KIRGIZİSTAN TÜRKİYE MANAS ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

MARKOV MODELİNİ KULLANARAK ROBOTLARIN YERİNİ KEŞFEDEN ALGORİTMA

Hazırlayan
Ulan BAYALİEV

Danışman
Prof. Dr. Ulan BRİMKULOV

Yüksek Lisans Tezi

Nisan 2015
KIRGIZİSTAN/BİŞKEK

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Ulan BAYALİEV

İmza :

YÖNERGEYE UYGUNLUK

"Markov Modelini Kullanarak Robotların Yerini Keşfeden Algoritma" adlı Yüksek Lisans Tezi, Kırgızistan Türkiye Manas Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan	Danışman
Ulan BAYALİEV	Prof.Dr. Ulan BRİMKULOV
İmza:	İmza:

Bilgisayar Mühendisliği ABD Başkanı

Doç.Dr. Rayimbek SULTANOV

İmza:

Prof. Dr. Ulan BRİMKULOV danışmanlığında Ulan BAYALİEV tarafından hazırlanan "Markov Modelini Kullanarak Robotların Yerini Keşfeden Algoritma" adlı bu çalışma, jürimiz tarafından Kırgızistan Türkiye Manas Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında **Yüksek Lisans tezi** olarak kabul edilmiştir.

…….. /…… / ……..

(Tez savunma sınav tarihi yazılacaktır.)

JÜRİ:

Danışman	: Prof. Dr. Ulan BRİMKULOV	…….
Üye	:…….	…….
Üye	:…….	…….
Üye	:…….	…….
Üye	:…….	…….

ONAY:
Bu tezin kabulü Enstitü Yönetim Kurulunun ……...… tarih ve ……..…… sayılı kararı ile onaylanmıştır.

……. /……../ ……

Prof. Dr. Zafer GÖNÜLALAN

Fen Bilimleri Enstitü Müdürü

ÖNSÖZ / TEŞEKKÜR

Çalışmalarım boyunca farklı bakış açıları ve bilimsel katkılarıyla beni aydınlatan, yakın ilgi ve yardımlarını esirgemeyen ve bu günlere gelmemde en büyük katkı sahibi sayın hocam Prof. Dr. Ulan BRİMKULOV'a teşekkürü bir borç bilirim.

Deneysel çalışmalarım sırasında karşılaştığım zorlukları aşmamda yardımlarından ve desteklerinden dolayı sayın Doç. Dr. Rayımbek SULTANOV ve bütün hocalarıma teşekkür ederim.

Ulan BAYALİEV

Bişkek, Nisan 2015

MARKOV MODELİNİ KULLANARAK ROBOTLARIN YERİNİ KEŞFEDEN ALGORİTMA

Ulan BAYALİEV

Kırgızistan Türkiye Manas Üniversitesi, Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi, Nisan 2015

Danışman: Prof. Dr. Ulan BRİMKULOV

KISA ÖZET

Robotları belli bir ortamda başarılı yönetebilmek için once robotun ortamdaki yeri ve ortamın haritası bilinmelidir. Robotun yerini bilmek için GPS sinyalleri kullanılabilirdi,

yalnız GPS'in hata oranı 5-10 metre olabiliyor ve sinyali kapalı ortamlarda alınamıyor.

Bu nedenlerden dolayı robotlar değişik sensörler aracılığı ile alınan bilgileri inceleyerek verilen ortamın haritasında kendi yerleri konusunda belli bir sonuca varmaları gerekmektedir. Robotlar bulunduğu yeri algılamak için ultrason, görüntü veya kızılötesi sensörler kullanılabilir. Fakat bu sensörler robotun konumunu tam olarak saptayamayacaktır ve yalnızca robotun ortamı ile ilgili bilgiler vermektedir. Robot bu verileri kullanarak bulunduğu konumu hakkında sadece belli inanca ulaşacaktır, yani kendi konumu hakkında kesin bir bilgiye ulaşamadığından dolayı en iyi tahmini yapmak için stokastik yöntemleri kullanması gerekir. Modern robot biliminde konum hakkında belli bir bilgiye ulaşmak için kullanılan stokastik yöntemler "Yer keşfetme algoritmaları" diye adlındırılır.

Tezin amacı, Markov modelini kullanan stokastik yer keşfetme algoritmasını incelemek ve başka yer keşetme algoritmaları ile karşılaştırmaktır. Bunu yapmak için görsel

uygulama üzerinde robotun ortamı, davranışı ve algılama kabiliyeti taklit edilmektedir ve algoritmanın çalışması gösterilmektedir. Uygulamanın başında robot, haritasını bildiği fakat bulunduğu konumu bilmediği bir ortama konuluyor. Sonuç olarak, robot bir kaç hareket sonrasında kendi konumu hakkında belli bir bilgiye ulaşmaktadır.

Anahtar Kelimeler: Robotlar, Yer keşfetme, Sensör, Stokastik yöntemler, Markov modeli.

МАРКОВ МОДЕЛИН КОЛДОНУУ АРКЫЛУУ РОБОТТОРДУН ОРУН ТАБУУ АЛГОРИТМАСЫ

Улан Баялиев

Кыргыз-Турк Манас Университети, Табигий илимдер институту

Магистрдык иш, чын куран2015

Илимий жетекчи: Проф. Док. Улан Бримкулов

Кеңири анотация

Роботторду белгилүү чөйрөдө башкарыш үчүн башында роботтун чөйрөдөгү ордуну жана айлана чөйрөнүн картасы билиниш керек. Роботтун ордун билиш үчүн GPS (Глобалдын позициондук система) колдонулса болот эле бирок GPS'тин катасы 5-10 метр болушу мүмкүн жана GPS сигналы жабык имараттарда алынбайт. Бул себептерден роботтор ар түрдүү сенсорлордон алынган маалыматтарды изилдеп чөйрөдөгү өзүнүн орду жөнүндө белгилүү жыйынтык чыгарышы керек. Роботтор болгон жери жөнүндө маалымат жыйноо үчүн ультраүн, сүрөт жана кызыл ды ашуу сенсорлорду колдонушу мүнкүн. Бирок бул сенсорлор роботтун так ордун биле албайт жана роботтун чөйрөсү жөнүндө гана маалымат берет. Робот бул маалыматтарды колдонуп болгон жери жөнүндө бир ишенишке ээ болот, демек робот өзүнүн орду жөнүндө так маалыматка ээ болбогондуктан жери жөнүндө эң туура ищеним табыш үчүн стохастикалык методторду колдонушу керек. Заманбап робот билиминде роботтун орду жөнүндө бир маалыматка жетиш үчүн колдонулган стохостикалык методторго "Орун табуу" алгоритмдери деп аталат.

Диссертациянын максаты, Марков моделин колдонгон стохастикалык орун табуу алгоритмини изилдеп көрсөтүп аны башка орун табуу алгоритмдери менен салыштыруу. Бул нерсени кылыш үчүн визуалдык программа үстүндө роботтун айлана чөйрөсү, аракет жылышы жана маалымат топтоо жөндөмдүүлүгү моделденип алгоритмдердин кантип иштегени көрсөтүлөт. Программа иштеп баштаганда робот картасын билген бирок өзүнүн ордун так билбеген чөйрөгө коюлат. Программа иштеп чыккандан кийин робот өзүнүн ордун табат.

Роботтордун орун табуу проблемалардын бир нече түрү бар. Эң түптүү орун табу проблемасы “Орун көзөмөлдөө” болуп эсептелет. Роботтун баш тапкы орду билинип бир нече кадамдан кийин роботтун ордун аныктоо “Орун көзөмөлдөө” проблемасы деп аталат. Андан дагы роботтун “Глобалдык орун табуу” проблемасы кыйын болуп эсептелет, эмне дегенде бул проблемада роботтун баш тапкы орду билинбейт жана роботтун сенсор жана арекетинде ката болушу мүмкүн. Бул кезде робот өзүнүн кайда болушу жөнүндү анык маалыматы жок болгондугундан оду жөнүндө башкача ищениш түзүшү керек. Жогорудагы проблемаларды чечиш үчүн бир нече алгоритм бар [1]. Мисалы “Орун көзөмөлдөө” проблемасын чечиш үчүн Калман фильтрин колдонсо болот [54]. Калман фильтр алгоритминде роботтун орду жөнүндөгү ишениш жана ыктымалдыгы Гаусс үлөщү менен көрсөтүлөт жана проблеманы чечиш үчүн одометр информациясы колдонулат [68].

Бул диссертацияда “Глобалдык Орун Табуу” проблемасын чечиш үчүн

Марков моделин колдонгон алгоритм изилденип башкача орун табуу алгоритмдери менен салыштырылат. Марков моделин колдонгон алгоритмде робот чөйрөдөгү болгон бүт орундар үчүн ыктымалдыкты эсинде кармайт. Мисалы робот башында аракет кылып жылып баштай электе өзүнүн орду жөнүндө маалыматка ээ болбогондуктан бүт орундар үчүн ыктымалдык бир болот. Робот бир нече аракет кылып сенсордон чөйрөсү жөнүндө маалымат чогулткандан кийин картадагы орундар үчүн ыктымалдык өзгөрөт. Эмне дегенде робот картадагы орундар жөнүндө маалыматы болот. Ал маалыматты колдонуп робот кее бир орундарда ыктымалдыкты көтөрүп кее бир жерлерде түшүрөт. Бул диссертацияда робот ыктымалдыктарды эсептеш үчүн кандай алгоритм колдонулганын баяндайт. Андан соң кее бир орундун ыктымалдыгы башка орундарга карата эң чон баага тең болгондо робот өзүнүн ордун тапкан болуп эсептелет. Марков моделинин проблеманы чечиш үчүн стохастикалык методу роботтун орду так болбогон кезде артыкчылыт болуп эсептелет. Бул нече стохастикалык методтор ар дайым роботторго артыкчылык берет, эмне дегенде автономиялык роботтордогу сенсор жана арекеттериндеги каталар стохастикалык методторду керектирет. Алгоритмдеги колдонулган картаны дискретизация жана эң кичине бөлүктөргө бөлүү ар бир ордун үчүн ыктымалдыкты эске алышты мумкун кылат. Бөлөкчө бул дискретизация методу роботтун аракеттеринден кийин ар бир ордунду көзгө алууну мүмкүнчүлүк берет. Бундан башка роботтун сенсор маалыматтарын бүт жерде колдонууга

уруксат берет. Топологиялык дискретизация картадагы жерлерди өөрөндөргө бөлүп ар бир өөрөндүн ыктымалдыгын эсептейт. Мисалы “Эшиктин жанынды” же болсо “Коридордо” деген орундар үчүн ыктымалдыктар эсептелет. Бул методто эгерде робот бир өөрөндүн ичинде эки башка позицияда болсо дагы алгоритм роботтун ордун бир деп эсептейт [35]. Орун табуу үчүн визуалдык алгоритмдер дагы көп [45]. Визуалдык алгоритмдерде роботко көрүнгөн сүрөт картанын сүрөтү менен салыштырылып роботтун орду табылат. Бул алгоритмдер көп эсептөө кылгандыгындан тез арада иштеп чыгышы мүмкүнчүлүктөрү жок болот [50].

Бул диссертацияда роботтордун ордун табуу үчүн колдонулган негизги үч алгоритм изилденет. Марков алгоритми, Калман Фильтры жана Монте-Карло алгоритми. Бул алгоритмдердин ар биринин кандай иштегени программа үстүндө көрсөтүлүп артыкчылыктары жана жетишпегендиктери көрсөтүлөт. Бул алгоритмдер негизинде автономиялык жана мобильдик роботтор үчүн стохастикалык методторун колдонуусу зарыл болгону көрсөтүлүп, бул диссертация мындай роботтордун орун табуу алгоритмдердин маанисин аныктап далилдейт.

Ачкыч сөздөр: Роботтор, Орун табуу, Сенсор, Стохастикалык методтор, Марков модели.

АЛГОРИТМ ДЛЯ МЕСТОНАХОЖДЕНИЯ РОБОТОВ ИСПОЛЬЗУЮЩИЙ МОДЕЛЬ МАРКОВА

Улан Баялиев

Кыргызско-Турецкий Университет Манас, Институт Естественных наук

Магистерская работа, Апрель 2015

Научный руководитель: Профессор, Доктор Улан Бримкулов

Для успешного управления роботами в определенной среде необходима информация о местонахождении робота и карта окружающей среды. Чтобы узнать местонахождение робота можно было бы использовать сигнал GPS, но ошибка GPS можеть составлять 5-10 метров и сигнал нельзя поймать в закрытых помещениях. Из-за этих причин роботы должны исследовать полученные данные от разных сенсоров и сделать вывод о своем метонахождении на карте окружающей среды. Чтобы познать окружающую среду робот может использовать ультразвуковые, визуальные или инфракрасные сенсоры. Но эти сенсоры не могут определить местонахождение робота и только дают информацию об окружающей среде робота. Используя эту информацию робот может только делать предположения о своем местонахождении, и так как робот не может точно сказать где находится, он должен использовать стохастические методы чтобы сделать самый правильный вывод. В современной науке робототехники такие стохоастические методы для определения позиции робота называются алгоритмами для

местонахождения.

Цель тезиса изучить алгоритм для местонахождения робота использующий модель Маркова и сравнить его с другими алгоритмами. Чтобы достичь этого, работа алгоритма демонстрируется на визуальном проиложении в котором симулируется среда, действия и восприятие робота. В начале приложения робот ставится в неизвестном ему месте но в известной ему среде. В результате робот делает вывод о своем местонахождении после несколько действий.

Ключевые слова: Роботы, Локализация, Сенсор, Стохастические методы, модель Маркова.

ALGORITHM FOR LOCATING ROBOT POSITION USING MARKOV MODEL

Ulan BAYALİEV

Kyrgyzstan-Turkey Manas University, Institute of Natural and Applied Sciences

M.Sc. Thesis, April 2015

Supervisor: Assoc. Prof. Dr Ulan BRİMKULOV

To successfully manage robots in some environment we need to know his position and map of the place beforehand. We could use GPS for acquiring the position of the robot it's error rate is about 5-10 meters and GPS signal cannot be received indoors. That's why robots should analyze their environment using different sensors and make inference about their current position on the map of the place. Robots may use ultrasound, vision or infrared sensors to perceive their environment. But these sensors do not locate robot's position and give only information about robots environment. Robot will only infer some belief about its curent position using this information, and since robot can not know its position for sure it should use stochastic methods to make the best guess. In modern robotics science these kind of stochastic methods used for locating the position are called “Localization algorithms”.

The goal of this thesis is analyzing the stochastic localization algorithm which uses Markov model and compare it with other localization algorithms. To achive this goal, robot perception, behaviour and environment are simulated using visual application for

demonstrating the execution of the algorithm. At the beginning the robot will be placed in a place where it knows the map of the environment but does not know the position of itself. As a result robot will reach certain belief about its position after moving itself some steps.

Keywords: Robots, Localization, Sensor, Stochastic methods, Markov model

İÇİNDEKİLER

MARKOV MODELİNİ KULLANARAK ROBOTLARIN YERİNİ KEŞFEDEN ALGORİTMA

Sayfa

3. BÖLÜM

MARKOV YER KEŞFETME ALGORITMASI

4. BÖLÜM

KALMAN FİLTRESİ

5. BÖLÜM

MONTE-CARLO ALGORİTMASI

6. BÖLÜM

ALGORİTMALARIN KARŞILAŞTIRILMASI

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1. GİRİŞ VE AMAÇ

Robotların yer keşfetme problemi robotun ortamdaki kendi konumunu bulmaya yönelik çalışmasıdır. Yer keşfetme kabiliyeti otonom robotlar için önemli rol oynar çünkü robotun başarılı olabilmesi için konumu bilinmelidir. Ayrica bu kabiliyet robotun tam otonom olabilmesi için en önemli şartlardandır [1].

Robot yer keşfetme probleminin birkaç türü vardır. En temel yer keşfetme problemi 'Konum takip etme'dir. Bu durumda robotun başlangıç konumu bilinmektedir ve bir kaç hareket sonrasında robotun konumunu kesinleştirme 'Konum takip etme' problemidir.

Robotun yer keşfetme problemi ise daha zordur, çünkü bu problemde robotun başlangıçtaki konumu bilinmemektedir ve ayrıca robotun hareket ve sensörlerinde hata olabilir. Bu durumda robot kendisinin nerede konumlandığı hakkında kesin bilgisi yoktur ve kendi konumu hakkında farklı tahmin ve ınanç oluşturması gerekiyor. 'Kaçırılmış robot' problemi ise çözülmesi daha zordur [37]. Bu problemde robot bildiği ortam ve konumdan 'kaçırılarak' başka konuma yerleştiriliyor. 'Kaçırılmış robot' probleminin 'Yer keşfetme' probleminden farkı robot kendisinin farklı bir konumda olduğunu zannetmesidir. 'Yer keşfetme' probleminde ise robotun kendi bilgisizliği hakkında haberdardır.

Yukarıdaki problemleri çözmek için çok algoritma mevcuttur [38]. Örnek olarak 'Konum takip etme' problemini çözmek için Kalman filtresi ve Geliştirilmiş Kalman filtresi kullanılabilir [81]. Kalman filtresi algoritmasında robotun konum hakkında tahmin ve ınancı Gaussian dağılımı ile gösterilmektedir ve sorunu çözmek için robotun odometri bilgilerikullanılmaktadır.

Bu tezde 'Yer keşfetme' problemini çözmek için Markov modelini kullanan algoritma incelenip başka 'Yer keşfetme' algoritmaları ile karşılaştırılmaktadır. Markov yer keşfetme algoritması ortamda robotun olabilecek tüm konumlar için olasılık tahminlerini hafızasında tutmaktadır. Örnek olarak hareket etmeye başlamadan önce robot kendi konumu hakkında hiçbir bilgiye sahip olmadığından dolayı olasılık dağılımı ortamdaki her konum için aynıdır. Bir kaç hareket sonrasında bazı konumlar için olasılıklar başka konumladan fazla olabilir çünkü robot kendi yerini tam olarak saptayamamıştır. Belli bir konum için olasılık değeri başka konumlara kıyasen yüksek değere ulaştığı zaman, robot kendi konumu keşfetmiş demektir. Markov

modelinin problemi çözmeye yönelik stokastik yaklaşımı robotun konumu hakkında kesin bilgisi olmadığı durumlar için avantaj sağlamaktadır. Bu tür 'çok ihtimalli' yaklaşım robot için her zaman gereklidir çünkü modern otonom robotların sensör bilgilerindeki ve hareket sonuçlarındaki hata payları stokastik yaklaşımı gerektirmektedir. Algoritmadaki kullanılan ortamın haritasını ayrıklaştırma ve küçük parçalara bölme her konum için olasılıkları göze almayı sağlamaktadır. Ayrıca ortamın çok küçük parçacık kümesi olarak algılanması robotun hareket sonucunda doğan her türlü konumu kapsamasını sağlamaktadır. Bunun dışında robotun sensör bilgilerinin her türlü yöndem kullanılmasını sağlamaktadır. Başka yer keşfetme algoritmaları ise ortamın haritasını belli bölgeleri bölerek, bölge olasılığı bilgisini hafızasında tutmaktadır. Bu durumda robot bir bölgenin içindeki iki farklı konumda bile olsa algoritma robotun bir konumda olduğunu gösterir[71]. Bunun dışında özel işaret kullanan 'Yer keşfetme' algorimaları mevcuttur. Bu tür algoritmalarda robot kendi yerini özel işaretlere bağıntılı olarak saptamaktadır [73]. Fakat robot her zaman özel işaretleri göremeyebilir ve görse bile bazen ortamdaki başka nesnelerden ayırmayablir [65].

Bu tezde yukarıdaki sorunları çözmek için stokastik yöntemleri kullanan Markov yer keşfetme algoritması incelenip görsel uygulaması yapılmaktadır. Sonuç olarak başka stokastik yer keşfetme algoritmaları ile karşılaştırılarak avantajları gösterilmektedir. Modern robotlardaki hata payları statistiksel ve stokastik yaklaşım gerektirmektedir. Bu çalışma ise tam seyyar ve tam otonom robotların oluşabilmesi için büyük katkı sağlayacaktır.

2. GENEL BİLGİLER

Markov yer keşfetme algoritması ortamın haritasını ve robotun sensör bilgilerini kullanalarak robotun kendi konumunu belirleme problemidir. Bu algoritmada ortamdaki ayrıklaştırılmış her konum için belli bir olasılık verilerek robotun her adımından sonra bu olasılıklar güncellenir.

Örnek olarak sadece duvalları ve 3 kapısı olan bir boyutlu ortamı alalım [Şekil 2.2.1]. Robotumuz ise sadece sağa hareket edebildiğini ve hareket hata payını şimdilik olmadığını varsayalım. Ayrıca robotun sadece kapı ve duvarı ayırt edebilecek sensörü var olduğunu varsayalım. Şimdi robotu bu ortamda bir yere koyduğumuzu varsayalım, fakat robot hangi yere koyulduğunu bilmemektdir. Markov yer keşfetme algoritmasında bu durumu her konum için aynı ihtimali vererek gösterilmektedir [Şekil a]. Bir adım sağa gittikten sonra robotun sensörü kapı algıladığını varsayalım. Bu bilgileri kullanarak robotun konumlar arasındaki olasılık dağılımı kapılar arasında eşit olacaktır. Bunun anlamı robot hangi kapının yanında olduğunu saptayamamıstır ve robotun elindeki bilgi konumunu saptamak için yeterli değildir [Şekil b].

Bir adım daha gittikten sonra robot duvalı görmektedir ve olasılık dağılımı kapılardan sonraki duvarlarda eşit olarak görülmektedir. Bu işlem olasılık dağılımların sağa kayması olarak da görülebilir [Şekil c]. En son adımda robot sağa gittikten sonra kapıyı algılamaktadır ve doğru olarak ikinci kapının yanında olma ihtimali çok yüksektir, çünkü (kapı, duval, kapı) üçlüsünü ancak bu 3 hareket sonrasında ikinci kapının yanında görebilir.

2. 1. Temel kavramlar

Algoritmayı detaylı olarak incelemeye başlamadan önce temel kavramlara bakalım. İlk başta robotun konumunu l değişkeni ile gösterelim. l_t ise robotun t zamannındaki gerçcek konumunu belirlesin, L_t ise ilgili rastsal değişkenimiz olsun. Normalde robot kendi konumu hakkında kesin bir bilgisi yoktur ve sadece nerde olabileceği konusunda bir olasılık dağılımına sahiptir. $Bel(L_t)$ robotun t zamanındaki ortamdaki olabileceği yerler üzerinde olasılık fonksiyonu olsun. Mesela $Bel(L_t=l)$ değeri robotun t zamanında l konumunda olabilme ihtimalini verir. Bu

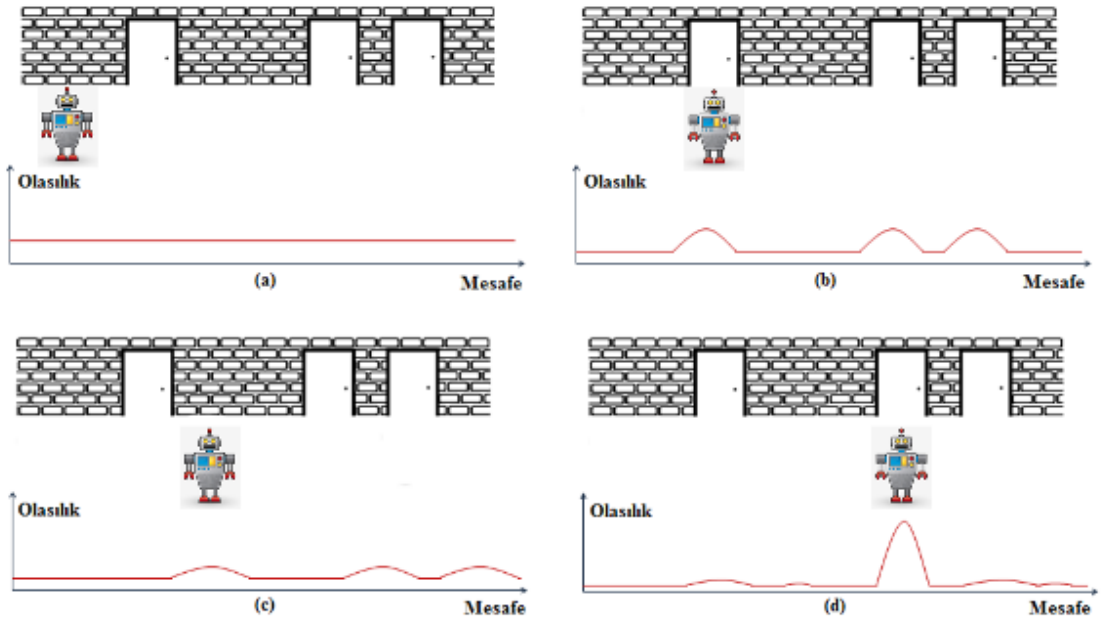
değer iki durumda değişir: birincisi eğer robot sensör aracılığı ile ortam hakkında belli bilgi edinirse, ve ikincisi robot hareket sonrasında yer değiştirirse. Sensör aracılığı ile alınan bilgiye *s* diyelim, ve

robotun hareketlerine *a* diyelim. Bunların ilgili rastsal değişkenleri *S* ve *A* olsun.

Robot zaman içinde sensör ve hareket bilgilerini bir dizi içinde algılamakta olduğunu varsayalım. Robot zaman içinde sensör ve hareket bilgilerini bir dizi içinde algılamakta olduğunu varsayalım:

$$d = \{d_0, d_1, \ldots, d_T\} \qquad (1)$$

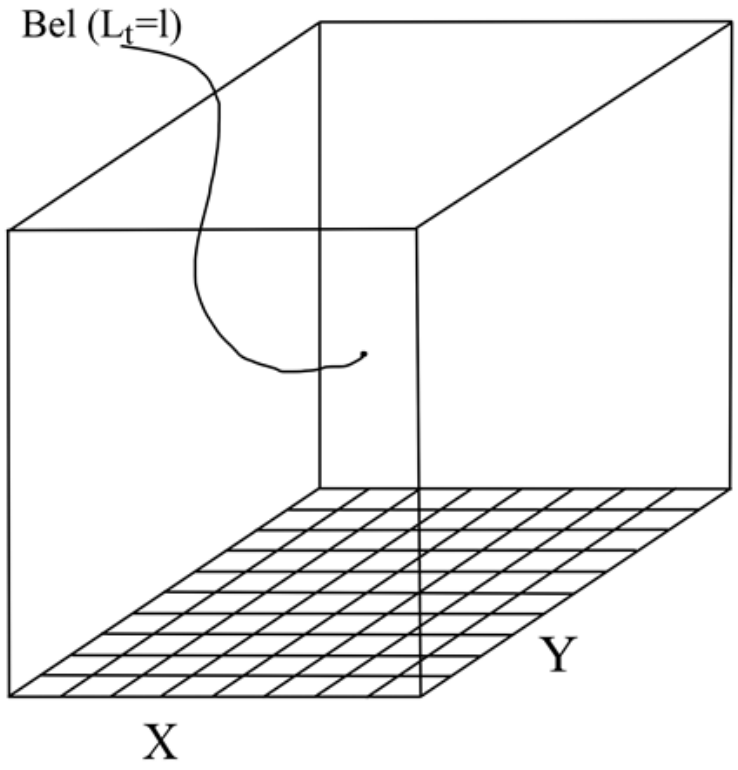
Bu dizideki her değer 0<=t<=T zaman aralığı içindeki sensör ya da hareket bilgislidir. Burdaki *t* bu bilgiyi zaman olarak indeksler, *T* ise en son alınan bilginin zamanıdır. Tam olarak *d* dizisi ise bütün alınan bilgileri simgeler, bu diziyi bundan sonra veri dizisi diye adladıracağız.



Şekil 2.1.1: Robotun konum hakkındaki ihtimal dağılımı

2.2. Ortam haritasının ayrıklaştırılması

Değişik yer keşfetme algoritmaları ortamı ayrıklaştırmak için haritayı topolojik özelliklerine bölmektedir [53]. Bu tür ayrıklaştırma türü yer keşfetme sorunun çözebilir fakat ortamın değişik konumları çok kaba bir şekilde bir bibirinden fark ettiği için konum belirleme de tam olarak yapılamıyor, çünkü ortamı topolojik özelliklere göre bölme tam olarak bir yerin konumunu tanımlayamaz. Ayrıca bu tür topolojik ayrıklaştırma sonucunda ortamda robotun algılayabileceği ve fark edebileceği özel veya soyut işaretler olmalı ki robot sensör aracılığı ile bu tür bilgileri başkalardan ayırlabilsin. Bu varsayımlardan dolayı yapılandırılmamış ortamları topolojik olarak konumlara ayrıklaştırmak çok zor ve verimsiz oluyor. Markov yer keşfetme algoritmasında ise ortamdaki en ince konumları bile göz önünde tutmak için haritayı ızgara gibi parçalara ayrıklaştırmaktadır [Şekil 2.2.1]. Robot ise bu ortamdaki her bir konumda olabileceği konusunda için belli bir inancı(belief) vardır ve bu olasılığı $Bel(L_T=l)$ ile ifade ederiz. Dikkat ettiğiniz gibi inanç her zaman değişebileceğinden dolayı zamana(T) bağlı olarak ifade ediyoruz.



Şekil 2.2.1: Ortamın ızgara şeklinde ayrıklaştırılması

Bu metod ile ortamdaki robotun olabilecek tüm yerler birim kare şeklinde gösteriliyor. L(x,y) boyutlarının en küçük karesinin uzunluğu yaklaşık 10 cm dir. Yalnız şekildeki ayrıklaştırma ortamı çok küçük parçacıklara bölebilmesine rağmen, robotun tüm olabilecek konum ve durumlar için hesaplama yapması ve hafızasında bilgi tutması robot için aleyhte durum oluşturur.

2.3. Hareket modelinin hesaplanması

Robot bir konumda hareket ettiği zaman hareketin hangi konuma götüreceğini gösteren olasılık dağılımı hareket modelidir, bu model $P(l \mid a,l')$ ile gösterilir. Bu ifadede l' robotun hareket etmeden önceki konumu, a ise robotun hareketi ve l o hareket sonrasındaki robotun yeni konumu. Bazen robotun hareketleri beklenen konuma getirmediği için bu model robotun hareketlerindeki hata paylarını belirler. Örnek olarak eğer robotun sağa gitme çabası 90% ihtimal ile robotu bir birim sağa kaydırıyorsa ve 10% ihtimal ile robotu aynı konumda hareketsiz birakıyorsa hareket modeli aşağıdaki gibidir:

$$P((x+1, y) \mid \text{'sağa git'}, (x, y)) = 0.9$$
$$P((x, y) \mid \text{'sağa git'}, (x, y)) = 0.1$$

Yukarıdaki ifadede görüldüğü gibi bu tür hareket modeli robotun hatalı işleyişinden doğan sonuçları da göz önünde bulundurur. Pratikte hareket modelinin değerini yaklaşık olarak bulabiliriz ya da robot kendi hareketetlerinin sonucunda bu modeli hesaplayabilir [7].

2.4. Sensör modelinin hesaplanması

Robot belli bir konumda sensörden s değerinde bir bilgi aldığını varsayalım. Bu bilgiyi kullanarak robot kendisinin hangi konumda olabileceği olasılığını gösteren modele sensör modeli deriz. Bu model $P(s \mid l)$ ile ifade edilir. Örnek olarak robotun 3 değişik renk sensörü olduğunu varsayalım ve belli bir anda robot sarı rengi algıladığını varsayalım. Ortamın ayrıklaştırılmış haritasında toplam 100 konumdan 25 konumda robot sarı renk algılayabileceği hesaplanmış ise sensör modeli aşağıdaki gibi olacaktır:

$$P(\text{'Sarı renk algıladım'} \mid \text{'Sarı renkte olan tüm konumlar'}) = 0.25$$

Robot sensör modelini hesaplamak için tüm sensör bilgisi göz önünde tutarak tüm değerleri hesaplaması gerekiyor. Mesela robotun sensörü kamera ise tüm görebileceği resim türleri için olabilecek konumları hesaplaması gerekir, ve şu anki bilgisayarlarda bunu gerçek zamanlı olarak yapmak imkansızdır. Onun için algoritma çalışmaya başlamadan önce sensörlere bağlı olan tüm olasılıklar bir defa hesaplanıyor. Yani haritadaki algılanabilecek tüm sensör bilgilerine bağlı olarak hangi konumda olabileceği ihtimali hesaplanıyor. Önceden örnek verdiğimiz robotun 3 tane renk(Sarı, Kırmızı, Mavi)sensörü varsa sensör modeli üç farklı değerlerden oluşur:

P('Sarı renk algıladım' |'Sarı renkte olan tüm konumlar')
P('Kırmızı renk algıladım' |'Kırmızı renkte olan tüm konumlar')
P('Mavi renk algıladım' |'Mavi renkte olan tüm konumlar')

Yalnız yukarıdaki durum sensörlerin hatasız olduğu zamanlar geçerlidir. Sensörün hatalı olabileceği durumlarda ise hata payı da göz önüne alınması gerekir.

3. MARKOV YER KEŞFETME ALGORITMASI

3.1. Özyinelemeli yer keşetme

Markov yer keşetme algoritması veri dizisini*(d)* kullanarak L_t değişkeni üzerindeki ihtimal dağılımını saptar, yani

$$P(L_T=l \mid d) = P(L_T=l \mid d_0,d_1,...,d_T). \qquad (2)$$

Bu olasılık fonksiyonunun nasıl hesaplandığını göstermeden önce, hesaplamalardaki önemli olan *Markov varsayımını* ele alalım. Markov varsayımı başka deyişle statik dünya varsayımı der ki: eğer t zamanındaki robot konumu belli ise l_t, gelecekteki alınan bilgiler geçmişteki alınan bilgilerden bağımsızdır, yani

$$P(d_{t+1},d_{t+2},... \mid L_t=l,d_0,d_1,...,d_T)=P(d_{t+1},d_{t+2},... \mid L_t=l) \qquad (3)$$

Başka deyişle robotun konumu gelecekteki veriyi tahmin etmek için tek kriterdir.
$P(L_T=l \mid d)$ hesaplarken gelen verinin türüne göre iki yol vardır: eğer d_T sensör bilgisi ise veya d_T hareket bilgisi ise.

1. Durum: En son alınan veri sensör bilgisi ise, $d_T = s_T$.
Bu durumda

$$P(L_T=l \mid d) = P(L_T=l \mid d_0,d_1,...d_{T-1},s_T) \qquad (4)$$

Bayes kuralına göre yukarıdaki ifade aşağıdaki ifadeye dönüştürülebilir ve olasılık dağılımı bu şekilde hesaplanır

$$\frac{P(s_T \mid d_0,d_1,...d_{T-1},L_T=l) * P(L_T=l \mid d_0,d_1,...d_{T-1})}{\text{————————————————————————}} \qquad (5)$$

$$P(s_T \mid d_0,d_1,...d_{T-1})$$

sonra bu ifade Markov varsayımını kullanarak aşağıdaki ifadeye dönüştürülür:

$$\frac{P(s_T \mid L_T=l) * P(L_T=l \mid d_0,d_1,...d_{T-1})}{P(s_T \mid d_0,d_1,...d_{T-1})} \qquad (6)$$

Ayrıca bölen ifadesi L_T'ye bağlı olmadığından dolayı, sabit bir h α_T sayısı ile değiştirilebilir. Demek ifade aşağıdaki gibi basitleştirilebir

$$P(L_T=l \mid d) = \alpha_T * P(s_T \mid L_T=l) * P(L_T=l \mid d_0,d_1,...d_{T-1}) \qquad (7)$$

Eğer aşağıdaki ifadeyi formule yerleştirsek

$$Bel(L_T=l) = P(L_T=l \mid d_0,d_1,...d_T) \qquad (8)$$

(7) numaralı formül aşağıdaki formule dönüşmüş olur

$$Bel(L_T=l) = \alpha_T * P(s_T \mid L_t=l) * Bel(L_{T-1}=l) \qquad (9)$$

Bu özyinelemeli ifade bize robotun *t* zamanında *l* konumunda olduğunun ihtimalini verir.

2. Durum: En son alınan veri hareket bilgisi ise, $d_T = a_T$.

Bu durumda $P(L_T=l \mid d)$ ifadesi Toplam Olasılık Teoremini kullanarak hesaplanır:

$$P(L_T=l \mid d) = \int P(L_T=l \mid d,L_{T-1}=l')* P(L_{T-1}=l' \mid d)*dl' \qquad (10)$$

Sağ taraftaki ilk terimi Markov varsayımına göre aşağıdaki gibi yazabiliriz

$$P(L_T=l \mid d,L_{T-1}=l') = P(L_T=l \mid d_0,d_1,...d_{T-1},a_T,L_{T-1}=l') \qquad (11)$$

$$= P(L_T=l \mid a_T,L_{T-1}=l') \qquad (12)$$

(10) İfadedeki sağdaki ikinci terimi, a_T bilgisi L_{T-1} hakkında hiçbir bilgi taşımadığı için, aşağıdaki gibi basitleştirebiliriz:

$$P(L_T=l' \mid d) = P(L_{T-1}=l' \mid d_0,d_1,...d_{T-1},a_T) \quad (13)$$
$$= P(L_{T-1}=l' \mid d_0,d_1,...d_{T-1})$$

(14)

(12) ve (14)'teki ifadeleri Denklem (10) eklersek sonuçtaki denklem aşağıdaki gibi olur:

$$P(L_T=l \mid d) = \int P(L_T=l \mid a_T,L_{T-1}=l')* P(L_{T-1}=l' \mid d_0,d_1,...d_{T-1}) \ *dl' \quad (15)$$

(15) Denklemin özyinelemeli olduğu görerek aşağıdaki gibi yazabiliriz:

$$Bel(L_T=l) = \int P(l \mid a_T,l') * Bel(L_{T-1}=l') * dl'. \quad (16)$$

Not olarak $P(L_T=l \mid a_T,L_{T-1}=l')$ ifadesini $P(l \mid a_T,l')$ ile değiştirdiğimizin sebebi ifadenin zamana bağlı olmayışındandır.

3.2. Markov yer keşfetme algoritması

(9) ve (16) ihtimal güncelleme ifadeleri Markov yer keşfetme algoritmasının temelini oluşturur. Algoritmanın tamamı Tablo 3.2.1'de gösterilmiştir. Genel olarak $P(l \mid a,l')$ ifadesini *robotun hareket modeli* çünkü bu ifade robotun hareketi onun konumunu nasıl etkilediğini modeller. $P(s \mid l)$ ifadesini ise *robotun sensör modeli* diye adlandırıyoruz çünkü bu ifade robotun sensörlerinin konuma bağlı olarak nasıl bir sonuç verdiğini modeller.

Markov yer keşfetme algoritmasında $P(L_0=l)$, yani $Bel(L_0)$ robotun başlangıçtaki konumunun ihtimal hesabını verir. Bu olasılık dağılımı farklı şekilde atanabilir, fakat genel olarak iki tür başlangıç vardır: Eğer robotun başlangıç konumu hiç bilinmiyorsa $P(L_0)$ ifadesi her konum için aynıdır, ama eğer robotun konumu yaklaşık olarak biliniyorsa $P(L_0)$ ifadesi o konumda merkezleştirilmiş Gauss dağılımına eşittir. Sonrasında herbir veri dizisinin elemanı için o verinin hareket veya sensör bilgisi olduğuna dayanarak robotun her bir konum için olasılık hesapları güncelleniyor. Bu güncelleme olasılık dağılımı belli bir değerlere ulaşıncaya kadar devam eder. Bizim kullandığımız algoritmada eğer belli bir konumun olabilme ihtimali başka

konumlarda fazla ise algoritma robotun yerini keşfetmiş olur. Bazı durumlarda bu ihtimalin belli bir sınırı geçmiş olması istenir [5].

Tablo 3.2.1. Markov Yer keşfetme algoritması.

Ortamdaki her konum *l* için başlangıç değerini ata:

$Bel(L_0=l) = P(L_0=l)$

Bir konunum ihtimal değeri diğerlerinen fazla oluncaya kadar tekrar et:

Eğer gelen veri s_T sensör verisi ise:

$\alpha_T = 0$

Ortamdaki her konum *l* için:

$Bel(L_T=l) = P(s_T \mid l) * Bel(L_{T-1}=l)$

$\alpha_T = \alpha_T + Bel(L_T=l)$

Ortamdaki her konum *l* için:

$\alpha_T = Bel(L_T=l) / \alpha_T$

Eğer gelen veri a_T hareket verisi ise:

Ortamdaki her konum *l* için:

$Bel(L_T=l) = \int P(l \mid a_T,l') * Bel(L_{T-1}=l') * dl'.$

3.3. Markov algoritmasının kısıtlamaları

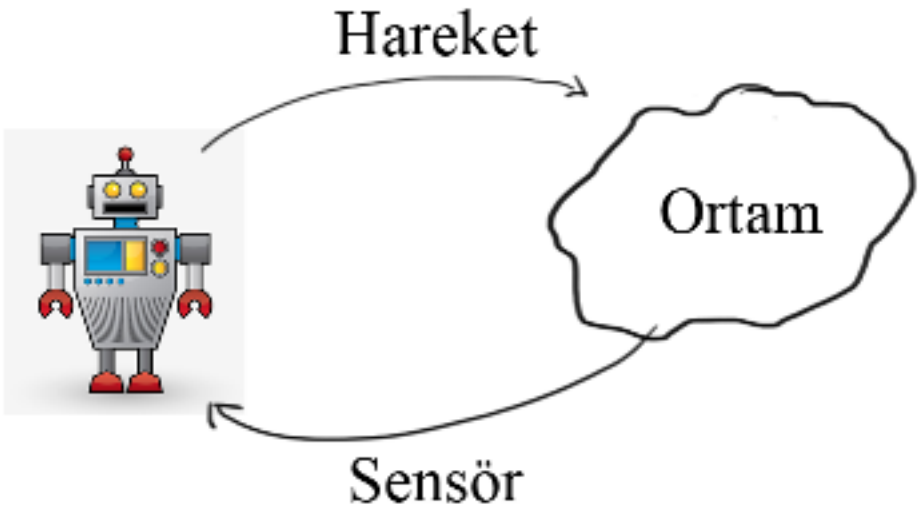
Bu tezdeki gösterilen Markov yer keşfetme algoritması robotun olasılık dağılımını tüm olabilecek konumlar için hesaplar. Algoritmanın bütün konumları göz önüne alması robotun herhangi bir konumdan başlayıp yer keşfetmesine yarıyor. Kalman filtresi algoritmasında ise robot kendi konumu hakkındaki olasılık hesabını belli bir yerde merkezleyerek yapması

algoritmanın genel yer keşfetmek için yararsız hale gelmesine nedendir [71]. Ortamı topolojik olarak bölen algoritmaların yanı sıra Markov yer keşfetme algoritması daha çok türlü ortamlar için geçerliğini gösteriyor ve ortamda robot için özel işaret gerektirmiyor. Robot ortamdaki özelliklere bağlı olmadığından dolayı sensörden gelen bilgileri olasılık hesapları için çok rahatlıkla kullanabilir. Bu durum algoritmanın daha net sonuçlar çıkarmasına el verir. Öte yandan ortamın çok küçük parçalara ayrıklaştırılması tüm konumlar için ihtimal değerlerin robotun hafızasında saklanmasına neden oluyor. Örnek olarak 30x30 m^2 ortam için, 15x15 cm^2 karelere ayrıklaştırma toplamda 7,200,000 konum için olasılık değerlerinin saklanmasına neden oluyor. Markov yer keşfetme algoritması her sensör ve hareket bilgisinden sonra tüm olasılık değerlerini güncellemesi gerekir. Şu anki bilgisayarlar bu boyuttaki matrislerin değerlerinin gerçek zamanlı hesaplanmasına izin vermiyor.

3.4. Markov algoritmasının özeti

Gördüğümüz gibi Markov algoritmasında yer keşfetmek için olasılık teorisi üzerinde kurulan stokastik yöntemleri kullanabiliriz. Yani belli bir konum hakkında ihtimal hesaplamak yerine robot tüm olabileceği konumları hesaba katıyor. Başta robot nerede başladığından haberdar olmadığından tüm konumlar için sabit bir olasılık verir. Veya belli bir konumda olduğunu düşünüyorsa o konuma daha fazla olasılık değeri atar. Hareket etmeye başlayınca kendi hareket ve sensör bilgilerini kullanarak kendi konumu hakkında bilgi edinir. Eğer hareket ve sensör modeli için olasılık dağılımlarını önceden hesapladıysa bu bilgi robotun yer keşfetmesinde kullanılacaktır. Bayes kuralını ve Markov varsayımını kullanarak robotun konum hakkında olasılık dağılımını hesapyacak algoritma elde ederiz. [Tablo 3.2.1] Bu algoritmanın gerçekleştirmek için bilmemiz gerek bilgiler: robotun konum hakkında başlangıç ınancı, hareket modeli ve sensör modeli. Yalnız ortamın ne kadar detaylı ayrıklaştırıldığını ve robotun hareketlerinin hata paylarını göz önünde bulundurarak algoritmanın çalısması çok kolay olmadığını gösterebiliriz. Bu zorlukları aşmak için farklı yollar vardır. Ortamdaki olasılık dağılımını devamlı Gaussian olarak kabul etmek hesaplama zorluluğunu Kalman filtersindeki gibi iyice kolaylaştırır. Yalnız tek-modlu olasılık dağılımı çok konumları göz önünde bulundurmaya imkan vermiyor. Onun için ortamı ızgara olarak detaylı ayrıklaştırmak çok-modlu olasılık dağılımına şmkan verir. Yalnız böyle yaklaşım olasılıkları hesaplamayı zorlaştırıyor. Parçaçık filtreleri yaklaşımı ortamdaki tüm konumların yerine belli konumlar için olasılık

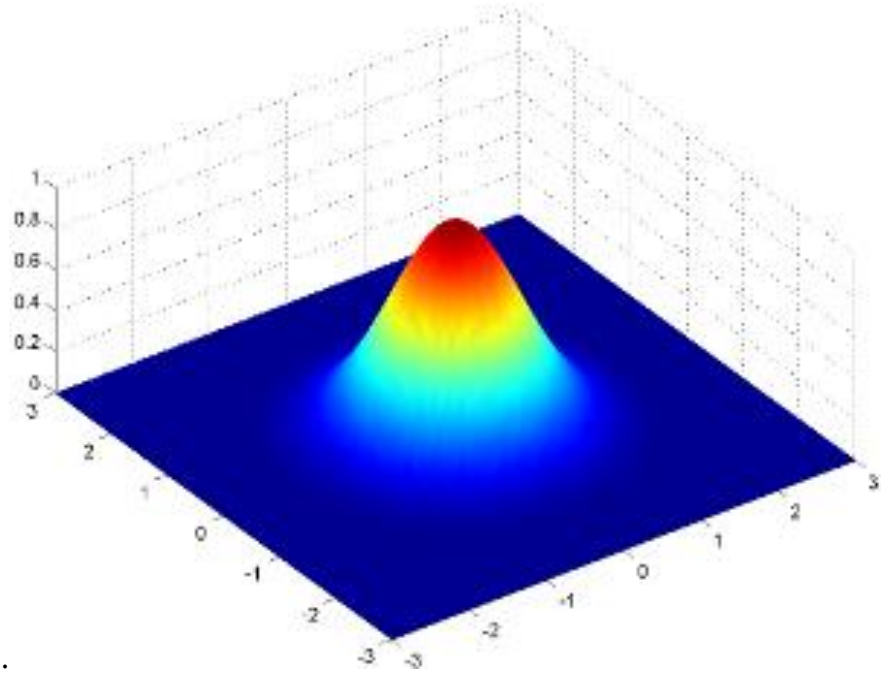
dağılımını hesapladığı için daha verimlidir. Ayrıca hareket ve sensör bilgisindeki hata paylarını da olasılıklar ile göstermemiz hesaplamamızı çok daha kolaylaştırır.



Şekil 3.4.1: Markov yer keşfetme algoritmasının özeti

4. KALMAN FİLTRESİ

Bazı bilim adamları Kalman filtresini statistiksel hesaplama teorisinde yüzyılın icadı olarak kabül ederler. Bu bölümde 'Konum Takip etme' problemi için kullanılan Kalman filtresi algoritması kullanılacaktır. Kalman filtreleri ortamı ayrıklaştırmadan devamlı bir değişken olarak kabül ediyor. Ve bu değişken üzerinde belli bir konum merkezli ve belli bir sapma ile Gaussian olasılığını hesaplıyor [Şekil 4.1]. Bu bölümde başta genel olarak Kalman fitresinin nasıl çalıştığını gösteriliyor, sonra örnek bir ortamda algoritmanın nasıl çalıştığı inceleniyor.



Şekil 4.1: Gaussian dağılımı

4.1 Konum takip etmek için Kalman filtresi

Kalman filtersi normalde sinyal işlemede kullanılan gürültüyü kaldırma metodudur. Bizin durumda ise robotun sensör ve hareketteki hata paylarını gürültü olarak kabül ediyoruz. Yani robot belli bir sensör verisi elde ettiği zaman belli bir oranda hatası olabilir. Örnek olarak robot bir konumda sensörü kırmızı ışık algıladığı zaman aslında görünen sarı ışık olabilir. Veya sağ tarafa hareket ettiği zaman sonuç olarak yerinde kalabilir. Kısaca Kalman filtresi robotun konumunu hata payı olan bir sistemden çıkaran algoritmasıdır. Konum dediğimiz zaman *(x,y)*

ikilisini kast ediyoruz. Kalman filtersinin çalışması için ayrıca robotun sensör ve hareket bilgilerinin olması gerekir. Bu bilgilerin hata payları Gaussian olasılık dağılımı ile modellenmiş ise Kalman filtresi robotun konumunu optimal bir şekilde bulacağı kesindir [71]. Ayrıca robotun ilk baştaki konumun da bilinmesi şarttır çünkü algoritma çalışmaya başlamadan önce robot kendi konumu hakkındaki inancı Gaussian olarak modellenmesi gerekir.

4.2. Kalman filtresi örneği

Algoritmanın denklemlerini incelemeye başlamadan önce genel olarak Kalman filtresinin nasıl çalıştığını incelemekte fayda var. Bir ortamda kendi konumunu belirlemek isteyen robot olduğunu kabül edelim. Gördüğümüz gibi robot hareket ettiği zaman hata paylarınan dolayı istenen konuma gitmeyebilir. Robotun konumunu kestirmek için tüm gerekli veriler elde ettiğimizi varsayalım ve robotun hareketini modelleyelim. Yani robotun konumu zaman içinde nasıl değiştiğinin formülünu yazalım. Robot sabit bir $\boldsymbol{h}$ hızı ile gittiğini varsayalım ve $\mathbf{x_k}$ ise robotun gerçek konumunu göstersin. Bu durumda denklem aşağıdaki gibidir:

$$x_k = x_{k-1} + h + w_k \qquad (17)$$

Yani robotun yeni konumu $x_{k,}$ önceki konumuna $x_{k-1,}$ h hızına ve w_k hata payına bağlı olduğunu varsayalım. Hata payını 0 ortalamalı rastgele Gaussian dağılımlı olduğunu varsayalım. Yani genel olarak hata payının 0 olduğunu ve yalnız bazı durumlarda 0'dan daha fazla ve daha az olabileceği anlamına gelir. Hata payındaki sapmanın ise σ_w olduğunu varsayalım.

Robotun konumunu sensörleri kullanarak keşfetmek istediğimize göre bu sensörlerin konuma nasıl bağlı olduğunu göstermemiz gerekir. Sensörlerin bilgisinin hesaplanması ile konumun arasındaki bağlantıyı aşağıdaki denklem ile gösterebiliriz. Bu denkleme hareket modeli denir:

$$z_k = x_k + v_k \qquad (18)$$

Yani robotun algılanan konumu z_k robotun gerçek konumunundan x_k bir v_k hatası ile algılanmıştır. Aynı şekilde bu hata payının 0 ortalamalı rastgele Gaussian dağılımlı olduğunu varsayalım.

Başlangıç: Robotun başlangıç konumunun x_0 olduğunu varsayalım. Ve bu konum hakkında robotun inancı kesin olmayıp varyans σ_0 olduğunu varsayalım.

Hareket: Robotun 1 adım ileri gittikten sonra sistemin modelinden (17) robotun konumunun yaklaşık h ile değişeceğini biliyoruz. Bu bilgi ile robotun konumunu hesaplayabiliriz. Yani bir adımdan sonra robotun yaklaşık nerde olduğunu bilebiliriz. Robotun k=1'deki yeni konumunu aşağıdaki denklem ile hesaplanır:

$$x_1 = x_0 + h + 0 \qquad (19)$$

yukarıdaki denklemde robotun sensör hata payını 0 olarak kabül ettikten sonra (17) denklemden ise robotun sensörlerinin hata payının olabileceğini bildiğimize rağmen kesin olarak bir an için ne kadar hata olduğunu bilmiyoruz. Fakat ortalama olarak hatanın 0 olduğunu bildiğimiz için yeni konumu hesaplamak için 0 değerini kullandık.

Aynı zamanda hata payının 0 etrafından nasıl değiştiğini biliyoruz. Bu bilgiyi yeni hesapta hatayı güncellemek için kullanarak yeni belirsizliğin varyansını σ_1^2 aşağıdaki gibi hesaplarız:

$$\sigma_1^2 = \sigma_0^2 + \sigma_w^2 \qquad (20)$$

Sensör bilgisi aracılığı ile doğrulama: Eğer robot kendi konumu hakkında kesin bilgi edinmeden hareket etmeye devam ederse konum hakkındaki belirsizlik (20) denklemde belirtildiği gibi artmaya devam edecektir. Fakat eğer kesin bir sensör bilgisi elde edersek konum hakkındaki inancımızı güncelleyeyip robotun belirsizliğini azaltırız. Yani algıladığımız bilgi yaptığımız tahmini doğrulamak için kullanabiliriz.

z_1 aldığımız kesin sensör ölçümü varsayalım. Şimdi bu bilgiyi yeni tahminimizin hesaplanmasına katmak istiyoruz. Bu bilgiyi ağırlıklı ortalama kullanarak yeni konum hesaplamasına katmak için aşağıdaki ortalama denklemini kullanırız:

$$\hat{x}_1^+ = \frac{\sigma_v^2}{\sigma_1^2 + \sigma_v^2}\hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}z_1$$
$$= \hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}(z_1 - \hat{x}_1).$$

Sensör bilgisi önceki konum bilgisine bağımlı olmadığından dolayı bağımsız göstergedirler. Bundan dolayı sensör bilgileri konum üzerindeki belirsizliği azaltma özelliğine sahiptirler. Matematiksel olarak konum inancının Gaussian dağılımının varyansı aşağıdaki gibidir:

$$\frac{1}{\sigma_1^{2,+}} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_v^2},$$

Yukarıdaki denklemden konum belirsizliğin σ_1^{2+} eski belirsizlikten her zaman daha küçük olduğunu görebilirsiniz. Ayrıca bu denklemi aşağıdaki gibi yazarak *K* ağırlık faktörü elde ederiz.

$$\sigma_1^{2+} = (1\text{-}K)\ \sigma_1^2$$

$$K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}.$$

K ağırlık faktörü ise yeni konumu güncellerken sensörlerden ne kadar bilgi alınması gerektiğini anlatır. Eğer en son konum hakkındaki hesaplamada belirsizlik oranı az ise *K* o kadar az olur çünkü konum hesaplaması doğru gittiğini ifade eder. Eğer K bire yaklaşırsa demek ki yeni konum belirlerken sensör bilgileri daha ağırlıklı olacak. Kalman filtresinin işleyişin örneğini [Şekil 4.3.1] görebilirsiniz. Robot harekete başlamadan önce belli bir konumda olduğu inancına sahiptir (a). Sağa bir defa hareket ettikten sonra gerçek konum hakkındaki ihtimal değeri düştü çünkü Gaussian'ın yeni varyansı hareketteki hata payı oranında büyüdü (b). Hareket ettikten sonra robot sensör bilgilerini kullanarak gerçek konum hakkındaki inancını artırıyor (c). Bir daha hareket ettikten sonra robotun konum hakkındaki inancı yine azalıyor.

4.3. Kalman filtresi konum takip etme algoritması

Yukarıdaki bir boyutlu örneği matris şeklinde yazarak Kalman filtresini istediğimiz boyuta kadar hesaplayabiliriz. İlk başta bilmemiz gereken bazı terimleri ele alalım:

K – Kalman katsayısı
P – Belirsizlik matrisi
F – Konum değişim matrisi
u – hareket vektörü
z – en son alınan sensör bilgisi
H – sensör fonksiyonu
R – sensördeki hata matrisi
x – konumu vektörü
I – özdeş matris

Bu bilgileri kullanarak Kalman filtersi algoritmasını aşağıdaki Tablo 4.3.1'de görebilirsiniz. Dikkat ettiğiniz gibi her bir sensör bilgisi alındıktan sonra robot kendi konumunu ve olasılık dağılımını güncellemektedir.

Tablo 4.3.1: Kalman filtresi algoritması

Her sensör ve hareket bilgisi için tekrarla:

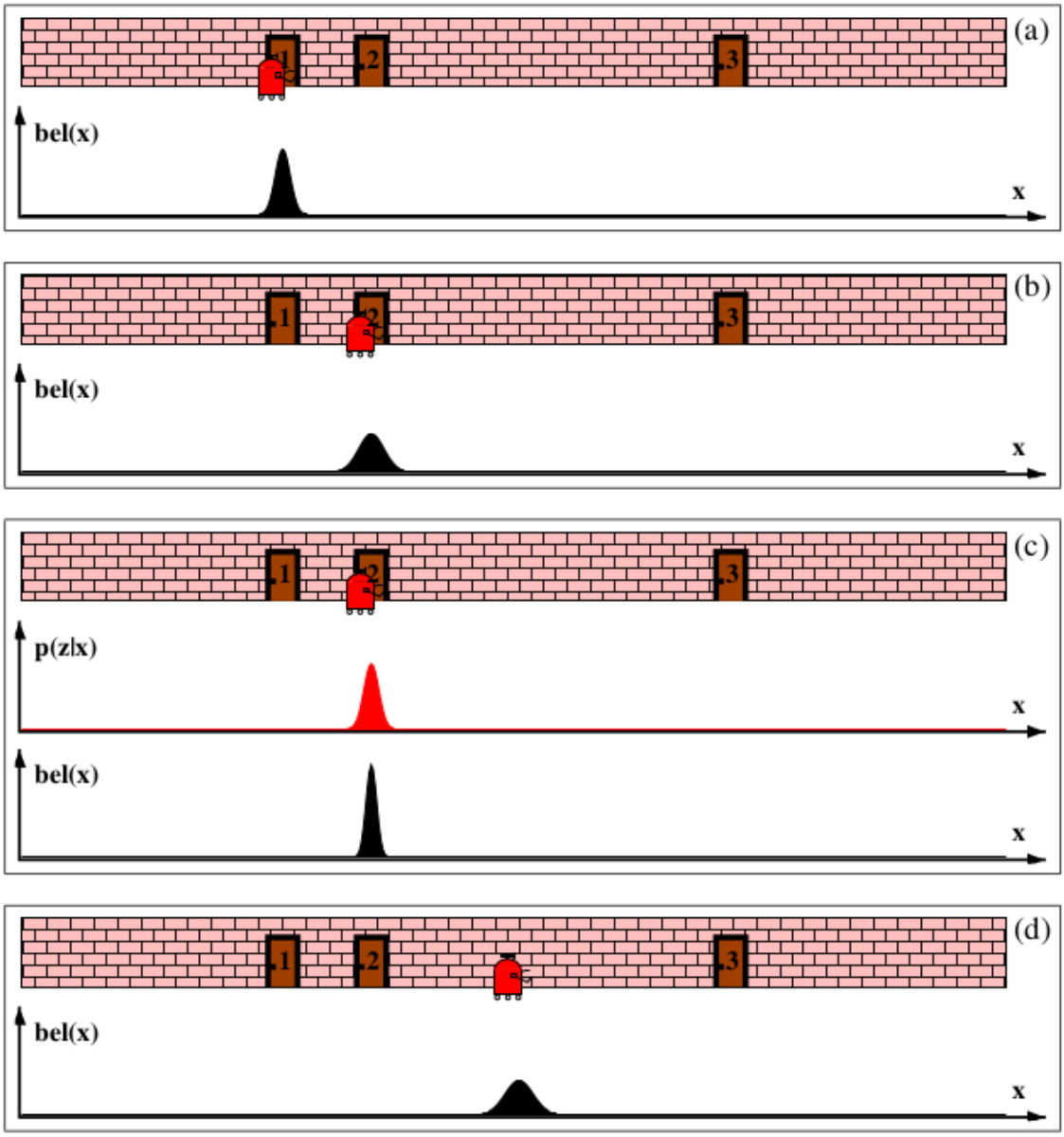
$$K = P*H^T*((H*P*H^T+R)^{-1})$$

$$x = x + K*(z-H*x)$$

$$P = (I-K*H)*P$$

$$x = F*x+u$$

$$P = F*P*(F^T)$$



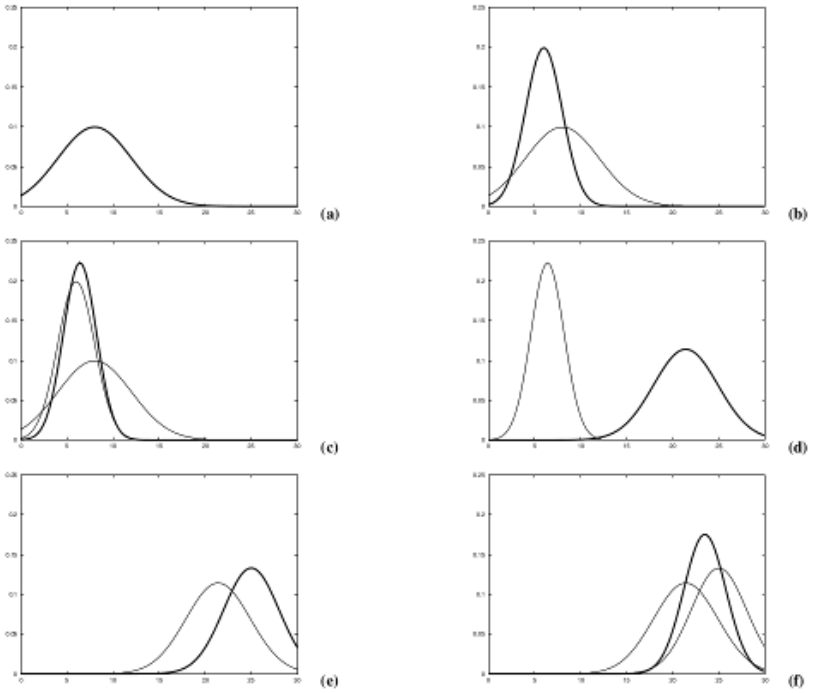
Şekil 4.3.1: Kalman filtresi örneği [Thrun 35]

4.4. Kalman filtresi özeti

Kalman filtresi konum takip etmek ve sensör bilgilerini robotun konum olasılık dağılımı hesabına katmak için ideal araçtır. Yalnız robun hareketteki ve sensördeki hata payları da Gaussian olasılık dağılımı olarak gösterilmesi gerekir. Markov algoritmasında da olduğu gibi Kalman filtresi algoritmasının iki adımı vardır: 'Hareket güncellemesi' ve 'Sensör güncellemesi'.

[Şekil 4.4.1.f]'te gördüğünüz gibi Hareket güncellemesi olasılık dağılımındaki belirsizliği arttırır, Sensör güncellemesi ise olasılık dağılımındaki belirsizliği azaltır [Şekil 4.4.1.c]. Dolaysıyla sensör bilgisi ne kadar hatasız ve çok olursa robotun kendi konumunu takip etmesi daha da kolaylaşıyor. Kalman filtresi ve Markov algoritmalarının temel farkı ise 'Sensör güncelleme' adımındadır. Markov algoritmasında sensörden gelen bilgi robotun her konuma olan inanç dağılımını Bayes kuralını kullanrak güncellemek için kullanılır. Kalman filtersindeki adımda ise sensörden gelen bilgi ortamdaki objelere bağlı olarak ayrı bir kümeler içinde konuma inanç olasılığını günceller. Yani Markov algoritmasında her bir konum için olasılık hesabı yapılıyorsa, Kalman filtresinde ise ortamdaki tüm konumlar bir Gaussian dağılımı içinde kabül edilir ve güncellenir.

Hareket güncelleme adımında robotun nerde olacağı tahmin edilir. Bu tahmini hesaplararken robotun hareketlerindeki hata payları Gaussian dağılım olarak gözönünde tutulur. Sensör güncellemesinde ise gelen bilgilere bağlı olarak robotun nerde olabileceği konusunda tahmin yapılır. Eğer hareket güncellemesindeki konum tahmini yanlış yapılmış ise bu sonradan sensör güncellemesi adımında doğrulanır. Bu hesabı yaparken sensörlerin hata payları da göz önünde bulundurulur.



Şekil 4.4.1: Kalman filtresinde hareket ve sensör güncellenmesi [34].

5. MONTE-CARLO ALGORİTMASI

Bu ana kadar Markov ve Kalman filtresi algoritmaları inceledik. Markov algoritması global yer keşfetmede kullanabilir olmasına rağmen ortamın çok küçük parçalara ayrıklaşmasından dolayı hesaplama zorlukları olduğunu gördük. Öte yandan Kalman filtresi robotun inancını bir Guassian olarak kolay hesaplamasına rağmen global yer keşfetme problemini çözemediğini gördük. MCL (Monte-Carlo) algoritması parçacık filtrelerin bir türü. Bu algoritma hem global yer keşfetme problemini çözebilmektedir hem de Markov algoritmasından daha hızlıdır. Bunun nedeni MCL, robotun inancını güncellemek için hızlı parçacık örnekleme teknikleri kullanmasıdır [Şekil 5.1.1]. Burda gördüğümüz parçacıklar robotun olabilecek konumların kümesidir. Başta robot nerde olduğunu bilmediği için parçacıklar ortam içinde her yerde dağılmıştır. Robot hareket ettiği veya sensörden bilgi aldığı zaman parçacıların önemine göre örneklenen küme yenilenmektedir. Parçacık kümesindeki konumlardan robotun gerçek konumuna daha yakın olanlar robotun her bilgi alışından sonra ağırlık gösterirler. Yani robotun belirsizliği azalınca parçacık sayısı düşmektedir ve sadece gerçek konuma yakın olan parçacıklar kalmaktadır. Güncellemeyi daha hızlı yapabilmek için parçaçık kümesinin büyüklüğü gerçek zamanlı olarak değişebilir. Dolaysıyla robot kendi konumu hakkında emin değil iken çok parçacıklar kullanılabilir, robotun konum hakkındaki inancı arttığı zaman ise bu küme küçülür. Ayrıca kümedeki her parçacığın robotun gerçek konumuna yakın olup olmadığını gösteren sayısal ağırlığı vardır. Bu ağırlık fazla ise o parçacık robotun gerçek konumuna yakın demektir, az ise uzaktır. Şekil 5.1.1'de ağır parçalar daha koyu renkte, hafif parçalar ise daha açık renkte gösterilmektedir.

5.1. Parçacık filtreleri

MCL algoritmasında robotun konum hakkındaki inancı $Bel(x)$ m tane ağırlıklı parça ile gösterilir. Yani:

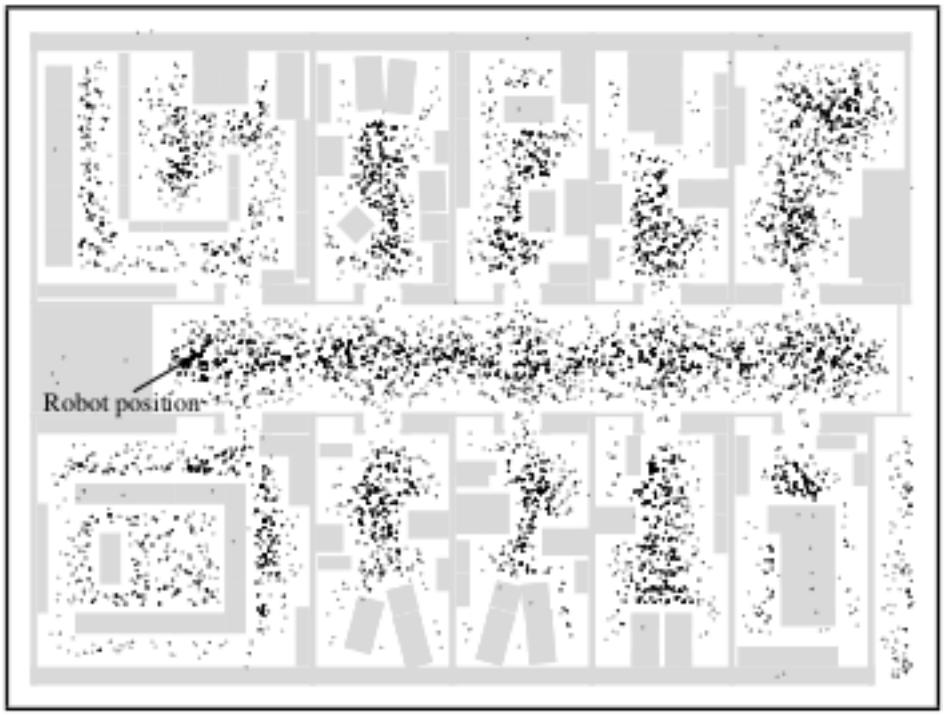
$$Bel(x) \quad = \quad \{x^{(i)}, p^{(i)}\}_{i=1,\ldots,m}$$

yukarıdaki denklemde her x_i bir konumdur, p_i ise o konum için pozitif ağırlık değeridir. Tüm ağırlıkların toplamı 1'dir. Her konumun ağırlığı o konumn önemini anlatır. Global yer keşfetme probleminde, robot hareket etmeden önce bu parçalar robotun tüm olabilecek konumlardan rastsal olarak seçilerek ağırlıkları *1/m* olarak atanır. Algoritmanın sonraki adımlarında bu parçacıklar aşağıdaki gibi güncellenir:

a) x_{t-1} parçacıklarını $Bel(x_{t-1})$ kümesinden p_{t-1} ağırlığına göre rastsal olarak seçmek

b) x_{t-1} ve u_{t-1} kullanarak x_t hesaplamak. Bu hesaplamada yeni konumu hesaplamak için $p(x_t \mid u_{t-1}, x_{t-1})$ kullanılır. Bu durumda yeni parçaçıkların ağırlığı $p(x_t \mid u_{t-1}, x_{t-1}) * Bel(x_{t-1})$ olur.

c) Sonunda x_t parçacığını normalize edilmemiş ağırlıklarına göre yani $p(y_t \mid x_t)$ göre hesaplarız.



Şekil 5.1.1: Robot hareket etmeye başlamadan önceki parçacıklar kümesi [91]

5.2. Monte-Carlo algoritması

Monte-Carlo algoritmasının 2 adımı vardır. Bunlar Hareket güncellemesi ve Sensör güncellemesidir. Hareket güncellemesini yaparak yeni x_t parçacıklarını rastlantısal olarak konumlardan seçilir. Sensör güncellemesi adımında ise z_t sensör bilgisini kullanarak $Bel(x_t)$ kümesindeki parçacıkların ağırlıkları yeniden hesaplanır. Sensör güncellemesinden sonra yeni küme robotun gerçek konumuna daha da yaklaşmış olur çünkü her parçacığın seçilmesi onun ağırlığına göre yapılmıştır. Parçacığın ağırlığı ise onun gerçek konumuna bağlı olarak güncellenir.

Tablo 5.2.1: Monte-Carlo algoritması

$Bel(x_t)$ kümesinde tek ya da çok az parçacık kalıncaya kadar tekrar et:

Hareket güncellemesi:

Her bir $Bel(x_{t-1})$ parçacı için:

x_t parçacıklarını $p(x_t \mid u_{t-1}, x_{t-1})$ göre seç

//Şu an itibari ile sensör bilgileri parçacıkların seçimini daha etkilememiştir

Sensör güncellemesi:

Her bir $Bel(x_{t-1})$ parçacı için:

x_t parçacıklarını p_i ağırlığını güncelle

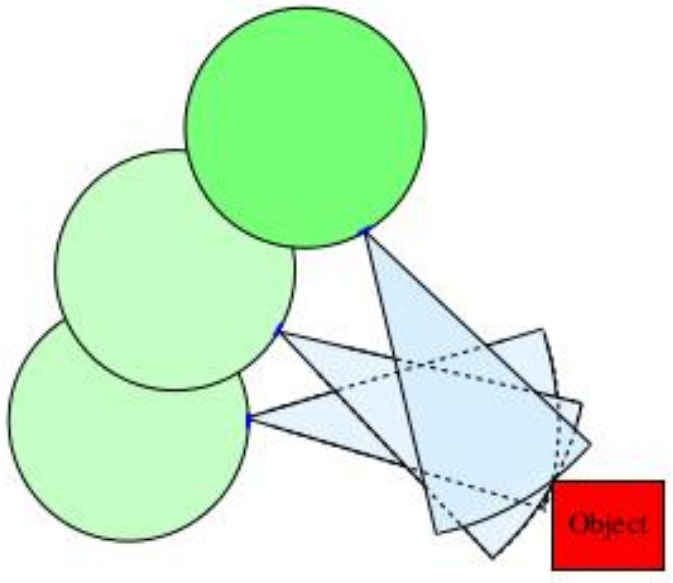
5.3. Monte-Carlo algoritmasının gerçekleştirilmesi

MCL algoritması her adımda parçacık kümesini gerçek konuma yakın olarak güncellediği için sonuca yaklaşır [10]. Yalnız p_i ağırlığın nasıl hesaplandığı çok önemlidir. Yani bir parçacığın robotun gerçek konumuna yakın olup olmadığını nasıl bileceğiz sorusuna cevap verilmesi

gerekir. Robot kendisinin konumu hakkında sadece sensör bilgisi olduğu için kümedeki parçacıklar da o bilgiyi kullanacaktır. Robot çevresinin değişik özelliklerini algılayan sensörlere göre ağırlık hesaplama türleri de değişmektedir. Örnek olarak robotun sonar sensörleri olduğu zaman bu özellik duvara ya da kapıya olan mesafe olarak algılanabilir. Görüntü sensör için de aynı durum söz konusudur. Burda önemli olan hangi özellik robotun konumunu daha iyi keşfetmesine yaramasıdır. Burda 'iyi' kriteri 3 faktörden etkilenmektedir:

1. Hesap zamanı: Bir özelliği algılamak ve hesaba katmak için geçen zaman.
2. Doğruluk: Özelliği kullanarak ne kadar doğru sonuçlar elde edildiğini gösterir.
3. Sonuca gitme: Bir çok kerek çalıştırıldıktan sonra sonuca gidilmesini gösteren kriter.

Bu özelliklerden ilki ultrason bilgiyi kullanarak robota olan mesafeyi gösteren üçgenleme noktalarıdır. Bazı ortamlarda belli yerler ultrason sensör çok net bilgiler verebilmektedir. Bu yerleri kullanarak onları ortamın belli özellikleri haline dönüştürüz [Şekil 5.3.1].



Şekil 5.3.1: Üçgenleme noktaları

Bu noktalar ile robotun arasındaki mesafe d_1, d_2, d_3 olduğunu varsayalım. Bu durumda her parçacığın bu noktalar olan uzaklıkları p_1, p_2, p_3 olduğunu hesaplarsak, eğer p_i değerleri d_i değerlerine yakınsa o zaman o parçacık robotun gerçek konumuna yakın olarak örneklendiği anlarız. Yani o parçacığın ağırlığı daha fazla olacaktır ve o parçacığının sonraki adımda seçilebilmesi şansı daha fazladır.

İkinci özellik ise ortamdaki düz çizgilerdir. Her ortam bir diktörtgen olarak modellenebildiğine göre dikdörtgenin her çizgileri bir özellik olarak kullanılabikir. Bu durumda lazer tarama ile her çizgiye olan uzaklık ölçülerek parçacık kümesinin ağırlıkları aynı şekilde hesaplanır.

MCL'de kullanabileceğimiz özelliklerden birisi de kapılardır. Lazer sensörleri kullanarak açık ve kapalık kapıları bularak kümedeki parçacıkların ağırlıkların o kapılara olan uzakları ile bağlantılı olarak hesaplarız.

5.4. Monte-Carlo algoritmasının özellikleri

MCL parçacılar örneklenmisini kullandığı için başka algoritmalardan avantajları çoktur. Mesela:

1. Kalman filtrelerine kıyasen robotun konum inancını çok parçacıklar ile ifade ettiği için robotun global yer keşfetme problemini çözebilir.
2. Markov algoritmasındaki gibi robotun tüm olabilecek konumlarını göz önünde bulundurmadığı için hafıza kullanımını azaltır.
3. Markov algoritmasına göre daha konum belirlemede daha titizdir çünkü ortamdaki konumları ayrıklaştırmadan bir devamlı fonksiyon olarak görür.
4. Parçacık filtreleri robotun sensör ve hareket modellerinin her türünde kullanılabilir.
5. Parçacıklar kümesi sadece robotun gerçek konumuna bağlı olarak değişir ve hesaplanması kolaydır.
6. Programlanması kolaydır ve hızlı çalışır.

Öte yandan algoritmanın dezavantajları da vardır. Parçacık kümesi küçük olduğu zaman o kümenin içinde robotun gerçek konumu bulunamayabilir çünkü algoritmanın her adımında parçacıklar azalmaktadır. Bu problemi çözmek için bazı durumlarda parçacık kümesine rastlantısal olarak parçacıklar eklenebilir. Bu tür yaklaşım matematiksel olarak problemi çözemediğine rağmen bazı durumlarda işe yarar [8]. MCL algoritması global yer keşfetme problemini çözebilmesine rağmen “Kaçırılmış robot” problemini çözmez çünkü robotun

kaçırıldıktan sonraki yeri bizim parçacık kümemizde olmayabilir ve bu durumda algoritmanın her işleyişi adımında robotun gerçek konumuna yaklaşılmaz. [9]'da bu problemi çözebilmek için Karışık-MCL algoritması önerilmiştir. Karışık-MCL'de parçacık kümesini matematiksel olarak değiştirerek çözmektedir. Bu algoritma örneklenen parçacık kümesinin yenilenme sürecini tersine yapmaktadır. MCL'de hereket bilgilerini kullanarak yeni parçacıkların yeni konumunu belirler sonra ise her parçacığın ağırlık değerini günceller. İkili MCL'de ise bu adımlar tersine yapılır. Karışık-MCL bu iki yaklaşımı birleştirerek global yer keşfetme problemini çözer. Bu tür yaklaşımlar global yer keşfetme problemini daha kolay ve verimli çözmek için iyi strateji oluşturuyor. Sonuç olarak Monte-Carlo algoritması ne kadar parçacıkların seçilme kalitesine baksa da genel olarak robotun konumunu keşfetmek için stokastik yöntemlerinen en iyilerine olarak sayılır. Çünkü bu algoritma robotun sensör ve hareket hata ve bilgilerinin türüne bakmadan sonuca ulaşır.

6. ALGORİTMALARIN KARŞILAŞTIRILMASI

Tezde incelediğimiz algoritmaları karşılaştırmak için 3 kriter belirlenmiştir. Bunlar: ortamı ayrıklaştırma metodu, konum hakkında inancın saklanması, etkinlik. Bunları deneysel olarak görmek için bilgisayar ortamında Python dilinde program yazıldı. Bilgisayar özellikleri Intel Core i3-3220 3.3GHz, 8 GB RAM, Ubuntu 14.04 işletim sistemi. Yazılan programda görsel uygulama yapılarak algoritmalar çalıştırıldı. Her bir algoritmalar için değişik sensör ve hareket modeli belirlenerek belli ortamlarda çalıştırıldı. Genel olarak algoritmaların özellikleri Tablo 6.1'deki gibidir:

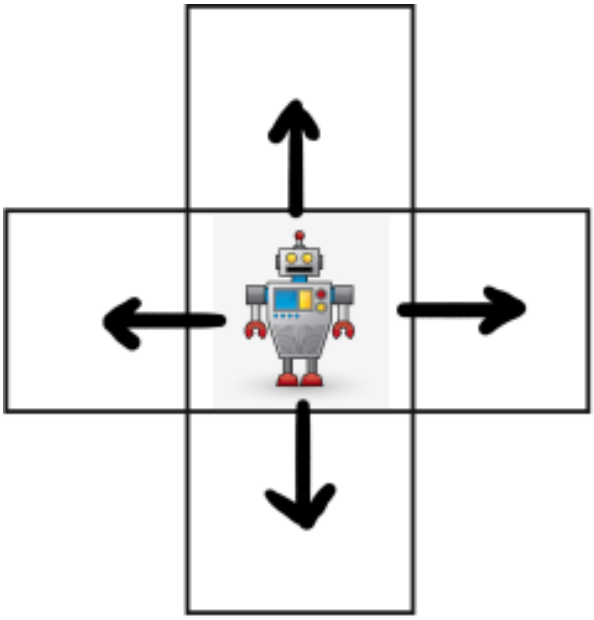
Tablo 6.1: Algoritmaların karşılaştırılması

Algoritma	Ortam	İnanç	Etkinlik	Çözdüğü Problem
Markov	Ayrıklaştırılmış	Çok modlu	Üstsel	GlobalYer Keşfetme
Kalman	Devamlı	Tek modlu	Kare	Konum takip etme
Monte-Carlo	Devamlı	Çok modlu	Etkin	Global Yer Keşfetme

Sonra bölümlerde algoritmaların çalışmasını görsel olarak karşılaştırarak her algoritmanın ne kadar sürede bittiğini ve ne kadar hafıza aldığını inceyerek karşılaştırılmıştır. Genel olarak her algoritma değişik problemleri çözmesine rağmen tüm algoritmalar seyyar ve otonom robotlar için büyük önem taşımaktadır. Bu konuda özellikle modern robotik biliminde ve teknolojide büyük gelişmeler kaydedilmektedir. Önceden kesin yani deterministik metodları kullanan algoritmalar üzerinde yoğun çalışmalar yürümekteydi [1]. Fakat ortamın ve robotun belirsizliklerinden dolayı statistiksel stokastik metodlar kullanılması gerektiği çok açıktır. Bu tezdeki algoritmaların karşılaştırılması robotlar için yer keşfetme algoritmalarının ne kadar önemli ve gerekli bir araç olduğunu ispatlar. Özellikle deneysel çalışmalar bu algoritmaların dezavantaj ve iyi yanlarını açık göstermektedir. Uygulamanın kaynak kodlarının bir kısmı tezde ek olarak verilmiştir fakat tüm olarak internet ortamında github.com sitesinde yayınlanmıştır. Uygulamalar robotun hareket ve sensör modellerinin hatalı ve hatasız durumlar için ayrıca yapılabilmektedir. Robot hatasız kabül edildiği zaman kendir yerini yüzde yüz keşfettiği gösterilmiştir. Hatalı olduğu durumlarda ise yanlış konumlar için olasılık dağılımları çok az bir şekilde artmıştır.

6.1. Markov algoritmasının uygulaması

Uygulamada kullandığımız robot üç ayrı rengi (Kırmızı, Sarı, Yeşil) hatasız algılayan robottur. Hareket olarak ise sadece (Sağ, Sol, Üst, Alt) olan komutlar ile o yönce bir birim hatasız olarak hareket ediyor [Şekil 6.1.1]. Uygulama için kullanılan ortam ise 10x10 ayrıklaştırılmış haritadır. Ortamdaki her ayrıklaştırılmış kare için bir renk verilir. Bu renk robot o konumda ne algıladığını gösterir. Gerçek ortamda bu renkler robot için sensörden gelen görsel, ultrason ya lazer bilgileri olacaktır, bizim durumda ise simulasyonu daha kolay yapabilmek için üç ayrı renk kullanılmıştır.



Şekil 6.1.1: Uygulamada kullanınılan robot modeli

Uygulama çalışmaya başlayınca robot kendi yeri hakkında bir bilgisi yoktur, ve bundan dolayı her kare için 0.01 değerinde olasılık atamıştır. Yani robot her yerde olabileceği anlamına gelir. Bu olasılık değerleri robot her hareket ettiğinde ve sensörlerinden bilgi alınca değişmektedir. Bu durum aşağıda gösterilmiştir [Şekil 6.1.2].

🟥 0.01	🟦 0.01	🟨 0.01	🟥 0.01	🟥 0.01	🟦 0.01	🟥 0.01	🟥 0.01	🟨 0.01	🟥 0.01
🟦 0.01	🟨 0.01	🟦 0.01	🟥 0.01	🟥 0.01	🟦 0.01	🟥 0.01	🟥 0.01	🟨 0.01	🟥 0.01
🟥 0.01	🟥 0.01	🟦 0.01	🟦 0.01	🟦 0.01	🟦 0.01	🟨 0.01	🟦 0.01	🟥 0.01	🟨 0.01
🟦 0.01	🟨 0.01	🟥 0.01	🟨 0.01	🟥 0.01	🟦 0.01	🟥 0.01	🟦 0.01	🟦 0.01	🟥 0.01
🟨 0.01	🟦 0.01	🟦 0.01	🟦 0.01	🟥 0.01	🟥 0.01	🟥 0.01	🟥 0.01	🟨 0.01	🟥 0.01
🟥 0.01	🟥 0.01	🟦 0.01	🟦 0.01	🟦 0.01	🟥 0.01	🟨 0.01	🟥 0.01	🟦 0.01	🟦 0.01
🟦 0.01	🟨 0.01	🟨 0.01	🟥 0.01	🟦 0.01	🟥 0.01	🟦 0.01	🟦 0.01	🟨 0.01	🟥 0.01
🟦 0.01	🟥 0.01	🟦 0.01	🟦 0.01	🟨 0.01	🟦 0.01	🟨 0.01	🟨 0.01	🟦 0.01	🟥 0.01
🟦 0.01	🟥 0.01	🟨 0.01	🟦 0.01	🟨 0.01	🟨 0.01	🟥 0.01	🟨 0.01	🟦 0.01	🟨 0.01
🟥 0.01	🟨 0.01	🟦 0.01	🟥 0.01	🟥 0.01	🟦 0.01	🟥 0.01	🟦 0.01	🟨 0.01	🟨 0.01

Şekil 6.1.2: Robotun hareket etmeye başlamadan önceki inancı

Robot bundan sonra 6 adım sağa hareket ederken her defasında sensörden bir renk okuyor, bu renklerin dizisi *['Sarı','Kırmızı','Mavi','Kırmızı','Mavi','Mavi']*'dır. Markov algoritmasının hareket güncellemesini (16) denklemini kullanarak yapılır. Ayrıca her sensör bilgisi için (9) denklemini kullanarak olasılık dağılımı güncellenir. Bu güncellemeleri yapmak için aşağıdaki tablodaki kaynak kodları kullanılmıştır:

Tablo 6.1.1: Markov algoritmasının hareket ve sensör güncelleme kodları

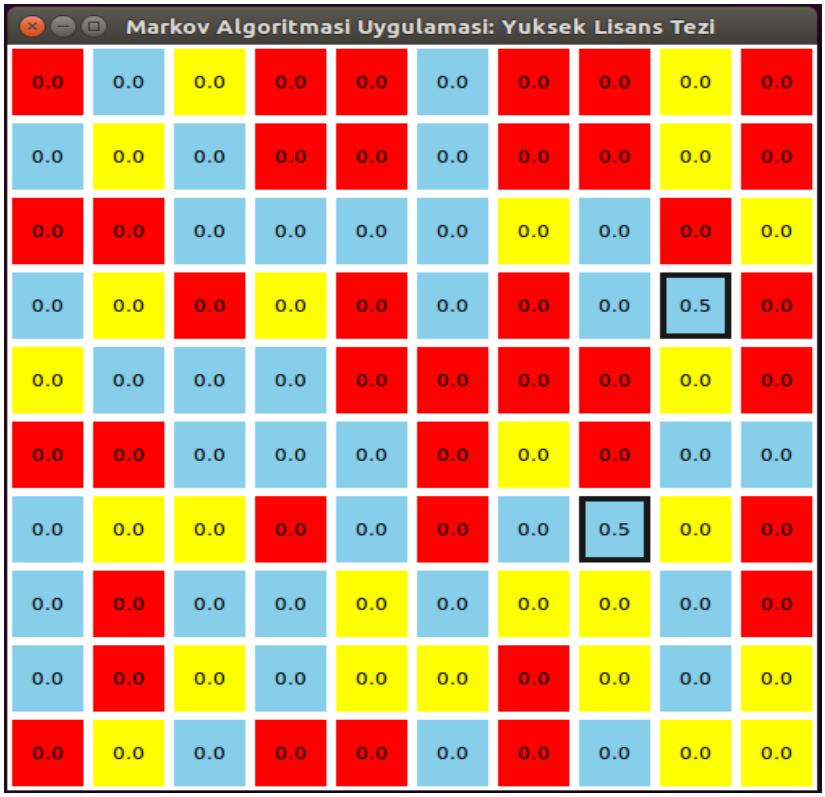
```
def sense(p, Z,colors,sensor_right):
   q= [[0 for row in range(len(colors[0]))] for col in range(len(colors))]
   for i in range(len(p)):
      for j in range(len(p[i])):
         hit = (Z == colors[i][j])
         q[i][j]=p[i][j] * (hit * sensor_right + (1-hit) * (1-sensor_right))

   s = sum(map(sum,q))
   for i in range(len(q)):
      for j in range(len(q[i])):
         q[i][j] = q[i][j] / s
   return q

def move(p, U,colors,p_move):
   q= [[0 for row in range(len(colors[0]))] for col in range(len(colors))]
   for i in range(len(p)):
      for j in range(len(p[i])):
         s = p_move * p[(i-U[0]) % len(p)][(j-U[1]) % len(p[i])]
         s = s + (1 - p_move)*p[i][j]
         q[i][j] = s
 return q
```

Sense fonksiyonu sensörden okunan *Z* rengi için *p* olasılık dağılımını değiştirir. Bu fonksiyonda algılanan renk ile aynı renkte olan karelerdeki olasılık değerlerini (9) denklemi ile günceller. *Sensor_right* değeri ise o sensörün yüzde kaç doğru çalıştığını gösterir. *Colors* dizisi ise ortamdaki her bir karenin renk değerini tutan bir matristir. Hesaplamalar yapılırken tüm kareler için olasılık

değerleri hesaplandıktan sonra ayrıca bu değerler normalize edilir çünkü ihtimal toplamlarının değeri bir olması gerekir. *Move* fonksiyonu ise robotun hareketine göre olasılık dağılımını kaydırır. *U* robotun nereye gittiğini gösteren vektör, *p* ise şu andaki olasılık dağılımıdır. Robot her hareket edince bu fonksiyon olasılık dağılımını o yönde kaydırır. Eğer robotun hareketlerinde hata payı çok ise bu fonksiyon belirsizliği arttırır eğer hata payı yoksa sadece olasılıkların yerleri değişir. Bizim durumda *Move* ve *Sense* fonksiyonu 6 defa her hareket ve sensör bilgisi için çalıştırılıp aşağıdaki gibi sonuçlar elde edilmiştir:



Şekil 6.1.3: Robot 6 defa sağa gittikten sonraki olasılık dağılımı

Gördüğünüz gibi bu durumda iki konum eşit olasılıkla robotun gerçek konumu için adaydır. Siyah çerçeve ile gösterilen ve ihtimalleri 0.5 olan kareler robotun gerçek konumu olmaya adaydır. Bunun nedeni *['Sarı','Kırmızı','Mavi','Kırmızı','Mavi','Mavi']* sensör dizisinin 6 defa sağa girerek algılanabilecek satırlar 4 ve 7'dir. Bu durumda robotun yeri tam olarak keşfedilememiştir çünkü iki konumun olasılık değeri aynıdır. Fakat robot eğer bir adım daha

sağa gidersek robotun konumu kesinleşecektir çünkü 0.5 değerindeki karelerin sağındaki kare rengi iki durumda da farklıdır. Bir adım sağa gittikten sonra robot kendi konumunu tam olarak keşfetmiştir ve bu durum Şekil 6.1.4'te gösterilmiştir. Bu örnekte robotun sensör ve hareketlerinin hata paylarının olmadığı durum gösterilmiştir ve robotun konumuna ilgisi olmayan kareler için 0 olasılık değeri hesaplanmıştır. Eğer robotun belli hata payları olsaydı bu değerler 0'dan farklı olurdu. Örneğin bazı durumlarda

robot sağa gitmeye çalışırken yerinde dikili kalsaydı bu durumda o konum için olasılık dağılımı robotun hata payı oranında artmış olacaktı. Aynı şekilde robotun Sarı renkte olan bir konumda Kırmızı algılasaydı, o zaman her Kırmızı karenin olasılık değeri sensörün hata payı oranında artmış olacaktır.

Şekil 6.1.4: Robot kendi konumunu keşfediyor

Yukarıdaki şekilde robot 7 adım sağa giderek sırasıyla *['Sarı','Kırmızı','Mavi','Kırmızı','Mavi','Mavi', 'Sarı']* renkleri algıladı. Arda arda bu renkler sadece 7'ci satırda olduğu için robot kendi konumunu tam olarak keşfetmiştir.

6.2. Kalman filtresinin uygulaması

Konum takip etme örneğinde ise robotu başlangıçta belli bir konuma (B) yerleştireceğiz. Ortamımın yine kare şeklinde 12x12 ortam olsun ve robotumuz bir birim mesafeye tüm yönlere hareket edebildiğini varsayalım. Robotun ilk baştaki konumu (4,12) olsun, yani 4 satırda 12 sütünda robotumuz olsun. Bu uygulamada robot hareket ederken belli odometri bilgilerini kullanarak ortamda ne kadar mesafe kat ettiğini bilecektir. Bizim örnekte robot bu bilgileri kullanarak her adımdan sonra [x,y] değerini hesaplamaktadır. Kalman filtresinin amacı bu bilgileri kontrol ederek doğrulamaktır. Bunun nedeni bazen sensör ve hareket bilgiler hatalı olduklarından dolayı robot kendisini başka yerde olduğunu düşünebilir. Örneğin robotun giderken tekerleri spin atmış ise robot kendisini spin attığı mesafe kadar gitmiş olduğunu düşünebilir. Bizim örnekte robotun odometri
bilgilerinden alınan değerler dizisi [[5., 10.], [6., 8.], [7., 6.], [8., 4.], [9., 2.], [10., 0.]] dir. Yani robot her hareketinden sonra [x,y] bilgisini yenilemektedir. Kalman filtresi sadece robotun konumunu doğrulamak ile kalmayıp robotun hızını da tespit etmektedir. Uygulama çalıştıktan sonra robotun tespit edilmiş konumu [9.99, 0.001] dir ve hızı *x* ekseninde 10 birim/saniye ve *y* -20 birimkare/saniye olarak hesaplanmıştır çünkü odometri ölçümleri her 0.1 saniyede güncellenmiştir. Tablo 6.3.2'de algoritmanın görsel uygulama sonucu gösterilmiştir.

6.3. Algoritmaların özellikleri

Algoritmalar çalıştırıldıktan sonra aşağıdaki gibi sonuçlar elde edildi. Bu sonuçlar gerçekçi olması için grafik arayüzü hesaba katılmadı ve her program bağımsız olarak çalıştırıldı.

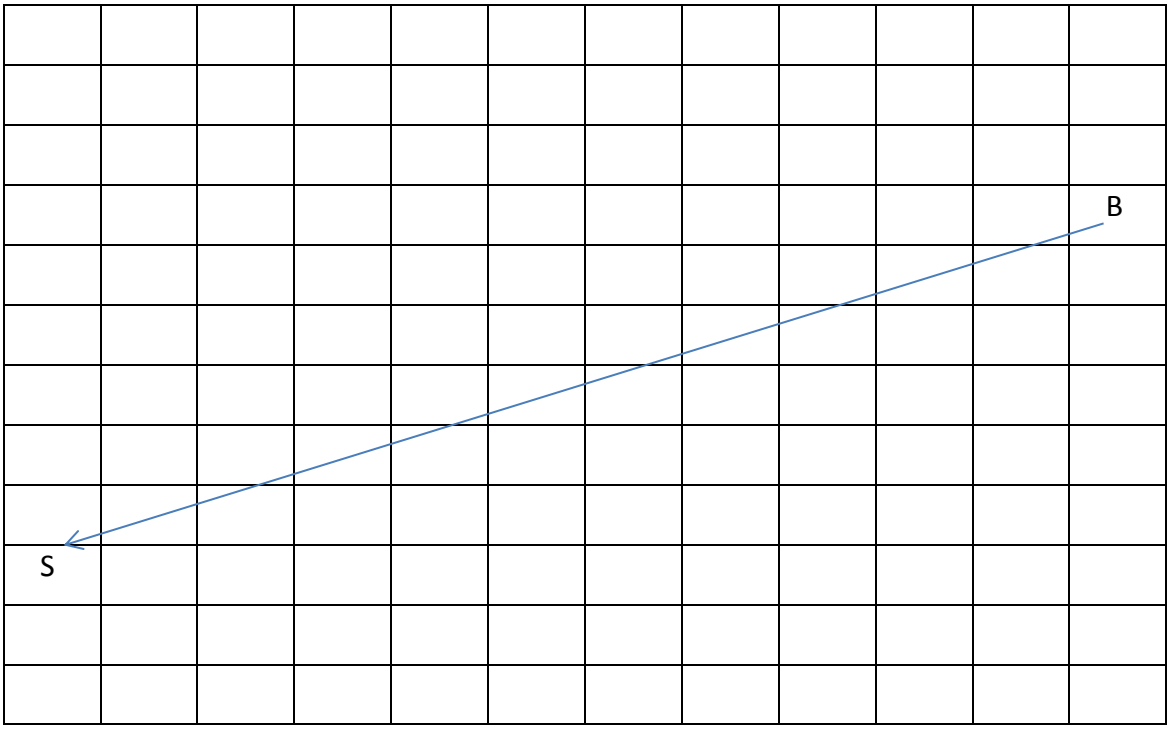
Tablo 6.3.1: Algoritmaların hafıza ve süre bilgileri

Algoritma	Hafıza	Süre
Markov	223 KB	560 ms

Kalman	60 KB	311 ms
Monte-Carlo	28 KB	602 ms

Yukarıdaki tabloda gördüğümüz gibi Markov algoritması hafızada çok yer kapladığına rağmen Monte-Carlo algoritmasından daha hızlı çalışmaktadır. Bunun nedeni Monte-Karlo algoritmasının parçacık kümesinin örneklenmesi ve daha sonra iyileştirilmesi tamamen rastlantısal olmasından. Öte yandan Monte-Carlo algoritması çok küçük parçacık kümesini kullanarak problemi çözebilmesinden dolayı hafıza kaybı daha azdır. Bu özellik gerçek zamanlı modern robotlar için çok önemlidir. Kalman filtresi ise sadece robotun konumunu takip ettiği için hem Markov algoritmasınan hem de Monte-Carlo algoritmasından daha hızlı çalışır.

Tablo 6.3.2: Kalman filtresi örneği



6.4. Sonuç

Bu tezde yer keşfetme problemlerini çözmek için algoritmalar incelenerek uygulamaları yapılmıştır. Konum takip etme probleminde robotun başlangıç konumu bilinmektedir ve bir kaç hareket sonrasında robotun konumunu kesinleştirilmesi gerekir. Robotun global yer keşfetme problemi ise robotun başlangıçtaki konumu bilinmemektedir ve robot başlangıçta hiçbir bilgisi olmadan kendi konumunu keşfetmesi gerekir. Bu durumda robot kendi konumu hakkında farklı tahmin ve ınanç oluşturmararak stokastik yöntemler ile problemi çözmeye çalışır. Örnek olarak 'Konum takip etme' problemini çözmek için Kalman filtresi ve Geliştirilmiş Kalman filtresi kullanılabilir [37]. Kalman filtresi algoritmasında robotun konum hakkında tahmin ve ınancı Gaussian dağılımı ile gösterilmektedir ve sorunu çözmek için robotun odometri bilgileri kullanılmaktadır. Fakat Kalman filtresi global yer keşfetme prolemini çözememektedir. 'Global Yer keşfetme' problemini çözmek için Markov modelini kullanan algoritma incelenip başka 'Yer keşfetme' algoritmaları ile karşılaştırılmaktadır. Markov yer keşfetme algoritması ortamda robotun olabilecek tüm konumlar için olasılık tahminlerini hafızasında tutmaktadır. Belli bir konum için olasılık değeri başka konumlara kıyasen yüksek değere ulaştığı zaman, robot kendi konumu keşfetmiş demektir. Markov modelinin problemi çözmeye yönelik stokastik yaklaşımı robotun konumu hakkında kesin bilgisi olmadığı durumlar için avantaj sağlamaktadır. Bu tür 'çok ihtimalli' yaklaşım robot için her zaman gereklidir çünkü modern otonom robotların sensör bilgilerindeki ve hareket sonuçlarındaki hata payları stokastik yaklaşımı gerektirmektedir. Algoritmadaki kullanılan ortamın haritasını ayrıklaştırma ve küçük parçalara bölme her konum için olasılıkları göze almayı sağlamaktadır. Başka yer keşfetme algoritmaları ise ortamın haritasını belli bölgeleri bölerek, bölge olasılığı bilgisini hafızasında tutmaktadır. Bu durumda robot bir bölgenin içindeki iki farklı konumda bile olsa algoritma robotun bir konumda olduğunu gösterir[36]. Monte-Carlo algoritmasi ise Markov algoritmasının zayıf taraflarını çözecek şekilde ayarlanmıştır. Örnek olarak Monte-Carlo algoritmasının robotun hafızasında sakladığı bilgi miktarı çok azdır. Bunun nedeni her bir konumun olasılık değerini tutmanın yerine sadece robotun gerçek konumuna yakın olan konulardaki parçacıkları hesaba katmasıdır.

Sonuç olarak her 'Yer keşfetme' problemlerin türleri için farklı algoritma kullanarak robotun verimliliği artırılabilir. Konum takip etme için Kalman filtresi kullanılırsa çok verimli olur, fakat aynı zamanda Markov algoritması kullanılabilir. Global yer keşfetme problemi için ise Monte-Carlo algoritması Markov algoritmasınan daha hızlıdır, ancak örneklem parçacıkların

özenle seçilerek ortamın özellikleri doğru seçilmelidir. Bu algoritmalar robotun 'Yer keşetmek' en önemli araçlardır ve robotun yer keşfetme kabiliyeti otonom ve seyyar olabilmek için en önemli şartlardandır.

KAYNAKLAR

[1] J. Borenstein, B. Everett, and L. Feng. D. Wehe. "Mobile Robot Positioning Sensors and Techniques", Journal of Robotic Systems 14(4), 231–249 (1997)

[2] Thrun S., "Probabilistic Algorithms in Robotics", CMU-CS-00-126, (2000)

[3] Thrun S., Burgard W., Fox D., "Active Mobile Robot Localization", Robotics, 1346-1352, (1996)

[4] Burgard W. et al., "Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localization Approach", International Conference on Intelligent Robots and Systems, (1998)

[5] Fox D., Burgard W., Thrun S., "Markov Localization for Mobile robots in Dynamic Environments", JAIR, 391-421, (1999)

[6] Thrun S., Burgard W., Fox D., "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots", Machine Learning and Autonomous Robots (joint issue), 31/5, 1–25 (1998)

[7] Thrun S. et al. "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva", The International Journal of Robotics Research Vol. 19, No. 11, pp. 972-999, (2000)

[8] Thrun S., "Particle Filters in Robotics", Uncertainty in AI, (2002)

[9] Thrun S., Fox D., Burgard W., Dellaert F., "Robust Monte Carlo Localization for Mobile Robots", IAAI, (2001)

[10] Fox D., Burgard W., Dellaert F., Thrun S., "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots", AAAI Proceeding. (1999)

[11] Dellaert F., Fox D., Burgard W., Thrun S., "Monte Carlo Localization for Mobile Robots", ICRA, (1999)

[12] Kose H., Celik B., H. Levent Akın. "Comparison of Localization Methods for a Robot Soccer Team", International Journal of Advanced Robotic Systems, Vol. 3, No. 4. (2006)

[13] Temizer S., Çağrı M., "Mesafe ölçümü tabanlı güvenilir konum tespiti

teknikleri ve kara ve hava araçlari için örnek uygulamalar", Havacilik ve uzay teknolojileri dergisi
cilt 6 sayi 2 pp.33-48, (2013)
[14] Qasem H., Ament C., Reindl L., "A new Particle Filter for Localization of a Mobile Base Station Based on Microwave Backscatter", Proceedings of the 3rd workshop on positioning, navigation and communication, (2006)
[15] Gustafsson G. et al., "Particle Filters for Positioning, Navigation and Tracking." IEEE Transactions on Signal Processing, (2002)
[16] Montemerlo M., Thrun S., Roller D., Wegbreit B.. "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges", Robotics, 1151-1156, (2002)
[17] Gadeyne K., Bruyninckx H., "Markov techniques for object localization with force-controlled robots", ICAR, (2001)
[18] Araneda A., Stephen E. Fienberg, Soto A., "A statistical approach to simultaneous mapping and localization for mobile robots", The Annals of Applied Statistics, Vol. 1, No. 1, 66–84 (2007)
[19] Adam J. Dean, Ryan D. Martini, and Sean N. Brennan. "Terrain-Based Road Vehicle Localization Using Particle Filters", American Control Conference, 236-241, (2008)
[20] Roy N., Thrun S., "Coastal Navigation with Mobile Robots." Robotics, 1043-1049, (1999)
[21] Georgiev A., Peter K. Allen. "Localization Methods for a Mobile Robot in Urban Environments", T-RO, (2004)
[22] Sırma YAVUZ et al., "Eş zamanli konum belirleme ve harita oluşturma amaçlı otonom bir robot", ELECO, (2006)
[23] Thrun S., "Bayesian Landmark Learning for Mobile Robot Localization", Machine Learning, (1997)
[24] Burguera A., González Y, Oliver Y.. "Mobile Robot Localization using Particle Filters and Sonar Sensors", Advances in Sonar Technology. ISBN 978-3-902613-48-6, pp. 232, (2009)
[25] Ivanjko E., Kitanov A., Petrovic I., "Model based Kalman Filter Mobile Robot Self-Localization", Robot Localization and Map Building, ISBN 978-953-7619-83-1, (2010)
[26] Milstein A., Sánchez J.N., Williamson E.T., "Robust Global Localization Using Clustered Particle Filtering", (2003)

[27] Fox D., Burgard W., Thrun S., “Active Markov Localization for Mobile Robots”, Robotics and Autonomous systems, Volume 25, Issues 3–4, Pages 195–207, (1998)

[28] Schulz D., Burgard W., “Probabilistic state estimation of dynamic objects with a moving mobile robot”, Robotics and Autonomous Systems 34, 107–115, (2001)

[29] Skrzypczynski P., “Simultaneous localization and mapping: a feature–based probabilistic approach”, Int. J. Appl. Math. Comput. Sci., Vol. 19, No. 4, 575–588, (2009)

[30] Kwok C., Fox D., Meila M., “Adaptive Real-time Particle Filters for Robot Localization”, ICRA. (2003)

[31] M. Di Marco, A. Garulli, A. Giannitrapani, A. Vicino. “A Set Theoretic Approach to Dynamic Robot Localization and Mapping”, AURO, (2002)

[32] Baltzakis H., Trahanias P. “Hybrid Mobile Robot Localization using Switching State-Space Models”, ICRA, (2002)

[33] Beeson P., Murarka A., Kuipers B., “Adapting Proposal Distributions for Accurate, Efficient Mobile Robot Localization”, ICRA, (2006)

[34] Martınez-Gomez J., Jimenez-Picazo A., Garcıa-Varea I., “A particle-filter-based self-localization method using invariant features as visual information”, CLEF, (2009)

[35] Nourbakhsh R.,, Siegwart R., “Introduction to Autonomous Mobile Robots”, MIT Press, (2004)

[36] Russell S., Norvig P., “Artificial Intelligence, A Modern Approach. Third Edition”, Prentice Hall, (2010)

[37] Fox D., Thrun S., Burgard W.. “Probabilistic Robotics”, MIT Press, (2005)

[38] Koenig S., “Robot Localization and Exploration with Agent-Centered Search”, International Joint Conference on AI, (1999)

[39] Chen L., Hu H., McDonald-Maier K.. “EKF based Mobile Robot Localization”, Third International Conference on Emerging Security Technologies, 149-154, (2012)

[40] H. Erickson L., Knuth J., Jason M. O’Kane, Steven M. LaValle. “Probabilistic localization with a blind robot”, (2008)

[41] Friedrich H., Dederscheck D., Krajsek K., Mester R., “View-based Robot Localization Using Spherical Harmonics: Concept and First Experimental Results”, LNCS 4713, pp. 21-31, (2007)

[42] Gasparri A., Panzieri S., Pascucci F., Ulivi G.. "Monte Carlo Filter in Mobile Robotics Localization: A Clustered Evolutionary Point of View", JIRS, (2006)

[43] Moreno L. et al. "A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors", JIRS, (2002)

[44] Gutmann J.S., Fox D., "An Experimental Comparison of Localization Methods Continued", (2003)

[45] Andreasson H., Treptow A., Duckett T., "Localization for Mobile Robots using Panoramic Vision, Local Features and Particle Filter", ICRA, (2005)

[46] Hornung A. et al. "Humanoid Robot Localization in Complex Indoor Environments", IROS, (2010)

[47] Howard A, Maja J Matari, G. S. Sukhatme. "Cooperative relative localization For mobile robot teams: an ego-centric approach", (2002)

[48] Agrawal M., Konolige K.. "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS", ICPR, (2006)

[49] Jensfeld P. et al. "Feature based condensation for mobile robots", ICRA, 2000

[50] Se S., Lowe D., Little J.. "Vision-based Mobile Robot Localization And Mapping using Scale-Invariant Features", ICRA, (2001)

[51] Lankenau A., Rofer T., "Mobile Robot Self-Localization in Large-Scale Environments", ICRA, 1359-1364, (2002)

[52] Kantor G., Singh S., "Preliminary Results in Range–Only Localization and Mapping", ICRA, (2002)

[53] Kosecka J., Li F., "Vision Based Topological Markov Localization", ICRA, (2004)

[54] Ankışhan H., Efe M., "Eşzamanlı konum belirleme ve harita oluşturmaya Kalman filtre yaklaşımları", Mühendislik dergisi, Cilt: 1, Sayı: 1, 13-20, (2010)

[55] Burgard W., Fox D., Hennig D., Schmidt T., "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids", Proc. of the Fourteenth National Conference on Artificial Intelligence, (1996)

[56] Koenig S., Joseph S.B. Mitchell, Mudgal A., Tovey C.. "A near-tight approximation algorithm for the robot localization problem", (2009)

[57] Kwok C., Fox D., Meila M.. "Real-time Particle Filters", Proceedings of IEEE, (2002)

[58] Miura J., Yamamoto K., “Robust View Matching-Based Markov Localization in Outdoor Environments”, IROS, (2008)

[59] Jason M. O’Kane, Steven M. LaValle. “Localization with Limited Sensing”, IEEE Transactions in Robotics, (2006)

[60] Jason M. O’Kane. “Global Localization Using Odometry”, (2006)

[61] M. T. L´azaro, J. A. Castellanos. “Localization of Probabilistic Robot Formations in SLAM”, ICRA, (2010)

[62] Levinson J., Thrun S., “Robust Vehicle Localization in Urban Environments Using Probabilistic Maps”, ICRA, (2010)

[63] Avots D., Lim E., Thibaux R., Thrun S., “A Probabilistic Technique for Simultaneous Localization and Door State Estimation with Mobile Robots in Dynamic Environments”, IROS, (2002)

[64] Turgut B., Richard P. Martin. “Restarting particle filters: an approach to improve the performance of dynamic indoor localization” (2005)

[65] Zhou H., Sakane S., “Sensor Planning for Mobile Robot Localization A Hierarchical Approach Using a Bayesian Network and a Paticle Filter”, IEEE Transactions on Robotics, vol. 24, no. 2, (2008)

[66] Koenig S., Mudgal A., Tovey C., “An Approximation Algorithm for the Robot Localization Problem”, (2003)

[67] R. G. Brown, B. R. Donald. “Mobile Robot Self-Localization without Explicit Landmarks”, Algorithmica 26: 515–559, (2000)

[68] Stergios I. Roumeliotis, George A. Bekey. “Bayesian Estimation and Kalman filtering: A unified framework for Mobile Robot Localization”, ICRA, 2985-2992, (2000)

[69] Kai O. Arras, José A. Castellanos, Siegwart R., “Feature-Based Multi-Hypothesis Localization and Tracking for Mobile Robots Using Geometric Constraints”, ICRA, (2002)

[70] Adams M., Zhang S., Xie L., “Particle Filter Based Outdoor Robot Localization Using Natural Features Extracted from Laser Scanners”, ICRA, (2004)

[71] Gutmann J.S., “Markov-Kalman Localization for Mobile Robots”, Digital Creatures Laboratory, Sony Corporation, (2002)

[72] Moreno L., Garrido S., Blanco D., “Mobile Robot Global Localization using an Evolutionary MAP Filter.” J Glob Optim 37:381–403, (2007)

[73] Sim R., Dudek G., “Mobile robot Localisation from learned Landmarks”, IROS, (1998)
[74] Matthew C. Deans, Martial Hebert. “Invariant Filtering for Simultaneous Localization and Mapping”, ICRA, (2000)
[75] G.Pires, A. Surrecio, U. Nunes, “A Simulation Environment for Robot Localization: application of Bayesian techniques.“Relatorio Tecnico, V1.0, (2006)
[76] Karlsson N., Enrico Di Bernardo, Ostrowski J., Goncalves L., Pirjanian P., Mario E. Munich. “The vSLAM Algorithm for Robust Localization and Mapping”, ICRA, (2005)

[77] Howard A., “Multi-robot Simultaneous Localization and Mapping using Particle Filters”, (2005)
[78] Varveropoulos V., “Robot Localization and Map Construction
Using Sonar Data”, (2000)
[79] Visser A, Sturm J., Groen F., “Robot companion localization at home and in the office”, (2007)
[80] Clark F. Olson, “Probabilistic Self-Localization for Mobile Robots”, IEEE transactions on robotics and automation, vol. 16, no. 1, (2000)
[81] R. Gonzalez, F. Rodriguez, J.L. Guzman, M. Berenguel. “Comparative Study of Localization Techniques for Mobile Robots based on Indirect Kalman Filter”, ISR, (2009)
[82] Michael J. Quinlan, Richard H. Middleton, “Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer”, (2009)
[83] Jean-Marc Valin, Michaud F., Rouat J., “Robust Localization and Tracking of Simultaneous Moving Sound Sources Using Beamforming and Particle Filtering”, (2006)
[84] James J.et al., “Framework for Natural Landmark-based Robot Localization”, Ninth Conference on Computer and Robot Vision, (2012)
[85] Ryan W. Wolcott, Ryan M. Eustice. “Visual Localization within LIDAR Maps for Automated Urban Driving”, (2014)
[86] Siagian C., Chang C.K., Itti L., “Autonomous Mobile Robot Localization and Navigation Using Hierarchical Map Representation Primarily Guided by Vision”, (2014)
[87] Kiriy E., Buehler M., “Three-state Extended Kalman Filter for Mobile Robot Localization” (2002)

[89] Sam T. Roweis, Ruslan R. Salakhutdinov. “Simultaneous Localization and Surveying with Multiple Agents”, Switching and Learning, LNCS 3355, pp. 313–332, (2005)

[90] Wang CC., Thorpe C., “Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects”, (2002)

[91] Fox D., Thrun S., Burgard W., "Particle filters for mobile robot localization", Sequential Monte Carlo Methods in Practice Statistics for Engineering and Information Science, pp 401-428, (2001)