MANAS Journal of Engineering MJEN 3 (1) 2015 21-34

Seyyar Robotlarda Kullanılan Stokastik Konum Belirleme Algoritmalarının Karşılaştırmalı Analizi

Ulan Brimkulov
Kırgızistan-Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bişkek, Kırgızistan
ulan.brimkulov@manas.edu.kg
Rayımbek Sultanov
Kırgızistan-Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bişkek, Kırgızistan
rayimbek.sultanov@manas.edu.kg
Ulan Bayaliev
Kırgızistan-Türkiye Manas Üniversitesi, Bilgisayar mühendisliği bölümü, Bişkek, Kırgızistan
bayalievu@yahoo.com
Received: *23.04.2015* *Reviewed:* *12.05.2015* *Accepted:* *14.05.2015*

Özet Seyyar robotların konum belirleme problemi robotun ortamdaki kendi konumunu bulmaya yönelik çalışmasıdır. Konum belirleme kabiliyeti seyyar ve otonom robotlar için önemli rol oynar çünkü robot amacına ulaşmak için konumunu bilmesi gerekir.
Konum belirleme probleminin birkaç türü vardır. En temel problem 'Konum takip etme'dir. Bu durumda robotun başlangıç konumu bilinmektedir ve bir kaç hareket sonrasında robotun konumunu doğrulamaya 'Konum takip etme' denir. Global konum belirleme ise daha zordur, çünkü bu problemde robotun başlangıçtaki konumu bilinmemektedir ve ayrıca robotun hareket ve sensörlerinde hata olabilir. Bu durumda robot sensör ve hareket bilgilerini kullanarak konumu hakkında farklı tahmin ve ınanç oluşturduktan sonra problemi stokastik yöntemler ile çözmesi gerekir.
Bu çalışmada robotun konum belirleme problemini çözen stokastik algoritmalar incelenip birbiri ile karşılaştırılmaktadır. Bunu yapmak için Markov, Kalman ve Monte-Carlo algoritmalar için görsel uygulama yapılarak problemi çözmedeki geçerliliği gösterilmektedir.

Anahtar sözcükler *Seyyar Robot, Konum belirleme, Markov, Kalman, Monte-Carlo, Stokastik yöntemler.*

A Comparative Analysis of Stochastic Algorithms Used for Mobile Robot Localization

Abstract The problem of mobile robot localization is the problem of finding robots position in its environment. Localization ability plays important role for mobile and autonomous robots because robot must know its position to reach the goal.
There are several types of localization problem. The fundamental one is 'Position tracking'. In this case the initial position of the robot is known and the problem is at correcting its position after moving some steps is 'Position tracking'. Global localization is harder because in this problem the robots initial position is not known and the movement and sensing of the robot may be erroneous. In this case robot must use the movement and sensor information to generate some belief about its position and solve the problem using stochastic methods.
This study analyzes and compares stochastic algorithms for solving robot localization problem. To achive this, visual applications for Markov, Kalman and Monte-Carlo algorithms are implemented and their validity in solving the robot localization problem is shown.

Keywords *Mobile Robot, Localization, Markov, Kalman, Monte-Carlo, Algorithm, Stochastic methods.*

1. GİRİŞ

Robotların konum belirleme problemi robotun ortamdaki kendi konumunu bulmaya yönelik çalışmasıdır. Konum tespit etme 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].

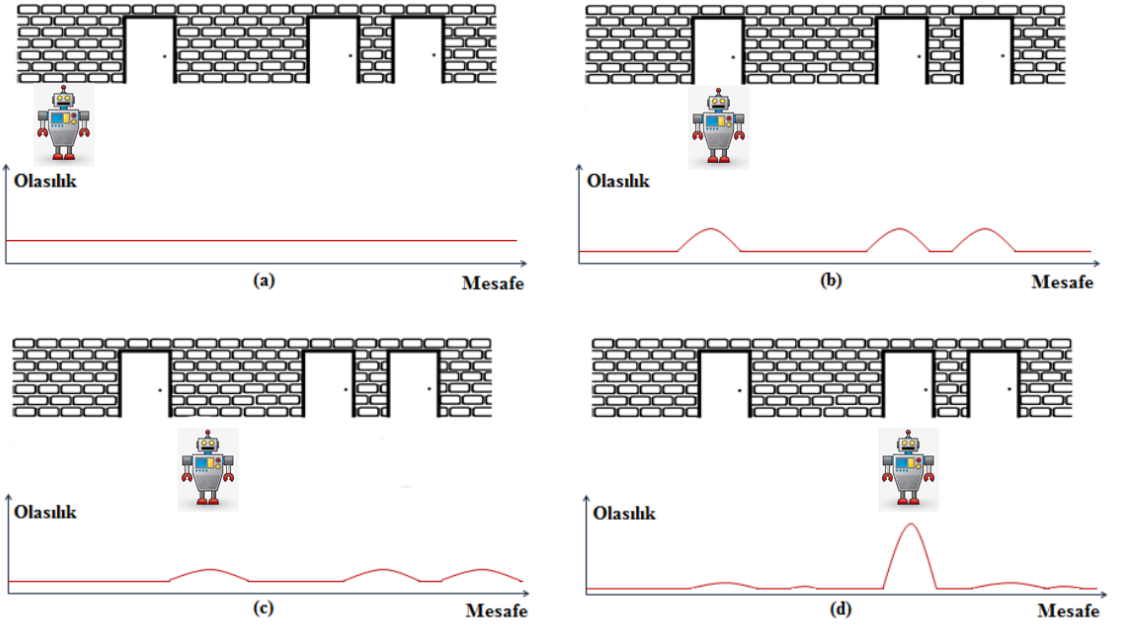
Robot konum belirleme probleminin birkaç türü vardır [2]. En temel konum belirleme problemi 'Konum takip etme'dir. Bu durumda robotun başlangıç konumu bilinmektedir ve bir kaç hareket sonrasında robotun konumunu kesinleştirme 'Konum takip etme' problemidir. Robotun 'Global konum belirleme' problem ise 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 [22]. Bu problemde robot bildiği ortam ve konumdan 'kaçırılarak' başka konuma yerleştiriliyor. 'Kaçırılmış robot' probleminin 'Global konum belirleme' probleminden farkı robot kendisinin farklı bir konumda olduğunu zannetmesidir. 'Global Konum belirleme' probleminde ise robotun kendi bilgisizliği hakkında haberdardır. Yukarıdaki problemleri çözmek için çok algoritma mevcuttur [23]. Örnek olarak Konum takip etme' problemini çözmek için Kalman filtresi ve Geliştirilmiş Kalman filtresi kullanılabilir [16]. Kalman filtresi algoritmasında robotun konum hakkında tahmin ve ınancı Gaussian dağılımı ile gösterilmektedir ve sorunu çözmek için robotun odometri bilgileri kullanılmaktadır.

Bu makalede 'Konum belirleme' problemini çözmek için Markov modelini kullanan algoritma incelenip başka 'Konum belirleme' algoritmaları ile karşılaştırılmaktadır. Markov konum belirleme 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 problem çözmeye yönelik stokastik yaklaşımı robotun konumu hakkında kesin bilgisi olmadığı durumlar için avantaj sağlamaktadır [3]. 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 yönden kullanılmasını sağlamaktadır. Başka konum belirleme 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 [4]. Bunun dışında özel işaret kullanan 'Konum belirleme' algorimaları mevcuttur. Bu tür algoritmalarda robot kendi yerini özel işaretlere bağıntılı olarak saptamaktadır [17]. Fakat robot her zaman özel işaretleri göremeyebilir ve görse bile bazen ortamdaki başka nesnelerden ayırmayablir [15].

Bu çalışmada yukarıdaki sorunları çözmek için stokastik yöntemleri kullanan Markov konum belirleme algoritması incelenip görsel uygulaması yapılmaktadır. Sonuç olarak başka stokastik konum belirleme 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.

1.1. Konum belirleme örneği

Markov konum belirleme 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 1.1.1]. Robotumuz ise sadece sağa hareket edebildiğini ve hareket hata payını şimdilik olmadığın 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 konum belirleme algoritmasında bu durumu her konum için aynı ihtimali vererekgösterilmektedir [Şekil 1.1.1.a]. Bir adım sağa gittikten sonra robotun sensörü kapı algıladığını varsayalım. Bu bilgileri kullanarak robotun konumlar arasındaki olasılık dağılımı kapılar arasında eşit olacaktır [13]. Bunun anlamı robot hangi kapının yanında olduğunu saptayamamıstır ve robotun elindeki bilgi konumunu saptamak için yeterli değildir [Şekil 1.1.1.b]. Bir adım daha gittikten sonra robot duvalı görmektedir ve olasılık dağılımı kapılardan sonraki duvallarda eşit olarak görülmektedir. Bu işlem olasılık dağılımların sağa kayması olarak da görülebilir[Şekil 1.1.1.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.



Şekil 1.1.1: Konum belirleme örneği

2. TEMEL KAVRAMLAR

Algoritmayı detaylı olarak incelemeye başlamadan önce temel kavramlara bakalım. İlk başta robotun konumunu *l* değişkeni ile gösterelim. l_t ise robotun *t* zamanındaki 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

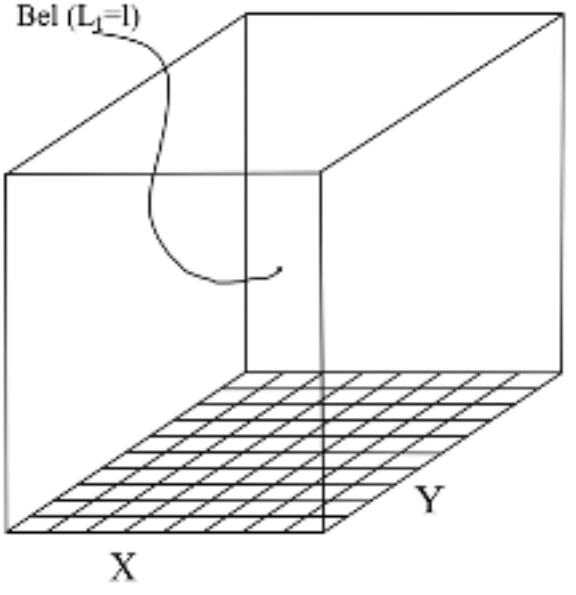
hakkında belli bilgi edinirse, ve ikincisi robot hareket sonrasında yer değiştirirse [5]. Sensör aracılığı ile alınan bilgiye *s* diyelim, ve robotun hareketlerine *a* diyelim. Bunların ilgili rastsal değişkenleri *S* ve *A* olsun. Robot zaman içinde sensör ve hareket bilgilerini bir dizi içinde algılamakta olduğunu varsayalım.

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

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

2.1. Ortam haritasının ayrıklaştırılması

Değişik konum belirleme algoritmaları ortamı ayrıklaştırmak için haritayı topolojik özelliklerine bölmektedir [6]. Bu tür ayrıklaştırma türü konum belirleme 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 [15]. Bu varsayımlardan dolayı yapılandırılmamış ortamları topolojik olarak konumlara ayrıklaştırmak çok zor ve verimsiz oluyor. Markov konum belirleme algoritmasında ise ortamdaki en ince konumları bile göz önünde tutmak için haritayı ızgara gibi parçalara ayrıklaştırmaktadır [Şekil 2.1.1]. 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 2.1.1: Ortamın 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.

2.2. Hareket modeli

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 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:

$$P((x+1,y) \mid \text{'sağa git'}, (x,y)) = 0.9 \quad (2)$$
$$P((x,y) \mid \text{'sağa git'}, (x,y)) = 0.1 \quad (3)$$

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].

2.3. Sensör modeli

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:

$$P(\text{'Sarı renk algıladım'} \mid \text{'Sarı renkte olan tüm konumlar'}) = 0.25 \quad (4)$$

Farkettiğiniz üzere 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 sensor 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:

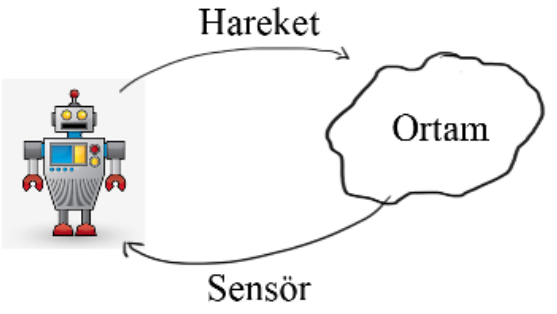
$$P(\text{'Sarı renk algıladım'} \mid \text{'Sarı renkte olan tüm konumlar'})$$
$$P(\text{'Kırmızı renk algıladım'} \mid \text{'Kırmızı renkte olan tüm konumlar'}) \quad (5)$$
$$P(\text{'Mavi renk algıladım'} \mid \text{'Mavi renkte olan tüm konumlar'})$$

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

3. KONUM BELİRLEME ALGORİTMALARI

Konum belirleme algoritmalarının genel olarak iki adımı vardır: birincisi sensör güncellemesi, ikinci adımı ise hareket güncellemesidir. Sensör günceleme adımında robot

ortamdan alınan bilgiye göre kendisinin konumu hakkındaki inancını günceller. Genellikle bu adımda robotun belli konuma olan inancı artarak biraz daha kesinleşir çünkü ortamdan alınan bilgi robotun konumunu belirlemek için önemli katkı sağlar. Hareket güncelleme adımında ise robotun konum ile ilgili olasılık dağılımı robotun hareket ettiği yönde kayar. Eğer robotun hareketlerinde hata payı varsa bu adımda robotun belli konuma olan inancı azalır. Bu iki adım bir konumdaki olasılık değeri başka konumlardan fazla oluncaya kadar devam eder.



Şekil 3.1: Konum belirleme algoritmasının adımları

Bu çalışmada konum belirmelek için üç algoritma incelenir. Markov, Kalman, ve Monte-Carlo. Markov algoritmasında robot her konum için olasılık değeri atayarak o değerlerin tümünü her adımda günceller. Kalman agoritmasında ise tüm değişkenler Gaussian dağılımı olarak modellernir ve sadece bir dağılım güncellenir. Monte-Carlo algoritmasında ortamdaki konumlar rastgele seçilerek robotun konumu tahmin edilmeye çalışılır. Sonraki bölümlerde bu algoritmalar detaylo olarak incelenir.

3.1. Markov algoritması

Algoritmanın tamamı Tablo 3.1.1'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 konum belirleme 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. Sensör güncelleme adımında 'Bayes' kuralı kullanarak robotun konuma olan inancını gösteren olasılık dağılımı güncellenir. Hareket güncelleme adımında ise 'Toplam olasılık' kuralı kullanılır. 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 konumunu keşfetmiş olur. Bazı durumlarda bu ihtimalin belli bir sınırı geçmiş olması istenir [7].

Tablo 3.1.1: Markov algoritmaının sözde kodu.

Ortamdaki her konum *l* için başlangıç değerini ata:

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

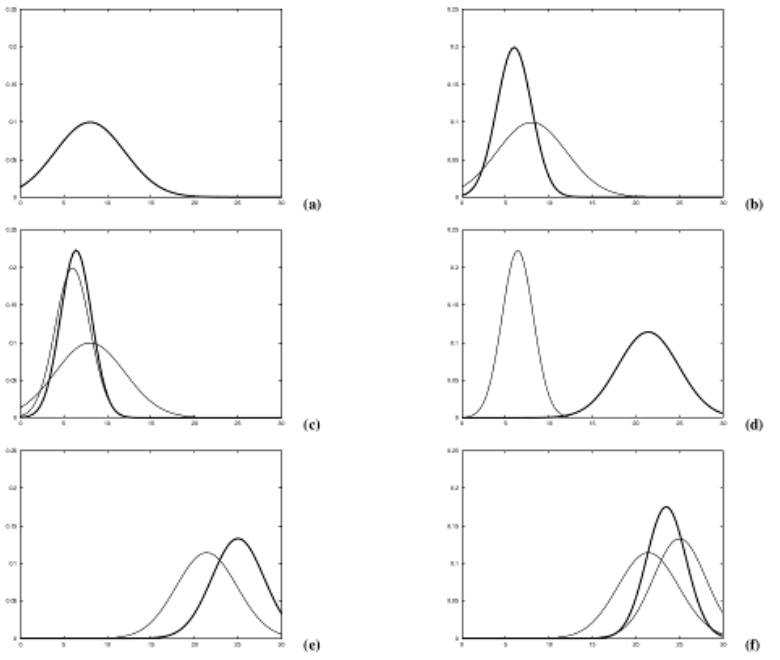
Bir konunum ihtimal değeri diğerlerinen fazla oluncaya kadar tekrar et:
Eğer gelen veri s_T sensör verisi ise:
$\alpha_T = 0$
Ortamdaki her konum l için:
$Bel(L_T=l) = P(s_T \mid l) * Bel(L_{T-1}=l)$
$\alpha_T = \alpha_T + Bel(L_T=l)$

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

3.2. Kalman algoritması

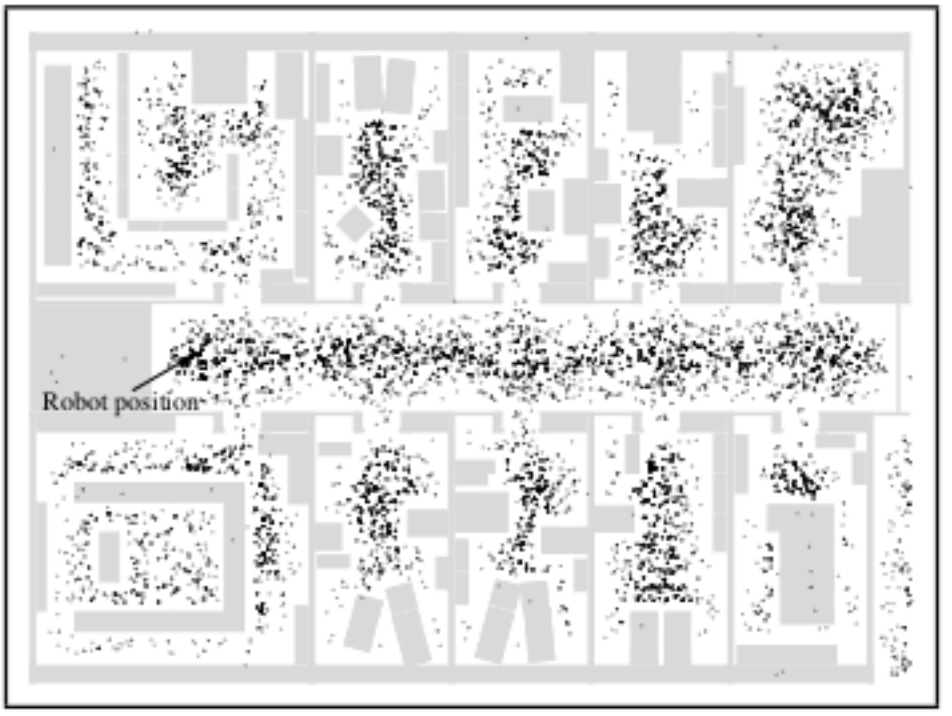
Kalman filtersi normalde sinyal işlemede kullanılan gürültüyü kaldırma metodudur. Bizin durumda ise robotun sensör ve hareketteki hata paylarını gürültü olarak kabül ediyoruz. Yani robot belli bir sensör verisi elde ettiği zaman belli bir oranda hatası olabilir. Örnek olarak robot bir konumda sensörü 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 [20]. Ayrıca robotun ilk baştaki konumun da bilinmesi şarttır çünkü algoritma çalışmaya başlamadan önce robot kendi konumu hakkındaki inancı Gaussian olarak modellenmesi gerekir. Kalman filtresi konum takip etmek ve sensör bilgilerini robotun konum olasılık dağılımı hesabına katmak için ideal araçtır. Markov algoritmasında da olduğu gibi Kalman filtresi algoritmasının iki adımı vardır: 'Hareket güncellemesi' ve 'Sensör güncellemesi'. [Şekil 3.2.1.f]'te gördüğünüz gibi Hareket güncellemesi olasılık dağılımındaki belirsizliği arttırır, Sensör güncellemesi ise olasılık dağılımındaki belirsizliği azaltır [Şekil 3.2.1.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.



Şekil 3.2.1: Kalman filtresinde 'Hareket' ve 'Sensör' güncellemesi [21]

3.3. Monte-Carlo algoritması

Markov algoritması global konum belirmede kullanabilir olmasına rağmen ortamın çok küçük parçalara ayrıklaşmasından dolayı hesaplama zorlukları vardır. Öte yandan Kalman filtresi robotun inancını bir Guassian olarak kolay hesaplamasına rağmen global konum belirleme problemini çözemiyor. Monte-Carlo algoritması hem global konum belirleme problemini çözebilmektedir hem de Markov algoritmasından daha hızlıdır [9][10]. Bunun nedeni ortamdaki her konum için olasılık hesabı yapmak yerine algoritma belli parçacık sayısı üzerinde hesaplamaları yapmaktadır. 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 [11][12]. 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 3.3.1'de ağır parçalar daha koyu renkte, hafif parçalar ise daha açık renkte gösterilmektedir.



Şekil 3.3.1: Robot hareket etmeye başlamadan önceki parçacıklar [8]

Monte-Carlo algoritmasında robotun konum hakkındaki inancı *Bel(x)* m tane ağırlıklı parça ile gösterilir. Yani:

$$Bel(x) = \{x_{(i)}, p_{(i)}\}_{i = 1,2,...m}$$

yukarıdaki denklemde her $x_{(i)}$ bir konumdur, $p_{(i)}$ ise o konum için pozitif ağırlık değeridir. Global konum belirleme probleminde, robot hareket etmeden önce bu parçalar robotun tüm olabilecek konumlardan rastsal olarak seçilerek ağırlıkları 1/m olarak atanır [14]. Sonra 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.

4. ALGORİTMALARIN KARŞILAŞTIRILMASI

Çalışmada incelenen algoritmaları karşılaştırmak için 3 kriter belirlenmiştir. Bunlar: ortamı ayrıklaştırma metodu, konum hakkında inancın saklanması ve algoritmanın etkinliği. Bu kriterleri 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 4.1'deki gibidir:

Tablo 4.1: Algoritmaların karşılaştırılması

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

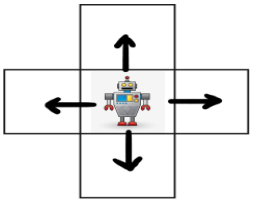
Sonra bölümlerde algoritmaların çalışmasını görsel olarak karşılaştırarak her algoritmanın ne kadar sürede bittiğini ve ne kadar hafıza aldığını inceyerek karşılaştırılmıştır. Algoritmalar çalıştırıldıktan sonra aşağıdaki gibi sonuçlar elde edildi. Bu sonuçlar gerçekçi olması için grafik arayüzü hesaba katılmadı ve her program bağımsız olarak çalıştırıldı.

Tablo 4.2: Algoritmaların hafıza ve süre bilgileri

Algoritma	Hafıza	Süre
Markov	223 KB	560 ms
Kalman	60 KB	311 ms
Monte-Carlo	28 KB	602 ms

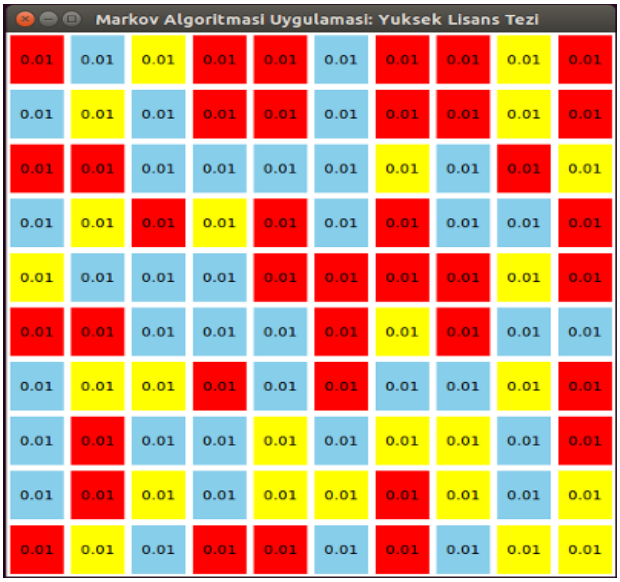
4.1. Algoritmaların uygulaması

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 4.1.1]. 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 4.1.1: Uygulamada kullanılan 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 4.1.2].



Şekil 4.1.2: Robot 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.

Şekil 4.1.3: Robot aldı adım saga gittikten sonra inancı

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 4.1.4'te gösterilmiştir.

Şekil 4.1.4: Robot 7 adımdan sonar kendi konumunu belirliyor

5. SONUÇ ve TARTIŞMA

Bu makalede konum belirleme 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 konum belirleme 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 [16]. Fakat Kalman filtresi global konum belirleme prolemini çözememektedir. 'Global Konum Belirleme' problemini çözmek için Markov modelini kullanan algoritma incelenip başka algoritmalar ile karşılaştırılmaktadır. Markov konum belirleme algoritması ortamda robotun olabilecek tüm konumlar için olasılık tahminlerini hafızasında tutmaktadır [19]. 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 konum belirleme 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 [18]. 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 'Konum belirleme' 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 konum belirleme problemi için ise Monte-Carlo algoritması Markov algoritmasınan daha hızlıdır, ancak örneklem parçacıkların özenle seçilerek ortamın özellikleri doğru seçilmelidir. Bu algoritmalar tümü robotun konum tespit etme kabiliyeti için en önemli araçlardır ve robotun otonom ve seyyar olabilmek için en önemli şartlardandır.

REFERANSLAR

[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]. 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)

[15]. James J.et al., "Framework for Natural Landmark-based Robot Localization", Ninth Conference on Computer and Robot Vision, (2012)

[16]. Gutmann J.S., "Markov-Kalman Localization for Mobile Robots", Digital Creatures Laboratory, Sony Corporation, (2002)
[17]. Thrun S., "Bayesian Landmark Learning for Mobile Robot Localization", Machine Learning, (1997)
[18]. Miura J., Yamamoto K., "Robust View Matching-Based Markov Localization in Outdoor Environments", IROS, (2008)
[19]. 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)
[20]. Ankışhan H., Efe M., "Eşzamanlı konum belirleme ve harita oluşturmaya Kalman filtre yaklaşımları", Mühendislik dergisi, Cilt: 1, Sayı: 1, 13-20, (2010)
[21]. Nourbakhsh R.,, Siegwart R., "Introduction to Autonomous Mobile Robots", MIT Press, (2004)
[22]. Russell S., Norvig P., "Artificial Intelligence, A Modern Approach. Third Edition", Prentice Hall, (2010)
[23]. Fox D., Thrun S., Burgard W.. "Probabilistic Robotics", MIT Press, (2005)