T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİNLİK KAROLARI İLE GÖRSEL ODOMETRİ

Nihal ALTUNTAŞ

DOKTORA TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Danışman
Doç. Dr. Mehmet Fatih AMASYALI

Aralık, 2019

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİNLİK KAROLARI İLE GÖRSEL ODOMETRİ

Nihal ALTUNTAŞ tarafından hazırlanan tez çalışması 25.12.2019 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programı **DOKTORA TEZİ** olarak kabul edilmiştir.

Doç. Dr. Mehmet Fatih AMASYALI
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Doç. Dr. Mehmet Fatih AMASYALI, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Sırma YAVUZ, Üye
Yıldız Teknik Üniversitesi _____

Doç Dr. Hatice KÖSE, Üye
İstanbul Teknik Üniversitesi _____

Dr. Öğr. Üyesi Erkan USLU, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Gökhan İNCE, Üye
İstanbul Teknik Üniversitesi _____

Danışmanım Doç. Dr. Mehmet Fatih AMASYALI sorumluluğunda tarafımca hazırlanan Derinlik Karoları ile Görsel Odometri başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Nihal ALTUNTAŞ

İmza

Bu çalışma Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK-BİDEB-2211/E Programı) ve Yıldız Teknik Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi (Proje No: FDK-2017-3125) tarafından desteklenmiştir.

Aileme ithafen

TEŞEKKÜR

Çalışmam boyunca yaşadığım bütün iniş çıkışlara rağmen yaptığı yapıcı eleştiriler ve değerli yorumlar ile bana yön veren danışmanım Doç. Dr. Mehmet Fatih AMASYALI hocama sonsuz teşekkür ederim. Ayrıca bana bu çalışmayı yapma fırsatı sağlayan Olasılıksal Robotik Grubu'na içten şükranlarımı sunarım.

Gösterdikleri anlayış ve sabır sayesinde tamamlamış olduğum tez sürecim boyunca bana en büyük desteği veren aileme de teşekkürü bir borç bilirim.

Son olarak, çalışmama BİDEB-2211/E Programı kapsamında destek veren Türkiye Bilimsel ve Teknolojik Araştırma Kurumu TÜBİTAK'a ve FDK-2017-3125 proje numarasıyla destek veren Yıldız Teknik Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi'ne teşekkür ederim.

Nihal ALTUNTAŞ

İÇİNDEKİLER

KISALTMA LİSTESİ

3B	3 Boyutlu
BA	Bundle Adjustment
BOW	Bag-Of-Words
BRIEF	Binary Robust Independent Elementary Features
CPU	Central Processing Unit
DoF	Degree of Freedom
FAST	Features from Accelerated Segment Test
FeDe	Feature Detector
GFTT	Good Features to Track
Hz	Hertz
ICP	Iterative Closest Point
kd-tree	k dimensional tree
LCC	Loop Closure Constraints
LSD-SLAM	Large-Scale Direct Monocular SLAM
LTM	Long-Term Memory
m	Metre
NNS	Nearest Neighbor Strategy
OdSt	Odometry Strategy
ORB	Oriented FAST and rotated BRIEF
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RGB-D	Red Blue Green - Depth
RMSE	Root Mean Square Error

ROS	Robot Operating System
RPE	Relative Pose Error
RTAB-Map	Real-Time Appearance-Based Mapping
s	Saniye
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SM	Sensory Model
STM	Short-Term Memory
SURF	Speeded Up Robust Features
TORO	Tree-based netwORk Optimizer
TSDF	Truncated Signed Distance Function
VWT	Visual Word Type
WM	Working Memory

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Derinlik Karoları ile Görsel Odometri

Nihal ALTUNTAŞ

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Doç. Dr. Mehmet Fatih AMASYALI

Düşük maliyetli, 3 boyutlu sensörlerin yaygınlaşmasıyla literatürde başarılı 3B haritalama yapan SLAM sistemleri geliştirilmiştir. Ancak, bu sistemlerin genel problemi tek renk duvarlar ya da uzun koridorlar gibi yetersiz öznitelik içeren ortamlarda sıkıntı yaşamalarıdır. 3 boyutlu haritalama yapılan ortamlarda çoğu alan RGB resimden öznitelik çıkarımı için yeterli dokuya sahip olsa da haritalanan her ortamın özniteliksiz durumlar içermesi mümkündür.

Bu tez, özniteliksiz ortamlarda odometri kaybını azaltmak için öznitelik tespitinde nesnelerin (örn. duvarlar, engeller) sadece renk bilgisinden değil aynı zamanda oluşturulan harita içerisindeki konumlarından da faydalanılmasına yönelik yeni bir yaklaşım getiren DepthTiling isimli sistemi sunmaktadır. Önerilen sistem, alınan RGB resmi, RGB-D kamera tarafından sağlanan ilgili derinlik verisi ile yeniden renklendirmektedir. Geliştirilen sistem üzerinde yapılan testler neticesinde, görsel SLAM sistemlerinin sahip olduğu odometri takip kabiliyetinin arttırılması noktasında umut vadeden sonuçlar elde edilmiştir. Bu çalışma ile RGB resmin yetersiz kaldığı durumlarda ilgili derinlik verisi kullanılarak öznitelik sayısının arttırılabileceği gösterilmiştir.

Mevcut sistemlerin çalışma prensipleri göz önüne alındığında, RGB resim verisine olan bağımlılıkları yüzünden belirli ortam şartlarında başarılı olamadıkları gözlemlenmiştir. Tez kapsamında, 3 boyutlu haritalama sistemlerindeki bu bağımlılığının azaltılması ile daha başarılı sonuçlar elde edilebilmesi mümkün hale getirilmiştir. Böylece mevcut sistemlerin istenilen haritalama sonucunu veremediği ortamlar için dahi başarılı bir 3 boyutlu haritalama yapılabilmektedir.

Anahtar Kelimeler: Görsel odometri, öznitelik eşleme, RGB-D sensörler, 3B SLAM

ABSTRACT

Visual Odometry via Depth Tiling

Nihal ALTUNTAŞ

Department of Computer Engineering
Doctor of Philosophy Thesis

Advisor: Assoc. Prof. Dr. Mehmet Fatih AMASYALI

Although various successful visual SLAM systems have been developed since the invention of low-cost 3-dimensional sensors, the common problem with those systems is that all suffer from featureless environments like one-colored walls, long halls etc. It is possible for all environments to have such featureless situations; even though most areas in mapped environments contain sufficient texture for feature extraction from RGB images.

This thesis proposes a system named DepthTiling which brings a new approach using not only RGB values of the objects (e.g., walls, obstacles) but also their positions in the generated map for feature extraction in order to decrease odometry loss in featureless environments. The proposed system recolors gathered RGB image using associated depth data provided by RGB-D camera. The experiments give promising results to increase odometry tracking capability of visual SLAM systems. This study shows that it is possible to increase number of features using related depth data when RGB images are insufficient.

Considering the operating principles of the existing systems, it has been observed that they can not be successful under specific environmental conditions due to their dependences on RGB image data. Within the scope of this thesis, it has been made possible to achieve more successful results by reducing this dependency in 3-dimensional mapping systems. Thus, a successful 3-dimensional mapping can be performed even in environments where existing systems cannot give the desired mapping results.

Keywords: Visual odometry, feature matching, RGB-D sensors, 3D SLAM

1
Giriş

Teknolojinin gelişmesiyle her geçen gün kullanımı yaygınlaşan robotların popülerliği arttıkça bu alanda çalışma yapılması gereksinimi de paralel olarak artmaktadır. İlk zamanlar, sadece sabit bir platform üzerinde belirli bir çalışma alanına sahip robotlar kullanılırken; öncelikle tekerlek, sonrasında pervane ile robotların sırasıyla yerde ve havada mobil olarak hareket edebilmesi mümkün hale gelmiştir. Bu durumda, robotların çalışma alanının sabit olarak belirlenmesi uygulanabilir olmadığından mobil robotun çalıştığı, başka bir ifadeyle gezindiği ortamı algılayarak tanıması zorunluluk oluşturmaktadır.

Günümüz robot sistemlerinde mobil robotların ek olarak otonom özelliğe sahip olması bir çok uygulama alanı için beklenti haline gelmiştir. Otonom gezinimin başarılı olabilmesi için ise düzgün bir haritanın oluşturulması ve mobil robotun başarılı bir odometri (konum) takibi yapması ön koşul niteliğindedir. Bu nedenle, bilinmeyen bir ortamda gezinimi sırasında, robotun çözmesi beklenenen iki temel problemin gezilen ortamın haritasının çıkarılması ve çıkarılan bu harita içerisinde robotun kendini konumlandırması olduğu söylenebilir.

Bu iki problemin çözümü için geliştirilmiş olan Eş zamanlı Lokalizasyon (Konumlandırma) ve Haritalama (SLAM - Simultaneous Localization and Mapping) [1] sistemleri 2 boyutlu haritaya ihtiyaç duyan yer robotları için yüksek oranda başarı sağlamaktadırlar. Ancak, 2 boyutlu haritaların masa ve sandalye gibi dikey düzlemde süreklilik arz etmeyen nesnelerin haritada gösteriminde yetersiz kalmaları ve hava robotlarının yaygınlaşması, 3 boyutlu (3B) SLAM sistemlerine olan ihtiyacı da arttırmıştır [2]. Bununla birlikte; düşük bütçeli, 3 boyutlu veri sağlayan sensörlerin geliştirilmesi bu ihtiyacın giderilmesine yönelik çalışmaların hızlanmasını sağlamıştır. Özet olarak, robotlara otonom gezinim özelliğinin kazandırılması konusu, yeni olmamakla birlikte, dronlar gibi 6 serbestlik derecesine (DoF - Degree of Freedom) sahip mobil robotların yaygınlaşması ve sensör teknolojisindeki gelişmeler dolayısıyla popülerliğini arttırarak farklı bir boyut kazanmıştır.

3B SLAM kapsamında yapılan bu çalışmayı anlatan okuduğunuz tez, şu şekilde organize edilmiştir. Bölüm 1.1 tezin amaç ve konusunu, Bölüm 1.2 bu alanda yapılmış olan ve literatürde bulunan çalşmaların detaylarını, Bölüm 1.3 ise tez kapsamında ileri sürülen hipotezi vermektedir. Bölüm 2, mevcut sistemlerin eksiklerinin çıkarılması ve literatürde bulunan ihtiyacın belirlenmesi için yapılmış olan ön çalışmayı anlatmaktadır. Bölüm 3 ile mevcut 3B SLAM sistemlerinin detaylı incelemesi yapılarak farklı ortam ve parametrelerin çalışma performanslarına etkisi verilmektedir. Bölüm 4, 3B SLAM sistemlerinin farklı ortam ve şartlarda performans karşılaştırması yapmaktadır. Bölüm 5 öncelikli olarak mevcut sistemlerde tespit edilen eksiklikler ve bu eksikliklerin giderilmesine yönelik ileri sürülen hipotezin ayrıntılarını sonrasında ise bu hipotezin uygulamasının detaylarını içermektedir. Bölüm 6, uygulanan sistemin deneysel sonuçlarını vermektedir. Son olarak, Bölüm 7 ise tezin sonucunda elde edilen kazanımları ve ileride yapılabilecek iyileştirmeleri açıklamaktadır.

1.1 Tezin Amacı

3B SLAM sistemlerinde, 2 boyutlu sistemlerde olduğu gibi, mobil bir robot bilinmeyen bir ortamda gezinim yaparak aynı anda hem ortam haritası oluşturmakta hem de kendini gezindiği ortamda konumlandırmaktadır [1, 2]. SLAM algoritmaları döngü halinde iki adımın tekrar edilmesini içermektedir; okunan sensör verisine göre harita güncellemesi ve yapılan hareket sonrası konumlandırmanın tekrar yapılması. Düzgün bir harita oluşturulabilmesi için konumlandırmadaki başarı önem arz etmektedir ve konumlandırmadaki başarı ise yapılan hareket miktarının ne kadar iyi hesaplanabildiğine bağlıdır. Bu nedenle, son yıllarda robotik alanında odometri takibi popüler bir konu haline gelmiştir.

Hem haritalama hem de lokalizasyon adımları için renk bilgisinin yanı sıra derinlik verisi de sağlayan (RGB-D - RedBlueGreen-Depth) sensörlerden gelen veriyi kullanan görsel SLAM en yaygın 3B SLAM tekniklerinden biridir. Bu teknik, robotların kameralarından gelen görüntüler üzerinden öznitelik çıkarımı ve eşlemesi yöntemlerini kullanır. RGB-D sensörden alınan RGB resim üzerinde çıkarılan öznitelikler, haritalama için 3 boyutlu harita üzerinde bulunanlarla eşleştirilirken lokalizasyon için eşleştirme bir önceki resim üzerinde çıkarılanlarla yapılır. Harita ve konum güncellemeleri, eşleşen özniteliklerin derinlik bilgileri kullanılarak yapılır. Örneğin, odometri takibi, peş peşe gelen iki RGB resim üzerinde eşleşen öznitelik noktalarına ait derinlik bilgisindeki değişimi kullanarak robotun hareket miktarının hesaplanmasıyla yapılır.

Mevcut görsel SLAM sistemleri, RGB resimlerinin yeterli öznitelik noktalarına sahip olduğu ortamlarda düzgün sonuçlar verse de, öznitelik noktalarının yetersiz olduğu durumlarda (örn. tek renk duvar, uzun koridorlar, yetersiz aydınlatılmış ortamlar) başarısız olmaktadır [3–5]. Tez kapsamında, yetersiz öznitelik barındıran bu tarz ortamlarda kullanılmak üzere RGB resme ek olarak derinlik bilgisinden de faydalanıp öznitelik sayısını arttırmayı hedefleyen yeni bir yöntem (DepthTiling) ileri sürülmektedir. DepthTiling RGB değerlerinin ilgili derinlik bilgisini kullanarak resmi tekrar renklendiren bir sistemdir.

Başka bir ifadeyle, DepthTiling okunan RGB-D verisinin içerdiği derinlik bilgisi yardımıyla oluşturulan harita üzerinde renk döşemesi yaparak; 3 boyutlu olan görsel odometri takibini tahmin ve müdahale edilemeyen alanlarda mevcut sistemlere göre daha başarılı hale getirmeyi amaçlamaktadır.

1.2 Literatür Özeti

Yinelemeli En Yakın Nokta (ICP - Iterative Closest Point), iki nokta bulutu verisi arasındaki farkı minimize etmek için gerekli transformasyonu hesaplayan bir algoritmadır [6]. Geleneksel 2 boyutlu SLAM algoritmaları iki sensör (lazer gibi) okuması arasındaki hareket miktarını hesaplamak için genellikle ICP kullanır. 2 boyutlu SLAM sistemlerinin yanı sıra ICP kullanılarak başarılı sonuçların elde edildiği 3 boyutlu obje modellemeleri de mevcuttur [7]. Ancak, 3B ICP ile büyük ortamların haritalanabilmesi için sensörün çözünürlük ve frekansının düşürülmesi gibi performansı olumsuz etkileyen bazı kısıtlamalar gerekmektedir [8, 9].

Ethzasl-Icp-Mapper [10], RGB-D sensörün derinlik verisini harita üzerine 3 boyutlu ICP [6] ile ekleyerek gezinim sırasında hem kamera takibi hem de haritalama yapmaktadır. Ancak Kinect varsayılan parametrelerle çalıştırıldığında, 300000'den fazla nokta içeren derinlik verisini saniyede 30 kere üretmektedir. Bu da karmaşıklığı çok olan sistemin gerçek zamanlı çalışamamasına, sensör çözünürlük ve veri akışının en aza indirilmesi ihtiyacının oluşmasına neden olmaktadır. Bu durumda dahi, yapılan testlerde istenilen performansa ulaşılamamıştır. Ethzasl-Icp-Mapper sistemi hakkında Bölüm 1.2.3 ile daha detaylı bilgi verilecektir.

Görsel SLAM, bu durumun üstesinden gelmek için, haritalama ve odometri takibinde gözlemlenen tüm veri yerine seçilen öznitelik noktalarını kullanmaktadır. SLAM algoritmasında sadece bu noktaların kullanılması hem işlem hızını artırmakta hem de hafıza gereksinimini azaltmaktadır. RGBD-SLAM metotlarında öncelikle görüntüler uzay düzleminde birbirleriyle benzeştirilmeye çalışılmaktadır. Bu işlemde, t ve $t+1$ anında alınan ardışık çerçevelerin öznitelik noktaları çıkartılmaktadır.

Öznitelik noktaları, köşe ve T-bağlantı alanları gibi resim üzerinde belirgin özelliğe sahip noktalardır. Bu öznitelik noktaları Scale-Invariant Feature Transform (SIFT) [11], Speeded Up Robust Features (SURF) [12], ve Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features (ORB) [13] gibi herhangi bir öznitelik çıkarma algoritması kullanılarak tespit edilebilir.

Öznitelik belirleme yöntemlerinde genellikle başarım ve çalışma süresi ters orantılıdır. Örneğin SURF [14] yöntemi SIFT [15] yönteminden daha hızlı çalışabilirken, SIFT'ın öznitelik belirleme performansı daha iyidir. Bazı çalışmalarda [16], daha doğru sonuç veren, ancak yavaş çalışan (SIFT gibi) yöntemlerin GPU'lar üzerinde çalıştırılarak performans iyileştirilmesi sağlandığı görülmüştür.

Bununla birlikte, Good Features to Track (GFTT) [17] ve Binary Robust Independent Elementary Features (BRIEF) [18] algoritmalarının kombinasyonunun lokalizasyon başarısı ve hesaplama maliyeti açısından en başarılı ikilemi sağladığı düşünüldüğünden [19], çoğu sistem öznitelik çıkarma ve açıklama için sırasıyla GFTT ve BRIEF algoritmalarını tercih etmektedir.

Görsel SLAM yaklaşımını kullanan ilk sistemler küçük çalışma ortamlarında tek kamera kullanılarak geliştirilmiştir [20, 21]. Sonrasında, Kinect gibi 3 boyutlu veri sağlayan ucuz ve hassas RGB-D sensörlerin üretilmesiyle çıkarılan öznitelik noktalarının derinlik bilgilerinin de kullanılabilmesiyle bu alanda yapılan çalışmalar hız kazanmış ve RGB-D veri kullanan bir çok sistem geliştirilmiştir.

RGB-D sensörü kullanan ilk çalışma olan KinectFusion çalışması, Kinect kamerasını çıkartan Microsoft şirketi tarafından yapılmıştır [22]. Bu çalışmada sensör olarak Kinect kullanılıyor olsa da, sadece yoğun yüzey haritalama ve takibi yapılarak görsel öznitelik noktaları kullanılmamıştır. Sistem cisim yüzeylerini temsil etmek için Truncated Signed Distance Function (TSDF) adını verdikleri 3 boyutlu hacimsel yer tutan işaretli bir mesafe fonksiyonu kullanmaktadır [23]. TSDF yapısının içerisindeki her bir hücre, mevcut alan içerisinde kendisine en yakın yüzeye olan mesafeyi işaretli olarak tutmaktadır. KinectFusion çerçeveden çerçeveye eşleme yaklaşımı yerine çerçeveden modele yaklaşımını kullanarak iyi sonuçlar elde etmeyi başarmış olsa da üstsel olarak büyüyen hafıza ihtiyacı ve yüksek performanslı grafik donanımı gereksinimi sebebiyle sadece küçük çalışma alanlarında uygulama yapılması mümkün olmuştur. Bu sistemin bir diğer dezavantajı ise kamera hareketi sırasında birikerek büyüyen kayma hatalarını iyileştirmek için kapalı-döngü (loop-closure) tespiti yapamıyor olmasıdır.

2012'de yayınlanan RGB-D SLAM sistemi [24], görsel öznitelik noktalarını genelleştirilmiş-ICP [25] ile kullanan ilk yaklaşım olarak kabul edilebilir. Sistem

geleneksel görsel SLAM mantığını kullanmaktadır; ilk olarak peş peşe gelen RGB resimler üzerinde öznitelik çıkarımı ve eşlemesi yapılır sonrasında ise ilgili noktaların derinlik bilgisi ICP performansını arttırmak için kullanılır. Haritayı oluştururken, sistem öznitelik noktalarının sadece konumlarını kullanarak kapalı-döngü tespiti yapmaktadır. Sistem aynı zamanda hafıza ihtiyacını indirgemek için surfel adını verdikleri ve konum, yüzey yönü, yüzey boyutu ile renk bilgisini içeren bir yapı kullanarak yüzey temsili yapmaktadır.

Kintinuous, RGB-D kameradan elde edilen görsel ve derinlik verisinin ikisini de kullanan diğer bir sistemdir [4, 26]. Bu sistem KinectFusion'un geliştirilmiş versiyonudur ve onun dezavantajlarını azaltmayı amaçlar. Örneğin, Kintinuous TSDF yapısını tüm haritayı temsil etmek için kullanmak yerine sadece kamera etrafındaki belirli bir alanın temsili için kullanır. Bu özellik üstsel olarak büyüyen hafıza ihtiyacı probleminin çözülmesini sağlar. Sistem kameranın TSDF yapısının içerisinde kalmasını sağlamaya çalışır. Bu nedenle, kamera pozisyonu değiştikçe TSDF alanını da kaydırır. Kintinuous, kamera konum tahmini için iki farklı metot kullanır. Geometrik tahmin olan ilk metot, elde edilen mevcut derinlik bilgisini TSDF içerisindeki yüzeyler ile ICP kullanarak eşleme yapar. Fotometrik tahmin olarak adlandırılan diğer metot ise kamera hareketinin tahminini ardışık RGB-D resimleri kullanarak gerçekleştirir. Kintinuous kapalı-döngü tahminini çıkarılan SURF öznitelik noktalarına bag-of-words yaklaşımını uygularak yapar [27]. Kapalı-döngü tespiti yapıldığında, hem kamera konum grafının hem de haritanın optimizasyonu için iSAM sistemini kullanan bir adım uygulanır [28]. Kintinuous sistemi hakkında Bölüm 1.2.4 ile daha detaylı bilgi verilecektir.

2014'te yayınlanan makalede görsel öznitelik benzerlikleri ve ICP algoritması tarafından tahmin edilen transformasyonu doğrulamak için ortam modeli kullanan bir sistem tanıtılmıştır [5, 29]. Sistem, RGB resimden çok boyutlu öznitelik tanımlayıcı vektörleri çıkarır ve bu vektörleri ilgili derinlik değerleri ve bağıl gözlem konumu ile birlikte saklar. İki gözlem arasındaki transformasyonu hesaplamak için benzer özniteliklere Random Sample Consensus (RANSAC) algoritması uygulanır [30]. Görüntü üzerinden belirlenen özniteliklerin merkez noktalarının, o görüntünün derinlik verisi üzerinde hangi nokta ile temsil edildiği bulunur. Ardışık 2 çerçevede bulunun öznitelik noktalarının en küçük kareler yöntemi ile birbirlerine olan mesafeleri ve açısal farkları bulunur. Böylece kamera pozunun transformasyonu hesaplanmış olur. Bu işlemin doğru gerçekleştirilmesi için RGBD kameranın derinlik bilgisi ile görüntü bilgisinin sekronize edilmiş olması gerekmektedir. Gerek sekronizasyonun yanlış yapılmasından, gerekse derinlik bilgisi için yapılan interpolasyon hatasından kaynaklanan görüntü-derinlik tutarsızlıkları ile karşılaşılabilmektedir. Bu tarz durumların üstesinden gelebilmek

için iki çerçeve arasındaki öznitelik tanımlayıcılarının eşleştirilmesinden sonra, aralarında sert transformasyonlar olan rastgele belirli sayıda eşleşmiş öznitelik ikilileri seçilir. Bu noktalar, birbirlerine göre öklid mesafelerindeki değişim oranlarının eşleşmemesi durumunda aykırı veri olarak belirlenir. Aksi durumda, test başarıyla tamamlandığında, tüm noktaların transformasyon işlemleri gerçekleştirilir. Diğer bir ifadeyle, bir çerçeve kendisinden önceki çerçeveyle eşleştirilirse, bir düğüm oluşturulur ve o çerçevenin bilgileri de hesaplanan transformasyonlar doğrultusunda haritaya eklenir. Az sayıda öznitelik olması ya da özniteliklerin okunan derinlik bilgisinin menzili dışında kalması gürültülü görsel tahmin elde edilmesine sebep olduğundan sistem bir kaç hareket tahminini birleştirerek doğruluğu arttırmaya çalışmaktadır. Sistem aynı zamanda hesaplama yükünü azaltmak için 3 boyutlu olasılıksal doluluk haritası kullanır.

Large-Scale Direct Monocular SLAM (LSD-SLAM) başka bir görsel SLAM sistemidir [31]. Bu sistem, öznitelik tabanlı görsel odometri yerine direkt resim hizalamaya dayanan bir yaklaşım kullanarak diğerlerinden farklılık gösterir. LSD-SLAM gerçek zamanlılık şartını sağlayabilmek için 3 boyutlu ortamı yarı-yoğun derinlik filtresi metodu ile oluşturmaktadır.

ICP yerine Bundle Adjustment (BA) [32, 33] yöntemini kullanan ORB-SLAM2 [34, 35] gibi sistemler de vardır. BA, 3 boyutlu sahne geometrisi, kameranın bağıl hızı ve kamera parametrelerini tanımlayabilmek için görsel modellemeyi düzeltme problemi şeklinde ifade edilebilir. ORB-SLAM2, her biri odometri takibi, lokal haritalama ve kapalı-döngü tespiti görevlerinden birini üstlenmiş olan üç thread içerir. Lokal haritalama yapan thread, harita optimizasyonu için lokal BA kullanırken kapalı-döngü tespiti yapan thread konum graf optimizasyonu sonrası tam BA uygulamak için dördüncü bir thread başlatır.

Real-Time Appearance-Based Mapping, (RTAB-Map) literatürdeki en başarılı sistem olarak gösterilebilir [3, 36, 37]. Sistem gerçek zamanda çalışma koşulunu karşılamak için temel olarak hafıza yönetimini sağlarken kapalı-döngü tespiti ve graf optimizasyonu algoritmalarını birleştirmektedir. RTAB-Map aynı zamanda farklı haritaların birleştirilebilmesi için çok oturumlu haritalamaya da izin vermektedir. Bu sistem harita konumlarını üretmek için RGB resimlerden çıkarılan öznitelik noktalarıyla oluşturulan görsel kelimeleri kullanmaktadır [38]. Bu görsel kelimeler, odometri takibi ve kapalı-döngü tespiti için önceden gezilmiş konumlarla karşılaştırılmak üzere bag-of-words yaklaşımıyla görsel bir sözlüğe eklenir. Kapalı-döngü tespiti haritanın içerisinde olabileceği gibi sistemin çok oturumlu olma özelliği sayesinde farklı haritalar arasında da tespit edilebilir. Sistem daha hızlı karşılaştırma yapabilmek için k-boyutlu ağaç (kd-tree - k-dimensional tree) [39]

kullanır. Başarılı bir kapalı-döngü tespiti yapıldıktan sonra özniteliklerden oluşturulan görsel kelimelere RANSAC yaklaşımı kullanılarak resim eşlemesi yapmak için 3 boyutlu transformasyon uygulanır. RTAB-Map konumları düğüm olarak saklayan graf tabanlı bir SLAM sistemidir ve odometri hatalarını indirgemek için harita optimizasyonunu Tree-based netwORk Optimizer (TORO) [40] yaklaşımını kullanarak yapar. RTAB-Map sistemi hakkında Bölüm 1.2.1 ile daha detaylı bilgi verilecektir.

Bazı SLAM yöntemlerinde sensör pozları arasında hesaplanan transformasyon ikilileri poz grafının kenarları olarak biçimlendirilir. Bu pozlar g^2o [41], TORO [40] vb. optimizasyon algoritmaları ile işlenerek global perspektifte daha doğru haritaların çıkarılması sağlanmaktadır. Özellikle kapalı-döngü tespiti problemlerinde düğümler tekrar dolaşılarak optimize edilmeye çalışıldığından verim arttırılmaktadır. Bu yöntemler gerçek zamanlı çalışmak için tasarlandığından, RGB-D kameralardan alınan görüntülerin her bir çerçevesi işlenemeyebilmektedir. Ardışık iki çerçevenin transformasyonunun hesaplanması bitmeden gelen çerçevelerin boşa çıkarıldığı yaklaşımlar mevcutken, bu çerçevelerin bir tamponda biriktirildiği ve her birisinin işlendiği yaklaşımlar da mevcuttur. Görüntülerin derinlik bilgileri bellekte oldukça yer tuttuğundan, RGBD-SLAM yöntemlerinde bir diğer ele alınması gereken konu da çıkarılan haritanın nasıl temsil edileceğidir. Bu bağlamda geliştirilen 3B harita temsil sistemleri de mevcuttur [42, 43].

Yukarıdaki sistemlerin hepsinin ortak özelliği başarılarının öznitelik noktalarının ne kadar iyi çıkarıldığına bağlı olmasıdır. Geliştirilen sistem ne kadar iyi olursa olsun özniteliklerin düzgün çıkarılamaması durumunda istenen başarı elde edilememektedir. Bu nedenle bu sistemler her ne kadar çoğunlukla odometri takibi ve haritalamayı başarılı bir şekilde yapabiliyor olsalar da, ilgili yayınlarda RGB-D kameranın derinlik menzili içerisinde yetersiz öznitelik noktası olması durumunda karşılaştıkları problemlerden bahsetmişlerdir [3, 5].

Haritanın oluşturulmasının yanı sıra keşif ve gezinim sistemleri tarafından da kullanılabilmeleri adına temsil edilme şekilleri de büyük önem arzetmektedir. Octo-Map [42], harita temsili için geliştirilmiş olan ve hafızada gereksinimin çok olduğu 3 boyutlu haritalamalarda avantaj oluşturan bir yaklaşımdır. Robotlara keşif ve navigasyon gibi otonom özelliklerin karşılaştırılmasında kullanışsız olan nokta bulutu verisine alternatif olarak ızgara tabanlı 3 boyutlu haritanın ağaç olarak hafızada temsil edilmesi sağlanmaktadır. Bu sistem odometri ve sensör verisinin doğru olduğu varsayımıyla direkt olarak gelen veriyi haritaya eklemektedir. Haritada hiçbir düzeltme yapılmadığından öncelikle düşük hata oranına sahip 3 boyutlu odometri takibinin sağlanıp farklı yöntemlerle düzgün bir haritanın nokta bulutu olarak elde edilmesi, sonrasında ise Octo-Map'e çevrilmesi haritanın istenen düzeyde başarılı

olmasında avantaj oluşturmaktadır. Octo-Map sistemi hakkında Bölüm 1.2.2 ile daha detaylı bilgi verilecektir.

Tez kapsamında, görsel SLAM sistemlerinin başarısının RGB verisine olan bağımlılığını azaltmak için haritalamada kullanılacak noktaların belirlenmesinde alternatif bir yöntem ileri sürülmektedir. DepthTiling, görsel SLAM algoritmaları için böyle ortamlarda daha fazla öznitelik noktası sağlamayı amaçlamaktadır.

Bölüm 1.2.1 - 1.2.4 ile tez sürecinde detaylı olarak incelenen, test edilen ve yapılan çalışmalar sonrası performans analizinde kullanılan bazı sistemlerin detaylı açıklaması yapılmaktadır.

1.2.1 RTAB-Map

Sistem genel olarak; haritalamada oluşan kaymaları düzeltmek için kullanılan kapalı-döngü tespit oranını düşürmeden, haritalanan ortamın büyümesiyle oluşan ve kaçınılmaz olan hafıza ve zaman problemlerine bir çözüm getirmektedir. Bunun için temelde 2 farklı hafıza tutmaktadır. Birincisi kapalı-döngü tespit ihtimali yüksek olan yerlerin bulunduğu ve yeni algıladığı veri üzerinde karşılaştırma yaptığı Çalışan Hafıza (WM - Working Memory), ikincisi ise haritanın geri kalan kısmının saklandığı ve veri tabanında bulunan Uzun-Vadeli Hafızadır (LTM - Long-Term Memory).

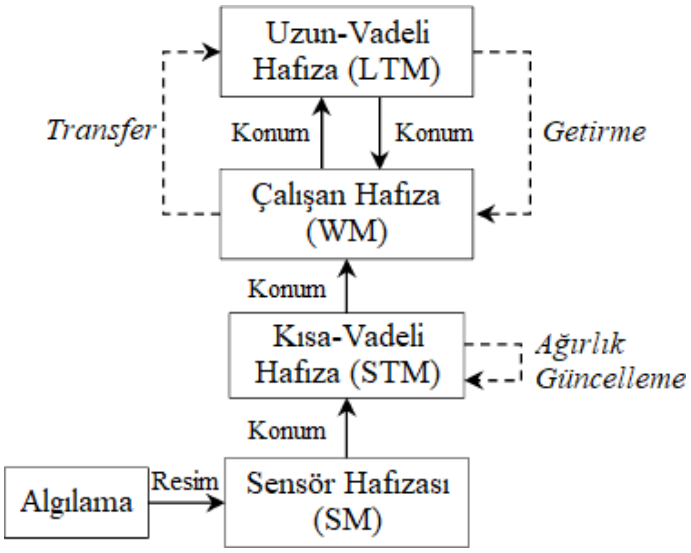
Gerçek dünyada, bir robotun bilinmeyen bir ortamda otonom hareket edebilmesini sağlamak için hem haritalamayı hem de lokalizasyonu aynı anda (SLAM) iyi bir şekilde yapabilmesi gerekmektedir. Kullanılan SLAM algoritmasının başarılı olabilmesi için anahtar özelliklerden biri olan daha önce gezilmiş olan yerlerin tespit edilebilmesi işlemine kapalı-döngü tespiti denmektedir. Kapalı-döngü tespiti için genelde iki yaklaşım kullanılmaktadır. İlk yaklaşımda tespit için robot pozisyonundaki belirsizliğe bağlı olarak lokal alana bakılmaktadır. Burada sıkıntı lokalizasyondaki hataların kapalı-döngü tespit oranını etkilemesidir. Diğer bir yaklaşım ise tahmini konumdan bağımsız olarak tüm haritaya bakılmasıdır. Burada da problem haritanın büyümesiyle sistemin gerçek-zamanlı olma özelliğini kaybetmesidir.

RTAB-Map, geniş-alanda uzun-zamanlı işlemler için gerçek zamanda çalışacak ve kabul edilebilir seviyede kapalı-döngü tespiti yapacak bir hafıza yönetim yaklaşımı ileri sürmektedir. Sistem kapalı-döngü tespiti ihtimalinin en yüksek olduğunu varsaydığı konum bilgilerini WM'de saklarken diğerlerini LTM'ye aktarmaktadır. Yakın zamanlarda görülmüş konumlar ile daha önce en çok gözlemlenen konumların kapalı-döngü tespit oranını yüksek tutmak için WM'de saklanma ihtimali daha yüksektir. Gerçek zamanlı çalışma durumunu sağlamak için ise WM'de bulunan

konum sayısı çeşitli parametrelere bağlı olarak önceden belirlenmiş bir sayının altında tutulmaktadır.

Sistem görüntü-tabanlı olduğu için kapalı-döngü tespitinde bag-of-words yaklaşımı kullanılmaktadır. Bag-of-words yaklaşımı aslında ilk olarak, adından da anlaşılacağı üzere, doğal dil işleme alanındaki çalışmalarda geliştirilmiştir [44]. Bu yaklaşımda, üzerinde çalışılan dile ait, her kelimenin yalnızca bir kere geçtiği bir sözlük tutulmaktadır. Verilen her doküman bu sözlük içerisindeki tüm kelimeleri teker teker kaç defa içerdiğini tutan bir array ile temsil edilmektedir. İki doküman arasındaki uzaklık yakınlık ilişkisi ise bunları temsil eden arraylar kullanılarak hesaplanmaktadır. Bilgisayarla görme alanında ise ilk defa 2003'te kullanılmaya başlanmıştır [38]. Bu yaklaşımda, her bir resim sözlükte bulunan görsel kelimeler tarafından temsil edilir. Bu görsel kelimeler genellikle SIFT gibi yöntemler kullanılarak çıkarılan özelliklerden oluşur. Bu özellikler çok boyutlu olduğu için sözlük oluşturup direkt bir karşılaştırma yapmak büyük maliyete sebep olacağından k-means ve kd-tree gibi yöntemler tercih edilmektedir. Sözlük oluşturma işlemi daha önce kaydedilmiş bir eğitim kümesi kullanılarak yapılabileceği gibi gerçek zamanlı olarak da yapılabilmektedir. Bu sistemde ise gerçek zamanlı çalışmayı sağlamak için tüm sözlük değil, yerine sadece seçilen anahtar kelimeler kullanılmaktadır.

Sistemin amacı zaman ve mekan sınırı olmaksızın gerçek zamanlı kapalı-döngü tespiti yapmak olduğundan tespit için kullanılan konum sayısı sınırlı tutulurken gerekli durumlarda tüm haritaya da ulaşabilmektedir. LTM'deki konumlar kapalı-döngü için kontrol edilmediğinden WM'den LTM'ye konum aktarımında temel mantık daha sık görülen yerlerin tekrar görülme ihtimali daha yüksek olduğu varsayımı olduğundan o bölgelere büyük ağırlıklar verilmektir. Aktarım gerektiğinde, en küçük ağırlığa sahip konumlardan en eskisi ilk seçilmektedir. Şekil 1.1, RTAB-Map'in büyük ortamları haritalarken dahi gerçek zamanlılık şartını sağlama için kullandığı hafıza yönetim modelini göstermektedir. Algılama (Perception) modülünün gönderdiği resmin boyutunu küçültmek için Sensör Hafızası (SM - Sensory Memory) o resme ait "imza"yı hesaplayıp buradan yeni bir *konum oluşturarak* Kısa-Vadeli Hafızaya (STM - Short-Term Memory) göndermektedir. SM'den yeni gelen konumun STM'de mevcut olanlarla benzerliği karşılaştırılır. Benzerse, birleştirme yapılarak ilgili *ağırlık güncellenir*. Burada peşpeşe alınan karelerin birbirine benzer olacağı varsayımından faydalanarak gereksiz yere kapalı-döngü tespit işlemine sokulmaması için indirgeme yapılmış olmaktadır. T_{STM}, STM içerisinde kaç konuma birleştirme yapılacağını belirlemektedir. Bu değişken, robotun hızı ve yeni konum ölçümleyebilme oranına göre belirlenmektedir. T_{STM} tane konum oluştuğu zaman, STM'ye ilk eklenen konum kapalı-döngü tespiti için WM'ye aktarılmaktadır.



Şekil 1.1 RTAB-Map hafıza yönetim modeli

WM'ye yeni gelen konum için WM'de kapalı-döngü değerlendirmesi ayrık *Bayes filtresi* kullanılarak yapılmaktadır. Yeni konum ile WM'de bulunan eskiler arasında *kapalı-döngü tespit* edildiğinde, bu iki konum arasında yüksek oranda kapalı-döngü olasılığına sahip bir link bağlanmaktadır. Sonrasında ise 2 adımlık *Getirme (Retrieval)* ve *Transfer* işlemleri sırasıyla gerçekleştirilmektedir. Bu işlemlerin amacı WM boyutunu belirli bir limit altında, gelecekte karşılaşılabilecek yeni kapalı-döngü tespitini de maksimum düzeyde tutmaktır. Getirme işlemiyle, WM'de en yüksek kapalı-döngü olasılığına sahip konumların komşuları ileride kapalı-döngü tespit olasılığını artırmak için LTM'den WM'ye aktarılmaktadır. Transfer işleminde ise kapalı-döngü tespitinin önceden belirlenen T_{time}'dan uzun sürmesi durumunda gerçek zamanlılık şartını sağlamak için WM'den en az gözlemlenen konumların en eskisi başka bir ifadeyle en eski ve/veya en düşük ağırlığa sahip konumlar LTM'ye aktarılmaktadır.

Aşağıda RTAB-Map'in genel sistem adımları ve her bir adımda neler yapıldığı liste halinde açıklanmaktadır.

- Konum Belirleme: Gerçek zamanlı olarak üretilen sözlük içerisindeki görsel kelimeler ile temsil edilen resim imzaları oluşturulurken bag-of-words yaklaşımı kullanılmaktadır. OpenCV [45] kullanılarak her bir resim üzerinde SURF hesabı yapılmaktadır. Sözlükteki her bir görsel kelime, bir öznitelik tanımlayıcısına denk gelmektedir. Öznitelikler kendi gücünü temsil eden tepki değerlerine sahiptir. Kötü temsilden kaçınmak ve her bir resim için kullanılan öznitelik sayısını sınırlı tutmak amacıyla tepki değeri $T_{response}$'tan büyük olan maksimum $T_{maxFeatures}$ sayısı kadar öznitelik, resim imzası için seçilmektedir. Eğer bir

resmin imzası için bulunan öznitelik sayısı her bir resme düşen ortalama öznitelik sayısının T_{bad} oranından küçükse (beyaz duvar gibi), bu imza kötü kabul edildiğinden kapalı-döngü tespitinde kullanılmamaktadır.

İyi imzaları oluşturan özniteliklerden zaten sözlükte olanların seçiminde ilgili özniteliğe en yakın ve ikinci en yakın komşular karşılaştırılmaktadır. Burada verimliliği arttırmak amacıyla en yakın ilk iki komşuyu aramadan önce sözlük üzerinde 4 kd-tree içeren rastgele orman oluşturularak kullanılmaktadır. Eğer en yakın komşusuyla arasındaki mesafe ile ikinci en yakın komşusu ile arasındaki mesafenin oranı (NNDR - Nearest-Neighbor Distance Ratio) önceden belirlenmiş T_{NNDR} değerinden küçükse aynı kelime olarak kabul edilerek sözlükteki ilgili kelime imzaya eklenmektedir. Eğer bu kriteri geçemezse, öznitelik tanımlayıcısından yeni bir kelime oluşturularak bu kelime hem sözlüğe hem de imzaya eklenmektedir. Sözlüğe yeni kelimeler eklendiğinde bir sonraki döngünün başında ağaçların tekrar oluşturulması gerekmektedir.

Sonrasında ise resmin imzası kullanılarak oluşturulan konumun ağırlığına 0 verilip bir önceki konumla arasına iki yönlü bir bağlantı eklenmektedir.

- Ağırlık Güncelleme: Edinilen konum bilgisi, ağırlığının güncellenmesi için STM'deki son konumla karşılaştırılmaktadır. İki konum arasındaki benzerlik ilişkisi, imzalarındaki ortak kelime sayısı iki imzadaki toplam kelime sayılarından büyük olanına bölünerek bulunmaktadır. Hesaplanan bu benzerlik önceden belirlenen $T_{similarity}$'den büyükse sadece eski imzadaki kelimeleri kullanmak için yeni imzaya kopyalama yapılarak yeni imza için sözlüğe eklenmiş kelimeler sözlükten geri silinmektedir. Burada sadece eski imza bilgilerinin kullanılma amacı yeni eklenen kelimelerin henüz kd-tree'de kullanılmamış olmasıdır. Yeni konumun ağırlığı eski konumun ağırlığına bir eklenerek hesaplanıp sonrasında ise eski konum üzerindeki komşu ve kapalı-döngü bağlantıları yeni konuma aktarılarak eski konum STM'den silinmektedir.

- Bayes Filtresi Güncelleme: Bayes filtresinin amacı mevcut konumun önceden ziyaret edilmiş konumlarla eşleşme olasılığını tutarak, kapalı-döngü takipi yapmaktır. Burada dikkat edilmesi gereken nokta karşılaştırmanın yapıldığı eski konumlar klasik yöntemlerde sabit iken burada WM ve STM tarafından tutulan konumlar kullanıldığından yeni konum gözlendiğinde ya da LTM'ye konum göderip LTM'den konum alındığında içerik bakımından değişiklik gösteren bir liste kullanılmaktadır. Gözlem modelinde, ağırlık güncellemedeki benzerlik yaklaşımı kullanılarak mevcut konumun listedeki tüm konumlarla kapalı-döngü olma ve yeni konum olma ihtimalleri hesaplanmaktadır. Geçiş modelinde ise robotun bir önceki konum olasılık dağılımları ve yaptığı hareket kullanılarak, konumların yeni olasılık dağılım hesabı yapılmaktadır. Burada "bir önceki

adımda kapalı-döngü olmamışken şimdiki adımda yeni konumda olma ihtimali", "bir önceki adımda kapalı-döngü olmamışken şimdiki adımda kapalı-döngü olma ihtimali", "bir önceki adımda kapalı-döngü varken şimdiki adımda yeni konumda olma ihtimali" ve "bir önceki adımda bir komşuyla kapalı-döngü varken şimdiki adımda kapalı-döngü olma ihtimali" olmak üzere geçiş modeli toplamda dört farklı olasılık değeri ile ifade edilmektedir.

- Kapalı-Döngü Hipotez Seçimi: Mevcut konum için hesaplanan olasılık dağılımının normalize edilmesinin ardından, eğer mevcut konumun yeni konum olma olasılığı başka bir ifadeyle kapalı-döngü tespit edilmemiş olma olasılığı T_{loop} değerinden küçükse, önceden gözlemlenmiş konumlar üzerinde en büyük kapalı-döngü olasılığına sahip konum seçilmektedir. Mevcut konumun ağırlığı bir artırılırken kapalı-döngü olarak seçilmiş eski konumun ağırlığı sıfırlanarak bu iki konum arasına kapalı-döngü linki bağlanmaktadır. Konumların birleştirilmemesinin sebebi dinamik ortamlarda ya da gece-gündüz gibi değişimler gösteren ortamlarda farklı konum temsillerini sağlamaktır.

- Getirme: Kapalı-döngü tespiti sonrası WM'de olmayan komşular LTM'den transfer edilmektedir. Bu çalışmada SQLite3 veritabanı kullanılarak oluşturulan LTM'de veri işlem bittikten sonra da saklanır. Link tablosunda, komşu ya da kapalı-döngü olmak üzere konumlar arasında iki farklı link tipi tutulmaktadır. Komşu linki peşpeşe gelen konumlar arasına eklenirken diğeri kapalı-döngü tespiti olduğunda ilgili konumlar arasına eklenir. Kapalı-döngü linki eklendikten sonra, odometri hatalarını azaltmak için TORO [40] yaklaşımı kullanılarak harita optimize edilir.

  LTM'den konumlar aktarıldığında bu konumların imzalarıyla ilişkili görsel kelimeler de aktarılmaktadır. İmzalar içerisindeki kelimelerden hala sözlükte olanlar için ilgili imza ile sözlükteki kelime arasına bir referans eklenmektedir. Diğer kelimeler için *Konum Belirleme*'de kullanılan yöntemle daha yeni bir temsil olup olmadığına bakılmaktadır. Çünkü yeni eklenen mevcut konum ile gelen öznitelik tanımlayıcıları LTM'den çekilenlerle eşleşebilmektedir. Bu durumda, LTM'den alınan konumların ilgili kelimeleri değiştirilmektedir. Ancak bu durum veri tabanına işlem maliyeti dolayısıyla yansıtılmamaktadır. Eşleşmeyen öznitelik tanımlayıcıları kalmışsa da sözlüğe ekleme yapılmaktadır.

  LTM'de bulunan konumlar kapalı-döngü tespiti ve graf optimizasyonunda işleme alınmıyor olsalar da bu konumların LTM'den WM'ye geri aktarılma ihtimalleri mevcuttur. Eğer yeni gelen konum ile WM'de olan konumlardan biri arasında kapalı-döngü tespit edilirse, yeni kapalı-döngü tespitlerinin ihtimalini arttırmak için WM'deki konumun komşuları LTM'den WM'ye aktarılmaktadır. Veri tabanından konum çekme işlemi zaman alan bir süreç olduğundan her bir

döngüde en fazla iki konum WM'ye aktarılmaktadır. Veri tabanında, aktarıma uygun ikiden fazla konum varsa, kapalı-döngü linki ile bağlı olanlardan ziyade zamansal komşuluk linki ile bağlı konumlara öncelik verilmektedir.

- Transfer: Kapalı-döngü tepiti ve graf optimizasyonu sadece WM'de bulunan yerel haritaya uygulanır. Eğer gelen bir resim için bu adımların işlem süresi önceden belirlenen T_{time} süresini geçtiğinde gerçek zamanlılık şartı tekrar sağlanıncaya kadar WM'den LTM'ye resim aktarımı yapılmaktadır. Bayes filtresi kullanan kapalı-döngü hipotezinin düzgün çalışabilmesi için en yüksek hipoteze sahip konumun komşularının transfer edilmesi engellenmektedir. T_{time} değişkenin daha büyük belirlenmesi WM'de daha çok konumun tutulabilmesini sağladığından kapalı-döngü tespit şansını artırsa da bu değişken robotun gerçek zamanlı olarak kaç resim işleyebildiğine göre deneysel olarak belirlenmelidir.

  RTAB-Map'in en maliyetli adımı kd-tree'nin gerçek zamanlı olarak üretilmesi olduğundan sözlük boyutunu küçültmek işlem hızını etkilemektedir. WM'den LTM'ye transfer edilen her bir resmin sözlükteki kelimelere olan referansları kaldırılmaktadır. WM'de kendisini referans eden bir resim bulunmayan kelimeler de LTM'ye transfer edilmektedir. Sözlükten LTM'ye transfer edilen kelime sayısı yeni eklenen konumdan ya da *Getirme* işlemi sırasında eklenen konumlardan gelen kelime sayısından az olduğu sürece WM'den LTM'ye konum transfer işlemi devam etmektedir. Böylece sözlüğün büyümesi engellenmiş olmaktadır.

  Gezinim sırasında belirli bir bölgenin diğer alanlardan çok daha fazla görülmesi o bölgenin ağırlığının çok fazla artmasına neden olmaktadır. Bu durumda yeni bir alana ilk kez girildiğinde, WM'den LTM'ye transfer için seçilen konumların bu yeni görülmüş alandan olma riskini artırdığı için bulunduğu konumda kapalı-döngü tespit oranını düşürmektedir. Bu riski azaltmak için en son kapalı-döngü tespitinden sonra eklenen konumlardan yüksek ağırlıklı olanlarından belirli sayıdaki konumun LTM'ye aktarılmasına izin verilmemektedir.

Eğer yeni bir konumla daha önceden üretilmiş bir harita arasında kapalı-döngü tespiti yapılırsa RTAB-Map global bir harita oluşturmak için iki harita arasında gerekli transformasyonları yaparak birleştirir. Çok oturumlu haritalamanın mümkün olması benzer bir mantıkla robot kaçırma problemini çözmektedir. Robot kaçırıldıktan sonra, oluşturulmuş olan harita LTM'ye kaydedilir ve yeni bir harita başlatılır. Kapalı-döngü tespit edilmesini sağlayacak ortak bir konuma ulaşıldığında ise iki harita birleştirilir.

1.2.2 Octo-Map

Octo-Map [42, 43] Octree isminde ağaç yapısını içeren bir harita temsil yöntemi kullanmaktadır. 3 boyutlu haritalamada göz önüne alınması gereken 3 gereksinim aşağıda detayları listelenen "olasılık temsili", "haritalanmamış alanların modellenmesi" ve "verimlilik" konularıdır. Octomap özellikle son iki gereksinimi karşılamayı hedeflemektedir.

- Olasılık Temsili: Sensörlerden alınan verilerdeki küçük hataların kümülatif olarak artmasının yanı sıra yansıma ve dinamik objeler gibi nedenlerden dolayı oluşan hatalar elde edilen verinin güvenilirliğini olumsuz yönde etkilemektedir. Böyle gürültülü veri kullanılarak doğru bir haritanın çıkarılması için ise mevcut belirsizliklerin olasılıksal olarak hesaba katılmasıyla mümkün olmaktadır.

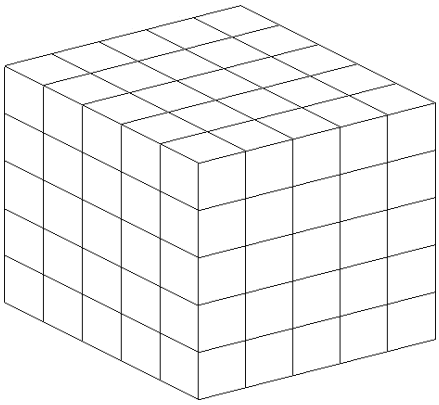
- Haritalanmamış Alanların Modellenmesi: Otonom gezinimlerde, robot ancak daha önce sensörler tarafından ölçülüp boş olduğu belirlenen alanlarda engellerden kaçınan bir yol hesaplaması yapabilmektedir. Bu nedenle oluşturulan haritanın gözlemlenmemiş alanlarının, robotun bu alanlardan kaçınabilmesini sağlamak amacıyla, temsil etmesi önem taşımaktadır. Bu alanların temsili keşif algoritmalarının çalıştırılmasında da önemlidir.

- Verimlilik: Harita, otonom sistemlerin çalışması sırasında robotun karar mekanizmasını aktif olarak etkilediği için merkezi bir öneme sahiptir. Bu nedenle haritanın hesap maliyeti ve hafıza gereksinimi olarak makul seviyede olması gerekmektedir. Pratik açıdan bakıldığında, hafıza gereksiniminin genel olarak dar boğaz oluşturduğu için büyük alanların da haritalanmasını sağlamak amacıyla kullanılan modelin hafızada mümkün olduğunca az yer kaplaması gerekmektedir.

Yukarıda açıklanan üç temel gereksinimin de sağlanabilmesi için harita temsilinin seçimi önemli rol oynamaktadır. Aşağıda, literatürde haritalama için kullanılagelmiş belli başlı temsil yöntemleri açıklanmaktadır.

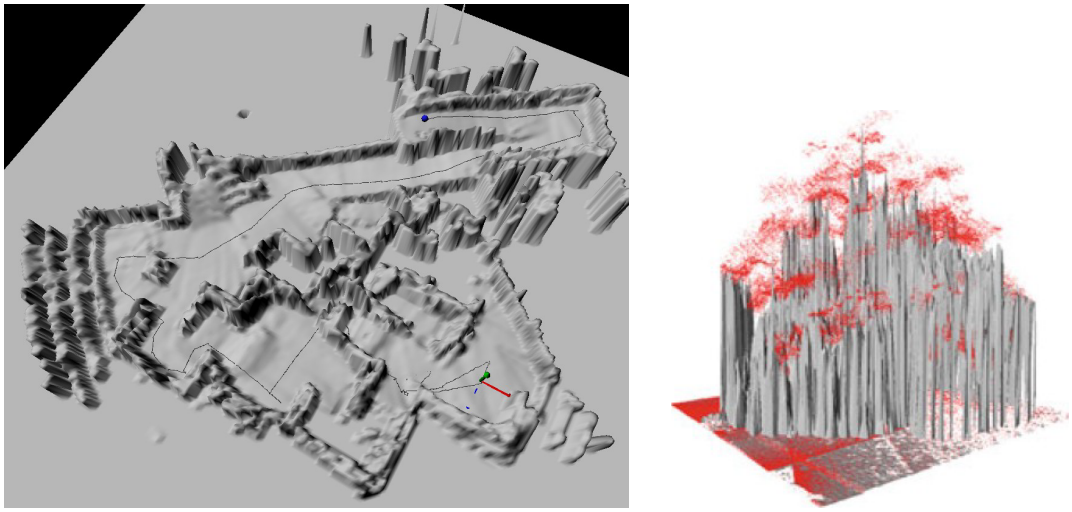
- 3 boyutlu Izgara Düzeni: Bu yapıda, Şekil 1.2 ile gösterildiği gibi 3 boyutlu alan birbirine eşit küplere bölünmektedir. Bu küplerin temsil ettiği alan ne kadar küçükse o kadar ayrıntılı bir harita elde edilmektedir. Bu yöntemin en temel problemi hafıza gereksinimidir. Özellikle büyük alanların iyi çözünürlüklerle temsil edilebilmesi için oluşan hafıza gereksinimi baş edilemez şekilde artabilmektedir. Ayrıca bu yapıda haritalanacak alanın boyutunun önceden bilinmesi zorunluluğu mevcuttur. Aksi takdirde temsil

alanının genişlemesi gerektiği durumlarda maliyeti yüksek kopyalama işlemleri kaçınılmaz olmaktadır.



Şekil 1.2 3 boyutlu Izgara Düzeni ile harita temsili

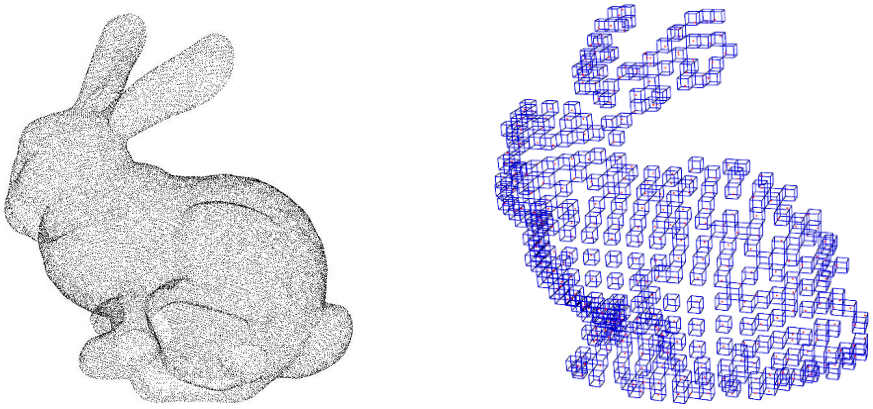
- Yükseklik Haritası (Elevation Map): Bu modele 2.5 boyutlu harita da denmektedir. Temelde ızgara yaklaşımına sahip olan bu modelde 2 boyutlu haritanın her bir hücresi kendi yüksekliğinin sayısal değerini tutmaktadır. Sadece belirli olarak yüksekliğe sahip nesnelerin bulunduğu ortamlarda gezinim gibi sistemler için kullanışlı haritalar üretilebilmektedir. Şekil 1.3a ile başarılı bir örnek harita verilmektedir [46]. Bu modelin bir dezavantajı Şekil 1.3b ile verilen ağaç örneğinde olduğu gibi belirli bir yükseklikten sonra başlayan engellerin bulunduğu ortamların temsilinin mümkün olmaması dolayısıyla 3 boyutlu dünyanın tam olarak modellenememesidir. Bir diğeri de önceki yaklaşımda da olduğu gibi ortam büyüklüğünün önceden bilinme gereksiniminin olmasıdır.



(a) **(b)**

Şekil 1.3 Yükseklik harita (a) başarılı [46] ve (b) başarısız [42] temsili örneği

- Nokta Bulutu: Şekil 1.4a ile örneği verilen nokta bulutu dolu olduğu belirlenen alanlara noktasal parçacıklar eklenerek oluşturulan yapıdır [47]. Lazer ve RGB-D kameralar gibi sensörlerden alınan 3 boyutlu verinin direkt olarak saklanmasıyla oluşan bir model olduğundan kullanımı nispeten kolaydır. Diğer durumlarda olduğu gibi alınan verinin haritaya uyarlanması gibi bir işlem adımı gerektirmemektedir. Bu yapının dezavantajı öncelikle hafıza sıkıntısıdır. Özellikle birden fazla gözlemlenen cisimlerin temsili için üst limiti olmayan sayıda parçacık oluşturulmasına yol açabilmekte ve bu nedenle gereksiz hafıza işgali ya da bu durumu indirgemek için ekstra kontrol gereksinimi oluşmaktadır. Ayrıca bu modelde boş ve gözlemlenmemiş alanların ayrılmamış olması ve tutulan noktacıklarla ilgili olasılık değerlerinin olmaması bu temsilin kullanımının getirdiği diğer dezavantajlardır.

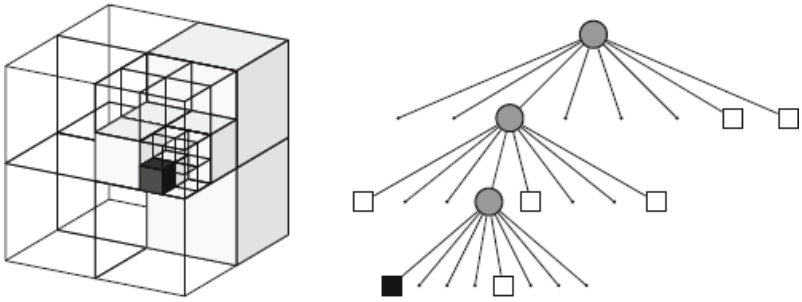


(a) **(b)**

Şekil 1.4 Aynı model için (a) nokta bulutu temsili [47] (b) octree temsili [42]

- Octree: Octo-Map tarafından kullanılan ve Şekil 1.4 ile örneği verilen bu modelde haritalanan alanın temsil edildiği 3 boyutlu küpler hafızada avantaj sağlaması adına ağaç olarak temsil edilmektedir. Şekil 1.5 ile de gösterildiği gibi ağaç üzerinde bir düğümü oluşturan her bir küp daha ayrıntılı temsil sağlayan sekiz alt küpe bölünmektedir. Daha önceden belirlenen minimum küp boyutuna ulaşılana kadar bu işlem devam etmektedir. Ancak her bir düğümü oluşturma ve altdüğümlere bölme işlemi, sensör verisinin okunmasından sonra yapıldığı için gereksiz hafıza işgalinden de kaçınılmış olmaktadır. Bu modelin nokta bulutuna göre bir avantajı dolu alanlarla birlikte boş ve gözlemlenmemiş alanların da haritada temsil edilebilmesidir. Şekil 1.5 ile gösterilen ağaçta da olduğu gibi düğüm oluşturulmamış kısımlar henüz gözlemlenmemiş alanları temsil etmektedir. Küplerin dolu olduğu bilgisi sensör ölçümlerinden gelen nokta bulutu verisi kullanılarak işlenirken boş bilgisi sensörün konumu ve nokta bulutunun herhangi bir noktası arasında kalan alanı kapsayacak şekilde

verilmektedir. Eğer bir düğümün tüm altdüğümleri aynı duruma sahipse, başka bir ifadeyle hepsi dolu ya da hepsi boş bilgisi içeriyorsa, budama işlemi yapılarak kullanılan hafızanın indirgenmesi sağlanabilmektedir.



Şekil 1.5 Octree'nin boş (gölgeli) ve dolu (siyah) alanları gösterimine örnek. Solda kübik model, sağda ise ilgili ağaç temsili gösterilmektedir [42]

Octree üzerinde gözlemlenmiş alanı temsil eden her düğüm kendi durumuna (dolu/boş) ait bir olasılık değeri tutmaktadır. Bu değer her bir sensör ölçümü sonrası kullanılan sensör modeline göre güncellenmektedir. Gezinim sırasında ise bir alanın dolu olup olmadığı bir eşik değeri ile hesaplanmaktadır. Bu eşik değerinin üzerinde olasılığa sahip alanlar dolu, diğer alanlar ise boş kabul edilmektedir. Ayrıca kullanılan güncelleme yöntemleri dolayısıyla bir düğümün durumunu değiştirmek için o düğümün oluşumunda mevcut durumun algılanma sayısı kadar yeni durumun da algılanması gerekmektedir. Bu özellik, statik ortamlarda haritada oluşan geçici değişikliklerin modeli etkilemesini engellemektedir. Kalıcı değişikliklerin haritaya entegre edilememesi durumundan kaçınmak için ise yeni durum için gerekli ölçüm tekrar sayısında bir üst sınır belirlenmiştir.

Her bir ölçüm sonrası, sadece yaprak düğümler için olasılık güncellemesi yapılmaktadır. Bu nedenle, çözünürlüğün azaltılmak istendiği herhangi bir durumda, içerideki bir düğümün olasılığı kendisine ait yaprak düğümlerinin olasılıkları kullanılarak hesaplanmaktadır. Alt düğüm bilgilerini birleştirmek için sistem tarafından bir kaç farklı yaklaşım kullanılabildiği halde gezinimin engellerden sakınan bir yol hesaplayabilmesini sağlamak adına alt düğümlerinden herhangi birinin dolu gözüktüğü tüm iç düğümler dolu kabul edilmektedir. Ancak bilgi kaybı yaşandığından bu tercih edilen bir öntem değildir. Bu nedenle, budama işlemi sadece tüm alt düğümlerinin durumları (dolu/boş) aynı olduğunda gerçekleştirilir. Burada kaybedilen tek bilgi alt düğümlerin her biri için durum olasılık yüzdeleri olduğundan makul bir bilgi kaybı sayılabilmektedir.

Haritalamada kullanılan ağaçların hafıza açısından optimizasyonunu sağlamak amaçlı düğümler tasarlanırken, büyük bir haritada bu ağaçların sayısında oluşacak artış göz

önüne alınmaktadır. Örneğin her bir düğüm için tutulan olasılık değerinin yanı sıra o düğümün olası 8 çocuğu için 8 farklı işaretçi kullanımı gerekmektedir. Ancak haritalama sırasında kullanılan ağaçlarda yaprak düğüm sayısındaki yüksek oran ve bunların çocuklarının bulunmamasıyla birlikte iç düğümlere de sadece sensör ölçümü yapıldığında çocuk eklemesinin yapılması, her bir düğüme eklenen 8 işaretçinin çoğunun kullanılmamasına ve hafızada gereksiz yer kaplamasına sebep olmaktadır. Bu durumun üstesinden gelmek için her bir düğüme yalnız bir işaretçi eklenmekte ve sadece o düğüme ait çocuk ekleme gereksinimi olduğunda bu işaretçi 8 işaretçi içeren bir diziyi göstermektedir. Böylece gereksiz hafıza kullanımından kaçınılmaktadır.

1.2.3 Ethzasl-Icp-Mapper

Daha önce de bahsedildiği gibi, ICP iki nokta bulutu arasındaki farkı minimize etmek için gerekli olan dönüşümü hesaplayan bir algoritmadır [6] ve 2 boyutlu haritalama algoritmalarında lazer verisi ile eldeki haritanın düzeltilmesi ve robotun odometri bilgisinin elde edilmesi gibi yöntemlerde kullanılmaktadır [8]. 2 boyutlu haritalamanın yanı sıra, ICP kullanılarak 3 boyutlu objeler üzerinde de başarılı sonuçlar elde edilmiştir [7]. Ancak 3 boyutlu haritalamada gezinilen yerlerin artmasıyla elde edilen haritanın büyümesi, ICP kullanımı için bazı kısıtların getirilmesini gerekli kılmaktadır.

Bu durumun üstesinden gelmek için RTAB-Map sadece RGB veri üzerinden belirlediği öznitelik noktalarını kullanarak görsel bir dönüşüm hesabı yapmaktadır [36]. Burada öznitelik eşlemesi yapılırken sadece konum bilgisinden değil aynı zamanda renk bilgisinden de faydalanılmaktadır. Bu durum, okunan her RGB veride yeteri kadar öznitelik bulunabildiği durumlarda avantaj sağlasa da aydınlatmanın yeterli olmadığı ya da yeterli öznitelik barındırmayan ortamlarda dezavantaj oluşturmaktadır. Bu nedenle, tez kapsamında yapılan araştırmanın yönü sadece derinlik bilgisinin ICP algoritmasında kullanılarak dönüşüm hesabının yapıldığı ve bu dönüşümün direkt olarak 3 boyutlu haritalamada kullanıldığı bir yaklaşıma çevrilmiştir.

Robot İşletim Sistemi (ROS - Robot Operating System) [48] paketi olarak geliştirilmiş olan Ethzasl-Icp-Mapper 2 ve 3 boyutlu ICP tabanlı haritalamayı libpointmatcher kütüphanesini kullanarak uygulamaktadır [9, 10]. Burada Kinect'ten alınan derinlik verisinin tamamı hem odometri takibinde hem de haritalamada kullanılmaktadır. Ancak Kinect'in varsayılan olarak ürettiği 640×480 boyutlarındaki derinlik resminin nokta bulutu verisine çevrildiğinden 307200 nokta oluşmaktadır. Alınan bu verinin SLAM algoritmasında direkt olarak kullanılması işlem maliyetini çok büyük oranda artırdığı için gerçek zamanlı haritalamayı imkansız hale getirmektedir. Bu durumun oluşturduğu olumsuzluğu engellemek için Kinect'ten alınan derinlik

resminin çözünürlüğü 160×120 olarak azaltılarak her bir aşamada haritaya 19200 noktanın eklenmesi sağlanmaktadır. Bununla birlikte verilerin işlenmesinin gerçek zamanlı olabilmesi için Kinect'in saniyede varsayılan olan 30 çerçeve yerine 5 çerçeve yayınlayacak şeklinde ayarlanması gerekmektedir. Ayrıca alınan anlık verinin harita üzerinde eşleştirilebilmesi için uzaklığı belirli bir mesafenin altında tutabilmek amacıyla kameranın çok yavaş hareket ettirilmesi gerekmektedir. Bunun yanı sıra ortam aydınlatmasından etkilenmiyor olması RTAB-Map'e karşı avantaj oluşturmaktadır.

Ethzasl-Icp-Mapper ile ilgili başka bir dezavantaj ise aynı bölgeden birden fazla veri alınması durumunda da ilgili bölgenin gereksiz sayıda fazla temsil edilmesine neden olmasıdır.

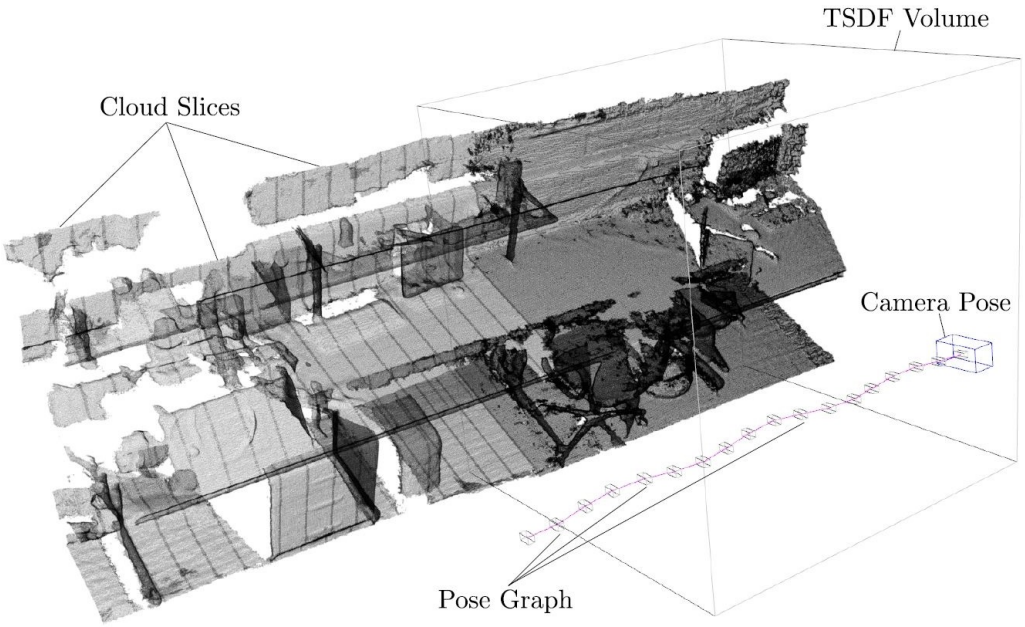
1.2.4 Kintinuous

Kintinuous RGB-D kameranın etrafındaki harita için TSDF olarak adlandırılan 3 boyutlu bir temsil şekli kullanmakta ve her zaman kamerayı bu TSDF alanının merkezinde tutmaya çalışmaktadır. TSDF yapısının içerisindeki her bir hücre, mevcut alan içerisinde kendisine en yakın yüzeye olan mesafeyi işaretli olarak tutmaktadır.

Kamera pozisyonu önceden belirlenen bir eşik değerinden fazla değiştiğinde, sistem kameranın TSDF alanının merkezinde kalmasını sağlamak için bu temsil alanını kaydırır. Kamera konum değişikliği her bir boyut için farklı değerlendirilmektedir. TSDF alanı kaydırıldığında, haritada daha önceden bu alan içerisinde temsil edilen bir kısım alanın dışına çıkarken yeni bir kısım da alanın içerisine girmiş olur. Çıkan kısım artık TSDF alanının bir parçası olmadığından, harita temsil modelini değiştirir ve içerisindeki yüzeyleri nokta bulutu verisine dönüştürür. Bu nedenle, ayrılan bu kısıma bulut dilimi (cloud slice) adı verilmektedir. Şekil 1.6 ile TSDF alanının dışına çıkmış olan ve nokta bulutu ile temsil edilen bulut dilimleri gösterilmektedir.

Kamera konumu harita koordinat düzlemine göre tutulur ve her bir adımdan sonra konum grafına yeni bir düğüm olarak eklenir. Şekil 1.6 ile kamera yörüngesini temsil eden konum grafı, kameranın konumu etrafında bulunan TSDF alanı ve kamera hareketi boyunca TSDF dışına çıkmış olan bulut dilimleri gösterilmektedir. Bulut dilimlerinin konum merkezleri, kamera konum grafının düğümlerine bağlıdır.

Sistem kamera konum tahmini için iki farklı metot kullanır. Geometrik tahmin olarak adlandırılan ilk metot, elde edilen mevcut derinlik bilgisini TSDF içerisindeki yüzeyler ile ICP kullanarak eşleme yapar. Fotometrik tahmin adı verilen diğer metot ise kamera hareketinin tahminini ardışık RGB-D resimleri kullanarak gerçekleştirir. Sonrasında



Şekil 1.6 Kintinuous'un TSDF alanını kaydırmasının görseli [4]

ise sistem geometrik ve fotometrik tahmin sonuçlarını ağırlıklandırarak toplayarak nihai bir konum tahmini hesaplar.

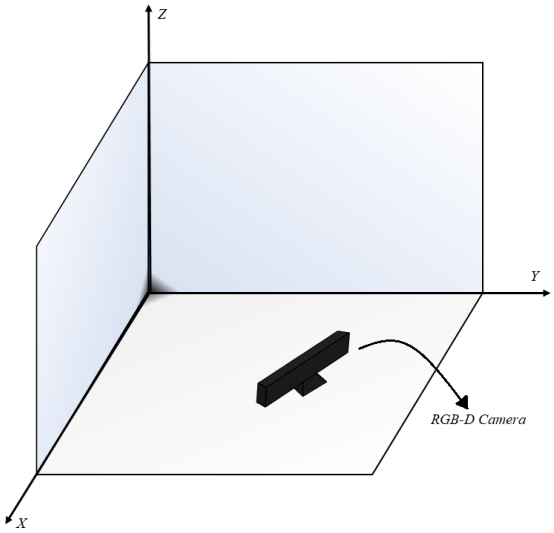
Kintinuous kapalı-döngü tahminini çıkarılan SURF öznitelik noktalarına bag-of-words yaklaşımını uygularak yapar. Eğer yeni gelen çerçeve ile bag-of-words içerisinde tanımlı bazı öznitelikleri barındıran çerçevelerden birisi arasında bir eşleşme bulunursa, üç adımlık bir doğruluma işleminden sonra bu çerçeveler arasında hesaplanan mesafe sisteme eklenir. İlk olarak, SURF öznitelikleri üzerinde k-en yakın komşu algoritması uygulanarak karşılık gelen noktalar bulunur. Sonra, bulunan bu noktaların bir kısmını elemek için RANSAC tahmini kullanılır. Son olarak, kalan noktaların derinlik bilgisi ve oluşturulan harita arasında lineer olmayan bir ICP uygulanır. Elde edilen transformasyon kapalı-döngü tespiti yapabilmek için kullanılır.

Bütün bu adımlar ve kapalı-döngü tespiti sonrasında hem kamera konum grafının hem de haritanın optimizasyonu için iSAM sistemini kullanan bir adım uygulanır [28]. Kintinuous paralel çalışma sağlamak için her bir modülü farklı bir thread içerisinde uygulamaktadır. Ancak Kintinuous, RTAB-Map'in kullandığı gibi bir hafıza yönetim sistemi kullanmadığından oluşturulan harita büyüdükçe gerçek zamanlı çalışma performansını da kaybetmektedir.

1.3 Hipotez

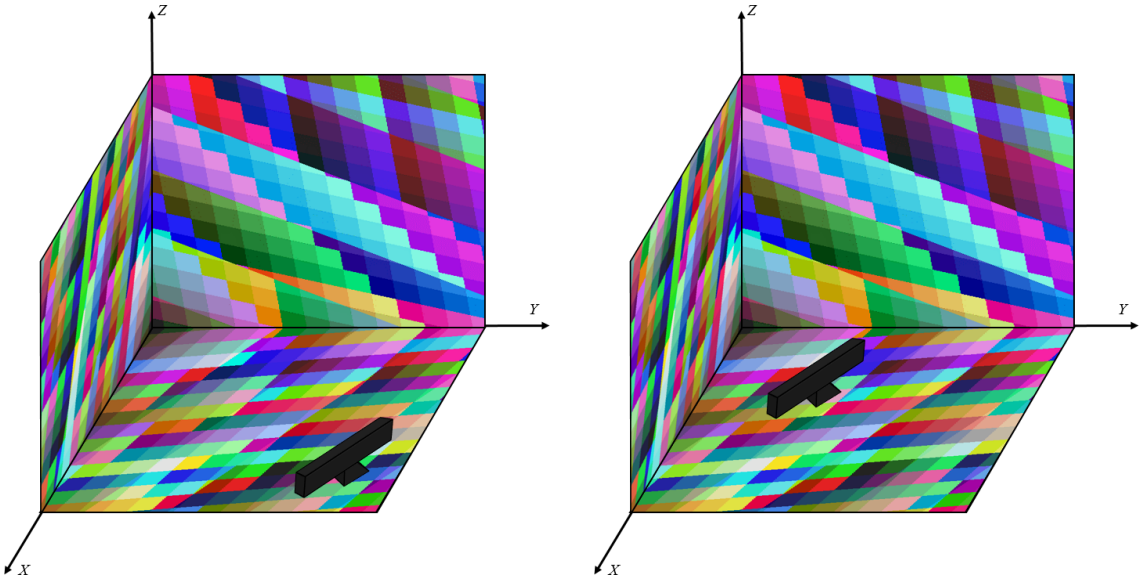
Tez kapsamında yapılan incelemeler ve karşılaştırmalar sonucu RTAB-Map'in mevcut sistemler içerisinde en başarılı haritalama sistemi olduğu anlaşılmıştır. Ancak

RTAB-Map'in RGB veriye olan bağımlılığı dolayısıyla Şekil 1.7 ile örnek olarak gösterilen bir ortamda olduğu gibi sadece tek renk duvar ve zeminden oluşan durumlarda yeterli öznitelik bulamaması dolayısıyla başarısız olmaktadır.



Şekil 1.7 Mevcut sistemlerin başarısız olduğu durumlara bir örnek

Bu nedenle tez kapsamında, görsel SLAM sistemleri tarafından tespit edilen ve odometri takibi ile haritalamada kullanılabilecek öznitelik sayısını arttırmak amacıyla Şekil 1.7 ile örneği verilen alanlarda alınan RGB resmin renklendirilmesi önerilmektedir. Renklendirme işleminin detayları Bölüm 5 ile anlatılmaktadır. Tez kapsamında geliştirilen sistemin Şekil 1.7 ile verilen alan için çıktısı Şekil 1.8 ile verildiği gibi olmakta ve kamera konumundan etkilenmemektedir.



Şekil 1.8 Farklı kamera pozisyonlarından etkilenmeden veri renklendirmenin yapılması

2
Haritalama Sistemlerinin Temelleri

3 boyutlu haritalama, adından da anlaşılabileceği gibi, temel olarak gerçek dünyadan bir ortamın sanal ortamda 3 boyutlu modelini çıkarmaktır. Modeli çıkarılan ortam bir oda ya da koridor gibi iç mekan olabileceği gibi sokak ya da meydan gibi dış mekan da olabilir. Haritalanan ortamın büyüklüğü, içerisinde bulunan nesnelerin çokluğu ve/veya karmaşıklığı haritalama işleminin zorluk derecesine etki etmektedir. Bununla birlikte oluşan haritadan beklenen özellikler de aslında aşağıda listelenen problemleri tanımlamak açısından önemli arz etmektedir.

- **Kesinlik/Doğruluk:** Haritalarda oluşan hatalar genel olarak haritalama sırasında karşılaşılan belirsizliklerden kaynaklanmaktadır. İlk etken fiziksel dünyanın aslında tahmin edilemez olmasıdır. Haritalamaya çalıştığımız ortamın statik/dinamik olması ya da hiç bir zaman bir ortamı tam olarak gözlemleyemiyor olmamız gibi çeşitli özellikler haritalama sırasında doğuruluğu etkileyen faktörlerdir. Bunun dışında sensörlerden alınan verilerin sınırlı olması ve gürültü içermesi de haritada oluşabilecek hata oranını arttırmaktadır. Sensörden gelen hatalı veri ve bu hatanın minimize edilmesi yapılan kalibrasyon işlemi Bölüm 2.3.1 ile anlatılmaktadır. Ancak kalibrasyon sonrası dahi sensörden gelen verinin hatalı olma ihtimalini göz ardı edilmemelidir. Doğruluğu olumsuz yönde etkileyen başka bir sebep de robota verilen komutun çeşitli iç ve dış etkenlerden dolayı %100 doğrululukla gerçekleştirilemiyor olmasıdır.
- **Hafıza:** Gerçek dünya, esasen, 3 boyutlu sonsuz veri içeren ve sürekli (continuous) olan bir özelliğe sahiptir. Herhangi bir ortamı modellemek ise o ortamı dijital hale indirgemek demektir. Bu indirgeme ise, kaçınılmaz olarak, bazı verilerin kaybolmasına yol açmaktadır. Bu nedenle kesinliği artırmak için bu indirmeyi minimuma çekme düşüncesi teoride mantıklı olsa da pratikte mümkün olmamaktadır. Çünkü bilgisayarların hafıza açısından kapasiteleri sınırlı olmasından dolayı; hafıza, haritalama sırasında kullanılan algoritmayı geliştirirken göz önüne alınması gereken önemli konulardan biri haline gelmektedir.

- **Zaman:** Bu problemi çözmek için bazı algoritmalar gerçek zamanlı değil de daha ziyade önceden kaydedilmiş veriler üzerinde çalışmak üzere geliştirilmiştir. Sürekli büyüme kapasitesine sahip bir modelin tamamı üzerinde çeşitli işlemler ve hesaplamalar yapmanın, ne kadar güçlü bir CPU kullanılırsa kullanılsın, masraflı bir süreç olduğu aşikardır. Bazı algoritmalar zaman problemini indirgemek için kesin işlemler yerine bazı yaklaşımlar kullanmaktadır. Böylelikle hesaplama maliyetini indirgeyerek bu probleme kısmen de olsa bir çözüm getirmektedirler.

Bu bölümde, tez kapsamında önerilen hipotezin öncesinde mevcut SLAM sistemleri üzerinde yapılan ön inceleme sonucu elde edilen kazanımlar anlatılmaktadır. Bölüm 2.1 haritalama sistemlerinin tarihçesi hakkında kısa bir bilgi verdikten sonra Bölüm 2.2, 2 boyutlu haritalamanın 3 boyutlu ortama entegresine yönelik tez kapsamında yapılan çalışmanın detaylarını açıklamaktadır. Son olarak, Bölüm 2.3, tez kapsamında giriş olarak kullanılan RGB-D verisini sağlayan sensörler ve bu sensörlerin özelliklerini anlatmaktadır.

2.1 Tarihçe

3 boyutlu haritalamanın gelişim tarihine bakıldığında temelinde 2 boyutlu haritalamanın yattığı görülmektedir. İlk haritalama modelleri olarak 80'lerin sonlarına doğru grid tabanlı [49] ve parçacık tabanlı [50] yaklaşımlar birbirlerine alternatif oluşturarak geliştirilmiştir.

Sonrasında, yapısal olarak tahmin edilemez ortamların da haritalanabilmesi adına robotik alanında sensörler ve öngörülemeyecek sayıda bulunan durumların üstesinden gelebilecek nitelikte olan robot yazılımları önem kazanmıştır [1]. Robot sistemlerinde karşılaşılan belirsizlik aşağıda belirtilen beş farklı etkenden kaynaklanmaktadır.

1. **Ortam:** Haritalanan fiziksel dünya tahmin edilemez bir yapıya sahiptir.
2. **Sensörler:** Haritalamada kullanılan sensörler iki farklı kısıtlamaya sahiptir. Birincisi menzil ve çözünürlük, ikincisi ise veride bulunan gürültüdür.
3. **Robotlar:** Robot hareketlerinde gürültü ve yıpranmadan kaynaklı hatalar olabilmektedir.
4. **Modeller:** Hiç bir harita temsil modeli gerçek dünyayı %100 temsil etme özelliğine sahip değildir.

5. **Hesaplama Maliyeti:** Robotik sistemlerde hesaplama maliyetinin yüksek olması sebebiyle bazı algoritmalar yuvarlama işlemi yaparak maliyeti azaltmaktadır. Ancak bu yuvarlama işlemi doğruluğun belirli bir oranda kaybedilmesine neden olmaktadır.

90'ların başlarında haritalama algoritmalarına yeni bir yaklaşım getirilerek robot sistemlerinde karşılaşılan bu belirsizliğin açıkça temsil edilebilmesi için olasılık tabanlı çözümler üretilmiştir [51]. Olasılıksal algoritmalar mevcut durumu, tek bir "en iyi tahmin" yerine tüm durumları içeren bir olasılık tahmin dağılımı ile temsil etmektedir. Durum temsili için örnek bir olasılık tahmini Denklem 2.1 ile verilmiştir. Olasılıksal yaklaşımlar yukarıda listelenen etkenlerden kaynaklanan belirsizliklere karşı daha dayanıklıdır.

$$P(x_t|z_t, u_t) \tag{2.1}$$

Burada x_t değişkeni t anında robotun konumunu, $z_{1:t}$ değişkeni t anında robotun sensör ölçümünü, u_t değişkeni ise t anında robotun hareket seçimini ifade etmektedir.

Olasılıksal haritalama sistemlerinde, robot ile haritalanan ortam arasında iki temel etkileşim tipi vardır: Robot, sensörlerden gelen verileri kullanarak mevcut durum hakkında bilgi güncellemesi yapabilmekte ve ortam içerisinde belirli bir kontrol hareketi gerçekleştirerek mevcut durumunu değiştirebilmektedir. Böylece robot iki farklı kaynaktan veri elde etmektedir:

- **Ölçüm verisi:** Algılama işlemiyle ortamın anlık durumu hakkında bilgi sağlamaktadır. Denklem 2.1 ile verilen z_t ifadesi t anında yapılan ölçüm verisini temsil etmektedir ve robotun tek seferde tam olarak bir ölçüm yaptığı varsayılmaktadır.
- **Kontrol verisi:** Hareket sonucu robotun durumunun değişimi hakkında bilgi taşır. Kontrol verisine örnek olarak robotun hızı ya da odometri bilgisi verilebilir. Denklem 2.1 ile verilen u_t kontrol verisi $(t-1; t]$ zaman aralığında robotun durumunun değişimini ifade etmektedir ve tek seferde tam olarak bir tane olduğu kabul edilmektedir.

Algılama işlemi mevcut durum hakkında bilgi sağlamaktadır; dolayısıyla robotun bilgisi artma eğilimindedir. Öte yandan, hareket, robotun çalışmasındaki doğal gürültü ve ortamın rastgeleliği nedeniyle bilgi kaybına neden olma eğilimindedir.

Hareket ve algılama işlemleri bu şekilde farklı görevler taşıdığından sistem içerisinde ayrıştırılması önem kazanmaktadır. Bu nedenle, haritalama sırasında, robot sensör okurken hareket etmemektedir.

Olasılıksal haritalamada bir başka anahtar kavram robotun iç durum bilgisini ifade eden inanç dağılımıdır ($bel(x_t)$). Robot mevcut durumunu direkt olarak ölçümleyemez, bunun yerine konumunu yukarıda anlatılan verilerden çıkarmak zorundadır. Bu nedenle, inanç ($bel(x_t)$) robotun durumundan ziyade robotun *bilgi durumu*'nu temsil etmektedir. Denklem 2.2 ile verilen inanç dağılımı mevcut verilere bağlı olarak durum değişkenleri üzerinde olasılık değerleri tutmaktadır.

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{2.2}$$

Burada $bel(x_t)$ inancı, bütün $u_{1:t}$ hareketleri ve $z_{1:t}$ ölçümleri sonrası t anında x_t konumu için olasılık dağılımını ifade etmektedir.

Denklem 2.2 ile verilen olasılık dağılımı, z_t algılama işlemiyle robot bilgi seviyesinin güncellenmesi sonrası elde edilen inancı vermektedir. Denklem 2.3 ise z_t algılama işlemi öncesinde u_t kontrol hareketinden hemen sonra elde edilen dağılımı ifade etmektedir.

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) \tag{2.3}$$

Denklem 2.3 ile verilen olasılık dağılımı *tahmin (prediction)* olarak geçmektedir. Çünkü, burada t anında herhangi bir sensör okuması yapılmadan $\overline{bel}(x_t)$ ile durum tahmini yapılmaktadır. $\overline{bel}(x_t)$ dağılımından sensör okunmasıyla $bel(x_t)$ inancının hesaplanmasına ise *düzeltme (correction)* denmektedir.

Bugün kullanılan haritalama yöntemlerine güçlü bir alt yapı oluşturmuş olan bu olasılıksal yaklaşım, sırayla yapılan hareket ve algılama işlemlerinin her birinin ardından harita ve robot pozisyonu hakkında tutulan $bel(x)$ ve $\overline{bel}(x)$ olasılık dağılımlarını birbirine bağlı olarak güncelleyip düzeltmektedir [52].

Bu tarz, haritalama ve konumlandırmayı eş zamanlı olarak tahmin edip güncelleyen yöntemlere, daha önce de belirtildiği gibi genel olarak SLAM denmektedir [53]. Bu zamana kadar ise bu yaklaşımı temel alan bir çok algoritma geliştirilmiştir. 2 boyutludan 3 boyutluya geçişte de bu yaklaşım üzerinden geliştirmeler yapılmıştır. Tez kapsamında üzerinde çalışılan ve iyileştirme yapılması hedeflenen sistemler de SLAM yaklaşımını kullanmaktadırlar.

2.1.1 2 Boyutludan 3 Boyutluya Geçiş

2 boyutlu haritalama yöntemlerinde belirli bir başarı elde edilmesi, ayrıca lazer ve kinect gibi araçlarla ortamdan 3 boyutlu veri alımının sağlanması araştırmaların 3 boyutlu haritalamaya kaymasına neden olmuştur. Ancak 3 boyutlu haritalama beraberinde ekstra masraflar da çıkarmıştır. Mesela 2 boyutlu haritalamada Denklem 2.4 ile gösterildiği gibi robotun konumunu belirlemek için (x, y) konumu ve robotun yönü olan α olmak üzere 3 değişken tutmak yeterli olurken 3 boyutlu haritalamada gerekli değişken sayısı (x, y, z) konumları ve α, β, γ değişkenleri ile belirtilen dikey (yaw), yanal (pitch) ve boylamsal (roll) eksendeki açı bilgileri olmak üzere iki katına çıkmaktadır [54]. Böylece 2 boyutlu sistemler 3 DoF ile çalışırken 3 boyutlu sistemlerin 6 DoF ile çalışma gerekliliği oluşmaktadır. Bu nedenle 2 boyutlu haritalamada dahi karşımıza çıkan hafıza sorunu burada daha bir ehemmiyet kazanmaktadır.

$$3DoF: (x, y, \alpha) \rightarrow 6DoF: (x, y, z, \alpha, \beta, \gamma) \tag{2.4}$$

Bunun dışında, 3 boyutlu haritalama algoritmaları incelendiğinde 2 farklı yaklaşımın olduğu göze çarpmaktadır. Bunlardan ilki kullanılan 2 boyutlu haritalama üzerinde 3 boyutlu sensör verisinin yerleştirilmesiyle oluşturulan ortam modelidir [55]. Burada direkt 3 boyutlu sensör kullanımı yerine 2 boyutlu sensörden çeşitli yöntemlerle 3 boyutlu veri alınmaktadır. Örneğin yatay ve dikey olarak yerleştirilmiş iki lazerden birisi kullanılan SLAM algoritmasının çalıştırılmasında veri kaynağı işlevi görürken diğerinden alınan veriler oluşturulan harita üzerine yerleştirilebilmaktedir. Bir başka yöntem ise 3 boyutlu verinin lazeri dikey eksen etrafında döndürmek suretiyle toplanmasıdır [55]. Bu yaklaşımda kullanılan SLAM algoritması ise 2 boyutlu olmaktadır. 3 boyutlu haritalamalardaki diğer bir yaklaşım ise SLAM algoritmasının direkt olarak 6 DoF ile çalışabilecek şekilde geliştirilmesidir [56]. Bu yaklaşımda, tez kapsamında kullanılan RGB-D sensörler kullanılabileceği gibi 3 boyutlu bir lazer sensörden alınan veriler kullanılarak ICP uygulamak mümkündür [56].

2.2 2 Boyutlu Haritalama

Literatürde başarılı sonuçlar elde edilen tüm 3 boyutlu haritalama yöntemleri RGB veriye bağımlı iken 2 boyutlu haritalama yöntemlerinde RGB veri olmadan yüksek başarılar elde edilebiliyor olması dolayısıyla, tez çalışması 2 boyutlu haritalama yöntemlerinin incelenerek 3 boyutlu sisteme entegresinin mümkün olup olmadığının araştırılmasına yöneltilmiştir.

Yıldız Teknik Üniversitesi Olasılıksal Robotik Grubu [57] katıldıkları 2015 Robocup German Open yarışmasında [58], yarışma alanı içerisinde Şekil 2.1 ile gösterilen robotun gezdirerek Kinect ve lazer dahil olmak üzere pek çok sensör verisi toplamışlardır.

Şekil 2.1 2015 Robocup Rescue German Open yarışmasında Yıldız Teknik Üniversitesi Olasılıksal Robotik Grubu tarafından kullanılan robot

Öncelikle literatürde başarılı bir 2 boyutlu haritalama yöntemi olarak bilinen Hector haritalaması [8] 3 boyutlu hareket tahmini de yapabildiğini ileri sürüldüğünden çalışma kapasamında incelenmek üzere seçilmiştir. 2015 Robocup German Open yarışma alanında toplanan lazer verisi kullanılarak Robocup Rescue [58] yarışma alanının Şekil 2.2 ile gösterilen 2 boyutlu haritası Hector haritalaması ile çıkarılmıştır.

Şekil 2.2 2015 Robocup Rescue German Open yarışma alanın Hector haritalaması ile lazer verisi kullanılarak çıkarılan 2 boyutlu haritası

Şekil 2.2 ile gösterilen beyaz çizgilerle belirtilen kısımlar lazerden okunan veriler olup, kırmızı çizgiyle belirtilen kısım ise ROS tabanlı depthimage-to-laserscan paketinin çalıştırılmasıyla Kinect'ten alınan derinlik resmi üzerinde bir kesitten elde edilen yapay lazer verisidir [59]. Hector algoritmasına giriş olarak lazer yerine 2 boyutlu uzayda kapsadığı alan çok kısıtlı olan bu yapay lazer verildiğinde yeteri kadar veri sağlanamadığından haritalamanın başarısız olduğu gözlemlenmektedir. Ayrıca Şekil 2.2 ile de açıkça gözüktüğü gibi Kinect'in hassasiyeti lazer sensörü kadar yüksek değildir.

2.3 RGB-D Sensör Verisi

Aşağıda listelenen sensörler sadece haritalama için değil genel olarak bilgisayar sistemlerinin dış dünyadan haberdar olmak için kullandıkları araçlardır [1]. Örneğin bumper (tampon) temas gerektirdiğinden aktif olarak haritalamada kullanılmak yerine hata kontrolü amaçlı çarpma durumlarında sistemi uyarmak için tercih edilmektedir.

- **Temas Sensörleri:** Bumper
- **İç Sensörler:** IMU, Gyroscope, Pusula, Eğimölçer
- **Mesafe Ölçer Sensörler:** Sonar, Radar, Laser Range Finder, Infrared
- **Görsel Sensörler:** Kamera, RGB-D Kamera, Stereo Kamera
- **Uydu Tabanlı Sensörler:** GPS

Bu sensörler içerisinde 3 boyutlu haritalamada mil taşı görevi gören sensörlerden ilki lazer ikincisi de RGB-D kameradır. 1960 yılında çıkan lazerin ortam hakkında hata payı düşük bilgiler vermesi, öncelikle haritalamada kullanmaya elverişli olmasını sağlamıştır. Sonrasında ise iki farklı lazerin farklı açılarla yerleştrilerek ya da bir lazerin hareketiyle ortam hakkında 3 boyutlu verilerin elde edilmesi düşüncesi doğmuştur. Ancak lazerin pahalı olması bu alanda yapılan çalışmaları oldukça kısıtlamıştır. 2010 yılında ise Microsoft'un oyun konsolu olarak Kinect'i piyasaya sürmesi maddi açıdan sıkıntıyı ortadan kaldırdığı için 3 boyutlu haritalama ile ilgili çalışmalar hız kazanmıştır.

Çalışmaların genel olarak görsel SLAM çerçevesinde yürütülmesinin planlanması dolayısıyla RGB-D veri sağlayan sensör kullanımı kaçınılmaz olmuştur. Piyasada RGB-D veri sağlayan iki tip kamera olması dolayısıyla tez kapsamında her iki kameranın karşılaştırılması yapılmıştır.

Bunlardan ilki içerisinde RGB kameranın yanı sıra derinlik sağlama amacıyla kızılötesi kamera da bulunduran RGB-D sensörlerdir. İlk olarak Microsoft'un 2010'da piyasaya sürdüğü Kinect XBOX (Şekil 2.3a) ile dünyaya tanıtılan ve diğer 3 boyutlu sensörlere göre çok daha düşük maliyete sahip olması dolayısıyla çalışmalarda tercih edilmeye başlanan bu kameraların günümüzde farklı marka ve modelleri de mevcuttur. Kinect 43° dikey ve 57° yatay görüş açısına sahiptir. RGB görüntü olarak saniyede 640×480 boyutlarında 30 çerçeve verebilmektedir. Her bir piksele karşılık gelen derinlik bilgisi de aynı frekansta ve aynı çözünürlükte üretilmektedir.

(a) Kinect XBOX **(b)** Zed stereo kamera

Şekil 2.3 RGB-D sensör çeşitleri

RGB-D sensörlerden bir diğeri de stereo kameralardır. Bu sensörler içerisinde bulundurduğu yan yana iki RGB kameradan alınan görüntülerin karşılaştırılmasıyla derinlik bilgisi elde etmektedir. Derinlik bilgisinin kızılötesi kamera kullanılarak elde edildiği sensörlele maksimum 4 metreye kadar derinlik verisi alınabiliyorken, stereo kameralarla daha uzak mesafelerden de derinlik verisi alınabiliyor olması kamera performans karşılaştırmasının yapılması ihtiyacını oluşturmaktadır. Tez kapsamında kullanılan Zed stereo kamera Şekil 2.3b ile verilmiştir. Zed derinlik bilgisini kameralarından okuduğu RGB görüntüde oluşan kayma miktarını kullanarak elde ettiği için ortam aydınlatmasının yetersiz olduğu durumlarda başarılı bir derinlik bilgisi sağlayamamaktadır. Bu nedenle, Kinect kullanımına karar verilmiştir.

2.3.1 Kinect İç ve Dış Kalibrasyonu

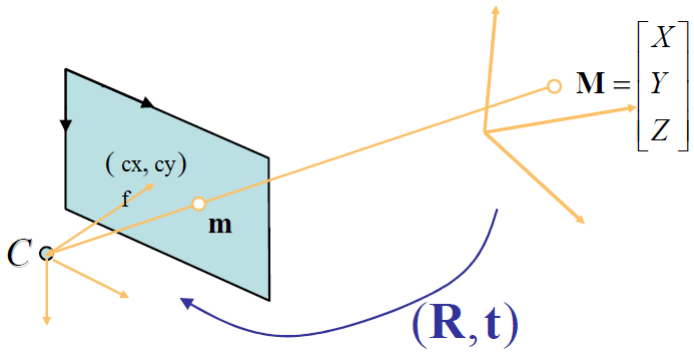
Tez kapsamında kullanılan sensörden alınan RGB resim ve derinlik bilgisinin üzerine oturtturularak 3 boyutlu düzlemde renkli bir nokta bulutu elde edilmekte ve haritalamada bu nokta bulutu kullanılmaktadır. Ancak kameranın varsayılan iç parametreleri kullanıldığı zaman Şekil 2.4 ile de gösterildiği gibi RGB ve derinlik resimlerinde yapılan eşlemede hatalar oluşabilmektedir. Şekil 2.4a RGB kameradan alınan resmi, Şekil 2.4b ise derinlik bilgisiyle RGB resmin birleştirilmesi sonucu elde edilen nokta bulutunu göstermektedir. Üretilen bu nokta bulutu bir sisteme verilecek olsa sistemin başarısını olumsuz etkileyeceği açıktır. Yapılan iç ve dış kalibrasyon ile bu hataların minimuma indirgenerek 3B SLAM sistemlerine daha iyi veri sağlanması amaçlanmıştır.

(a) (b)

Şekil 2.4 Kinectten okunan (a) RGB resim ve (b) RGB ve derinlik resimlerinden oluşturulan nokta bulutu

2.3.1.1 İç Kalibrasyon

Şekil 2.5 ile görselleştirilen iğne deliği kamera modelinde 3 boyutlu uzayda bulunan ve $\mathbf{M} = [X, Y, Z]^T$ olarak verilen bir noktanın, 2 boyutlu bir düzlemdeki görüntüsünün $\mathbf{m} = [x, y]^T$ noktasında oluştuğu varsayılmaktadır.



Şekil 2.5 İğne deliği kamera modeli

$\mathbf{m}$, $\mathbf{M}$ için genişletilmiş gösterimler sırasıyla $\tilde{\mathbf{m}} = [x, y, 1]^T$ ve $\tilde{\mathbf{M}} = [X, Y, Z, 1]^T$ şeklindedir. 3 boyutlu noktanın perspektif dönüşümü ile oluşan görüntüsü Denklem (2.5) ile verilebilir.

$$s\tilde{\mathbf{m}} = \mathbf{A}[\mathbf{R}|\mathbf{T}]\tilde{\mathbf{M}} \tag{2.5}$$

$[\mathbf{R}|\mathbf{T}]$ döndürme-öteleme matrisi, $\mathbf{A}$ ise kamera matrisi veya içsel parametreler olarak isimlendirilir. s değeri ise keyfi bir ölçek parametresini göstermektedir. $\mathbf{A}$ matrisi Denklem (2.6) ile verildiği şekilde tanımlanır.

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

c_x ve c_y görüntüleme sisteminin optik merkez koordinatlarını, f_x ve f_y ise odak uzaklıklarını vermektedir.

Görüntüleme sistemlerinde kullanılan lensler radyal ve teğetsel (tangential) deformasyonlara sebep olmaktadır. Radyal deformasyon görüntüde balık gözü etkisi yaratır. Teğetsel deformasyon ise görüntü düzlemi ile lens düzleminin pratikte tam paralel olarak yerleştirlememesinden kaynaklanmaktadır. Radyal ve teğetsel bozulma sonucunda (x, y) noktasında görünen bir noktanın gerçek koordinatı (x^*, y^*), polinom deformasyon modelleri kullanılarak, $r^2 = x^2 + y^2$ olmak üzere sırasıyla Denklem (2.7) ve (2.8) ile verilebilir.

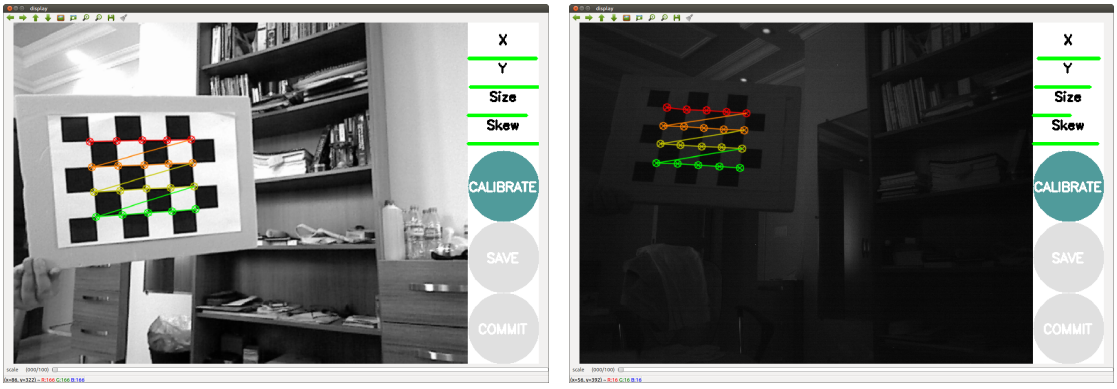
$$\begin{aligned} x^* &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y^* &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{2.7}$$

$$\begin{aligned} x^* &= x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right] \\ y^* &= y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right] \end{aligned} \tag{2.8}$$

Kamera iç kalibrasyonu ile c_x, c_y, f_x, f_y, k_1, k_2, k_3, p_1 ve p_2 parametreleri kestirilerek düzeltilmiş görüntü elde edilmektedir.

Kinect temelde iki farklı kamera bulundurmaktadır; RGB kamera ve derinlik verisinin ölçülmesi için kızılötesi kamera. İç kalibrasyonun her iki kameraya da yapılarak her iki kamera için kamera parametrelerinin elde edilmesi gerekmektedir. Tez kapsamında iç kalibrasyon için ilgili ROS paketi kullanılmıştır [60]. İç kalibrasyon yapılırken kamera önünde boyutları bilinen bir dama tahatası farklı açı ve uzaklıklarda görülecek şekilde hareket ettirilerek yapılmaktadır. Şekil 2.6 kinect kamera iç kalibrasyonu sırasında örnek bir ekran görüntüsü vermektedir.

Kızılötesi kameranın iç kalibrasyonu yapılırken RGB kameradan farklı olarak öncelikle, kızılötesi projectör bir engel ile kapatılmış ve dış aydınlatma ile aydınlatılan dama tahtası deseni kullanılarak iç kamera parametreleri elde edilmiştir.



(a) RGB kamera **(b)** Kızılötesi kamera

Şekil 2.6 İç kalibrasyon örnek ekran görüntüsü

2.3.1.2 Dış Kalibrasyon

Kamera dış kalibrasyonu, kalibre edilen iki kameranın birbirlerine göre konumlarını belirlemek için yapılmaktadır. Döndürme-öteleme matrisi [**R**|**T**], dışsal parametreler olarak isimlendirilir, sahne ile sabit kamera koordinat sistemi arasındaki dönüşümü tanımlar ve 3×3 döndürme matrisi ile 3×1 öteleme vektöründen oluşur. Her iki kameranın birbirine göre konumu bir referans koordinat sistemine göre hesaplanarak kameralar arası dış kalibrasyon yapılmış olur.

Kinect içerisinde bulunan iki kameranın birbirlerine olan mesafe ve açıları varsayılan olarak üretici firma tarafından verilmiştir. Ancak kullanılan sensör için dış kalibrasyonun yapılması elde edilen verinin kesinliğini arttıracaktır.

Dış kalibrasyonun yapılabilmesi için ilk aşama olarak her iki kamera için de iç kalibrasyonun yapılmış olması beklenmektedir. Dış kalibrasyon, boyutları bilinen bir dama tahtasının her iki kamera ile alınan görüntüsü kullanılarak yapılmaktadır. Ancak, Kinect üzerinden aynı anda hem RGB hem de kızılötesi kameradan görüntü alınamamaktadır. Bu nedenle, kamera ve dama tahtasının konumu değiştirilmeden bu resimlerden birisi önden kaydedilerek işlemin gerçekleştirilmesi gerekmektedir. Dış kalibrasyon sonucunda kameraların birbirlerine olan konum bilgisi elde edilir. Böylece RGB ve derinlik resimlerinin birleştirilmesindeki hata payı minimuma indirgenmektedir.

2.3.2 Sensör Verisinin İncelenmesi

Farklı ortam aydınlatması sonucu başarıda oluşan değişimlerin objektif bir şekilde ölçülebilmesi için ilk olarak ortam ve sensör verisinde oluşacak olası hatadan bağımsız sonuçlar elde etmek amacıyla Şekil 2.7 ile panoramik resmi verilen ortam çeşitli ışık kaynakları ile aydınlatılarak Kinect'ten elde edilen farklı çözünürlüklü RGB-D veriler

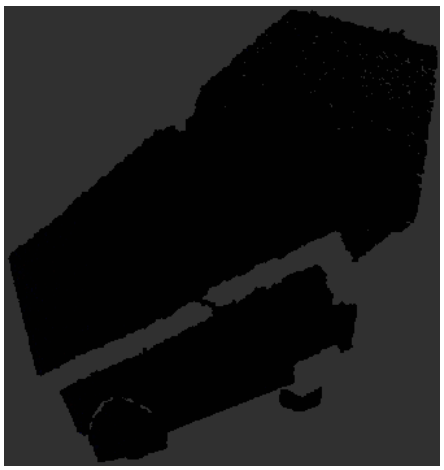
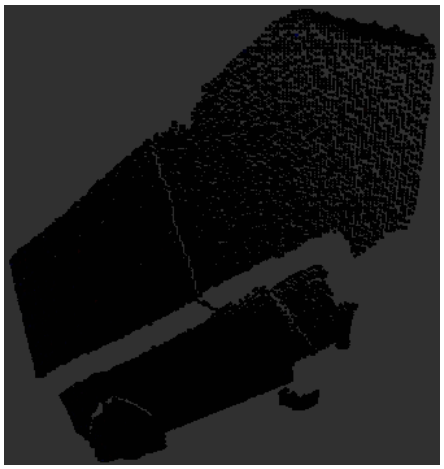
karşılaştırılmıştır.

Şekil 2.7 Test ortamının panoramik görüntüsü

Şekil 2.7 ile verilen ortamda kırmızı çerçeve ile alınan kısmın karanlık ve aydınlık olduğu durumlarda Kinect'in yüksek (640×480), orta (320×240) ve düşük (160×120) çözünürlüklü olarak elde edilen anlık nokta bulutu verileri Tablo 2.1 ile gösterilmektedir.

Şekil 2.7 ile verilen ortamın farklı seviyelerdeki ışık kaynakları ile aydınlatıldığı sırada alınan RGB-D verileri de karşılaştırılmıştır. Sonuçlar Tablo 2.1 ile gösterilenle aynı olduğundan ekran görüntüleri eklenmemiştir. Tablo 2.1 ile de görüldüğü üzere Kinect sensörden gelen verilerin güvenilirliğinin ortam aydınlatmasından ve çözünürlükten etkilenmediği görülmüştür. Bu durum farklı 3 boyutlu haritalama yöntemlerinin değişik şartlarda karşılaştırılabilmesine olanak sağlamaktadır.

Tablo 2.1 Farklı aydınlatma ve çözünürlükte Kinect'ten alınan nokta bulutu verisi sonuçları

	Aydınlık Ortam	**Karanlık Ortam**
Yüksek Çözünürlük 640×480		
Orta Çözünürlük 320×240		
Düşük Çözünürlük 160×120		

3
3 Boyutlu Haritalama Sistemlerinin Detaylı İncelemesi

Tez kapsamında yapılan çalışmalarda, daha önce de belirtildiği gibi, haritalama başarısının test edilmesinde ilk olarak RTAB-Map kütüphanesine ağırlık verilmiştir. Kullanım olarak RTAB-Map programı direkt yüklenebildiği gibi ROS paketi de mevcuttur. RTAB-Map kurulum ve kullanım detayları Ek A ile anlatılmıştır. Bölüm 3.1 RTAB-Map sisteminin farklı parametrelerde performans sonuçları incelenirken Bölüm 3.2 ile değişik özelliklere sahip farklı ortamlarda haritalama sonuçları verilmiştir. Bölüm 3.3 ile RTAB-Map sisteminin kendi görsel odometrisi yerine farklı odometriler kullanılarak oluşturulan haritalardaki başarı oranı açıklanmış böylece sadece 3 boyutlu değil aynı zamanda 2 boyutlu yöntemlerle de karşılaştırılmış imkanı elde edilmiştir. Son olarak, Bölüm 3.4 ile giriş olarak RGB resim yerine farklı bir veri ile RTAB-Map sistem performansı incelenmiştir.

3.1 Farklı Parametrelerle 3 Boyutlu Haritalama

İlk olarak, RTAB-Map, içerdiği parametrelerin sistem başarısına etkisi ve haritalanan ortam ile parametrelerin ilişkisi açısından incelenmiştir. Çok sayıda parametre olduğundan sadece aşağıda listelenen ve performans üzerinde en etkili olan parametreler ve etkileri incelenmiştir. Sadece bu parametreler için binden fazla olası kombinasyon söz konusu olduğundan parametrelerin birbirleriyle olan etkileşimini ve sisteme genel etkisini görmek için tek seferde en fazla üç değişkene varsayılanı dışında değer verme gibi bir kısıt kullanılmıştır.

- Visual Word
  - Visual Word Type (VWT): SURF (varsayılan), SIFT, ORB, FAST+FREAK, FAST+BRIEF, GFTT+FREAK, GFTT+BRIEF, BRISK, GFTT+ORB, KAZE, ORB-OCTREE
  - Nearest Neighbor Strategy (NNS): FLANN Linear (varsayılan), FLANN KdTree, FLANN LSH, Brute Force

- RGBD-SLAM
  - Loop Closure Constraints (LCC): Vis (varsayılan), ICP, VisICP
  - ICP
- Odometry
  - Odometry Strategy (OdSt): Frame-to-Map (varsayılan), Frame-to-Frame, Fovis, viso2, DVO-SLAM, ORB_SLAM2, OKVIS, LOAM, MSCKF_VIO
  - Feature Detector (FeDe): SURF, SIFT, ORB, FAST+FREAK, FAST+BRIEF, GFTT+FREAK, GFTT+BRIEF (varsayılan), BRISK, GFTT+ORB, KAZE, ORB-OCTREE

"Visual Word" parametreleri resimler üzerinden konum hesabında kullanılmak üzere çıkarılan öznitelikler için kullanılan yöntemi ve bu özniteliklerden oluşturulan kelimelerin bulunduğu sözlük yapısını belirlemektedir. Burada parametre seçiminde dikkat edilmesi gereken nokta her öznitelik tanımlayıcı yönteminin her sözlük yapısıyla kullanılamıyor olmasıdır. Örneğin SURF ya da SIFT yöntemlerinden biri kullanıldığında, sistem sözlüğün modellenmesinde FLANN LSH kullanılmasına izin vermemektedir. Tablo 3.1 ve 3.2 "Visual Word" parametrelerinin sisteme etkisini vermektedir.

Tablo 3.1 ile özellik sütununda ilgili parametrenin diğerine göre öne çıkan ve tercih sebebi olmasına neden olacak özellik vurgulanmıştır. Tercih sütununda ise sistemin kullanımı sırasında, hangisinin tercih edileceğini belirleyen şartlar verilmiştir. SIFT normalde SURF öznitelik tanımlayıcısına göre daha çok öznitelik çıkarabildiği için daha başarılı bir algoritma kabul edilir ancak şu da bir gerçek ki SURF başarısız bir algoritma olarak tanımlanamaz. Başka bir ifadeyle, SURF algoritmasıyla bulunan özniteliklerin sayısı genel olarak haritalama için yeterli olmaktadır. Bununla birlikte, SIFT daha fazla öznitelik bulduğu için çerçeve başına düşen algılanan öznitelik sayısı daha yüksek olmakta, bu durum ise işlem maliyetini artırıp özniteliklerin daha az bulunabileceği bir bölgeye dönüldüğünde daha çabuk kopmaya (odometri kaybının yaşanmasına) sebebiyet vermektedir. Tez kapsamında, SIFT ile haritalama yapıldığında, SURF ile başarılı olunan bazı durumlarda, kopma gerçekleşmiştir. SIFT yaklaşımının daha başarılı bir algoritma olması ise daha iyi sonuç verdiği durum ise haritalanan alanda öznitelik bulunabilecek çok az materyal olduğunda oluşmaktadır.

Tablo 3.2 ile detaylandırılan NNS için ise ilgili makalede [36] sürekli kd-tree'den bahsedilmesi dolayısıyla sistemin test edilmesinden önce en başarılı sonucun o olacağı beklenmekteydi. Ancak, tez kapsamında test edilen durumlar için en başarısız

Tablo 3.1 VWT parametrelerdeki değişimlerin RTAB-Map sistemine etkisi

Visual Word Type	Özellik	Tercih
SURF	Tespit hızı	Karmaşık / Çok nesneli alan
SIFT	Öznitelik sayısı	Tek renk / Az nesneli alan

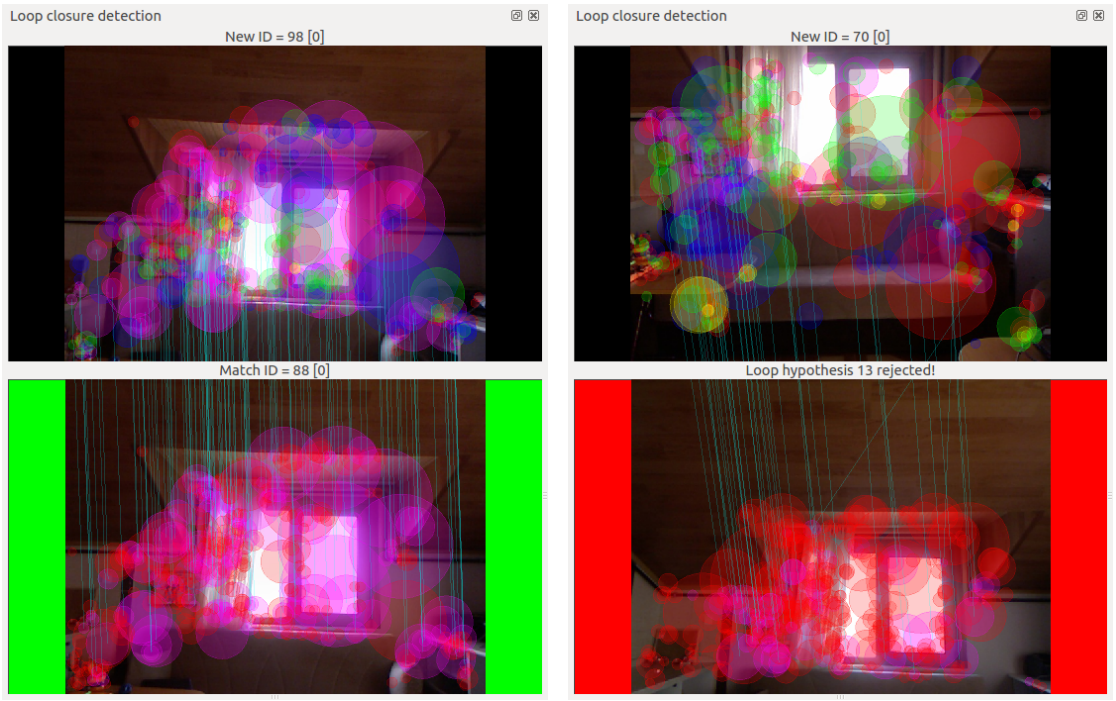
sonuçlar kd-tree kullanımında elde edilmiştir. "Brute Force" yaklaşımı dahi "FLANN Linear"e göre daha yavaş olmasına rağmen doğruluk olarak yakın bir sonuç vermiştir. Burada kullanılan strateji ile öznitelik arasındaki uyum önem arz etmektedir. Tez kapsamında yapılan testler büyük ölçüde SIFT ve SURF ile yapılmıştır ve parametre dokümantasyonunda özellikle bu yöntemlerle "Brute Force"un uyuşmadığı vurgulanmaktadır. Tablo 3.2 ile de gösterildiği gibi "Brute Force" yaklaşımının ikili (binary) öznitelik tanımlayıcılarla kullanılmasını önerilmektedir. "Brute Force" çalışma mantığı düşünüldüğünde ise daha yavaş olmasının beklenildiği söylenebilir. Kesirli tanımlayıcı için ise "FLANN Linear" ya da "FLANN kd-tree" tavsiye edilmektedir. SIFT ile çıkarılan özniteliklerin tanımlayıcı boyutu daha büyük olduğundan kd-tree SURF ile kullanımına göre daha başarısız olmuştur. Kd-tree bazı noktalarda diğer stratejilerin kapalı-döngü buldukları yerlerde dahi tespitte başarısız olmuştur. Bu nedenle sistem tarafından varsayılan olarak "FLANN Linear" belirlenmiş olduğu düşünülmektedir.

Tablo 3.2 NNS parametrelerdeki değişimlerin RTAB-Map sistemine etkisi

Nearest Neighbor Strategy	Avantaj/Dezavantaj	Tercih
FLANN Linear	Hızlı, Başarılı	Büyük öznitelik tanımlayıcı
FLANN KdTree	Yavaş, Başarısız	Küçük öznitelik tanımlayıcı
Brute Force	Yavaş, Başarılı	İkili öznitelik tanımlayıcı

RGBD-SLAM parametrelerinden LCC için “Vis” seçildiğinde kapalı-döngü tespiti için sadece alınan görüntü üzerinden çıkarılan konum bilgileri kullanılmaktadır. Varsayılan “Vis” parametreleriyle (kabul edilebilir maksimum uzaklık ve açı farkı) iki konum kapalı-döngü ile eşleştirilmesi için Şekil 3.1a ile verildiği gibi büyük oranda benzemesi gerekmektedir. Şekil 3.1b ile de gösterildiği gibi iki resim arasında ortak görüntünün bulunması tespit edilen kapalı-döngünün kabul edilmesi için yeterli olmamaktadır. Bu nedenle varsayılan olan “Vis” seçildiğinde, kapalı-döngü tespiti için gerekli benzerlik oranının belirlenmesi de önem arz etmektedir.

Odometry parametreleri ise dışarıdan odometri bilgisi verilmediğinden sistemin sağladığı görsel odometri kullanımını ayarlamaktadır. Genel haritalamanın dışında burada çerçeveler arasında ilişki kurularak kameranın ne kadar hareket ettiğini tahmin etmektedir. Bu tahminde başarının yanı sıra haritalamaya göre hızın önemi daha büyüktür. Çünkü bu tahmin sonrası haritalama için işlem yapılabilmektedir.



(a) **(b)**

Şekil 3.1 Resimlerden oluşturulan konumların uzaklıkları varsayılan değer olan 0.02 metreden (a) düşük olduğu için kapalı-döngü tespiti (b) uzak olduğu için kapalı-döngü reddi

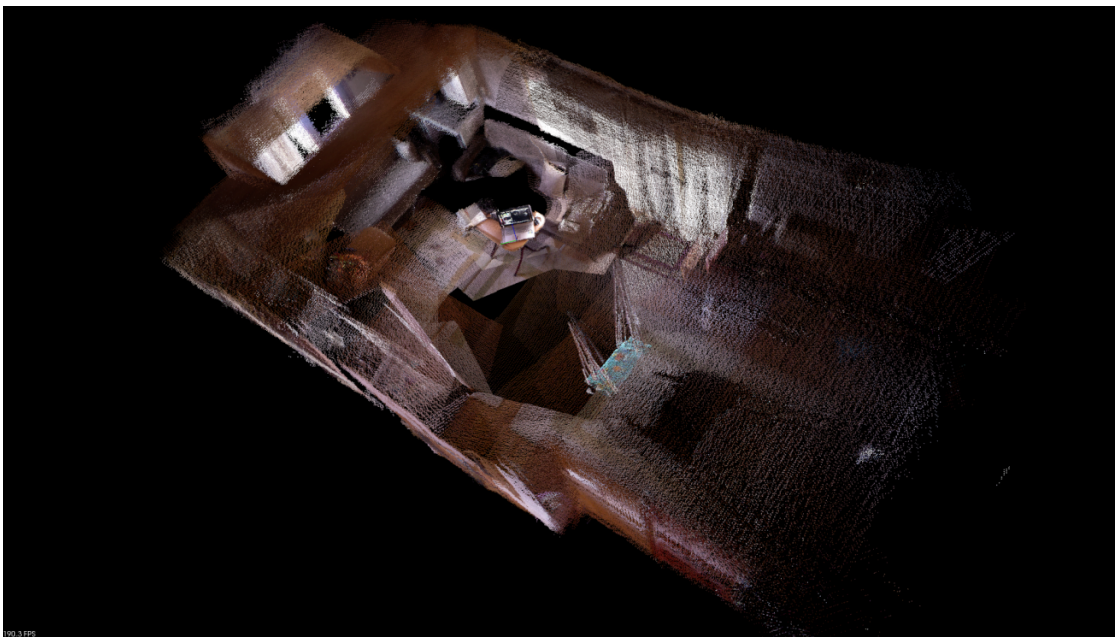
Bu nedenle hızı artırmak amacıyla odometri hesabında kesirli tanımlayıcı kullanımı mümkün olsa da ikili tanımlayıcı tercih edilmesi önerilmektedir. Yapılan testler sonucu, en yüksek başarıyı veren ikili tanımlayıcı ise varsayılan olarak belirlenen GFTT+BRIEF'dir. Bununla birlikte programın dokümantasyonunda FLANN KdTree modeli kullanıldığında SURF ya da SIFT gibi kesirli tanımlayıcıların, Brute Force kullanıldığında ise ikili tanımlayıcı olan ORB/FREAK/BRIEF/BRISK yöntemlerinden birinin kullanılması önerilmektedir.

3.2 Farklı Ortam Şartlarında 3 Boyutlu Haritalama

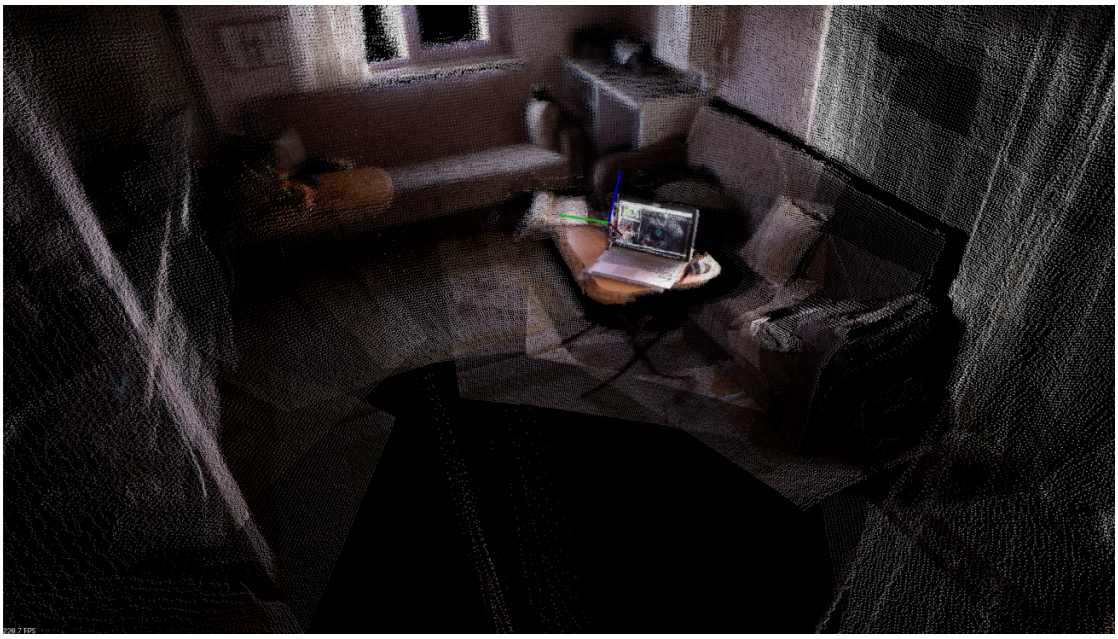
RTAB-Map performansında parametrelerin etkisini ortamdan bağımsız bir şekilde anlatmak mümkün olmamaktadır. Mesela, daha önce de belirtildiği gibi, SIFT ya da SURF kullanımının tercihi haritalamak istenen ortamın özelliklerine bağlıdır. Bu nedenle, sistem 4 farklı ortamda test edilmiştir.

1. 3 boyutlu haritalama ilk olarak, içerisinde çok sayıda eşyanın bulunduğu ve RGB görüntü üzerinden sistemin rahat bir şekilde ayırt edici öznitelikler tespit edebileceği bir ortamda test sonuçları verilmektedir. Yukarıda verilen farklı parametrelerin kullanımında haritalama başarı ve hızında değişiklikler

gözlemlense de Şekil 3.2 ile de görülebileceği gibi bu ortamda sistemin başarılı olduğu söylenebilmektedir.



(a)

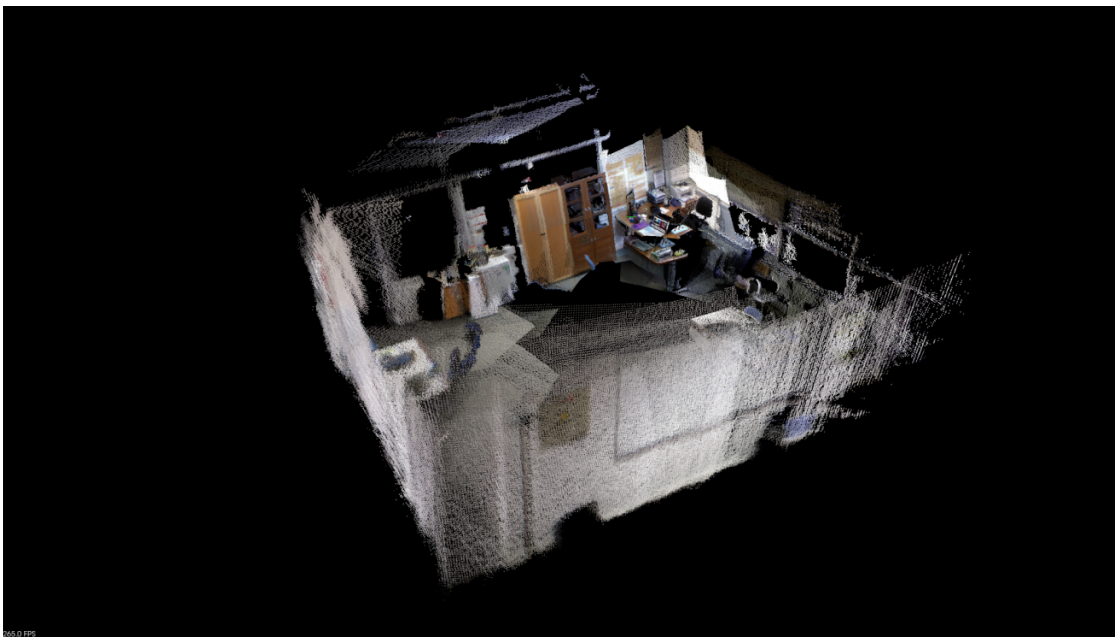


(b)

Şekil 3.2 RTAB-Map ile ilk ortamın 3 boyutlu haritalama sonucunun farklı açılardan gösterimi

2. İkinci olarak 3 boyutlu haritalamanın test edildiği ortamda duvarlar beyaz renk olup içerisinde az sayıda eşya içermektedir. Bu durumda görsel odometrinin kopmaması ve haritanın daha düzgün çıkması için haritalama sırasında Kinect'in ilk duruma göre daha yavaş hareket ettirilmesi gerekmektedir. Buna rağmen, ilk durumdaki harita başarısı elde edilememektedir. Şekil 3.3 ile de görüldüğü gibi

odanın etafındaki tam turun sonunda sistem kapalı-döngü tespitinde başarısız olduğu için haritayı ICP ile düzeltme işlemini gerçekleştirmemektedir.



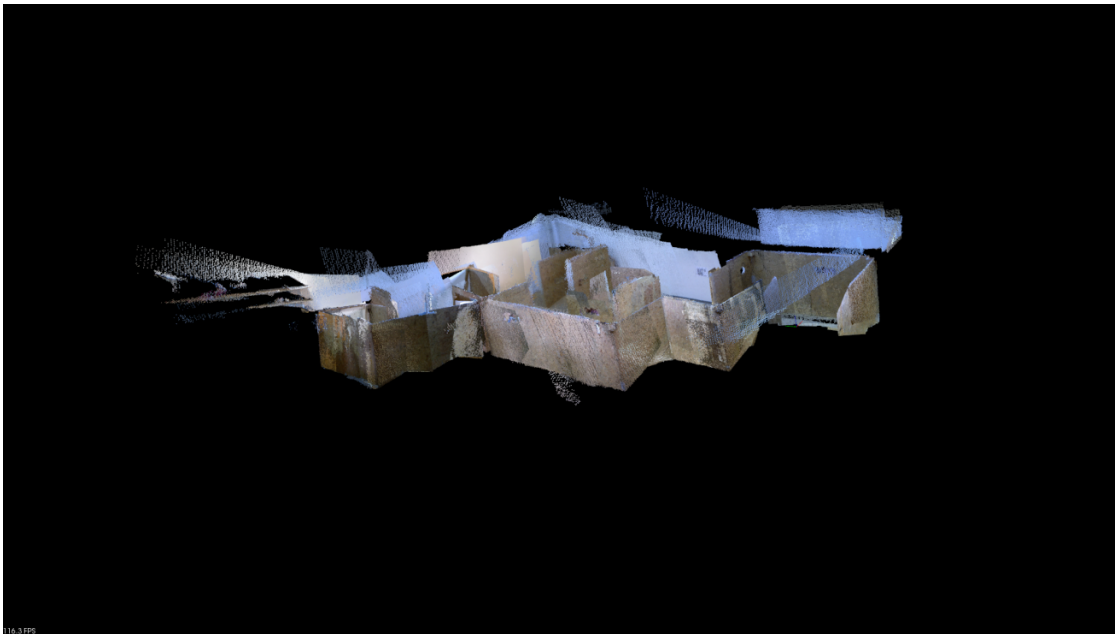
(a)

(b)

Şekil 3.3 RTAB-Map ile ikinci ortamın 3 boyutlu haritalama sonucunun farklı açılardan gösterimi

3. Haritalamanın test edildiği üçüncü ortam olan labirent ise Robocup [58] yarışmalarında afet bölgelerini temsil eden labirentlerin standartlarına uygun olarak sunta duvarlar ile oluşturulan yapay bir ortamdır. Şekil 3.4 ile görüldüğü gibi sunta duvarların kendinden desenli oluşu RGB veri üzerinden öznitelik çıkarımını kolaylaştırdığı için başarılı bir harita elde edilmektedir.

(a)

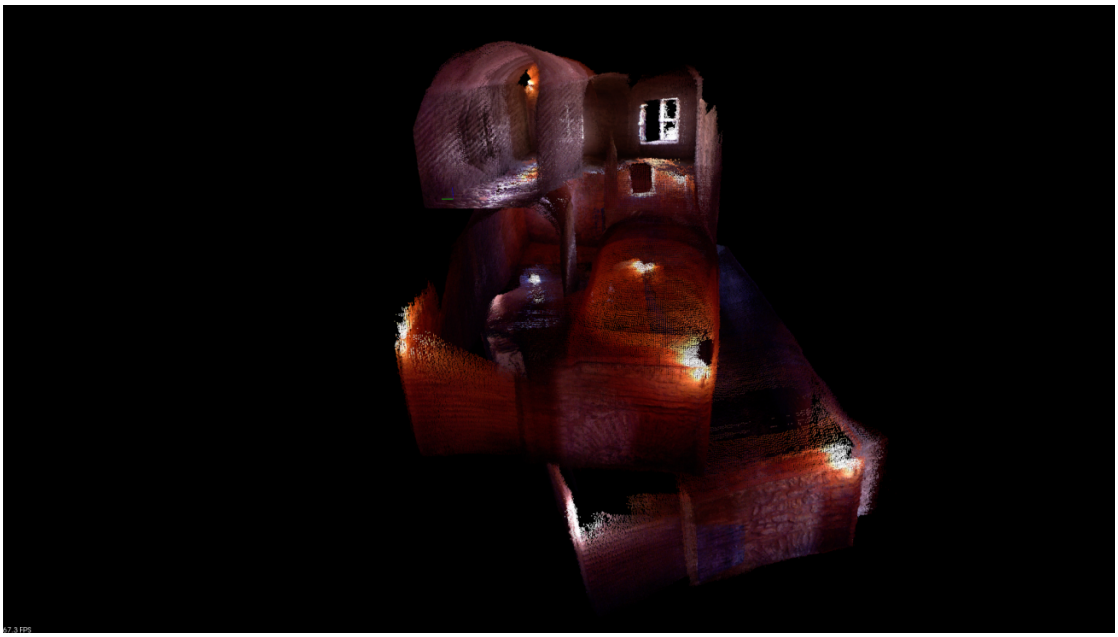


(b)

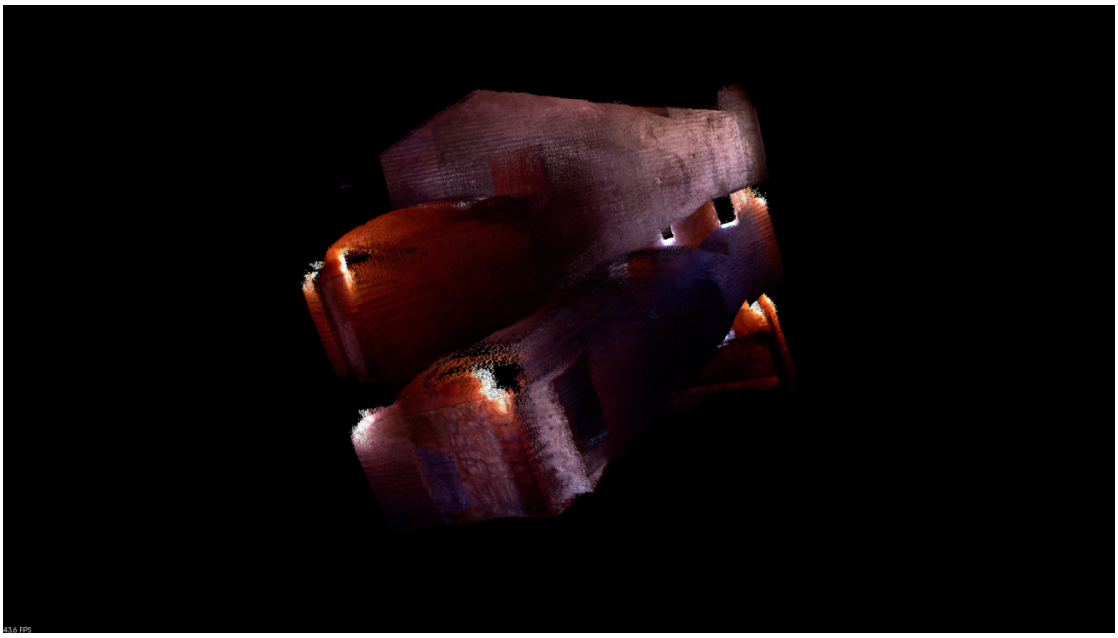
Şekil 3.4 RTAB-Map ile üçüncü ortamın 3 boyutlu haritalama sonucunun farklı açılardan gösterimi

4. Haritalamanın test edildiği son ortam olan Ayasofya'da üç kat rampadan oluşan kapalı bir alanın haritalanmasının diğerlerine göre daha zorlu olmasının temel nedeni ortamın ışıklandırmasındaki problemdir. Bu problem ile haritalanacak herhangi bir ortamda da karşılaşılma ihtimali göz ardı edilemeyeceğinden bu durumun diğerlerine göre gerçeğe daha yakın şartlar sağladığı söylenebilmektedir. Şekil 3.5 ile verilen haritada, alan yukarıdan aşağıya doğru haritalanmışken Şekil 3.6 ile aşağıdan yukarı çıkılarak yapılan

haritalama verilmektedir. Aşağıdan yukarıya çıkışta görüntülerin daha koyu olması nedeniyle daha kötü haritalar elde edilmektedir.

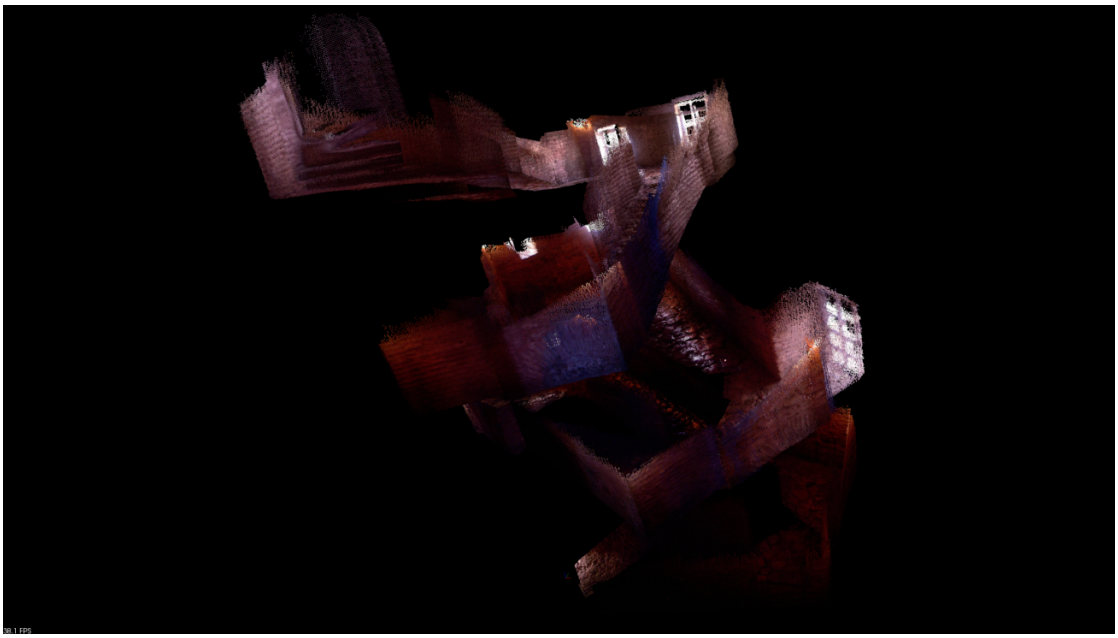


(a)



(b)

Şekil 3.5 RTAB-Map ile Ayasofya rampaların yukarıdan aşağıya 3 boyutlu haritalama sonucunun farklı açılardan gösterimi



(a)

(b)

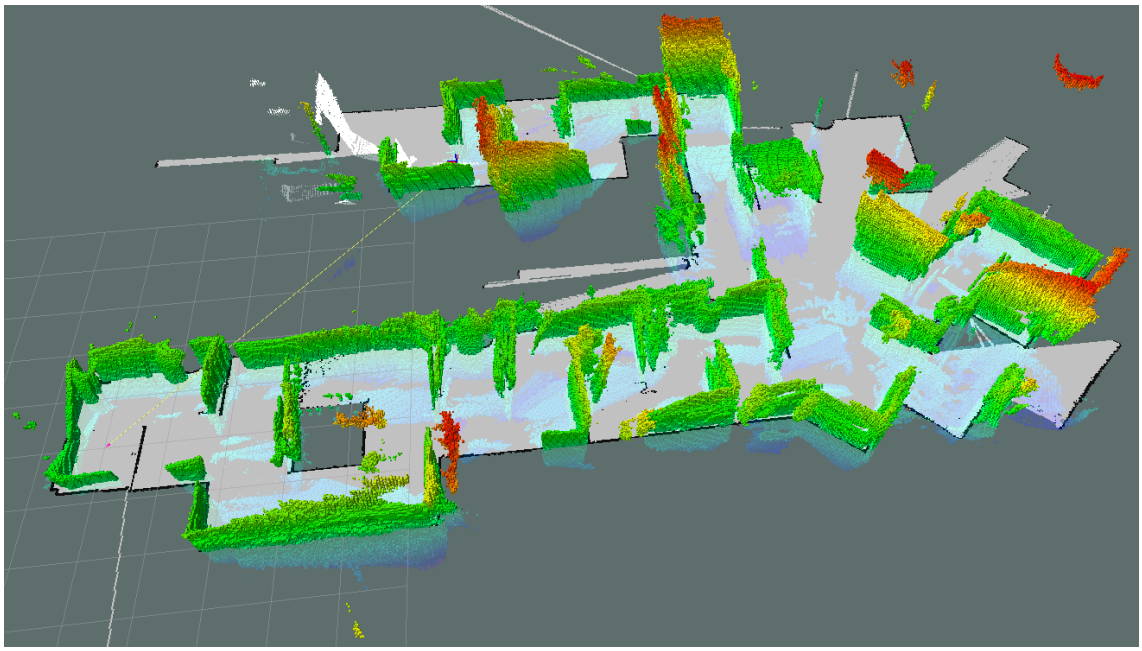
Şekil 3.6 RTAB-Map ile Ayasofya rampaların aşağıdan yukarıya 3 boyutlu haritalama sonucunun farklı açılardan gösterimi

3.3 Farklı Odometri Yöntemleri ile 3 Boyutlu Haritalama

Dört farklı durumun karşılaştırıldığı bu aşamada test alanı olarak 2015 German Robocup Rescue alanı kullanılmıştır. Bu bölümde gösterilen 3 boyutlu haritalama sırasında oluşturulan duvarların 2 boyutlu haritalamada olduğu gibi her yerde devamlılık göstermemesinin sebebi, Kinect sensörünün lazer sensörü kadar geniş bir bakış açısına sahip olmaması, bu nedenle gezinim sırasında Kinect'in tüm duvarları görememesidir.

Test edilen ilk iki durumda, Hector [8] kullanılarak 2 boyutlu odometri üretilip sırasıyla Octo-Map [42] ve RTAB-Map [3] algoritmalarına giriş olarak verilmiştir. Verinin toplandığı alanda rampaların bulunması dolayısıyla, robot gezinim sırasında farklı yüksekliklerde veri okuması yapmıştır. Ancak odometri bilgisi 2 boyutlu uzayda üretildiği için robotun her zaman zemin seviyesinde olduğu varsayılmış, başka bir ifadeyle Z ekseni her zaman sıfır olarak kabul edilmiştir; bu sebeple elde edilen 2 boyutlu odometrinin 3 boyutlu haritalamada kullanılması yüksek hataya sebep olmuştur.

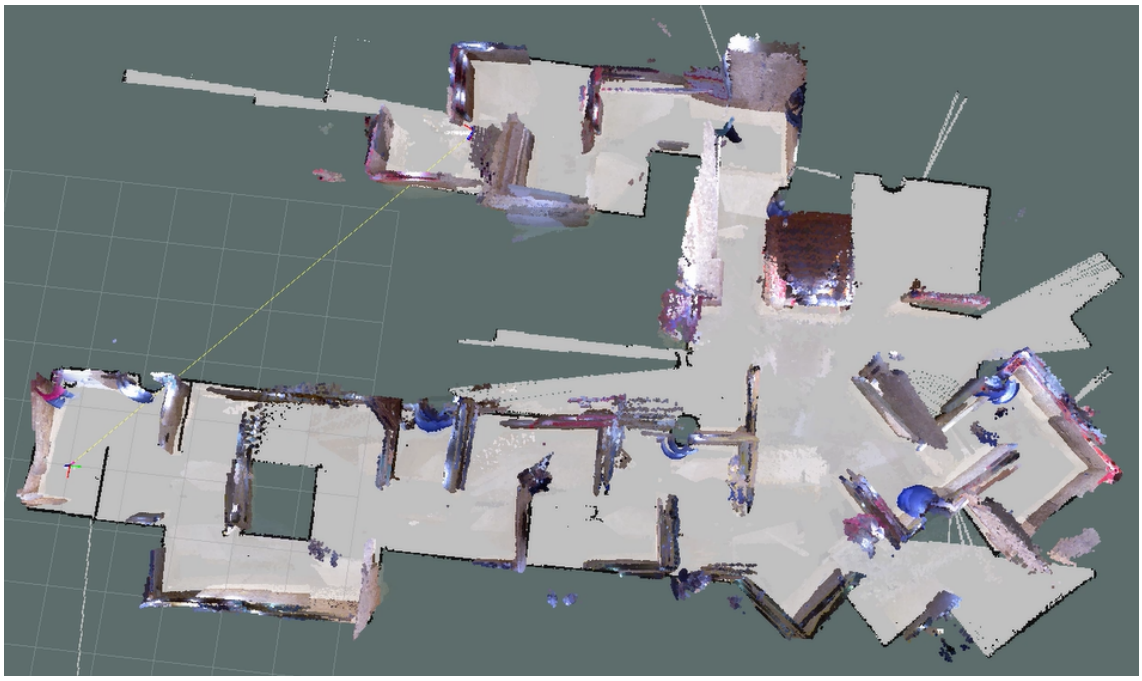
Şekil 3.7 ile Hector tarafından üretilen odometri bilgisinin Octo-Map tarafından kullanılmasıyla elde edilen harita gösterilmektedir. Octo-Map alınan odometri verisinin ve Kinect derinlik bilgisinin hatasız olduğunu varsaymaktadır. Bu nedenle gelen veriyi haritaya ekleme yaparken düzeltme işlemi yapılmamakta, bu durum ise haritada oluşturulan duvarların her yeni veri eklenmesi sonrası kalınlaşma olasılığı artırmaktadır.



Şekil 3.7 2015 Robocup Rescue German Open yarışma alanında Hector'un 2 boyutlu odometrisini kullanarak Octo-Map'in ürettiği 3 boyutlu harita

Şekil 3.8 ile RTAB-Map tarafından Hector'un oluşturduğu 2 boyutlu odometri kullanılarak üretilen harita gösterilmektedir. Odometrinin 2 boyutlu olmasından kaynaklı Z ekseni problemi burada da gözüküyor olsa da RTAB-Map haritalama sırasında veri eklemesi yaparken görsel bir düzeltme de yaptığı için duvarların kalınlaşma problemi bu durumda söz konusu olmamaktadır.

Test edilen sonraki iki durumda ise odometri kaynağı olarak RTAB-Map paketinde bulunan görsel odometri kullanılmaktadır. Burada üretilen 3 boyutlu odometri

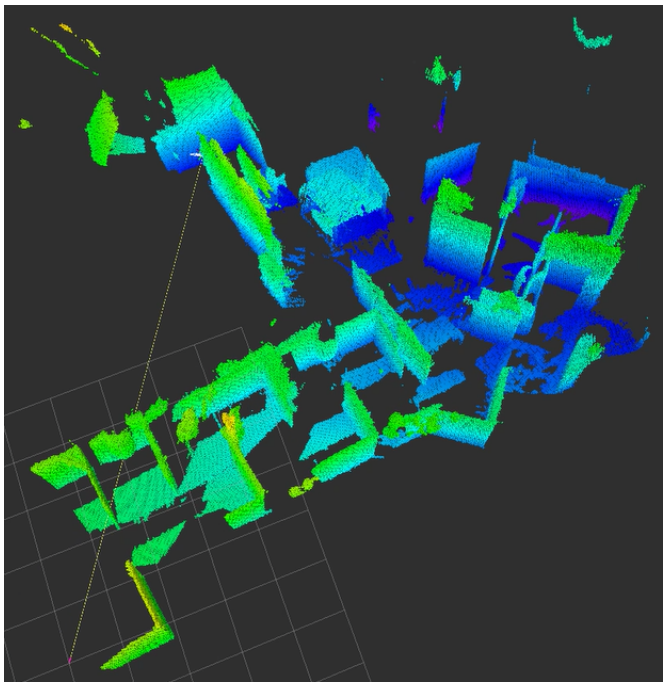


Şekil 3.8 2015 Robocup Rescue German Open yarışma alanında Hector'un 2 boyutlu odometrisini kullanarak RTAB-Map'in ürettiği 3 boyutlu harita

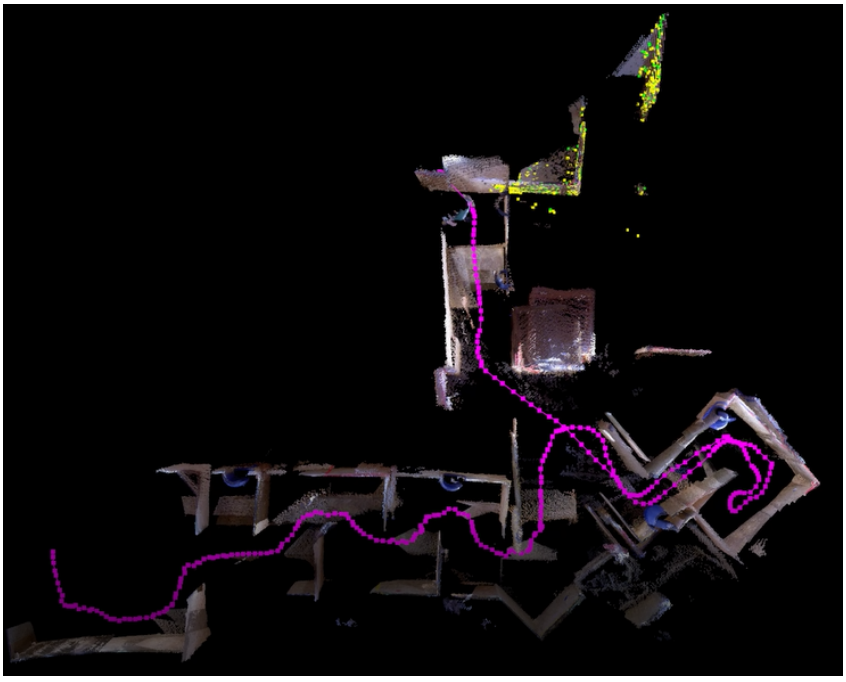
verisinin rampaların oluşturduğu yükseklik bilgisini de içermesi avantaj sağlasa da görsel odometri takibinin kopma olasılığının daha yüksek olması dezavantaj oluşturmaktadır. Mesela test edilen yarışma alanında hector tarafından üretilen odometride hiç kopma yaşanmazken görsel odometride belirli noktalarda kopmalar gerçekleşmektedir. Odometride kopma gerçekleştiğinde haritalamanın sıfırlanması gereksinimi doğduğundan, görsel odometri kullanılarak alanın tamamının haritalanması mümkün olmamıştır. Ancak odometrinin kopmadığı kısımlarda yüksek başarı ile haritalama yapılması sağlanmıştır.

Test alan içerisindeki veriler üzerinde görsel odometrinin kopmadan hesaplanabildiği en büyük alana ait olan ve Octo-Map tarafından üretilen harita Şekil 3.9 ile gösterilmektedir. Kullanılan odometrinin 3 boyutlu olması dolayısıyla daha önceki durumda karşılaşılan kayma problemlerinin çoğuna çözüm olduğu için alanın işlenen kısmında daha başarılı bir harita elde edilmesi mümkün olmuştur.

Son olarak Şekil 3.10 ile de görüldüğü üzere hem odometri hesabında hem de 3 boyutlu haritalamada RTAB-Map paketi kullanılmıştır. Görsel odometride oluşan kopmalar doğal olarak bu durumda da gözlemlenmiş ancak onun dışında hem 3 boyutlu odometri oluşturulabilmesi hem de haritaya veri ekleme sırasında algoritmanın kullandığı düzeltme işlemi dolayısıyla alanın işlenen en büyük bölgesi için elde edilen en başarılı harita bu durumda oluşturulmuştur.



Şekil 3.9 2015 Robocup Rescue German Open yarışma alanında RTAB-Map'in görsel odometrisini kullanarak Octo-Map'in ürettiği 3 boyutlu harita

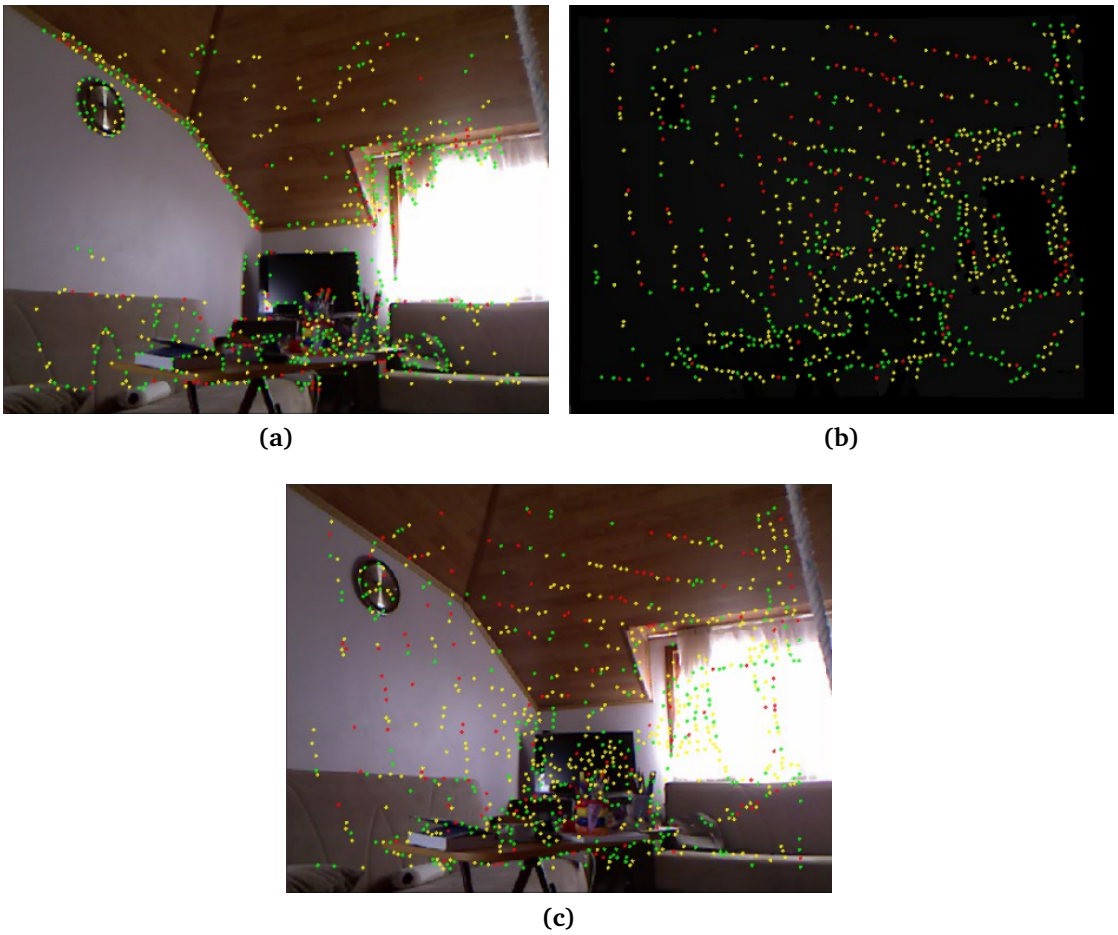


Şekil 3.10 2015 Robocup Rescue German Open yarışma alanında RTAB-Map'in ürettiği 3 boyutlu harita

3.4 Farklı Giriş Verileri ile 3 Boyutlu Haritalama

Tez kapsamında yapılan çalışmalarda, RTAB-Map yönteminin karanlıkta başarısız olduğu gözlemlenmiştir. Bununla birlikte, Bölüm 2.3.2 ile Kinect'ten alınan derinlik bilgisinin ortam aydınlatmasından bağımsız olduğu anlatılmıştır. Bu bilgiler ışığında,

çalışmalar RTAB-Map'e giriş olarak RGB resim yerine derinlik resmi verilip test edilmesi yönünde ilerlemiştir. Şekil 3.11 ile görüldüğü gibi RGB ve derinlik resimleri üzerinde bulunan öznitelikler farklılık gösterse de RTAB-Map aslında derinlik resmi üzerinde de yeterince öznitelik çıkarımı yapmayı başarmaktadır. Ancak derinlik resmi üzerinde bulunan nesnelerin renginin, kameranın uzaklığına göre değişiklik göstermesi dolayısıyla; derinlik resmi üzerinden bulunan öznitelik noktalarına sabit renk bilgisi eklenememesi haritalamanın başarısız olmasına yol açmaktadır.



(a) **(b)**

(c)

Şekil 3.11 RTAB-Map'e giriş olarak verilen (a) RGB resim üzerinde bulunan öznitelik noktaları (b) derinlik resmi üzerinde bulunan öznitelik noktaları (c) derinlik resmi üzerinde bulunan öznitelik noktalarının RGB resim üzerinde gösterimi

RTAB-Map sistemi RGB resmi üzerinden öznitelik çıkarımını iki farklı modülde kullanmaktadır. Bunların ilki görsel odometri hesabını yapan modül diğeri ise haritalama modülüdür. Bu iki modüle giriş verisi ayrı ayrı girilebilmektedir. Görsel odometriye RGB verisi verilirken haritalamaya giriş olarak derinlik resminin verildiği durumda hareket takibi başarıyla gerçekleştirilebilirken haritalamada başarısız olunmuştur. Sistem aldığı verileri haritaya ekleyememiştir. Tam tersi olarak görsel odometriye derinlik resmi verilip haritalama modülüne RGB resim verildiğinde ise odometri takibinde kopmalar meydana gelmektedir. Kopma olmadan sensor hareketi

sağlanabildiği durumda ise başarılı bir harita oluşturulabilmektedir.

3.4.1 3 Boyutlu Termal Haritalama

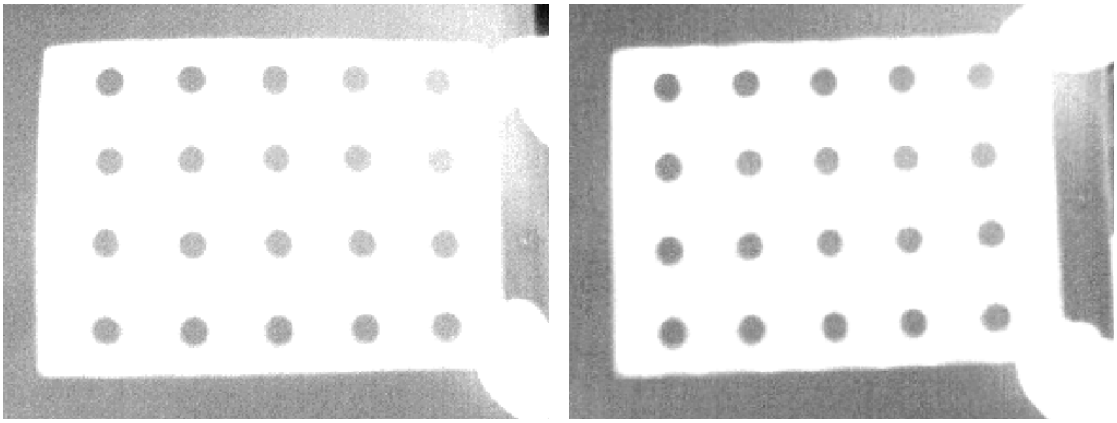
Tez kapsamında, 3B SLAM sistemlerine RGB-D sensörler dışında sensörlerden faydalanarak 3 boyutlu haritalama performansının arttırılmasına yönelik yapılan çalışmalarda termal sensör verisinin de kullanılmasına karar verilmiştir [61]. Termal görüntünün 3 boyutlu olarak modellenmesi ise haritalanan ortamda bulunan ısı kaynaklarından elde edilen veriyi daha kullanışlı hale getirmektedir. Termal bilginin derinlik bilgisi üzerine oturtularak anlık sahneler için termal 3 boyutlu görüntü oluşturan sistemler mevcuttur [62]. Tez kapsamında ise benzer bir yaklaşımdan hareketle, 3 boyutlu termal görüntüler, sahneler arasında kullanılarak bir ortama ilişkin 3 boyutlu termal harita elde edilmiştir.

RGBD-SLAM yöntemi termal 3B harita çıkarılması için kullanılmıştır. Termal görüntü ile birlikte derinlik bilgisi verebilen kameralar olmadığı için derinlik bilgisi okuyan kameraların RGB görüntü alan sensörü yerine termal bir kamera kullanılmıştır. Başka bir ifadeyle, çalışma kapsamında gerçekleştirilen deneylerde gri seviyede görüntü ile oluşturulmuş TermalD-SLAM haritaları elde edilmektedir. RBGD-SLAM yöntemi için ROS platformu üzerinde gerçeklenmiş "rgbdslam" paketi kullanılmıştır [29].

RGB-D sensör olarak Bölüm 2.3 ile anlatılan Kinect kullanılmıştır. Termal kamera olarak ise Şekil 3.12 ile verilen Optris PI400 kamerası kullanılmıştır. Bu kamera, -20°C ila 100°C arasında sıcaklık bilgisi okuyabilmektedir. Çözünürlüğü 382×288 olan kamera, saniyede 80 çerçeve oranında görüntü vermektedir. Sıcaklık değerlerini farklı renklerle temsil edebilen kamera, deneyler kapsamında sadece gri seviye sıcaklık değeri üretmesi için ayarlanmıştır.

Şekil 3.12 Optris PI400 termal kamera

Termal kamera iç kalibrasyonu için, öncelikle, bir karton üzerinde kare ızgara şeklindeki yapının köşelerine denk gelen alanlar dairesel olarak kesilmiştir. Sonrasında ise karton uygun şekilde ısıtılmış ve sonra otomatik olarak içsel parametreler elde edilmiştir. Elde edilen parametreler çerçevesinde, termal kamera için deformasyonlar giderilmeden önce ve giderildikten sonraki halleri Şekil 3.13 ile verilmektedir.



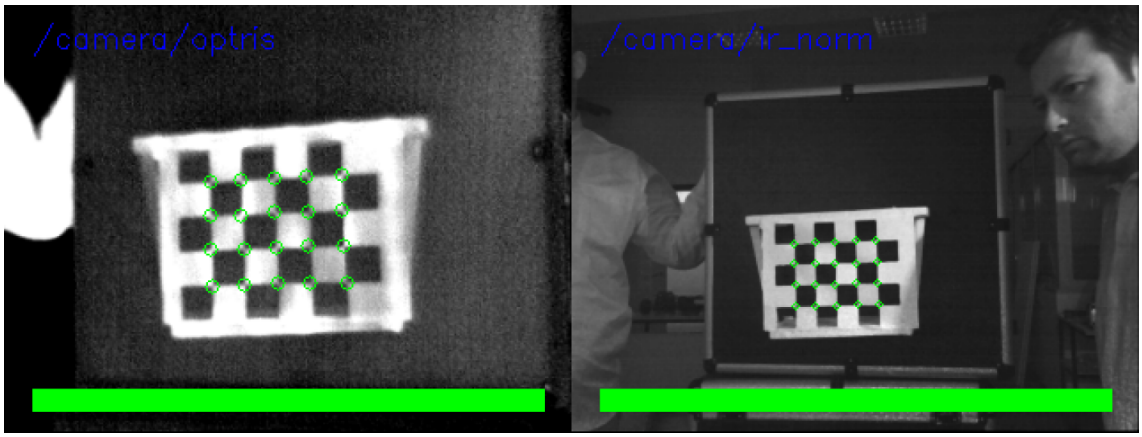
(a) (b)

Şekil 3.13 Termal kamera için deformasyonlar (a) giderilmeden önce ve (b) giderildikten sonraki görüntüler

Termal kamera ile Kinect arasında sabit bir düzenek oluşturularak termal derinlik bilgisi elde edilmeye çalışılmışıtır. Bunun ile ilgili olarak oluşturulan yapı Şekil 3.14 ile verilmiştir.

Şekil 3.14 Termal kamera ve Kinect için bağlantı düzeneği

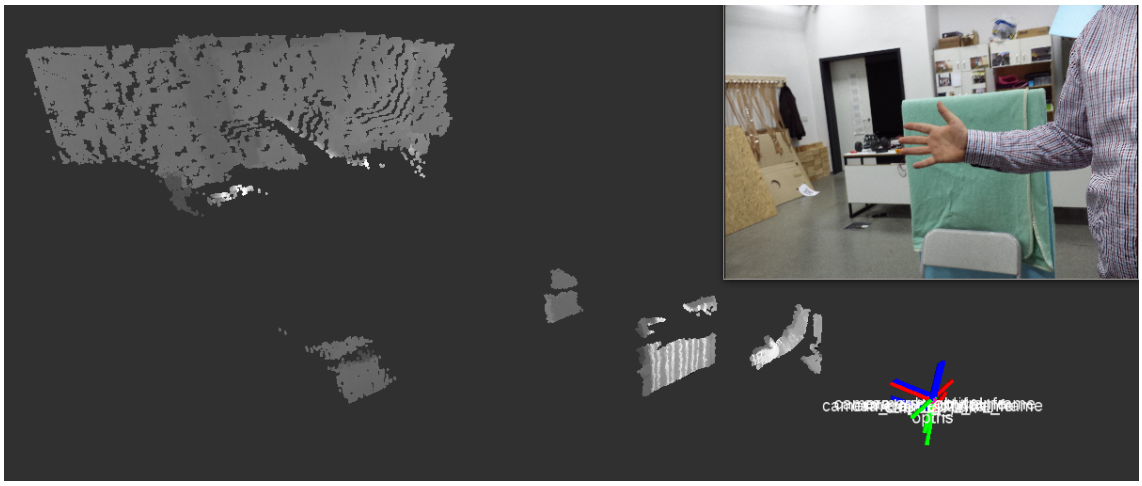
Termal kamerada kağıt üzerinde dama tahtası deseni görülebilir olmadığından iki kameranın birbirine göre konumları belirlenirken kızılötesi ve termal kameralarının ikisininde görebileceği bir düzenek kurulmuştur. Buna göre siyah kareleri kesilmiş bir dama tahtası deseni uygun şekilde ısıtıldıktan sonra, desen kağıdının 5cm arkasına siyah arkaplan oluşturulmuştur. Bu sayede yapılan kalibrasyon ile iki kameranın birbirlerine göre öteleme ve bakış açıları elde edilmiştir. Şekil 3.15 ile kızılötesi ve termal kamera arası dış kalibrasyon işlemi sırasında elde edilen görüntüye yer verilmiştir.



Şekil 3.15 Termal kamera (solda) ve kızılötesi (sağda) arası dış kalibrasyon işlemi

Dış kalibrasyon ile kameraların konumları arasındaki ilişkinin belirlenmesi, kameralardan alınan verilerin birbirlerine göre yorumlanabilmesine olanak sağlamaktadır. Diğer bir ifadeyle Kinect'ten alınan derinlik bilgisi üzerine termal kameradan alınan görüntü oturtulabilmektedir.

Şekil 3.16, sağ üst köşede RGB görüntüsü verilen ortamın, derinlik ve sıcaklık verilerinin örtüştürülmesi sonucu gösterilmektedir. Sıcaklık derecesi yüksedikçe şekildeki gri tonu beyaza yaklaşırken, soğuk alanların gösterimi koyu gri ile sağlamaktadır.



Şekil 3.16 Derinlik bilgisi üzerine termal resmin oturtulması sonucu elde edilen nokta bulutu

3.4.1.1 3 Boyutlu Termal Haritalama Sonuçları

Daha önce de bahsedildiği gibi termal 3B haritalama çalışmasında ROS "rgbdslam" paketinin özellik çıkarımı için Kinect'ten gelen RGB resim verisi yerine termal kameradan gelen gri seviyedeki görüntüyü kullanması sağlanarak TermalD-SLAM

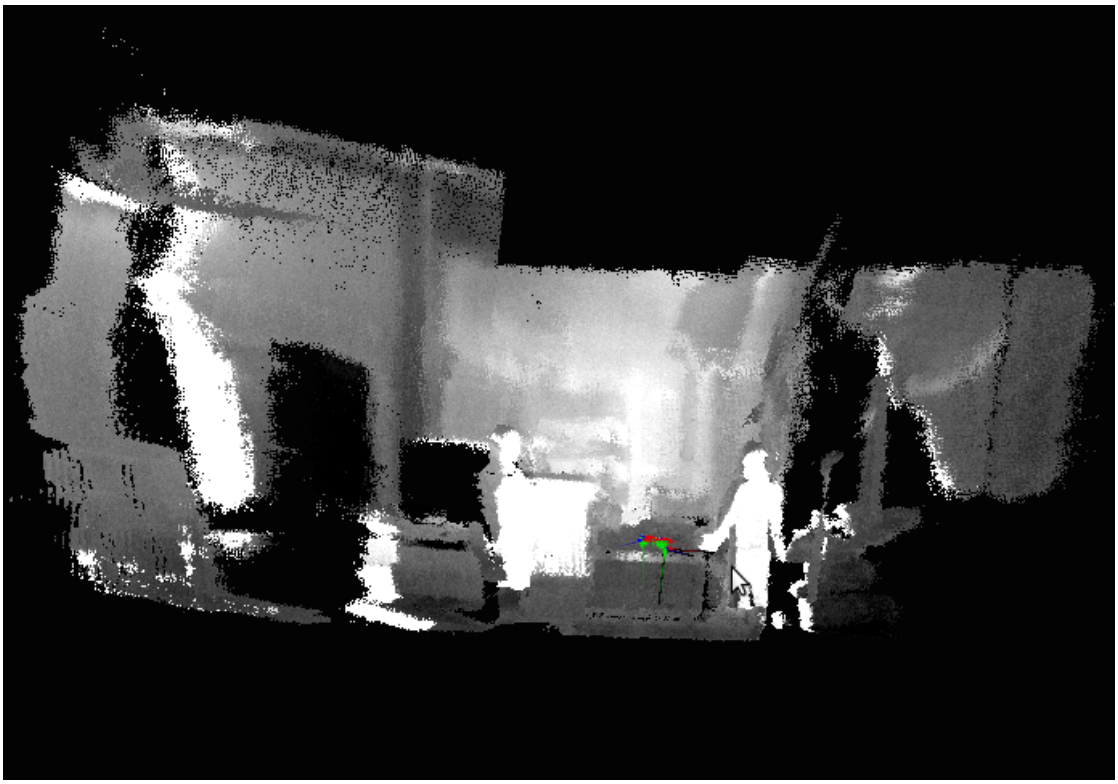
haritaları başarılı bir şekilde üretilmiştir. Termal görüntünün haritalama başarısına etkisini artırmak amacıyla elde edilen resimdeki gri tonların çeşitliliğinin sağlandığı test alanı panaromik bir resim olarak Şekil 3.17 ile gösterilmektedir. RGBD-SLAM için özellik belirleme aşamasında hızlı sonuç veren SURF yöntemi kullanılmıştır.

Şekil 3.17 3 boyutlu termal haritalama uygulama alanı

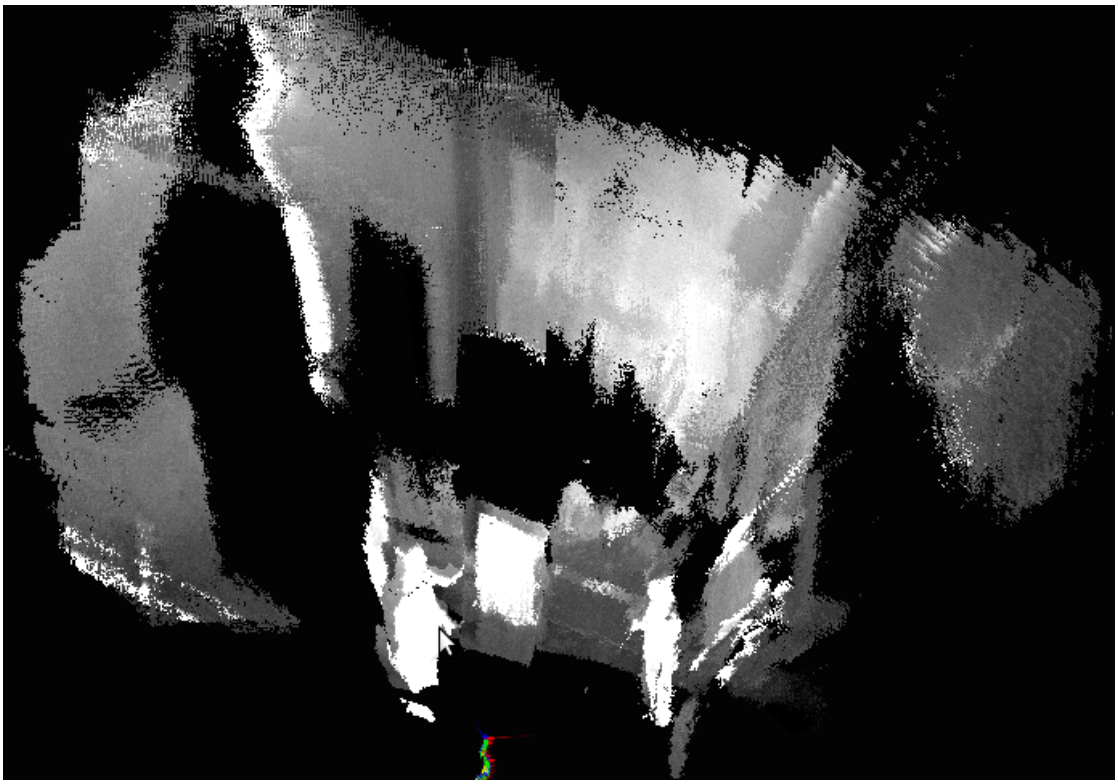
Şekil 3.18, Şekil 3.17 ile gösterilen alanda Kinect ve termal kameradan alınan veriler kullanılarak kümülatif olarak oluşturulan 3 boyutlu haritayı farklı açılarda göstermektedir.

Termal kamera ve Kinect kızılötesi kamera verilerinin birleştirilerek kullanılmasıyla 3 boyutlu termal görüntü elde edilmiştir. Derinlik bilgisi ve termal bilgi kullanılarak RGBD-SLAM algoritması ile 3 boyutlu termal haritalama yapılmıştır. 2 farklı kamera verisinin birbirine entegre edilmesi aşamasında iyileştirmenin sabit bir mesafede (1,5 m) yapılması sonucunda 3 boyutlu termal haritada farklı derinliklerde elde edilen sonuçların bazılarında termal ısı bilgisine sahip piksellerin arkaplana kaçtığı görülmüştür. Bu durum Şekil 3.18a sol arka duvara kaçan termal ısı piksellerinde gözlenebilir.

Çalşmanın termal 3 boyutlu kısmında umut vaadeden bir başarı elde edilmiş olsa da termal kameranın odaklandığı mesafe dışındaki durumlarda net sonuçlar döndürememesi dolayısıyla kullanımının yeterince efektif olmadığı farkedildiğinden bu alanda daha fazla araştırma yapılmamasına karar verilmiştir.



(a)



(b)

Şekil 3.18 3 boyutlu termal haritalama sonuç

3.5 3 Boyutlu Sistemlerin Hesaplama Maliyet Analizi

Sistem performansının önemli kriterlerinden biri de sistemin çalışması için gereken minimum gereksinimlerdir. Bu nedenle çalışma detaylarının incelenmesinin ardından RTAB-Map sistemi üzerinde haritalama sırasında kullandığı CPU ve hafıza miktarının tespit edilmesi amacıyla profiler kullanılmasına karar verilmiştir. Bu amaçla, hesaplama ve hafıza maliyeti açısından problem oluşturan kısımların öncelikle tespiti sonrasında da iyileştirilmesine yönelik çalışma yapılması için ilk olarak Valgrind [63] ve Qt Creator [64] editörüne ait profiler da olmak üzere bir çok profiler kullanımı denenmiştir. Ancak 3B haritalama profiler ile beraber çalıştırıldığında sistemin çökmesi durumu yaşanmış o nedenle başarılı bir sonuç elde edilememiştir. 3B haritalama sistemlerinin profiler ile beraber çalıştırılması sağlanamadığından bu alanda yapılan çalışmalar sonlandırılarak araştırmalar başka alanlara kaydırılmıştır.

4
3 Boyutlu Haritalama Sistemlerinin Karşılaştırılması

Literatürde yapılan çalışmalar sonucu bulunan güncel 3B SLAM sistemlerinin ikili karşılaştırması bu bölümde anlatılmaktadır. İlk olarak, Bölüm 4.1 ile hem tez kapsamında incelenen 3B SLAM sistemlerinin performans analizini hem de önerilen sistem sonrası oluşan performans değişimini incelemek için kullanılan veri koleksiyonu ve özellikleri açıklanmaktadır. Sonrasında, Bölüm 4.2 RTAP-Map ve Octo-Map arasındaki ilişkiyi anlatmakta, son olarak, Bölüm 4.3 ve 4.4 RTAB-Map ile sırasıyla Ethzasl-Icp-Mapper ve Kintinuous karşılaştırmasını yapmaktadır.

4.1 Kullanılan Veri Koleksiyonu

Tez çalışması kapsamında sistem performans ölçüm karşılaştırması için deneylerde kullanılmak üzere kullanılan veri koleksiyonu [65] farklı ortam ve sensör hareketleri için RGB ve derinlik resimlerinin yanısıra senkronize bir şekilde kamera konumu da sağlamakta ve bu verilerin zaman bilgilerini de içermektedir. Veri koleksiyonunda, hem 6 serbestlik derecesiyle elde dolaştırılarak hem de gezinim yapan bir robotun üzerine yerleştirilerek RGB-D sensörden toplanan verileri içeren farklı veri kümeleri mevcuttur.

Resimler Kinect ve benzeri RGB-D sensörler kullanılarak 640×480 çözünürlük ve 30 Hz frekansla sağlanmıştır. Kamera gerçek konum bilgisi ise kameranın gezdirildiği alana yerleştirilmiş yüksek hızla hareket çekim özelliği olan ve 100 Hz ile yayın yapan sekiz farklı kameradan alınan veriler kullanılarak hesaplanmıştır.

Veri koleksiyonu, 8 farklı kategori altında toplam 89 veri kümesi içermektedir. Her bir veri kümesi faklı uzunluğa, farklı konfigürasyona ve farklı zorluk derecesine sahiptir. Tablo 4.1 tez kapsamında kullanılan veri kümeleri ve özelliklerini listelemektedir.

Veri koleksiyonu hataların karelerinin ortalamalarının kökü (RMSE - root-mean-square-error) ve görece pozisyon hatası (RPE - relative pose error) gibi hesaplamaları içeren çeşitli performans karşılaştırma yöntemlerinin kullanılabilmesi

Tablo 4.1 Tez kapsamında kullanılan veri kümelerinin detayları

Veri Kümesi Adı	Süre (s)	Uzunluk (m)	Ort. Çiz. Hızı (m/s)	Ort. Açı. Hızı (°/s)
Kategori: Testing and Debugging				
freiburg1_xyz	30.00	7.11	0.24	8.92
freiburg1_rpy	27.42	1.66	0.06	50.15
Kategori: Handheld SLAM				
freiburg1_desk	23.35	9.26	0.41	23.33
Kategori: Robot SLAM				
freiburg2_pioneer_360	72.00	16.12	0.22	12.05
freiburg2_pioneer_slam	145.86	40.38	0.26	13.38
freiburg2_pioneer_slam2	109.49	21.74	0.19	12.21
freiburg2_pioneer_slam3	105.04	18.14	0.16	12.34
Kategori: Structure vs. Texture				
freiburg3_nostructure_notexture_far	15.80	2.90	0.20	2.71
freiburg3_nostructure_notexture_near_withloop	37.72	11.74	0.32	11.24

için gerçek konum bilgilerini de (ground-truth) sağlamaktadır. Tez aşamasında yapılan çalışmada ise gezinim sırasında, tarih bilgisi baz alınarak, kameranın gerçek pozisyonları ile sistemlerin tahmin ettiği kamera pozisyonları arasındaki Öklid mesafelerinin RMSE değerleri, Denklem 4.1 ile hesaplanıp karşılaştırma yöntemi olarak kullanılmaktadır. Gerçek konumlar, yukarıda belirtildiği gibi, kullanılan veri küme tarafından sağlanmaktadır [65].

$$RMSE = \sqrt{|x-\hat{x}|^2 + |y-\hat{y}|^2 + |z-\hat{z}|^2} \tag{4.1}$$

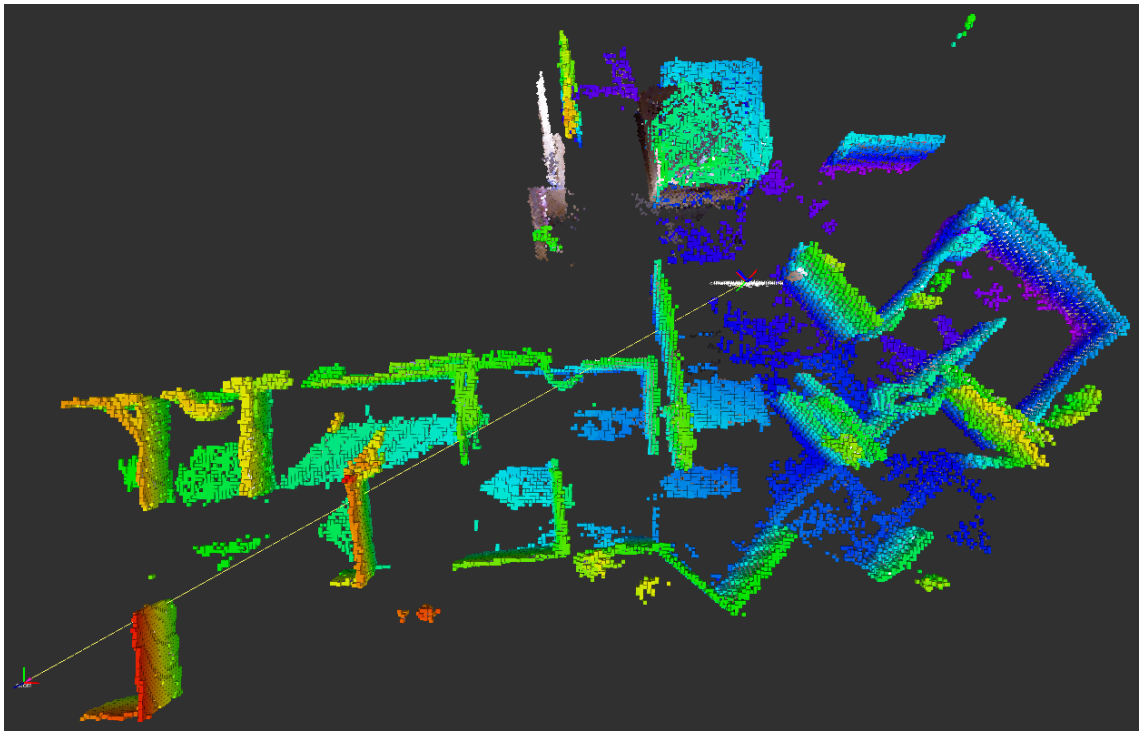
Burada, x, y, z sembolleri kameranın gerçek konumunu ve $\hat{x}, \hat{y}, \hat{z}$ sembolleri ise sistemler tarafından tahmin edilen konumları temsil etmektedir.

4.2 RTAB-Map ve Octo-Map İlişkisi

Yapılan testler ve çalışmalar sonucunda Octo-Map'in haritalamada güçlü bir yöntem kullanmadığı bunun yerine alınan verinin direkt olarak haritaya eklendiği fark edilmiştir. Bunun üzerine yapılan detaylı inceleme sonucu Octo-Map'in kazandırdığı

avantajın 3 boyutlu ve sürekli olan gerçek dünyayı hafızada minimum yer kaplayacak şekilde ızgara tabanlı bir yaklaşıma bölmesi olduğu anlaşılmıştır.

Her ne kadar RTAB-Map tarafından üretilen haritalar, RGB verinin de kullanımının getirdiği görsel bir avantaja sahip olsa da, bu haritalar üzerinde keşif ve gezinim algoritmalarının efektif bir şekilde çalıştırılması mümkün olmamaktadır. Bu nedenle RTAB-Map, elde edilen haritanın keşif ve gezinim algoritmalarının kullanımına daha uygun olan Octo-Map veri tipi olan octree'ye döndürülmesine imkan sağlamaktadır. Şekil 4.1 ile gösterilen Octo-Map haritası, Bölüm 3.3 içerisindeki Şekil 3.10 ile verilen ve 2015 Robocup Rescue German Open yarışma alanında RTAB-Map tarafından oluşturulan 3 boyutlu haritadan elde edilmiştir.



Şekil 4.1 2015 Robocup Rescue German Open yarışma alanında RTAB-Maptarafından oluşturulan 3 boyutlu haritadan elde edilen Octo-Map haritası

4.3 RTAB-Map ve Ethzasl-Icp-Mapper Karşılaştırması

Bu bölümde yapılan karşılaştırmalar sırasında RTAB-Map ve Ethzasl-Icp-Mapper algoritmaları Bölüm 2.3.2 içerisindeki Şekil 2.7 ile verilen ortamda farklı Kinect çözünürlükleri ile, aydınlatmanın iyi olduğu, az olduğu ve hiç olmadığı durumlarda test edilmiştir.

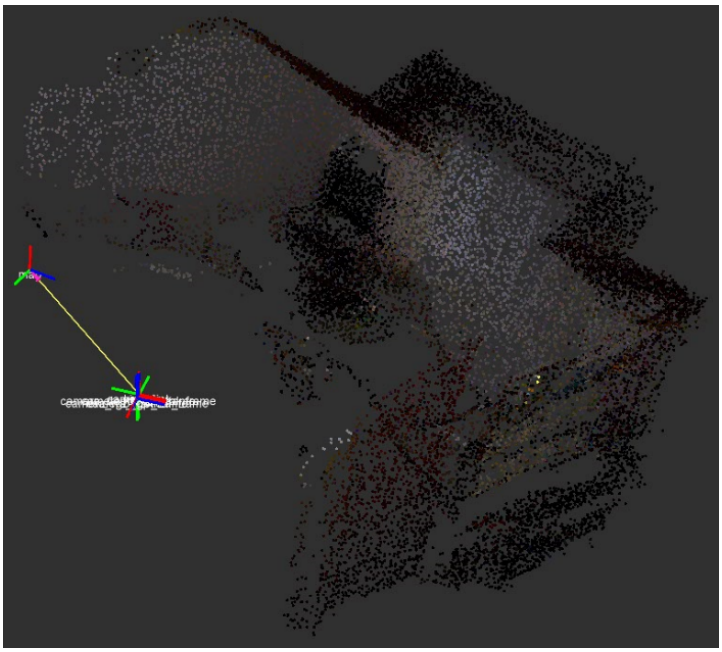
Aydınlık ortamda RTAB-Map kullanıldığında, en iyi başarı yüksek çözünürlüklü sensör verisiyle elde edilirken; sensör çözünürlüğünün düşürülmesi test ortamı için çok

büyük performans kayıplarına yol açmamıştır. Ethzasl-Icp-Mapper ise en iyi başarıyı düşük çözünürlüklü sensör verisiyle verirken haritalama hızı RTAB-Map'e göre yavaş kalmıştır. Hem sensörün daha hızlı hareket edebilmesine olanak sağladığından hem de daha düzgün harita elde edildiğinden RTAB-Map'in aydınlık ortam haritalamasında Ethzasl-Icp-Mapper'a göre çok daha başarılı olduğu söylenebilir. Şekil 4.2 ve 4.3, aydınlık ortamda, sırasıyla RTAB-Map kullanılarak yüksek çözünürlüklü sensör verisi ile elde edilen harita ve Ethzasl-Icp-Mapper kullanılarak düşük çözünürlüklü sensör verileri ile elde edilen haritayı göstermektedir.

Şekil 4.2 RTAB-Map yöntemi kullanılarak aydınlık ortamda yüksek çözünürlüklü sensör verisi ile elde edilen harita

Aydınlatmanın az olduğu ortamda ise RTAB-Map yönteminin başarısı aydınlık ortama göre düşüş gösterdiği halde yine Ethzasl-Icp-Mapper yöntemine göre daha iyi sonuç vermektedir. Burada dikkat edilmesi gereken konu, RTAB-Map RGB verisi kullandığından aydınlatmanın az olduğu ortamda haritalamada başarı elde etmek için sensör hareketinin aydınlık ortama göre daha yavaş olması gerekmesidir. Ethzasl-Icp-Mapper yönteminin başarısında ise kayda değer bir düşme gözlemlenmemiştir. Şekil 4.4 ve 4.5, aydınlatmanın az olduğu ortamda, sırasıyla RTAB-Map kullanılarak yüksek çözünürlüklü sensör verisi ile elde edilen harita ve Ethzasl-Icp-Mapper kullanılarak düşük çözünürlüklü sensör verileri ile elde edilen haritayı göstermektedir.

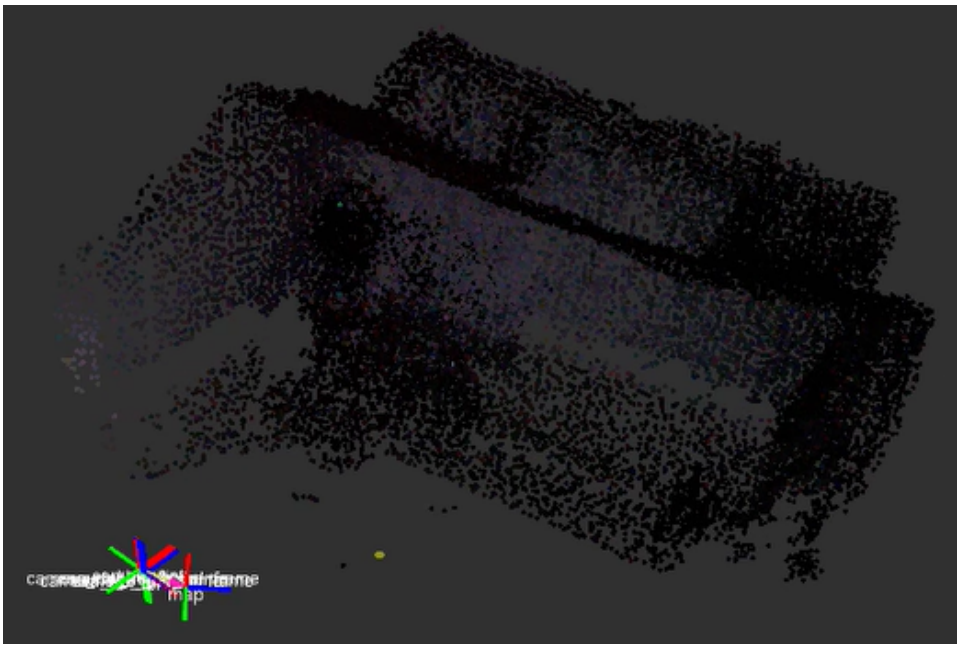
Son olarak RTAB-Map'in karanlık ortamda test edilmesiyle beklenildiği gibi başarısız bir sonuç elde edilerek 3 boyutlu harita oluşturulamamıştır. Bu yöntemin RGB veriye dayanması ve karanlık ortamda alınan RGB verinin kullanışsız olması, dolayısıyla da öznitelik çıkarımı yapılamaması sonucun başarısız olmasının temel sebebidir.



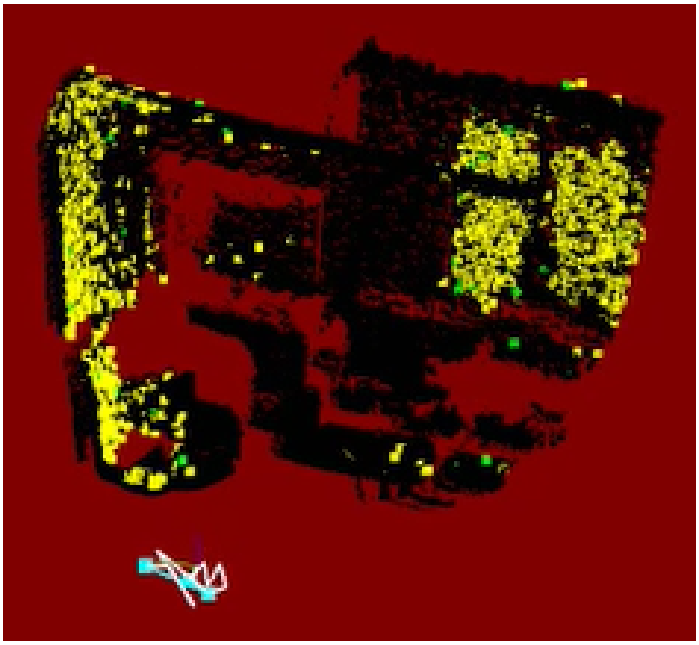
Şekil 4.3 Ethzasl-Icp-Mapper yöntemi kullanılarak aydınlık ortamda düşük çözünürlüklü sensör verisi ile elde edilen harita

Şekil 4.4 RTAB-Map yöntemi kullanılarak aydınlatmanın az olduğu ortamda yüksek çözünürlüklü sensör verisi ile elde edilen harita

Ethzasl-Icp-Mapper yönteminin başarısında ise daha önceki durumlarda da olduğu gibi gözle görülür bir düşüş olmamıştır. Şekil 4.6 ve 4.7, karanlık ortamda, sırasıyla RTAB-Map kullanılarak yüksek çözünürlüklü sensör verisi ile elde edilen harita ve Ethzasl-Icp-Mapper kullanılarak düşük çözünürlüklü sensör verileri ile elde edilen haritayı göstermektedir. Şekil 4.6 ile verilen haritada arka planın kırmızı olmasının sebebi yeterli öznitelik bulunamamasından dolayı odometrinin kaybedilmesidir.

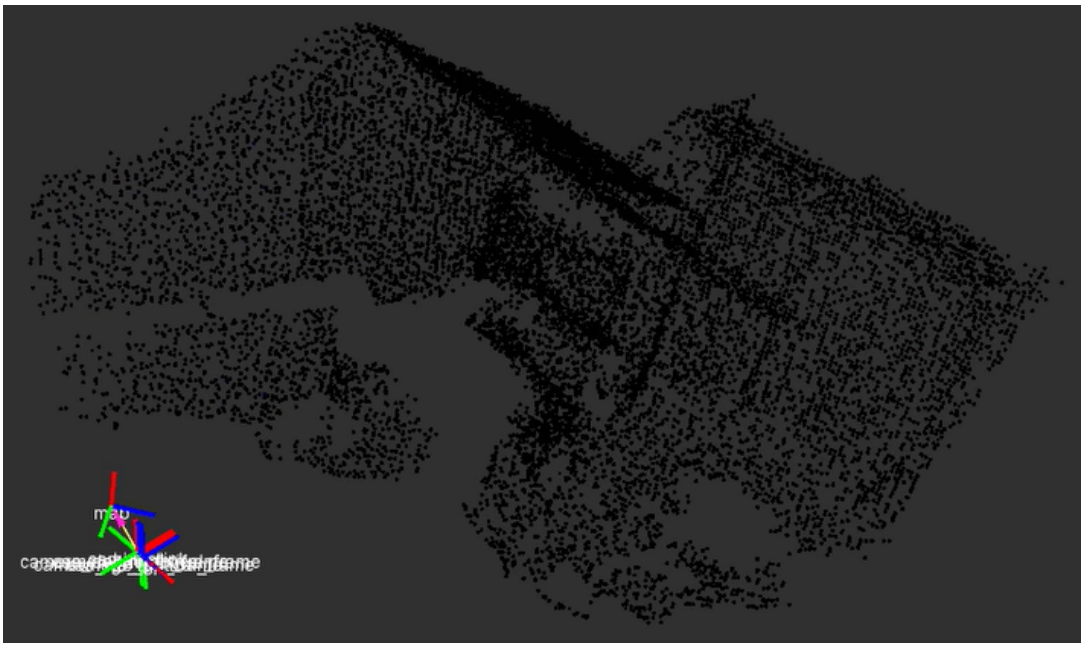


Şekil 4.5 Ethzasl-Icp-Mapper yöntemi kullanılarak aydınlatmanın az olduğu ortamda düşük çözünürlüklü sensör verisi ile elde edilen harita



Şekil 4.6 RTAB-Map yöntemi kullanılarak karanlık ortamda yüksek çözünürlüklü sensör verisi ile elde edilen harita

Özetle 3 boyutlu haritalamada RGB verisini kullanan RTAB-Map en iyi performansı aydınlık ortamda yüksek çözünürlükle çalıştırıldığında verirken sadece derinlik kullanan Ethzasl-Icp-Mapper en yüksek başarıyı düşük çözünürlükte verip haritalama başarısı aydınlatma seviyesinden etkilenmemektedir. Bununla birlikte, Ethzasl-Icp-Mapper kullanılarak başarılı bir haritanın elde edilebilmesi için sensör hareketinin çok yavaş olma zorunluluğu vardır.



Şekil 4.7 Ethzasl-Icp-Mapper yöntemi kullanılarak karanlık ortamda düşük çözünürlüklü sensör verisi ile elde edilen harita

4.4 RTAB-Map ve Kintinuous Sayısal Karşılaştırması

RTAB-Map ve Kintinuous sistemlerinin karşılaştırması ilk olarak Denklem 4.1 ile her bir sistemin robot yörüngesini tahmin performansları hesaplanıp karşılaştırmasıyla yapılmıştır. Sayısal karşılaştırma sonrasında ise her bir sistemin Bölüm 2.2 ile anlatılan 2015 Robocup Rescue German Open yarışma alanında toplanan veriler üzerinde oluşturdukları haritaların görsel bir karşılaştırılması yapılmıştır [66].

Her iki sistem de kendi varsayılan parametre değerleriyle test edilmiştir. Her bir veri kümesi için ayrı ayrı parametre ayarlarının yapılmasıyla, hem RTAB-Map hem de Kintinuous için iyileştirmeler yapmak mümkündür. Ancak, buradaki amaç maksimum performansa ulaşmaktan ziyade parametre ayarı olmaksızın sistemlerin performanslarının sayısal karşılaştırmasını yapmak olduğundan herhangi bir parametre düzeltmesi uygulanmamıştır.

Sistem testleri Bölüm 4.1 ile anlatılan veri koleksiyonunda bulunan 3 veri kümesi üzerinde yapılmıştır. Freiburg1_desk veri kümesi kameraya 6 serbestlik derecesinde hem öteleme hem de rotasyon hareketleri yaptırılarak toplanırken freiburg1_xyz ve freiburg1_rpy veri kümeleri, sırasıyla, sadece öteleme ve sadece rotasyon hareketleri ile toplanmıştır. Tablo 4.2 ve 4.3, sırasıyla RTAB-Map ve Kintinuous sistemlerinin bu veri kümeleri üzerindeki performans sonuçlarını vermektedir. Burada hatalar, veri kümeleri içerisinde bulunan gerçek konum bilgileri ile sistemlerin tahminleri arasındaki Öklid mesafesinin verilerin zaman bilgilerine göre eşleştirilmesiyle hesaplanmıştır.

Tablo 4.2 RTAB-Map performans sonuçları

Veri Kümesi	RMSE (m)	Ort. (m)	Std. Sapma (m)	Maks. (m)
freiburg1_desk	0.055525	0.050831	0.022343	0.113154
freiburg1_xyz	0.027489	0.023926	0.013536	0.067538
freiburg1_rpy	0.092531	0.083185	0.040524	0.214590

Tablo 4.2, RTAB-Map'in, sadece öteleme hareketlerinin bulunduğu freiburg1_xyz veri kümesi üzerinde kabul edilebilir bir hata oranıyla takip edebildiğini ancak daha çok rotasyon hareketi bulunduran freiburg1_desk ve freiburg1_rpy veri kümeleri üzeride belirgin bir performansın düşüşü olduğunu göstermektedir. Esasen rotasyon hareketinin takibini daha zor olması da beklenen bir durumdur [4].

Tablo 4.3 Kintinuous performans sonuçları

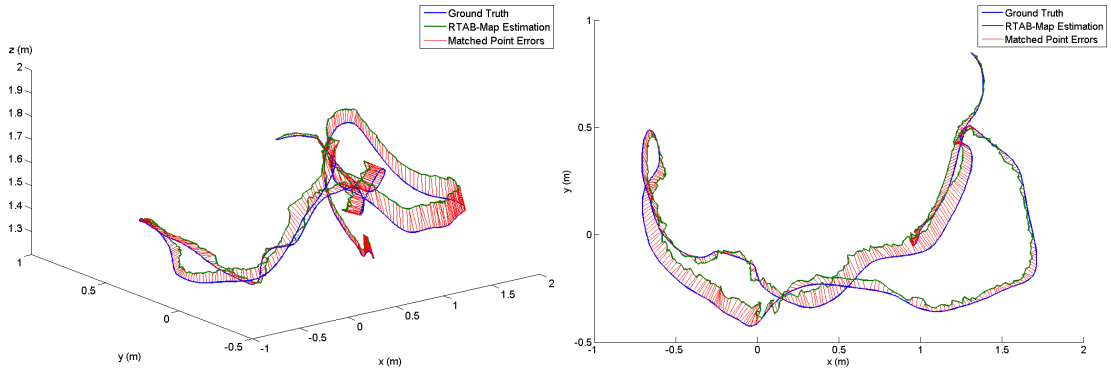
Veri Kümesi	RMSE (m)	Ort. (m)	Std. Sapma (m)	Maks. (m)
freiburg1_desk	0.202061	0.192311	0.062011	0.350430
freiburg1_xyz	0.079541	0.072708	0.032255	0.165750
freiburg1_rpy	0.092651	0.080228	0.046343	0.212796

Tablo 4.3, Kintinuous'un en düşük performansını kameranın hem öteleme hem rotasyon hareketini içeren freiburg1_desk veri kümesinde elde ettiğini göstermektedir. Bu da bize kamera hareketinin karmaşıklığı arttığında, Kintinuous için konum tahminin dramatik bir şekilde zorlaştığını göstermektedir.

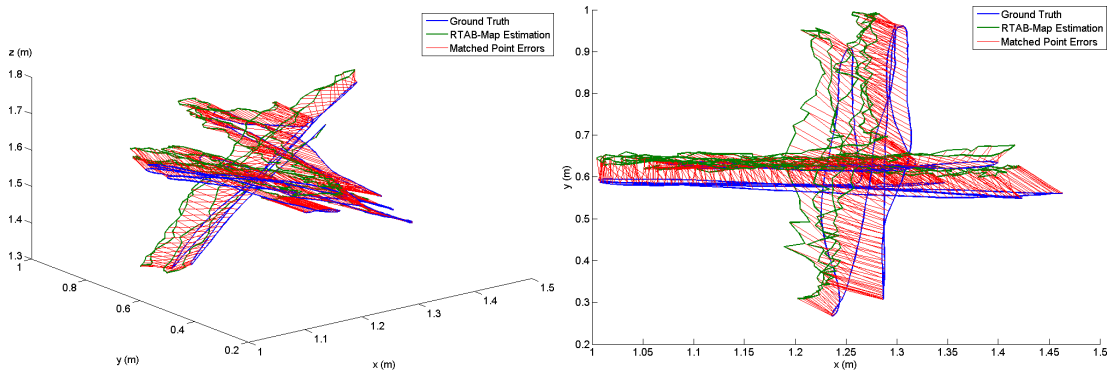
Sonuçlar genel olarak, varsayılan parametreler kullanıldığında tüm veri kümeleri için RTAB-Map'in daha iyi bir performans sergilediğini göstermektedir. Bununla birlikte her iki sistemin en iyi sonucu sadece öteleme hareketleri içeren freiburg1_xyz veri kümesinde verdiğini görüyoruz.

Şekil 4.8 ve 4.9 ile verilen grafikler, Tablo 4.2 ve 4.3 ile istatistiksel verileri verilen sonuçları görselleştirmektedir. Şekil 4.8 ve 4.9 her bir veri kümesi için oluşturulan 3 boyutlu sonuç grafiğini iki farklı açıdan göstermektedir. Soldaki grafiklerde orijin merkezli bakılarak 3 koordinat düzlemi de gösterilirken sağdaki grafiklerde X-Y düzlemlerine göre gösterim yapılmaktadır. Her bir grafikte, mavi çizgi kameranın gerçek konum grafiğini, yeşil çizgi sistem tarafından tahmin edilen konum grafiğini, kırmızı çizgiler ise zaman bilgilerine göre eşleştirilen gerçek konum ve tahmin edilen konum ikililerinin arasındaki hataları temsil etmektedir. Grafikler tablolarla tutarlı sonuçlar vermektedir.

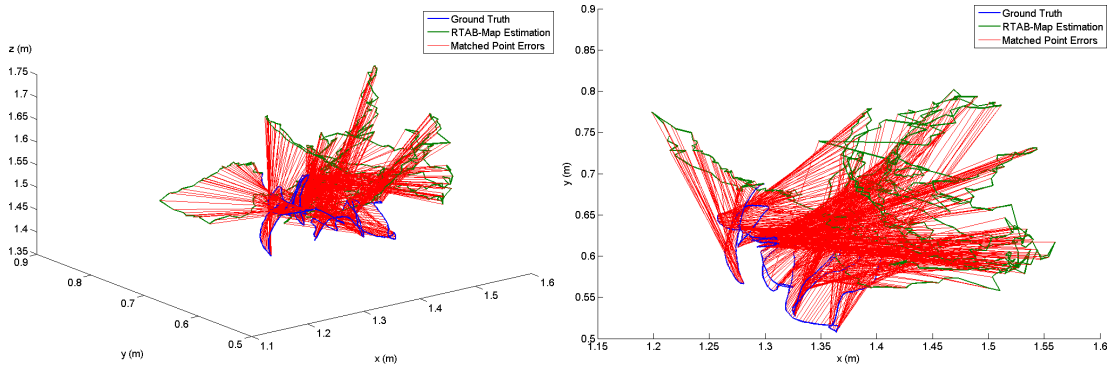
Sadece rotasyon hareketinin olduğu freiburg1_rpy veri kümesinde her iki sistem de



(a) Freiburg1_desk veri kümesi



(b) Freiburg1_xyz veri kümesi

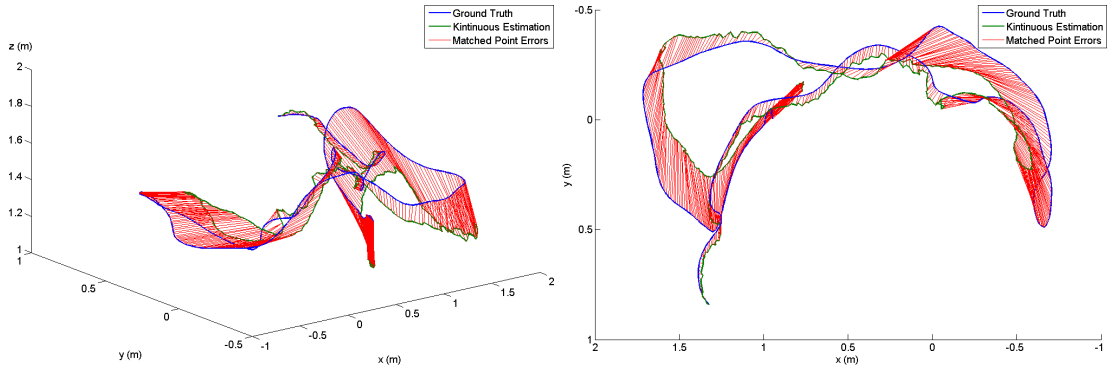


(c) Freiburg1_rpy veri kümesi

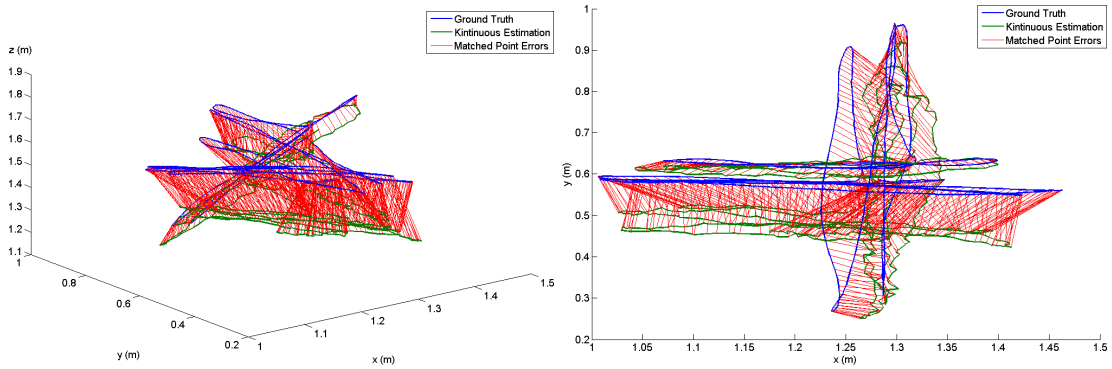
Şekil 4.8 3 farklı veri kümesi üzerinde RTAB-Map konum tahmin performans grafiklerinin orijin merkezli (solda) ve X-Y düzlemlerine göre gösterimi (sağda)

benzer performanslar gösterirken diğer iki veri kümesinde RTAB-Map daha başarılı sonuçlar vermektedir. Sistem performansları arası bu farklılık sadece Tablo 4.2 ve 4.3 ile verilen sonuçlardan değil aynı zamanda Şekil 4.8 ve 4.9 ile verilen grafiklerden de görülmektedir.

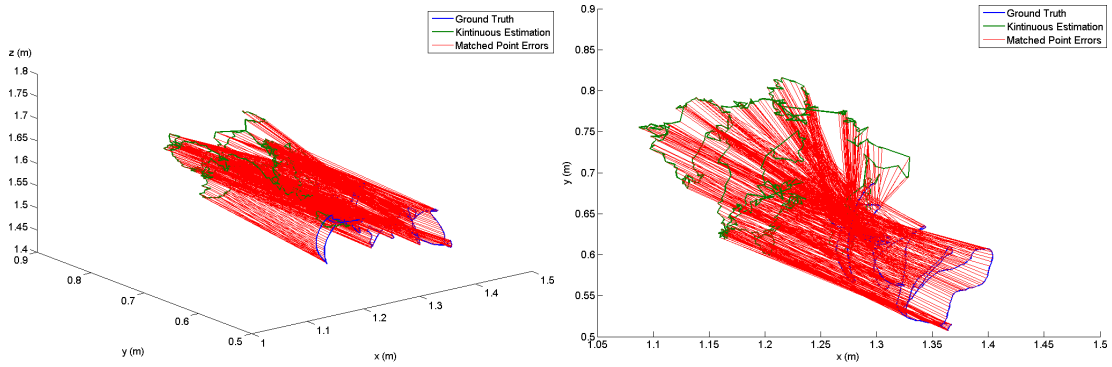
Bir sonraki RTAB-Map ile Kintinuous karşılaştırması ise daha önceden de bahsedildiği gibi Bölüm 2.2 ile anlatılan ve 2015 Robocup Rescue German Open yarışma alanında YTÜ Olasılıksal Robotik Grubu tarafından geliştirilen robotun gezdirilmesi ile elde edilen veriler kullanılarak yapılmıştır.



(a) Freiburg1_desk veri kümesi



(b) Freiburg1_xyz veri kümesi

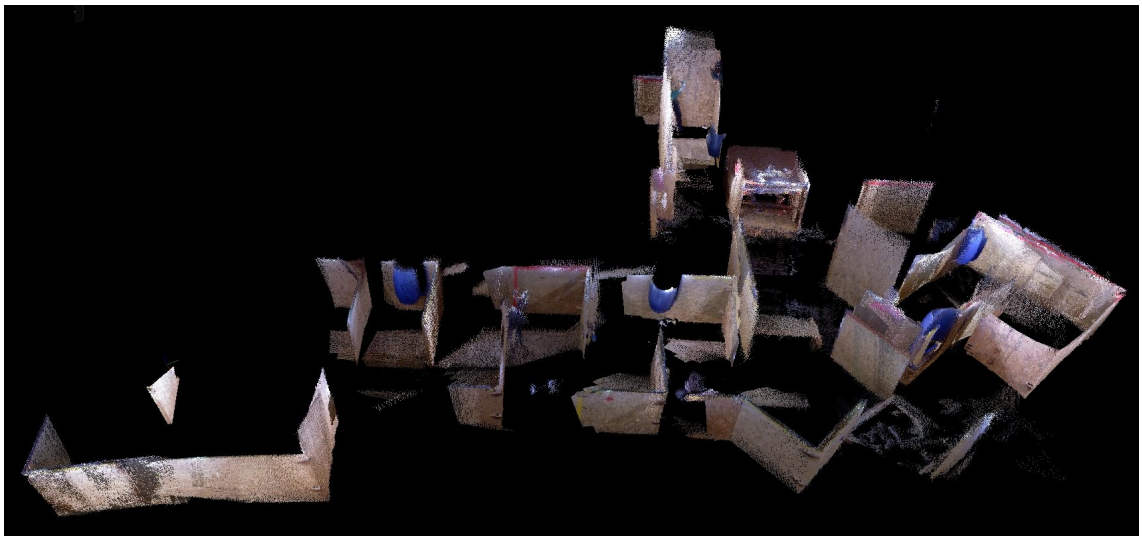


(c) Freiburg1_rpy veri kümesi

Şekil 4.9 3 farklı veri kümesi üzerinde Kintinuous konum tahmin performans grafiklerinin orijin merkezli (solda) ve X-Y düzlemlerine göre gösterimi (sağda)

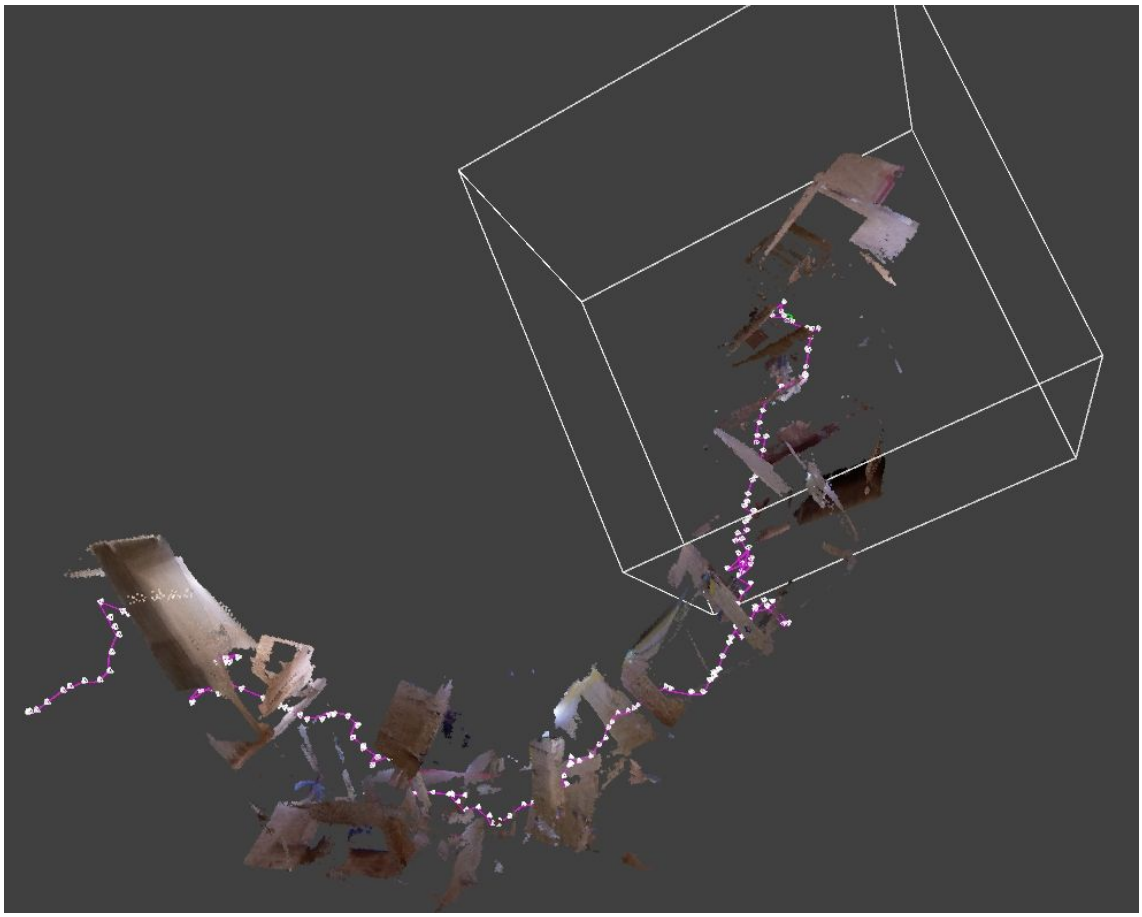
Bu gezinim sırasında, robotun gerçek konum bilgisinin toplanan veri içerisinde bulunmaması bu durum için sayısal karşılaştırmanın yapılamamasına sebep olmaktadır. Bununla birlikte, Şekil 4.10 ve 4.11 ile sırasıyla verilen RTAB-Map ve Kintinuous sistemlerinin oluşturduğu haritalar arasında görsel karşılaştırma yapıldığında performans farkı açıkça ayırt edilebilmektedir.

Şekil 4.10 ile gösterilen haritanın detaylarında duvarların oturtulması noktasında sıkıntılar olsa da oldukça başarılı olduğu söylenebilir. Burada haritalama sırasında robot labirentin başlangıç noktasına yakın bir konumdayken odometri kopması olduğundan odometri sıfırlama işlemi yapılmıştır. Başka bir ifadeyle, alanın tamamını,



Şekil 4.10 2015 Robocup Rescue German Open yarışma alanında toplanan veri kullanılarak RTAP-Map ile elde edilen harita

bir kopma ile karşılaşmadan, RTAB-Map ile haritalamak mümkün olmamıştır. Şekil 4.11 ile gösterilen harita ise tamamen başarısız bir sonuç vermektedir.



Şekil 4.11 2015 Robocup Rescue German Open yarışma alanında toplanan veri kullanılarak Kintinuous ile elde edilen harita

Tez kapsamında yapılan çalışmalarda, rotasyon hareketlerinde yaşanan performans düşüşüne rağmen, RTAB-Map kullanılmasına karar verilmesinin sebeplerinde birisi burada yapılan karşılaştırmalar sonucu her iki sistem tarafından üretilen haritaların başarıları arasında çok büyük fark olduğunun görülmesidir.

5
Önerilen Sistem

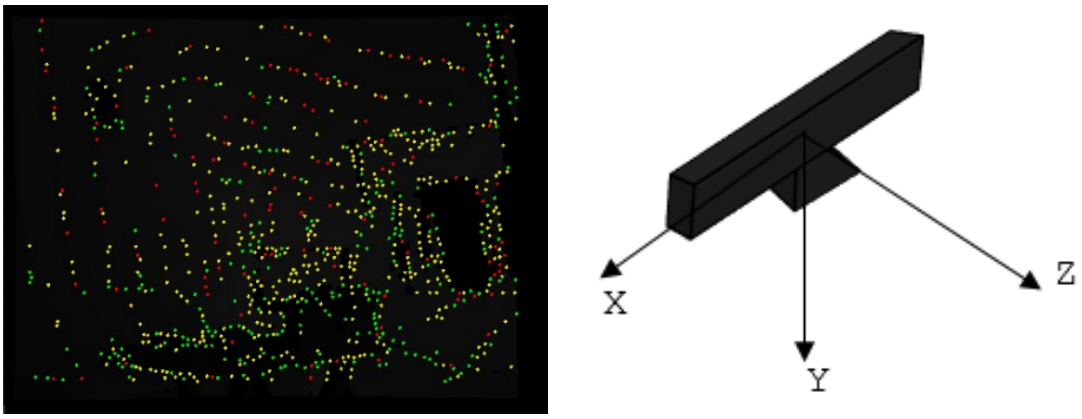
Bu bölümde, öncelikle tez kapsamında yapılan araştırmalar sonucu mevcut sistemlerin yetersiz olduğu belirlenen noktalar belirtilmiş ve bu noktaların giderilmesine yönelik geliştirilen hipotez anlatılmıştır. Sonrasında, geliştirilen hipotezin gerçekleştirilmesinde uygulanan sistem adımlarının detayları verilmiştir. Son olarak ise önerilen sistemin oluşturduğu ekstra maliyet dolayısıyla performansa etkisinden bahsedilmektedir.

5.1 Mevcut Eksikler ve Hipotez

Daha önce de bahsedildiği gibi mevcut sistemler kamera derinlik menzili içerisinde yeterli öznitelik noktasının bulunamaması durumunda başarısız olmaktadır. Bu nedenle önerilen hipotez Bölüm 1.3 ile örnek verilen ortamlara benzer durumlarda iyileştirme yapılmasına yönelik geliştirilmiştir.

Tez kapsamında yapılan çalışmalarda 3B haritalama sistemlerinde RGB veriye olan bağımlılığın azaltılabilmesi için mevcut sistemler araştırılmış ve Bölüm 3.4 ile anlatıldığı gibi farklı giriş verileri denenmiştir. Elde edilen sonuçlardan biri olan derinlik resmi üzerinde bulunan öznitelik noktaları Şekil 5.1a ile gösterilmiştir. Derinlik noktalarının giriş olarak verilebilmesi için Şekil 5.1b ile gösterilen kamera koordinatına göre okunmuş olan derinlik bilgisi 0-255 arasında renk düzlemine dönüştürülerek resim haline getirilmektedir. Ancak kamera konumu değiştiğinde aynı noktaya olan derinlik değiştiğinden yapılan renklendirme de değişmektedir. Sonuç olarak, iki farklı resimde bulunan aynı öznitelik noktasının farklı renklere sahip olması dolayısıyla haritalama ve odometri takibi için eşleştirme yapılamamaktadır. Bu da derinlik resminin direkt kullanımının RGB veriye alternatif olarak sunulamamasına sebep olmaktadır.

Harita üzerindeki aynı nokta için yapılan renklendirmenin sabit kalabilmesi, verinin kamera üzerinde bulunan hareketli bir koordinat yerine sabit bir koordinata göre renklendirilmesinin sağlanmasıyla mümkün olabilir. Bu nedenle, önerilen sistem



(a) (b)

Şekil 5.1 (a) Giriş olarak verilen derinlik resmi üzerinde bulunan öznitelik noktaları (b) Derinlik resminin renklendirilmesinde kullanılan Kinect üzerindeki koordinat sistemi

olan DepthTiling derinlik resminin sabit bir koordinat olan haritaya göre yeniden renklendirilmesiyle bu duruma bir çözüm sunmaktadır.

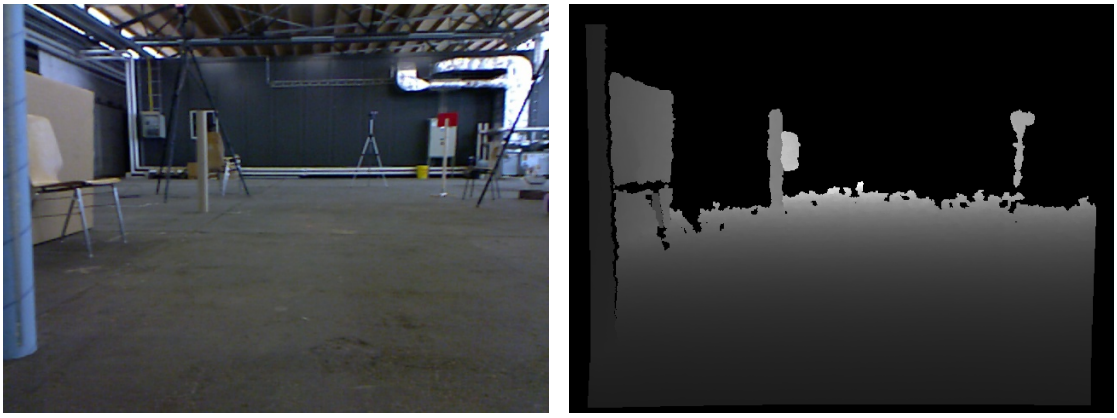
Tez sürecinde, deneylerde kullanılan RGB-D veri koleksiyonunun [65] detayları Bölüm 4.1 ile anlatılmaktadır. Bu veri koleksiyonu, çeşitli ortamlarda Kinect'ten alınan RGB-D verileri bulundurmaktadır.

Önerilen sistem olan DepthTiling'in anlatımı sırasında freiburg2_pioneer_slam isimli veri kümesi kullanılmıştır. Bu veri kümesinin seçilmesinin sebebi, derinlik menzili içerisinde yeterli öznitelik noktası bulunmadığından Bölüm 1.2 ile anlatılan sistemlerin çoğunun başarısız olmasıdır.

Şekil 5.2 ile test amacıyla kullanılan freiburg2_pioneer_slam isimli veri kümesinden alınmış örnek bir RGB görüntü ve bu görüntüye ait Kinect'ten alınan derinlik resmi verilmiştir. Şekil 5.2b ile verilen resimdeki siyah kısımlar derinlik verisinin ölçülemediği alanları göstermektedir.

5.2 Önerilen Sistemin Uygulama Aşamaları

DepthTiling temel olarak RGB-D kameradan okunan nokta bulutu verisini harita üzerindeki konumlarına göre 0-255 aralığında renklendirmektedir. Nokta bulutundaki her bir noktanın konumunun RGB kanal renkleri olarak temsil edilebilmesi için 5 adım uygulanmaktadır. Renklendirmedeki çeşitlilik her bir adımda biraz daha artmakta ve işlem sonunda, derinlik resmi, öznitelik çıkarımı ve eşleme için uygun hale gelmektedir. DepthTiling işlem adımları kısaca Algoritma 1 ile verilmiştir.



(a) (b)

Şekil 5.2 freiburg2_pioneer_slam isimli veri kümesinden (a) örnek bir RGB resim ve (b) bu resme ait derinlik görüntüsü

Algoritma 1: DepthTiling Adımları

Input: RGB-D Veri

Output: Renklendirilmiş Resim

1 Nokta koordinatlarını (lowerBound, upperBound] aralığında normalize et.

2 XYZ koordinat aralıklarını RGB renk kanal aralıklarına çevir.

3 Renk kanallarının arasındaki farklılığı arttırarak karolar elde et.

4 Karolar arasındaki renk değişimini arttır.

5 Veri kaybını azaltmak için resim üzerinde morfolojik closing işlemi uygula.

Algoritma 1 ile verilen ilk adım her bir noktanın hangi koordinat aralığında bulunduğunun belirlenmesi için Eşitlik 5.1 ile koordinatların normalize edilmesidir. Daha önceden belirlenen $intervalSize$ içerisindeki tüm noktalar aynı aralıkta kabul edilmektedir. Bu adım, sistemin sonsuz koordinat düzlemine sahip gerçek dünyanın $256 \times intervalSize$ boyutuna indirgenmesini sağlamaktadır. Deneysel sonuçlarda en yüksek başarının $intervalSize = 0.1\ m$ olduğunda elde edildiği gözlemlenmiştir. Bu nedenle, tez boyunca anlatılan deneylerde bu değer kullanılmıştır.

$$
\begin{aligned}
xIntv &= \left\{ \frac{x - lowerBound}{intervalSize} \;\middle|\; lowerBound < x \leq upperBound \right. \\
yIntv &= \left\{ \frac{y - lowerBound}{intervalSize} \;\middle|\; lowerBound < y \leq upperBound \right. \\
zIntv &= \left\{ \frac{z - lowerBound}{intervalSize} \;\middle|\; lowerBound < z \leq upperBound \right.
\end{aligned}
\tag{5.1}
$$

Burada, x, y ve z noktanın haritadaki koordinatları, $xIntv$, $yIntv$ ve $zIntv$ normalize edilmiş XYZ koordinat aralıkları, $intervalSize$ hareketin sonucu etkilemesi için gerekli minimum eşik değeri, $lowerBound$ ve $upperBound$ ise sonucun 0-255 aralığında normalize edilmesini sağlamak için $intervalSize$ değerine göre belirlenmiş alt ve üst limit değerleridir.

Şekil 5.3, ilk adımın Şekil 5.2b ile verilen derinlik verisine uygulanmasıyla elde edilen $xIntv$, $yIntv$ ve $zIntv$ sonuçlarının RGB değerleri olarak kullanıldığında oluşan derinlik resmini göstermektedir. Burada pozisyona bağlı olarak renk değişimi gözlemlenebildiği halde renk geçişleri öznitelik çıkarımı için yeterli değildir.

Şekil 5.3 Şekil 5.2b ile verilen derinlik verisine uygulanan ilk adım sonucu elde edilen resim

Algoritma 1 ile verilen ikinci adım, Eşitlik 5.2 ile XYZ koordinat aralıklarını RGB kanal aralıklarına çevirmektedir.

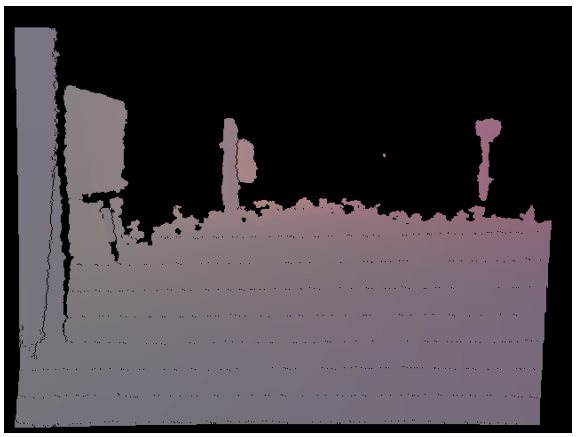
$$\begin{aligned} rIntv &= \alpha * xIntv + \beta * yIntv + \gamma * zIntv \\ gIntv &= \gamma * xIntv + \alpha * yIntv + \beta * zIntv \\ bIntv &= \beta * xIntv + \gamma * yIntv + \alpha * zIntv \end{aligned} \tag{5.2}$$

Burada, $xIntv$, $yIntv$ ve $zIntv$ değişkenleri Eşitlik 5.1 ile hesaplanan değerlerdir. α, β ve γ değişkenleri $xIntv$, $yIntv$ ve $zIntv$ değerlerinin her bir renk kanalında ne kadar etkin olacağını belirleyen katsayılardır. α, β ve γ değerleri belirlenirken $rIntv$, $gIntv$ ve $bIntv$ değişkenlerinin 0-255 aralığında kalmalarının sağlanabilmesi için $\alpha + \beta + \gamma = 1$ koşulu uygulanmaktadır. Bu katsayılar deneysel testler ile en iyi sonucu verecek şekilde belirlenmiştir. Deneysel tesler ile en başarılı sonuç $\alpha = 0.7$, $\beta = 0.1$ ve $\gamma = 0.2$ durumunda elde edilmiştir. Bu nedenle, tez boyunca anlatılan deneylerde bu değerler kullanılmıştır. Son olarak, $rIntv$, $gIntv$ ve $bIntv$ ise RGB

rengi olarak neyin kullanılacağını belirleyen RGB kanal aralıklarını temsil etmektedir.

Hesaplanan koordinat aralıkları tüm RGB kanallarını farklı oranlarda etkilemektedir. Bu durum ise renk çeşitliliği sağladığından mevcut noktaların koordinatları sadece belirli koordinat düzlemlerinde değişiyor olsa dahi öznitelik çıkarımını mümkün kılmaktadır. Örneğin, Şekil 5.2 ile verilen resimde olduğu gibi RGB-D kameranın sadece zemine ait derinlik verisi verdiği durumlarda, nokta bulutu içerisindeki noktaların yükseklik verisinde bir değişiklik olmamaktadır. Bu nedenle $zIntv$ değişkeni tüm noktalar için aynı sonucu vermektedir. Ancak, bu noktalar için Mavi kanal rengini belirleyen $bIntv$ değişkeninin değeri sadece $zIntv$ değişkenie değil aynı zamanda $xIntv$ ve $yIntv$ değişkenlerine de bağlı olduğu için farklılık göstermektedir.

Şekil 5.4, ikinci adımın Şekil 5.2b ile verilen derinlik verisine uygulanması sonucu elde edilen $rIntv$, $gIntv$ ve $bIntv$ sonuçlarının RGB değerleri olarak kullanıldığında oluşan derinlik resmini göstermektedir. Her ne kadar zemin rengindeki değişim Şekil 5.3 ile verilen resme göre daha iyi olsa da, öznitelik çıkarım algoritmaları için yetersiz kalmaktadır.



Şekil 5.4 Şekil 5.2b ile verilen derinlik verisine uygulanan ikinci adım sonucu elde edilen resim

Öznitelik çıkarım algoritmaları resimlerde kenar ve köşe gibi en belirleyici noktaları bulmaktadırlar. Bu nedenle, Şekil 5.4 ile verilen resimde yapılan renklendirmenin zemin gibi düz yüzeylerde kenarlar oluşturacak şekilde çeşitlendirilmesi gerekmektedir. Algoritma 1 ile verilen üçüncü adım her bir kanalın renginin belirlenmesinde farklı durumlar için değişik koordinatların etkili olmasını sağlamaktadır. Eşitlik 5.3 her bir kanal için bulundukları aralığa göre renklendirmede yedi farklı seçenek sağlamaktadır. Bu adımdan sonra, derinlik verisine bağlı olarak renk karoları gözlemlenebilmektedir.

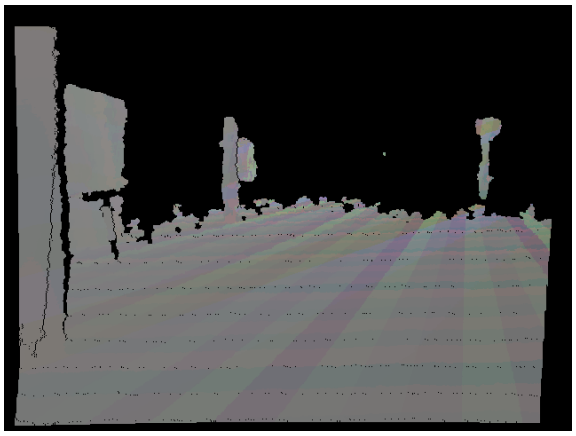
$$
rCube = \begin{cases} rIntv & mod(rIntv,7) = 0 \\ gIntv & mod(rIntv,7) = 1 \\ bIntv & mod(rIntv,7) = 2 \\ \frac{rIntv+gIntv}{2} & mod(rIntv,7) = 3 \\ \frac{rIntv+bIntv}{2} & mod(rIntv,7) = 4 \\ \frac{gIntv+bIntv}{2} & mod(rIntv,7) = 5 \\ \frac{rIntv+gIntv+bIntv}{3} & mod(rIntv,7) = 6 \end{cases}
$$

$$
gCube = \begin{cases} gIntv & mod(gIntv,7) = 0 \\ bIntv & mod(gIntv,7) = 1 \\ \frac{rIntv+gIntv}{2} & mod(gIntv,7) = 2 \\ \frac{rIntv+bIntv}{2} & mod(gIntv,7) = 3 \\ \frac{gIntv+bIntv}{2} & mod(gIntv,7) = 4 \\ \frac{rIntv+gIntv+bIntv}{3} & mod(gIntv,7) = 5 \\ rIntv & mod(gIntv,7) = 6 \end{cases} \tag{5.3}
$$

$$
bCube = \begin{cases} bIntv & mod(bIntv,7) = 0 \\ \frac{rIntv+gIntv}{2} & mod(bIntv,7) = 1 \\ \frac{rIntv+bIntv}{2} & mod(bIntv,7) = 2 \\ \frac{gIntv+bIntv}{2} & mod(bIntv,7) = 3 \\ \frac{rIntv+gIntv+bIntv}{3} & mod(bIntv,7) = 4 \\ rIntv & mod(bIntv,7) = 5 \\ gIntv & mod(bIntv,7) = 6 \end{cases}
$$

Burada, $rIntv$, $gIntv$ ve $bIntv$ değişkenleri Eşiklik 5.2 ile hesaplanan değerlerdir. $rCube$, $gCube$ ve $bCube$ ise bulunduğu kanal aralığına bağlı olarak hesaplanan sonuçlardır.

Şekil 5.5, üçüncü adım sonucu elde edilen $rCube$, $gCube$ ve $bCube$ sonuçlarının RGB değerleri olarak kullanıldığında oluşan derinlik resmini göstermektedir. Her ne kadar zemine döşenmiş olan küçük karoların kenarları belli olsa da öznitelik eşleme algoritmalarında daha iyi sonuç vermesi adına değişkenliğin arttırılması gerekmektedir. Karolar arasındaki renk değişim miktarının arttırılması, daha doğru öznitelik çıkarımı ve eşlemesi yapılmasını sağlayacağından dördüncü adım uygulanmaktadır.



Şekil 5.5 Şekil 5.2b ile verilen derinlik verisine uygulanan üçüncü adım sonucu elde edilen resim

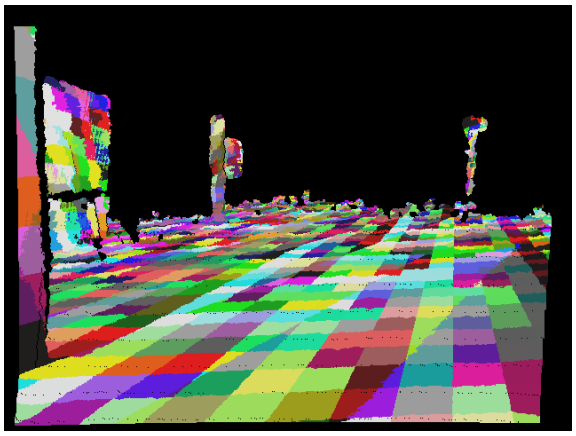
Dördüncü adımda, Eşitlik 5.4 kullanılarak, Şekil 5.5 ile gösterilen, zemine döşenmiş olan karolar arasındaki renk değişimi arttırılmaktadır. Bu adımın uygulanmasıyla, daha iyi öznitelik eşlemesi yapılmasını sağlayan daha belirleyici öznitelik noktalarına sahip resimler üretilmektedir.

$$
\begin{aligned}
r &= \left\{ stpSize * mod(rCube, stpNum) + \frac{rCube}{stpNum} \;\middle|\; stpSize * stpNum = 256 \right. \\
g &= \left\{ stpSize * mod(gCube, stpNum) + \frac{gCube}{stpNum} \;\middle|\; stpSize * stpNum = 256 \right. \\
b &= \left\{ stpSize * mod(bCube, stpNum) + \frac{bCube}{stpNum} \;\middle|\; stpSize * stpNum = 256 \right.
\end{aligned}
\tag{5.4}
$$

Burada, $rCube$, $gCube$ ve $bCube$ değişkenleri Eşiklik 5.3 ile hesaplanan değerlerdir. $stpSize$ ilgili kanalda aralıkların geçişlerindeki renk değişim miktarıdır. $stpNum$ değişkeni renk değerlerinin 0-255 aralığında kalması şartını korumak üzere belirli $stpSize$ için uygulanabilir adım sayısıdır. r, g ve b değişkenleri ise ilgili RGB kanalına ait renk değerlerini temsil etmektedir.

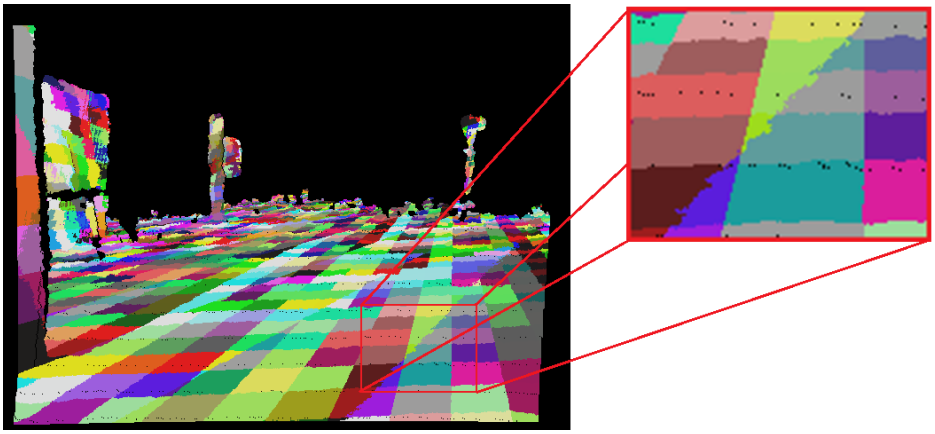
Şekil 5.6 dördüncü adım sonucu elde edilen r, g ve b sonuçlarının RGB değerleri olarak kullanıldığında oluşan derinlik resmini göstermektedir. Bu adımdan sonra renklendirme ile ilgili herhangi bir ayar yapmaya gerek yoktur. Ancak derinlik verisi içeren öznitelik noktalarının tespit edilebilmesi için son bir adımın daha uygulanması gerekmektedir.

Şekil 5.7, Şekil 5.6 ile verilen resimde bulunan zemin renklendirmesinin detaylarını göstermektedir. Daha önce de bahsedildiği gibi siyah noktalar derinlik



Şekil 5.6 Şekil 5.2b ile verilen derinlik verisine uygulanan dördüncü adım sonucu elde edilen resim

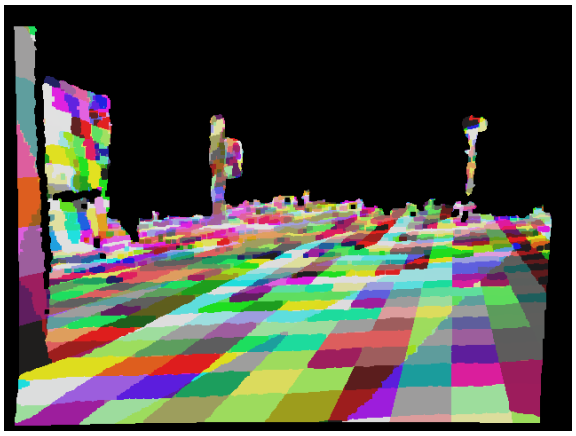
verisinin olmadığı alanları ifade etmektedir ve burada zemin üzerinde veri kaybının yaşandığı gözlemlenebilmektedir. Dolayısıyla, renklendirme işlemi bu noktalara uygulanamadığından Şekil 5.7 ile verilen resimde siyah benekler olarak gözükmektedirler. Bu benekler ise resim üzerinde öznitelik çıkarım algoritmaları için en belirleyici noktaları oluşturmaktadır. Ancak derinlik verisi olmadığı için bu noktalar kullanışsız olmaktadırlar.



Şekil 5.7 Şekil 5.2b ile verilen derinlik verisine uygulanan dördüncü adımın detaylı sonucu

Algoritma 1 ile verilen son adımda, derinlik verisine sahip olmayan noktaların öznitelik çıkarımında seçilmesini engellemek amacıyla, dördüncü adım sonucu elde edilen resim üzerine morfolojik closing işlemi uygulanmaktadır. Böylece bu noktalar yok edilmiş ve kullanışlı öznitelik noktalarının seçilme ihtimali arttırılmış olmaktadır. Şekil 5.8, beşinci adımdan sonra elde edilen resmi göstermektedir.

RGB resmin haritalamada yeterli olduğu durumlarda yeniden renklendirme işleminin uygulanması gerekmemektedir. Haritalama sırasında kullanılan RGB resim ile odometri kaybının oluştuğu durumlarda, DepthTiling sisteminin aktive edilmesi



Şekil 5.8 Şekil 5.2b ile verilen derinlik verisine uygulanan beşinci adım sonucu elde edilen resim

sistem maliyeti açısından daha uygundur.

5.3 Sistem Maliyeti

Önerilen sistem olan DepthTiling, 7. nesil Intel Core i7-7700K işlemcisine sahip, 16 GB DDR4 RAM kapasiteli ve Geforce GTX 1070 ekran kartı donanımına sahip bir bilgisayarda geliştirilip test edilmiştir. Yazılımsal olarak ise DepthTiling'in sonuç ve sağladığı iyileştirmenin değerlendirilmesi için literatürde performans açısından en başarılı haritalama sistemi olması dolayısıyla RTAB-Map uygulaması tercih edilmiştir [67].

Bir sistemin gerçek zamanlı kabul edilebilmesi için en az 1 Hz frekansında çalışması, başka bir ifadeyle haritalama sırasında saniyede 1 resim çerçevesini işleyerek konum takibi yapabilmesi gerekmektedir ve RTAB-Map RGB resim ile haritalamada ortalama 0.71 saniye işlem süresiyle gerçek zamanlı çalışma şartını yerine getirmektedir [36]. DepthTiling üzerinde yapılan testlerde ise 3 saniyede 1 resim renklendirilmesi yapılabilmiştir. Sistemin mevcut durumunun gerçek zamanlı çalışmaya uygun olmaması dolayısıyla sistem testleri sadece kaydedilmiş veri üzerinde yapılmıştır.

6
Deneysel Sonuçlar

Performans karşılaştırmasında öznitelik çıkarma ve eşleme işlemleri için GFTT ve BRIEF algoritmalarının kombinasyonu kullanılmıştır. Bu algoritmaların kullanılmasının sebebi literatürde lokalizasyon başarısı ve hesaplama maliyeti açısından en başarılı ikilemi sağlıyor olduğunun düşünülmesidir [19]. Bu kombinasyon, Bölüm 4.1 ile anlatılan veri koleksiyonu içerisindeki 9 farklı veri kümesi üzerinde çalıştırılan görsel SLAM algoritması tarafından kullanılarak test edilmiş ve ortalama 0.1913 saniye çalışma süresi içerisinde 0.0387 m ortalama hata performansı ile odometri takibinin başarılı bir şekilde gerçekleştirilmesini sağlamıştır [68]. Bu durum GFTT+BRIEF kombinasyonunun eşleştirdiği öznitelik noktalarının odometri takibinde güvenilir bir şekilde kullanılabileceğini göstermektedir.

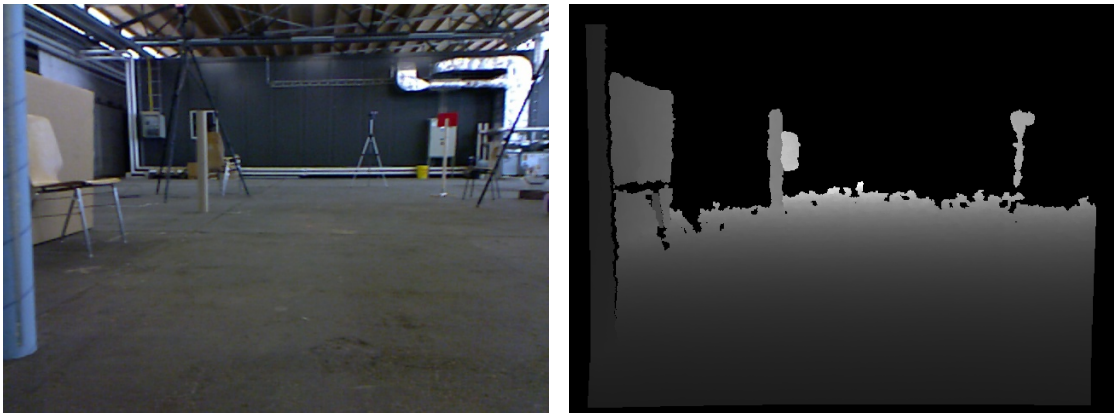
Bu bölümde, ilk olarak DepthTiling'in uygulanmasıyla 3B haritalama sistemlerinde sağlanabilecek geliştirmeye bir örnek verilmiştir. Sonrasında ise deneysel sonuçların sayısal karşılaştırması yapılarak DepthTiling'in sağladığı avantajlar belirtilmiştir.

6.1 Örnek Sistem Çıktısı

Daha önce de bahsedildiği gibi, sadece peş peşe gelen iki resim arasında öznitelik eşlemesinin yapılmış olması değil ayrıca bu eşleştirilen noktaların derinlik verisine de sahip olması odometri takibi ve 3B harita oluşturulması için büyük önem arz etmektedir. Başka bir ifadeyle, öznitelik noktalarının tespitinin yanında bu noktaların derinlik verisine sahip olmasının sağlanması görsel SLAM sistemleri için bir zorunluluktur.

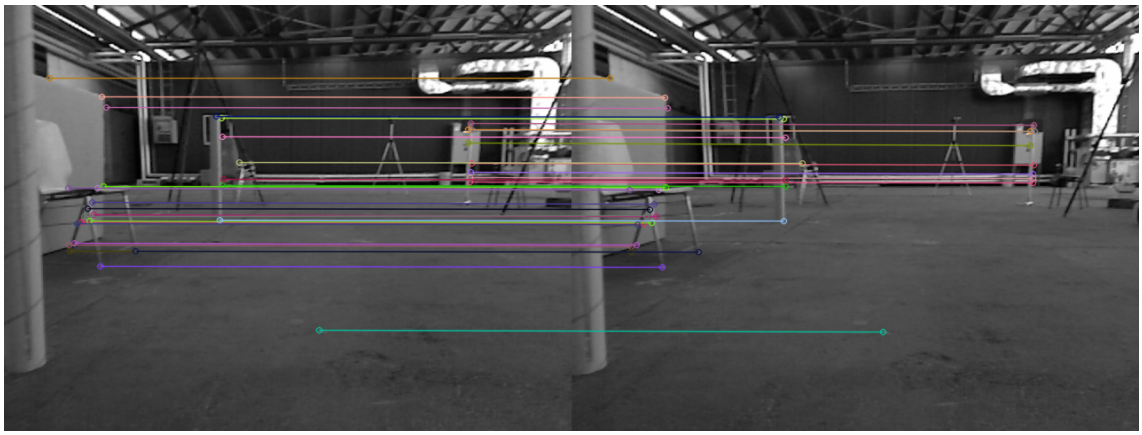
Şekil 6.1a ile gösterilen RGB resim üzerine GFTT algoritması uygulandığında, 641 öznitelik noktası tespit edilebilmektedir. Ancak bu noktaların sadece 41 tanesi Şekil 6.1b ile gösterilen derinlik resminde veri içermektedir. Bu da öznitelik eşleştirmesinde yalnız bu 41 noktanın kullanılabileceği anlamına gelmektedir. Bir sonraki RGB resmi üzerinde bulunan 666 öznitelik noktasının 54 tanesi derinlik verisi içermektedir.

Peş peşe gelen iki resim üzerinde bulunan bu, sırasıyla, 41 ve 54 noktanın eşleştirilmesinden elde edilen 34 öznitelik ikilileri Şekil 6.2 ile gösterilmektedir. Bu durumun odometri takibi için oldukça zayıf bir kaynak oluşturduğu söylenebilir.



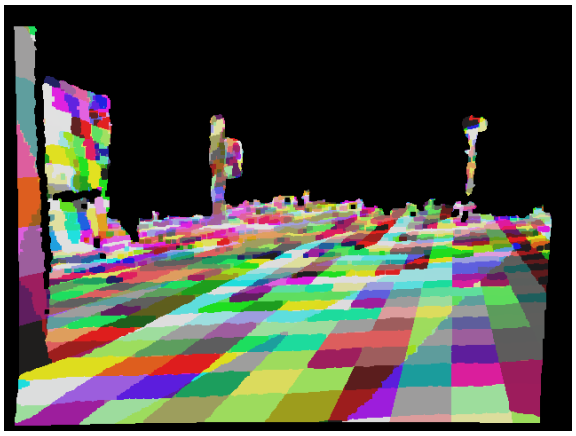
(a) (b)

Şekil 6.1 freiburg2_pioneer_slam isimli veri kümesinden (a) örnek bir RGB resim ve (b) bu resme ait derinlik görüntüsü

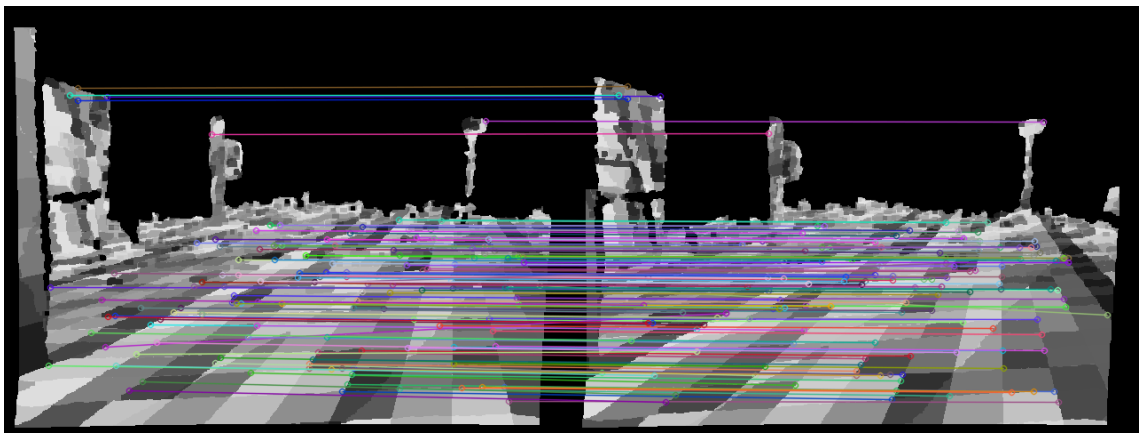


Şekil 6.2 Şekil 6.1 ve ardından gelen resim üzerinde derinlik verisine sahip eşlenmiş öznitelik noktaları

Diğer tarafta ise, Şekil 6.1 üzerine DepthTiling uygulanması sonucu elde edilen ve Şekil 6.3 ile gösterilen resme GFTT algoritması uygulandığında çıkarılan 1000 öznitelik noktasından 835 tanesi derinlik verisi içermektedir. Bir sonraki resim üzerinde bulunan 1000 noktanın da 835 tanesi derinlik verisi içermekte ve yapılan eşleme sonucu elde edilen 135 öznitelik ikilileri Şekil 6.4 ile gösterilmektedir. Bu örnekte, DepthTiling tarafından yeniden renklendirilen resmin daha güvenilir bir sonuç verdiği açıkça gözükmektedir. DepthTiling uygulaması ile ilgili daha fazla deney sonucu Bölüm 6.2 ile anlatılmıştır.



Şekil 6.3 Şekil 6.1 ile verilen resim üzerine DepthTiling sisteminin uygulanması sonucu elde edilen resim



Şekil 6.4 Şekil 6.3 ve ardından gelen resim üzerinde derinlik verisine sahip eşlenmiş öznitelik noktaları

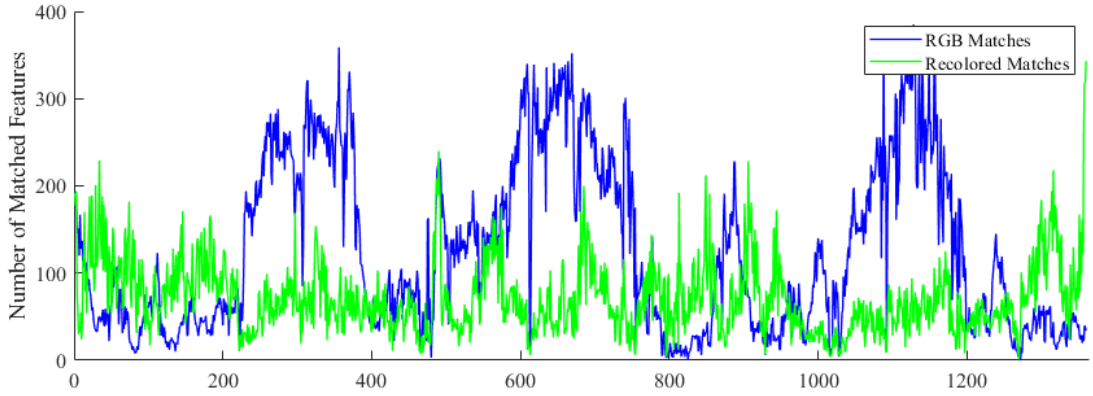
6.2 Sayısal Sonuçlar

Tez kapsamında, deneysel sonuçlar iki farklı kriterle değerlendirilmiştir. İlk olarak, farklı veri kümeleri için RGB resim ve DepthTiling tarafından yeniden renklendirilen resimler üzerinde eşleştirilen öznitelik sayıları karşılaştırılmıştır. Ne kadar fazla eşleşme mümkün olursa, odometri takibi için o kadar güvenilir sonuç elde edilir demektir. İkinci olarak, Hem RGB resim hem de renklendirilen resim ayrı ayrı giriş olarak verildiğinde odometri kaybının oluşup oluşmadığına bakılmıştır. Odometri kaybının olması durumunda, her iki giriş seçeneği için kopma zamanlarının değişkenlik göstermesi önemlidir. Böylece, DepthTiling'in, RGB resmin yetersiz kaldığı durumlarda alternatif olması mümkün olmaktadır.

DepthTiling detayları Bölüm 4.1 ile verilen veri koleksiyonu içerisinde bulunan 6 farklı veri kümesi (freiburg2_pioneer_360, freiburg2_pioneer_slam, freiburg2_pioneer_-slam2, freiburg2_pioneer_slam3, freiburg3_nostructure_notexture_far, freiburg3_no-structure_notexture_near_withloop) üzerinde test edilmiştir. Bazı veri kümeleri

resimlerde öznitelik çıkarımı için yeterli doku bulundurmazken, bazıları derinlik kamerasının menzilini aşan geniş alanlarda kaydedilmiştir. Kısacası, kullanılan veri kümeleri 3B öznitelik çıkarımı için farklı zorluklar içermelerine göre seçilmiştir.

Veri kümelerinden bir tanesi Şekil 6.1 ile içerdiği örnek bir resim verilen freiburg2_pioneer_slam isimli veri kümesidir. Bu veri kümesi, bir robot üzerine takılı Kinect'ten yapılan 155 saniyelik kayıttan oluşmaktadır. Robot masa, konteyner ve duvarlardan oluşan bir labirent içerisinde joystick ile sürülmüştür. GFTT ve BRIEF algoritmalarının kombinasyonu bu veri kümesi içerisindeki RGB resimler ve DepthTiling tarafından yeniden renklendirilmiş resimler üzerinde uygulanmış, derinlik bilgisine sahip eşleşmiş öznitelik sayıları Şekil 6.5 ile verilen grafikte gösterilmiştir. Daha önce de belirtildiği gibi GFTT ve BRIEF algoritmaları görsel SLAM sistemlerinde en çok tercih edilen kombinasyonlardan biridir [19].



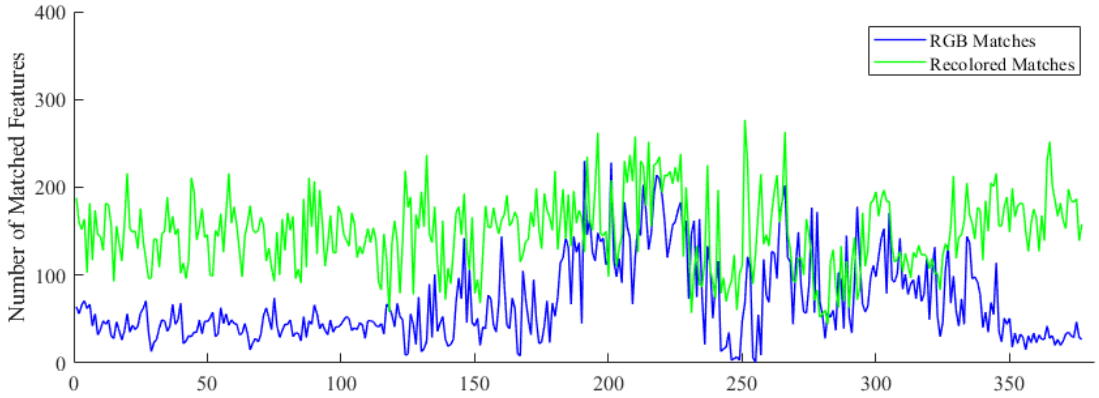
Şekil 6.5 freiburg2_pioneer_slam veri kümesi için derinlik bilgisine sahip öznitelik eşlerinin sayısı

Şekil 6.5 ile verilen grafikteki mavi ve yeşil çizgiler sırasıyla 1360 farklı RGB ve DepthTiling ile renklendirilmiş resim sonucunu göstermektedir. Belirli zaman aralıkları için veri kümesi 3B haritalamaya uygun RGB resimler içerdiğinden bu zaman aralıklarında başka herhangi bir sistemin kullanılması gerekli değildir. Ancak Şekil 6.5 ile sonucu verilen ilk 200 resimde olduğu gibi bazı RGB resimleri hem haritalama hem de odometri takibi için yetersiz sayıda öznitelik eşleşmesine sahiptir. Bu gibi durumlarda renklendirilmiş resimler odometri kaybından kaçınmak için kullanılabilir.

Şekil 6.5, RGB ve renklendirilmiş resimler kullanıldığında eşleşen öznitelik sayısında bir ters orantı olduğunu göstermektedir. RGB resimlerde yeterli sayıda öznitelik bulunmadığında, renklendirilmiş resimler üzerinde daha fazla öznitelik eşlemesi yapılabilmektedir. Bu durum her iki giriş tipinin birbirine alternatif olarak kullanılabileceğini göstermektedir. Giriş seçeneklerinden biri yetersiz kaldığında diğer seçeneğin giriş olarak kullanılması mümkündür. Yeniden renklendirme işlemi

ekstra hesaplama maliyeti gerektirdiğinden, RGB resimler haritalamada önceliğe sahip olması daha uygun gözükmektedir.

Freiburg3_nostructure_notexture_near_withloop, 37 saniyelik bir veri kümesidir ve yaklaşık 3×3 m boyutundaki düz bir tahta üzerinde yerden yaklaşık bir metre yükseklikte Asus Xtion'ın gezdirilmesiyle kaydedilmiştir. Bu kayıt içerisinde yapısal ve dokusal öğeler minimal düzeyde tutulmuştur. GFTT ve BRIEF algoritmalarının kombinasyonu bu veri kümesi içerisindeki RGB resimler ve DepthTiling tarafından yeniden renklendirilmiş resimler üzerinde uygulanmış, derinlik bilgisine sahip eşleşmiş öznitelik sayıları Şekil 6.6 ile grafikte gösterilmiştir. Bu veri kümesinde 377 farklı durum karşılaştırılmıştır. Bu veri kümesi üzerinde DepthTiling sisteminin çalıştırılmasıyla elde edilen sonuçları ($intervalSize = 0.1$, $stepSize = 64$) gösteren video dipnotta verilen linkte bulunabilir[1].



Şekil 6.6 freiburg3_nostructure_notexture_near_withloop veri kümesi için derinlik bilgisine sahip öznitelik eşlerinin sayısı

RGB-D kamera menzili içerisinde yeterli sayıda öznitelik bulunmadığında kapalı-döngü tespiti yapılamadığı bilinmektedir [3]. Öznitelik eşlerinin sayısı 50'nin atında olduğunda zayıf bilgi olarak kabul edilirken, 10'un altına düştüğünde odometri kaybı yaşanmaktadır. Tablo 6.1, farklı veri kümeleri için hem RGB hem de renklendirilmiş resimler kullanıldığında sırasıyla 50 ve 10'un altında öznitelik eşlemesinin yaşandığı durumları vermektedir.

DepthTiling ile yeniden renklendirilen resimler RGB resme altenatif olduğundan sistemin başarısı RGB resmin yetersiz kaldığı durumlarda önem kazanmaktadır. Bu nedenle Tablo 6.1 ile verilen RGB ve renklendirilmiş resimlerin aynı anda yetersiz öznitelik eşleşmesi içerdiği durumların sayısının düşük olması DepthTiling'in başarısının bir göstergesidir.

[1] `https://youtu.be/32rpoS1D13Y`

Tablo 6.1 Yetersiz öznitelik eşleşmesi içeren resim sayısı

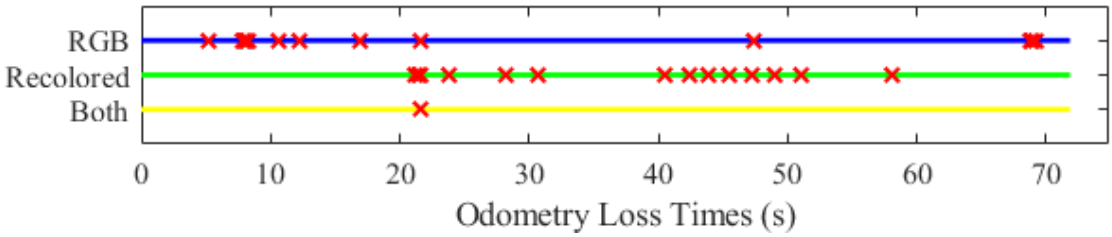
Veri Kümesi Adı	eşleşen_öznitelik < 50			eşleşen_öznitelik < 10		
	RGB	Depth-Tiling	Aynı Anda	RGB	Depth-Tiling	Aynı Anda
freiburg2_pioneer_360	113	94	3	17	15	1
freiburg2_pioneer_slam	106	48	13	35	19	7
freiburg2_pioneer_slam2	43	85	2	20	7	1
freiburg2_pioneer_slam3	73	18	2	8	13	0
freiburg3_nostructure_notexture_far	50	0	0	1	0	0
freiburg3_nostructure_notexture_near_withloop	134	0	0	9	0	0

Freiburg2_pioneer_360 veri kümesi robotun üzerine yerleştirilen Kinect'ten alınan verilerin 72 saniye boyunca kaydedilmesiyle oluşturulmuştur. Robot büyük bir alan içerisinde joystick ile 360 derece döndürülerek veri toplanmıştır. Bu veri kümesi üzerinde DepthTiling sisteminin çalıştırılmasıyla elde edilen sonuçları ($intervalSize = 0.2$, $stepSize = 64$) gösteren video dipnotta verilen linkte bulunabilir[2].

Şekil 6.7 ile verilen grafik, freiburg2_pioneer_360 veri kümesi için eşleşen öznitelik sayısının 10'dan az olduğu (tahmini odometri kaybının yaşandığı) zamanları göstermektedir. Grafikte, mavi, yeşil ve sarı çizgiler sırasıyla temsili SLAM uygulamasına giriş olarak RGB resim, DepthTiling ile renklendirilmiş resim ve her iki çeşit resim de alternatifli olarak verilmesini göstermektedir. Bu çizgiler üzerindeki kırmızı çarpılar ise eşleşen öznitelik sayısı 10'nun altına düştüğü için odometri kaybının yaşandığı tahmini zamanları göstermektedir. Her ne kadar RGB yerine renklendirilmiş resim kullanıldığında oluşan odometri kaybının yaşandığı durum sayısında kayda değer bir düşüş yaşanmasa da; Şekil 6.7, her iki resim alternatif olarak kullanıldığında büyük bir düşüşün beklenebileceğini açıkça göstermektedir.

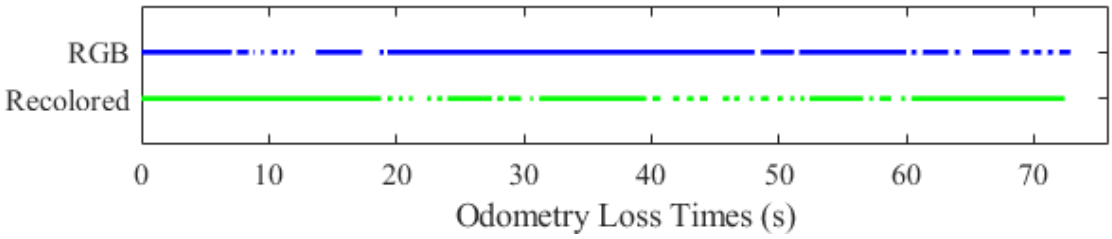
Şekil 6.8, sırasıyla freiburg2_pioneer_360 veri kümesi içerisindeki RGB resimleri ve bu resimlerin yeniden renklendirilmiş versiyonları üzerindeki RTAB-Map performansını vermektedir. Mavi ve yeşil çizgiler üzerindeki boşluklar, robot konumunun kaybedilmesi sonucu odometri sıfırlama işleminin gerçekleştirilmesi ihtiyacının olduğu durumları temsil etmektedir. Bu durumlarda, sistem tekrar yeterli sayıda öznitelik içeren yeni bir resim okuyana kadar SLAM algoritması uygulanamamaktadır.

[2] `https://youtu.be/aLA6V3OtanM`



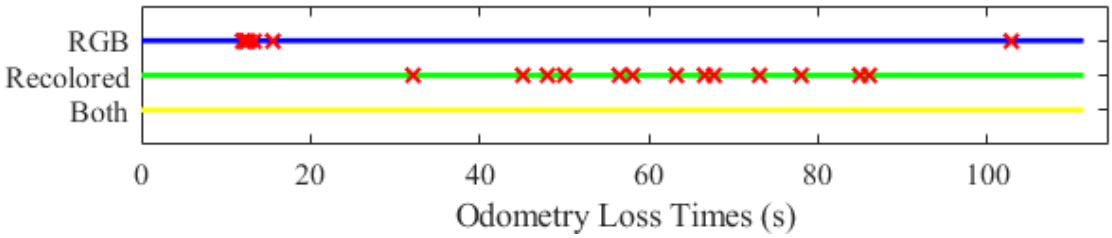
Şekil 6.7 freiburg2_pioneer_360 veri kümesi içerisinde sırasıyla RGB ve renklendirilmiş resimler ile aynı anda (her iki durum için de) eşleşen feature sayılarının 10'dan az olduğu zamanlar

Şekil 6.8 ile verilen grafikte, RTAB-Map sisteminin robotun konumunu kaybettiği ve odometrinin sıfırlanmasına ihtiyaç duyduğu zaman aralıkları Şekil 6.7 ile gösterilen 10 taneden az sayıda öznitelik içeren resimlerin bulunduğu zaman aralıklarına denk düşmektedir. Başka bir ifadeyle, Şekil 6.8, Şekil 6.7 ile verilen grafiği doğrulamaktadır.



Şekil 6.8 freiburg2_pioneer_360 veri kümesi içerisinde sırasıyla RGB ve renklendirilmiş resimler için RTAB-Map çalışma performansı

Şekil 6.9, freiburg2_pioneer_slam3 veri kümesi için eşleşen öznitelik sayısının 10'dan az olduğu tahmini odometri kayıp zamanlarını göstermektedir. Her ne kadar yeniden renklendirilmiş resimlerin kullanıldığı durumlarda, olası odometri kaybının yaşanma sayısında bir artış olsa da, iki giriş tipinin alternatif olarak kullanımıyla odometri kaybı yaşanmadan haritalama yapılması mümkün olabilmektedir.



Şekil 6.9 freiburg2_pioneer_slam3 veri kümesi içerisinde sırasıyla RGB ve renklendirilmiş resimler ile aynı anda (her iki durum için de) eşleşen feature sayılarının 10'dan az olduğu zamanlar

7
Sonuç ve Öneriler

Düşük maliyetli, 3 boyutlu sensörlerin yaygınlaşmasıyla literatürde oldukça başarılı 3B haritalama yapan SLAM sistemleri geliştirilmiştir [3–5]. Ancak bu sistemlerin genel sıkıntısı tek renk duvarlar, uzun koridorlar ya da yetersiz aydınlatma gibi durumların az sayıda öznitelik oluşmasına sebep olması dolayısıyla konum takibi ve haritalamayı zorlaştırmasıdır. 3 boyutlu haritalama yapılan ortamlar, genel olarak, RGB veriden çıkarılan öznitelik noktaları için uygun materyal içermektedir ancak haritalanan her ortam için öznitelik tespitinin sıkıntı oluşturacağı alanların bulunma ihtimali mevcuttur.

Tez kapsamında önerilen sistem olan DepthTiling ile böyle durumlarda oluşacak problemi minimuma indirmek için öznitelik tespitinde cisimlerin sadece renk bilgisinden değil aynı zamanda oluşturulan harita içerisindeki konumlarından da faydalanılmasına yönelik bir yaklaşım getirilmiştir. Bu yaklaşım ile RGB-D kameranın sağladığı derinlik verisinden kullanılarak odometri takibi ve haritalamada performansın arttırılması sağlanmıştır. Bunun için DepthTiling RGB-D kameradan alınan RGB resmi ilgili derinlik verisi ile yeniden renklendirmektedir. Böylece, bu tez çalışması ile RGB resmin 3B haritalama için yetersiz kaldığı durumlarda derinlik verisi yardımıyla kullanılabilir öznitelik sayısının arttırılabileceği gösterilmiştir.

Yapılan deneylerde, haritalama boyunca sadece DepthTiling tarafından üretilen resimlerin kullanılmasının performans açısından uygun olmayacağı gözlemlenmiştir. Bunun yerine, RGB verinin odometri takibi için yetersiz olduğu durumlarda DepthTiling'in alternatif olarak kullanılması hesaplama maliyeti ve performans ikilemi açısından en uygun durumu oluşturmaktadır. Geliştirilen sistem üzerinde yapılan testler neticesinde, görsel SLAM sistemlerinin sahip olduğu odometri takip kabiliyetinin arttırılması noktasında umut vadeden sonuçlar elde edilmiştir.

7.1 Öneriler

DepthTiling sistemi deneysel sonuçlarında yüksek başarı oranları elde etmiş olsa da en büyük dezavantajı veriye uyguladığı önişlem dolayısıyla oluşan gecikme sonucu gerçek zamanlı çalıştırılamıyor olmasıdır. Bu nedenle, sistemin daha hızlı ve efektif çalışabilmesi için optimizasyon yapılması gerekmektedir.

Optimizasyon adımı olarak kamera konum takibinde iyileştirme yapılması mümkündür. Öznitelik çıkarma algoritmalarının seçtiği noktalar RGB resim üzerinde ayırt edici özellik taşıyor olsa da güvenilir bir derinlik bilgisi taşıdıkları kesin olmamaktadır. Bu durumun giderilmesi için kullanılan GFTT [17] öznitelik çıkarma yaklaşımına, derinlik verisi maske olarak verilebilmektedir. Böylece sadece derinlik verisi bulunan alanlarda öznitelik çıkarımı yapılmaktadır. Ancak, bu maskeleme işlemi, bulunan öznitelik noktalarına ait derinlik verisinin güvenilir olmasını garantilememektedir. Bu nedenle, seçilen noktalara aşağıdaki 3 adımlık filtreleme işlemi uygulanarak derinlik verisi güvenilir olanlar ayırt edilebilir.

1. **Seçilen öznitelik noktası güvenilir derinlik menzili içerisinde olmalıdır.** Kinect menzilinin 4 metreye kadar olduğu föyünde belirtilmektedir. Ancak, tez kapsamında yapılan çalışmalar sırasında Kinect'ten okunan veriler incelendiğinde 7 metreyi geçen bilgi döndüğü gözlemlenmiştir. Bununla birlikte, verdiği derinlik bilgisi 0.4 - 4 m aralığının dışında olduğu zaman hata oranı arttığından bu aralık dışında kalan veriler güvenilir kabul edilmemektedir. Bu nedenle, sadece 0.4 - 4 m aralığında derinlik bilgisine sahip öznitelikler seçilmektedir.

2. **Seçilen öznitelik noktasının 8 komşusu da derinlik verisine sahip olmalıdır.** Eğer seçilen nokta, kameranın derinlik menziline yakın bir yerde ise içerdiği derinlik verisinin hata oranı yükselmektedir. Kamera kalibrasyonu ile bu hata oranı düşürülebiliyor olsa da sıfıra indirgenememektedir. Bu nedenle, sadece öznitelik noktasının kendisi değil etrafındaki noktaların da Kinect menzili içerisinde güvenilir derinlik bilgisi tutması beklenmektedir. Burada öznitelik noktasının sadece birinci dereceden yakınlıktaki komşularının kontrol edilmesi yerine, filtreleme için kullanılan çerçeve boyutunun ayarlanması ile önceden belirlenen bir yakınlık derecesine göre komşuların kontrolü yapılabilir.

3. **Seçilen öznitelik noktasının 8 komşusu ile derinlik farkı belirli bir eşik değerini aşmamalıdır.** Örneğin öznitelik noktası olarak bir dolabın köşesi seçildiği zaman dolap ile duvar arasında bir derinlik farkı oluşmaktadır. Bu derinlik farkının, derinlik resminde tam olarak doğru piksellerde gösteriliyor olması kamera iç ve dış kalibrasyonunun kusursuz yapılmış olmasına bağlıdır.

Bu da böyle durumlarda okunan verinin güvenilirliğini düşürmektedir. Bu nedenle, seçilen öznitelik noktalarının haritalanan ortamda nesnelerin sınırları üzerinde olmaması beklenmektedir.

Filtreleme sonrası elde edilen öznitelik sayısının 10'nun altında kalma ihtimali artmaktadır. Bu az sayıdaki öznitelik noktalarının sahip olduğu derinlik bilgisinin hata oranı filtreleme sayesinde en aza indirgenmiş olsa da mevcut görsel odometri sistemlerinde kullanılamamaktadır. Bu nedenle, çalışmalar daha az öznitelik ile odometri takibinin mümkün olup olmadığının araştırılması yönünde ilerletilebilir.

Kaynakça

[1] S.Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[2] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[3] M. Labbé and F. Michaud, "Long-term online multi-session graph-based splam with memory management," *Autonomous Robots*, vol. 42, no. 6, pp. 1133–1150, Aug. 2018, ISSN: 1573-7527. DOI: 10.1007/s10514-017-9682-5.

[4] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015. DOI: 10.1177/0278364914551008.

[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, Feb. 2014, ISSN: 1552-3098. DOI: 10.1109/TRO.2013.2279412.

[6] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, Oct. 1994, ISSN: 1573-1405. DOI: 10.1007/BF01427149.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992, ISSN: 0162-8828. DOI: 10.1109/34.121791.

[8] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Nov. 2011, pp. 155–160.

[9] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ıcp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013, ISSN: 1573-7527. DOI: 10.1007/s10514-013-9327-2.

[10] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast ıcp," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 3824–3829. DOI: 10.1109/IROS.2011.6094861.

[11] D. Lowe, "Distinctive ımage features from scale-ınvariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Apr. 2004, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sıft or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[14] R. Funayama, H. Yanagihara, L. V. Gool, T. Tuytelaars, and H. Bay, *Robust interest point detector and descriptor*, US Patent 8,165,401, Apr. 2012.

[15] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Sep. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[16] W. Changchang, *Siftgpu: A gpu ımplementation of scale ınvariant feature transform sıft*, 2013. [Online]. Available: http://cs.unc.edu/~ccwu/siftgpu (visited on 06/19/2017).

[17] I. Oliveira, E. Todt, and K. Fonseca, "Igftt: Towards an efficient alternative to sıft and surf," in *23rd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, May 2015. DOI: 10.13140/RG.2.1.1911.3048.

[18] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brıef: Binary robust independent elementary features," in *Computer Vision - ECCV 2010*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792, ISBN: 978-3-642-15561-1.

[19] T. Pire, T. Fischer, and J. Faigl, "Impact assessment of image feature extractors on the performance of slam systems," *Acta Polytechnica CTU Proceedings*, vol. 2, p. 45, Dec. 2015. DOI: 10.14311/APP.2015.1.0045.

[20] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV '03, Washington, DC, USA: IEEE Computer Society, 2003, pp. 1403–, ISBN: 0-7695-1950-4.

[21] H. Jin, P. Favaro, and S. Soatto, "Real-time 3d motion and structure of point features: A front-end system for vision-based control and interaction," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, 2000, pp. 778–779. DOI: 10.1109/CVPR.2000.854954.

[22] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range ımages," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA: ACM, 1996, pp. 303–312, ISBN: 0-89791-746-4. DOI: 10.1145/237170.237269.

[24] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012. DOI: 10.1177/0278364911434148.

[25] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ıcp," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, Jun. 2009. DOI: 10.15607/RSS.2009.V.021.

[26] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous : Spatially extended kinectfusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[27] D. Gálvez-López and J. D. TardLós, "Real-time loop detection with bags of binary words," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 51–58. DOI: 10.1109/IROS.2011.6094885.

[28] M. Kaess, A. Ranganathan, and F. Dellaert, "İsam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008, ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2006706.

[29] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 1691–1696. DOI: 10.1109/ICRA.2012.6225199.

[30] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to ımage analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[31] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision (ECCV)*, Sep. 2014.

[32] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*, Springer, 2000, pp. 298–372.

[33] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge, U.K.: Cambridge university press, 2004.

[34] R. Mur-Artal and J. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[35] R. Mur-Artal, J. Montiel, and J. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[36] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, Jun. 2013.

[37] ——, "Online global loop closure detection for large-scale multi-session graph-based slam," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2661–2666.

[38] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, vol. 2, Oct. 2003, pp. 1470–1477. DOI: 10.1109/ICCV.2003.1238663.

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383172.

[40] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2007, pp. 3472–3478.

[41] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3607–3613.

[42] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013, ISSN: 0929-5593. DOI: 10.1007/s10514-012-9321-0.

[43] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[44] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[45] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[46] A. Kleiner and C. Dornhege, "Real-time localization and elevation mapping within urban search and rescue scenarios," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 723–745, 2007.

[47] B. Mederos, L. Velho, and L. H. D. Figueiredo, "Moving least squares multiresolution surface approximation," in *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, Oct. 2003, pp. 19–26. DOI: 10.1109/SIBGRA.2003.1240987.

[48] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[49] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, Jun. 1987, ISSN: 0882-4967. DOI: 10.1109/JRA.1987.1087096.

[50] R. Chatila and J. Laumon, "Position referencing and consistent world modeling for mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, Apr. 1985, pp. 138–145. DOI: 10.1109/ROBOT.1985.1087373.

[51] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990. DOI: https://doi.org/10.1007/978-1-4613-8997-2_14.

[52] S. Thrun, "Exploring artificial ıntelligence in the new millennium," in, G. Lakemeyer and B. Nebel, Eds., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, ch. Robotic Mapping: A Survey, pp. 1–35, ISBN: 1-55860-811-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=779343.779345.

[53] G. Dissanayake, H. Durrant-Whyte, and T. Bailey, "A computationally efficient solution to the simultaneous localisation and map building (slam) problem," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, Apr. 2000, 1009–1014 vol.2. DOI: 10.1109/ROBOT.2000.844732.

[54] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3d mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, 2008.

[55] O. Wulf, K. O. Arras, H. I. Christensen, and B. Wagner, "2d mapping of cluttered indoor environments by means of 3d perception," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 4, Apr. 2004, pp. 4204–4209. DOI: 10.1109/ROBOT.2004.1308934.

[56] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg, "Automated 3d reconstruction of interiors with multiple scan views," *Proceedings of SPIE - The International Society for Optical Engineering*, Dec. 1998. DOI: 10.1117/12.333775.

[57] S. Yavuz, M. F. Amasyalı, E. Uslu, F. Çakmak, and N. Altuntaş, *Ytu ce probabilistic robotics group*, 2006. [Online]. Available: http://www.robotics.yildiz.edu.tr/ (visited on 06/19/2019).

[58] H. L. Akin, N. Ito, A. Jacoff, A. Kleiner, J. Pellenz, and A. Visser, "Robocup rescue robot and simulation leagues," *AI magazine*, vol. 34, no. 1, pp. 78–86, 2013. DOI: https://doi.org/10.1609/aimag.v34i1.2458.

[59] C. Rockey, *Depthimage_to_laserscan - ros wiki*, 2011. [Online]. Available: http://wiki.ros.org/depthimage_to_laserscan (visited on 05/30/2019).

[60] J. Bowman and P. Mihelich, *Camera_calibration - ros wiki*, 2009. [Online]. Available: http://wiki.ros.org/camera_calibration (visited on 06/12/2019).

[61] N. Altuntaş, F. Çakmak, E. Uslu, M. Balcılar, M. F. Amasyalı, and S. Yavuz, "3 dimensional thermal mapping," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, May 2016, pp. 1201–1204. DOI: 10.1109/SIU.2016.7495961.

[62] S. Vidas, P. Moghadam, and S. Sridharan, "Real-time mobile 3d temperature mapping," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1145–1152, Feb. 2015, ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2360709.

[63] M. D. Bond, N. Nethercote, S. W. Kent, S. Guyer, and K. S. McKinley, "Tracking bad apples: Reporting the origin of null and undefined value errors," in *ACM SIGPLAN Notices*, vol. 42, Oct. 2007, pp. 405–422. DOI: 10.1145/1297027.1297057.

[64] TheQtCompany, *Qt creator manual*, 2007. [Online]. Available: https://doc.qt.io/qtcreator/index.html (visited on 07/14/2019).

[65] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

[66] N. Altuntaş, E. Uslu, F. Çakmak, M. F. Amasyalı, and S. Yavuz, "Comparison of 3-dimensional slam systems: Rtab-map vs. kintinuous," in *2017 International Conference on Computer Science and Engineering (UBMK)*, Oct. 2017, pp. 99–103. DOI: 10.1109/UBMK.2017.8093567.

[67] M. Labbé, *Rtabmap_ros - ros wiki*, 2011. [Online]. Available: http://wiki.ros.org/rtabmap_ros#rtabmapviz (visited on 10/30/2019).

[68] O. Guclu and A. B. Can, "A comparison of feature detectors and descriptors in rgb-d slam methods," in *Image Analysis and Recognition*, M. Kamel and A. Campilho, Eds., Cham: Springer International Publishing, 2015, pp. 297–305, ISBN: 978-3-319-20801-5.

[69] I. I. Y. Saito, *Openni_launch - ros wiki*, 2012. [Online]. Available: http://wiki.ros.org/openni_launch (visited on 10/30/2019).

[70] J. O'Quin, *Freenect_launch - ros wiki*, 2012. [Online]. Available: http://wiki.ros.org/freenect_launch (visited on 10/30/2019).

[71] R. Haschke, *Rviz - ros wiki*, 2009. [Online]. Available: http://wiki.ros.org/rviz (visited on 10/30/2019).

A
RTAB-Map Kurulum ve Kullanımı

Tez kapsamında yapılan çalışmalarda Linux tabanlı Ubuntu 16.04 LTS işletim sistemi kllanılmıştır. Bilgisayarın robotla olan iletişimini sağlamak için ise ROS platformunun Kinetic versiyonu kullanılmıştır. ROS, farklı robot çeşitleri ile çalışmayı kolaylaştıran ve robot yazılımı için geliştirilmiş bir platformdur. Önerilen DepthTiling sistemi bu platform üzerinde geliştirilmiştir. Haritalama başarısının test edilmesi için ise RTAB-Map kütüphanesi kullanılmıştır.

Bölüm A.1, ilk olarak, Ubuntu 16.04 işletim sistemi üzerine ROS Kinetic kurulumunu anlatmaktadır. Sonrasında, Bölüm A.2 ve A.3 ile ROS paketi mevcut olan RTAB-Map programının sırasıyla kurulum ve kullanım detayları verilmektedir.

A.1 ROS Kurulumu

ROS çeşitli linux tabanlı platformlarda çalışmaya uygun olsa da geliştirilen her bir versiyon için kullanılması gereken işletim sistemleri belirtilmektedir. Bu nedenle tez kapsamında, Ubuntu 16.04 üzerine yüklenmesi tavsiye edilen ROS versiyonu Kinetic ile çalışılmıştır. Ubuntu 16.04 üzerinde ROS Kinetic kurulumu için terminalde çalıştırılması gereken komut adımları aşağıda verilmiştir.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

sudo apt-get update

sudo apt-get install ros-kinetic-desktop-full

sudo rosdep init

rosdep update
```

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

source ~/.bashrc

sudo apt-get install python-rosinstall
```

A.2 RTAB-Map Kurulum

ROS Kinetic için rtabmap_ros sistem dosyaları aşağıdaki kod kullanılarak indirilebilmektedir. Sistem dosyaları inerken gerekli kütüphaneler de yüklenmektedir.

```
sudo apt-get install ros-kinetic-rtabmap-ros
```

Ancak kaynak kodu da indirerek son versiyonu kullanmak için RTAB-Map'in aşağıda verilen kod kullanılarak yüklenmesi gerekmektedir.

```
(RTAB-Map Bağımsız Kütüphanelerinin Yükleme)

cd ~

git clone https://github.com/introlab/rtabmap.git rtabmap

cd rtabmap/build

cmake -DCMAKE_INSTALL_PREFIX=~/catkin_ws/devel .. [<— double dots included]

make -j4

make install


(RTAB-Map ROS Paketinin Yükleme)

cd ~/catkin_ws

git clone https://github.com/introlab/rtabmap_ros.git src/rtabmap_ros

catkin_make
```

RTAB-Map sistemini güncellemek için ise çalıştırılacak kod aşağıda verilmiştir.

```
(RTAB-Map Güncelleme)

cd rtabmap

git pull origin master
```

```
cd build

make

make install


(RTAB-Map ROS Paketi Güncelleme)

roscd rtabmap_ros

git pull origin master

cd ~/catkin_ws

catkin_make
```

A.3 RTAB-Map Kullanım

ROS ile kullanımında Kinect kameradan veri alımını sağlamak için openni_launch [69] veya freenect_launch [70] kütüphanelerinin birinin başlatma (launch) dosyası aşağıdaki kodlardan biri yardımıyla çalıştırılmalıdır. Burada dikkat edilmesi gereken konu RTAB-Map RGB ve derinlik bilgisinin birleştirildiği veriyi kullandığından ilgili parametrenin yazılması gerekmesidir.

```
roslaunch openni_launch openni.launch depth_registration:=true
```

veya

```
roslaunch freenect_launch freenect.launch depth_registration:=true
```

Haritalama sırasında RTAB-Map sonuçlarını ROS üzerinden görüntüleme için iki farklı seçenek bulunmaktadır: RViz [71] ve RTABMapViz [67].

RViz ROS için 3 boyutlu görselleştirme sağlayan bir uygulamadır. ROS platformu içerisinde çalıştırılan sistemler ve robotlar iletişimlerini mesaj yoluyla sağlamaktadır. RViz gönderilen bu mesajların görselleştirilmesinde kullanılmaktadır. RTAB-Map tarafından yayınlanan mesajların RViz'de gösterimi için bir eklenti geliştirilmiştir. Böylece oluşturulan harita direkt gözlemlenebilmektedir. Aşağıda verilen kod haritalama için RTAB-Map'i, görselleştirme için ise RViz'i çalıştırmaktadır.

```
roslaunch rtabmap_ros rgbd_mapping.launch rtabmap_args:="--delete_db_
on_start" rviz:=true rtabmapviz:=false
```

Varsayılan olan RTABMapViz ise RTAB-Map tarafından görselleştirme için geliştirilen RViz'e benzer bir arayüzdür. Amaca yönelik olarak geliştirildiğinden programın parametrik özelliklerinin RViz'e göre daha iyi kullanılmasını sağlamaktadır. Aşağıda verilen kod RTAB-Map'i varsaılan arayüzü olan RTABMapViz ile çalıştırmaktadır.

```
roslaunch rtabmap_ros rgbd_mapping.launch rtabmap_args:="--delete_db_
on_start"
```

Yukarıda RTAB-Map'in çalıştırılması için verilen kodlarda dikkat edilmesi gereken bir diğer parametre ise "--delete_db_on_start"dır. Bu parametre sisteme verildiğinde eski haritayı tutan veritabanı silinir ve yeni haritalama başlar. Sisteme verilmediğinde ise haritalama veritabanında bulunan mevcut haritanın üzerinde devam niteliğinde yapılır. Veri tabanını temizlemenin bir başka yolu da RTABMapViz üzerindeki menüden "Edit → Delete Memory" ile silmek ya da aşağıdaki kodu terminalden çağırmaktır.

```
rosservice call /rtabmap/reset
```

ROS ile bağlantılı olarak çalıştırılan sistem veri tabanını ∼/.ros/rtabmap.db adresine kaydetmektedir. Aşağıdaki kod ile açılan arayüzde haritalama sırasında veritabanına kaydedilen tüm verilerin teker teker analizi yapılabildiği gibi bu verilerin 3 boyutlu haritasını ve TORO grafik yapısını da çıkarmak mümkündür.

```
rtabmap-databaseViewer <veritabanı_adı>
```

Sistemi ROS'dan bağımsız olarak çalıştırmak da mümkündür. Aşağıdaki kod ile çalıştırılabilen bu seçenekte, arayüz menüsünden "Detection → Select Source" menüsü altıda verinin hangi kaynaktan alınacağı (örneğin Kinect) seçilmelidir. Ayrıca veritabanı ayarının da yapılması gerekmektedir. Yine yukarıdaki menüden yeni bir veritabanı oluşturulabileceği gibi eskiden kaydedilmiş bir tane de seçilebilir. Burada veritabanları ROS ile çalıştırılmasından farklı olarak kurulum sırasında oluşturduğu ∼/Documents/RTAB-Map/ klasörünün altına kaydedilmektedir.

```
rtabmap
```

Görsel odometrinin kullanıldığı sistemde, aşağıdaki nedenlerden biri dolayısıyla odometri takibi mümkün olmadığında arka plan kırmızıya boyanmaktadır.

- Boş duvar ya da karanlık alanlar gibi öznitelik barındırmayan ortamlardan veri alındığında
- Kinect etki alanı olan 0.4 - 4 m aralığında boş olan ortamlara bakıldığında
- RGB-D kamera hızlı hareket ettirildiğinde
- Derinlik ve RGB verisi üzerinde yanlış eşleme olduğunda (Bu durumu gidermek için kamera kalibrasyonu yapılması gerekmektedir.)

Odometrinin tekrar bulunabilmesi için RGB-D kamerayı odometri takibinin ilk kaybolduğu yere kadar geri götürmek gerekmektedir. Böylece WM'de bulunan konum bilgileriyle bir eşleme yapılarak tekrar odometri takibi yapılabilmektedir (bkz. Bölüm 1.2.1). Bir başka yol ise odometri takibini sıfırlamaktır. Bu durumda RGB-D kamera öznitelik içeren bir alandan veri alırken sıfırlama işlemi arayüzün "Detection → Reset Odometry" menüsü ya da ROS'da aşağıdaki kod kullanılarak yapılabilmektedir.

```
rosservice call /reset_odom
```

Odometri sıfırlandığında sistem aynı veritabanı üzerinde yeni bir harita oluşturup onun üzerinde işlem yapmaktadr. Diğer harita ile arasında bir kapalı-döngü tespit edildiğinde ise iki harita birleştirilmektedir.

B
Tezde Kullanılan Terimler Sözlüğü

Bu ekte tez içerisinde kullanılan terimlerin İngilizce - Türkçe ve Türkçe - İngilizce karşılıkları yer almaktadır.

B.1 İngilizce - Türkçe

6 degree of freedom	6 serbestlik derecesi
accuracy	doğruluk
array	dizi
benchmark	veri koleksiyonu
binary	ikili
correction	düzeltme
dataset	veri kümesi
default	varsayılan
descripter	tanımlayıcı, betimleyici
description	açıklama
dimension	boyut
extraction	çıkarma
feature	öznitelik
float	kesirli
frame	çerçeve
framework	uygulama iskeleti

hardware	donanım
input	giriş
launch	başlatma
localization	lokalizasyon, konumlandırma
loop-closure	kapalı-döngü
matching	eşleme
navigation	gezinim
noise	gürültü
observation	gözlem
occupancy	doluluk
odometry	odometri, konum
online	gerçek zamanlı
optimization	optimizasyon
output	çıkış
perception	algılama
plug-in	eklenti
point cloud	nokta bulutu
prediction	tahmin
process	işlem
range	menzil
reset	sıfırlama
resolution	çözünürlük
response	tepki (değeri)
retrieval	getirme
set	küme
software	yazılım

state	durum
texture	doku
transition	geçiş
visual	görsel

B.2 Türkçe - İngilizce

6 serbestlik derecesi	6 degree of freedom
açıklama	description
algılama	perception
başlatma	launch
boyut	dimension
çerçeve	frame
çıkarma	extraction
çıkış	output
çözünürlük	resolution
dizi	array
doğruluk	accuracy
doku	texture
doluluk	occupancy
donanım	hardware
durum	state
düzeltme	correction
eklenti	plug-in
eşleme	matching
geçiş	transition
gerçek zamanlı	online

getirme	retrieval
gezinim	navigation
giriş	input
görsel	visual
gözlem	observation
gürültü	noise
ikili	binary
işlem	process
kapalı-döngü	loop-closure
kesirli	float
konum	odometry
konumlandırma	localization
küme	set
lokalizasyon	localization
menzil	range
nokta bulutu	point cloud
odometri	odometry
optimizasyon	optimization
öznitelik	feature
sıfırlama	reset
tahmin	prediction
tanımlayıcı	descripter
tepki	response
uygulama iskeleti	framework
varsayılan	default
veri koleksiyonu	benchmark

veri kümesi	dataset
yazılım	software

Tezden Üretilmiş Yayınlar

İletişim Bilgileri: nihaltuntas@gmail.com

Makale

1. N. Altuntaş, M. F. Amasyalı, "DepthTiling: A Novel Way to Increase Visual SLAM Performance in Featureless Environments", *Electronics Letters*, vol. 55, no. 25, pp. 1338-1340, Dec. 2019, doi: 10.1049/el.2019.1384, print issn: 0013-5194, online issn: 1350-911X.

Konferans Bildirisi

1. N. Altuntaş, F. Çakmak, E. Uslu, M. Balcılar, M. F. Amasyalı ve S. Yavuz, "3 Dimensional Thermal Mapping", *24th Signal Processing and Communications and Applications Conference (SIU)*, Zonguldak, TÜRKIYE, 2016, pp. 1201-1204. doi: 10.1109/SIU.2016.7495961.

2. N. Altuntaş, E. Uslu, F. Çakmak, M. F. Amasyalı ve S. Yavuz, "Comparison of 3-Dimensional SLAM Systems: RTAB-Map vs. Kintinuous", *International Conference on Computer Science and Engineering (UBMK)*, Antalya, TÜRKIYE, 5-8 Ekim 2017, pp. 99-103.

Proje

1. "Otonom Arama Kurtarma Robotları için Keşif, Haritalama ve Afetzede Tespit Algoritmalarının Geliştirilmesi", *TÜBİTAK 1001*, EEEAG-113E212, 2013-2016, Bursiyer.

2. "Kısıtlı İletişime Sahip Robot Takımları için Otonom Keşif Algoritmalarının Tasarlanması", *YTÜ BAP Normal Araştırma Projesi*, 2015-04-01-KAP01, 2015-2017, Araştırmacı.

3. "Gerçek Robot Platformu ile Tasarlanmış ve Tasarlanmamış Engebeli Alanda Otonom Gezinim", *YTÜ BAP Kariyer Geliştirme Projesi*, 2015-04-01-GEP01, 2015-2018, Araştırmacı.

4. "Afet robotları için Navigasyon ve Keşif Algoritmalarının Tasarımı", *YTÜ BAP Tez Projesi, Doktora*, FDK-2017-3125, 2017-2020, Araştırmacı.

Ödül

1. "Rescue Robot League Best in Class Small Robot Exploration", *NIST - RoboCup*, Leipzig Almanya, Temmuz 2016.

2. "Rescue Robot League Best in Class Autonomy", *NIST - RoboCup*, Nagoya Japoya, Temmuz 2017.

3. "Rescue Robot League Best in Class Exploration", *NIST - RoboCup*, Nagoya Japoya, Temmuz 2017.

4. "Rescue Robot League Best Outdoor CarryBot", *NIST - RoboCup*, Nagoya Japoya, Temmuz 2017.

5. "Rescue Robot League Best in Class Small Robot Exploration", *NIST - RoboCup*, Nagoya Japoya, Temmuz 2017.