Programlama Labaratuvarı 1.Proje Raporu

Meryem AKARSU

meryemakarsu1301@gmail.com

Nuray KARABULUT

nuraykarabulut95@gmail.com

I. PROJE AMACI VE ÖNEMİ

Bu projenin temel amacı, LIDAR sensörlerinden elde edilen ham mesafe verilerini işleyerek çevredeki doğrusal yüzeyleri tespit etmek, bu doğrular arasındaki kesişimleri analiz etmek ve robotik sistemlerin çevresel farkındalığını artırmaktır. Böylece sensör verileri yalnızca bir sayı dizisi olmaktan çıkarak, çevrenin geometrik yapısını temsil eden anlamlı bilgiye dönüştürülür.

Bu sistem, özellikle otonom kara araçlarında, insansız hava araçlarında ve endüstriyel robotlarda çevre haritalaması, engel tespiti ve rota planlama süreçlerinde kullanılabilecek bir altyapı sağlar. Doğrusal yapıların doğru tespiti, aracın çevresindeki duvarları, yolları veya nesneleri güvenli şekilde tanımlamasını mümkün kılar. Böylelikle, sistem çevredeki fiziksel engelleri yorumlayarak daha güvenli ve verimli hareket stratejileri geliştirebilir.

Günlük yaşamda bu yaklaşım, akıllı şehirlerdeki otonom ulaşım sistemlerinden depo otomasyonlarına kadar geniş bir kullanım alanına sahiptir. LIDAR tabanlı bu yöntem, makinelerin çevreyi "görmesini" ve geometrik olarak "anlamlandırmasını" sağlayarak güvenli, kararlı ve akıllı hareket kararları almalarına katkı sunar.

Kısaca, proje sensör verisini geometriye dönüştürerek makinelerin çevresel farkındalığını insan benzeri algı düzeyine yaklaştırmayı hedeflemektedir.

II. KULLANILAN ALGORİTMALAR VE KODA ENTEGRASYONU

1-) Veri Ön İşleme (LIDAR Nokta Seçimi ve Filtreleme):

LIDAR sensöründen elde edilen x–y koordinat verileri öncelikle gürültüden arındırılmıştır. Bunun için belirli menzil dışı veya yoğunluk bakımından düşük nokta kümeleri filtrelenmiştir. Bu adım, hatalı tespitleri azaltarak doğruların daha kararlı belirlenmesini sağlar.

2-)Doğru Tespiti (RANSAC destekli, Hough Benzeri Parametrik Yaklaşım):

Doğruların tespiti için klasik Hough dönüşümüne benzer bir yöntem uygulanmıştır; ancak bu projede parametre uzayı yerine **örnek tabanlı bir yaklaşım** tercih edilmiştir.
Öncelikle, rastgele seçilen nokta çiftlerinden olası doğrular oluşturulmuş, ardından **RANSAC (Random Sample Consensus)** algoritması kullanılarak bu doğruların ne kadarının mevcut nokta bulutuyla uyumlu olduğu test edilmiştir.

RANSAC, veri kümesindeki olası gürültüye rağmen en iyi uyumu sağlayan modeli (doğru denklemini) seçerek robust bir tespit sağlar. Her yinelemede:

- Rastgele bir alt kümeden doğrusal model tahmini yapılır,
- Bu modele yakın düşen noktalar "inlier" olarak etiketlenir,
- En fazla inlier içeren model nihai doğru olarak belirlenir.

Bu süreç, hatalı ölçümlerin (outlier) etkisini büyük ölçüde azaltır.
Elde edilen her doğru, eğim–kesişim formunda ($y = mx + b$) temsil edilmiştir ve segment uç noktaları (x_1, y_1)–(x_2, y_2) olarak kaydedilmiştir.
Tespit edilen tüm doğrular, `detectedLines` veri yapısında saklanmış ve görselleştirme modülüne aktarılmıştır.

3-)Kesişim Analizi ve Açı Hesabı:

Her doğru çifti arasındaki kesişim noktası analitik olarak hesaplanmış, ardından aralarındaki açı `atan2` fonksiyonu ile belirlenmiştir. Elde edilen kesişim noktalarının konumu (x, y) ve açısı (radyan cinsinden) `Intersection` yapısında saklanmıştır.

Bu adım, özellikle 60° üzeri kesişimlerin belirlenmesine olanak sağlar. Bu açı eşiği, çevredeki "keskin dönüş" veya "engel köşesi" gibi kritik noktaları tanımak için seçilmiştir.

Koda entegrasyon sürecinde bu algoritmalar modüler C++ sınıfları halinde uygulanmış, görselleştirme için ImPlot kütüphanesi kullanılmıştır. Hesaplamalar tamamlandıktan sonra, `intersections` dizisindeki noktalar doğrudan grafik ortamına aktarılır ve renklendirme/etiketleme dinamik olarak yapılır.

III. KULLANILAN KÜTÜPHANELER

<cmath>: Trigonometrik (atan2, sin, cos) ve geometrik hesaplamalar (sqrt, pow) için kullanılmıştır. Özellikle doğru denklemleri, açı farkı ve mesafe hesaplamaları bu kütüphane üzerinden yapılmıştır.

- **<vector> ve <float.h>:** Dinamik veri saklama, minimum–maksimum mesafe hesaplamaları ve kesişim noktalarının yönetimi için kullanılmıştır. `std::vector`

yapısı, doğrular ve kesişim noktalarının esnek biçimde tutulmasını sağlamıştır.

- **<string> ve <fstream>:** TOML dosyalarından LIDAR verilerini okumak, satırları parçalamak ve metin tabanlı veri işlemek için kullanılmıştır. Bu sayede sensörden gelen ham veriler doğrudan C++ içinde işlenebilir hale getirilmiştir.

- **<sstream>:** Metin tabanlı sayısal ayrıştırma (örneğin `"12.45"` → `float`) işlemlerinde kullanılmıştır. Dosya girişinden gelen veriler `stringstream` aracılığıyla doğru tür dönüşümüne tabi tutulmuştur.

- **<curl/curl.h>:** (isteğe bağlı modül) Harici kaynaklardan veri çekmek için eklenmiştir. Özellikle, uzak sunucuda barındırılan TOML veya LIDAR veri dosyalarına erişim bu kütüphane ile gerçekleştirilmiştir.

- **ImGui / ImPlot:** Gerçek zamanlı grafik arayüz ve 2D görselleştirme işlemleri için kullanılmıştır. Bu kütüphaneler, tespit edilen doğruların ve kesişim noktalarının etkileşimli biçimde çizilmesini sağlamış, kullanıcıya parametreleri dinamik olarak değiştirme imkânı sunmuştur.

- **<utility> ve <algorithm>:** Doğru segmentlerinin birleştirilmesi, sıralanması ve filtrelenmesi gibi temel veri manipülasyonlarında kullanılmıştır. `std::min`, `std::max`, `std::sort` ve `std::find_if` gibi işlevlerle kodun verimliliği artırılmıştır.

- **<iostream>:** Konsol çıktısı, hata kontrolü ve debug aşamasında verilerin izlenmesi için kullanılmıştır.

-

IV.SİSTEM MİMARİSİ VE GENEL YAPI ŞEMASI

Geliştirilen sistem, LIDAR sensöründen gelen ham mesafe verilerini alarak bunları anlamlı geometrik bilgilere dönüştüren çok katmanlı bir yazılım mimarisine sahiptir. Sistem mimarisi genel olarak dört temel bileşenden oluşur: **veri alımı**, **veri işleme**, **analiz** ve **görselleştirme**.

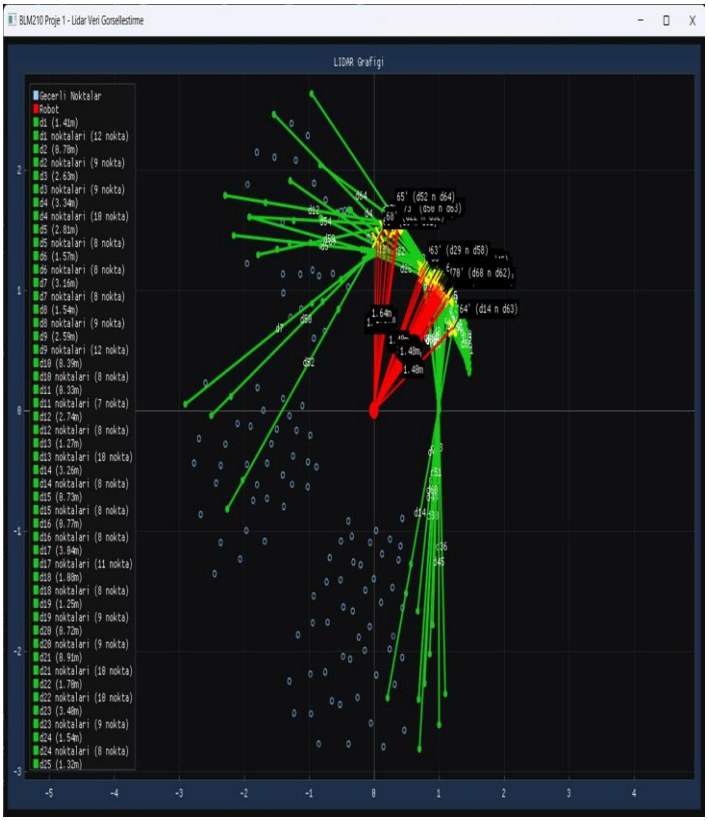
İlk aşamada, sistem LIDAR sensöründen veya harici bir dosya kaynağından alınan menzil verilerini ham biçimde okur. Bu veriler tipik olarak açı–mesafe (polar) formatındadır. Veriler, `polarToCartesian()` fonksiyonu kullanılarak Kartezyen koordinat sistemine dönüştürülür. Ardından, belirli menzil sınırlarının dışında kalan veya gürültü niteliğindeki noktalar filtrelenerek temiz bir nokta bulutu elde edilir.

İkinci aşamada, **doğru tespiti modülü** devreye girer. Bu modül, RANSAC tabanlı bir yaklaşım kullanarak nokta bulutundaki doğrusal yapıların parametrelerini belirler. Gürültüye dayanıklı bu yöntem sayesinde, veri setinde yer alan gerçek duvar, kenar veya sınır çizgileri doğru biçimde modellenir. Her doğru segmenti iki uç noktasıyla (x_1, y_1)–(x_2, y_2) temsil edilir ve bu doğrular sistemin `detectedLines` veri yapısında tutulur.

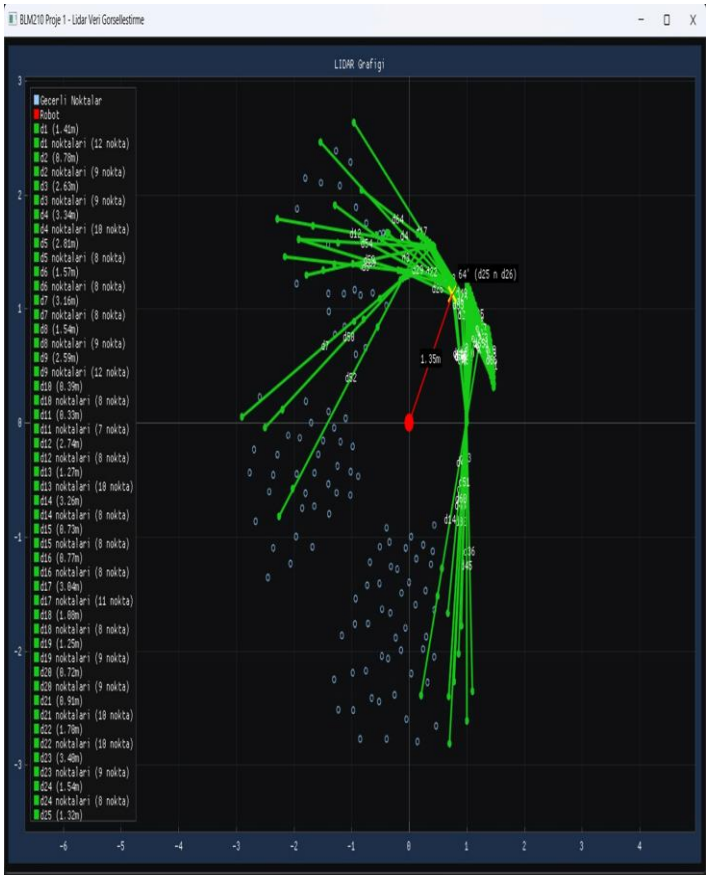
Üçüncü aşamada, **kesişim analizi modülü** devreye girer. Bu modül, tespit edilen her doğru çifti arasında analitik kesişim hesabı yapar. Hesaplanan kesişim noktalarının konumu ve doğrular arası açı farkı atan2 fonksiyonu ile bulunur. Ardından, yalnızca belirli bir eşik değeri (örneğin 60°) üzerindeki kesişimler seçilerek çevredeki kritik geometrik noktalar (köşeler, dönüşler, duvar birleşimleri vb.) belirlenir.

Son aşamada, **görselleştirme katmanı**, elde edilen tüm doğruları ve kesişim noktalarını **ImPlot** ve **ImGui** kütüphaneleri aracılığıyla grafik arayüzde gösterir. Kullanıcı, bu arayüz üzerinden analiz edilen noktaları, açıları ve mesafeleri etkileşimli olarak gözlemleyebilir. Bu sayede sistem hem analiz aracı hem de eğitim/gösterim amaçlı bir platform olarak kullanılabilir.

Bu mimari, modüler yapısı sayesinde farklı sensör türleri veya veri formatlarıyla kolayca uyarlanabilir. Her katman bağımsız olarak çalışabildiği için sistem, gerçek zamanlı robotik uygulamalara veya simülasyon tabanlı test ortamlarına entegre edilebilir.



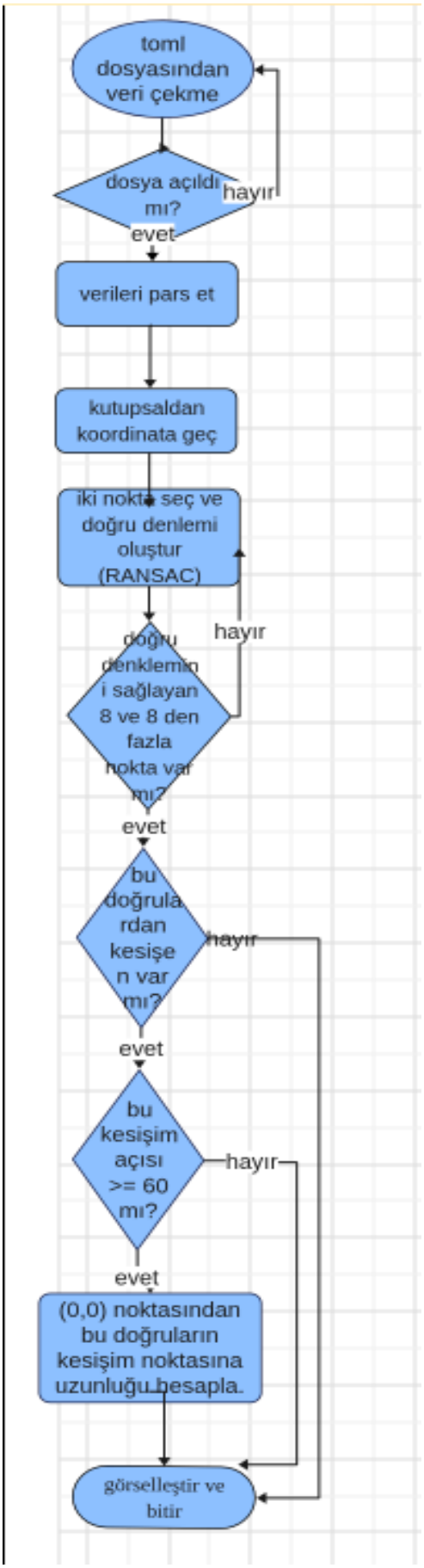
şekil-1



şekil-2

Projede temel akış, LIDAR verilerinden elde edilen tüm kesişim noktalarının analiz edilmesi üzerine kurulmuştur. Ancak sistemi uygulama aşamasında, yalnızca tüm kesişimleri göstermek yerine robot konumuna en yakın kesişimi belirlemenin çok daha işlevsel olduğunu fark ettik. Bu ek adım aslında proje gereklilikleri arasında yoktu, fakat sistemin karar verme yeteneğini artırmak ve kullanıcı müdahalesini ortadan kaldırmak için özellikle ekledik.

Bu sayede sistem artık yalnızca çevredeki tüm kesişimleri göstermekle kalmıyor, aynı zamanda **robot için en anlamlı ve kritik noktayı — yani en kısa mesafedeki kesişimi —** otomatik olarak işaretliyor. Böyle bir geliştirme, sistemi pasif bir analiz aracından çıkarıp, çevresine tepki verebilen akıllı bir yapıya dönüştürdü. Şekil 2'de, bu iyileştirilmiş sürümün yalnızca en yakın kesişimi vurgulayan görselleştirmesi yer almaktadır.



V.SONUÇ

Proje sonucunda, LIDAR verilerinden elde edilen doğruların görsel olarak başarılı şekilde temsil edildiği, kesişim noktalarının ve açılarının dinamik olarak hesaplanabildiği bir sistem geliştirilmiştir.

Bu sistem, gelecekteki otonom navigasyon yazılımlarına doğrudan entegre edilebilecek düzeydedir.
Özellikle görsel çıktı ve geometrik analiz bir arada sunulduğu için, hata ayıklama ve çevre algısı testlerinde önemli bir avantaj sağlamaktadır.
Sonraki aşamalarda bu yapı, gerçek zamanlı LIDAR akışıyla çalışacak şekilde genişletilebilir; örneğin robotlarda, keskin dönüşleri önceden tahmin ederek yönelimini proaktif biçimde değiştirebilir.

VI. YAZAR KATKILARI

Bu proje, ekip üyeleri Meryem ve Nuray'ın ortak emeğiyle yürütülmüştür. Projenin tüm aşamalarında — dosya okuma ve veri işleme, algoritma geliştirme, matematiksel modelleme, kod entegrasyonu, görselleştirme ve rapor hazırlığı süreçlerinde — iki ekip üyesi birlikte çalışmış, fikir alışverişi ve ortak kararlarla ilerlemiştir. Çalışma boyunca ekip ruhu korunmuş, her adım karşılıklı katkı ve iş birliğiyle tamamlanmıştır.

VII. KAYNAKÇA

https://www.thinkautonomous.ai/blog/ransac-algorithm/https://www.baeldung.com/cs/ransac?utm_

https://github.com/ocornut/imgui/wiki/Getting-Started?utm_#setting-up-dear-imgui--backends

https://www.youtube.com/playlist=PLh9ECzBB8tJOS7WQKdeUaAa5fmPLYAouD

https://chatgpt.com