizge modelinin çıkarılması ve sonrasında eğer çizge düzlemsel çizge ise çizgenin düzlemsel görünümünün elde edilmesi amaçlanmıştır.

Anahtar: Çizge, OpenCV, Düzlemsel Çizge, Görüntü İşleme, Graphml

ABSTRACT

OPTAINING PLANAR VIEW OF GRAPHS

ÖNER, Uğur

MSc in Computer Engineering

Supervisor: Prof. Dr. Vecdi AYTAÇ

February 2023, 60 pages

It is possible to model all objects in the universe and their relationships with graphs. Analyzes can be made using these models, scientific calculations can be solved faster, and data can be stored. For example, critical systems such as internet, electricity and natural gas infrastructures can be represented with a graph model. The planar representation of the graphs is of great importance in order to analyze the relations between the objects modeled as graphs and to make the relations understandable and simple.

In this thesis, it is aimed to analyze complex graphs using image processing algorithms and to provide a planar view of a planar graph that can be easily understood by everyone as a result of the analysis. The OpenCV library gives successful results in determining the vertices of the graphs, but it has been observed that it is not sufficient to determine the edges. For this purpose, a study has been carried out to eliminate and optimize the errors of image processing algorithms in this study. Graph visualization libraries such as matplotlib are good at rendering data formats such as graphml, but are not suitable for processing files in Png data format. In the second phase of the thesis work, the optimized edges and vertices have been studied to bring a suitable data format that the matplotlib library can handle. At the end of the study, it is aimed to extract the graph model by processing the image file transmitted to the system and then to obtain the planar view of the graph if the graph is a planar graph.

Keywords: Graph, OpenCV, Planar Graph, Image Processing, Graphml

ÖNSÖZ

Çizge teorisi tepeler ve ayrıtlar olarak nesneleri ve aralarındaki ilişkileri modelleyerek hayatın her alanında modelleme yapabilme olanağı sunan geniş yelpazeli bir bilimsel inceleme alanıdır. Çizgeleri görsel olarak veya simgeler ve belirli gösterilimlerle tutmak mümkündür. İnsan doğası görsel olarak tutulan verileri, belirli gösterilimlerle tutulan verilere göre daha kolay ve hızlı kavramaktadır. Görüntü işleme yöntemleri ve görüntüden anlamlı veriler çıkarma günden güne artarak ilerlemektedir. Bu tez çalışmasının ilk aşamasında, görsel olarak verilen bir çizge dosyasını bilgisayarın anlayacağı formata dönüştürmek amacıyla çalışılmıştır.

Düzlemsel çizgeler birçok kritik alanda kullanım senaryosu bulunan önemli bir çizge türüdür. Bu tez çalışmasının ikinci kısmında görüntü işleme yöntemiyle analiz edilen çizgenin düzlemselliğinin kontrol edilmesi düzlemsel görünümünü insanların daha kolay kavrayabileceği görüntü formunda temsil edilmesi hedeflenmiştir.

İZMİR

09/02/2023

Uğur Öner

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (DEVAM)

ŞEKİLLER DİZİNİ

Sekiller **Sayfalar**

TABLOLAR DİZİNİ

KISALTMALAR DİZİNİ

Kısaltma	Açıklama
NPM	Node Paket Yöneticisi(Node Package Manager)
IDE	Tümleşik Geliştirme Ortamı (Integrated Development Enviroment)
HTTP	Üstmetin Transfer Protokolü (Hypertext Transfer Protocol)
NoSQL	Sadece Yapılandırılmış Sorgu Dili Değil(Not Only SQL)
2B	İki Boyutlu (Two Dimensional)
PNG	Taşınabilir Ağ Çizgeiği (Portable Network Graphic)
OpenCV	Açık Bilgisayar Görüşü (Open Computer Vision)
3B	Üç Boyutlu(Three Dimensional)
API	Uygulama Programlama Arayüzü
PCB	Baskılı Devre Kartı (Printed Circuit Board)
XML	Genişletilmiş İşaretleme Dili (Extended Markup Language)
EJS	Gömülü Javacript (Embedded Javascript)

1. DÜZLEMSEL ÇİZGELER VE GÖRÜNTÜ İŞLEME

Çizge teorisi, matematik, fizik bilgisayar bilimleri, endüstriyel üretim haberleşme teknolojisi gibi çok geniş yelpazede ilişki ağı bulunan bir kuramdır. Çizgeler, birçok nesnenin birbirleriyle nasıl ilişkide olduğunu açıklamaya yardımcı olan matematiksel yapılardır. Örneğin, bir çizge, bir ağın yapısını gösterebilir ve bu ağda verilerin nasıl aktarıldığını açıklayabilir. Çizge teorisi, ayrıca birçok farklı matematiksel yapıyı inceleyebilme yeteneğine sahip olduğundan, birçok farklı matematiksel problemi çözmeye yardımcı olabilir. Bu nedenle, çizge teorisi özellikle, matematik ve fizik gibi alanlarda önemli bir konudur ve bu alanlarda çalışan bilim insanları tarafından sıklıkla kullanılır**.**

Çizge teorisi, birçok farklı alanda kullanılabilir. Bilgisayar ağlarının yapısını verilerin nasıl aktarıldığını anlamaya, birçok bilgisayar programının nasıl çalıştığını anlamaya, işletmelerin ürünlerinin nasıl dağıtıldığını ve nasıl pazarlandığını anlamaya, ekonomik sistemlerin nasıl çalıştığını anlamaya yardımcı olur. Ayrıca Çizge teorisi, sosyal bilimlerde de kullanılabilir. Örneğin, bir çizge, sosyal ağların yapısını gösterebilir ve bu ağlardaki ilişkileri anlamak için faydalı olabilir. Ayrıca, bir çizge, bir şirketin organizasyon yapısını gösterebilir ve bu şirketin nasıl yönetildiğini anlamaya yardımcı olabilir.

Düzlemsel çizge 2B bir platformda tepeler arasında ayrıtları kesişmeden gösterilebilen çizgelerdir. Bu, çizgedeki herhangi iki ayrıtın birbirleriyle tek bir noktada kesişmemesi demektir. Bu özellik, düzlemsel çizgeleri diğer tür çizgelerden ayıran önemli bir özelliktir.

Düzlemsel çizgeler, genellikle matematik ve bilgisayar bilimlerinde kullanılır ve özellikle çizge teorisi ve algoritma çalışmalarında önemli rol oynamaktadır. Düzlemsel çizgeler, bilgisayar grafikleri çalışmalarında da kullanılır ve özellikle, 3B nesnelerin 2B ekranlarda görüntülenmesi için kullanılır. Haritalar da düzlemsel çizgeler gibi düşünülebilir ve haritalarda, yerler arasındaki mesafeler gibi bilgileri gösteren ayrıtlar kullanılır.

Gündelik yaşamda pek çok alanda düzlemsel çizgeyle karşılaşmak mümkündür.

- İletişim ağları: Örneğin, bir bilgisayar ağında bilgisayarlar arasındaki bağlantıları gösteren bir çizge oluşturulabilir ve bu da bir düzlemsel çizge gibi düşünülebilir.
- Elektrik şebekeleri: Elektrik şebekelerinde, transformatörler ve elektrik dağıtım hatları arasındaki bağlantılar da bir düzlemsel çizge gibi düşünülebilir.
- Öğrenme haritaları: Öğrenme haritaları, bir konuyu anlamaya yardımcı olan bir araçtır ve bu haritalar da bir düzlemsel çizge gibi düşünülebilir. Öğrenme haritalarında, farklı konular arasındaki bağlantıları gösteren ayrıtlar kullanılır.

Ayrıca düzlemsel çizgeler endüstriyel sistemler fabrikalar ve iş hayatında da büyük öneme sahiptir.

- Fabrikalarda, düzlemsel çizgeler genellikle üretim sürecinde farklı adımları ve bu adımlar arasındaki ilişkiyi göstermek için kullanılır. Örneğin, bir otomobil üretim tesisinde, araçların nasıl üretildiğini göstermek için bir düzlemsel çizge kullanılabilir. Bu çizgede, her bir adım bir tepe olacak ve adımlar arasındaki ilişki ayrıtlar aracılığıyla gösterilecektir.
- Düzlemsel çizgeler, ayrıca fabrikalarda kullanılan makine ve ekipmanların nasıl birbirleriyle ilişkili olduğunu göstermek için de kullanılabilir. Örneğin, bir metal üretim tesisinde, farklı makine ve ekipmanların nasıl birbirlerine bağlandığını göstermek için bir düzlemsel çizge kullanılabilir. Bu çizgede, her bir makine ve ekipman bir tepe olacak ve makine ve ekipmanlar arasındaki ilişkiler ayrıtlar aracılığıyla gösterilecektir.
- Düzlemsel çizgeler, ayrıca fabrikalarda üretim sürecinde kullanılan malzemelerin nasıl birbirleriyle ilişkili olduğunu göstermek için de kullanılabilir. Örneğin, bir gıda üretim tesisinde, farklı malzemelerin nasıl birbirlerine dönüştürüldüğünü göstermek için bir düzlemsel çizge kullanılabilir. Bu çizgede, her bir malzeme bir nokta olacak ve malzemeler arasındaki ilişki çizgiler aracılığıyla gösterilecektir

Düzlemsel çizgeler endüstriyel üretim tesislerinden enerji ve iletişim altyapılarına şirket yönetim ve iş akış süreçlerinden öğrenme yol haritalarına kadar çok farklı alanlarda kullanılmaktadır. Düzlemsel çizgelerdeki problem ise şudur: Pek çok çizge düzlemsel olabilir ama bir çizgenin birden çok gösterim

yöntemi olduğu için düzlemsel çizge düzlemsel olmayan bir şekilde görselleştirilmiş olabilmektedir. Fakat üretim tesisi veya enerji altyapısı gibi bir sistemin gereksinimlerini bir çizgeye aktardığımızda onu hayata geçirmeden önce düzlemsel olmasa bile düzlemsel görünümü varsa düzlemsel hale getirip uygulamak en az maliyetli çözüm olacaktır. Bu çalışma kapsamında bunu gerçekleştirmek için görüntü yoluyla hızlıca alınan bir çizgenin görüntü işleme algoritmaları kullanarak analizi ve girilen çizge düzlemselse düzlemsel görünümü elde etmek amaçlandı.

Görüntü işleme, görüntülerin analizi, değiştirilmesi ve üretilmesi için kullanılan bir bilgi teknolojisi alanıdır. Bu saha, görüntülerin elemanlarının tanımlanması, özelliklerinin belirlenmesi, ölçülmesi ve değiştirilmesi gibi işlemleri içerir. Görüntü işleme, birçok farklı alanda kullanılır, ancak en yaygın kullanım alanları medikal görüntüleme, bilgisayar grafiği, insansız hava araçları ve otomatikleştirilmiş kontrol gibi endüstriyel uygulamalardır. Medikal görüntüleme: Görüntü işleme, radyografik görüntüler, tomografik görüntüler ve MR görüntüleri gibi tıbbi görüntülerin incelenmesi ve analizi için kullanılır. Bilgisayar grafiği: Görüntü işleme, 3B modellerin üretilmesi ve görselleştirilmesi için kullanılır. İnsansız hava araçları: Görüntü işleme, insansız hava araçlarının yer tespiti, hava koşullarının takibi ve hava koşullarına göre yol bulma gibi görevleri yerine getirir. Otomatikleştirilmiş kontrol: Görüntü işleme, otomatikleştirilmiş kontrol sistemlerinde kullanılır. Bu sistemler, üretim hatlarında üretilen ürünlerin kalitesini kontrol etmek için kullanılır. Özellikle otomatikleşmiş kontrol hatları ve insansız hava araçları gibi alınan görüntünün anlık incelenmesi gereken alanlarda kontrol görüntülerin bir çizge modeline aktarılıp analiz edilmesi daha etkili olacaktır. Bu kapsamda alınan görüntünün çizge formatına çevrilmesi için ve sonrasında üretim hattı gibi düzlemsel çizgelerin kullanıldığı sistemlerde alınan ve işlenen resim dosyasının düzlemsel olarak tekrar görselleştirilmesi için bu tez çalışmasının yapılması planlandı.

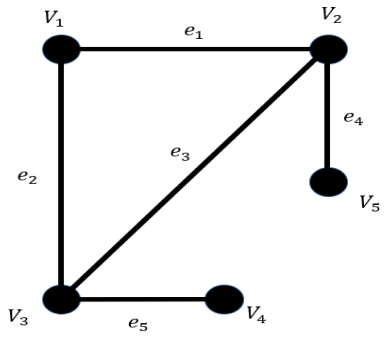
2. ÇİZGE TEORİSİ

2.1 Çizge Kavramları

Bu bölümde, çizgeler hakkında genel teorileri ve bunlarla ilgili temel kavramlar ve teoremler üzerinde durulacaktır.

Tanım 2.1: V; boş olmayan, elemanları tepe (vertex) olarak adlandırılan $V = \{v_1,\ v_2,\ v_3, \dots, v_n\}$ şeklinde bir küme olmak üzere bu noktaları veya noktalarının kendisini birleştiren $E = \{e = v_i v_j : v_i,\ v_j \in V\}$ şeklinde ayrıtlar kümesinden oluşan (V, E) ikili yapısına çizge denir (Büyükköse ve Gök 2019).

Tanım 2.2: $G = (V, E)$ bir çizge olmak üzere, G'nin herhangi bir v_1 ve v_2 tepeleri arasında en az bir ayrıt bulunuyorsa v_1 ve v_2 tepelerine komşudur denir (Büyükköse ve Gök 2019). Benzer şekilde herhangi bir çizgedeki ortak bir tepeye sahip olan ayrıtlar, komşu ayrıtlar olarak adlandırılır (Büyükköse ve Gök 2019).



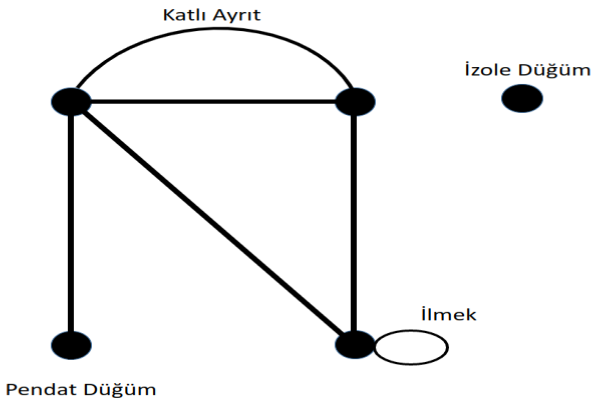
Şekil 2.1 (V,E) Çizgesi

Şekil 2.1'de verilen çizgede; v_1 ile v_2, v_1 ile v_3, v_2 ile v_3, v_2 ile v_5 ve v_3 ile v_4 komşu tepelerdir. Benzer şekilde e_1 ile e_2, e_1 ile e_3, e_1 ile e_4, e_2 ile e_3, e_2 ile e_5, e_3 ile e_4 ve e_3 ile e_5 komşu ayrıtlardır.

Tanım 2.3: Bir G çizgesinde $V = \{v_1,\ v_2,\ v_3, \dots, v_n\}$ tepeler kümesi olmak üzere V kümesinin eleman sayısına, G çizgesinin mertebesi denir ve $|V(G)|$ ile gösterilir. Bir G çizgesine $E = \{e = v_1 v_2 : v_1,\ v_2 \in V\}$ ayrıt kümesi olmak üzere E kümesinin eleman sayısına, G çizgesinin boyutu denir ve $|E(G)|$ şeklinde gösterilir.

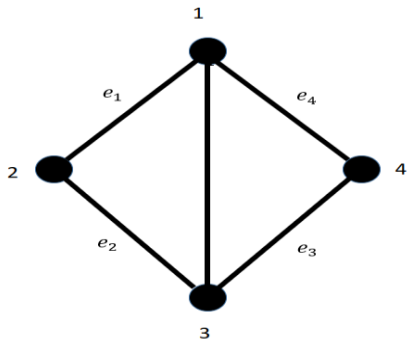
Tanım 2.4: v_1 ve v_2 bir G çizgesinde herhangi iki tepe olsun. v_1 ve v_2 tepelerini birleştiren birden fazla sayıda ayrıt varsa bu ayrıtlar, katlı ayrıt veya paralel ayrıt olarak adlandırılır. Bu çizgedeki başlangıç ve bitiş noktası aynı olan bir ayrıt, bir döngü olarak adlandırılır (Büyükköse ve Gök 2019).

Tanım 2.5: v, bir G çizgesinin herhangi bir tepe olsun. Buna göre v'nin derecesi, v'ye bağlı ayrıtların sayısıdır ve v'nin derecisini göstermek için d_v sembolü kullanılır. Çizgede; en küçük değerli dereceye sahip tepeye minimum dereceli tepe denir ve bu tepenin derecesini göstermek için $\delta(G)$ sembolü kullanılır, en büyük değerli dereceye sahip tepeye ise maksimum dereceli tepe denir ve bunu göstermek için $\Delta(G)$ sembolü kullanılır. Çizgenin döngü içeren bir tepesinin derecesine bu durum iki (+2) olarak eklenir. Eğer $d_v = 0$ ise v'ye izole tepe, $d_v = 1$ ise v'ye ise pendant (uç tepe) adı verilir (Büyükköse ve Gök 2019).



Şekil 2.2 Bir Çizge Örneği (Şener 2021)

Tanım 2.6: Bir çizgedeki komşu tepeler ve ayrıtlardan oluşan, sayıda elemana sahip diziye bir yürüme denir ve yürümeyi ifade etmek için W sembolü kullanılır. $W = 1e_1 2e_2 3 \dots ke_k(k+1)$ biçimindeki dizi, bir yürümeyi ifade eder ve bu yürümenin uzunluğu, dizide yer alan ayrıt sayısı ile belirlenir. Eğer bir yürümedeki her bir ayrıt ve her bir tepe, bir kez yer alıyorsa bu yürümeye yol adı verilir ve bir yolu göstermek için P sembolü kullanılır. Eğer bir yol, aynı tepeden başlayıp yine aynı tepede sonlanıyorsa devre adını alır. n sayıda tepeye sahip olan bir devreyi göstermek için C_n sembolü kullanılır (Büyükköse ve Gök 2019).

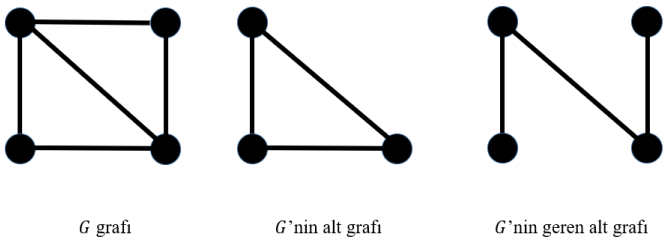


Şekil 2.3 Bir Çizge Örneği (Şener 2021)

Şekil 2.3 de verilen çizgede; $1e_1 2e_2 3e_3 4e_4 1e_5 3e_2 2$ bir yürüme, $1e_1 2e_2 3e_3 4$ bir yol ve $1e_1 2e_2 3e_5 1$ bir devredir. (Şener 2021).

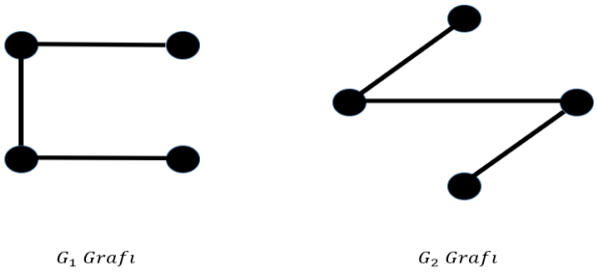
Tanım 2.7: Bir G çizgesinin bazı tepe ya da ayrıtlarının silinmesiyle oluşan çizgeye, bu G çizgesinin alt çizgesi denir. Yani, S ve G herhangi iki çizge olmak üzere $V(S) \subseteq \mathrm{V(G)}$ ve $E(S) \subseteq \mathrm{E(G)}$ oluyorsa S'ye G'ni bir alt çizgesi denir.

G herhangi bir çizge olmak üzere, bu çizgenin tepelerinin tamamını ve bazı ayrıtlarını kapsayan bir alt çizgeye, G çizgesinin geren alt çizgesi (spanning subgraph) adı verilir (Büyükköse ve Gök 2019). Yani, S ve G herhangi iki çizge olsun. Eğer $V(S) = V(G)$ ve $E(S) \subset E(G)$ oluyorsa S çizgesine G çizgesinin bir geren alt çizgesi denir.



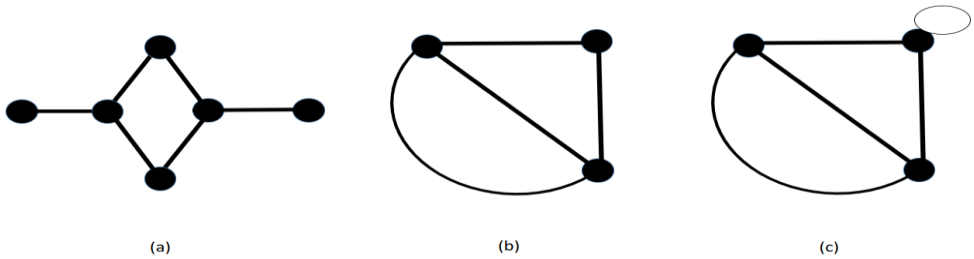
Şekil 2.4 Bir Çizge ve Ona Ait Bir Altçizge ile Geren Altçizge Örneği (Şener 2021)

Tanım 2.8: $G_1 = (V, E)$ ve $G_2 = (V', E')$ iki çizge olsun. Eğer $ab \in \mathrm{E}$ biçimindeki her $a, b \in \mathrm{V}$ tepe çifti için $f(a)f(b) \in E'$ olacak şekilde birebir ve örten bir $f: V_1 \rightarrow V_2$ dönüşümü mevcutsa bu çizgelere izomorfik çizgeler denir ve bu durumu belirtmek için $G_1 \cong G_2$ gösterimi kullanılır (Büyükköse ve Gök 2019).



Şekil 2.5 Bir İzomorf Çizge Örneği (Şener 2021)

Tanım 2.9: Bir çizgede yer alan herhangi iki tepe arasında bir ve yalnız bir adet ayrıt varsa ve hiçbir döngü yoksa bu çizge, basit çizge adını alır (Şekil 2.6 (a)) (Büyükköse ve Gök 2019). Bir çizgenin ayrıtlarına bir yön tayin edilebilir.

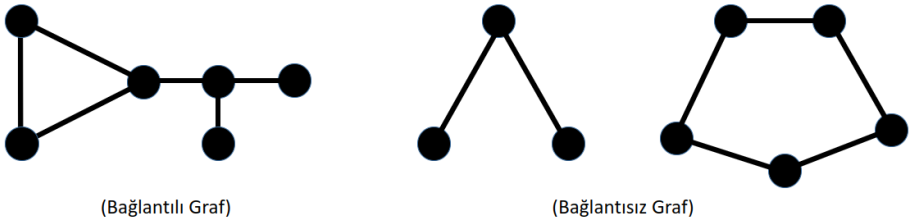


Şekil 2.6 Basit Çizge, Çoklu ve Preudo Çizge Örnekleri

Tanım 2.10: Bir çizgede yer alan herhangi iki tepe arasında birden çok sayıda ayrıt varsa diğer bir ifadeyle paralel ayrıtlar mevcutsa bu çizge, çoklu çizge adını alır (Şekil 2.6 (b)) (Büyükköse ve Gök 2019).

Tanım 2.11: Bir çizgede paralel ayrıtlar ve döngüler mevcutsa bu çizge, pseudo çizge adını alır (Şekil 2.6 (c)) (Büyükköse ve Gök 2019).

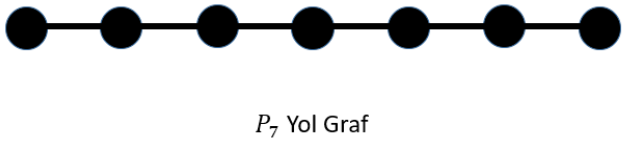
Tanım 2.12: Bir çizgenin keyfi olarak alınan herhangi iki tepesi arasında daima bir yol mevcutsa bu çizgeye bağlantılı çizge adı verilir. Aksi durumda bu çizgeye bağlantısız çizge denir (Büyükköse ve Gök 2019).



Şekil 2.7 Bağlantılı ve Bağlantısız Çizge Örnekleri (Şener 2021)

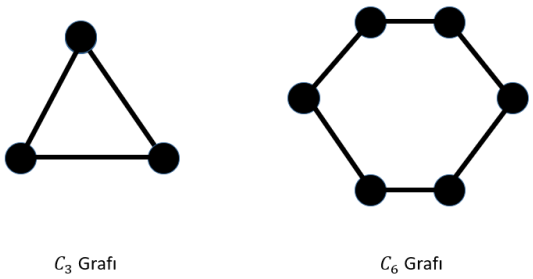
Tanım 2.13: Sadece izole tepe içeren çizgeye boş çizge denir ve boş çizgenin ayrıtlar kümesi boş kümedir (Büyükköse ve Gök 2019).

Tanım 2.14: Başlangıç ve bitiş tepelerinin derecesi 1, diğer tepelerinin derecesi 2 olan çizgeye yol çizge denir. n tepeli bir yol çizge, $n-1$ sayıda ayrıta sahiptir ve P_n ile gösterilir.



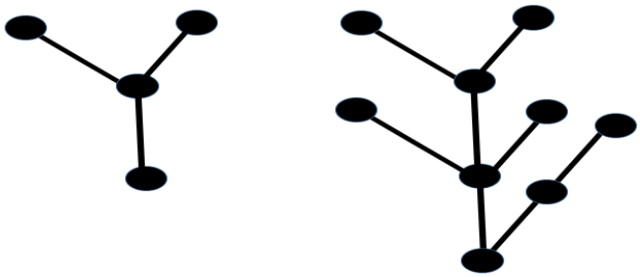
Şekil 2.8 Bir Yol Çizge Örneği (Şener 2021)

Tanım 2.15: Tepe sayısı üç ve daha fazla olan, tek bir döngüden oluşan çizgeye devre çizge denir. Böyle bir çizgede; herhangi bir tepeden, kendisine komşu olan iki tepeye giden birer ayrıt vardır ve böylece her bir tepenin derecesi 2'dir. n noktalı bir çevre çizge, n ayrıta sahiptir ve C_n ile gösterilir.



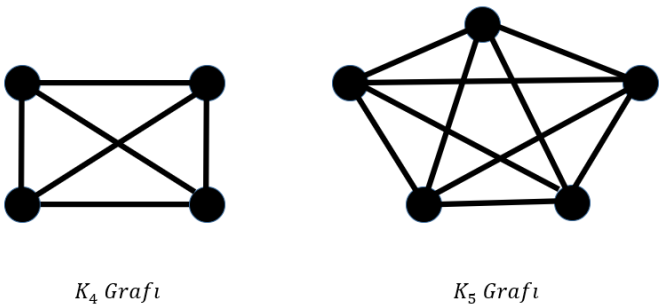
Şekil 2.9 Çevre Çizge Örnekleri (Şener 2021)

Tanım 2.16: İçinde devre bulundurmayan ve ayrıt sayısı tepe sayısının bir eksiği olan bağlantılı çizgelere ağaç adı verilir ve n sayıda tepeye sahip bir ağaç çizgeyi göstermek için $\overline{T_n}$ sembolü kullanılır (Büyükköse ve Gök 2019).



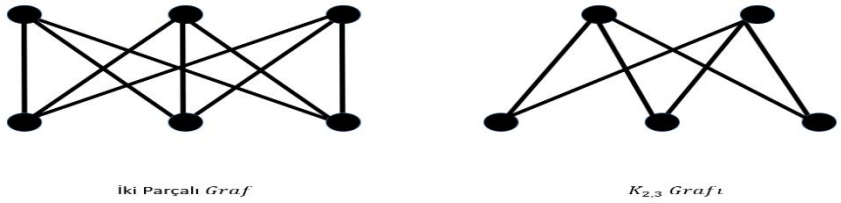
Şekil 2.10 Ağaç Çizge Örnekleri (Şener 2021)

Tanım 2.17: Basit bir çizgenin her tepe çifti arasında bir ayrıt bulunuyorsa böyle çizgelere tam çizge adı verilir ve n sayıda tepeye sahip bir tam çizgeyi göstermek için K_n sembolü kullanılır (Büyükköse ve Gök 2019).



Şekil 2.11 Tam Çizge Örnekleri (Şener 2021)

Tanım 2.18: Tepe kümesi iki ayrık alt kümeye ayrıştırılabilen ve ayrıtları, sadece bu alt kümelerin elemanları olan tepeler arasında oluşan çizgelere iki parçalı çizge adı verilir. İki parçalı bir çizgede; D ve F ayrık tepe kümeleri olmak üzere D'nin her elemanı F'nin her elemanı ile birleştiriliyorsa bu çizge, iki parçalı tam çizge adını alır ve $s(D) = p$ ve $s(F) = t$ olmak üzere iki parçalı tam çizgeyi göstermek için $K_{p,t}$ sembolü kullanılır (Büyükköse ve Gök 2019).



Şekil 2.12 İki Parçalı Çizge ve İki Parçalı Tam Çizge Örneği (Şener 2021)

Tanım 2.19: (Bondy and Murty 2008) Bir G çizgesi verilsin. G'nin karşılıklı olarak birbirine komşu olan tepelerinin kümesine klik kümesi denir. Bir G grafının klik kümelerindeki maksimum eleman sayısı, grafın klik sayısı olarak adlandırılır. Bu değer (G) ile gösterilir ve matematiksel olarak aşağıdaki şekilde ifade edilir:

$(G) = m\{\ |V_i| : V_i$ klik küme$\}$.

Tanım 2.20 Bükümlü tepeler, önceki yıllarda üzerinde derinlemesine çalışmalar yürütülen bir tepe ailesidir. 2001 yılında Hoste and Shanahan, bükümlü düğümlerin topolojik özelliklerini araştırdı (Hoste and Shanahan 2001).

2.2 Düzlemsel Çizge

Tanım 2.21: (V, E) kümesi olmak üzere: V'nin elemanları "tepeler" ve E'nin elemanları "ayrıtlar" olarak belirlenirse:

- Her ayrıt ve tepeler arasında bir yay;
- Farklı tepeler farklı uç nokta kümelerine sahip;
- Bir tepenin içi, bir diğer ayrıtın herhangi bir tepesini ve noktasını içermiyor ise, (V, E) çiftine bir "düzlemsel çizge" denir.
- $V \subseteq R^2$

Tanım 2.22: G, bir düzlemsel çizge olsun. $R^2 \backslash G$'nin bölgelerine, G'nin "bölge"leri denir. Bir çizgenin bölgeleri, R^2'nin açık kümeleridir. Bu sebeple, herhangi bir yüzün sınırı G'nin alt kümesidir. G çizgesi sınırlı olduğundan, yani yeterince büyük bir D diskinin içinde uzandığı için; bölgelerinden tek bir tanesi sınırsızdır ve o da $R^2 \backslash D$'yi içeren bölgedir. Bu bölge, G'nin "dış bölgesi", diğer bölgeler ise G'nin "iç bölgeleri" olarak adlandırılır. G'nin bölgelerinin kümesi $F(G)$ ile gösterilmektedir.

Tepe noktası, 3-boyutlu uzay R^3'de kendi arakesit noktaları olmaksızın herhangi bir basit kapalı eğridir ve halka, kesişmeyen basit kapalı eğrilerin herhangi bir birleşimidir. Bir çizge, noktalardan ve doğru parçalarından oluşan bir şekil olarak düşünülebilir (topolojik olarak buna 1-kompleks denir) (Bollobas 1998). Her bir tepeye karşılık gelen bir çizge tayin edilebilir ve böyle bir çizgeye bir tepe çizgesi denir. Tepe diyagramlarının bir uygun temsili 1877'de Peter Tait tarafından tanıtıldı. Herhangi bir tepe diyagramı; tepe noktaları, kesişme noktaları olan ve ayrıtları birbirini takip eden kesişmeler arasındaki yollar olan bir düzlemsel çizgesini tanımlar. Bu düzlemsel çizgenin tam olarak bir bölgesi sınırsızdır; diğerlerinin her biri 2 boyutlu bir diske homomorfiktir.

Çizge teorisiyle ilişkili herhangi bir çalışma için, düzlemsel çizgeler ele alınır ve çizge sabitleri için gereken formüller, onların üzerinde hesaplanır veya ispatlanır. Bu durum çizge teorisindeki bazı karmaşık cebirsel topolojik ispatların yerine çizge teorik ispatlarının alınabileceğini göstermektedir (Murasugi 1989).

Düzlemsel çizgelerin bölgeleri ve alt çizgeleri arasında aşikâr bir bağlantı vardır:

Lemma 2.1: G, bir düzlemsel çizge, $f \in F(G)$ bir yüz ve $H \subseteq \mathrm{G}$ bir alt çizge olsun.

- H alt çizgesinin f'yi içeren bir f' yüzü vardır.
- Eğer f'nin sınırı H çizgeı içinde kalıyor ise, $f' = f$'dir.

İspat:

Açık şekilde, $f \subseteq R^2 \setminus G$ bölgesindeki noktalar, $R^2 \backslash H$ içinde de denktirler. ($x, y \in f \Leftrightarrow x$ ve y noktaları f içinde kalan bir yay ile birleştirilebilirler. $\Rightarrow H \subseteq G$ olduğunda, x ve y noktaları, $R^2 \backslash H$ içinde kalan bir yay ile birleştirilebilir. $\Rightarrow x$ ve y, H'nin aynı yüzüne aittirler), f', $R^2 \backslash H$'nin, f yüzündeki noktaları içeren denklik sınıfı olsun.

Karşıt tersini ispat edelim. $f' \neq f$ olsun. Bu durumda, $f \nsubseteq f'$ olduğundan, $f' \nsubseteq f$, yani $f' \backslash f \neq \emptyset$'dir. Bu sebeple, f ve $f \backslash f'$ arasındaki her yay, f'nin sınırı X'i keser. Bu nedenle, f' içindeki böyle bir yay üzerinde bulunan X'in sınırına ait nokta(lar), H çizgesine ait olamaz (çünkü $R^2 \backslash H$'nin içindedir(ler)). Bu yüzden, $X \nsubseteq H$'dir.

Lemma 2.2: G bir düzlemsel çizge ve e, G'nin bir ayrıtı olsun. Bu durumda:

- Eğer X, G'nin bir yüzünün sınırı ise, ya $e \subseteq$ X'dir ya da $X \cap \dot{e} = \emptyset$'dir.
- Eğer e, bir $C \subseteq$ G çevriminin üzerinde ise bu durum da e, G'nin tam olarak iki yüzünün sınırındadır ve bu iki yüz, C'nin farklı bölgelerinin içinde uzanmaktadır.
- Eğer e ayrıtını içeren hiçbir çevrim yok ise bu durumda e, G'nin tam olarak bir yüzün sınırındadır.

Sonuç 2.1: Bir yüzün sınırı, her zaman bir alt çizgenin nokta kümesidir.

Tanım 2.23: Nokta kümesi, bir f bölgesinin sınırı olan G'nin alt çizgesine, f'nin "sınırı" adı verilir ve bu alt çizge f'yi "sınırlandırıyor" denir ve $G[f]$ biçiminde gösterilir. Bir yüz, sınırının ayrıtları ve tepeleri ile "bağlıdır" denir.

Teorem 2.1: Bir düzlemsel ormanının tam olarak tek bir bölgesi vardır.

Tek bir istisnayla, bir düzlemsel çizgenin farklı bölgeleri farklı sınırlara sahiptirler:

Lemma 2.3: Eğer bir düzlemsel çizgenin sınırları aynı olan iki yüzü var ise, bu çizge bir çevrimdir.

İspat: G, bir düzlemsel çizge ve $H \subseteq \mathrm{G}, \mathrm{G}$'nin birbirinden farklı f_1 ve f_2 bölgelerinin sınırı olsun. f_1 ve f_2, H'nin de bölgeleridir; bu sebeple, bir önceki teoremden, H bir C çevrimi içerir. (Eğer H hiçbir çevrim içermeseydi, bir orman olurdu. Fakat önceki teoremden, bir ormanın tek bir yüzü olduğunu biliyoruz.) Bir önceki lemma'nın (ii) kısmından, f_1 ve f_2 bölgeleri C'nin farklı bölgeleri içinde uzanırlar. H alt çizgesinin tamamı f_1 ve f_2 bölgelerinin sınırı olduğundan, $H = C$'dir.

Diyelim ki $H \neq C$, yani $C \subseteq \mathrm{H}$ olsun. Bu durumda, H'nin C devresinde olmayan bir tepe veya bir ayrıtı vardır. Fakat bu durumda, bu tepe veya ayrıt, C'nin iki yüzünden birisinde olacağından ve f_1, f_2 bölgeleri C'nin farklı bölgeleri içinde bulunduklarından, bu tepe veya ayrıt, kendisiyle aynı tarafta olmayan yüzün sınırı üzerinde olmayacaktır. Ki bu bir çelişki verir. Bu sebeple, $H = C$' dir ve f_1, f_2, C'nin farklı bölgeleridir. C'nin sadece iki yüzü olduğundan, $f_1 \cup C \cup f_2 = R^2$'dir ve bu nedenle, $G = C$'dir.

Teorem 2.2: Bir iki bağlantılı düzlemsel çizgede, her yüz bir çevrim tarafından sınırlandırılır.

Bir üç bağlantılı düzlemsel çizgede, bölge sınırları olan devreleri, tamamen kombiratorik terimlerle tanımlayabiliriz:

Teorem 2.3: Bir üç bağlantılı düzlemsel çizgenin yüz sınırları tam olarak, bu çizgenin ayrılmayan devreleridir.

Tanım 2.25: G, bir düzlemsel çizge olsun. Eğer G çizgesine yeni bir ayrıt ekleyerek, $V(G') = V(G)$ olmak şartıyla (yani yeni bir tepe eklemeksizin), $G \subseteq G'$ olacak şekilde bir düzlemsel çizge elde edilemiyor ise; G, bir "maksimal (düzlemsel)" çizgedir denir. Eğer G'nin her bölgesi bir üçgen tarafından sınırlandırılmış ise, G çizgesi bir "düzlemsel üçgenleme" olarak adlandırılır.

Teorem 2.4: G, en az üç tepe içeren bir düzlemsel çizge olsun. Bu durumda:G bir maksimal düzlemsel çizgedir, G bir düzlemsel üçgenlemesidir.

Euler teoremi, bir düzlemsel çizgede tepe ve ayrıtları ve yüz sayıları arasındaki ilişkiyi ifade eder. Teorem şöyle söylemektedir: Eğer doğru işaretleri alırsak, bu sayıların toplamı her zaman 2'dir. Bu teorem, diğer yüzeyler için de geçerlidir ve elde edilen toplam, sadece yüzeye bağlı olarak, çizgede bağlı

olmayan sabit bir sayıdır. Ayrıca, farklı yüzeyler için elde edilen sabit sayılar farklıdır. Bu nedenle, bu sabit sayılar ile farklı yüzeyler arasındaki ilişki, yüzeyleri tanımlamak için kullanılabilir.

Teorem 2.5 (Euler Formülü): G, n tepeyi, m ayrıtı ve l yüzü olan bir bağlantılı düzlemsel çizge olsun. Bu durumda: $n - m + l = 2$ eşitliği sağlanır.

İspat: n'yi sabit tutalım ve m üzerinden tümevarım uygulayalım. $m \leq n - 1$ için, G bağlantılı olduğundan $m = n - 1$'dir. Başka bir deyişle G bir ağaçtır. Bu sebeple, bir ağacın tek bir yüzü olduğundan, iddia ispatlanmış olur.

Şimdi $m \geq n$ olsun. Bu durumda, G'nin bir çevrim üzerinde olan bir e ayrıtı vardır. $G' := G - e$ olsun e ayrıtı, G çizgesinin tam olarak iki f_1, f_2 yüzünün sınırı üzerinde bulunduğundan ve $\dot{e}$ içindeki tüm noktalar, $R^2 \backslash G'$ de birbirine denk olduklarından, G' çizgesinin $\dot{e}$ nokta kümesini içeren bir f_e bölgesi vardır.

$F(G) \backslash \{f_1, f_2\} = F(G') \backslash \{f_e\}$ **(*)**

olduğunu gösterelim. Bu durumda, G' çizgesi, G çizgesinden tam olarak bir eksik bölge ve bir eksik ayrıt içerdiğinden, G' çizgesi için tümevarım hipotezi iddianın doğruluğunu ortaya koyar.

(*)'in bir ispatı için, önce $f \in F(G) \backslash \{f_1, f_2\}$ verilsin. Bu alt bölümün ikinci lemmasının (i) kısmından, e bir yüzün sınırına ait veya ait olmadığı bir sınırla içinin kesişimi boş küme olduğundan, f yüzünün sınırı $G[f] \subseteq \mathrm{G} \backslash \dot{e} = G'$ dir. Bu sebeple, bu alt bölümün ilk lemmasının (ii) kısmından, $f \in F(G')$'dir. Açık bir şekilde $f \neq f_e$ olduğundan, (*) eşitliğinde, ($\subseteq$) bağıntısı gösterilmiş olur.

Karşıt olarak, $f' \in F(G') \backslash \{f_e\}$ verilsin. Açık bir şekilde, $f' \neq f_1, f_2$ ve $f' \cap \dot{e} = \emptyset$'dir. (Çünkü e, f_1 ve f_2'nin sınırı üzerindedir ve $\dot{e} \subseteq f_e$'dir.) Bu sebeple, ($f', G' = G - e$'nin bir yüzü ve $f' \cap \dot{e} = \emptyset$ olduğundan,) f' yüzü içindeki herhangi iki nokta, $R^2 \backslash G$ kümesi içindedir ve birbirlerine denktirler. Bu sebeple, G çizgesinin f' yüzünü içeren bir f yüzü vardır. Bununla birlikte, bu alt bölümün ilk lemmasının (i) kısmından, f, G' çizgesinin bir f^n yüzünün içinde uzanır. Bu nedenle, $f' \subseteq f \subseteq f''$ dir. Diğer yandan, f' ve f'', G' çizgesinin iki

yüzü olduklarından, $f' = f = f''$ elde edilir. Bu sebeple, $f' \in F(G)$'dir, yani (*) eşitliğinde, (⊇) bağıntısı da gösterilmiş olur.

Sonuç 2.2: Tepe sayısı $n \geq 3$ olan bir düzlemsel çizge en fazla $3n - 6$ tane ayrıt içerir. n tepe içeren her düzlemsel üçgenlemesinde tam olarak $3n - 6$ tane ayrıt vardır.(Rosen 2012)

İspat: İki önceki teoremden, ikinci iddianın doğruluğunu göstermek yeterlidir. Bir düzlemsel üçgenlemesi G de, her yüzün sınırı tam olarak üç ayrıt içerir ve bu alt bölümünün ikinci lemmasının (ii) kısmından, her ayrıt tam olarak iki yüzün sınırına aittir. Bu sebeple, ayrıt kümesi $\{ef \mid e \subseteq$ G[f]$\}$ olan, $E(G) \cup F(G)$ üzerindeki iki parçalı çizgenin tam olarak $2|E(G)| = 3|F(G)|$ ayrıtı vardır. (Çünkü her bir ayrıt iki yüzün sınırına aittir ve her bir yüzün sınırı bir üçgendir.) Şimdi bu eşitliğe göre, Euler formülünde, l yerine $2m/3$ yazarsak, istenilen sonucu elde ederiz:

$$n - m + \frac{2m}{3} = 2 \Leftrightarrow 3n - 3m = 6 \Leftrightarrow m = 3n - 6$$

Euler formülü, bazı çizgelerin düzlemsel çizge olamayacağını göstermek için kullanılabilir:

Teorem2.6: Bir çizge düzlemseldir ⇔ Çevrim uzayının bir seyrek tabanı vardır. (MacLane 1937)

Çizge teorisinin saklı güzelliklerinden birisi, soyut ve görünüşte sezgisel olmayan iki sonucun örneği, MacLane'in devre uzayındaki üreteç kümeleri ile ilgili bu teoremi ve Tutte'nin bir 3-bağlantılı çizgesinin devre uzayının, ayrılmayan indüke çevrimler tarafından üretildiği ifade eden teoreminin bir araya gelerek, 3−bağlantılı çizgeler için çok somut bir düzlemsellik kriteri ortaya koymasıdır:

Teorem 2.7: Bir 3-bağlantılı çizge düzlemseldir ⇔ Her bir kenarı en fazla (denk olarak: tam) iki ayrılmayan indüke çevrim üzerinde bulunur. (Kelmans 1978)

Tanım 2.26: Eğer bir çizge, diğer bir çizgenin ayrıtlarına derecesi 2 olan düğümler ekleyerek veya çıkararak elde edilebiliyorsa bu iki çizge izomorfiktir. (Kuratowski 1930)

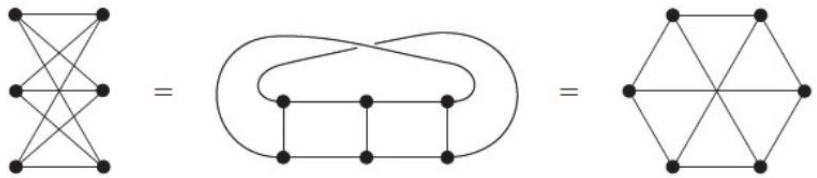
Örnek 2.1: K_5 tam çizgesi bir düzlemsel çizge değildir:

K_5'in ayrıt sayısı $m = \frac{5}{2} = 10$ ve $10 > 3.5 - 6$'dır. Yani K_5'in ayrıt sayısı, önceki sonuçta verilen sınırı aşmaktadır. Bu sebeple K_5 bir düzlemsel çizge olamaz.

Örnek 2.2: $K_{3,3}$ iki parçalı tam çizgesi de bir düzlemsel çizge değildir:

$K_{3,3}$ çizgesi iki parçalı bir tam çizge olarak tanımlanabilir. Ancak, $K_{3,3}$ çizgesi düzlemsel çizge olamaz. Çünkü, $K_{3,3}$ çizgesi iki-bağlantılı olduğu halde, üçgen içermemektedir ve bu nedenle tüm bölgeleri uzunluğu en az 4 olan bir çevrim ile sınırlıdır. Bu nedenle, her ayrıt iki bölge sınırında yer alır ve 2m≥4l olmalıdır. Ancak, $K_{3,3}$ çizgesinde m=9 olsa da 2n-4 >9 olduğundan, $K_{3,3}$ çizgesi düzlemsel çizgesi olamaz.

Yukarıdaki ispatlardan çıkarımla K_5 ve $K_{3,3}$ birer düzlemsel çizge olmadığı için herhangi bir alt çizgesi $K_{3,3}$ çizgesini barındıran çizgeler düzlemsel çizge değildir.

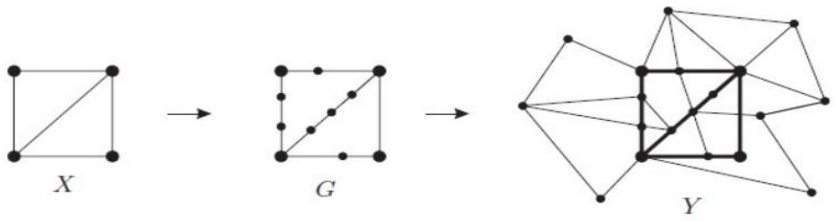


Şekil 2.13 $K_{3,3}$ iki parçalı tam çizgesi (Diestel 2017)

Bu ispatların sonucunda şunu diyebiliriz: Bir düzlemsel çizge, ne K_5 tam çizgesini ve ne de $K_{3,3}$ iki parçalı tam çizgesinin topolojik minörünü alt çizgelerinde barındırmaz. Eğer bir çizgenin alt çizgelerinden en az 1 tanesi K_5 tam çizgesini veya $K_{3,3}$ çizgesini barındırıyorsa düzlemsel değildir.

G; X'in bir alt-bölümlenmesi olsun. G içindeki X'in orijinal tepelerine, "dal tepeleri"; diğer tepelere, "alt-bölümlenme tepeleri" denir. Dikkat edilirse, alt-bölümlenme tepelerinin her birinin derecesi 2; dal tepelerinin dereceleri ise, X deki dereceleri ile ayınıdır. Eğer bir Y çizgesi, X çizgesinin bir alt-bölümlenmesi olan bir G çizgesini bir alt çizgesi olarak içeriyor ise; X; Y'nin bir "topolojik minörüdür" denir.

Çizge G, çizge X'in bir alt bölümlemesi olması durumunda. G çizgesi içinde olan, X çizgesinin orijinal tepelerine "dal tepeleri" denir, diğer tepelere ise "alt bölüm tepeleri" denir. Alt bölüm tepelerinin dereceleri her zaman 2'dir, ancak dal tepelerinin dereceleri X çizgesindeki dereceleri ile aynıdır. Eğer Y çizgesi, X çizgesinin alt bölümü olan G çizgesinin alt çizgesi olarak içeriyorsa, X çizgesi Y çizgesinin bir "topolojik minörü" olarak tanımlanır.



Şekil 2.14 G, X in Bir Alt bölgesi ve X, Y nin bir Top. Minörüdür (Distel 2017)

Aynı şekilde, X çizgesinin x tepelerini, ayrık bağlantılı G_x çizgesiyle ve X çizgenin xy ayrıtlarını, $G_x - G_y$ ayrıtlarının boştan farklı alt kümeleriyle yer değiştirirsek, bir G çizgesi elde edilir. X çizgesi G'nin bir "büzülme minörü" olarak adlandırılır. Eğer bir Y çizgesi, X çizgenin bir büzülme minörü olduğu bir G çizgesini bir alt çizgesi olarak içeriyor ise; X, Y'nin bir "minörüdür" denir ve $X \preccurlyeq Y$ biçiminde gösterilir. Ayrıca G çizgesi, X'in Y içinde bir "modeli" olarak adlandırılır.

Teorem 2.8: (Kuratowski 1930, Wagrer 1937) Bir G çizgesi için aşağıdaki ifadeler birbirine denktir:

(i) G düzlemseldir.
(ii) G, ne K_5 tam çizgesini ve ne de $K_{3,3}$ iki parçalı tam çizgesini bir minör olarak içermez.
(iii) G, ne K_5 tam çizgesini ve ne de $K_{3,3}$ iki parçalı tam çizgesini bir topolojik minör olarak içermez.

Teorem 2.9: R^2 düzlemindeki her J Jordan eğrisinin, tüm düzlemi iki tane ayrık, her ikisi de J eğrisini sınır kabul eden iki açık ve bağlantılı bölgeye ayırdığını söyler. Bunun ayrıntılı analizi Stillwell'de bulunabilir (Stillwell 1993).

Teorem 2.10: Bir çizge düzlemseldir ⇔ Bu çizgenin bağlılık kısmi sıralı kümelerinin boyutu ≤ 3 dür (Schnyder 1989).

3. TEKNOLOJİK ALTYAPI VE UYGULAMA MATERYALLERİ

Bu tez çalışması için Nodejs platformu, Python programlama dili, Javascript programlama dili, Debian tabanlı Linux İşletim Sistemi olan Ubuntu, Anaconda, Spyder IDE, MongoDB veritabanı ve NPM materyalleri kullanıldı.

3.1 Nodejs

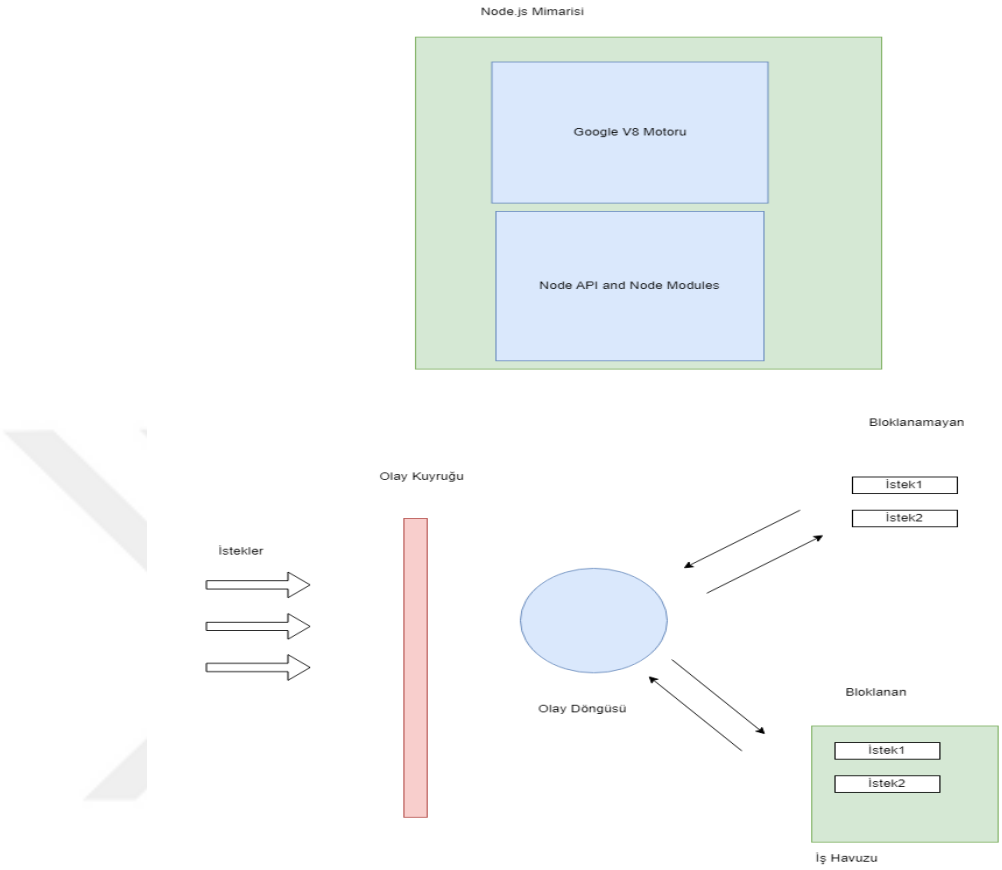
Node.js, JavaScript diliyle yazılmış, çalıştırılabilen bir programlama dilidir. Bu dil, tarayıcıda çalıştırılan JavaScript kodlarının dışında, aynı zamanda server-side (sunucu tarafında) çalıştırılabilir. Özellikle, web sunucuları üzerinde çalışan uygulamaların geliştirilmesinde kullanılmaktadır.

Node.js, JavaScript dili üzerinde çalışan bir çalışma zamanıdır. Bu çalışma zamanı, web tarayıcıları gibi bir JavaScript motoru kullanarak JavaScript kodunu çalıştırır. Google tarayıcıda Javascript çalışması çalıştırılması için V8 motorunu geliştirmiştir. Daha sonar bu V8 motoru tarayıcıdan bağımsız bir şekilde masaüstü bilgisayarlarda da çalışabilecek hale getirildi. Node.js için, V8 motoru ve Node.js modüllerinin birleşimidir diyebiliriz. Node.js, JavaScript kodunu çalıştırmak için Google'ın V8 JavaScript motorunu kullanır. Bu motor, JavaScript kodunu hızlı bir şekilde derler ve çalıştırır. Node.js genel yapısı aşağıdaki Şekil 3.1'deki gibidir. Node.js, ayrıca birçok iş parçacığı çalıştırmak için birçok iş parçacığı çalışma zamanına sahiptir.

Single Threaded Event Loop" mimarisi, birden fazla eşzamanlı isteği işlemek için node.js tarafından kullanılır. Çalışma modeli, geri arama mekanizmasıyla birlikte JavaScript olay tabanlı modeline dayanmaktadır.

Node.js sunucusunu, iş parçacığı havuzu, olay döngüsü ve olay sırası olarak gruplandırabiliriz. İstekleri, kullanıcının web uygulamalarında gerçekleştirdiği, bloklanabilen ve bloklanamayan olmak üzere ikiye ayrılan kullanıcının sunucuya sunucuya gönderdiği mesajlar olarak tanımlanabilir. Olay kuyruğu Node.js sunucusuna gelen kullanıcı isteklerinin olay döngüsüne gönderilmeden önce depolandığı yerdir. Mevcut tüm iş parçacıklarının işlenmek için bulunduğu yere iş parçacığı havuzu denilir. Olay Döngüsü: Asenkron olarak yürütülmesi gereken olay kuyruğundaki olayları kesintisiz takibe alır. Asenkron istekleri koşturur ve sonrasında yapılması gereken iş tamamlandığından bir geri

alma işlevi çağırır. Aşağıdaki Şekil 3.1'de bu döngü yalın bir şekilde açıklanmıştır.



Şekil 3.1 Node.js Mimarisi ve Olayların İşlenme Mekanizması

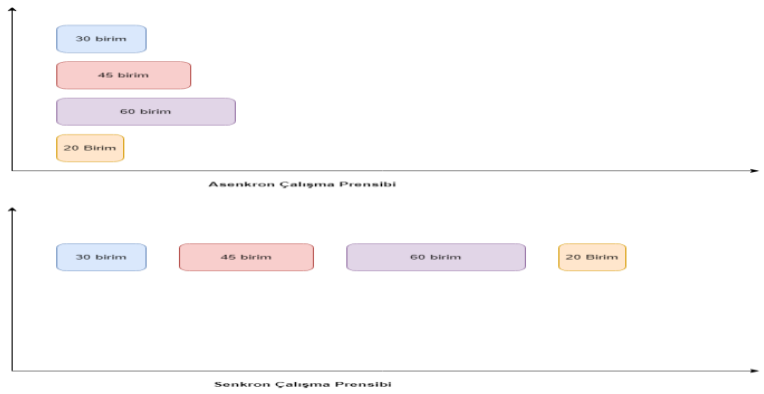
Node.js, JavaScript dilini kullanan çoklu platformlu bir çalışma zamanı (runtime) ortamıdır. Bu ortam, JavaScript kodlarının çalıştırılması için bir çerçeve sağlar ve bu kodların, web tarayıcıları dışında da çalıştırılmasını mümkün kılar. Örneğin, Node.js kullanarak bir sunucuda çalışan bir web sunucusu oluşturulabilir ve bu sunucu, HTTP isteklerini işleyerek, istekleri karşılayan dinamik web sayfaları oluşturabilir.

Node.js'in diğer programlama dillerinden farklı olarak en büyük özelliği, JavaScript dilini kullanmasıdır. Bu sayede, web tarayıcılarında çalışan JavaScript kodları ile benzer şekilde, Node.js ile de sunucu tarafında çalışan uygulamalar geliştirilebilir. Bu sayede, bir web uygulaması için gerekli olan tüm kodlar, aynı dil olarak JavaScript ile yazılabilir ve böylece tek bir dil öğrenmek

suretiyle, hem tarayıcı hem de sunucu tarafında çalışan uygulamalar geliştirilebilir.

Nodejs aşağıda değinilen özellikleri sebebiyle birçok büyük şirket, yazılım geliştiricileri ve proje yöneticisi tarafından tercih edilmektedir

Asenkron ve etkileşimli: Node.js, asenkron bir programlama modelini kullanır. Bu, bir işlem tamamlandıktan sonra bir callback (geri çağırma) fonksiyonu çağıran bir yapıya sahiptir. Bu sayede, işlemler arasında etkileşim kurulabilir ve bir işlem tamamlandıktan sonra diğer işlemlerin çalıştırılması sağlanabilir. Asenkron çalışma Node.js'in çok yönlülüğünü ve verimliliğini sağlamak için kullanılan bir yöntemdir. Asenkron çalışma, bir işlem tamamlanana kadar diğer işlemlerin de yapılmasını sağlar. Bu, Node.js'te çok sayıda kullanıcıyı aynı anda işleyebilme yeteneğini verir ve bu da web uygulamaları için önemlidir, çünkü web uygulamaları sıklıkla çok sayıda kullanıcı tarafından aynı anda kullanılır. Bu sayede, Node.js web uygulamaları çok daha hızlı ve verimli hale getirilebilir. Bunun sonucunda her kullanıcı sistem kaynaklarının imkan sağladığı sürece bekleme yapmadan eşzamanlı olarak sisteme ulaşıp sistemi kullanabilir. Bunu daha elle tutulur bir örnekle şöyle açıklayabiliriz: Klasik programlama dili çalışma mantığında çalışan bir markete gidersek alışverişi tamamlamamız için bizden önce alışverişi tamamlayacak tüm müşterileri beklememiz gereklidir. Asenkron çalışma prensibiyle çalışan Nodejs'de ise boş bulunan diğer kasaya geçerek bizimde aynı anda veya daha hızlı alışverişi tamamlama imkanı mevcuttur. Bunu aşağıdaki şekil ile daha açık aktarabiliriz.



Şekil 3.2 Senkron ve Asenkron Çalışma Prensibi

Şekil 3.2’de görüldüğü gibi senkron çalışma prensibiyle 155 birim sürede tamamlanan iş parçaçıkları asenkron çalışma prensibiyle 60 birim sürede tamamlanmaktadır. Bu aynı anda birden fazla kullanıcının yüksek performansla sisteme erişmesine ve sistemi kullanmasına olanak sağlamaktadır.

Çoklu platform desteği: Node.js, Windows, macOS ve Linux gibi çeşitli işletim sistemlerinde çalıştırılabilir. Bu tez kapsamında debian tabanlı bir linux işletim sistemi olan Ubuntu 20.04 kullanıldı.

Modüler yapı: Node.js, modüler bir yapıya sahiptir ve bu sayede kodlar daha kolay yönetilebilir ve paylaşılabilir. Node.js'te, kütüphaneler "paket" olarak adlandırılır ve bu paketler, npm (Node Package Manager) adı verilen bir paket yöneticisi aracılığıyla yüklenir ve kullanılır. Npm (Node Package Manager): Node.js geliştirme çerçevesine ait bir paket yöneticisidir. npm, Node.js ile yazılmış kod parçacıklarının (paketlerin) yönetimi, paylaşımı ve kullanımını sağlar. npm, Node.js'in kurulu olduğu bilgisayarlarda varsayılan olarak gelir ve Node.js kurulu olmayan bilgisayarlara da ayrıca kurulabilir. Npm, açık kaynaklı bir paket yöneticisidir ve npm üzerinde bulunan paketler de açık kaynaklıdır. npm üzerinde, çeşitli kütüphaneler, araçlar ve uygulamalar mevcuttur ve bu paketler, Node.js ile geliştirilen uygulamalar için çok yararlıdır. npm, Node.js paketlerini yönetmek için kullanılır ve bu paketler, kodların daha kolay yönetilebilmesi, paylaşılabilmesi ve kullanılabilmesi amacıyla oluşturulmuştur. npm üzerinden, çeşitli Node.js paketlerine erişilebilir ve bu paketler, projelerde kullanılmak üzere indirilebilir. Ayrıca, npm üzerinden kendi paketlerinizi de paylaşabilirsiniz. Örneğin, npm üzerinden kullanıcı yönetimi, test otomasyonu, veri tabanı işlemleri gibi işlemleri gerçekleştirebilecek paketler indirilebilir. Npm ile sadece ihtiyaç olan paketler kurulup kullanılır. Bu daha çevik yönetilebilir bir yapının ortaya çıkmasını sağlar. Bu tez çalışması için: Mongoose, Express, Multer, Passport, EJS gibi npm modülleri kullanıldı.

Performans: Node.js, JavaScript dilinin hızlı çalışması sayesinde, yüksek performanslı uygulamalar geliştirilebilir. Özellikle, çok sayıda bağlantıyı aynı anda işleyebilme yeteneğine sahiptir ve bu sayede, örneğin bir chat uygulaması gibi çok sayıda kullanıcıyı destekleyebilir.

Geliştirme araçları: Node.js, geliştirme sürecine yardımcı olacak çeşitli araçlar içerir. Örneğin, Node.js'in kendi içinde yer alan REPL (Read-Eval-Print Loop) adı verilen bir konsol aracı vardır ve bu aracı kullanarak, hızlı bir şekilde

kodlar yazıp çalıştırılabilir. Ayrıca, Node.js'in çeşitli açık kaynaklı paketleri de mevcuttur ve bu paketler, örneğin test otomasyonu veya kod kalitesini artırmaya yönelik olarak kullanılabilir.

Tek dil kullanımı: Node.js ile geliştirilen uygulamalar için tek bir dil olarak JavaScript kullanılır. Bunun sayesinde, hem tarayıcıda çalışan hem de sunucu tarafında çalışan kodlar için aynı dil kullanılabilir ve böylece tek bir dil öğrenilerek, web uygulamalarının tamamı için yeterli hale gelinir.

3.2 Mongo DB

MongoDB, NoSQL veritabanı yönetim sistemidir. NoSQL veritabanı yönetim sistemleri, verilerin anahtar-değer ikilisi, doküman, çizge veya koleksiyon gibi şekillerde saklandığı veri tabanlarıdır. Bu veri tabanları, verileri ilişkisel veri tabanlarındaki gibi tablo ve satır şeklinde saklamazlar ve bu nedenle de "NoSQL" (Not Only SQL) olarak adlandırılırlar.

MongoDB, verileri JSON benzeri bir formatta saklar ve bu verileri sorgulamak için JavaScript benzeri bir dil kullanır. MongoDB, çok yüksek performanslı ve ölçeklenebilir bir veritabanı yönetim sistemidir ve bu nedenle, özellikle web uygulamaları gibi yüksek trafikli uygulamalar için uygun bir seçenektir.

MongoDB, çok sayıda sunucuda dağıtılabilir ve bu sayede verilerin çok daha hızlı işlenmesine yardımcı olur. Ayrıca, MongoDB, verileri sınırsız bir şekilde genişletebilme özelliğine sahiptir ve bu sayede, özellikle büyüyen veri yapıları için uygun bir seçenektir.

MongoDB, açık kaynaklı bir veritabanı yönetim sistemidir ve ücretsiz olarak kullanılabilir. Bu veritabanı yönetim sistemi, çeşitli işletim sistemlerinde çalıştırılabilir ve birçok programlama diline destek verir. Özellikle, Node.js gibi JavaScript tabanlı programlama dilleri ile uyumlu bir şekildedir ve bu nedenle, Node.js ile geliştirilen uygulamalar için uygun bir seçenektir. Bu tez çalışması kapsamında sunucu tarafında Node.js kullanılacağı için veritabanı yönetim sistemi olarak da MongoDB seçilmesi uygun görüldü.

NoSQL veri tabanları herhangi dar kalıplı bir şema formatına ve ilişkisel bir yapıya ihtiyaç duymadan verileri depolayabilen ve yönetebilen sistemlerdir.

Veri tipleri önceden belirlenmiş ve genişletilebilir veri şeması kullanmaktadır. NoSQL veritabanı özellikleri: Sql sorgulama arayüzü gibi gelişmiş bir platform üzerinden sorgulama ve bilinen Sql sorgulama betikleri yerine daha yalın ve anlaşılır bir sorgu sözdizimine sahiptir. Veri depolama işlemleri kolay ve hızlıdır Hem konsol hem de üçüncü parti arayüzler üzerinden işlemleri yapmaya uygundur. Yatay ölçeklendirmeye göre tasarlanmalarından ötürü global ölçekte büyük şirketler tarafından tercih edilmektedir. (Facebook, Amazon, Google gibi her gün terebaytlarca verileri işleyen büyük firmaların NoSQL veri tabanlarını tercih etmesinin ana nedenlerindendir)

MongoDB, NoSQL veritabanı yönetim sistemlerindendir ve NoSQL veritabanı yönetim sistemlerinin sahip olduğu avantajlara sahiptir. Bunlar: Eşzamanlı işleme yeteneği: MongoDB, çok sayıda işlemi aynı anda gerçekleştirme yeteneğine sahiptir ve bu sayede, özellikle yüksek trafikli uygulamalar için uygun bir seçenektir. Ölçeklenebilirlik: MongoDB, verileri çok daha hızlı işleyebilme yeteneğine sahiptir ve bu sayede, veri yapıları ölçeklendirilebilir. Bu sayede, özellikle büyüyen veri yapıları için uygun bir seçenektir. Sınırsız genişleme: MongoDB, verileri sınırsız bir şekilde genişletebilme özelliğine sahiptir ve bu sayede, özellikle büyüyen veri yapıları için uygun bir seçenektir. Hızlı işleme: MongoDB, verileri hızlı bir şekilde işleyebilme yeteneğine sahiptir ve bu sayede, veritabanı işlemlerinin hızı artırılabilir. Doküman tabanlı veri saklama: MongoDB, verileri doküman tabanlı bir şekilde saklar ve bu sayede, verilerin daha kolay yönetilebilmesine yardımcı olur. Çoklu platform desteği: MongoDB, Windows, macOS ve Linux gibi çeşitli işletim sistemlerinde çalıştırılabilir. Açık kaynaklı: MongoDB, açık kaynaklı bir veritabanı yönetim sistemidir ve ücretsiz olarak kullanılabilir.

3.3 Python

Python programlama dilini Guido van Rossum adlı Hollandalı bir yazılım geliştirici tarafından geliştirilmiştir. Van Rossum, Python dilini 1989 yılında ilk kez yayınladı ve dil, günümüzde birçok alanda kullanılmaktadır. Python, açık kaynak kodlu bir dil olduğu için, kullanıcılar tarafından da geliştirilebilmektedir. Python, yüksek seviyeli, güçlü bir programlama dilidir. Python, yazılım geliştirme, veri bilimi, makine öğrenimi gibi birçok alanda kullanılır. Python, okunabilirliği yüksek, anlaşılır ve kolay bir dil olduğu için, özellikle başlangıç seviyesi programcılar tarafından sıklıkla tercih edilir. Python, çeşitli işletim sistemlerinde çalıştırılabilir ve birçok programlama diline destek verir. Python,

konsol üzerinden, betikler olarak yazılıp çalıştırılabilir veya Tümleşik Geliştirme Ortamı (IDE) gibi geliştirme ortamları kullanılarak da kullanılabilir.

Yorumlayıcı dil, yazılım dilinin yorumlanarak çalıştırıldığı dil tipidir. Yorumlayıcı dil, derleyici dillerine göre daha yavaş çalışabilirler ancak derleyici dillerine göre daha kolay program yazılmasına olanak sağlar. Python, yorumlayıcı bir dil olarak tasarlandı ve çalıştırılır. Bu, Python kodunun çalıştırılmadan önce derlenmediği anlamına gelir. Bunun yerine, Python kodu, satır satır yorumlandığında çalıştırılır. Bu, Python kodunun hızlı bir şekilde yazılıp test edilebilmesine olanak sağlar, ancak çalıştırma hızını biraz düşürmektedir.

Python, birçok çeşitli paket ve kütüphanelere sahiptir ve bu paketler ve kütüphaneler, özellikle veri bilimi, makine öğrenimi gibi alanlarda çok yararlıdır. Python, açık kaynaklı bir dil olduğu için, kullanıcılar tarafından geliştirilen paketler de mevcuttur ve bu paketler, Python ile geliştirilen uygulamaların çok daha kolay ve hızlı geliştirilebilmesine yardımcı olur. Python programlama dilinin esnek olması zengin görüntü işleme ve görüntü görselleştirme kütüphanelerinin bulunması bu tez çalışması kapsamında kullanılması için diğer programlama dillerine göre daha fazla ön plana çıkmıştır. Özellikle görüntü işleme için kullanılan openCV kütüphanesi ve görüntü bitlerinin 2B görselleştirilmesi için matplotlib kütüphanesi gelişmiş özelliklere sahip kütüphaneler olarak öne çıkmaktadır.

Python, görüntü işleme için ideal bir programlama dilidir. Python, birçok önemli görüntü işleme kütüphanesine sahiptir ve bu kütüphaneler, görüntüler üzerinde çeşitli işlemler gerçekleştirilebilmesine yardımcı olur.

OpenCV (Open Computer Vision) bir görüntü işleme kütüphanesidir. OpenCV, Python ve diğer programlama dilleri ile birlikte kullanılabilir ve görüntüler üzerinde çeşitli işlemler gerçekleştirilebilmesine olanak sağlar. OpenCV, 2001 yılında Itseez tarafından kurulmuş ve daha sonra Intel tarafından satın alınmıştır. OpenCV, görüntüler üzerinde çeşitli işlemler gerçekleştirilebilmesine olanak sağlar ve bu işlemler arasında, görüntülerin okunması, ölçeklendirilmesi, dönüştürülmesi, filtrelemesi gibi işlemler yer alır. OpenCV, yaşam bilimleri dahil olmak üzere çok çeşitli bağlamlarda uygulanan yaygın bir bilgisayar görüşü ve makine öğrenimi kitaplığıdır. (Kaehler and Bradski 2007). OpenCV'nin gücü, bu kitaplık tarafından sağlanan hem klasik

hem de son teknoloji bilgisayar görme algoritmalarının büyük miktarına (2500'den fazla) dayanmaktadır. OpenCV yardımıyla pek çok bilgisayarlı görü sistemi gerçekleştirmek mümkündür (Sirkov and Novikov 2020). OpenCV, görüntü işleme, özellik algılama, nesne algılama, makine öğrenimi ve video analizi için algoritmalar sağlar. OpenCV'yi yaşam bilimlerinde kullanmanın başlıca zorlukları, kullanılabilirliği ve etkileşimidir: OpenCV, ne varsayılan olarak bir grafik arabirimi ne de ilgi alanları (ROI'ler) ile etkileşim işlevselliği sağlar. Bu, OpenCV ile etkileşimi kodlamak gerektiği anlamına gelir ve bu, yaşam bilimcileri için bir sorun olabilir (Dominguez and Heras 2017). OpenCV (Open Source Computer Vision) adı verilen bir kütüphane ve açık kaynak kodlu bir bilgisayar görüşü kütüphanesidir. OpenCV, bilgisayar görüşü algoritmalarının kullanımını ve uygulanmasını kolaylaştıran bir yapıya sahiptir. OpenCV, görüntü işleme, nesne tespiti ve tanıma, veri madenciliği gibi birçok alanda kullanılabilir. Örneğin, OpenCV kütüphanesi kullanılarak, görüntüler ve video dosyaları okunabilir, düzenlenebilir ve işlenebilir. Görüntülerdeki nesneler tanımlanabilir ve tespit edilebilir ve görüntüler arasındaki benzerlikler analiz edilebilir. Görüntünün daha önceden sizin belirlediğiniz özniteliklere sahip olup olmadığının kontrolü yapılabilir. OpenCV kütüphanesinin birçok özelliği vardır. Örneğin, OpenCV kütüphanesi, görüntülerdeki nesneleri tespit etmek için Haar cascades yöntemini kullanır. Bu yöntem, nesne tanıma için kullanılan bir makine öğrenmesi yöntemidir ve OpenCV kütüphanesi tarafından desteklenir. OpenCV ayrıca, görüntülerdeki nesnelerin yerlerini tespit etmek için template matching yöntemini de destekler Ayrıca, OpenCV, görüntüler üzerinde nesne tanıma, yüz tanıma, el yazısı tanıma gibi işlemleri de gerçekleştirilebilmesine olanak sağlar. Ayrıca “Python ve OpenCV ile bir velodrom pistinde takım değişimi sırasında tek bir bisikletçiyi izleme” isimli çalışmada OpenCV kütüphanesi kullanarak hareket eden bir nesneyi tespit eden ve takip edebilen sistemlerin kurgulanmasının mümkün olduğu görülmüştür (Burden and Cleland 2010).

OpenCV, birçok farklı alanda kullanılır. Örneğin: Endüstriyel otomasyon: OpenCV, endüstriyel otomasyon sistemlerinde kullanılır ve bu sistemlerde, görüntüler kullanılarak üretim süreçlerinin izlenmesi ve kontrolü sağlanır. Sağlık: OpenCV, sağlık alanında da kullanılır ve bu alanda, görüntüler kullanılarak hastalıkların tanısı ve tedavisi için kullanılır. Örneğin, MR görüntülerini kullanılarak beyin tümörlerinin tespiti ve takibi yapılabilir.

OpenCV Hough Dönüşüm yöntemleriyle bir resim dosyasındaki daire ve çizgileri algılayabilir. Çizge teorisinde çizgelerin tepeleri genelde daire şeklinde ayrıtları ise çizgiler olarak temsil edilmektedir. Özellikle bir çizge barındıran resim dosyasından çizgenin tepeleri ve ayrıtlarının tespit edilmesi aşamaları için openCV ve Hough dönüşüm algoritmaları yüksek öneme sahiptir.

Matplotlib, Python dilinde kullanılan bir veri görselleştirme kütüphanesidir. Matplotlib, çeşitli grafikler, histogramlar, günlükler, dağılım grafikleri gibi çeşitli veri görselleştirme yöntemleri oluşturmak için kullanılır. Matplotlib kütüphanesi, Python kodu kullanılarak kolayca kullanılabilir ve açık kaynak kodlu olduğu için ücretsizdir.

Matplotlib, veri görselleştirme öğelerinin oluşturulması için birkaç yöntem sunar. Örneğin, plot() fonksiyonu, veri setlerini bir grafikte göstermek için kullanılabilir. Ayrıca, scatter() fonksiyonu, veri noktalarını bir grafikte dağılımını göstermek için kullanılabilir. Matplotlib ayrıca, veri görselleştirme öğelerine çeşitli özelleştirme seçenekleri sunar, örneğin veri etiketleri ve legendler ekleyebilir, eksen etiketleri ve başlıklar ekleyebilir ve grafikleri farklı renklerde ve stil seçenekleriyle özelleştirebilirsiniz.

Matplotlib kütüphanesi, birçok farklı veri görselleştirme yöntemleri destekler ve bu yöntemlerin birçoğu, veri görselleştirme için kullanılan diğer popüler kütüphanelerin özelliklerine benzerdir. Bu nedenle, Matplotlib kütüphanesi, veri görselleştirme için ihtiyacınız olan birçok özelliği sunar.

Python, yüksek seviyeli, güçlü ve çok yönlü bir programlama dilidir ve bu nedenle bazı avantajları vardır: Kolaylık: Python, okunabilirliği yüksek ve anlaşılır bir dil olduğu için, özellikle başlangıç seviyesi programcılar tarafından kolayca öğrenilebilir. Çok yönlülük: Python, farklı alanlarda kullanılabilir ve bu sayede, çok çeşitli projeler geliştirilebilir. Örneğin, veri bilimi, makine öğrenimi, web geliştirme, masaüstü uygulamaları gibi alanlarda kullanılabilir. Büyük topluluk: Python, dünya çapında büyük bir topluluğa sahiptir ve bu topluluk, Python ile ilgili yardım almak, öğrenmek ve geliştirmek isteyenler için çok yararlıdır. Ayrıca, Python topluluğu tarafından geliştirilen paketler ve kütüphaneler de mevcuttur ve bu paketler ve kütüphaneler, Python ile geliştirilen uygulamaların çok daha kolay ve hızlı geliştirilebilmesine yardımcı olur. Ölçeklenebilirlik: Python, ölçeklenebilir bir dil olduğu için, büyük ve karmaşık projeler için de uygun bir seçenektir. Açık kaynaklı: Python, açık kaynaklı bir

dil olduğu için, ücretsiz olarak kullanılabilir ve kullanıcılar tarafından geliştirilebilir.

Pillow da bir görüntü işleme kütüphanesidir. Pillow, Python ile birlikte kullanılabilir ve görüntüler üzerinde çeşitli işlemler gerçekleştirilebilmesine olanak sağlar. Pillow, PIL (Python Image Library) adıyla bilinen eski bir görüntü işleme kütüphanesinin devamıdır ve PIL'in bazı sınırlamalarını aşmayı amaçlar. Pillow, görüntüler üzerinde çeşitli işlemler gerçekleştirilebilmesine olanak sağlar ve bu işlemler arasında, görüntülerin okunması, yazılması, ölçeklendirilmesi, dönüştürülmesi gibi işlemler yer alır. Pillow, birçok farklı görüntü dosya türünü destekler ve bu dosya türleri arasında, JPEG, PNG, TIFF gibi dosya türleri yer alır.

Pillow, birçok farklı alanda kullanılır. Örneğin: Web geliştirme: Pillow, web geliştirme projelerinde kullanılır ve bu projelerde, görüntülerin yüklenmesi, düzenlenmesi ve gösterilmesi gibi işlemler gerçekleştirir.

NumPy, bir Python kütüphanesidir ve veri işleme ve cinsinden ciddi bir hız kazanç sağlar. NumPy, verilerin düzenlenmesi, ölçeklendirilmesi, filtrelemesi gibi işlemler için optimizasyonlar sağlar. NumPy, verilerin düzenlenmesinde diziler (ndarray - n-boyutlu dizi) adı verilen veri yapılarını kullanır. Bu diziler, verilerin belirli bir şekilde düzenlenmesini sağlar ve verilerin hızlı bir şekilde işlenmesine olanak sağlar. NumPy dizileri, Python listelerine benzerdir ancak daha hızlıdır ve daha fazla özellik sunar.

NumPy, birçok farklı alanda kullanılır. Örneğin: Veri bilimi: NumPy, veri bilimi projelerinde sıklıkla kullanılır ve bu projelerde, verilerin düzenlenmesi, ölçeklendirilmesi, filtrelemesi gibi işlemler için kullanılır. Makine öğrenimi: NumPy, makine öğrenimi projelerinde de kullanılır ve bu projelerde, verilerin hazırlanması ve işlenmesi için kullanılır. Öğretim: NumPy, öğretim amaçlı projelerde de kullanılır ve bu projelerde, verilerin öğrenciler tarafından daha kolay anlaşılmasını sağlar. NumPy, Python ile birlikte kullanılır ve Python kurulumunda otomatik olarak yüklenir. NumPy kütüphanesinin kullanımı, verilerin yüklenmesi ve düzenlenmesi gibi işlemlerle başlar ve daha sonra veriler üzerinde çeşitli işlemler gerçekleştirilebilir.

3.4 Spyder

Spyder (Scientific Python Development Environment) adı verilen bir Python geliştirme ortamıdır. Spyder, birçok popüler Python kütüphanesini (örneğin NumPy, Matplotlib ve SciPy) destekler ve bu kütüphanelerin kullanımını kolaylaştırır. Spyder ayrıca, birçok Python ile ilgili araç ve özellik içerir ve bu araçlar sayesinde Python kodunun yazımı, test edilmesi ve depolanması kolaylaştırılır.

Spyder, birçok farklı Python dilinde çalışabilen bir geliştirme ortamıdır ve bu nedenle Python ile ilgili birçok projede kullanılabilir. Örneğin, Spyder kullanılarak veri görselleştirme projeleri yapılabilir, machine learning modellemeleri yapılabilir ve sayısal hesaplamalar gerçekleştirilebilir.

Spyder, birçok araç ve özelliğe sahiptir ve bu araçlar sayesinde Python kodunun yazımı, test edilmesi ve depolanması kolaylaştırılır. Örneğin, Spyder, Python kodunun yazılması için bir kod editörü içerir ve bu kod editörü, Python kodunun düzenlenmesi ve yazılması için faydalı araçlar sunar. Spyder ayrıca, Python kodunun test edilmesi için bir konsol ve bir hata ayıklayıcı (debugger) da içerir. Bu araçlar sayesinde, Python kodunun çalışma durumu izlenebilir ve hata ayıklanabilir. Spyder ayrıca, Python kodunun depolanması için bir proje yöneticisi de içerir ve bu proje yöneticisi sayesinde, Python kodunun sürümleri yönetilebilir ve kodun değişiklikleri takip edilebilir.

Spyder, Python ile ilgili birçok projede kullanılabilen bir geliştirme ortamıdır ve birçok araç ve özelliğe sahiptir. Bu araçlar sayesinde, Python kodunun yazımı, test edilmesi ve depolanması kolaylaştırılır ve bu sayede Python ile ilgili projeler daha verimli bir şekilde yürütülebilir.

Kod editörü: Spyder, kodların yazımı için bir kod editörü içerir ve bu kod editörü, kodların yazımını kolaylaştıran birçok özellik sunar. Örneğin, otomatik tamamlama, sözcük önerileri, açıklamalı işaretler gibi özellikler vardır. Çalıştırma penceresi: Spyder, kodların çalıştırılması için bir çalıştırma penceresi içerir. Bu çalıştırma penceresi, kodların çalıştırılmasını ve çıktılarının görüntülenmesini sağlar. Hata ayıklama: Spyder, kodların hata ayıklanması için bir hata ayıklama aracı içerir. Bu hata ayıklama aracı, kodların çalıştırılması sırasında oluşan hata mesajlarını görüntüler ve hata ayıklanmasına yardımcı olur. Konsol: Spyder, konsol üzerinden de kodların yazılıp çalıştırılmasına

olanak sağlar. Bu konsol, kodların daha hızlı çalıştırılmasını ve test edilmesini sağlar.

3.5 Javascript

JavaScript, bir web tarama programı (web browser) üzerinde çalışan bir programlama dilidir. İnternet sayfalarına dinamik özellikler katmak için kullanılır. Örneğin, bir web sayfasında bir form doldurulurken hata mesajlarının otomatik olarak gösterilmesi veya bir butona tıklandığında bir pencerenin açılması gibi işlemler JavaScript ile yapılabilir.

JavaScript, bir script dilidir ve genelde HTML (HyperText Markup Language) veya XML (Extensible Markup Language) dosyalarına yerleştirilerek kullanılır. HTML ve XML dosyaları, bir web sayfasının görünümünü ve yapısını tanımlayan dosyalardır ve JavaScript, bu sayfalara dinamik özellikler eklemek için kullanılır. JavaScript, diğer programlama dillerinden farklı olarak, çoğunlukla tarama programı üzerinde çalışır ve bilgisayarınızda çalışan bir program değildir. Bu, JavaScript'in internet tarayıcıları tarafından desteklenmesi gerektiği anlamına gelir. Internet tarayıcıları, JavaScript kodunu anlayarak çalıştırır ve sonuçları web sayfasında gösterir.

Javascript aynı zamanda bir prototype tabanlı dil olup, nesne yönelimli programlamanın temel kavramlarını benimser. Bu, Javascript'te bir nesnenin özelliklerini ve davranışlarını tanımlayan bir "çalışma şablonu" olarak düşünülebilir. Bu şablonlar, diğer nesneler tarafından kullanılarak yeni nesneler oluşturulabilir.

Javascript ayrıca çok yönlü bir dil olup, web tarayıcılarında yalnızca değil, aynı zamanda masaüstü uygulamaları, mobil uygulamalar ve sunucu tarafı uygulamaları gibi farklı platformlarda da kullanılabilir. Örneğin, bu tez çalışması sürecinde de kullanılan Node.js adlı bir platform sayesinde JavaScript, sunucu tarafında da çalıştırılabilir. Javascript bu çalışmada sunucu tarafında tarayıcıdan gelen isteklerin doğru kanalize edilmesi için kullanıldı. MVC yapısı kurularak tasarlanan sistem mimarisinde javascript omurgada yer almaktadır. Javascript yardımıyla Node.js ortamında router ve controller oluşturularak gelen isteklerin anaconda platformundaki Python backend ile iletişimi sağlandı.

JavaScript, birçok programlama dilinden farklı olarak, prototipli bir dil olarak tasarlandı. Bu, bir nesnenin özelliklerini ve davranışlarını belirten bir şablon olarak düşünülebilen bir prototip üzerinde çalışmayı mümkün kılar. Bu sayede, JavaScript'te yeni nesneler oluşturmak ve özelliklerini ve davranışlarını bu prototip üzerinden kalıtım yoluyla geçirilebilir. JavaScript, ayrıca dinamik bir dil olarak da bilinir. Bu, bir değişkenin türünün ve değerinin çalışma zamanında belirlenmesine olanak sağlar.

3.6 Embedded Javascript

EJS (Embedded JavaScript), web geliştiricileri için bir şablon motorudur. EJS, bir web sayfasının HTML kodunun içine birkaç satır Javascript kodu ekleyebilmenizi sağlar. Bu sayede, dinamik olarak oluşturulmuş verileri bir web sayfasına dahil edebilirsiniz. EJS avantajları: Dinamik web sayfaları oluşturmayı kolaylaştırır, verileri çektiğiniz zaman otomatik olarak güncelleyen bir web sayfası oluşturabilirsiniz, kod tekrarını azaltır. Örneğin, aynı verileri birden fazla yerde göstermeniz gerektiğinde, tek bir yerde bu verileri tanımlamanız yeterlidir, kullanımı kolaydır. EJS, birkaç satır Javascript kodu eklemenize olanak sağlar ve bu kodlar HTML etiketleri içine yerleştirilebilir.

Genel kullanım kuralları şu şekildedir: Etiketler arasına veri yerleştirme: <%= veri %> gibi bir etiketle HTML etiketleri içine veri yerleştirilebilir, kod bloğu oluşturma: <% kod %> gibi bir etiketle bir kod bloğu oluşturulabilir. Bu kod bloğu, sadece EJS tarafından işlenir ve web tarayıcısında çalıştırılmaz, fonksiyon çağırma: <%= fonksiyon() %> gibi bir etiketle bir fonksiyon çağırılabilir ve fonksiyonun döndürdüğü değer yerine geçirilebilir, EJS, dinamik web sayfaları oluşturmak için kullanılan popüler bir şablon motorudur, verileri çektiğiniz zaman otomatik olarak güncelleyen bir web sayfası oluşturmayı ve kod tekrarını azaltmayı amaçlar. Etiketler arasına veri yerleştirme, kod bloğu oluşturma ve fonksiyon çağırma gibi özellikleri vardır.

3.7 Anaconda

Anaconda, bir Python dağıtımıdır ve birçok çalışma ortamını, kütüphane ve araç setini içerir. Bu dağıtım, veri bilimi, makine öğrenimi, çoklu-ortam uygulamaları ve bulut hizmetleri gibi alanlarda kullanılır.

Anaconda, birçok açık kaynaklı ve özel kütüphane ve araç setini içerir. Örneğin, NumPy, SciPy, Pandas, Matplotlib, Jupyter Notebook ve PyTest gibi popüler kütüphaneler, Anaconda ile birlikte gelir. Bu kütüphaneler, veri bilimi, makine öğrenimi ve matematiksel işlemler gibi alanlarda kullanılır.

Anaconda, ayrıca birden fazla ortam oluşturmayı ve yönetmeyi destekler. Bu ortamlar, birbirinden bağımsız olarak çalışan Python uygulamaları için kullanılabilir ve her ortamda farklı kütüphane ve araç setleri kullanılabilir. Bu sayede, aynı bilgisayarda birden fazla projede çalışabilecek ve her proje için gerekli olan kütüphaneleri kurabileceksiniz.

Anaconda, ayrıca birçok platformda çalışabilir, bu sayede Windows, macOS ve Linux gibi farklı işletim sistemlerinde de kullanılabilir.

Sonuç olarak, Anaconda, bir Python dağıtımıdır ve veri bilimi, makine öğrenimi ve matematiksel işlemler gibi alanlarda kullanılan birçok kütüphane ve araç setini içerir. Birden fazla ortam oluşturmayı ve yönetmeyi destekler ve farklı platformlarda çalışabilir. Bu yüzden bu tez çalışması arka ucu için anaconda platform kullanıldı.

3.8 Linux Ubuntu 20.04

Linux, çeşitli cihazlarda çalışan bir işletim sistemidir. Bu cihazlar, masaüstü bilgisayarlar, sunucular, mobil cihazlar ve hatta bazı ev aletleri gibi geniş bir yelpazede yer alır. Linux, özellikle sunucu ve ciddi işletmelerde yaygın olarak kullanılır, ancak masaüstü bilgisayarlar için de çeşitli dağıtımları mevcuttur.

Linux, özellikle özelleştirilebilirliği ve açık kaynaklı yapısıyla bilinir. Bu, Linux işletim sisteminin kaynak kodlarının herkes tarafından görülebilir ve değiştirilebilir olması demektir. Bu sayede, Linux'un özelleştirilebilir ve özel ihtiyaçları karşılayacak şekilde özelleştirilebilir.

Linux, ayrıca çok sayıda komut satırı araçları içerir. Bu araçlar, sistem yönetimi görevlerini gerçekleştirmeyi ve sistemler arasında veri taşımayı kolaylaştırır. Linux ayrıca birçok ücretsiz ve açık kaynaklı yazılım ve uygulamaları destekler.

Linux, genellikle bir dağıtım olarak dağıtılır. Bir dağıtım, bir Linux işletim sisteminin çeşitli parçalarının bir arada toplandığı bir pakettir. Örneğin, Ubuntu ve Fedora gibi popüler dağıtımlar, Linux işletim sistemini ve gerekli araçları içerir. Bu sayede, Linux'u kurmak ve kullanmak için çeşitli parçaları tek tek indirmenize gerek kalmaz. Ubuntu dağıtımı, Linux işletim sisteminin çeşitli parçalarının bir arada toplandığı bir pakettir. Ubuntu, masaüstü bilgisayarlar için tasarlandı ve özelleştirilebilirliği, güçlü komut satırı araçları ve çeşitli ücretsiz yazılımları ile bilinir.

Ubuntunun avantajları: Özelleştirilebilirlik: Ubuntu, özelleştirilebilirliği ile bilinir. Bu, Ubuntu işletim sisteminin özel ihtiyaçlarınızı karşılayacak şekilde özelleştirilebilmesini sağlar. Güçlü komut satırı araçları: Ubuntu, çok sayıda komut satırı araçları içerir. Bu araçlar, sistem yönetimi görevlerini gerçekleştirmeyi ve sistemler arasında veri taşımayı kolaylaştırır. Çeşitli ücretsiz yazılım ve uygulamalar: Ubuntu, birçok ücretsiz yazılım ve uygulamaları destekler. Bu yazılımlar arasında, ofis uygulamaları, resim düzenleme programları ve oyunlar gibi çeşitli türlerde uygulamalar bulunur.

Sonuç olarak, Linux, çeşitli cihazlarda çalışan bir işletim sistemidir ve özelleştirilebilirliği ve açık kaynaklı yapısıyla bilinir. Birçok komut satırı araçları ve ücretsiz yazılım ve uygulamaları destekler ve genellikle dağıtım olarak dağıtılır. Bu avantajlar ve özelliklerin olumlu olmasından dolayı Ubuntu işletim sistemi bu tez kapsamında geliştirme ortamı olarak kullanıldı.

3.9 Visual Studio Code

Visual Studio Code (VSCode), bir kod editörüdür ve çeşitli programlama dilleri için kullanılabilir. VSCode, özellikle web geliştirme için popülerdir ve HTML, CSS ve JavaScript gibi dilleri destekler. VSCode, ayrıca birçok eklenti ve araç seti ile genişletilebilir ve bu sayede özel ihtiyaçlarınızı karşılayacak şekilde özelleştirilebilir.

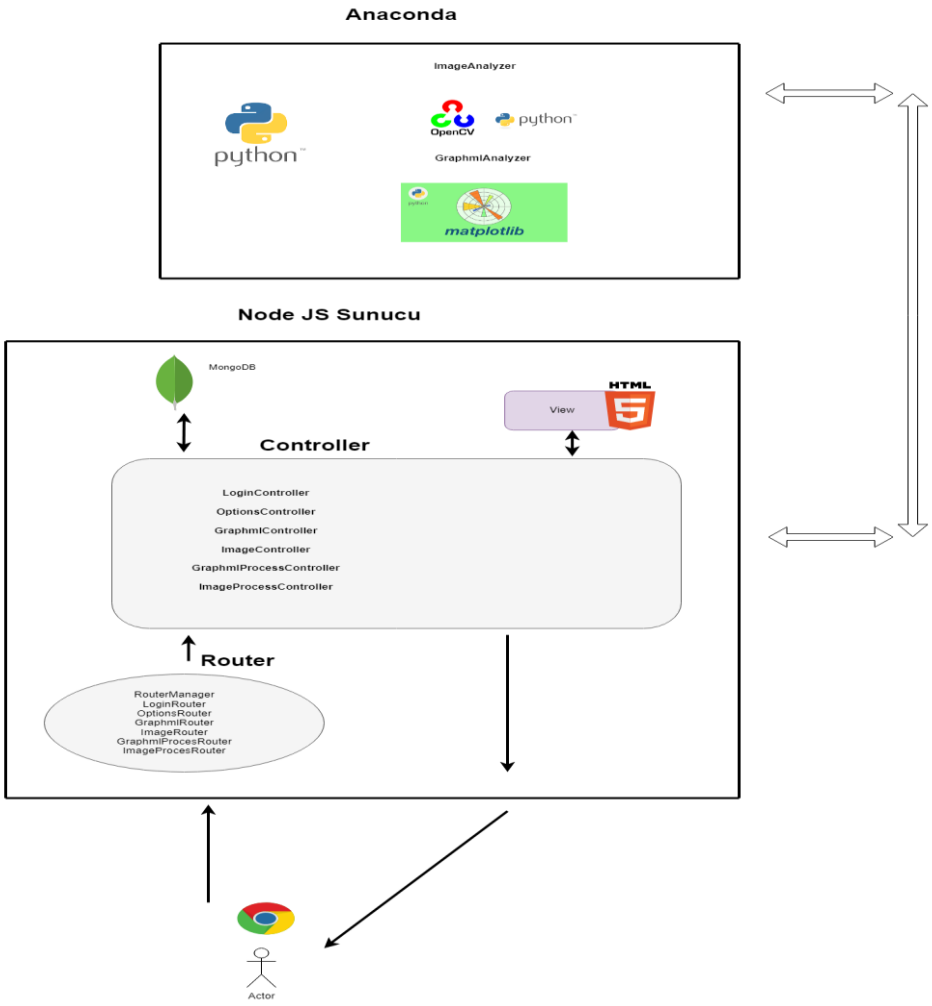
VSCode, JavaScript geliştirme için uygun bir kod editörüdür. Özellikle, VSCode'un JavaScript Debugger eklentisi sayesinde JavaScript kodlarınızı hata ayıklama ve test etmeyi kolaylaştırır. Ayrıca, VSCode'un IntelliSense özelliği sayesinde kod yazarken otomatik tamamlama ve hata ayıklama özellikleri sunar. Bu sayede, hızlı ve doğru bir şekilde JavaScript kodları yazmanıza yardımcı olur. VSCode ayrıca, Git özelliği sayesinde kodlarınızı sürüm yönetimi için

kullanabilirsiniz. Bu sayede, kodlarınızın değişikliklerini izleyebilir ve daha önceki sürümlerine geri dönebilirsiniz.

Sonuç olarak, Visual Studio Code (VSCode), JavaScript geliştirme için uygun bir kod editörüdür. JavaScript Debugger eklentisi sayesinde hata ayıklama ve test etmeyi kolaylaştırır ve IntelliSense özelliği sayesinde kod yazarken otomatik tamamlama ve hata ayıklama özellikleri sunar. Bu proje tez çalışmasının ön yüzünde ve sunucu tarafında javascript ve javascript tabanlı teknolojiler kullanıldığı için VSCode geliştirme esnasında kullanılmaya uygun görüldü.

4. PROJE GERÇEKLEŞTİRİM ADIMLARI

4.1 Proje Genel Mimarisi



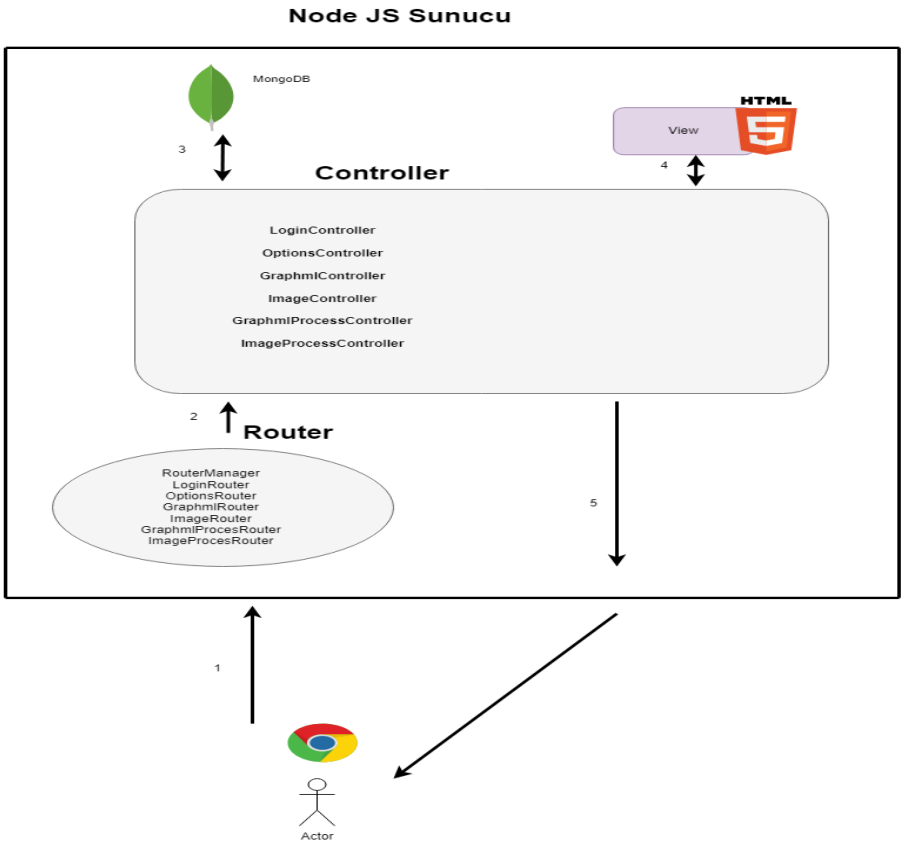
Şekil 4.1 Sisteme Erişim Senanaryosu

4.2 Sisteme Erişim Senaryosu

Şekil 4.1'de sistemin genel mimarisi görselleştirildi. Sistem iki ana yapıdan oluşmaktadır. Birincisi gelen isteklerin adreslenip yönetilmesinde kullanılan Node.js sunucu yapısı ve MongoDB veritabanı ikinci anayapı ise Python ve görüntü işleme ve görselleştirme işlemlerinin kurgulandığı Anaconda çerçevesidir.

Sistemin kullanımı adım adım şu şekilde sınıflandırarak aktarabiliriz:

- Kullanıcı sistemin olduğu adrese istekte bulunur daha sonra kullanıcı adı ve şifresiyle sisteme giriş yapar.
- Kullanıcı başarılı giriş işlemi yapması sonucunda sistem tarafından graphml ve png dosyası için iki seçenek sunulur.
- Kullanıcı seçtiği seçeneğe göre uygun formattaki dosyayı sisteme yükler.
- Sunucu ilgili dosyayı Anaconda çerçevesinin işleyebileceği konum ve formata getirir ve anaconda çerçevesini tetikler.
- Anaconda çerçevesindeki ilgili kod parçaçıkları gelen dosyayı işler. Sunucu işlenen dosyaları kullanıcıya aktarır.

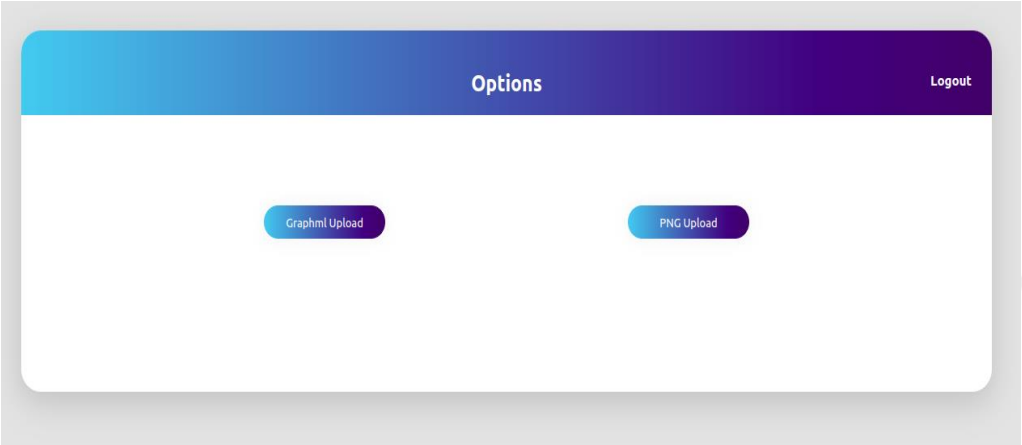


Şekil 4.2 Sisteme Giriş

Şekil 4.2'de: Kullanıcı düzlemsel çizge sistemine erişmek için bir web tarayıcısı üzerinden sunucunun adresine istek atar. Gelen istek yönlendirme yöneticisine gider. Yönlendirme yöneticisi isteği çözümler ve ilgili kontrolcü birimine aktarır. Kontrolcü gelen isteğin giriş sayfasına get isteği olduğunu görür ve giriş işlemleri için kullanılan EJS sayfasını kullanıcıya gösterir.

İkinci adım: 1: Kullanıcı gelen giriş sayfasına kullanıcı ve şifresini girer ve http post isteği olarak sunucuya gönderir. 2: Sunucu tarafında yönlendirme yöneticisi isteği giriş yönlendiricisine giriş yönlendiricisi de giriş kontrolcüsüne gönderir. 3: Giriş kontrolcüsü gelen kullanıcı adı ve şifrenin sistemdeki

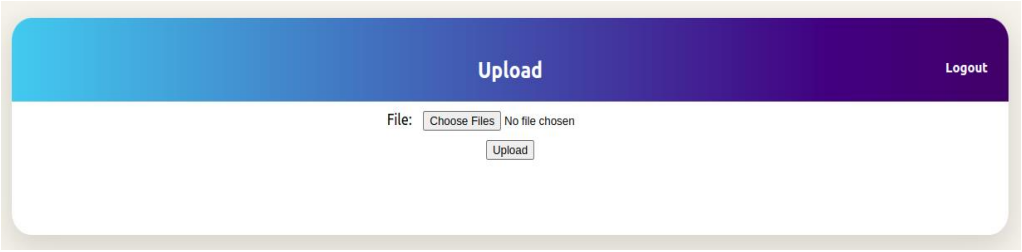
veritabanında bulunup bulunmadığını kontrol eder. 4: Düzlemsel çizge veritabanı sisteminde girilen bilgilerle eşleşen bir kullanıcı bulunması durumunda kontrolcü kullanıcının sistemi kullanabilmesi için sistem kaynaklarını kullanıcıya gönderir. Girilen kullanıcı ve şifre parametreleri yanlış bir değere sahipse kullanıcı giriş sayfasına tekrar gönderilir. Sisteme girmesine izin verilmez.



Şekil 4.3 Graphml ve Png Yükleme Seçenekleri

Kullanıcı adı ve şifresini doğru giren kullanıcı şekil 4.3'deki dosya yükleme seçeneklerinin olduğu sayfaya yönlendirilir. Kullanıcı daha sonra seçenek kontrolcüsü tarafından yüklemek istediği dosyaya göre ilgili dosya yükleme sayfasına yönlendirilir.

4.3 Görüntünün Sisteme Yüklenmesi Süreci



Şekil 4.4 Dosya Yükleme Sayfası

Png dosyayı yükleme servisi kullanıldığında yönlendirici yöneticisine giden istek oradan image yönlendiricisine aktarılır. Image yönlendiricisi image kontrolcüsüne mantıksal işlemlerin yapılması için başvuru yapar. Png uzantılı dosya multer modülü kullanılarak anaconda platformunun işleme yapacağı ortama aktarılır.

```
var express = require('express');
var router = express.Router();
var ctrlsigned = require('../controller/imageprocessController');

const multer = require('multer');
const path = require('path');
const storage = multer.diskStorage({

  destination: function(req, file, cb) {
  cb(null,path.join(__dirname,'../../public'));

  },

  filename: function(req, file, cb) {
     cb(null, file.fieldname+ path.extname(file.originalname))
  }
});

var upload = multer({ storage: storage })
module.exports=upload
router.get('/', ctrlsigned.index);
router.post('/',upload.array('slot'), ctrlsigned.indexPost);
module.exports = router;
```

Yukarıdaki yöntemle gelen png dosyası düzlemsel çizge sistemine alındıktan sonra anaconda platformunun işlemesi için gereken konuma depolanır. Bu adımların gerçekleştirilmesi için multer ve express modülleri kullanılır.

Express, JavaScript dili ile yazılmış bir web çerçevesidir. Bu çerçeve, web uygulamaları ve API'lerin geliştirilmesi için kullanılır. Express, Node.js üzerinde çalışır ve birçok farklı özelliğe sahiptir. Örneğin, Express ile: HTTP isteklerini işleyebilir ve yanıt verebilir Dinamik içerik oluşturabilir ve gönderebilir Önbellekleme, yönlendirme ve diğer web özelliklerini kullanabilir Veritabanına erişebilir ve veri depolayabilir Express, diğer web çerçevelerine göre daha hafif ve esnektir. Bu, özellikle küçük ve orta ölçekli projeler için iyi bir seçim olabilir. Ayrıca, Express'in yüksek seviyede kullanım kolaylığı ve açık kaynak kodlu olması, geliştiriciler tarafından sıklıkla tercih edilir.

Multer, Node.js için yazılmış bir eklentidir ve HTTP isteklerinde yer alan dosya yüklemelerini işlemeyi amaçlar. Multer, Express çerçevesi ile birlikte kullanıldığında, dosya yüklemelerini kolayca kullanılabilir hale getirir. Multer, dosya yüklemelerini yönetmek için çeşitli seçenekler sunar. Örneğin, yüklenen dosyaların nereye kaydedileceğini ve dosya isimlerinin nasıl değiştirileceğini belirleyebilirsiniz. Ayrıca, Multer ile dosya türlerini ve dosya boyutlarını sınırlayabilir ve yüklenen dosyaların geçerli olup olmadığını doğrulayabilirsiniz.

Multer, yalnızca dosya yüklemeleri için değil aynı zamanda dosya indirme ve dosya güncelleme işlemleri için de kullanılabilir. Bu, web uygulamalarında dosya yönetimi için çok yönlü bir araç sağlar. Bu avantajlarından dolayı express ve multer modüllerini birlikte kullanılarak dosya transferlerinde görev aldı.

4.4 Görüntü İşleme Hazırlığı ve Anaconda Platformu Tetiklenmesi

```
const spawn = require('child_process').spawn;
const delay = require('delay');
const fs = require('fs');
const rimraf = require('rimraf');

const {
  read
} = require('fs');
const {
  session
} = require('passport');

module.exports.index = function (req, res) {
  if (req.session.userId) {
    res.render('imageprocess')
  } else {

    res.redirect('/');
  }
}
module.exports.indexPost = function (req, res) {
  if (req.session.userId) {

    const process = spawn('python3', ['image.py']);
    process.stdout.on('data', data => {});
    process.stderr.on('data', function (data) {});
    process.on('close', function (code) {});
    process.on('close', function (code) {});
    (async () => {

    while(!fs.existsSync('public/graph.png')) {
      await delay(2000);
    }

    res.render('imageprocess');

    })();
    //res.redirect('/imageprocess');

  } else {
    res.redirect('/');
  }
}
```

Yukarıdaki resim işleme kontrolcüsü resim işleme yönlendiricisine gelen get ve post isteklerini karşılamak için geliştirilmiştir. Gelen istekler eğer kullanıcının sessionid'si hala geçerliyse gerçekleştirilir. Bu bloğun temel görevi anaconda çerçevesindeki python kodlarını bir childprocess olarak çalıştırmak ve

bu işlemin çıktı zamanına göre son kullanıcıya uygun EJS sayfasının gösterilmesinin organizasyonunu yapmaktır. Bu fonksiyonel özellikleri sağlamak için child_process, delay, rimraf ve fs modülleri kullanılmıştır.

Node.js "child_process" modülü, bir Node.js uygulaması içinde diğer bir işlem oluşturmanızı ve bu işlemleri yönetmenizi sağlar. Bu modül, Node.js çerçevesinin "process" modülünden türetilmiştir ve aşağıdaki işlevleri sağlar:

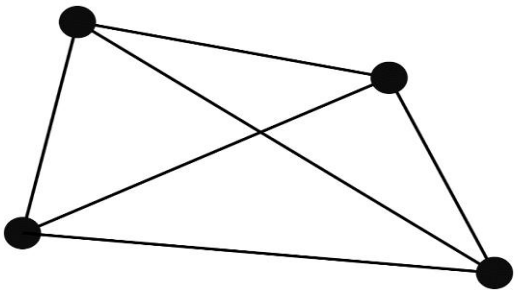
- Yeni bir işlem oluşturma: "spawn", "exec", "execFile" ve "fork" gibi fonksiyonlar kullanılarak yeni bir işlem oluşturulabilir.
- Işlemlerin haberleşmesi: Işlemler arasında veri gönderme ve alma işlemleri yapılabilir.
- Işlemlerin durumunun izlenmesi: Işlemlerin çalışma durumu ve çıkış kodları izlenebilir.

Child_process modülü, çeşitli senaryolarda kullanışlıdır. Örneğin, bir Node.js uygulamasında bash komutlarını çalıştırmak için kullanılabilir veya Node.js uygulamasının birden fazla CPU çekirdeği kullanarak paralelleştirilmesini sağlayabilir. Bu tez projesi kapsamında child_process nodejs sunucusundan anaconda platformundaki python kodlarını çalıştırmak için kullanıldı.

Node.js "rimraf" modülü (REMOVE Rm -rf), Node.js çerçevesi içinde bir dizin veya dosya silme işlemini gerçekleştirir. Bu modül, bir komut satırı uygulaması olarak kullanılan "rm -rf" komutunun Node.js çerçevesinde kullanılabilir hale getirilmiş halidir. Rimraf modülü, dizinler ve dosyalar içindeki tüm alt dosya ve dizinleri de siler. Bu, dizinler ve dosyaları tamamen temizlemeyi sağlar. Rimraf modülü, genellikle Node.js uygulamalarında dosya ve dizinlerin silinmesi gerektiğinde kullanılır. Örneğin, bir Node.js uygulaması çalıştırılmadan önce gerekli olan dosya ve dizinlerin silinmesi için kullanılabilir.

PNG Formatındaki Resim Dosyasının Görüntü İşleme Yöntemleriyle Analiz Edilmesi.

Node.js tarafında kontrolcünün anacondayı tetiklemesiyle birlikte resim dosyasının işlenmesini aşağıdaki kod bloğu sağlamaktadır.



Şekil 4.5 Düzlemsel Görünümü İstenen Çizge

```
        import cv2
        import numpy as np
        from PIL import Image
        import networkx as nx
        import matplotlib.pyplot as plt
        from subprocess import call
        import math
        import os
        G = nx.DiGraph()

        img = cv2.imread("./public/slot.png",cv2.IMREAD_COLOR)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray = cv2.medianBlur(gray, 11)
        rows = gray.shape[0]
        circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT,
0.01, rows / 15,
                           param1=500, param2=28 ,
                           minRadius=1, maxRadius=70)

        output = img.copy()
        if circles is not None:

          circles = np.round(circles[0, :]).astype("int")

          for (x, y, r) in circles:

                cv2.circle(output, (x, y), r, (255, 0,0), 4 )
                cv2.rectangle(output, (x - 5, y - 5), (x + 5, y + 5), (0,
128, 255), -1)
                cv2.rectangle(img,(x , y ), (x ,y), (255, 255, 255), 3*r)

          cv2.imwrite("./public/circle.png",output)
        else:
           print("not found")


        default_file ='./public/slot.png'
        src = cv2.imread(default_file, cv2.IMREAD_GRAYSCALE)
```

```
image = cv2.Canny(src, 50, 200, None, 3)



cimage = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
cimageP = np.copy(cimage)
def find_line_slope(point1, point2):
  x1, y1 = point1
  x2, y2 = point2


  if (x2-x1)!=0:
    m = (y2 - y1) / (x2 - x1)

    angle = np.arctan(m) * 180 / np.pi
  else:
    angle=90
  return angle


linesP = cv2.HoughLinesP(image, 1, np.pi / 180, 50, None,2*r,
r)

def lineboy(x0, y0, x1, y1):
  return math.sqrt((x0 - x1)**2 + (y0 - y1)**2)


def minimum_distance(a0, b0, x0, y0, x1, y1):
  if x1-x0==0:
    distance=abs(x1-a0)

  else:

    a = (y1 - y0) / (x1 - x0)
    b = y0 - a * x0
    # minumum mesafe formülü
    distance = abs(a0 * a - b0 + b) / math.sqrt(a**2 + 1)
  return distance

edges=np.zeros((0, 4))


for a in circles:
    for b in circles:
        mcircle=find_line_slope((a[0],a[1]),(b[0],b[1]))
        if a[0]==b[0] and a[1]==b[1]:

            break
```

```
                else:
                    for i in range(0,len(linesP)):
                        l=linesP[i][0]
                        cv2.line(cimageP, (l[0], l[1]), (l[2], l[3]), (255,0,0), 3,
cv2.LINE_AA)
                        boy=lineboy(l[0],l[1],l[2],l[3])

mindist1=minimum_distance(l[0],l[1],a[0],a[1],b[0],b[1])

mindist2=minimum_distance(l[2],l[3],a[0],a[1],b[0],b[1])
                        medge=find_line_slope((l[0],l[1]),(l[2],l[3]))
                        mdiff=abs(mcircle-medge)
                        if mdiff<0.8:
                            if boy>3*r:
                                if mindist1<r/4 and mindist2<r/4:
                                    edges                                    =
np.append(edges,[[a[0],a[1],b[0],b[1]]],axis=0)

        finaledges=np.zeros((0, 4))
        finaledges=np.append(finaledges,[[1,2,3,4]],axis=0)

        for a in edges:
            flag=False
            for b in finaledges:
                if a[0]==b[0] and a[1]==b[1]:
                    if a[2]==b[2] and a[3]==b[3]:
                        flag=True
                if a[0]==b[2] and a[1]==b[3]:
                    if a[2]==b[0] and a[3]==b[1]:
                        flag=True

            if flag==False:
                finaledges                                                   =
np.append(finaledges,[[a[0],a[1],a[2],a[3]]],axis=0)
            flag=False

        finaledges=finaledges[1:]
        for i in finaledges:
            x1=i[0]
            y1=i[1]
            x2=i[2]
            y2=i[3]
            G.add_node(tuple((x1, y1)))
            G.add_node(tuple((x2, y2)))
            G.add_edge((x1, y1), (x2, y2))
        Ga = G
        def main(G):
            fig = plt.figure()
            graph = nx.Graph()
```

```
        for v in G.nodes():
            graph.add_node(v)

        for delta in G.edges():
            for w in delta:
                graph.add_edge(delta[0],delta[1])
        nx.draw_planar(graph)
        fig.savefig('./public/graph.png')
    try:
      main(Ga)
    except:
        ca = ' cp graph.png public/'
        cal=call(ca, shell=True)
    cv2.imwrite("./public/line.png", cimageP)

    nx.write_graphml(G, "./public/graph.graphml")
    file_path = './public/graph.graphml'
    if os.path.exists(file_path):
        create='mkdir public/graph'
        cpg= 'cp public/graph.png public/graph/'
        cpline='cp public/line.png public/graph/'
        cpcircle='cp public/circle.png public/graph/'
        cpgml='cp public/graph.graphml public/graph/'
        zipf='zip -r public/graph.zip public/graph '
        decrypted = call(create, shell=True)
        decrypted = call(cpg, shell=True)
        decrypted = call(cpline, shell=True)
        decrypted = call(cpgml, shell=True)
        decrypted = call(zipf, shell=True)
        decrypted = call(cpcircle, shell=True)
```

Cv2 (OpenCV) kütüphanesi, görüntü işleme ve bilgisayar görüşü gibi alanlarda kullanılan bir kütüphanedir. OpenCV, görüntülerden veri çıkarma, görüntüler arasındaki ilişkileri tanımlama ve görüntüler üzerinde değişiklikler yapma gibi işlemleri yapmak için kullanılabilir. Örneğin, OpenCV kullanılarak görüntülerde nesne tespiti, görüntü stabilizasyonu, görüntü ön işleme ve görüntüler arasında geçiş yapma gibi işlemler gerçekleştirilebilir. Bu tez kapsamında resim dosyasındaki bir çizgenin tepe noktaları ve ayrıtlarını tespit etmek için cv2 kütüphanesinden faydalanıldı. Tepe noktalarını tespit etme konusunda OpenCV kütüphanesi kullanılarak başarılı sonuçlar elde edilmektedir. Çizgeye ait ayrıtları tespit etme konusunda OpenCV kütüphanesi hatasız sonuçlar verme konusunda başarılı değildir bu nedenle ayrıtların tespiti konusunda özgün bir optimizasyon çalışması yapıldı. Bu çalışmanın temel çıkış noktası iki tepe noktası arasında bulunma ihtimali olan ayrıtın tespit edilmesi

oldmaktadır. İki tepe nokta arasında hayali bir doğru parçasından yola çıkılarak elde edilen ayrıt parçalarının hayali doğru parçasına benzerliği kullanıldı. Benzerlik oranına göre hayali doğrunun çizildiği tepe noktaları arasında bir ayrıt bulunup bulunmadığının tespiti yapıldı.

```
img = cv2.imread("./public/slot.png",cv2.IMREAD_COLOR)
```

Yukarıdaki kod satırı openCV kütüphanesiyle bir resim dosyasının okunması için kullanıldı.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Yukarıdaki kod satırı, verilen görüntüyü (img) gri seviye görüntüsüne dönüştürür. Örneğin, bir resim üzerinde çalışılıyorsa, bu işlem resmi siyah beyaz hale getirir. Bu işleme "gri seviye dönüşümü" denir. cv2.cvtColor() fonksiyonu, OpenCV kütüphanesinin bir renk dönüşüm fonksiyonudur. Bu fonksiyon, verilen görüntüyü belirtilen renk espasından başka bir renk espasına dönüştürür. Bu örnekte, img değişkeni BGR (Blue, Green, Red - Mavi, Yeşil, Kırmızı) renk espasında bir görüntüdür ve cv2.COLOR_BGR2GRAY renk dönüşümü kullanılarak bu görüntü GRAY (Gri) renk espasına dönüştürülür. Bu dönüştürülmüş resimde aranılan spesifik şekilleri daha az hatayla tespit etmek mümkün olacaktır.

Hough Dönüşümü, bir görüntüdeki çizgilerin, dairelerin ve diğer eğrilerin tespit edilmesine yardımcı olan bir yöntemdir. Bu yöntem, görüntü işleme alanında sıklıkla kullanılır ve özellikle nesne tespiti ve tanıma gibi uygulamalarda önemlidir. Hough Dönüşümü, görüntüdeki çizgilerin ve eğrilerin parametrik bir formülle ifade edilmesine dayanır. Örneğin, bir düz çizgi için, y = mx + b formülü kullanılır, bu formül çizginin eğimi (m) ve y eksenine olan kesişim noktasını (b) gösterir. Bu formül kullanılarak, görüntüdeki çizgiler belirlenebilir ve bu çizgilerin koordinatları bulunabilir. Hough Dönüşümü, görüntüdeki çizgilerin tespit edilmesinde sıklıkla kullanılır ancak aynı zamanda dairelerin ve diğer eğrilerin tespit edilmesinde de kullanılabilir. Hough Dönüşümü, görüntüdeki çizgilerin ve eğrilerin tespit edilmesinde çok etkili bir yöntemdir ancak bu yöntem biraz zaman alıcı olabilir ve bazı durumlarda görüntülerdeki gürültüler yüzünden hassas tespitler yapılamayabilir.

Aşağıdaki kod bloğu HoughCircles yöntemiyle gri tonlara getirdiğimiz resim dosyası üzerindeki dairelerin tespitini yapmaktadır. Resim dosyası çizge içeren bir çizge dosyasıdır ve bu çizgedeki her bir daire çizgenin bir tepesini oluşturmaktadır.

```
circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 0.01, rows / 15,param1=500, param2=28 ,minRadius=1, maxRadius=70)
```

Yukarıdaki kod parçası gri seviye görüntüde (gray) daireleri tespit etmek için HoughCircles yöntemini kullanır. Bu yöntem, görüntü üzerinde dairelerin merkezlerini ve yarıçaplarını tespit etmek için kullanılır.

cv2.HoughCircles() fonksiyonu, aşağıdaki parametreleri alır:

- gray: Tespit edilecek dairelerin olduğu gri seviye görüntü.
- cv2.HOUGH_GRADIENT: Dairelerin tespit edilebilmesini kolaylaştıran bir yöntem. Bu yöntem, Hough Dönüşümünün bir modifikasyonudur.
- 0.01: Dairelerin tespit edilebilmesini kolaylaştıran bir parametre.
- rows / 15: Tespit edilecek dairelerin merkezlerinin birbirine olan uzaklığını belirten bir parametre. Bu parametre, görüntünün satır sayısının 15'e bölümüdür.
- param1: Dairelerin tespit edilebilmesini kolaylaştıran bir parametre. Bu parametre, Hough Dönüşümü için kullanılan bir threshold değeridir.
- param2: Dairelerin tespit edilebilmesini kolaylaştıran bir parametre. Bu parametre, Hough Dönüşümü için kullanılan bir threshold değeridir.
- minRadius: Tespit edilecek dairelerin minimum yarıçaplarını belirten parametre.
- maxRadius: Tespit edilecek dairelerin maksimum yarıçaplarını belirten parametre.

Sonuç olarak, circles değişkeni, görüntü üzerinde tespit edilen dairelerin merkezlerini ve yarıçaplarını tutan bir dizidir. Bu dizi, her bir daire için 3 tane eleman içerir: daire merkezinin x koordinatı, daire merkezinin y koordinatı ve daire yarıçapıdır.

```
image = cv2.Canny(src, 50, 200, None, 3)
```

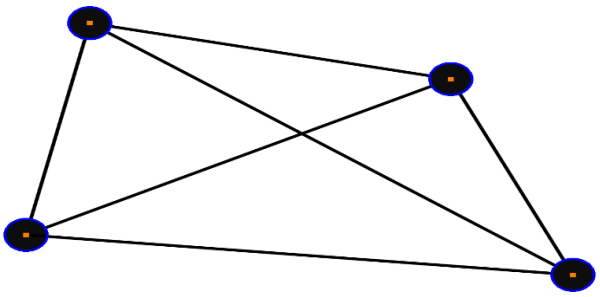
Yukarıdaki kod satırınki openCV fonksiyonu, girdi olarak verilen görüntü (src) için Canny ayrıt tespiti yapar. Bu fonksiyon, girdi olarak verilen görüntüyü iki adımda işler: Öncelikle, görüntünün Gauss gürültüsünü azaltmak için bir

Gauss filtresi uygular. Daha sonra, Canny ayrıt tespiti algoritmasını uygular. Bu algoritma, bir görüntüdeki ayrıtları bulmak için iki adımda işler:

- Görüntünün gradient değerlerini hesaplar. Gradient, görüntünün piksel değerlerinde meydana gelen değişimleri ölçen bir değerdir.
- Daha sonra, görüntüdeki potansiyel ayrıtları belirler ve bu ayrıtların güçlüğünü ölçer. Güçlü ayrıtlar, olası ayrıtlar arasından seçilir ve çıktı olarak döndürülür.

Bu fonksiyonun parametreleri şunlardır:

- src: İşlem yapılacak görüntü (8 bitlik renkli veya siyah-beyaz).
- 50: İşlem yapılacak görüntünün üst eşik değeri. Bu değer, görüntüdeki ayrıt piksellerinin gradient değerlerine göre seçilir ve bu değerden daha düşük olan pikseller atlanır.
- 200: İşlem yapılacak görüntünün alt eşik değeri. Bu değer, görüntüdeki ayrıt piksellerinin gradient değerlerine göre seçilir ve bu değerden daha yüksek olan pikseller atlanır.
- None: Bu parametre opsiyoneldir ve görüntü üzerinde işlem yapılmadan önce uygulanacak bir kernal (çapraz matris) verilebilir. Bu parametre verilmezse, standart bir 3x3 kernal kullanılır.
- 3: Bu parametre de opsiyoneldir ve görüntü üzerinde işlem yapılmadan önce uygulanacak bir kernal (çapraz matris) verilebilir. Bu parametre verilmezse, standart bir 3x3 kernal kullanılır.



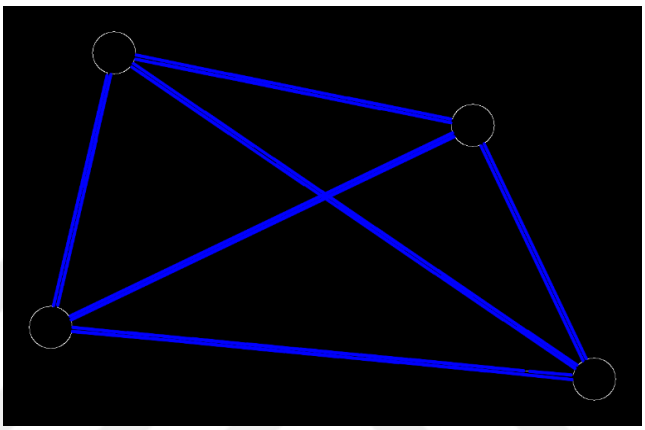
Şekil 4.6 HoughCircles Yöntemiyle Belirlenen Daireler

Yukarıda Şekil 4.5'de girilen resim dosyasının görüntü işleme yöntemiyle belirlenen daireler ve koordinatları verilmiştir. Tablo 4.1'de koordinatlar ayrıntılı olarak verilmiştir.

Kordinat X	Kordinat Y	Yarıçap
1120	720	40
890	230	40
210	90	40
90	620	40

Tablo 4.1 Resim Dosyasındaki Dairelerin Düzlemdeki Koordinatları

linesP = cv2.HoughLinesP(image, 1, np.pi / 180, 50, None,2*r, r)



Şekil 4.7 Resim Dosyasında Belirlenen Çizgiler

cv2.HoughLinesP(), OpenCV kitaplığında bir görüntüdeki çizgileri algılamak için kullanılan bir fonksiyondur. Bu fonksiyon birkaç bağımsız değişken alır:

- image: Çizgilerin algılanacağı görüntü. Bu, ikili bir görüntü olmalıdır (yani yalnızca iki renkli bir görüntü).
- rho: r parametresinin piksel cinsinden çözünürlüğü.
- teta: teta parametresinin radyan cinsinden çözünürlüğü.
- eşik: Bir çizgiyi "algılamak" için minimum kavşak sayısı.
- minLineLength: Geçerli bir satır olarak kabul edilecek bir satırın (piksel olarak) minimum uzunluğu.
- maxLineGap: Tek bir çizgi oluşturmak için bağlanacak segmentler arasındaki maksimum mesafe (piksel olarak).

Bu fonksiyon, her biri iki öğeli bir vektörle (r, teta) temsil edilen bir numpy ndarray dizisi döndürür. Daha sonra bu numpy dizisinin içinden kirli verileri temizleyerek yalın ve doğru bir numpy dizisi oluşturulacak. Yeni oluşturulan dizide sadece tepeler arası bulunan ayrıtlar temsil edilecek bu sayede çizgenin doğru verilerle temsil edilme olanağı oluşacaktır.

Aşağıdaki tablo 4.2'de belirlenen tüm çizgilerin bulunduğu kirli data verilmiştir.

X1	Y1	X2	Y2
127	598	852	244
128	605	854	251
129	599	854	251
128	606	855	252
905	269	1099	685
824	510	1086	693
911	285	1105	681
824	905	1099	693
382	204	696	423
829	522	1084	700
707	429	1008	639
246	110	475	270
130	628	337	647
130	628	213	635
517	306	816	516
532	666	750	688
563	662	791	585
234	630	452	652
429	131	711	190
103	578	176	258
250	102	414	136
532	666	740	650
552	177	849	218
234	630	452	652
297	111	579	170
851	697	1079	721
251	94	507	148
997	705	1077	712
461	652	657	667
677	673	864	692
770	682	987	704
512	294	761	468
972	621	1084	699
95	579	153	328
358	642	544	661
243	117	428	246
603	357	891	558
438	140	715	198

Tablo 4.2 Resim Dosyasında Belirlenen Ayrıtların Kordinatları

K_4 çizgesinde 4 tepe noktası 6 ayrıt bulunmaktadır. OpenCV kütüphanesi fonksiyonları kullanırak 4 tepe noktasının tespit edilmesi sağlandı. Çizgenin 6 ayrıtı olmasına rağmen yukarıdaki tabloda OpenCV kütüphanesinin 39 tane ayrıt tespit ettiği gözlenmektedir. Bu tespit edilen ayrıtlarla hatasız bir çizge modellemesi olabilmesi için ayrıtların optimize edilerek hataların elimine edilmesi gerekmektedir. Bunun optimizasyon için tepe noktalarını referans alan bir algoritma geliştirildi. Aşağıdaki fonksiyon verilen koordinatların arasındaki eğim değerinden faydalanarak (x0,y0) (x1,y1) koordinatlarından geçen bir doğrunun koordinat düzlemindeki açısını bulmaktadır. Bu fonksiyonla hem iki tepe noktasından geçen doğrunun hem de resim dosyasında belirlenen doğruların açısı derece biriminden bulunmaktadır. Eğer herhangi iki tepenin arasındaki açıyla HoughlinesP fonksiyonuyla belirlenen belirlenen çizgeler arasındaki açı farkı 1 dereceden az ise bu iki node arasında bir ayrıt olma ihtimali ortaya çıkar. Bunun doğruluğunu ispatlamak için birkaç gereksinimin daha karşılanması gerekmektedir. Aşağıda diğer gereksinimler incelenmiştir.

```
def find_line_slope(point1, point2):
  x1, y1 = point1
  x2, y2 = point2
  if (x2-x1)!=0:
    m = (y2 - y1) / (x2 - x1)
    angle = np.arctan(m) * 180 / np.pi
  else:
    angle=90
  return angle
```

Aşağıdaki fonksiyon iki tepe kordinatlarından geçen bir doğru parçasına, HoughlineP algoritmasıyla belirlenen ayrıtların minimum mesafesini bulur. Eğer minimum mesafe doğrunun yarıçapının çeyreğinden küçük ise ve yukarıdaki fonksiyon ile de eğim farkı sınır değerin daha altındaysa bu doğru parçasının o iki node arasında olduğu kesinleşir. Son olarak doğru parçasının gerçek doğru mu yoksa tepe noktalarından kaynaklanan hata olup olmadığının tespiti yapılır. Son adımda da sonuç istenilen gibi olduğunda ele alınan bu iki tepe arasında bir ayrıt olduğu kesinleşir.

```
def minimum_distance(a0, b0, x0, y0, x1, y1):
  if x1-x0==0:
    distance=abs(x1-a0)

  else:

    a = (y1 - y0) / (x1 - x0)
    b = y0 - a * x0
    # minumum mesafe formülü
    distance = abs(a0 * a - b0 + b) / math.sqrt(a**2 + 1)
  return distance

def lineboy(x0, y0, x1, y1):
  return math.sqrt((x0 - x1)**2 + (y0 - y1)**2)
```

Son adımda lineboy() fonsiyonu ele alınır. Girdi olarak belirlenen çizgilerin başlangıç ve bitiş koordinatlarını alan bu fonksiyon sayesinde çizgilerin boyları bulunur. Eğer çizginin boyu belirlenen dairelerin çapından büyükse ve yukarıdaki diğer iki koşul da sağlanıyorsa, ele alınan iki tepe noktası arasında bir ayrıt olduğu kesinleşir ve o ayrıt ayrıtların bulunduğu numpy array dizisine eklenir.

X1	Y1	X2	Y2
890	230	1120	720
210	90	1120	720
210	90	890	230
90	620	1120	720
90	620	890	230
90	920	210	90

Tablo 4.3 Hataları Giderilmiş Ayrıt Tablosu

Çizgeye ait tepe noktaları $v=\{v_1,v_2,v_3,\ldots,v_n\}$ ve çizgenin belirlenen hatalı hatasız tüm ayrıtları $e=\{e_1,e_2,e_3,\ldots,e_m\}$ olsun. Yukarıdaki işlemler her bir tepe noktası noktası için kendisi hariç diğer tepe noktalarıyla çift oluşturacak şekilde (v_1-v_2, v_1-v_2, v_1-v_3, v_1-v_m) iki tepe noktası merkezinden geçen bir hayali doğru parçası oluşturulur. O doğru parçasının $e=\{e_1,e_2,e_3,\ldots,e_m\}$ dizisi içerisinde yer alan ayrıtlarla olan benzerliği kontrol edilir. Ayrıtların bulunduğu dizideki uygun çizgeler optimize edilmiş çizgelerin olduğu listeye eklenir. Bu optimizasyon

algoritmasının çalıştırılması sonucunda çizgeye ait tepe noktalarının ve çizgeye ait ayrıtların bulunduğu iki farklı diziye ulaşılır.

Networkx kütüphanesi graphml gibi çizgelerin modellenebileceği bir veri tutma formatının görselleştirilmesi konusunda doğru sonuçlar verebilmektedir. Fakat bir resim dosyasındaki ayrıtları ve tepe noktalarını belirleyip çizgenin sınıflandırılmasını yapmak konusunda herhangi bir fonksiyonu bulunmamaktadır. Yine Networkx kütüphanesi numpyarray gibi çizgelerin ayrıt ve tepe noktalarının tutulabileceği veri yapıları için de bir fonksiyon kullanmamaktadır. Bu problemin de çözüme kavuşturulması için numpyarray olarak tutulan ayrıtların ve tepe noktalarının graphml veri formatına dönüştürülmesi için bir çalışma yapıldı.

Tablo 4.3'de de görüldüğü gibi 4 tepe noktası ve 6 ayrıtı buluna çizgenin tüm tepeleri ve ayrıtları elde edildi. Sonraki adımda elde edilen tepeler ve ayrıtlar kullanılarak bir graphml verisi elde edilecek ve eğer elde edilen graphml datası çizgeyi düzlemsel olarak göstermeye uygun ise çizgenin düzlemsel olarak gösterimi yapılacak.

```
for i in finaledges:
    x1=i[0]
    y1=i[1]
    x2=i[2]
    y2=i[3]
    G.add_node(tuple((x1, y1)))
    G.add_node(tuple((x2, y2)))
    G.add_edge((x1, y1), (x2, y2))
```

Yukarıdaki kod parçasıyla numpy ndarray dizisi aşağıda verilen graphml verisine dönüştürüldü. Yukarıda elde edilen bilgiler sayesinde çizgenin her bir ayrıtının başlangıç ve bitiş noktalarında bir tepe noktası olduğu biliniyor. Ayrıtların bulunduğu dizi ele alınarak dizinin her bir elemanının başlangıç ve bitiş noktalarının koordinatları için iki adet node graphml dosyasına eklendi. Yine ayrıtlar dizinin her bir elemanı için graphml dizisine bir adet ayrıt eklenerek resim dosyasında bulunan bir çizgenin veri kaybı olmadan orijinal halinin graphml veri formatına aktarılması sağlandı.

Aşağıdaki graphml dosyasında K_4 çizgesinin tepe noktalarının ve ayrıtlarının bilgileri bulunmaktadır.

```
<?xml version='1.0' encoding='utf-8'?>
<graphml
xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"><g
raph edgedefault="directed"><node id="(890.0, 230.0)"/>
<node id="(1120.0, 720.0)"/>
<node id="(210.0, 90.0)"/>
<node id="(90.0, 620.0)"/>
<edge   source="(890.0,   230.0)"   target="(1120.0,
720.0)"/>
<edge   source="(210.0,   90.0)"   target="(1120.0,
720.0)"/>
<edge   source="(210.0,   90.0)"   target="(890.0,
230.0)"/>
<edge   source="(90.0,   620.0)"   target="(1120.0,
720.0)"/>
<edge   source="(90.0,   620.0)"   target="(890.0,
230.0)"/>
<edge   source="(90.0,   620.0)"   target="(210.0,
90.0)"/>
</graph></graphml>
```

Yukarıda verilen graphml dosyasında çizgenin tepe noktalarının sayısı ve ayrıtlarının sayısı bilinmektedir. Eulerin düzlemsellik formülü kullanarak çizgenin düzlemsel olmadığının kontrolü yapıldı. Eğer G bir bağlantılı ve basit bir çizgeyse e G çizgesinin ayrıt sayısı ve v G çizgesinin tepe noktası sayısı olmak üzere;

- $v \geq 3$ ise
- $e \leq 3v - 6$

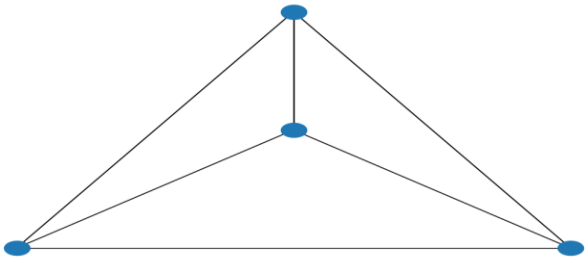
Formülü sağlanmalıdır. Bu formülü sağlamayan çizgelerin düzlemsel olmadığı kesinleştiği için direkt çizgenini düzlemsel gösteriliminin mümlün olmadığı bilgisi aktarıldı. Eğer bu formülü sağlıyorsa graphml verileri aşağıdaki fonksiyonla görselleştirildi.

```
def main(G):
    fig = plt.figure()
    graph = nx.Graph()

    for v in G.nodes():
        graph.add_node(v)

    for delta in G.edges():
        for w in delta:
            graph.add_edge(delta[0],delta[1])
    nx.draw_(graph)
    fig.savefig('./public/graph.png')
try:
  main(Ga)
except:
    ca = ' cp graph.png public/'
    cal=call(ca, shell=True)
cv2.imwrite("./public/line.png", cimageP)
```

Yukarıda fonksiyon graphml dosyasından dataları tek tek alıp çizgenin düzlemsel görünümünü elde etmektedir. Eğer çizgenin düzlemsel görünümü mümkün değer bir istisna yardımıyla fonksiyon sonlandırılarak çizgenin düzlemsel olmadığı ve bu yüzden düzlemsel gösteriminin mümkün olmadığı bilgisi aktarılır. Bu fonksiyona yukarıdaki şekil 4.5'ten elde edilen graphml giriş olarak verildiği zaman şekil 4.8'deki resim dosyası oluşmaktadır.



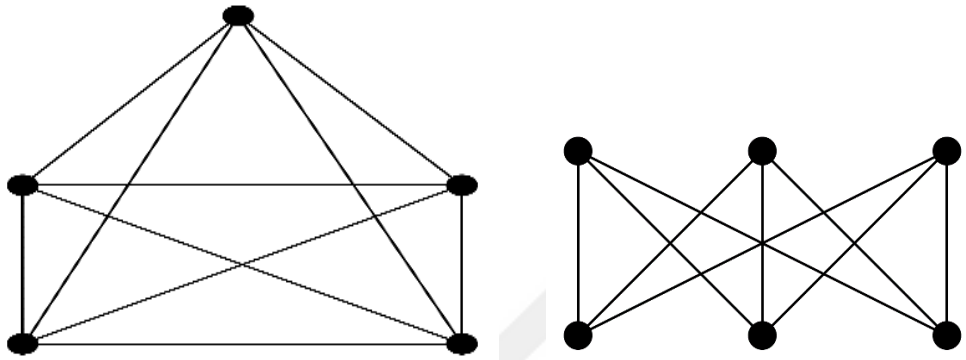
Şekil 4.8 Şekil 4.5'de Verilen Çizgenin Düzlemsel Gösterimi

4.5 Düzlemsel Olmayan Çizgenin Analizi

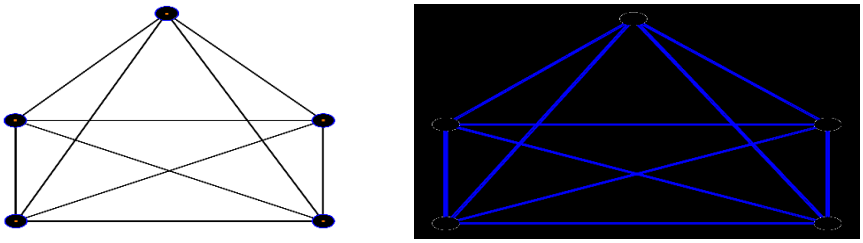
Programa alt çizgelerinden bir tanesi veya kendisi düzlemsel olmayan png dosyası yüklendiği senaryoda sistem çizgenin tepelerini ve ayrıtlarını görüntü işleme teknikleri kullanarak analiz etmektedir ve çizgeye ait graphml dosyasını

oluşturmaktadır. Fakat çizgenin düzlemsel gösterimi mümkün olmadığı için çizgenin düzlemsel gösteriminin mümkün olmadığı uyarısını kullanıcıya bilgi olarak vermektedir.

Şekil 4.9'da verilen çizgelerin analizleri yapıldıktan sonra aşağıdaki çıktılar sistem tarafından elde edildi.



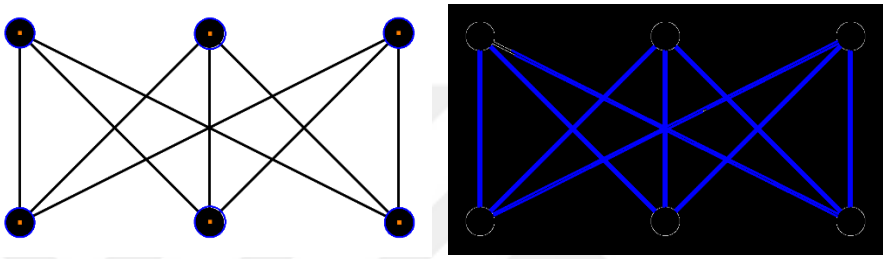
Şekil 4.9 K_5 Tam Çizgesi ve $K_{3,3}$ İki Parçalı Çizgesi



Şekil 4.10 K_5 Çizgesinin Tepelerinin ve Ayrıtlarının Tespiti

OpenCV kütüphanesi kullanılarak sisteme yüklenen K_5 çizgesinin tepeleri belirlendi. Koordinatlar kullanıcıya belirtilmek amacıyla resim üzerinde görüntü işleme teknikleri kullanıldı. Tespit edilen koordinatlar bir numpy dizisi içine eklenildi. Daha sonra belirlenen ayrıtların bu numpyarray noktaları kullanılarak optimize edilmesi için veriler uygun hale getirildi. Şekil 4.11'de sağdaki görselde çizgenin ayrıtlarının belirlenmesi png formatındaki dosya üzerinde

canny metodu çalıştırıldı. Daha yalın bir formata dönüştürülen png dosyası daha sonra sınır belirleme algoritması kullanılarak analiz edildi Belirlenen ayrıtların hepsi bir numpy dizisine eklendi. Daha sonra tepe koordinatları ve optimizasyon algoritmaları kullanılarak herhangi iki tepe arasında bulunan ayrıtlar ayrı bir veri olarak elde edildi. Elde edilen ayrıtlar ve tepeler kullanılarak bir grapml dosyası oluşturuldu. Daha sonra graphml dosyasında temsil edilen çizgeye düzlemsellik testi yapıldı ve düzlemsel bir çizge olmadığı için düzlemsel gösterim elde edilemedi. Kullanıcıya bununla ilgili bilgi verildi.



Şekil 4.11 $K_{3,3}$ Çizgesinin Tepelerinin ve Ayrıtlarının Tespiti

Şekil 4.11 soldaki görselde $K_{3,3}$ çizgesinin tepeleri belirlenmiştir sağdaki görselde ise $K_{3,3}$ çizgesinin ayrıtları HoughLine yöntemiyle belirlenmiştir.

```
<?xml version='1.0' encoding='utf-8'?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"><graph
edgedefault="directed"><node id="(94.0, 1330.0)"/>
<node id="(92.0, 730.0)"/>
<node id="(1214.0, 730.0)"/>
<node id="(1214.0, 1330.0)"/>
<node id="(644.0, 92.0)"/>
<edge source="(94.0, 1330.0)" target="(92.0, 730.0)"/>
<edge source="(1214.0, 730.0)" target="(92.0, 730.0)"/>
<edge source="(1214.0, 730.0)" target="(94.0, 1330.0)"/>
<edge source="(1214.0, 1330.0)" target="(92.0, 730.0)"/>
<edge source="(1214.0, 1330.0)" target="(94.0, 1330.0)"/>
<edge source="(1214.0, 1330.0)" target="(1214.0, 730.0)"/>
```

```
<edge source="(644.0, 92.0)" target="(92.0, 730.0)"/>
<edge source="(644.0, 92.0)" target="(94.0, 1330.0)"/>
<edge source="(644.0, 92.0)" target="(1214.0, 730.0)"/>
<edge source="(644.0, 92.0)" target="(1214.0, 1330.0)"/>
</graph></graphml>
```

Yukarıdaki graphml formatındaki dosya K_5 çizgesini içeren bir png dosyasını analiz ettikten sonra elde edilen veriler görüntü işleme yöntemleriyle tepeleri ve ayrıtları belirlenip yukarıdaki formata dönüşümü sağlandı.

```
<?xml version='1.0' encoding='utf-8'?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"><graph
edgedefault="directed"><node id="(1350.0, 802.0)"/>
<node id="(1340.0, 142.0)"/>
<node id="(90.0, 94.0)"/>
<node id="(90.0, 772.0)"/>
<node id="(610.0, 774.0)"/>
<node id="(600.0, 132.0)"/>
<edge source="(1350.0, 802.0)" target="(1340.0, 142.0)"/>
<edge source="(1350.0, 802.0)" target="(90.0, 94.0)"/>
<edge source="(90.0, 772.0)" target="(1340.0, 142.0)"/>
<edge source="(90.0, 772.0)" target="(90.0, 94.0)"/>
<edge source="(610.0, 774.0)" target="(1340.0, 142.0)"/>
<edge source="(610.0, 774.0)" target="(90.0, 94.0)"/>
<edge source="(600.0, 132.0)" target="(1350.0, 802.0)"/>
<edge source="(600.0, 132.0)" target="(90.0, 772.0)"/>
<edge source="(600.0, 132.0)" target="(610.0, 774.0)"/>
</graph></graphml>
```

Yukarıdaki graphml formatındaki dosya $K_{3,3}$ çizgesini içeren bir png dosyasını analiz ettikten sonra elde edilen veriler görüntü işleme yöntemleriyle tepeleri ve ayrıtları belirlenip yukarıdaki formata dönüşümü sağlandı.

5. TARTIŞMA VE SONUÇ

Çizge teorisi matematikten fiziğe bilgisayar bilimlerinden enerji sistemlerine hatta günümüz teknolojik ürünlerinin PCB kartlarının tasarımının modellenmesine kadar çok geniş alanlarda kullanılan optimizasyonu yapılan bir konu olduğu için önümüzdeki yıllarda öneminin daha da artacağı öngörülmektedir. Özellikle PCB kart tasarımı ve enerji dağıtım sistemlerinde bir çizgenin düzlemsel gösterimi büyük öneme sahip oldu için bu konudaki soruna bir çözüm getirecek çalışma gerçekleştirilmiştir. Çalışma sonucunda çizge teorisinin bir alt konusu olan düzlemsel çizgeler ve çizgelerin düzlemsel gösterimine yönelik önemli katkılar sağlayacak bir çalışma gerçekleştirilmiştir.

Günümüz bilgisayarlar ve görüntü kaydetme aygıtlarının donanımlarının gelişmesiyle birlikte görüntülerin işlenmesi ve anlamlandırılması adına büyük gelişmeler sağlandı. Bu çalışma kapsamında da alınan png dosyası formatındaki bir görüntünün görüntü işleme algoritmalarıyla çizge modelinin belirlenmesi hususunda önemli konular çözüme kavuşturuldu. Png dosyasındaki çizgenin tepelerinin hatasız olarak belirlenmesi için gerekli geliştirmeler yapıldı. Uygun formatta çizge içeren bir png dosyasının tepelerin koordinatları hatasuz olarak belirlendi. Tepe bazlı bir yaklaşım kullanarak iki tepe arasındaki ayrıtların hatalardan arındırılması için optimizasyon yapıldı OpenCV görüntü işleme algoritmaları çizgelerin tepe noktalarını tanımada başarılı sonuçlar vermektedir fakat ayrıtları belirlemek için kullanılan algoritmalar yüksek hata oranıyla çalışmaktaydı. Bu çalışma kapsamında ayrıtların belirlenmesi sırasında oluşan hatalara spesifik çözümler getirmiştir.

Çalışmanın ikinci fazında graphml moduna dönüştürülen çizgenin türünün belirlenmesi için çizge belirleme algoritmaları çalıştırıldı. Çizgenin düzlemsellik kontrolü yapıldı. Son aşamada eğer çizge düzlemsel bir çizge olarak ifade edilebiliyorsa çizgenin düzlemsel olarak png formatında gösterilmesi sağlandı. Bu çalışma sırasında matplotlib kütüphanesinden faydalanıldı. Matplotlib kütüphanesi graphml veri formatında verilen çizgeleri görselleştirme konusunda oldukça başarılıdır fakat png dosyaları için bir desteği bulunmamaktadır. Bu çalışmada png dosyasından görüntü işleme yoluyla belirlenen niteliklerin matplotlib kütüphanesinin işlem yapabileceği graphml veri formatına dönüşümü sağlanmıştır.

KAYNAKLAR DİZİNİ

Bollobas, B., 1998. Modern Graph Theory. Springer Science + Business Media, Inc,New York.

Bondy, J.A., Murty, U.S.R., 2008, Graph Theory, Springer., New York.

Burden, J., Cleland, M., "Tracking a single cyclist during a team changeover on a velodrome track with Python and OpenCV", https://www.sciencedirect.com/science/article/pii/S1877705810003449 (Erişim Tarihi: 20 Eylül 2022)

Büyükköse, Ş., Kaya, G.K., 2019, Çizge Teoriye Giriş. Nobel Yayınları, Ankara.

Diestel, R., 2017. Graph Theory, Fifth Edition (Graduate Texts in Mathematics, 173). Springer, 448p, Berlin.

Dominguez, C., Heras, J., "IJ-OpenCV: Combining ImageJ and OpenCV for processing images in biomedicine", https://www.sciencedirect.com/science/article/abs/pii/S0010482517300823 (Erişim Tarihi: 05 Mayıs 2020)

Hoste, J., Shanahan., P.D., 2001. Trace fields of twist knots. Journal of Knot Theory and Its Ramifications, 10(4), 625–639.

Kaehler, A., Bradski, G., 2007. Learning OpenCV 3, O'Reilly Media

Kelmans, A.K., 1981. The concept of a vertex in a matroid, the non-separating cycles in a graph and a new criterion for graph planarity. In: Algebraic Methods in Graph Theory, Issue 25, Vol. 1. Lovasz, L. and Sos, V. T. (eds.). North-Holland, pp. 345–388, Amsterdam.

Kuratowski, K., 1930. Sur le probleme des courbes gauches en Topologie. Fundamenta Mathematicae, 15(1): 271-283.

MacLane, S., 1937. A combinatorial condition for planar graphs. Fundamenta Mathematicae, 28: 22–32.

Murasugi, K., 1989. Invariants of Graphs and Their Applications to Knot Theory. Algebraic Topology Poznan, pp. 83–97 and (1991). Lecture Notes in Mathematics, Vol. 1474, Springer, Berlin, Heidelberg.

Rosen, K., 2012. Discrete Mathematics and its Applications", Mc Graw Hill.

Schnyder, W., 1989. Planar graphs and poset dimension. Order 5: 323–343.

Sirkov, S., Novikov, L.,"The algorithm development for operation of a computer vision system via the OpenCV library", https://www.sciencedirect.com/science/article/pii/S1877050920303161 (Erişim Tarihi 20 Aralık 2022)

Stillwell, J., 1993. Classical topology and combinatorial group theory (2nd ed.). Springer, 348 pages, Berlin.

TEŞEKKÜR

Bu çalışmamın her safhasında ve her anında, manevi desteklerini her an hissettiğim ve hiç eksik etmeyen aileme, bilgi ve tecrübesiyle bana yön veren, beni destekleyen ve bu çalışmamda benden tüm yardımlarını esirgemeyen sayın hocam Prof. Dr. Vecdi AYTAÇ'a ve her zaman yanımda olan arkadaşlarıma çok teşekkür ederim.