ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

QUANTIZED FEEDBACK CONTROL OF AUTONOMOUS ROBOTS

Ph.D. THESIS

Onur ŞENCAN

Department of Control and Automation Engineering

Control and Automation Engineering Programme

JULY 2018

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

QUANTIZED FEEDBACK CONTROL OF AUTONOMOUS ROBOTS

Ph.D. THESIS

Onur ŞENCAN
(504092108)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Prof. Dr. Hakan TEMELTAŞ

JULY 2018

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM ROBOTLARIN NİCEMLENMİŞ GERİBESLEMELİ KONTROLÜ

DOKTORA TEZİ

Onur ŞENCAN
(504092108)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Hakan TEMELTAŞ

TEMMUZ 2018

Onur ŞENCAN, a Ph.D. student of ITU Graduate School of Science Engineering and Technology 504092108 successfully defended the thesis entitled "QUANTIZED FEEDBACK CONTROL OF AUTONOMOUS ROBOTS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Hakan TEMELTAŞ** Istanbul Technical University	....
Jury Members :	**Prof. Dr. Işıl BOZMA** Boğaziçi University	....
	Prof. Dr. Müjde GÜZELKAYA Istanbul Technical University	....
	Prof. Dr. Veysel GAZİ Marmara University	....
	Assoc. Prof. Dr. Tufan KUMBASAR Istanbul Technical University	....

Date of Submission : **8 June 2018**
Date of Defense : **18 July 2018**

To my dear spouse Elif and my dear family,

FOREWORD

I wish to express my sincere gratitude to my supervisor Prof.Dr. Hakan Temeltaş, for his supervision, generous support and guidance provided during the research and also providing access to the ITU Robotics Laboratory facilities during my research. I would also like to express my sincere thanks to the academic and technical staff in the Department of Control Engineering for their support.

I also wish to express my sincere thanks for valuable contributions and comments made by my thesis progress committee members Prof.Dr. Işıl Bozma and Prof.Dr. Müjde Güzelkaya along with the thesis defense committee members Prof.Dr. Veysel Gazi and Assoc.Prof.Dr. Tufan Kumbasar.

I would like to thank all my friends and colleagues for their support and indulgence even in very harsh moments during these years.

Above all, I would like to express my gratefulness to my wife and my family for their support, tolerance, and sacrifice of our spare time of uncountable hours.

July 2018 Onur ŞENCAN
(M.Sc.)

TABLE OF CONTENTS

Page

ABBREVIATIONS

QFC	**:** Quantized Feedback Control
LQR	**:** Linear Quadratic Regulator
qLQR	**:** Quantized Linear Quadratic Regulator
PID	**:** Proportional Integral Derivative (Controller)
ROI	**:** Region of Interest
LTI	**:** Linear Time Invariant (System)
CLF	**:** Control Lyapunov Functions
3D	**:** Three Dimensional
DOF	**:** Degree of Freedom
LIDAR	**:** Light Detection And Ranging
LADAR	**:** Laser Detection And Ranging
RADAR	**:** Radio Detection And Ranging

SYMBOLS

e(t)	: Continuous error signal
$\mathbf{e_s(t)}$	: Piecewise constant error signal
e(kT)	: Discrete error signal
$\mathbf{h_0}$	: Zero-order hold function
$\delta(\mathbf{t})$	: Impulse function
$\Re^{\mathbf{k}}$	: k-dimensional Euclidean space
$\mathscr{P}_{\mathbf{fin}}$	: Finite subset operator
q	: Quantization operator
f(x,u)	: Nonlinear system characteristic function
h(x)	: Nonlinear measurement function
k(x)	: Control law on state feedback
G(s)	: System transfer function
$\mathbf{G_x,G_y,G_z}$	: Graph trees associated with x, y, z
idx,idy,idz	: Spatial graph identifiers of x, y, z
γ	: Sampling rate of corresponding node
$\mathbf{B_{i,n}(x)}$	: Bernstein polynomial
$\beta_{\mathbf{i}}$	: Bernstein coefficient
$\mathbf{C_n}$	: Linear combination of Bernstein polynomials
$\mathbf{P_i}$	: Bezier control points
υ	: Linear velocity of the vehicle
ϕ	: Heading angle of the vehicle

LIST OF TABLES

Page

LIST OF FIGURES

QUANTIZED FEEDBACK CONTROL OF AUTONOMOUS ROBOTS

SUMMARY

In this dissertation, a new robot navigation framework is introduced to close the gap between the environment mapping, path planning computations and the vehicle control problem. General approaches for robot navigation deals with the path planning and navigation with the assumption that the control system of the robot can track the given reference values of the planners. On the other hand, the control system architecture assumes that the incoming reference signals are suitable for its control range or commonly thresholds the out of range signals. In both situations, these systems act separately from each other, and there is no backwards information passed through these systems. As a result, even the path and motion planner outputs looks like optimal outcomes, if these outputs do not have any compliance with the control system, then the total system behavior is not accomplished as expected.

To ensure connection with the planning and control structures and establish a feedback mechanism to the overall system, navigation framework employs the quantization operator both in the environment mapping and in the controller architecture. Using the quantizer in grid segmentation problem leads to a novel approach within occupancy grid literature. By using computer graphics methodologies, an efficient occupancy grid method is derived for assigning the point clouds that are gathered from 3D sensors to the grids. This assignment uses a hybrid approach, which combines the topological and grid space partitioning methods. Geometric properties of the grid voxels are interpreted as quantized spatial information from the environment and used as an input to the motion planner that generates reference signals to the controller side of the robotics system.

The first main part of the navigation framework is called as input quantizer. Sensor inputs and the path splines are represented with the independent graph nodes for each axis in the 3D Euclidean space. Space partitioning methodology is executed with a binary segmentation independent for each axis until the corresponding node of the path splines and the obstacle nodes have no intersection. This binary segmentation and its reverse operation are called as zoom-in and zoom-out properties respectively in the study.

After the motion planner that shapes the reference signals with using the information from the grid voxels, another main block of the framework is in charge of synthesizing an asymptotically stable controller, which is called output quantizer. As its name suggests, this structure also consists of a quantizer, which is denominated as qLQR. This method is adapted to the proposed framework to satisfy both the stability and the performance concerns.

OTONOM ROBOTLARIN NİCEMLENMİŞ GERİBESLEMELİ KONTROLÜ

ÖZET

Bu tez çalışmasında, ortam haritalanması, yol planlaması ve araç kontrol sistemi arasındaki bağlantının sağlandığı, yenilikçi bir robot yöngüdümü taslağı sunulmuştur.

Robot yöngüdüm problemi ile ilgili literatürdeki genel yaklaşımlar, yol planlama ve navigasyon sonuçlarının araç kontrol sisteminin(motor sürücü ve eyleyiciler) bu çıktılara riayet edeceği varsayımına dayanmaktadır. Bununla birlikte, kontrol sistem mimarisi de yol planlama bloğundan gelecek olan referans sinyalinin kontrol edilebilir sınırlarda olduğu varsayımıyla tasarlanır ve bu sınırları aşan referans işaretleri limitlenerek sistem davranışının kararlı halde tutulması sağlanır. Her iki durumda da, yol planlama ve kontrol sistemleri birbirlerinden bağımsız olarak hareket etmekte ve bu sistemler arasında geriye yönelik bir bilgi akışı bulunmamaktadır. Sonuç olarak yol planlayıcılar optimal yol planları üretseler dahi bu referansların kontrol sistemi tarafından takip edilememesi durumunda beklenen sistem yanıtı alınamayacak ve sistem optimalden uzaklaşarak kararsızlığa doğru gitme eğilimi gösterecektir. Bu davranış genellikle robot sistemlerinde verilen yol referansında testere hareketi olarak kendini göstermektedir. Yüksek hızlara çıkıldığında ise bu testere hareketi sistemi kararsızlığa iterek robotun yoldan çıkmasına neden olacaktır.

Bu çalışmada yol planlama ve kontrol sistemleri arasında bir bağlantı kurmak ve sistemler arasında bir geribesleme döngüsü oluşturmak için navigasyon taslağında hem ortam haritalandırılmasında, hem de kontrol mimarisinde nicemleme işleci kullanılmaktadır. Nicemleyicinin ızgara bölütleme probleminde kullanılması ile, yöngüdüm mimarisinden bağımsız olarak literatürdeki doluluk ızgarası yöntemlerine yeni bir katkı sağlanmıştır. Bilgisayar grafiklerinde kullanılan yöntemlerin de yardımıyla, üç boyutlu sensörler tarafından elde edilen nokta bulutlarının doluluk ızgaralarına aktarılması için etkin bir yöntem ortaya konulmuştur. Bu yöntemde karma bir yaklaşım kullanılarak, topolojik ve ızgaralama ile bölütleme yöntemlerinin öne çıkan taraflarından hem hesaplama hem de bellek optimizasyonu amacıyla faydalanılmıştır. Doluluk ızgaralarının geometrik özellikleri, ortamdan elde edilen uzamsal verilerin nicemlenmesi gibi yorumlanarak, hareket planlayıcısına bir girdi oluşturulmuştur. Hareket planlayıcısı aldığı bu girdilerle robot sisteminin kontrolör tarafına referans giriş işareti üretmesi sağlanmaktadır.

Bahsedilen bu ilk kısım, robot navigasyon taslağı içerisinde giriş nicemleyicisi olarak adlandırılmaktadır. Sensör girişleri ve yol eğrileri üç boyutlu Öklid uzayında bağımsız çizge ağaçlarının düğümleri olarak temsil edilmektedir. Uzamsal bölütleme yöntemi, engelllere bağlı olarak hesaplanan yol eğrisi ve engellere atanan en büyük ızgara geometrisinden bağlayarak, ikili bir çizge ağacı ile ilgi alanındaki uzayın yol eğrisi ve engellere ait ızgaralar örtüşmeyinceye değin dallara ayrılmasına dayanmaktadır.

Sonuç olarak ortaya çıkan doluluk ızgarası, her boyut için yol eğrisi ile çakışmayan en büyük ızgara geometrileri ile sonuçlanacaktır. Dinamik yol planı ve engellere göre ızgara boyutlarının değişmesine yakınlaşma ve uzaklaşma adı verilmektedir. Ortaya çıkan ızgara geometrilerinin boyut bilgileri hareket planlayıcısı için giriş işaret bilgisi üretmek için kullanılacaktır. Izgaralardan ve yol eğrisinden alınan bilgilere göre hareket planlayıcısı hız ve hareket yönü bilgilerini üretecektir. Aracın yol eğrisi bir engelin yakınlarında ise, yol eğrisi ve engele ait ızgaraların uzamsal olarak örtüşmemesi için ızgara boyutları yinelemeli bir şekilde küçülecektir. Bu nedenle bir genel kural olarak araç hızının ızgara boyutuyla doğru orantılı olarak arttığı öngörülmektedir. Benzer şekilde ızgaralar küçüldükçe köşegenlerinin tanjantı yol eğrisinin tanjantına yaklaşacağından, aracın yönlenim referans değerinin ızgara köşegen tanjantı ile yol eğrisi tanjantının arasında olacağı öngörülebilir.

Izgara geometrilerinden gelen verinin hareket planlayıcısında şekillendirilerek referans işaretlerin üretilmesinin ardından, işlem sırası yöngüdüm taslağının diğer bir temel bloğu olan çıkış nicemleyicisine gelmektedir. Çıkış nicemleyicisi bloğu, hareket planlayıcısı trafından üretilen referans işaretlerine asimptotik olarak kararlı bir nicemlenmiş kontrolör ile alt düzey motor sürücü sistemlerine işaret üretmekten sorumludur. Bahsi geçen asimptotik olarak kararlı bir nicemlenmiş kontrolör qLQR olarak tanımlanmakta ve robot kontrolünde bu çalışma ile uygulama alanı bulmaktadır. Bu yöntemle hem kararlılık hem de performans isterlerinin karşılanması hedeflenmiştir. Temel olarak bu kontrolör ile geleneksel LQR yapısından farklı olarak, üretilen kontrol işaretlerinin tekil bir değeri değil, asimptotik kararlılığı garantilenen bir alt ve üst eşik değer tanımı bulunmaktadır. Bu sayede alt düzey motor sürücülerinin çıktı üretebileceği referans değerine bağlı olarak bu iki eşik değer arasından en uygun değerin atanması sağlanabilemektedir. Böylelikle giriş nicemleyicisinde bölütlenmiş olan ızgaralar üzerinde asimptotik olarak kararlı hareket edilmesinin garantilendiği bir robot yöngüdüm çalışma alanı elde edilmektedir.

Bu çalışma robot yöngüdümünde planlama ve kontrol mimarilerinin birbirlerinden yalıtılmış bir biçimde planlanması problemine karşı bir çözüm üretmeyi amaçlamaktadır. Bu doğrultuda çalışmanın literatüre olan temel katkıları:

- Robot yöngüdüm ve kontrol sistemlerini birbirlerine bağlayacak bir genel çalışma alanı,
- Genelleştirilmiş ve aynı zamanda giriş nicemleyicisinde de kullanılan bir doluluk ızgaralama yöntemi,
- Nicemlenmiş LQR yönteminin robot yöngüdümü probleminde aktarılması,

şeklinde sıralanabilir.

Tez genel hatlarıyla şu şekilde sunulmuştur: Giriş kısmında problem genel hatlarıyla ele alınmış ve doluluk ızgarası ile nicemlenmiş kontrol yöntemi ile ilgili kısımlar iki ayrı literatür özeti olarak verilmiştir. Bölüm 2'de öncelikle nicemleme yöntemi kontrol teorisi üzerinden örneklemler ile aktarılmıştır. Bu giriş bölümünün ardından nicemlemenin bu çalışma alanındaki görevi genel hatlarıyla açıklanmıştır. Bölüm 3'e gelindiğinde, giriş nicemleyicisinde yol eğrilerinin ve engellerinin üç boyutlu Öklid uzayında katkılanan yöntem ile nasıl temsil edildiği teorik altyapıları ve algoritmik çözümlemeleriyle birlikte gösterilmiştir.

Örnekleme ve nicemlemenin ilişkisinin açıklanmasının devamında, araç dinamik kısıtları ve engel sakınma için güvenlik çemberi gibi farklı senaryolar için doluluk ızgaralarının ne şekilde bölütleneceği ile anlatıma devam edilmektedir. Bölüm 4'e gelindiğinde, doluluk ızgaralarının kullanıldığı giriş nicemleyicisiyle birlikte çıkış nicemleyicisi de konuya dahil edilerek genel çalışma alanı sunulmuştur. Kısım 4.3'te giriş nicemleyicisinin genel çalışma alanına nasıl eklemlendiğini ve yol planlamasından gelen parçalı eğrilerin nasıl işlendiğini anlatılmaktadır. Devam eden Kısım 4.4'te, çıkış nicemleyicisi tanıtılmakta ve hareket planlayıcısı ile giriş nicemleyicisinden alınan işaretlerin çıkış nicemleyicisine referans işaret olarak dönüştürülmesi anlatılmaktadır. Kısım 4.5'te, nicemlenmiş LQR yönteminin robot yöngüdüm çalışmasında kullanımı açıklanmıştır. Devam eden kısımlarda, nicemlenmiş kontrol yapısı geleneksel kontrol terimleri üzerinden tanımlanmış ve hibrit sistemlerdeki yapıya paralel bir şekilde kontrol işasretlerinin kararlılık tanımları yapılmıştır. Son bölümde ise qLQR ve genel olarak robot yöngüdüm çalışma alanının simülasyon ve laboratuvar ortamındaki gerçek zamanlı sonuçlarıyla bellek kullanımı ve hesaplama karmaşıklığının da irdelendiği çıktılar sunulmuştur.

Ekler kısmına gelindiğinde ise, Ek A'da önerilen yöntemlerin uygulaması için kullanılan, Ackermann modeli bir aracın doğrusal olmayan ve doğrusallaştırılmış sistem modelleri ile kontrol işaretleri ile aracın manevra kabiliyetleri arasındaki ilişki irdelenmiştir. Bahsi geçen manevra kısıtlamaları, giriş nicemleyicisi ve hareket planlayıcısındaki eşik değerleri oluşturmaktadır. Bu çalışmada örnek olarak bir kara taşıtı üzerinden çıkarımların yapılmasına karşın bu çalışma hava, deniz, sualtı gibi farklı ortamlardaki robotlar için de uygulama için elverişlidir.

Ek B'de ise yöngüdüm sırasında aracın engellerden sakınması amacıyla araç etrafındaki engellerin aracın etrafındaki hayali çarpışma eğrisine olan uzaklıkların belirlenmesi işleminin, statik ve dinamik durumlar için teorik irdelenmesi gerçekleştirilmiştir. Giriş nicemleyicisinde kullanılar bu uzaklık normlarının teorik çıkarımları, tezin genel akışını bozmamak için ekler kısmında sunulmuştur.

Ek C'ye gelindiğinde, nicemlenmiş kontrol işaretleri arasındaki geçişlerin ve aracın ortamdaki hareketinin Markov Zincirleri ile bağıntısı kurularak, çalışmanın robotik bilimindeki olasılıksal yaklaşımlara olan denkliği ve ne şekilde dönüştürülebileceği ile ilgili bilgi sunulmuştur. Bu çalışmayı takip eden gelecek çalışmalarda, öne sürülen bu yöngüdüm çalışma alanının olasılıksal tanım kümesindeki karşılıkları irdelenecektir.

Son olarak Ek D bölümünde, giriş nicemleyicisinde doluluk ızgaralarının engel varlığına göre boyutlarının bir ikili çizge yapısıyla yakınlaştırılması ve uzaklaştırılması işleminin asimptotik kararlılığa olan etkisi ile ilgili kanıtlar verilmektedir.

1. INTRODUCTION

Representation of the real-world environment for path planning and mapping algorithms is not a trivial problem. Autonomous vehicles or the agents in the perspective of artificial intelligence, perceive the surrounding via their exteroceptive sensors. These exteroceptor data come up with six degrees of freedom (DOF); three DOF for the Euclidean coordinates and three DOF for their angles along with the exact measurements plus the noise. However, the majority of the methods for navigation, path planning, and environment mapping are based on Cartesian coordinates, which are defined in two or three dimensions almost with a heading angle. Using the exact points for calculation is costly, especially for real-time applications. Besides, abstraction with Cartesian coordinates comes with the cost of information lost. The effectiveness of the approaches strictly depends on the existing trends of the research methods and hardware architectures.

In the first part of this dissertation, a novel space partitioning approach that assigns both obstacles and path spline occlusions to 3D Euclidean space with individual graph trees to use the superiority of both the topological and grid space partitioning. In a specific region of interest (ROI), every obstacle is observed as point clouds and candidate vehicle paths with the trajectory splines in the scope. In a general approach, the occupancy grids only gives information about the Cartesian or world coordinate of the obstacles and the sampled position of the trajectory with or without heading info. However, in the proposed approach, outcomes of the individual graph trees are utilized to generate reference inputs for motion planner in the framework. Thus, the occupancy grid structure is directly connected to the control mechanism of the vehicle with the guaranteed stability asserted with quantized LQR mechanism in the second part of the framework.

The main contribution of this dissertation is a navigation framework with an input quantizer that also has a novel grid segmentation methodology and output quantizer, which uses generated reference inputs from motion planner to accomplish asymptotic stability with a quantized LQR. For this purpose, a new LQR system namely qLQR is adapted from the quantizer definitions of [1] to the navigation framework with theoretical proofs and simulation results. Using quantization both on creating occupancy grids from sensor data and in control system architecture is a novel approach to robot navigation problem.

1.1 Purpose of Thesis

This thesis aims to introduce a new robot navigation framework that uses quantization operator both in occupancy gridding and control mechanism. To achieve this, a new occupancy grid mechanism is developed to generate reference inputs from the spatial information of the environment. This gridding method is suitable for real-time applications and can be utilized independently from this framework.

The proposed navigation framework is capable of generating asymptotically stable control outputs, which is achieved by introducing the quantized LQR control mechanism in robot control. The framework is designed to run under real-time constraints from sensor data capture to control signal generation.

1.2 Literature Review of Related Studies

Literature Review is separated into two sections that cover the main contributions of the proposed framework. The first part includes the literature review for the occupancy grid methodology used in the input quantizer. The second review section is a survey about the existing control methods that are used in robot navigation problem similar to the proposed approach in the output quantizer.

1.2.1 Input grid quantization

Previous studies in occupancy grid segmentation problem can be clustered under two separated branches. One of this branches is the topological methods of representing the locations, which uses graph trees where each node has information about the environment or directly the object itself. Grid space partitioning is the other branch that is based on dividing the environment into the cells and tracking the record of a particular cell is occupied or not. These methods are also known as occupancy grids.

In the meantime, up-to-date studies tend to merge both ideas in a mixed framework for the sake of using superior parts of the graph trees and grid cells to open a new perspective on this vital area.

K-d trees [2] are familiar node tree form of occupancy grids, which are suitable for space clustering and partitioning. Greenspan et al. [3] proposed a method that promotes a fast search speed with a small memory requirement, which comprises search accuracy in low dimensional datasets without inducing a backtracking step. This study improved by the addition of backtracking and storing mechanism in [4]. In a high-dimensional image dataset, randomized K-d search trees with SIFT descriptors improves the performance [5]. K-d tree methods divide the local search space unevenly regarding data feature occurrences, which cause arbitrarily large regions of search space. Schauer et al. [6] analyze the CPU and GPU performances with regard to state of the art collision detection methods.

Another prominent study is Octree-based OctoMap [7]. Octrees are designed to be applied in 3-D spaces and cannot be easily converted into planar surfaces. It is convenient to combine with different approaches like partial map fusion with Iterative Closest Points (ICP) for environment mapping [8–10] and frontier-based navigation tools [11, 12].

Quad-trees [13] have a structure that divides the 2D surface into four quadrants. Probabilistic grid map implementation on quad-trees is performed in [14]. Cell decomposition with grid-based path planning in a hierarchical framework [13], searching for feasible paths using connectivity graphs [15], weighted distance transforms for heterogeneous regions [16] and mesh grid generation in 2D surfaces [17] are the expansions on this framework. Quad-trees are not suitable for independent representation of the 3D dimensions by virtue of the identical division sizes in the orthogonal dimensions, and lack of the elevation information.

R-tree [18] and its derived studies [19, 20] are the most common approaches in topological classification methods. Rapidly exploring random trees (RRT) are the subgroup of the R-trees with numerous variants, such as [18, 19, 21, 22]. However, if there exist large configuration spaces relating to the presence of many obstacles, then the computational cost of RRT algorithms grows significantly.

K-d trees are proper solutions for space partitioning in nearest neighborhood search problems. Moreover, splitting the local search space intermittently with obstacle or feature occurrences is not suitable for efficient data clustering while considering the path allocation within the map. Octrees are designed to be used in 3D spaces and cannot be converted into planar surfaces easily. The proposed input quantization method uses separated trees for all dimensions to overcome the 3D to 2D conversion problem. Quadtrees have a structure that divides the 2D surface into four regions. The proposed method deviates from the quadtrees because of orthogonal dimensions are split into regions independently. Besides, the elevation information is not included in quadtrees. R-trees are useful in topological map classification problems. However, R-trees are based on sparsely distributed bounding box methods, so they are not suitable to estimate input signals for path planning step. R-trees and its derived approaches are straightforward algorithms to implement, but can only be applicable after preprocessing of the obstacle space. RRT algorithms also consist of multiple line segments, which are also not suitable for direct use in path planning.

1.2.2 Quantized LQR method for output quantization

Systems that contain both continuous dynamics and discrete events are called hybrid or discontinuous dynamical systems. The discontinuity in the system can be based on the control system transitions like the operation of gearbox shift pattern in vehicles or the steep system dynamics like a change of direction or final stop of a bouncing ball. In the early studies of control theory, this well-known phenomenon in the system dynamics interpreted as a predictable disturbance [23] or the noise on the signal [24]. After the widespread usage of the pulse width modulation signals in electric drives at the beginning of the 90s, sliding mode control techniques [25, 26] use this nonlinear switching behavior as a control method for stabilizing nonlinear systems with an on-off (bang-bang) controller. Following the broad range of industrial practices, usage areas of sliding mode control reached many application areas in robotics like stabilization of autonomous surface vessels in rough open seas [27] and reactive position control of quadrotors [28]. Sliding mode control frameworks define the switching regions as discontinuity surfaces and aim to design feedback controllers that direct the optimal solutions of system states to settle around these surfaces. The near-optimal solutions around discontinuity surfaces induce chattering problem, which is a high-frequency switching phenomenon between various controllers that designed for different set points originated by neglected effects of actuator and system dynamics [29].

Projection of this chattering situation into robot navigation problem can be seen in a seesaw movement of a vehicle among attractive goal point and different repellent sources like the obstacles around the vehicle. At this point, derived studies of artificial potential fields [30–32] and vector fields [33, 34] for navigation also tend to behave in the same manner around this instability boundaries. Different approaches like assigning coefficients for cost functions to stay away from these regions as in the dynamical window approaches [35, 36] do not guarantee a stabilizing controller and works only the predefined areas where the parameters are tuned for these special cases. Decentralized cooperative control of swarm robots as in [37] has related approaches but in a different context with a switched control system. This phenomenon of using existing control methods with different notation can be seen very often in the robotics literature. Similarly, optimal control theory literature dismisses antecedent study of Carathéodory's [38] and this leads up to related works on discontinuous dynamics with Carathéodory's solutions [39–41] and its variant Filippov solutions [42]. These studies suggest the equivalent solutions of Pontrayagin's minimum principle [43] by inspecting the vector fields of a specific operating point or neighborhood of an operating point respectively. Cortés [44] contributes a comprehensive review paper on discontinuous dynamics systems with Carathéodory's and Filippov's solutions. However, these approaches take a rather different way by focusing on the understanding of complex dynamics systems then discussing the stability or design control laws on discontinuous dynamics. The solutions of piecewise continuous vector fields are used in the area of state-dependent switching dynamical systems. In [45], the hybrid systems with sensor and actuator constraints are studied with a different notion called quantizers. It is evident that robotics applications are a perfect match for a hybrid system both with sensor and actuator constraints.

Quantization term is used to describe the levels or sets that are separated by discontinuous events like the change of gear in a vehicle. Quantized sets have their unique continuous dynamics, which may have similar properties like the shift change between second and third gears or completely different ones like switching between first and rear gears. Using quantization effect not as an error source to be predicted

with white noise models, but rather as a stabilizer of an unstable discrete-time system by introducing the usage of quantized states as an input to state feedback mechanism is based on the study of Delchamps [46]. This strategy changes the traditional view of quantization from a simple rounding operator to a system state to be utilized in control system design process. The characteristics of quantization effects in hybrid systems are extensively studied in [45, 47]. However, these ideas applied in hybrid systems were not acquainted adequately in the robot navigation and control problem.
The proposed study is influenced by the idea of using the measurement quantization as information coding of the real data, which is used directly by the controller. Also, in [48], quantization intervals are associated with the stability of the system in an explicit relation with the systems closed-loop unstable poles. This relation is exhibited by the minimum number of quantization levels and the system poles in the right-half plane (RHP). Therefore, it is essential to determine the minimum information that is needed for the higher levels of the control system after applying quantizer.

1.3 Contributions of the Thesis

This research aims to address the problem in robot navigation, which is manifested as the isolation of the planning and control architectures. Fundamental contributions of the study are:

- A navigation framework that bonds the navigation and control sections.
- A novel occupancy grid method to serve in general and in the input quantization block of the proposed framework.
- Adaptation of the quantized LQR method into the robot control problem.

1.4 Thesis Outline

The rest of the thesis organized as follows; first of all, the quantization concept in control theoretical sense with some illustrations is given in Chapter 2. After the brief introduction to the quantization with an example, the role of the quantization in the framework is explained in general. In Chapter 3, the input quantizer, which is responsible for space partitioning that assigns both obstacle and path spline occlusions to three-dimensional Euclidean space with individual graph trees is explained in detail. Following the sections that explain the relationship between sampling and quantization, grid allocation for obstacles and paths is presented with a detailed procedural approach including the vehicle constraints and collision avoidance concerns. In Chapter 4, the general framework of input and output quantization for navigation is introduced. Section 4.3 explains input quantizer implementation in the general framework and how it is handling path chunks from path planner with quantization. The following Section 4.4 is about the second quantizer in the framework, which is the output quantizer. Here, a motion planner is utilized to generate reference control signals to the output quantizer from the outcomes of the input quantizer block. In Section 4.5, the quantized LQR system is derived for this framework, which is used in previous sections. In Section 4.5.1, the quantized control structure is derived in classical control terms. In the next section(4.5.2), procedure continues with the derivation of stability boundaries of control signals as the same structure in hybrid systems [45]. The effectiveness of qLQR compared to the traditional LQR system and qLQR usage in the proposed framework is demonstrated both with real-time application in an indoor environment and simulation results in Chapter 5.

2. QUANTIZED FEEDBACK CONTROL THEORY

2.1 Purpose

In this section, the theory behind the quantization operator is explained briefly in control theoretical way with examples. The second part is devoted to the usage of the quantizer in the framework and denoted concepts mentioned in the dissertation. The section serves as a guideline and reference to the chapters where the detailed analysis is conducted.

2.2 Quantizer Operator

The standard method for sampling a continuous error signal $e(t)$ to a sequence of impulse functions ($e_s(t)$) is achieved by a sampling switch with an appropriately chosen sampling time T. The area of this impulse sequence is the sampling instant, which is denoted as $e(kT)$. Through the property of the area of Dirac-delta impulse function equals to 1 (i.e., $\int \delta(t)\,\mathrm{d}t = 1$), the sampled error signal $e_s(t)$ is represented as,

$$e_s(t) = \sum_{k=-\infty}^{\infty} e(kT)\delta(t-kT) \tag{2.1}$$

To convert this sampling instants to a physical signal, there should be a holder, which is engaged to hold the previous signal value until the next signal is sampled. Zero-order holder is the simple and effective solution, denoted by h_0,

$$h_0(kT) = e(kT) \tag{2.2}$$

This sample and hold mechanism obeys the superposition rule. Thus, the linearity of the sampled continuous signal is not affected at this stage. In addition to that, a quantizer, which can also be interpreted as a nonlinear discontinuous sampler is added just before the system. The quantization operation is a mapping operator that is directed from a Euclidean space $\Re^k$ to a finite subset of this space $\mathscr{P}_{\text{fin}}(\Re^k)$,

$$q\,:\,\Re^k \mapsto \mathscr{P}_{\text{fin}}(\Re^k) \tag{2.3}$$

The operation introduced with a quantization operator q sets the sampled values to corresponding quantized set values. In Fig. 2.1, sampling, hold and quantization operations are illustrated together. Here, the quantizer operation for this example defined as,

$$q(x) := \begin{cases} L1 & 0 \leqslant h_0(kT) \leqslant l1 \quad (setA) \\ L2 & l1 \leqslant h_0(kT) \leqslant l2 \quad (setB) \\ L3 & l2 \leqslant h_0(kT) \leqslant l3 \quad (setC) \end{cases} \tag{2.4}$$

quantizer operator is arranged to switch between three different sets A, B and C depending on the value of signal against intervals $l1$, $l2$, $l3$ and get corresponding set values, which are $L1$, $L2$, $L3$.

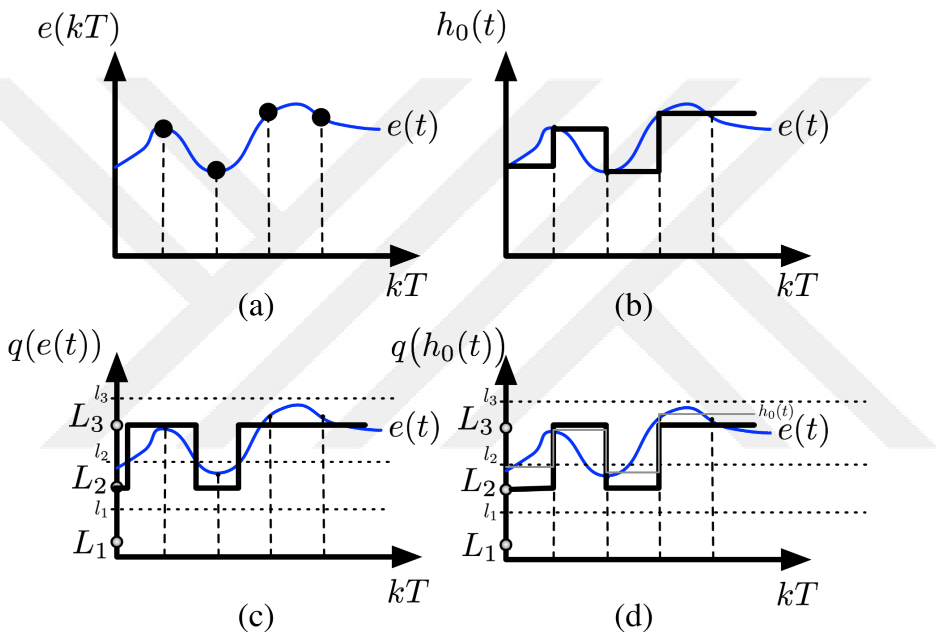


Figure 2.1 : Sampling, hold and quantization operations on a random error signal. (a) Discrete sampler. (b) Zero order hold operator. (c) Quantization operator applied on continuous signal. (d) Quantizer operator applied on hold signal.

The unified framework considers quantization, time delay and disturbance to handle nonlinearities in system dynamics even the system is modeled or behaved like a linear system. Thus, the first step is modeling the additive nonlinear effects in deterministic error signals (e), such as Gaussian white noise. The second step is to design a control law k disregarding the error signals with static feedback. Generalized dynamic equations for such a robotics system can be written as;

$$\begin{aligned} \dot{x} &= f(x,u) && (2.5) \\ y &= h(x) && (2.6) \end{aligned}$$

where, $x \in \Re \rightarrow \Re^n$ denotes system states, $u \in \Re \rightarrow \Re^m$ is control signal, $y \in \Re^n \rightarrow \Re^l$ is the system output, f is the nonlinear system characteristic function and h is the nonlinear measurement function. The aim is to design a control law k that asymptotically stabilizes the given system, $u = k(x)$. Now suppose that the state feedback x is quantized, hence the control law becomes

$$u = k(q(x)) \tag{2.7}$$

$$u = k(x+e) \tag{2.8}$$

here $e := q(x) - x$ is *quantization error*. The goal is to reduce the e as time goes to infinity;

$$\mathbf{G} : \lim_{t \to \infty} e(t) = 0 \tag{2.9}$$

2.3 Quantization Application in the Framework

Among the best definition of quantization is given by Kalman [49]: "A process of replacing a sequence of points by a sequence of a set is called *Quantization*". Many control and robotics systems are operated with intermittent sensor signals like LIDAR, LADAR, RADAR, digital cameras. Besides, digital computers and actuators like servomechanism have a sampled-data systems and saturation effects.

The conventional method of sampling a continuous error signal $e(t)$ to a sequence of impulse functions $e^*(t)$ is achieved by a sampling switch with appropriately chosen sampling time T_s. The area of this impulse sequence is the sampling instant, which is shown as $e(kT)$ [1]. In order to convert sampling instants to a physical signal, a holder block is engaged to hold the previous signal value until the next signal instance is sampled. Zero-order holder is a simple and effective solution with the transfer function of $(1 - e^{-s\tau})$. This sample and zero-order hold mechanism obeys the superposition rule. Consequently, the linearity of the sampled continuous signal is not affected at sample and hold stage. In addition to that, a quantizer is added just before the system

[1] The area of Dirac-delta impulse function equals to 1. i.e., $\int \delta(t)\, dt = 1$

$G(s)$, which is also interpreted as a nonlinear discontinuous sampler in this context. This feedback control model is illustrated in Fig. 2.2. In real-time applications, the

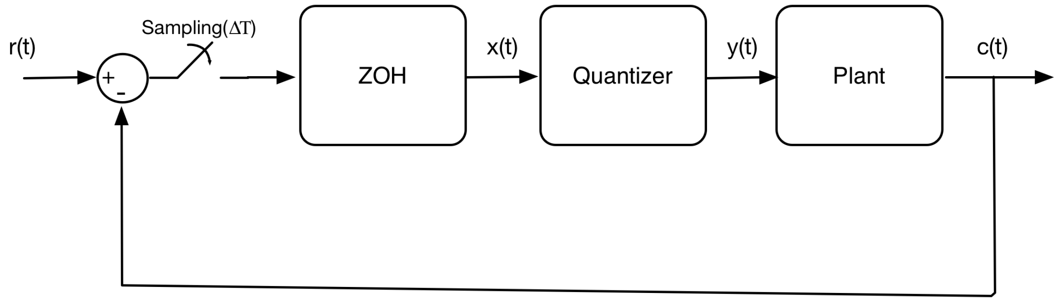


Figure 2.2 : Feedback control scheme with quantizer.

memoryless predictions like Markov process models are vital tools to approximate the stochastic nature of dynamic environments. With a process that satisfies the Markov property, one can predict the next system state from the current state and the *chosen* control signal. The probabilistic equivalent of a deterministic system with a one-dimensional phase line in phase space is a Markov process of first order. However, association of the nonlinear systems with Markov processes is not straightforward. On the contrary, linking Markov process to nonlinear differential equations is quite easy as shown in [49]. For that matter, with using the proposed method, portions of the quantization intervals that associate the sampled data to specific sets are correlated with the probabilistic approach of a Markov process on the same system. The relationship between the quantization process and Markov process is explained in Appendix C.

A quantizer is an information coder. The main question here is how much information needs to be gathered either by sensors or communication channel in order to achieve a certain control objective, which is also a quantized process. Quantizers are classified under two categories: *Uniform* and *Logarithmic* [47]. While the logarithmic quantizers handle the quantization error as uncertainty or nonlinearity, uniform quantizers have *zoom-in* and *zoom-out* properties, which are the two main spatial gridding structure of the proposed solution. Zoom-out property allows measuring the state of the system, and the zoom-in property is used to reduce the quantization error.

In the framework of this thesis, the quantization process and its effects on autonomous vehicle path planning are defined in a robot traversal formation versus human driver analogy after this stage. In an empty scene, these intermittent sensors are actively scanning the area and act instantly to the changes even if it does not directly affect the vehicle navigation. This behavior is similar to a novice driver who has full of anxieties that checks his/her rear mirror over and over and exaggeratedly monitors the road to aware all small details. What happens next is the driver exhausted after this concentrated monitoring process, likewise the excessive overloading of processor and sensor units of the autonomous vehicle. In an empty road, somewhat lazy-driving is expected optimal behavior of the human driver. The phenomenon of lazy-driving is handled by the zoom-out principle of the quantization process, where the grids are coarse as far as the security and stability concerns are satisfied. The stability is satisfied with the quantized feedback controller in output quantizer, which is shown in Section 4.5.

The internal adaptation characteristic of the quantizer is analogous to open-loop control strategy in classical control theory. Similar to the sensor mechanisms, actuators both in the steering wheel and the throttle unit have coarser control regions in slow dynamic systems. As the static/dynamic obstacles or other vehicles (human-driven or autonomous) are detected in the surrounding environment, increasing switching events are expected to occur between quantization regions. This event triggers the zoom-in portion of the quantizer that has denser grids both in the sensors and actuator drives. This type of quantizer is analogous to closed-loop control, which is more accurate and robust to external disturbances and the noise. Derivation of the zoom-in property for this behavior explained in Chapter 3.

2.4 Discussion

This chapter briefly explains the quantizer operator and how it assigns a signal to different sets in mathematical notation. Sampling, hold, and quantization operators are compared concerning each other through their behavior on a given error signal. Quantization process in the framework is stated with types and zoom-in/out concepts. Ties between the Markov Process and the quantization operation, which are adressed in detail in Appendix C are mentioned. A real-life scenario is adapted to explain the concepts articulately.

3. SENSOR QUANTIZATION AND KINEMATIC PLANNING

3.1 Purpose

In this chapter, a novel space partitioning approach that assigns both obstacle and waypoint occlusions to three-dimensional Euclidean space with individual graph trees is proposed. In the specified region of interest (ROI), every obstacle is represented by point clouds and candidate vehicle paths with waypoints in the space. Obstacles and waypoints correspond to a node of separated 3D node trees concerning their spatial localization. Every node has the property of ROI sampling rate and a spatial threshold that divides its corresponding ROI into two half parts. The division has a dynamic feature inside, which is triggered by the existence of the obstacle. Achievements in this chapter has been published and include material from the paper by the author [50].

3.2 Quantized Allocation in Grid Maps

The procedure in this section is based on a node tree representation of each spatial dimension in a Euclidean space independently. Hence, a point p in 3D space is represented by three different spatial graph identifiers idx, idy, idz for each dimension. Associated graphs in 3D Euclidean space always occlude a region if p is in the corresponding ROI, which is denoted by,

$$\begin{aligned} &\forall p \in \mathbb{R}^3 \; \exists \{idx, idy, idz\} \in \mathbb{N}, \\ &p \in ROI \implies Occ(p, G_x, G_y, G_z, idx, idy, idz) \end{aligned} \tag{3.1}$$

here, G_x, G_y, G_z are indicating graph trees associated with the Euclidean space dimensions x, y, z respectively. $Occ()$ is the occupancy assignment method that assigns each point to the graphs. In the implementation level, $Occ()$ method assigns graph identification (id) numbers to the obstacle or path spline points for each dimension and returns true or false if the operation is successfully completed or not. ROI is the region of interest area defined arbitrary by the user due to the mission requirements. The ROI is used in a range check function to determine if the point is inside of the

ROI or not. Classifying the features or assigning id numbers has a monotonically increasing computational complexity. Initially, the predetermined ROI area with the vehicle in its origin is divided into two halves for each axis. This division starts with the assignment of the root node with id: $(0,0,0)$. After this step, every half part is divided by two if there is a need for denser representations. The generated nodes are called as siblings of the parent node with direction prefixes: left-child and right-child. For each dimension, identifier number of the sibling nodes n_s of a parent node n_p are assigned by the following rule,

$$\forall n_p \in \{G_x, G_y, G_z\} \left(\mathbf{hasChild}(n_p) \implies id(n_s)\right) \tag{3.2}$$

where,

$$id(n_s) = \begin{cases} 2 \times id(n_p) + 1, & \text{if } n_c = \mathbf{rightChild}(n_p) \\ 2 \times id(n_p) + 2, & \text{if } n_c = \mathbf{leftChild}(n_p) \end{cases} \tag{3.3}$$

The spatial sampling rate of a grid is defined as the number of leaves, where the branching is performed symmetrically for every left and right children starting from the root node (Fig. 3.1(a)). Here, the branching is achieved in connection with the obstacle or path occurrences, which causes the binary trees need not to be symmetrical like shown in Fig. 3.1(b). Therefore, another property of the nodes is accommodated to show how many sections that the ROI is separated apart(i.e. spatial sampling rate). The depth of node property is a relevant attribute to specify the ROI of each node in the ROI of its root node. The spatial sampling rate of the node γ is defined as;

$$sampling\_rate(\gamma) \triangleq 2^{depth(\gamma)} \tag{3.4}$$

The depth of node property is also useful in general usage when multi-vehicle operations that depend on each other's movements are taken into consideration. An example for relative node assignment with numbering and a case for a multi-vehicle operation where a slave vehicle's root node is assigned as a child of a master vehicle is shown in Fig. 3.2. The height knowledge of the node is useful to predict how much denser the grids can be scaled. This property is also significant when the vehicle has kinematic constraints. For example, when a steering wheel has a servo motor driver that provides $\delta\theta$ heading angle resolution for each turn, it is not reasonable to use a grid size less than $\upsilon\Delta t \sin(\delta\theta)$ where υ is defined as the linear velocity of the vehicle.

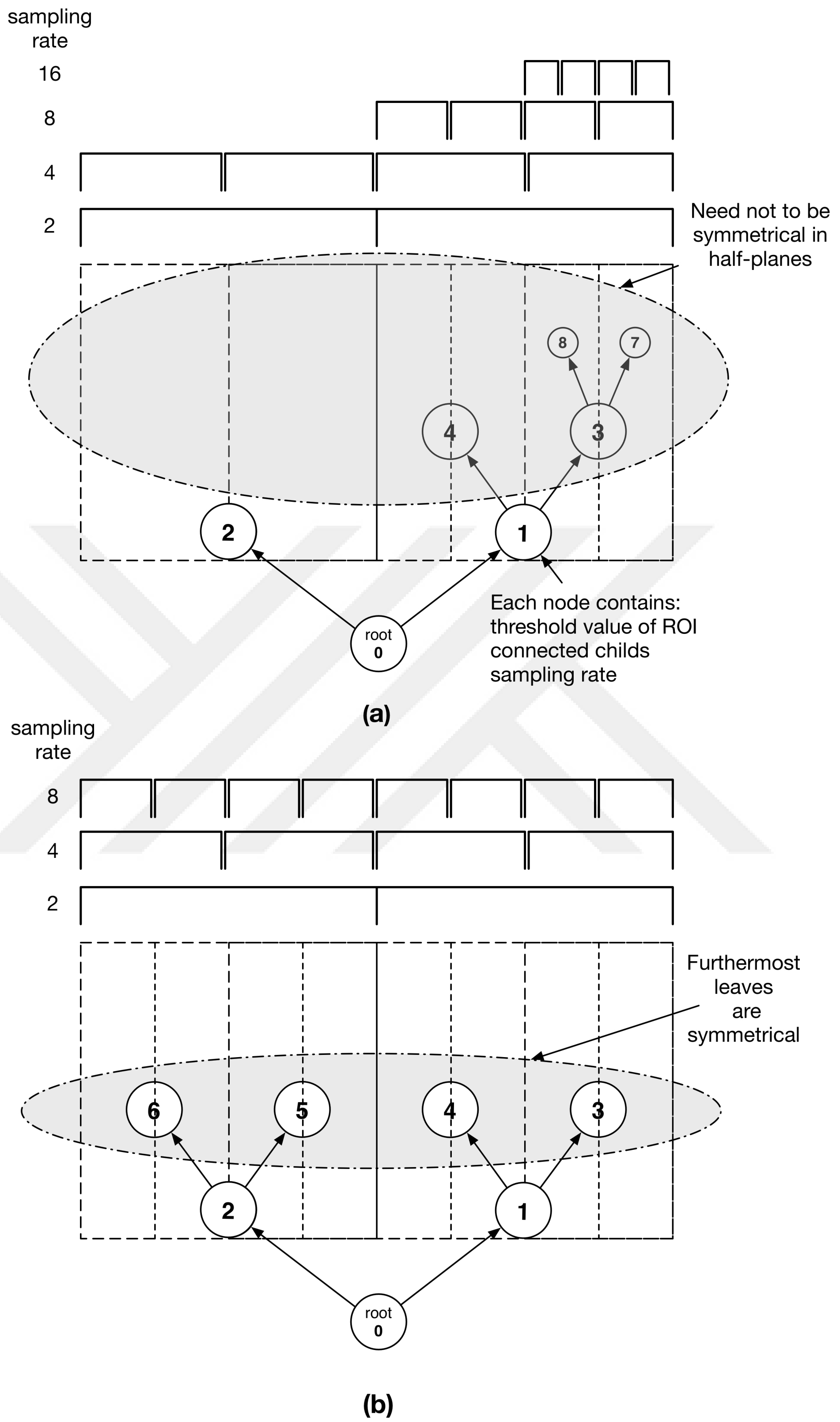


Figure 3.1 : (a) Relation between spatial division and sampling rate. Every child/leaf node split the region that belongs to their parent node into half.
(b) Asymmetric division due to the obstacle occurrence. This property provides independent sampling rates while holding the interdependence via parent nodes.

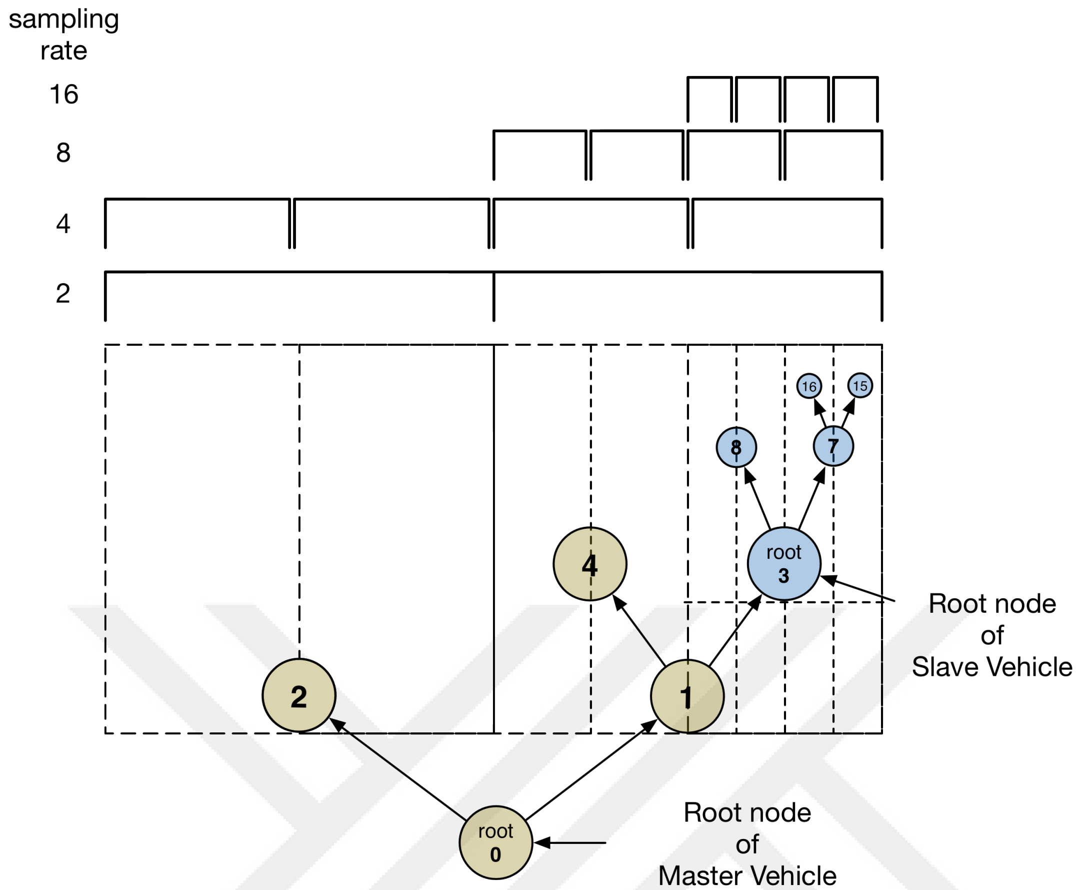


Figure 3.2 : Relative node assignment in Multi-vehicle operation scenario.

3.3 Obstacle Allocation in Grid Maps

Assigning scan data to graph trees for each dimension indicates a binary tree search pattern using node threshold values. Obstacles are represented as 3D point clusters after the preprocessing step of the raw sensory data from collected point clouds. Hereafter, either the point cluster or a unique point belongs to an obstacle is denoted as *landmark* or *feature*. Landmarks are represented as points in 3D space with an indexed tuple (x_i, y_i, z_i), in which i is the index value for the ith landmark. Landmark tuples are indexed with spatial graph identifiers idx, idy, idz for each dimension. Graph identifiers are the branch numbers of the graph trees, which is derived in Eq. (3.2). Each landmark is represented by the edge node as long as its possible. The number of nodes or the depth of each graph shaped with the navigation plan and the existence of the landmarks in the scene. In the Eq. (3.1), all the points in the Euclidean space can be grouped under one graph node. In this case, an obstacle point p_o is a tuple (x_i, y_i, z_i)

which is also defined in 3D Euclidean space. Thus, this point has a unique node id by definition,

$$\forall p_o \in \mathbb{R}^3 \; \exists \{idx, idy, idz\} \in \mathbb{N} \tag{3.5}$$

and if this obstacle point is in ROI boundaries, then the associated node for this point is indexed as an occluded node,

$$p_o \in ROI \implies Occ(p_o, G_x, G_y, G_z, idx, idy, idz) \tag{3.6}$$

Allocation of the obstacle points into the graph nodes is described in Fig. 3.3. Here, an obstacle point p_o is assigned to the independently partitioned regions that belong to the grid graph trees for each spatial dimension. Coarseness of the grids and the branching differences between graph trees depend on the vehicle path and the obstacle localizations. A graph node is partitioned into its child nodes when both the vehicle path and obstacles in the ROI overlap in a grid that belongs to the same graph node until the coincidence is over. The algorithmic structure of this association is given in Section 3.3.1.

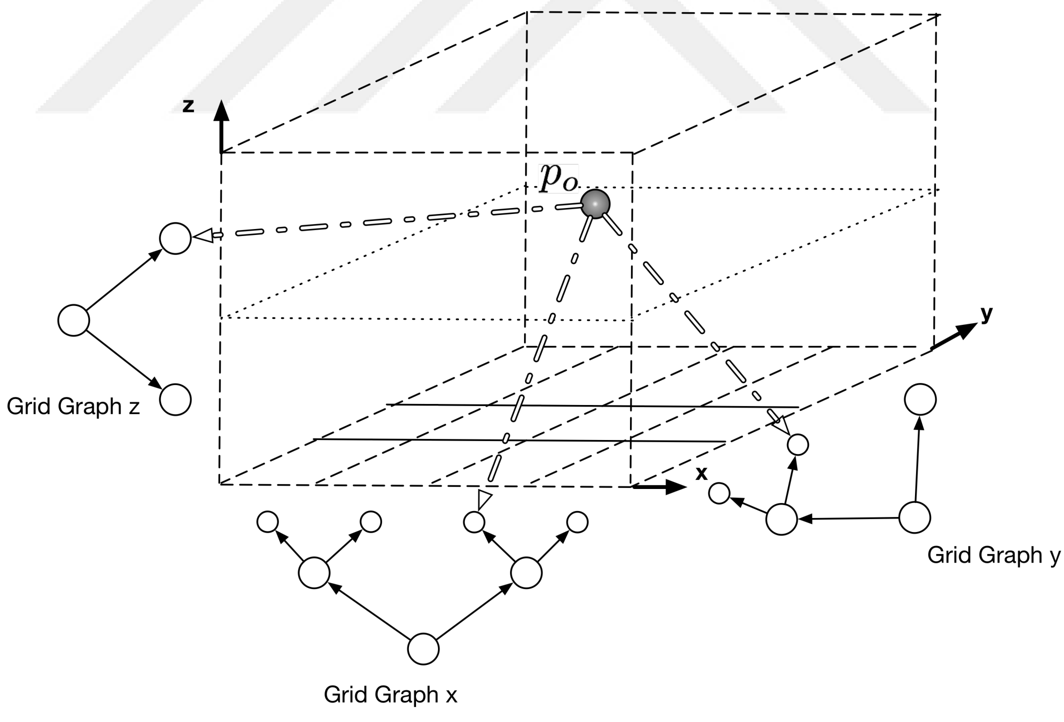


Figure 3.3 : Assigning an obstacle point to registered graphs in all dimensions. The grid graph trees for each dimension are independent of each other. Every point in ROI space belongs only to a unique node and these nodes are stored in memory for the next steps. This property provides fast and flexible structure to deal with significant numbers of point clouds.

3.3.1 Procedural approach for obstacle grid allocation

The pseudocode in Algorithm 1 explains the assignment procedure for the landmarks. This procedure checks the landmarks in the ROI and assigns to the corresponding left-child or right-child through the graph tree branching. Each node has a sampling rate that belongs to the height of that node. These landmarks are stored in an associative array to list the occluded nodes that assigned to a specific node. The time complexity of this method takes $log_2\ m$ steps in all cases (best, worst and the average). Variable m represents the depth of the edge leaves in the branch. Therefore, the ultimate worst scenario takes $log_2\ max(grid\_size\_allowed)$ and this term is defined as,

$$max(grid\_size\_allowed) = 2^{depth\_size\_allowed} \tag{3.7}$$

and the worst case for searching the dependent node of an obstacle takes,

$$worst\_time = \log_2 2^{depth\_size\_allowed} = depth\_size\_allowed \tag{3.8}$$

Algorithm 1 Assign node to obstacles.

```
Input: landmark_list
Data: graph class
Result: Assigning Node to Obstacles
initialization;
foreach item in landmark_list do
    if item in range(ROI) then
        while current_node has_child() do
            if landmark_list[item,'dimension'].value > current_node.threshold then
                current_node ← current_node.right_child()
            else
                current_node ← current_node.left_child()
            end
            landmark_list[item,'dimension'].value←current_node.samp_rate()
        end
    else

    end
end
```

3.4 Path Allocation in Grid Maps

With the intent of clarifying the ideas and focusing the effort on gridding problem, quadratic and Bezier curves [51] are implemented to demonstrate a steerable path partition in ROI area. Bezier curves are a special case of Bernstein polynomials, which provides a practical way to approximate continuous functions like the path splines.

Bernstein polynomials are defined by,

$$B_{i,n}(k) = \binom{n}{i} \cdot k^i \cdot (1-k)^{n-i} \tag{3.9}$$

n is the number of control points that defines either the curve is quadratic ($n = 2$) or cubic ($n = 3$). Variable k defines the step size with $k \in \mathbb{R}$, $0 \leq i \leq 1$ and the index variable is defined with $i \in \mathbb{N}$, $0 \leq i < n$. A linear combination of Bernstein polynomials C_n is shown as,

$$C_n(k) = \sum_{i=0}^{n} \beta_i B_{i,n}(k) \tag{3.10}$$

here, β_i is called Bernstein coefficient or Bezier coefficient in the terminology and substituted by $P_i \triangleq [x_i, y_i, z_i]$ which are called as Bezier control points. The points are used to shape the Bezier curvature to navigate without collision. The algorithmic explanation for path association for quadratic or cubic splines is defined in Section 3.4.1.

3.4.1 Procedural approach for path grid allocation

The problem of occupying the traversal points of the Bezier curve in grid space is addressed with two different approaches. The first approach is a sampling-based method that takes samples with a given step size which is related to the grid size on the Bezier spline. Pseudocode for this occupation algorithm explains the sampling procedure of the quadratic and cubic Bezier splines for the given Bezier control points P_0, P_1, P_2, P_3 in Algorithm 2.

After sampling the Bezier splines, corresponding grid nodes can be assigned by Algorithm 1. However, using Bezier splines in the computations can be time-consuming when the ROI is chosen too large. Bezier spline calculation has $O(n)$ complexity in the worst case [52], where n is the number of sample steps (pseudo variable k in Algorithm 2) chosen independently from the grid size. In the best case, it approximates to O(max(grid_size['x'], grid_size['y'], grid_size['z'])). A rule of thumb here for the cases when the Bezier steps are intended to be chosen fixed, half-length of the minimum grid size is a good choice for not to skip a grid cell through the path. On the other hand, Bresenham's line algorithm has a computational complexity of O(max(grid_size['x'], grid_size['y'], grid_size['z'])) in the worst scenario [53]. Besides, Bresenham's line algorithm is designed for

Algorithm 2 Calculation of Quadratic and Qubic Bezier spline.

Input: P_0, P_1, P_2, P_3
Result: Get Bezier Points for control points
initialization; step_size = max_depth_size * samp_rate / grid_size['dimension'];
if *type=="Qubic"* **then**
    foreach *k in range[0, 1, step_size]* **do**
        $P^* \leftarrow k^3 \cdot P_3 + 3k^2(1-k) \cdot P_2 + 3x(1-k)^2 \cdot P_1 + (1-k)^3 P_0$
        bezier_path.append(P^*)
    end
end
if *type=="Quadratic"* **then**
    foreach *k in range[0, 1, step_size]* **do**
        $P^* \leftarrow (1-k)^2 \cdot P_0 + 2k(1-k) \cdot P_1 + k^2 P_2$
        bezier_path.append(P^*)
    end
end

computer graphics, which is based on assigning pixels in screen coordinate system like the occupancy grid maps in this case. Bezier spline algorithm has strength on its recursive nature and is based on a differentiable and derivable (such as quadratic, cubic and higher-order forms) function. Conversely, Bresenham's line algorithm is an optimization tool based on minimizing the error between the continuous function and its quantized counterpart. For these reasons, the following approach is derived from Bresenham's line algorithm [54] to occupy path footprints. Bresenham's line algorithm is used in computer graphics from early days to modern times to represent continuous lines in pixels. Primarily, this algorithm calculates the slope of the line from sampled points and allocates the neighborhood pixels. Pseudocode for the occupancy via Bresenham's line algorithm is given in Algorithm 3. Here, the Bezier control points are given as an input, and the first thing is to check that these points are suitable for a proper Bezier curve. If this statement is true, use the longer edge to continue the process. Another check is for the curvature is a straight line or not. If the curvature is a straight line, it is not differentiable. Thus, this special case is treated individually to handle its non-differentiable behavior. After checking for this special case, if it is an ordinary curve, the first and second order differentials are calculated and interpolated through the guidelines.

Algorithm 3 Occupancy grids with Bresenham's line algorithm.

Input: $x_0, y_0, x_1, y_1, x_2, y_2$
Result: Occupied grids
initialization; $path\_queue = []$; $dx_0 \leftarrow x_2 - x_1$; $dy_0 \leftarrow y_2 - y_1$; $dx_1 \leftarrow x_0 - x_1$; $dy_1 \leftarrow y_0 - y_1$; $curv \leftarrow dx_1 \times dy_0 - dy_1 \times dy_0$;
if $dx_0 \times dx_1 > 0$;***and***;$dy_0 \times dy_1 > 0$ **then**
  Return false # not a proper Bezier curve
else
  #Begin with longer edge ;
  if $(dx_0^2 + dy_0^2) > (dx_1^2 + dy_1^2)$ **then**
    $x_2 \leftarrow x_0$; $x_0 \leftarrow x_1 + dx_0$; $y_2 \leftarrow y_0$; $y_0 \leftarrow y_1 + dy_0$; $curv \leftarrow -curv$;
  else
    if $curv \neq 0$ **then**
      #If line is not straight #step forward in x,y-dim ;
      $dx_1 \leftarrow dx_1 + dx_0$;
      $dx_1 \leftarrow 1\,\mathbf{if}(x_0 < x_2)\,\mathbf{else} - 1$;
      $dx_1 \leftarrow dx_1 \times dx_0$; $dy_1 \leftarrow dy_1 + dy_0$;
      $dy_1 \leftarrow 1\,\mathbf{if}(y_0 < y_2)\,\mathbf{else} - 1$;
      $dy_1 \leftarrow dy_1 \times dy_0$;
      #second order differentials ;
      $dxy \leftarrow 2 \times dx_1 \times dy_1$;
      $dx_1 \leftarrow dx_1 \times dx_1$; $dy_1 \leftarrow dy_1 \times dy_1$
      # check for negated curvature;
      if $curv \times dx_0 \times dy_0 < 0$ **then**
        $dx_1 \leftarrow -dx_1$; $dy_1 \leftarrow -dy_1$; $dxy \leftarrow -dxy$; $curv \leftarrow -curv$;
      end
      #first order differentials ;
      $dx \leftarrow 4 \times dy_0 \times curv \times (x_1 - x_0) + dx_1 - dxy$ $dy \leftarrow 4 \times dx_0 \times curv \times (y_0 - y_1) + dy_1 - dxy$ $dx_1 \leftarrow 2 \times dx_1$ $dy_1 \leftarrow 2 \times dy_1$ $err \leftarrow dx + dy + dxy$
      #define error margins for termination ;
      $err\_margin\_x \leftarrow \varepsilon_x$; $err\_margin\_y \leftarrow \varepsilon_y$;
      while $dy < dx$ **do**
        if $\|x_0 - x_2\| < err\_margin\_x$ ***or*** $\|y_0 - y_2\| < err\_margin\_y$ **then**
          Return $path\_queue$;
        end
        if $(2 \times err) > dy$ **then**
          $x_0 \leftarrow x_0 + x\_step\_scaler \times dx_0$;
          $dx \leftarrow dx - x\_step\_scaler \times dxy$;
          $dy \leftarrow dy + y\_step\_scaler \times dy_1$;
          $err = err + y\_step\_scaler \times dy$;
        end
        if $(2 \times err) < dx$ **then**
          $y_0 \leftarrow y_0 + y\_step\_scaler \times dy_0$;
          $dy \leftarrow dx - x\_step\_scaler \times dxy$;
          $dx \leftarrow dx + x\_step\_scaler \times dx_1$;
          $err = err + x\_step\_scaler \times dx$;
        end
        $path\_queue.append[(x_0, y_0)]$
      end
    end
    $y\_curr \leftarrow y_0$; $\varepsilon_t \leftarrow error\_margin$
    while $\|y\_curr - y_2\| > \varepsilon_t$ **do**
      $y\_curr \leftarrow y\_curr + y\_step\_scaler$ $path\_queue.append[(x_0, y\_curr)]$
    end
  end
end

Quantization process on an allocated grid of an obstacle or the sampled path spline point is denoted with p_o, which is a tuple in Euclidean coordinates (x_i, y_i, z_i). Quantization of this point to the coordinate specific set levels is a special floor operation on the respected graph tree of the axis. This special floor operation returns a quantized coordinate (x_q, y_q, z_q) that can be assigned to any set level in its boundaries defined by the corresponding graph tree. Moreover, the special floor operation can be designed to assign lower, middle, upper or any required value within the grid coordinates. This operation is shown as follows,

$$x_q \leftarrow \lfloor x_i \rfloor_{G_x},\ y_q \leftarrow \lfloor y_i \rfloor_{G_y},\ z_q \leftarrow \lfloor z_i \rfloor_{G_z} \tag{3.11}$$

At this assignment, the state variables that hold the quantized position values for each axis are formed from the current vehicle and landmark positions in corresponding graph tree, which is shaped from path planners and sensor measurements collectively. This assignment visualized in Fig. 3.4.

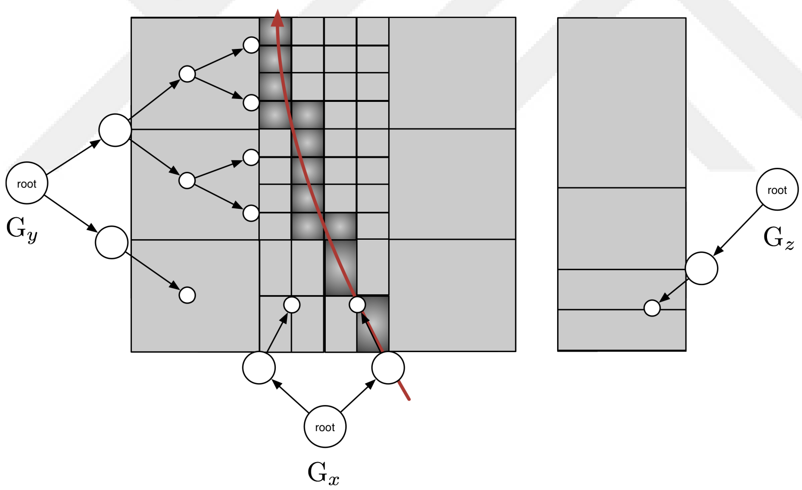


Figure 3.4 : Assignment process of the sensor inputs/path splines in quantization notation. Dark boxes are quantized values that can be assigned to any value in its boundaries.

3.4.2 Path grid allocation with vehicle steering constraints

The sparsity of the regions or in other words, the zoom-in/zoom-out ratio of a specific area in ROI is determined by two main factors. One of them is the vehicle steering constraint. If the resolution of the steering angle or the sampling rate of the odometry is higher than the dimensions of the divided sections of the sibling nodes, then duplicating

the sampling rate of the region do not get a response from vehicle controller. This phenomenon is analogous to direct filtering of the high-frequency noises with the low-pass filter characteristic of the system. Consequently, the map representation and the vehicle dynamics are considered together in this framework. For the sake of simplifications, the vehicle is assumed as a unicycle model. Linearization of the nonlinear model of unicycle model is given in Appendix A. This model is assumed to be an ideal model with no-slip conditions. Besides, more complex models as in [55] for high speeds with slip constraints is also applicable within the framework.
A trivial lower boundary for grid sizes is shown as,

$$dx \geq \upsilon \cos\phi \; \Delta\mathrm{t}, \; dy \geq \upsilon \sin\phi \; \Delta\mathrm{t} \tag{3.12}$$

where υ is the linear velocity of vehicle, and the ϕ is heading angle. Boundary representations with the unicycle model in grid space are shown in Fig. 3.5. However,

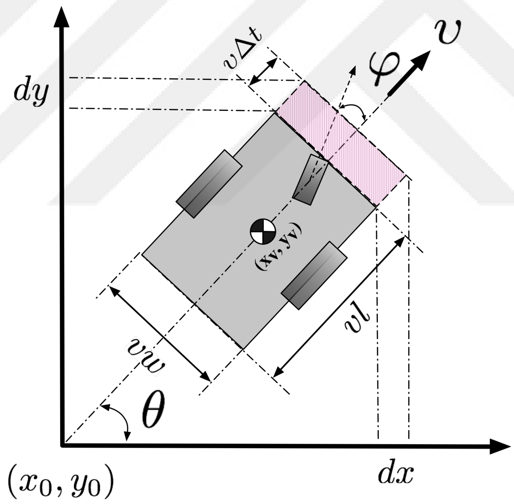


Figure 3.5 : Lower boundaries of the grid space in unicycle vehicle model.

with a slight change of the position of the vehicle, assignment can switch between the adjacent grids that have different sampling rates during the operation. This rapid mode change can cause sudden accelerations and decelerations in the movement. The proposed solution to this chattering situation is initiating a collision border around the vehicle, which serves for both collision checks and avoiding fast transition between the nodes. A method for this ellipsoidal collision border is given in Algorithm 4. This method is an extended version of the Brezenham's line algorithm to represent the closed circular objects like an ellipse. This extension serves as a handler for

Algorithm 4 Occupancy grids for ellipse collision region of the vehicle.

Input: x_0, y_0, x_1, y_1
Result: Occupied grids for vehicle
Initialization: $a \leftarrow \|x_1 - x_0\|$, $b \leftarrow \|y_1 - y_0\|$, $b_1 \leftarrow b \mod 2$, $dx \leftarrow 4 \times (1-a) \times b \times b$, $dy \leftarrow 4 \times (b_1+1) \times a \times a$, $err_{xy} \leftarrow dx + dy + b_1 + a \times a$, $err_{double} \leftarrow 0$

```
if x_0 > x_1 then
    x_0 ← x_1
    x_1 ← x_1 + a
end
if y_0 > y_1 then
    y_0 ← y_1
end
y_0 ← y_0 + (b+1)/2
y_1 ← y_0 − b_1
a ← 8 × a × a
b_1 ← 8 × b × b
while x_0 ≤ x_1 do
    path_queue.append[(x_0, y_0)] path_queue.append[(x_1, y_0)]
    path_queue.append[(x_0, y_1)] path_queue.append[(x_1, y_1)]
    err_double ← 2 × err_xy
    if err_double ≤ dy then
        y_0++; y_1−; dy ← dy + a; err_xy ← err_xy + dy
    end
    if err_double ≥ dx||||2 × err_xy > dy then
        x_0++; x_1−; dx ← dx + b_1; err_xy ← err_xy + dx
    end
end
while y_0 − y_1 < b do
    path_queue.append[(x_0 − 1, y_0)]
    path_queue.append[(x_0 − 1, y_0)]
    path_queue.append[(x_1 + 1, y_0++)]
    path_queue.append[(x_0 − 1, y_1)]
    path_queue.append[(x_1 + 1, y_1−)]
end
```

the chattering effect and specifies a safety zone to avoid collisions. In Algorithm 3, the states of the vehicle are assumed to be measured precisely. However, in real life situations, there is always an uncertainty about vehicle states or obstacle positions that need to be considered. Therefore, an explicit need for an extension to the previous occupation algorithm is evident to represent this uncertainty. This problem has a counterpart in digital image processing, which is known as anti-aliasing. Anti-aliasing filters allow expressing high-resolution images in low-resolution ones. This structure is compatible with the zoom-in/zoom-out principle of the quantization framework. For this purpose, an anti-aliasing filter is implemented to compensate the uncertainty of the vehicle position in a discrete grid environment by an extension to the Bresenham's line algorithm. Bezier curves do not have an anti-aliasing extension, and the closest alternative is the Xiaolin Wu's algorithm [56], which is slower than the Bezier spline algorithm and cannot be used for Bezier splines due to the complexity of representing each coordinate regarding other coordinates. The method chosen here uses the tangents of the Bezier curves at each sampling time. The derivative of the Bezier splines comes from the solution for the derivative of Bernstein polynomial given in Eq. (3.9),

$$\frac{\mathbf{d}}{\mathbf{dk}} B_n = n\left(B_{i-1,n-1}(k) - B_{i,n-1}(k)\right) \tag{3.13}$$

and the linear combination of the Bezier spline in Eq. (3.10) is derived by substituting Bernstein coefficients with Bezier control points to the derivative of Bernstein polynomial given in Eq. (3.13) as,

$$\frac{\mathbf{d}}{\mathbf{dk}} C_n = \dot{C}_n = \sum_{i=0}^{n-1} B_{i-1,n-1}(k)\left[n\left(P_{i+1} - P_i\right)\right] \tag{3.14}$$

Hence, for a cubic spline ($n = 3$), slope of Bezier spline is,

$$\begin{aligned} \dot{C}_3 = 3k^2 P_3 + \left(6k - 9k^2\right) P_2 + \left(3 - 12k + 9k^2\right) P_1 + \\ + \left(3 - 6k - 9k^2\right) P_0 \end{aligned} \tag{3.15}$$

A perpendicular line to the Bezier spline tangent initiates the vehicle boundaries strictly if there is a need for a clear boundary box of a vehicle instead of an ellipse collision region given in Algorithm 4. It is not possible to obtain parallel splines by Bezier's formula as a consequence of the cubic Bezier spline including left and right sides defined in a 10th order polynomial. A solution for the boundary box extension is given in Algorithm 5. In this approach, current vehicle position is received from the Bezier Spline function. Following that, the tangential lines for these points are calculated within the inline function *TangentBezierSpline*. Then, the boundaries are set by using the vehicle's volumetric dimensions. For all dimensions, there is a counter to check how many occupancy points should be generated to represent the volume occupied by the vehicle. Finally, the kinematic model of the vehicle is applied to the generated spline points and added to the cluster of occupancy path.

Algorithm 5 Boundary box calculation through qubic Bezier spline.

Input: P_0, P_1, P_2, P_3, G_x,G_y,G_z
Result: Get Bezier Points for control points
initialization; $k \leftarrow 0.0$; $cond_x \leftarrow True$; $cond_y \leftarrow True$; $cond_z \leftarrow True$

if *type=="Qubic"* **then**
  while $k \leq 1.0$ **do**
    #Get the current vehicle position on spline;
    $x_c \leftarrow BezierSpline(k, P_0.x, P_1.x, P_2.x, P_3.x)$
    $x_y \leftarrow BezierSpline(k, P_0.y, P_1.y, P_2.y, P_3.y)$
    $x_z \leftarrow BezierSpline(k, P_0.z, P_1.z, P_2.z, P_3.z)$
    #Get the current tangent intersects the spline;
    $m_x \leftarrow TangentBezierSpline(k, P_0.x, P_1.x, P_2.x, P_3.x)$
    $m_y \leftarrow TangentBezierSpline(k, P_0.y, P_1.y, P_2.y, P_3.y)$
    $m_z \leftarrow TangentBezierSpline(k, P_0.z, P_1.z, P_2.z, P_3.z)$
    #Get the sampling rates in vehicle position;
    $\delta_x \leftarrow G_x.sample\_rate(x_c)$
    $\delta_y \leftarrow G_y.sample\_rate(y_c)$
    $\delta_z \leftarrow G_z.sample\_rate(z_c)$
    #Set box points as long as either condition is true ;
    $count_x \leftarrow 0, count_y \leftarrow 0, count_z \leftarrow 0$
    while $cond_x \vee cond_y \vee cond_y$ **do**
      $cond_x \leftarrow True\,\mathbf{if}(\frac{2vw}{\delta_x} \leq count_x)\,\mathbf{else}\,False$
      $cond_y \leftarrow True\,\mathbf{if}(\frac{2vl}{\delta_y} \leq count_y)\,\mathbf{else}\,False$
      $cond_z \leftarrow True\,\mathbf{if}(\frac{2vh}{\delta_z} \leq count_z)\,\mathbf{else}\,False$

      $\psi \leftarrow \arctan2(m_y, m_x)$ # Heading angle wrt x-axis
      $\phi \leftarrow \arctan2(m_z, m_x)$ # Pitch angle in y-axis
      $\theta \leftarrow \arctan2(m_z, m_y)$ # Roll angle in x-axis

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = Rotation(\phi, \theta, \psi) \times \begin{bmatrix} x_c + \frac{vl}{2} & x_c - \frac{vl}{2} & x_c + \frac{vl}{2} & x_c - \frac{vl}{2} \\ y_c + \frac{vw}{2} & y_c + \frac{vw}{2} & y_c - \frac{vw}{2} & y_c - \frac{vw}{2} \\ z_c - vh & z_c - vh & z_c - vh & z_c - vh \end{bmatrix}$$

      bezier_box.append(x_a, y_a, z_a)

    end
    $k \leftarrow k + step\_size$

  end
end
function TangentBezierSpline(k, P_0, P_1, P_2, P_3)
  $tangent \leftarrow 3k^2P_3 + \left(6k - 9k^2\right)P_2 + \left(3 - 12k + 9k^2\right)P_1 + \left(-3 + 6k - 3k^2\right)P_0$
  Return *point*
  end function
  function BezierSpline(k, P_0, P_1, P_2, P_3)
  $point \leftarrow k^3P_3 + 3(1-k)k^2P_2 + 3(1-k)^2P_1 + (1-k)^3P_0$
  Return *point*
  end function
  function Rotation(ϕ, θ, ψ)

$$R \leftarrow \begin{bmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \sin\phi\cos\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix}$$

  Return R
  end function

3.5 Discussion

In this section, a new method is proposed to represent the occupancy grids maps. This structure has novelty in the sense of expressing the path splines with grids, which are shaped by the occurrence of the obstacles. The spatial feedback to continuous, somewhat ideal vehicle path splines defines the maneuverability boundaries. These boundaries are associated both with the environmental conditions (i.e., existence of the obstacles) and the kinematic and dynamic constraints of the vehicle. Theoretical backgrounds for quantized allocation are explained and applied for the obstacles and paths in the grid map. Path allocation is presented both for quadratic and cubic Bezier curves in an exemplary case. Vehicle constraints are established after the ideal case approach in the first place. Bresenham's line algorithm is introduced for fast real-time allocation for practical approaches. Anti-aliasing alike extension is included to compensate the uncertainty both in the vehicle positions and the sensor readings.

4. GENERAL NAVIGATION FRAMEWORK WITH QFC

4.1 Purpose

In this section, a navigation framework is proposed by using input and output quantizers to express sensor data in grid segmentations and generate reference inputs within motion planner to achieve asymptotic stability with quantized LQR (Linear Quadratic Regulator) sequentially. A new LQR system particularly qLQR is adapted from [1] to a navigation framework with theoretical and simulation results. Using quantization to generate occupancy grids from sensor data and in control system architecture is a novel approach for robot navigation problem. Achievements in this chapter are published and include material from the paper by the author [57].

This section is organized as follows. First, the general navigation framework is explained briefly in Section 4.2. After that, implementation of the sensor quantization into the framework under the input quantization concept (which is presented in Chapter 3) is represented in Section 4.3. Then, the output quantizer, which contains motion planner to generate reference inputs and quantizer that introduces a new paradigm to convert spatial sampling from input quantizer to control signals under stability constraints, by using quantized LQR in Section 4.4 is discussed. Finally, the quantized LQR method is explained from the view of traditional Linear Quadratic Regulator sense.

4.2 A Navigation Framework with Quantization: Overview

The principal objective of using quantizers is the robustness of the controllers under measurement and modeling errors. This purpose is achieved via input to state stability (ISS), which uses Lyapunov functions and small-gain theorem. The quantized variable can be any signal like measurement output, control input or state variable. The quantizer maps these continuous variables to their quantized conjugates. Succeeding sections explain the implementation of this quantization operator to describe grid maps in input quantizer and generate reference values and control a stabilizable control system with discontinuous dynamics of steering and throttle subsystems through output quantizer.

The general framework in Fig. 4.1 can be summarized (from left to right) as follows. Traffic planner block stands for the communication layer with the infrastructure. This block provides decision logic and constraints like a temporary stop in traffic lights and precedence at the intersection points or speed limits in different roads, which are also resources of discontinuity in navigation problem. Obstacle detection layer represents pre-occupation of the sensor data into initial grid layout in a raw form or after some outlier elimination process [58, 59]. *Input quantizer* samples the 3D sensors as an input signal to path and motion planner by taking the static and dynamic obstacles and the current vehicle states into consideration. The output of the motion planner yields a maneuverable path formed from grid cells. Each grid cell is information encoded shaped reference input to the system. At this point, the second quantizer participates in the framework, which is called *output quantizer*. Output quantizer transforms the motion planner outputs (linear and angular velocities) to the corresponding discrete levels. Output quantizer produces reference inputs to the low-level controllers within the last layer. These controllers are typical PID controllers in the actuator drives with protections like anti-windup, saturation delimiters.

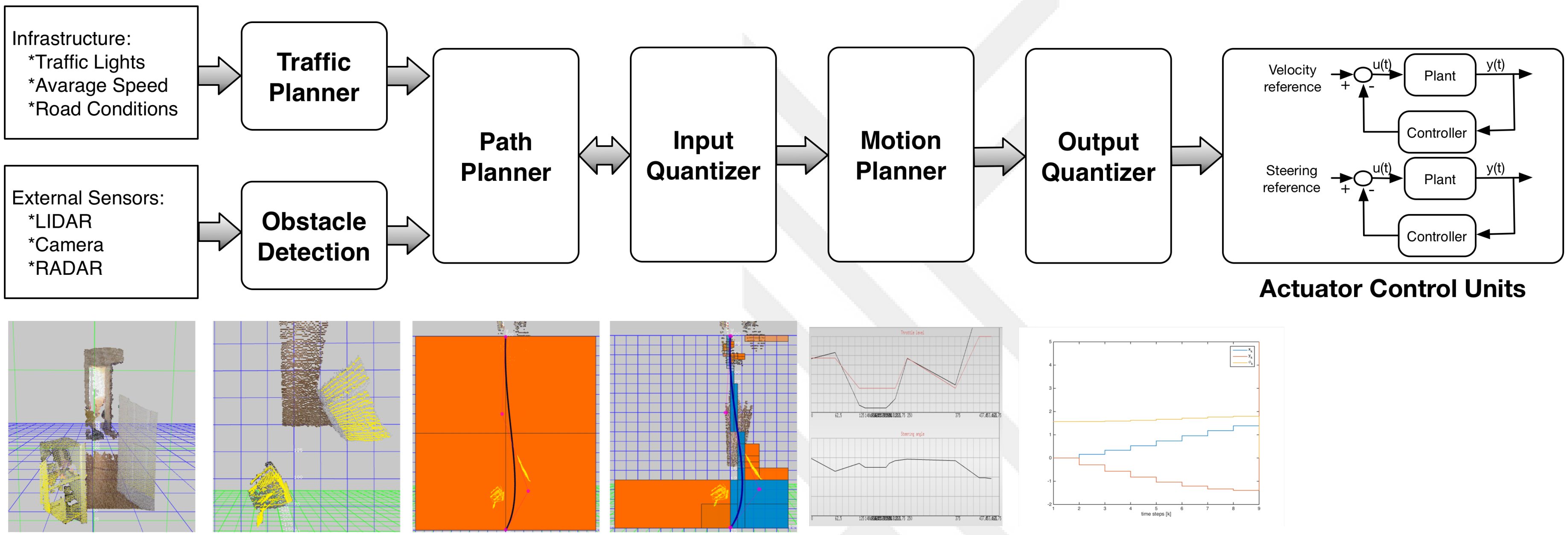


Figure 4.1 : General Framework of the Quantized Feedback Control System.

4.3 Input Quantizer

There are two cases that should be considered for stabilization of the system under quantization. The first case is the quantization levels, which are dynamic in the creation of occupancy grids in Region of Interest (ROI) and static in vehicle control systems. The second case is to find a feedback control law that stabilizes the system. First of all, the system model is derived from the mathematical model of Ackermann type autonomous/automated guided vehicle (AGV). The vehicle kinematics for the vehicle model in Fig. 3.5 is given as a nonlinear function f, subject to state and control variables (or constraints),

$$\dot{x} = f(x,u) \tag{4.1}$$

The vehicle state variables x, y are locations in x-y coordinates, θ is vehicle orientation to global origin with linear velocity v and steering angle φ. Control variables of the vehicle are linear acceleration a and angular velocity ω. In vector notation:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \varphi \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} a \\ \omega \end{bmatrix}$$

Eventually, the open form of the nonlinear system dynamics given in Eq. (4.1) can be derived as follows:

$$\dot{\mathbf{x}} = \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} x \\ y \\ \theta \\ v \\ \varphi \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{\theta}\tan(\varphi) \\ a \\ \omega \end{bmatrix} \tag{4.2}$$

For every point (x_i, y_i, z_i) in the Bezier path curve, there is a slope variable (mx_i, my_i, mz_i) for each dimension with respect to the sampling steps of the path curve as follows,

$$\begin{aligned} mx_i &= \tfrac{x_{i+1} - x_i}{x\_step\_scaler} \\ my_i &= \tfrac{y_{i+1} - y_i}{y\_step\_scaler} \\ mz_i &= \tfrac{z_{i+1} - z_i}{z\_step\_scaler} \end{aligned} \tag{4.3}$$

Here the x_step_scaler, y_step_scaler and z_step_scaler are the Bezier path sampling rates used in the Algorithm 3, which are the $step\_size$ variable associated with each axis in Algorithm 2. By using the graph trees associated with the Euclidean space dimensions and the slope of the Bezier path grids, quantization process on vehicle state variables expressed as follows,

$$\begin{gathered} x_i \leftarrow G_x(x_i),\ y_i \leftarrow G_y(y_i),\ z_i \leftarrow G_z(z_i) \\ \psi_i \leftarrow \operatorname{arctan2}(my_i, mx_i), \phi_i \leftarrow \operatorname{arctan2}(mz_i, mx_i), \\ \theta_i \leftarrow \operatorname{arctan2}(mz_i, my_i) \end{gathered} \tag{4.4}$$

In this assignment, state variables that denote the positions for every axis, get values from the current vehicle position in corresponding graph tree, which is shaped by path and motion planners. Roll, pitch and yaw angles of the vehicle take values from the tangents of the path spline. This assignment visualized in Fig. 4.2. Input planner

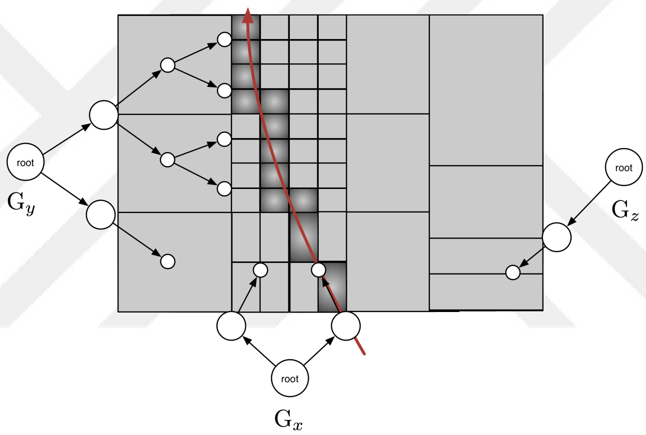


Figure 4.2 : Assignment of the sensor inputs in input quantization block.

can also return feedback input to the path planner if the minimum gridding size is exceeded. The mission of the input quantizer is to divide the region into grid partitions until minimum allowed grid size, which is independent for each axis. The detailed input quantizer block of the framework is given in Fig. 4.3.

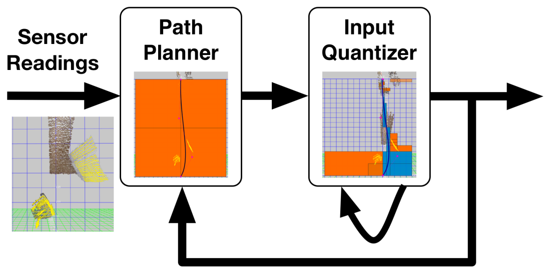


Figure 4.3 : Input Quantizer and Path Planning blocks.

Division starts with coarse grids on parent nodes. Grids are assigned to the path (which are shown with blue tetragons in Fig. 4.3) if segmented path chunks (generated by using one of RRT planners in [20, 60, 61]) are located in grid regions. Next, the point clouds that belong to obstacles are initiated to the system via external sensor inputs. Depending on these inputs; some of the grids are occupied if there is an obstacle in grid region (which are shown with red tetragons in Fig. 4.3). In each step, the grids become denser with splitting a coarse grid into a new grid with the half size (not necessarily to be symmetrical in all axes). This division is iterated until the occupied obstacle grids, and grids associated with the path chunks are not overlapped (Fig. 4.4). All grids are symbolized with a graph tree data structure, which is independent in each axis. If the overlapping thus far exists to the minimum grid size allowed (which is determined by the steering and throttle actuator constraints, vehicle dimensions), then a feedback signal is generated to pick a new route among the alternative solutions from path planner.

The very first path is inducted through the internal processes of the selected path planner algorithm. However, the afterward process is executed with the cooperation of the input quantizer. Grid segmentation logic is shown in Fig. 4.5.

4.4 Output Quantizer

The output quantizer of the controller takes reshaped quantized reference inputs from motion planner and generates outputs harmonious with the controllers sampling levels. After finding non-overlapping grid segmentation through the selected path, formed grid partitions are used in the motion planner section to generate velocity and steering angle references. Intuitively, if the path goes through in a close proximity of an obstacle, then the grids become denser in these areas. Also, the steering angle gets a value between the hypotenuse of the grid tetragon and the tangent of the spline. These references are assigned from a proper command set of the actuator drives reference inputs. As a result of the motion planner layer, reference values are obtained for path traction. However, the stability of the system should be adequately sustained while sweeping through the reference values. At this stage, the second quantizer type in the system, the *output quantizer* becomes a part of the framework. The connection between these blocks is illustrated in Fig. 4.6.

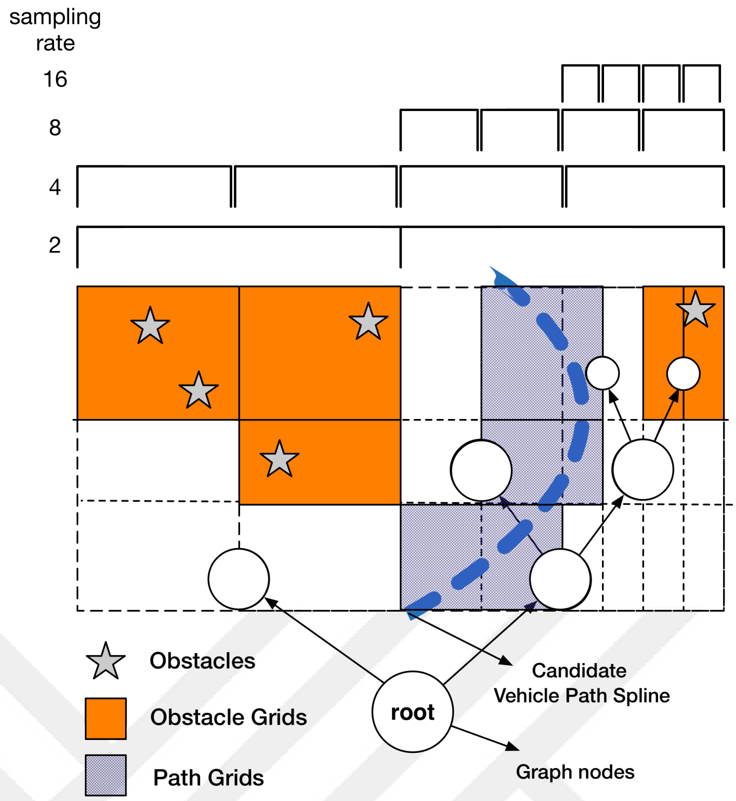


Figure 4.4 : Representation of the obstacle grids and path grids with node graphs in Input Quantizer.

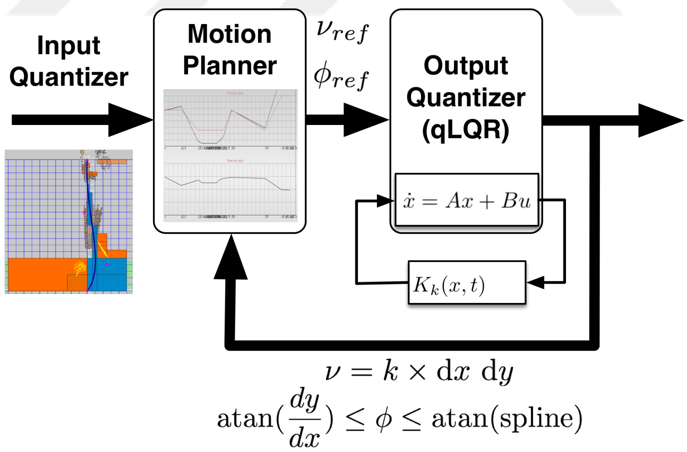


Figure 4.6 : Motion Planner and Output Quantizer blocks.

In the simplest case of modeling an Ackerman type vehicle, there are two control variables ν and ω, linear and angular velocities respectively. If one need a smooth ride, then use of the derivatives $\dot{\nu}$ and $\dot{\omega}$ is necessary in control system. The linear velocity of the vehicle can be assigned as fixed set values similar to the generic approaches in the navigation literature. However, in this study, the grid size of each axis assures

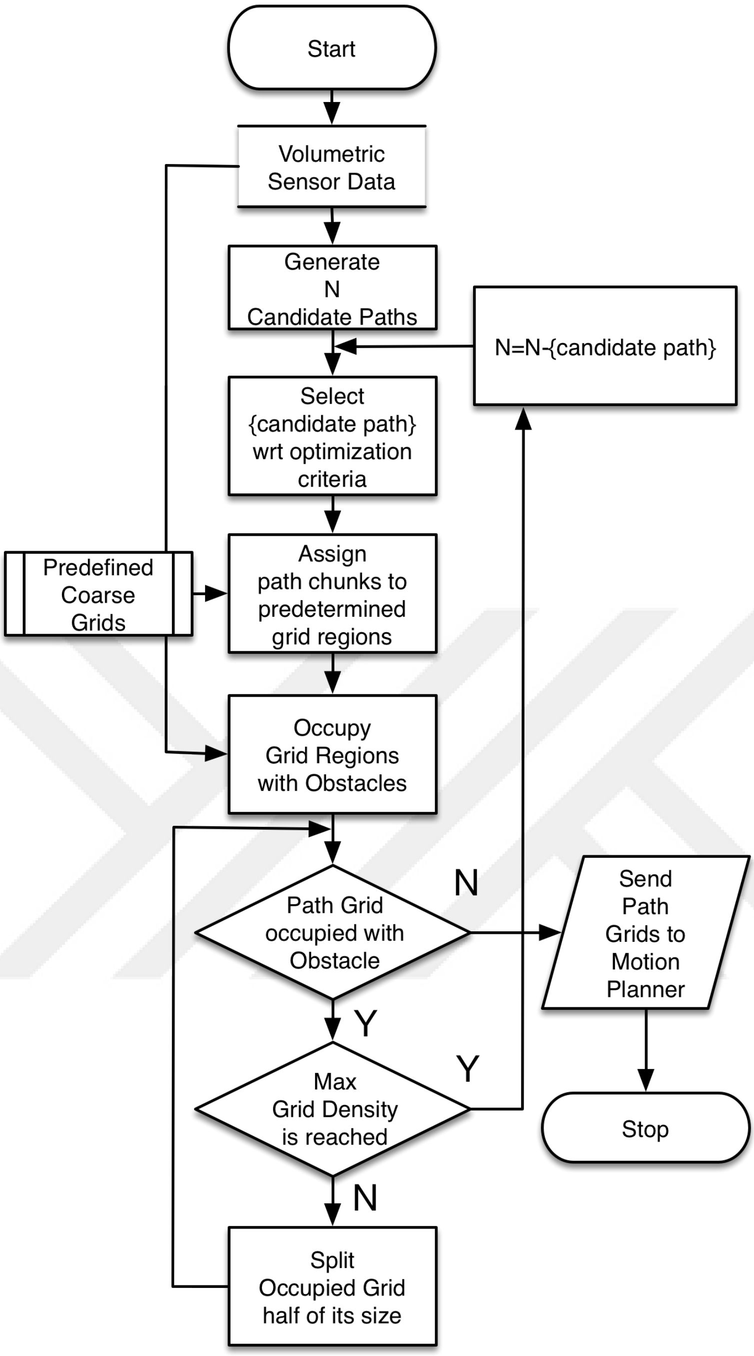


Figure 4.5 : Flow chart of the grid segmentation logic.

a throughput that can be used to select the optimal velocity set values. To do this, expected mean value of the velocity $\bar{v}$ is defined in terms of the grid structure,

$$\bar{v} = \frac{\sqrt{(dy_0 + dy^+)^2 + (dx_0 + dx^+)^2}}{\Delta t} \tag{4.5}$$

Here, dx_0, dy_0 are the spatial samplers of the current grid and dx^+, dy^+ are the spatial samplers of the next grid that is predicted in motion controller (Fig. 4.7). Quantized value of the expected mean velocity is denoted as $\lfloor \bar{v} \rceil$. This special floor operator is a

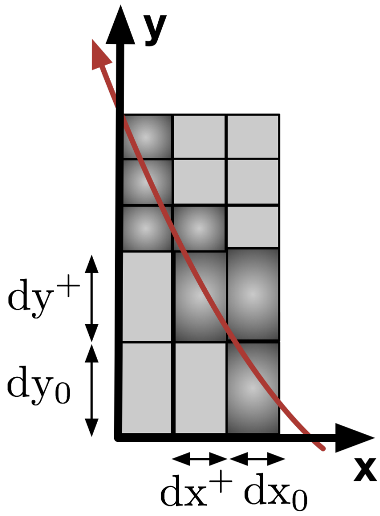


Figure 4.7 : Visualization of the parameters in expected mean velocity calculation.

direct consequence of the vehicle throttle levels which is shown in an example with a gear-shift mechanism within Fig. 4.8.

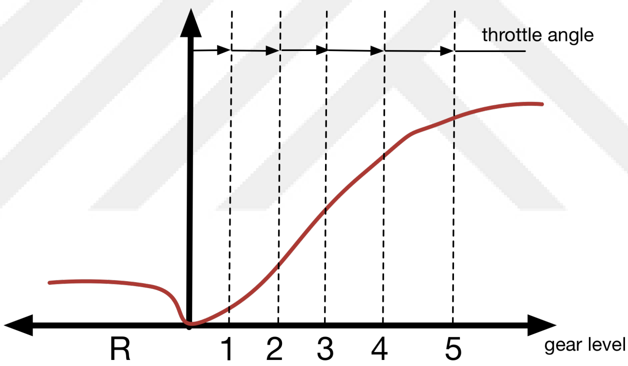


Figure 4.8 : Throttle with a gear-shifting is an example of switched system.

Heading angle of the vehicle is either can be found by using the tangent of the curvature (i.e., $\psi = \lfloor \arctan2\,(my, mx) \rfloor$) or tangent of sequential grids (i.e., $\psi = \lfloor \arctan2\left(\frac{dy_0+dy^+}{dx_0+dx^+}\right) \rfloor$). The latter one is definitely a coarse estimation than the former one. State space of the closed-loop system with these control inputs can be found as follows,

$$
\begin{aligned}
x_{k+1} &= Ax_k + Bq_k \\
Q(y_k) &= Cx_k
\end{aligned}
\tag{4.6}
$$

Here, $q_k = Q(u_k)$ is the quantized output of the output quantizer to the controller. Output quantizer signal with state feedback is defined as,

$$\begin{aligned} q_k &= r_k + Kx_k \quad (4.7) \\ r_k &= \begin{bmatrix} \lfloor \bar{v} \rfloor_{V_t} \\ \lfloor \bar{\psi} \rfloor_{V_s} \end{bmatrix} \quad (4.8) \end{aligned}$$

K is the gain matrix that gives us flexibility in controller design. Special floor functions $\lfloor\ \rfloor_{V_t}$ and $\lfloor\ \rfloor_{V_s}$ quantize the reference output from motion planner to the system-specific set levels for throttle and steering. The chosen optimization criteria is minimizing the quantized values of the system outputs (i.e., state variables where $\mathbf{C} = \mathbf{I}(n)$) which is defined as,

$$\begin{aligned} \underset{error}{\text{minimize}} \quad & y_k - Q(y_k) \\ \text{subject to} \quad & \dot{x} = f(x,u). \end{aligned} \quad (4.9)$$

Starting from this point of view, state space equations can be written as follows,

$$\begin{aligned} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}q_k \\ &= \mathbf{A}x_k + \mathbf{B}\left(r_k + \mathbf{K}x_k\right) \\ &= \left(\mathbf{A} + \mathbf{BK}\right)x_k + \mathbf{B}r_k \\ &= \mathbf{A}_c x_k + \mathbf{B}r_k \end{aligned} \quad (4.10)$$

Here $\mathbf{A}$ is a constant-valued system matrix and relates the state changes with previous states. The term $\mathbf{B}$ is the control matrix and has a dependency on heading angle ψ. Apparently, with static quantization levels of the controller denoted by r_k the only alternative to gain control surface is through $\mathbf{K}$ gain matrix. By changing values of the gain matrix, one can determine the eigenvalues of the closed-loop system. The system with a linear state-space model is asymptotically stable, if all eigenvalues of the $\mathbf{A}$ are negative. This condition is equivalent to the following Lyapunov equation in terms of Lyapunov stability.

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{PA} = -\mathbf{Q} \quad (4.11)$$

The term $\mathbf{P}$ is positive definite Hermitian matrix and $\mathbf{Q}$ is a positive definite matrix to form left-hand side of the equation as negative definite. The corresponding Lyapunov function is,

$$V(x) = x^{\mathrm{T}}\,\mathbf{P}\,x \quad (4.12)$$

For the same linear system, applying the Lyapunov stability equation in Eq. (4.11) to closed-loop stabilizable control system under the conditions $B \neq 0$, $K \neq 0$ gives,

$$\begin{aligned} \mathbf{A}_c^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A}_c &= -\mathbf{Q} \\ (\mathbf{A}+\mathbf{B}\mathbf{K})^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{B}\mathbf{K} &= -\mathbf{Q} \end{aligned} \tag{4.13}$$

Parameters λ_{min} and λ_{max} stand for the minimum and maximum eigenvalues of the system respectively. These are the decisive parameters of the system boundaries and hold for the following inequality.

$$\lambda_{min}(P)\,|x|^2 \leq x^{\mathrm{T}}Px \leq \lambda_{max}(P)\,|x|^2 \tag{4.14}$$

The feedback law cannot be implemented directly because of the quantization takes place in the control system. Thus, quantizer $q_\mu(.)$ is defined for a variable z as,

$$q_\mu(z) := \mu q_\mu \left(\frac{z}{\mu}\right) \tag{4.15}$$

μ is a strictly positive scalar value ($\mu > 0$). One can think μ denoting a zoom variable as mentioned in [45]. Increasing μ value yields coarser quantization, which increases the ROI and at the same time quantization error. Conversely, decreasing μ value leads to a denser quantization that decreases the quantization error but limits the horizon range. The range of quantizer defined as $ROI = R\mu$ and quantization error with $\Delta\mu$. Proof of asymptotic stability holds for zoom-in/out cases given in Appendix D.
The output quantizer can be formed as state feedback,

$$u = Kq_\mu(x) \tag{4.16}$$

For a closed-loop system, derivation of the quantized version of the Eq. (4.11) is as follows,

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u \\ &= \mathbf{A}x + \mathbf{B}\mathbf{K}\mu q_\mu\left(\frac{x}{\mu}\right) \\ &= (\mathbf{A}+\mathbf{B}\mathbf{K})x + \mathbf{B}\mathbf{K}\mu\left(q_\mu\left(\frac{x}{\mu}\right) - \frac{x}{\mu}\right) \end{aligned} \tag{4.17}$$

To inspect the behavior of the system stability, one should check the trajectories of the system given in Eq. (4.17). Principally, the overbounding polytopes are defined as stability regions of the state space. The ball B_1 is associated with the state value range, which is defined with $R\mu$ before.

$$B_1 := \{x : |x| \leq R\mu\} \tag{4.18}$$

The second polytope is defined to specify the lower boundary of the state space. At this stage, Lyapunov stability theorem can be used to define lower boundaries of the state space. Suppose there is a *Lyapunov function* V, which has the following gradient,

$$\dot{V}(x) \leq 0,\ \forall x \neq 0,\ \dot{V}(0) = 0 \tag{4.19}$$

The common interpretation of the Lyapunov function V is a generalized energy function which continuously loses energy (except at the origin). For this system, Lyapunov function defined as in Eq. (4.12).

$$V(x) = x^{\top}\ \mathbf{P}\ x \tag{4.20}$$

Substituting closed-loop system in Eq. (4.17) into Eq. (4.19) satisfies,

$$\begin{aligned} \dot{V}(x) &= -x^{\top}\ \mathbf{Q}\ x + 2x^{\top}\mathbf{PBK}\mu\left(q_{\mu}\left(\frac{x}{\mu}\right) - \frac{x}{\mu}\right) \\ &\leq -\lambda_{min}(Q)\,|x|^2 + 2|x|\|\mathbf{PBK}\|\Delta \\ &\leq -|x|\lambda_{min}(Q)\,(|x| - \Theta_x\Delta) \end{aligned}$$

where,

$$\Theta_x := \frac{2\|\mathbf{PBK}\|}{\lambda_{min}(Q)} \tag{4.21}$$

For a small $\varepsilon > 0$, boundaries can be set for the system states,

$$\Theta_x\Delta(1+\varepsilon) \leq |x| \leq R\mu \tag{4.22}$$

Therefore, the rate of change of the Lyapunov function in Eq. (4.21) has a lower limit, which is bounded by the lower thresholds of the state variables. This lower limit constitutes the second polytope B_2,

$$B_2 := \{x : |x| \leq \Theta_x\Delta(1+\varepsilon)\} \tag{4.23}$$

Ellipsoid for the operating point is formed as,

$$R_1(\mu) := \left\{x : x^{\top}\mathbf{P}x \leq \frac{\lambda_{min}(\mathbf{P})\,(R\mu)^2\mu^2}{\|\mathbf{K}\|^2}\right\} \tag{4.24}$$

Ellipsoids for attraction regions are defined as,

$$R_2(\mu) := \left\{ x : x^\top \mathbf{P} x \leq \lambda_{max}(\mathbf{P}) \mathbf{Q}^2 \Delta^2 (1+\varepsilon)^2 \mu^2 \right\} \tag{4.25}$$

All of these operating regions are illustrated in Fig. 4.9.

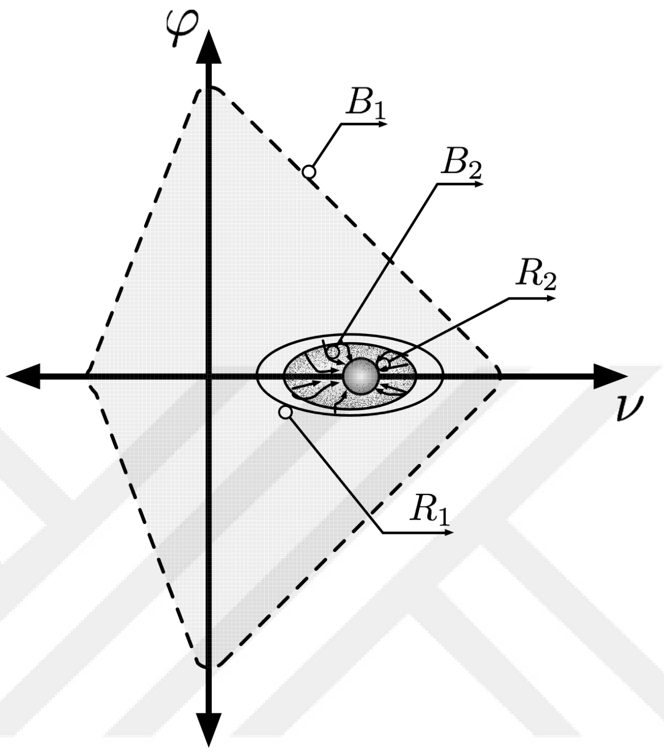


Figure 4.9 : Boundary polytopes of control variables for Lyapunov stability. B_1 region denotes the state minimum and maximum values (max velocity and steering angles). R_1 is the selected operating or linearization point of vehicle dynamics. R_2 is the attraction (both appear as discontinuity or stability surface) region. B_2 shows the rate of change constraints of the state variables because speed and steering angle are dependent variables in terms of stability analysis.

All solutions initialized in region $R_1(\mu)$ will be entered to inner region $R_2(\mu)$ in finite dwell time given by the following formula in [45].

$$T := \frac{\lambda_{min}(\mathbf{P})(R\mu)^2 - \lambda_{max}(\mathbf{P}) \mathbf{Q}^2 \|\mathbf{K}\|^2 \Delta^2 (1+\varepsilon)^2}{\mathbf{Q}^2 \|\mathbf{K}\| \Delta^2 \varepsilon (1+\varepsilon) \lambda_{min}(\mathbf{Q})} \tag{4.26}$$

Asymptotic stability proof for a quantized feedback controller is also given in [45]. Adaptation of this proofs to the proposed framework is given in Section 4.5.

Output quantizer layer is the most sophisticated part of the framework from the aspect of control theoretical practice. In Section 4.5, an adaptation of hybrid control systems in [1] with a quantized LQR controller is introduced as a stabilizable controller for the framework, when there is traditional LQR controller can be found for a controllable and stabilizable system. The reason behind that a quantized LQR controller is preferred instead of a traditional LQR controller is to take account of the reference levels that can be able to follow by the low-level actuators. Without considering the low-level capability of the system, one is always to be mistaken on the assumption of a perfect low-level actuation system, and this causes undesirable results in real-life scenarios. From the nature of the switching systems, switching between two stable modes of control operations may cause unstable behaviors. On the other hand, switching between two unstable modes may cause a stable operation, which is contrary to the expected response of the system.

4.5 Quantized LQR for Output Quantizer

In this section, the control system, which is denoted as K in previous sections, is derived for output quantizer in the framework. First, the idea of stabilization for traditional Linear Quadratic Regulator (LQR) is explained in discrete systems and expands it to the new approach of quantized LQR (qLQR) structure. Both theoretical results and simulation results are given in order.

4.5.1 Optimal control for discrete system

Discrete time system with k sampling steps is defined as follows,

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k \tag{4.27}$$

with given initial condition x_0. The design objective is to find an optimal control u_k^* so that the performance index J, which is designed for the control demands,

$$J = \frac{1}{2}\, x_N^T S_N x_N + \frac{1}{2} \sum_{k}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{4.28}$$

can be minimized. Here N is the final time, S, Q and R are the weight coefficient matrices for the final state, run-time states and the control inputs. To solve this optimization problem, first, a Hamiltonian function H_k needs to be derived with Lagrange multiplier p,

$$H_k(x_k, p_{k+1}, u_k) = \frac{1}{2}\, x_k^T Q x_k + \frac{1}{2}\, u_k^T R u_k + p_{k+1}^T A x_k + p_{k+1}^T B u_k \tag{4.29}$$

The necessary conditions where the performance index is to be minimum or maximum (i.e., optimum) are given as follows,

$$\begin{aligned} \frac{\partial H_k}{\partial u_k^*} &= 0 = Ru_k^* + B^T p_{k+1}^* && (4.30) \\ \frac{\partial H_k}{\partial p_k^*} &= x_{k+1}^* = Ax_k^* + Bu_k^* && (4.31) \\ \frac{\partial H_k}{\partial x_k^*} &= p_k^* = Qx_k^* + A^T p_{k+1}^* && (4.32) \end{aligned}$$

Optimal control function for given discrete system can be found from Eq. (4.30) as,

$$u_k^* = -R^{-1}B^T {p_{k+1}^*}^T \tag{4.33}$$

Substituting the optimal control signal in Eq. (4.33) to Eq. (4.31) gives,

$$x_{k+1}^* = Ax_k^* - BR^{-1}B^T p_{k+1}^* \tag{4.34}$$

Using the Eq. (4.32) and Eq. (4.34), one can show the feature states and the Lagrange multipliers in left-hand side,

$$\begin{aligned} x_{k+1}^* &= Ax_k^* - BR^{-1}B^T p_{k+1}^* && (4.35) \\ A^T p_{k+1}^* &= p_k^* - Qx_k^* && (4.36) \end{aligned}$$

Conjugated equations Eq. (4.35) and Eq. (4.36) are difficult to solve. However, there is a special solution for this case,

$$p_k = S_k x_k \tag{4.37}$$

Rearranging the Eq. (4.35) with Eq. (4.37) gives,

$$x_{k+1}^* = \left(I - BR^{-1}B^T S_{k+1}\right)^{-1} Ax_k^* \tag{4.38}$$

It can be shown that the S_{k+1} is positive semi-definite and thus inverse will always exist. Substituting the p_k term in Eq. (4.37) into Eq. (4.36) brings us,

$$A^T S_{k+1} x_{k+1}^* = (S_k - Q)\, x_k^* \tag{4.39}$$

By using Eq. (4.39) with Eq. (4.38),

$$A^T S_{k+1} \left(I - BR^{-1}B^T S_{k+1}\right)^{-1} Ax_k^* = (S_k - Q)\, x_k^* \tag{4.40}$$

If one inspects the Eq. (4.40), can mark the equation is independent of the states when $x_k^* \neq 0$,

$$A^T S_{k+1} \left(I - BR^{-1}B^T S_{k+1}\right)^{-1} A = (S_k - Q) \tag{4.41}$$

Eq. (4.41) is called as *Discrete Riccati Equation*. Here, S_k is referred as *Riccati matrix*. To find the optimal control, substituting the Eq. (4.37) to Eq. (4.33) gives,

$$u_k^* = -R^{-1}B^T S_{k+1} x_{k+1}^* \tag{4.42}$$

with substitution x_{k+1} term of Eq. (4.38) into Eq. (4.42),

$$\begin{aligned} u_k^* &= -R^{-1}B^T S_{k+1}\left(I - BR^{-1}B^T S_{k+1}\right)^{-1} A x_k^* \\ &= -\left(R + B^T S_{k+1}\right)^{-1} B^T S_{k+1} A x_k^* \\ &= -K_k x_k^* \end{aligned} \tag{4.43}$$

here K_k is the state feedback coefficient matrix for the optimal controller,

$$K_k = \left(R + B^T S_{k+1} B\right)^{-1} B^T S_{k+1} A \tag{4.44}$$

Also, the Riccati term can be extracted using Eq. (4.44) and Eq. (4.41),

$$S_k = Q + A^T S_{k+1}\left(A - BK_k\right) \tag{4.45}$$

Some rule of the thumb information for performance index are,

- Increasing the Q, increases the bandwidth,
- Increasing exclusively the diagonals of the Q, increases the damping ratio,
- Response of the state x_j can be accelerated by increasing the diagonal entry q_{jj} in the weight matrix Q

4.5.2 Quantized LQR as minimum energy controller

If an LTI system $[A,B]$ is *stabilizable*, it is also quadratically stabilizable. Thus, there is a state-feedback control input u that shapes the states and performs in conformity with a decreasing Lyapunov function, which is an indicator of the stabilization. These Lyapunov functions are called *control Lyapunov functions* (CLF) [62]. A quadratic CLF is constituted by,

$$V(x) = x_k^T S_k x_k \tag{4.46}$$

The gradient of the Lyapunov function is denoted as $\Delta V(x)$ and it should always have a negative sign if $V(x)$ is a continuously decreasing function.

$$\Delta V(x) := V(x_{k+1}^*) - V(x_k^*) < 0 \tag{4.47}$$

Now, the problem is transformed into finding a control set U that minimizes the given CLF in Eq. (4.46),

$$U : \{u_i \in \mathbb{R}; i \in \mathbb{Z}\}$$

a quantizer is a function that maps the state set X to the control set U with one to one correspondence,

$$f : X \mapsto U$$

where,

$$X : \{x_i \in \mathbb{R} \big| f(x) = u_i,\ i \in \mathbb{Z}\}$$

Substituting the quantizer into the Eq. (4.47) and using the discrete system equation $x_{k+1} = Ax_k + Bu_k$,

$$\Delta V(x) = V(Ax_k + Bf(x_k)) - V(x_k) < 0 \tag{4.48}$$

The problem of the quantization is to find the minimum coarsest control level that stabilizes the system. Hence, let ρ is a multiplier on the unit control set U. The aim is to find the ρ value that minimizes the performance index,

$$f : X \mapsto \beta U,\ \ X : \{x_i \in \mathbb{R} \big| f(x) = \beta u_i;\ i \in \mathbb{Z}, \beta \in \mathbb{R}, \beta > 0\} \tag{4.49}$$

Eq. (4.48) shows there is no loss of generality with the scaling [48] and hereafter, the β value is assumed to be $\beta = 1$, and CLF is a robust CLF for these given fixed control values. Like the expanded version of standard LQR controllers in Eq. (4.42), same equation can be written for the quantized LQR controller;

$$u_k^* = -\left(R + B^T S_{k+1} B\right)^{-1} B^T S_{k+1} A x_k^* = K_k x_k^* \tag{4.50}$$

Closed-loop system transition matrix can be written using new feedback matrix in Eq. (4.50),

$$A_{ck} = A + BK_k \tag{4.51}$$

So, the new CLF gradient $\Delta V(x)$ becomes,

$$\Delta V(x) = V(x_{k+1}) - V(x_k) \tag{4.52}$$

$$= x^T A_{ck} S_k A_{ck} x - x^T S_k x \tag{4.53}$$

$$= x^T \left(A_{ck} S_k A_{ck} - S_k\right) x \tag{4.54}$$

The middle term $(A_{ck}S_kA_{ck} - S_k)$ corresponds to the Q term in the classical LQR performance index in Eq. (4.28). Considering $\Delta V(x) < 0$, a positive Q_k matrix is constructed as,

$$\begin{aligned} Q_k &= S_k - A_{ck}S_kA_{ck} \\ &= S_k - A^T S_k A - \\ &\quad - \left(R + B^T S_{k+1} B\right)^{-1} A^T \left(S_{k+1} B B^T S_{k+1}\right) A \end{aligned} \tag{4.55}$$

After determining the quantized LQR law, what follows next is the control inputs with boundaries. Let $V(x) = x_k^T S_k X_k$ a CLF with $S_k > 0$ and positive semi-definite Riccati matrix. A quantizer $f : X \mapsto U, f(x) = u$ is defined and the control law is,

$$U = \{\pm u_i : u_{i+1} = \rho u_i, i \in \mathbb{Z}\} \tag{4.56}$$

Control law satisfies an interval, which is the solution of the $\Delta V(x) = 0$. If one extends the Eq. (4.48),

$$\begin{aligned} \Delta V(x) &= V(x_{k+1}) - V(x_k) \\ &= x_{k+1}^T S_{k+1} x_{k+1} - x_k^T S_{k+1} x_k \\ &= x^T \left(A^T S_k A - S_k\right) x + \\ &\quad + 2x^T A^T S_k B u_k + u_k^T B^T S_k B u_k \end{aligned} \tag{4.57}$$

$u^{(1)}$ and $u^{(2)}$ are the roots of the Eq. (4.57) and one can found the solution as,

$$u^{(1),(2)} = \frac{B^T S_k A Q_k^{-1} A^T S_k B}{B^T S_k B} \pm \sqrt{\frac{B^T S_k A Q_k^{-1} A^T S_k B}{B^T S_k B}} \tag{4.58}$$

These roots are the control boundaries for the quantized controller. Now, the optimization problem can be expressed as follows. The objective in the optimization is to find the *coarsest* quantizer that satisfies the control boundaries $[u_k^{(1)}, u_k^{(2)}]$ and decreases the performance index in every step(i.e. $\forall k \Delta V(x_k) < 0$). The performance criteria is defined as the *minimum energy control* that stabilizes the system.

$$\min_{x_{k+1} = Ax_k + Bu_k} \sum_{k=0}^{\infty} u_k^2 \tag{4.59}$$

Positive semi-define Riccati equation for this system is,

$$S_k^* = A^T S_k^* A - \left(B^T S_k^* B + 1\right)^{-1} A^T S_k^* B B^T S_k^* A \tag{4.60}$$

besides, Eq. (4.60) also gives the same solution of the following LQR problem,

$$\min_{x_{k+1}=Ax_k+Bu_k} stable \sum_{k=0}^{\infty} u_k^2 = x^T R x \tag{4.61}$$

where $R = S_k^*$. State feedback gain is also evaluated as,

$$K_{lqr} = \left(B^T R_k B\right)^{-1} B^T R_k A \tag{4.62}$$

4.6 Discussion

In this chapter, a new framework for robot navigation problem concerning the asymptotic stability of the system with input and output quantizers is proposed. In control theory, hybrid or switched systems literature produce multiple approaches to solve stability and control under nonlinearity for many industrial usages, including robotics. However, these application areas on robotics restricted with only control designs or optimizations on multi-robot swarm formations. Using quantization for representing the occupancy grid maps and control the discontinuous dynamics of robot systems is novel to this study.

First, theoretical backgrounds for quantization as a projection to the robot navigation problem have given. Quantized control framework is represented in general and in detail as input and output quantizers. The qLQR controller in the output quantizer is derived with similar steps in the traditional LQR derivation techniques, which is explained in detail. Next, the upper and lower boundaries for the control signal that ensures asymptotical stability for the system have derived. The results are given both in the simulation environment and real-time experiments including the comparison between traditional LQR and qLQR for an arbitrary run. In this particular application of the framework with LQR derived qLQR quantizers have limitations likely the same as the limitations in LQR controllers. Obtaining an analytical solution for the Riccati equation could be difficult in complex systems. The states that are used in the state feedback could not be observed in all situations. Thus, an observer system should be implemented to the system when observability issues are occurred. Using the quantizers eliminates unbounded input nature of traditional LQR methods and this assures the bounded-input, bounded-output (BIBO) stability for the system in the linearization point.

5. CASE STUDIES

5.1 Purpose

In this chapter, simulation and the real-time results are given to conclude effectiveness of the contribution of this thesis.

5.2 Real-time Experimental Results for QFC Framework

Both the simulation results and experiments are implemented and illustrated to verify the effectiveness of the algorithms. In the Fig. 5.1, simulation results are shown for multiple (4, 8, 16 and 32) grid resolutions. Here the lines with big (blue) circles represent four different Bezier quadratic curves that are calculated by using Algorithm 2. The small (yellow) dots are the 2D laser scan data (here count is limited with 20 for demonstration purposes). The red (gray in black&white print) squares are the occupancy carpets of the obstacles. Black squares show overlapping partitions of the path and the obstacle grids. Grid density is increased from upper left figure to the lower right figure by the multiples of two in both dimensions. For each step, individual nodes represent both the obstacles and path splines.

A real life equivalence of this representation is demonstrated by using a 3D Laser sensor data to generate point clouds in Fig. 5.2. This figure shows the general concepts with a Bezier spline that passes the waypoints with corresponding grids, obstacles in the real environment and their representation in occupancy grid space. Second experiment in Fig. 5.3 demonstrates the independent node depths, hence different sampling rates in all dimensions with bird eye view. Red (dark gray in black&white print) rectangles indicate the obstacle grids and blue (light gray in black&white print) rectangles indicate the path allocations like before. In the beginning, grid graphs of both dimensions (G_x, G_y) have one root node with two siblings. Thus, the ROI is separated into two halves, and very coarse representation occurs for obstacles and the path, which is shown in Fig. 5.3(a). Using the space partitioning algorithms that are

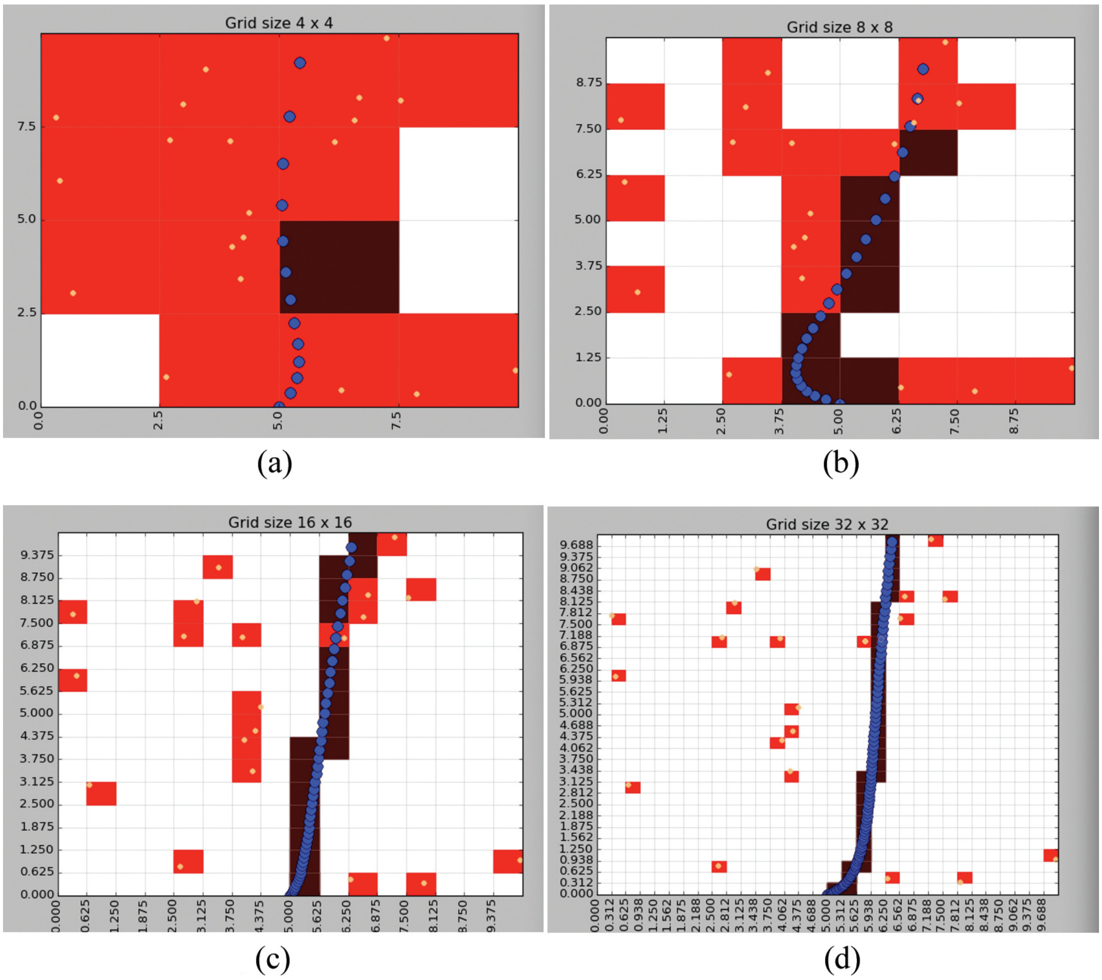


Figure 5.1 : Occupancy Grid allocation with different resolutions. Notice that this demonstration is symmetric in both axes. Red squares represent the occluded areas from the simulated sensor readings, which are shown in small green dots. Predetermined Bezier curve based paths are shown in blue -bigger- dots. Black squares are the path grids cells which are not masked with obstacle cells. (a) ROI divided to 4 by 4 regions, path is unachievable. (b) ROI divided to 8 by 8 and path grid is partly occluded. (c) ROI divided to 16 by 16. The path now only has one barrier. (d) ROI divided to 32 by 32. Now vehicle is free to go in unoccupied path.

given in the Chapter 3, dimensions are divided into different partitions on the existence of obstacles in the path route locations. This effect can be seen clearly in Fig. 5.3(b). While the left half of the obstacle grids remains coarser, the right half plane grids are denser due to the path route passes nearby. This procedure proceeds continuously in real-time, so it is convenient for representing both static and dynamic objects in the framework. This approach has a disadvantage when the ROI is selected with large scale with a long path. In this case, further objects that intersect with the path may affect the spatial sampling rate along their axes. Using cascade ROI areas or setting

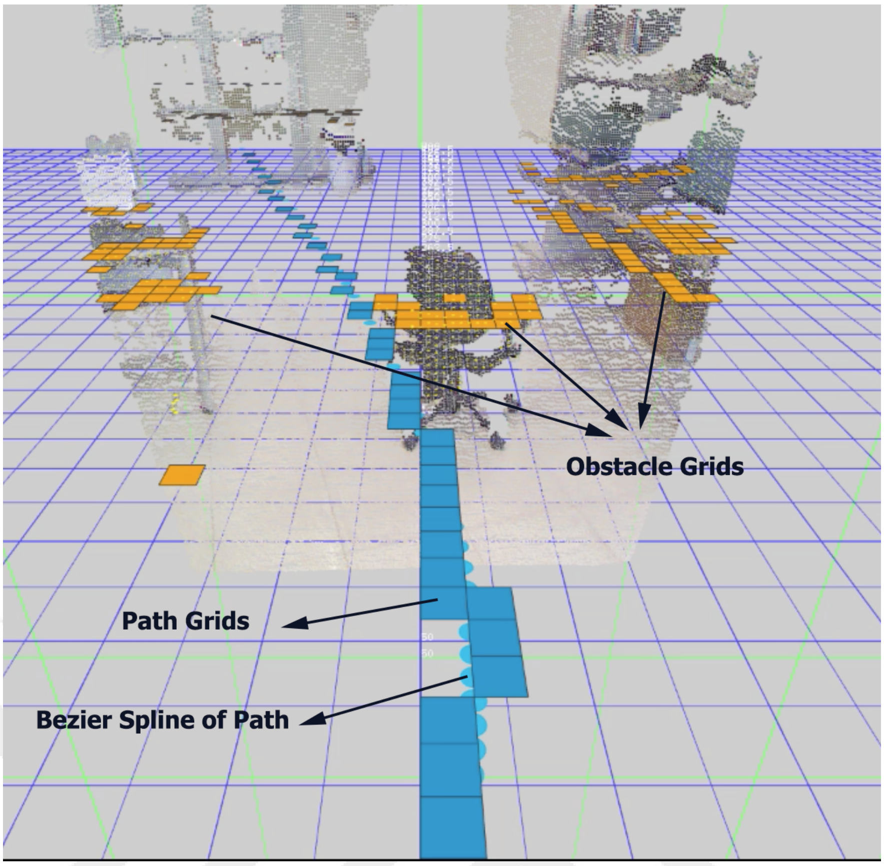
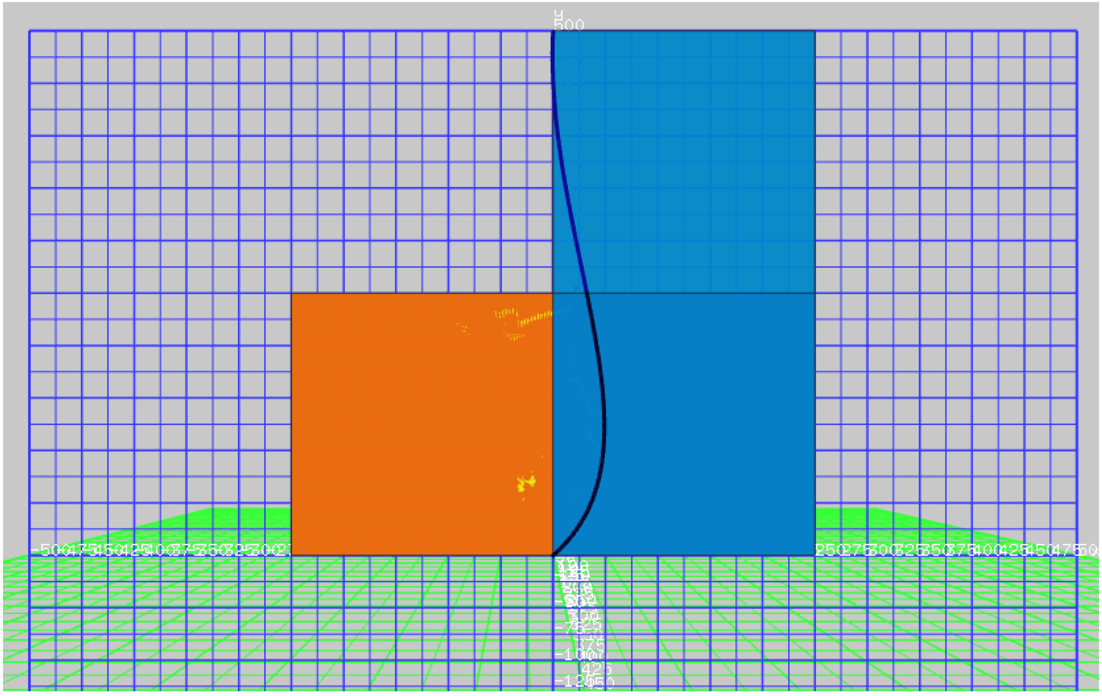


Figure 5.2 : Occupancy and Path Grid allocation in real life example. Point clouds are gathered from Kinect sensor. Occupied obstacle grids are shown in red squares. Path spline and path grids are also shown in the 3D plane. All these calculations are evaluated in real-time.

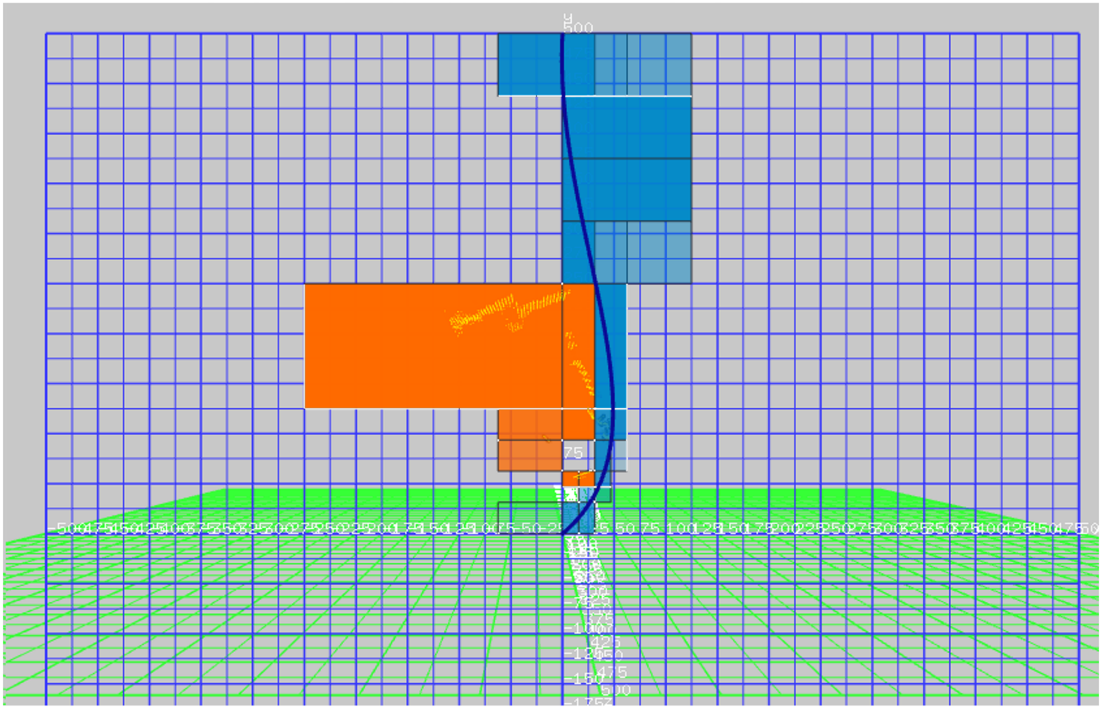
the path spline only in local waypoints can solve this drawback. Next results are given to compare the computational cost and the memory efficacy of the Q-Tree gridding process with the static grid assignment of landmarks and path splines. Simulation results show the distribution of 100 runs both for fixed scanned point count with several grid sizes and fixed grid size with numerous scanned point counts. In Fig. 5.4, for a fixed size of landmarks (arbitrarily chosen as 200), Q-Tree assignment process is 25 times faster than the static assignment, which can be observed with given numerical values in Table 5.1 and Table 5.2.

Table 5.1 : Computation time [s] change with increasing grid size in Q-Trees.

Grid Size	Mean	Median	Best	Worst
100	0.00124	0.00111	0.00107	0.00190
1000	0.00127	0.00116	0.00108	0.00173
10000	0.00122	0.00109	0.00108	0.00294
100000	0.00118	0.00108	0.00107	0.00305



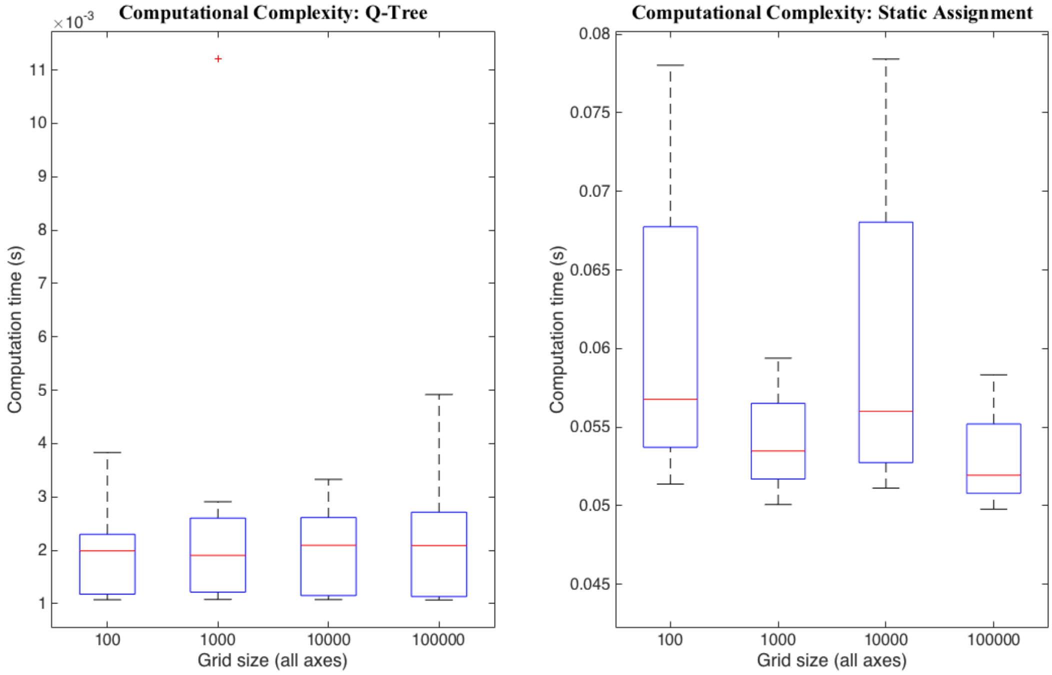
(a)



(b)

Figure 5.3 : Demonstration of the independent sampling rates in real life example. (a) In the beginning, both visible dimensions divide the ROI into two halves with two sibling nodes. (b) After algorithm runs, each visible dimension is separated into different partitions due to the existence of the obstacles and intersected path splines.

Another test is applied to 128×128 fixed-size grids with 100 to 100,000 landmarks shown in Fig. 5.5. Computational cost is better than static assignment up to 60 times



(a) **(b)**

Figure 5.4 : Comparison of the Q-Tree with the static allocation of the landmarks and path in terms of computational complexity. Simulations are a result of 100 runs to show the distributions with arbitrarily chosen 200 landmarks on increasing grid sizes. (a) Q-Tree (b) Static Assignment.

Table 5.2 : Computation time [s] change with increasing grid size in static allocation.

Grid Size	Mean	Median	Best	Worst
100	0.05748	0.05605	0.05138	0.07802
1000	0.05364	0.05332	0.05008	0.05939
10000	0.05766	0.05437	0.05112	0.07842
100000	0.05208	0.05180	0.04978	0.05833

in the average and up to 1000 times in the best, which the numerical results given in Table 5.3 and Table 5.4. Best cases occur when the landmarks are clustered together in certain parts, and the worst cases occur when the landmarks distributed evenly in the ROI.

Table 5.3 : Computation time [s] change with increasing landmark count in Q-Trees.

Landmark Count	Mean	Median	Best	Worst
100	0.00119	0.00119	0.00110	0.00129
1000	0.00134	0.00134	0.00108	0.00160
10000	0.00107	0.00107	0.00106	0.00109
100000	0.00107	0.00107	0.00106	0.00108

On the other hand, comparison of the memory usages for both methods is shown in Fig. 5.6 with the numerical results in Table 5.5 and Table 5.6. In the case of the

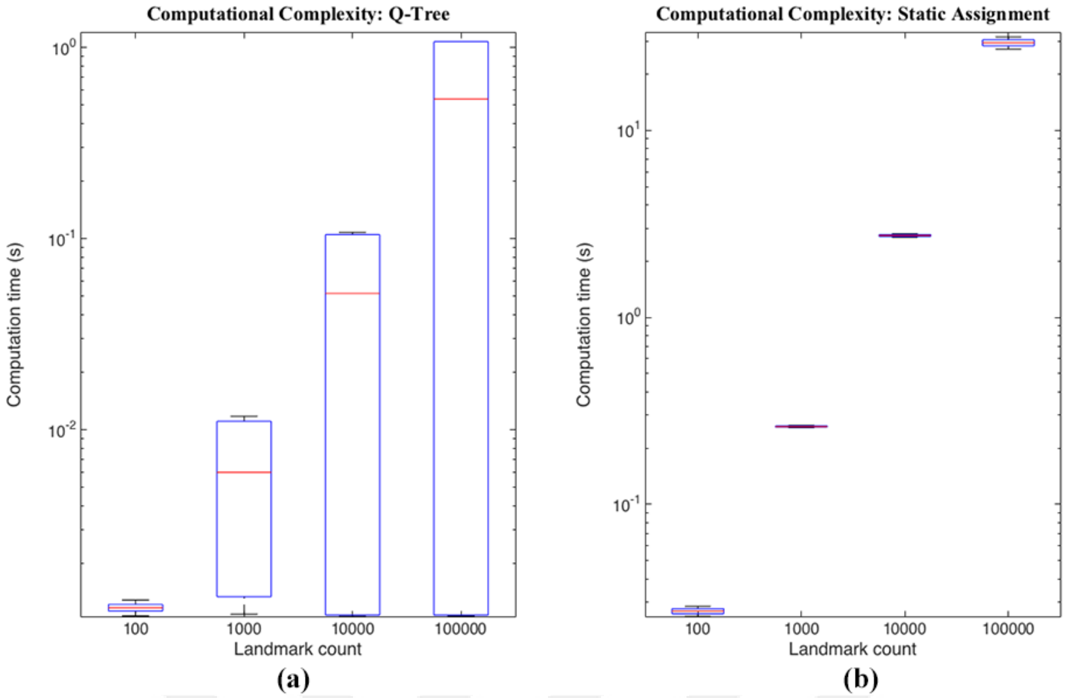
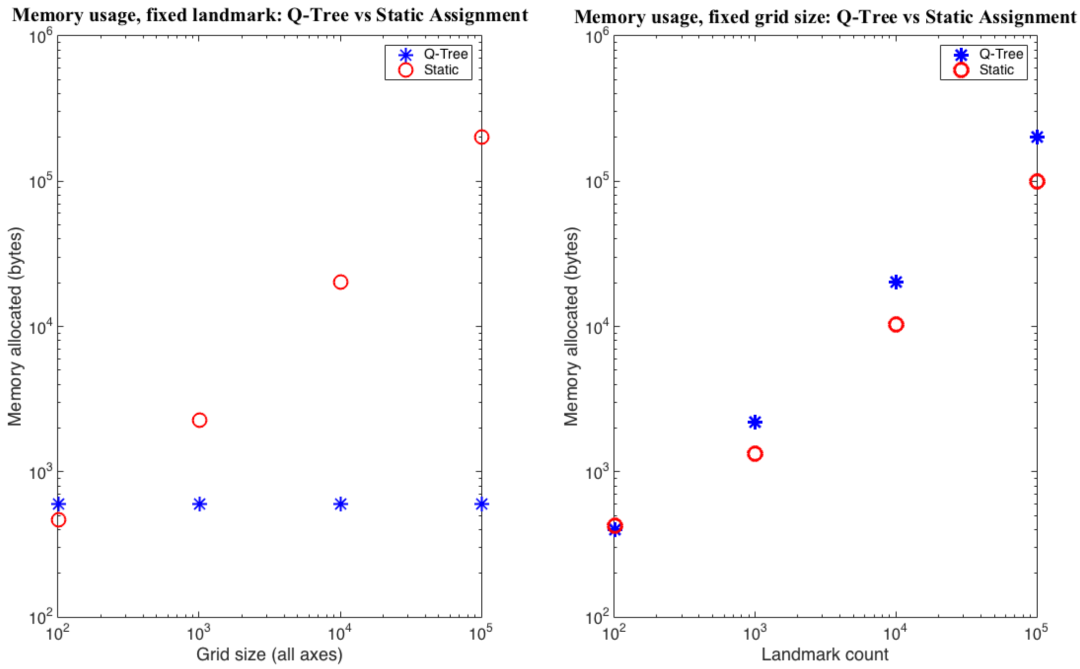


Figure 5.5 : Comparison of the Q-Tree with the static allocation of the landmarks and path concerning computational complexity. Simulations are a result of 100 runs to show the distributions for a chosen 128×128 fixed-size grid with increasing number of landmarks. (a) Q-Tree (b) Static Assignment.

Table 5.4 : Computation time [s] change with increasing landmark count in static allocation.

Landmark Count	Mean	Median	Best	Worst
100	0.02684	0.02684	0.02518	0.02849
1000	0.26074	0.26074	0.25687	0.26462
10000	2.73850	2.73850	2.68009	2.79691
100000	29.38996	29.38996	27.20141	31.57851

invariable size of landmarks (200), memory usage of the Q-Tree method is almost flat due to the exponential coverage of the binary-tree tessellation, merely with a small underperformance in small grid sizes (100×100). For the chosen fixed-size grids (128×128), a linear growth occurs with the increasing landmark counts. A slight over memory usage of the proposed solution occurs on the occasion of the additional cost of the parent-child associations with regard to the static case. Real-time experiments are accomplished in a small size laboratory environment through a Turtlebot robot that runs ROS [63] middleware, and the onboard Asus Xtion Pro sensor is used to get 3D depth sensor data. Experiment results and the test setup with test scene view are shown in Fig. 5.7. Three different scenarios are conducted to show the feasibility and the features of the proposed algorithm. The first scenario shows the behavior of the method when there is a little or no interaction between the path and the obstacles. The second scenario presents the response when spline paths pass nearby to the obstacles. Moreover, the last scenario exposes the situation that path goes directly to the obstacle neighborhood and force the system into forming the smallest allowed grid sizes.



(a) (b)

Figure 5.6 : Simulation results for the memory usages of the Q-Tree compared with the Static allocation. (a) Arbitrarily chosen 200 landmarks with increasing grid sizes. (b) Using 128×128 fixed-size grids with increasing number of landmarks.

Table 5.5 : Memory usage[bytes] comparison with increasing grid size.

Method	Grid Size			
	100	1000	10000	100000
Q-Tree	600	600	600	600
Static	466	2266	20266	200266

Table 5.6 : Memory usage[bytes] comparison with increasing landmark count.

Method	Landmark Count			
	100	1000	10000	100000
Q-Tree	400	2200	20200	200200
Static	422	1322	10322	100322

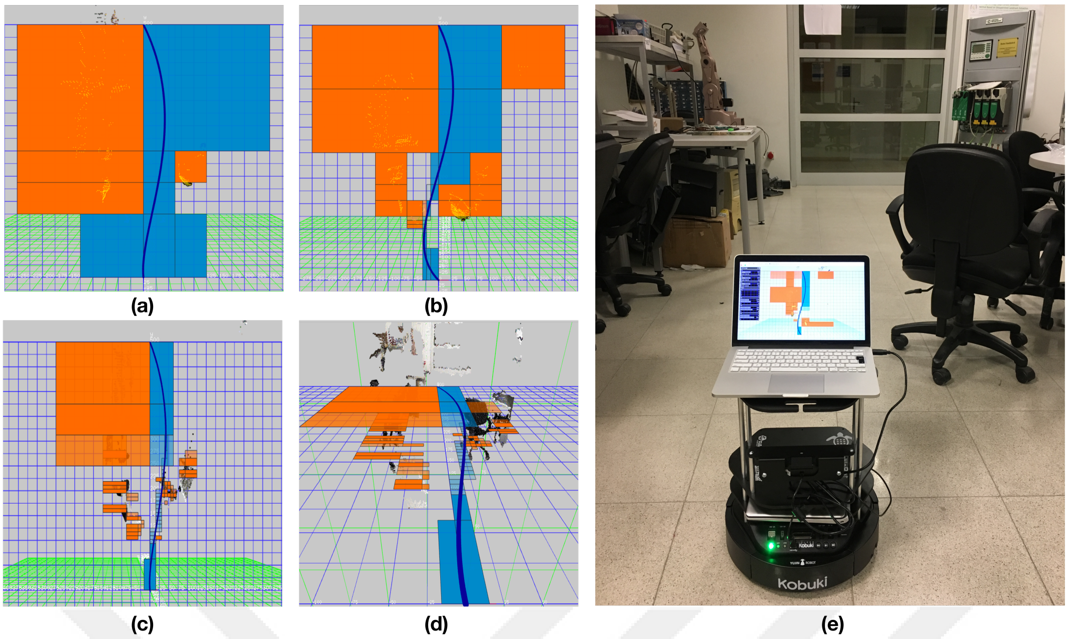


Figure 5.7 : Real-time experiment results in a small scale lab environment with a Turtlebot testbed. Red squares represent the occluded areas from the 3D depth sensor readings, which are shown as green dots. Blue squares represent the allocated path grids for Bezier curve based paths. (a) Scenario 1: A candidate path spline is not intersected with the obstacles. Thus, both path and obstacle grids are divided into relatively coarse grids. (b) Scenario 2: The candidate path spline passes through obstacle grids slightly, which is resulted in denser obstacle grid partitions in a local area. (c) Scenario 3: The candidate path spline slightly touches the obstacle. Therefore, both path and obstacle grids are divided into the densest grid size allowed. Creating a memory outcome for alternative paths is possible as the effect of another candidate spline (from Scenario 2) causes denser grids on the left side. (d) Orthographic view for Scenario 3. (e) Robot testbed and experiment environment.

5.3 Comparison of the LQR and qLQR System Outputs within the Framework

First simulation results are aimed to show the superiority of the quantized LQR to the traditional LQR approach. For this purpose, linearized driftless model in Eq. (A.2) is used in Appendix A with initial conditions $\theta = 0, v = 1$ and $\omega = 0$. A and B matrices are used with the iterative linearization of the nonlinear system $f(x,u)$ on system and input states as,

$$\mathbf{A} = \frac{\delta f(x,u)}{\delta x}\Big|_{(x_k,u_k)}, \quad \mathbf{B} = \frac{\delta f(x,u)}{\delta u}\Big|_{(x_k,u_k)} \tag{5.1}$$

Quadratic cost function parameters are chosen as, $Q = diag(0.1, 0.1, 0.01)$ and $R = diag(0.5, 0.2)$. Calculation of the traditional and quantized LQR control gains and closed-loop system matrices is given as a pseudo-code in Algorithm 6. By using this control gains, states and system outputs is given for traditional LQR controller and for quantized LQR in Algorithm 7. Here, n is denoted the number of the states, m is number of the control inputs and l is the number of the state outputs. Remaining notation is compatible with the derivations in Chapter 4.

Algorithm 6 Calculating the control gains for LQR and qLQR controllers

---

Input: N, A, B, x_0, Q, R, S_{final}
Result: Generated control gains and closed-loop system matrices for the vehicle model
Initialization: #Allocate for N+1 steps
 $S_k \leftarrow zeros(n,n,N+1)$;#Riccati States
 $R_k \leftarrow zeros(m,m,N+1)$;#Quantized Riccati States
 $V_k \leftarrow zeros(n,n,N+1)$;#Lyapunov States
 $K_k \leftarrow zeros(N,n)$;
 $K_{lqr} \leftarrow zeros(N,n)$;
 $Ac_k \leftarrow zeros(n,n,N)$;# closed-loop system matrix ($Ac_k = A_k - B_k * K_k$)
 $Acq_k \leftarrow zeros(n,n,N)$;# closed-loop quantized system matrix ($Ac_k = A_k - B_k * K_{lqr}$)
 $S_k(:,:,N+1) \leftarrow S_{final}$; # final state of Riccati

foreach *i in range[0, 1, N]* **do**
  #calculate the controller feedback gain
   $ind \leftarrow N - k + 1$;
   # inverse usage changes the results
   $K_k(ind,:) \leftarrow inv(R + B' * S_k(:,:,ind+1) * B) * B' * S_k(:,:,ind+1) * A$;
   $S_k(:,:,ind) \leftarrow Q + A' * S_k(:,:,ind+1) * (A - B * K_k(ind,:))$;
   $S_{current} \leftarrow S_k(:,:,ind)$;
   $temp1 \leftarrow A' * S_{current} * A$;
   $temp2 \leftarrow A' * S_{current} * B * B' * S_{current} * A$;
   $temp3 \leftarrow B' * S_{current} * B + 1$;
   $R_k(:,:,ind) \leftarrow temp1 - temp2/temp3$;
   $K_{lqr}(ind,:) \leftarrow inv(B' * R_k(:,:,ind) * B) * B' * R_k(:,:,ind) * A$;
   $Ac_k(:,:,ind) \leftarrow A - B * K_k(ind,:)$;#c/l system matrix
   $Acq_k(:,:,ind) \leftarrow A - B * K_{lqr}(ind,:)$;#c/l quantized system matrix

end

Algorithm 7 Calculation of the traditional and quantized LQR states and system outputs for the vehicle

Input: N, A, B, C, D, x_0

Result: Generated control inputs and outputs for the vehicle

Initialization: #Allocate for N+1 steps

$x_k \leftarrow zeros(n, N+1)$,
$u_k \leftarrow zeros(m, N+1)$,
$y_k \leftarrow zeros(l, N+1)$,
$xq_k \leftarrow zeros(n, N+1)$,
$uq_k \leftarrow zeros(m, N+1)$, # median of the control signal
$uqb_k \leftarrow zeros(2 \times m, N+1)$;# control signal with upper and lower boundaries
$yq_k \leftarrow zeros(l, N+1)$,
#temporary eigenvalue placeholders to check stability
$eigval_k \leftarrow zeros(n, N+1)$
#initial state assigned from previous vehicle state
$x_k(:,1) \leftarrow x_0$
$xq_k(:,1) \leftarrow x_0$
Traditional LQR calculation

foreach *i in range[0, 1, N]* **do**

- $KK \leftarrow K_k(i,:)$;
- $u_k(i) \leftarrow (-KK) * x_k(:,i)$;
- $x_k(:,i+1) \leftarrow A * x_k(:,i) + B * u_k(i)$;
- $y_k(i) \leftarrow C * x_k(:,i)$;
- $eigval_k(:,i) \leftarrow eig(A - B * KK)$; #pole(state_space(A,B,C,D))
- #check if the eigenvalues are in the inner circle
- $eigvals \leftarrow abs(eigval_k(:,i))$;
- **if** *eigvals(1)>1* || *eigvals(2)>1* **then**
  - printf('Eigenvalues are out of the circle!!')
  - # take actions for this situation (e.g. safe stop)
  - exit(0)
- **end**

end

Quantized LQR calculation

foreach *i in range[0, 1, N]* **do**

- $KK \leftarrow K_{lqr}(i,:)$;
- $uq_k(i) \leftarrow (-KK) * xq_k(:,i)$;
- $bound\_limits \leftarrow sqrt(inv(B' * S_k(:,:,i) * B) * xq_k(:,i)' * Q * xq_k(:,i))$;
- $uqb_k(:,i) \leftarrow [uq_k(i) + bound\_limits; uq_k(i) - bound\_limits]$;
- $xq_k(:,i+1) \leftarrow A * xq_k(:,i) + B * uq_k(i)$;
- $yq_k(i) \leftarrow C * xq_k(:,i)$;
- # Control Lyapunov function
- $V_k(:,:,i) \leftarrow xq_k(:,i)' * S_k(:,:,i) * xq_k(:,i)$;
- #check if the clf is a decreasing function
- **if** $i > 0$ **then**
  - $\delta V \leftarrow V_k(:,:,i) - V_k(:,:,i-1)$
  - **if** $\delta V > 0$ **then**
    - printf('Control Lyapunov Function is not decreasing!!')
    - # take actions for this situation (e.g. slowing down)
    - warning()
  - **end**
- **end**

end

In Fig. 5.8, generated control inputs in traditional LQR and qLQR are compared for the same system. While the control inputs nearly have the same values, qLQR control signal has an additional feature, which is the upper and lower boundary stability thresholds from Eq. (4.58). System outputs for the controller inputs within

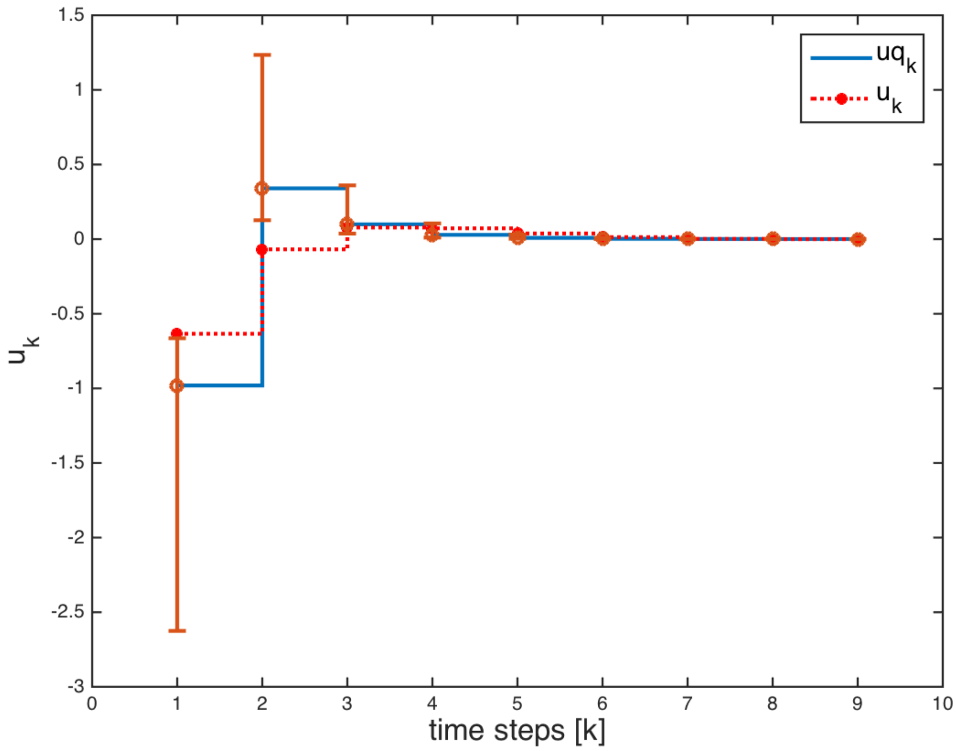


Figure 5.8 : Comparison between the LQR and qLQR control inputs. Control signal of the qLQR has additional upper and lower boundaries.

Fig. 5.8 is showed in Fig. 5.9. One can see that the system controlled with qLQR controller is stabilized faster than the traditional one. Final results are the given for the implementation of the qLQR to the given path in the Fig. 5.3. Linearized system states are specified as the x position, y position and the heading angle as $x_k = [x_i, y_i, \phi_i]$. All states are gradually converged to the reference points. Gradual amplitude changes in the system states for a small route are given in Fig. 5.10.

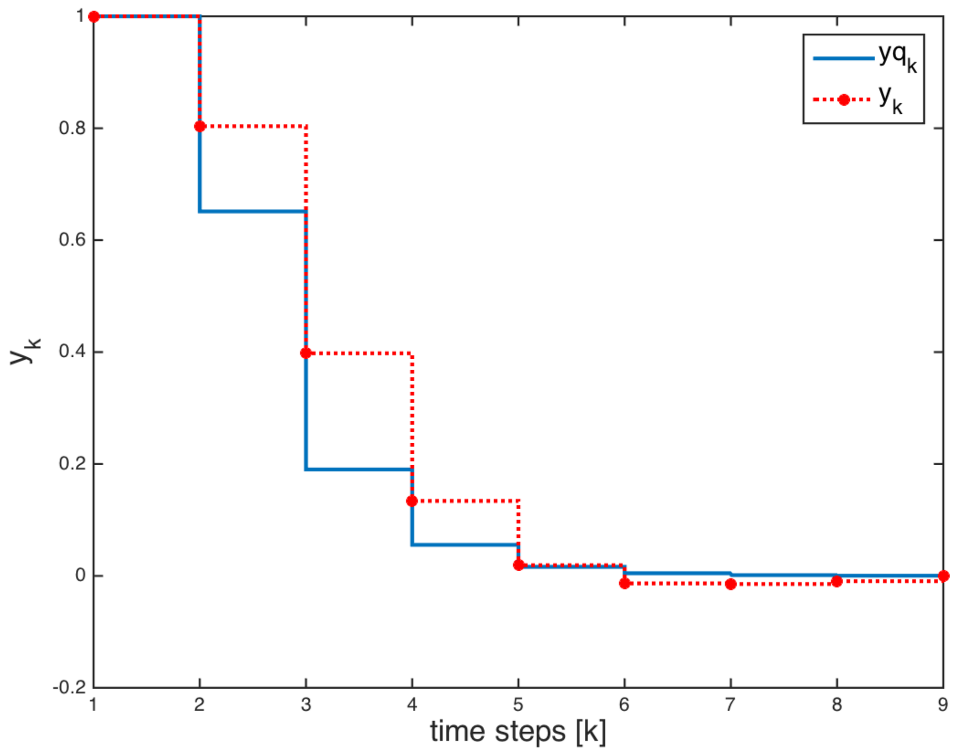


Figure 5.9 : Comparison between the LQR and qLQR system outputs. System controlled with qLQR is stabilized faster than the traditional one.

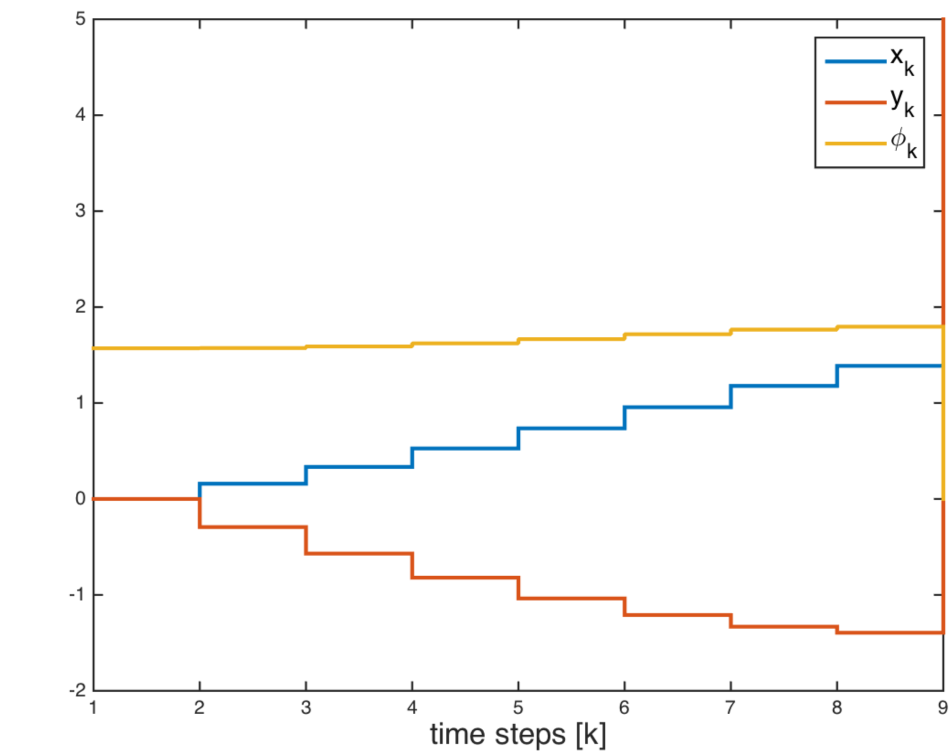


Figure 5.10 : System state changes with qLQR controller in vehicle control system.

5.4 Discussion

In this chapter, both simulation and real-time results are presented to show and compare the effectiveness of the proposed method.

In Section 5.2, simulations and real-time experiments are performed for the proposed method on the occupancy grids map problem. The results are given both in simulation environment and real-time experiments. Computational complexity and the memory usage indicate better results by far regards to the static assignment, which is preferred for a relative measurement concerning the similar methods.

In Section 5.3, the effectiveness of qLQR compared to the traditional LQR system and qLQR usage in the framework is shown both with simulation and real-time results.

This structure has novelty in the sense of expressing the path splines with grids, which are formed with the occurrence of the obstacles. The spatial feedback to continuous, somewhat ideal vehicle path splines defines the maneuverability boundaries. These boundaries are associated both with the environmental conditions and the kinematic and dynamic constraints of the vehicle. First, quantized allocation is applied to the obstacles and paths in the grid map. Then, path allocation is presented both for quadratic and cubic Bezier splines for an ideal case. Vehicle constraints are introduced after the idealistic approach in the first place. Bresenham's line algorithm is used for fast real-time allocation for practical purposes. An extension similar to anti-aliasing added to Bresenham's line algorithm to compensate the uncertainty both in the vehicle states and the sensor readings. Control inputs are generated in the motion planner, which are applied to the qLQR controller in output quantizer. Output quantizer provides stability in the sense of Lyapunov within the specified operating points.

6. CONCLUSION

In this thesis, a novel navigation framework is proposed to fulfill the disconnection among the path generation, environmental modeling, and the control system. This framework has two prominent contributions to the robotics literature. One of them is the input quantizer, and the other one is the output quantizer. This thesis consists of four main chapters that examine these contributions.

The first part of the study includes the definition of the quantization and quantized feedback control theory. This part discussed how it is implemented as a spatial sampler in the input quantizer and a stable controller synthesis in the output quantizer.

The second part of the study is devoted to sensor quantization and kinematic planning, which includes a new approach for assigning both the path and the obstacles into occupancy grids. In this approach, 3D Euclidean space is represented with independent graph tree nodes implemented with very efficient computations in comparison to the alternative methods. In order to achieve this efficiency, binary tree structures implemented with a type of post-order sequentialization that assigns a point to the graph. This assignment terminates after k steps that equal to the corresponding length of the graph tree in the worst case. The final time boundary ensures a worst case estimation, which is very valuable for real-time applications. Additionally, by introducing computer graphics methods into the problem, both for memory usage and computationally efficient grid allocations become possible including the allocation of the uncertainty ellipsoids.

In the third part of the study, navigation framework as a whole introduced after the explanation of the occupancy grid structure. This part focused on the output quantizer, where a quantized LQR mechanism is applied to robot control problem. Asymptotic stability proofs are given in order to show qLQR's supremacy over the traditional LQR approach. By using qLQR, it is revealed that lower and higher thresholds of the control signal are obtained which yields to control signal range that holds asymptotic stability for that operating point. This safety check is significant when the certification of an algorithm is matter in the case of approval for automated driving licenses.

The last part shows both practical experiments and the simulations carried out for a mobile robotic system to demonstrate the effectiveness of the framework.

Further studies will be achieved by expanding this framework with the Markov equivalence of the methods used, which is examined in Appendix C. Another aspect will include modifying other control methodologies rather than LQR system, especially the modern control approaches like Model Predictive Control as in [64–66]. Practical demonstrations may include the autonomous vessels, underwater and aerial vehicles in addition to mobile robots to show the effectiveness of the proposed navigation framework in a different type of robots and environments. The context is also compatible with multi-robot applications where there is a need for merging multiple local maps both in centralized and decentralized cases.

As a summary, a new robot navigation framework is presented to link the occupancy grid mechanism of the robot navigation problem with the robot control systems in an efficient way while guaranteeing the stability criteria in operation.

REFERENCES

[1] **Liberzon, D.** (2003). Hybrid feedback stabilization of systems with quantized signals, *Automatica*, *39*(9), 1543–1554.

[2] **Murphy, O.J. and Selkow, S.M.** (1986). The efficiency of using k-d trees for finding nearest neighbors in discrete space, *Information Processing Letters*, *23*(44), 215–218.

[3] **Greenspan, M. and Yurick, M.** (2003). Approximate kd tree search for efficient ICP, *3-D Digital Imaging and Modeling*.

[4] **Nuchter, A.**, **Lingemann, K. and Hertzberg, J.** (2007). Cached k-d tree search for ICP algorithms, *Sixth International Conference on 3-D Digital Imaging and Modeling*, Osnabruck, Germany, IEEE, pp.419–426.

[5] **Silpa-Anan, C. and Hartley, R.** (2008). Optimised KD-trees for fast image descriptor matching, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Canberra, Australia, IEEE, pp.1–8.

[6] **Schauer, J.**, **Bedkowski, J.**, **Majek, K. and Nuchter, A.** (2016). Performance comparison between state-of-the-art point-cloud based collision detection approaches on the CPU and GPU, *IFAC-PapersOnLine*, *49*(30), 54–59.

[7] **Hornung, A.**, **Wurm, K.M.**, **Bennewitz, M.**, **Stachniss, C. and Burgard, W.** (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees, *Autonomous Robots*, *34*(33), 189–206.

[8] **Yue, Y.**, **Wang, D.**, **Senarathne, P.G.C.N. and Moratuwage, D.** (2016). A hybrid probabilistic and point set registration approach for fusion of 3D occupancy grid maps, *IEEE International Conference on Systems, Man, and Cybernetics*, Nanyang Technological University, Singapore City, Singapore, IEEE, pp.1975–1980.

[9] **Newcombe, R.A.**, **Izadi, S.**, **Hilliges, O.**, **Molyneaux, D.**, **Kim, D.**, **Davison, A.J.**, **Kohi, P.**, **Shotton, J.**, **Hodges, S. and Fitzgibbon, A.** (2011). KinectFusion: Real-time dense surface mapping and tracking, *10th IEEE International Symposium on Mixed and Augmented Reality*, pp.127–136.

[10] **Andreas, N.**, **Kai, L.**, **Joachim, H. and Hartmut, S.** (2007). 6D SLAM—3D mapping outdoor environments, *Journal of Field Robotics*, *24*(8-9), 699–722.

[11] **Zhu, C.**, **Ding, R.**, **Lin, M. and Wu, Y.** (2015). A 3D Frontier-Based Exploration Tool for MAVs, *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, pp.348–352.

[12] **Prieto, R.A.**, **Troncoso, J.M.C.**, **Sánchez, J.R.Á. and Santosjuanes, I.N.** (2013). Reactive Navigation and Online SLAM in Autonomous Frontier-Based Exploration, *5th International Work-Conference on the Interplay Between Natural and Artificial Computation, (IWINAC)*, pp.45–55.

[13] **Chen, D.Z.**, **Szczerba, R.J. and Uhran, J.J.** (1995). Planning conditional shortest paths through an unknown environment: A framed-quadtree approach, *Intelligent Robots*.

[14] **Cocaud, C. and Jnifene, A.** (2011). Environment Mapping Using Probabilistic Quadtree for the Guidance and Control of Autonomous Mobile Robots, *Control and Intelligent Systems (CIS) 39(4)*, *39*(44).

[15] **Kumpel, J.** (1999). Quadtree decomposition of configuration space for robot motion planning, *European Control Conference (ECC)*, 277–282.

[16] **Vörös, J.** (2000). Mobile robot path planning among weighted regions using quadtree representations, *Lecture Notes in Computer Science*, *1798*, 239–249.

[17] **Pochet, A.**, **Celes, W.**, **Lopes, H. and Gattass, M.** (2016). A new quadtree-based approach for automatic quadrilateral mesh generation, *Engineering with Computers*, *33*(2), 275–292.

[18] **LaValle, S.M. and Kuffner, J.J.** (2001). Rapidly-exploring random trees: Progress and prospects, *Algorithmic and Computational Robotics: New Directions*, Wellesley, MA, A K Peters/CRC Press, pp.293–308.

[19] **Kuffner, J.J. and LaValle SM, S.M.** (2000). RRT-connect: An efficient approach to single-query path planning, *2000 ICRA. IEEE International Conference on Robotics and Automation*, *2*, 995–1001.

[20] **Mukai, N. and Ishii, N.** (2009). R-Tree Based Path Representation for Vehicle Routing Problem, *21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, pp.758–761.

[21] **Karaman, S.**, **Walter, M.R.**, **Perez, A.**, **Frazzoli, E. and Teller, S.** (2011). Anytime Motion Planning using the RRT*, *2011 IEEE International Conference on Robotics and Automation(ICRA)*, 1478–1483.

[22] **Otte, M. and Frazzoli, E.** (2015). RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning, *The International Journal of Robotics Research*, *35*(77), 797–822.

[23] **Keenan, J. and Lewis, J.** (1976). Estimation with quantized measurements, *IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, IEEE, pp.1284–1291.

[24] **Curry, R.**, **Mirchandani, P. and Price, C.** (1975). State Estimation with Coarsely Quantized, High-Data-Rate Measurements, *IEEE Transactions on Aerospace and Electronic Systems*, *AES-11*(4), 613–621.

[25] **Edwards, C.** (1997). Sliding mode control using only output information, *IEE Colloquium on Robust Control: Theory, Software and Applications*, IEE, pp.10–10.

[26] **Utkin, V.I.**, (1992). Sliding Modes in Control of Electric Motors, Sliding Modes in Control and Optimization, Springer, Berlin, Heidelberg, Berlin, Heidelberg, pp.250–264.

[27] **Ashrafiuon, H.**, **Nersesov, S.**, **Mahini, F. and Clayton, G.** (2015). Full state sliding mode trajectory tracking control for general planar vessel models, *American Control Conference (ACC*, IEEE, pp.5158–5163.

[28] **Sanchez, A.**, **Parra-Vega, V.**, **Tang, C.**, **Oliva-Palomo, F. and Izaguirre-Espinosa, C.** (2012). Continuous reactive-based position-attitude control of quadrotors, *American Control Conference (ACC)*, IEEE, pp.4643–4648.

[29] **Lee, H. and Utkin, V.I.** (2007). Chattering suppression methods in sliding mode control systems, *Annual Reviews in Control*, *31*(2), 179–188.

[30] **Khatib, O.**, (1986). The Potential Field Approach And Operational Space Formulation In Robot Control, Adaptive and Learning Systems, Springer, Boston, MA, Boston, MA, pp.367–377.

[31] **Mora, M.C. and Tornero, J.** (2008). Path planning and trajectory generation using multi-rate predictive Artificial Potential Fields, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp.2990–2995.

[32] **Baroudi, U.**, **Sallam, G.**, **Al-Shaboti, M. and Younis, M.** (2015). GPS-free robots deployment technique for rescue operation based on landmark's criticality, *International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, pp.367–372.

[33] **Feiten, W.**, **Bauer, R. and Lawitzky, G.** (1994). Robust obstacle avoidance in unknown and cramped environments, *IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, pp.2412–2417.

[34] **Zhou, D. and Schwager, M.** (2014). Vector field following for quadrotors using differential flatness, *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.6567–6572.

[35] **Brock, O. and Khatib, O.** (1999). High-speed navigation using the global dynamic window approach, *International Conference on Robotics and Automation (ICRA)*, IEEE, pp.341–346.

[36] **Fox, D.**, **Burgard, W. and Thrun, S.** (1997). The dynamic window approach to collision avoidance, *IEEE Robotics & Automation Magazine*, *4*(1), 23–33.

[37] **Pereira, G.A.S.**, **Campos, M.F.M. and Kumar, V.** (2004). Decentralized Algorithms for Multi-Robot Manipulation via Caging, *The International Journal of Robotics Research*, *23*(7-8), 783–795.

[38] **Pesch, H.J. and Bulirsch, R.** (1994). The maximum principle, Bellman's equation, and Carathéodory's work, *Journal of Optimization Theory and Applications*, *80*(2), 199–225.

[39] **Bernardo, M.**, **Budd, C.**, **Champneys, A.R. and Kowalczyk, P.**, (2008). Piecewise-smooth dynamical systems: theory and applications.

[40] **Coron, J.M. and Rosier, L.** (1994). A Relation Between Continuous Time-Varying and Discontinuous Feedback Stabilization, *Journal of Mathematical Systems, Estimation, and Control*, *4*, 67–84.

[41] **Bacciotti, A. and Ceragioli, F.** (2006). Nonpathological Lyapunov functions and discontinuous Carathéodory systems, *Automatica*, *42*(3), 453–458.

[42] **Filippov, A.F.**, (1988). Existence and General Properties of Solutions of Discontinuous Systems, Differential Equations with Discontinuous Righthand Sides, Springer, Dordrecht, pp.48–122.

[43] **Blatt, J.M. and Gray, J.D.** (1977). An elementary derivation of Pontrayagin's maximum principle of optimal control theory, *The Journal of the Australian Mathematical Society*, *20*(02), 142–156.

[44] **Cortes, J.** (2008). Discontinuous dynamical systems, *IEEE Control Systems Magazine*, *28*(3), 36–73.

[45] **Liberzon, D.** (2003). *Switching in Systems and Control*, Springer.

[46] **Delchamps, D.F.** (1990). Stabilizing a linear system with quantized state feedback, *IEEE Transactions on Automatic Control*, *35*(8), 916–924.

[47] **Brockett, R. and Liberzon, D.** (2000). Quantized feedback stabilization of linear systems, *IEEE Transactions on Robotics*, *45*(7), 1279–1289.

[48] **Elia, N. and Mitter, S.K.** (2001). Stabilization of linear systems with limited information, *IEEE Transactions on Automatic Control*, *46*(9), 1384–1400.

[49] **Kalman, R.E.** (1956). Nonlinear Aspects of Sampled-data Control Systems, *Nonlinear Circuit Analysis*, Polytechnic Institue of Brooklyn, pp.273–314.

[50] **Sencan, O. and Temeltas, H.** (2018). A quantized approach for occupancy grids for autonomous vehicles: Q-Trees, *Advanced Robotics*, *32*(11), 1–15.

[51] **Mehdi, S.B.**, **Choe, R.**, **Cichella, V. and Hovakimyan, N.** (2015). Collision Avoidance through Path Replanning using Bézier Curves, *AIAA Guidance, Navigation, and Control Conference*.

[52] **Tucker, A.**, **Gonzalez, T.**, **Topi, H. and Diaz-Herrera, J.** (2014). *Computing Handbook, Third Edition*, Taylor & Francis.

[53] **Joy, K.I.**, (1999), Bresenham's Algorithm, `http://www.idav.ucdavis.edu/education/GraphicsNotes/Bresenhams-Algorithm.pdf`, [Online 20-September-2017; Retrieved from University of California, Dept. of Computer Science Lecture Notes].

[54] **Gill, G.W.** (1994). N-step incremental straight-line algorithms, *Computer Graphics and Applications, IEEE*, *14*(33), 66–72.

[55] **Pepy, R.**, **Lambert, A. and Mounier, H.** (2006). Path Planning using a Dynamic Vehicle Model, *2nd International Conference on Information Communication Technologies*, volume 1, pp.781–786.

[56] **Wu, X.** (1991). An efficient antialiasing technique, *SIGGRAPH '91: Proceedings of the 18th annual conference on computer graphics and interactive techniques*, *25*(44), 143–152.

[57] **Sencan, O. and Temeltas, H.** (2018). A New Motion Planning Framework based on the Quantized LQR Method for Autonomous Robots, *International Journal of Advanced Computer Science and Applications*, *9*(1).

[58] **Yang, S.W.**, **Wang, C.C. and Chang, C.H.** (2010). RANSAC matching: Simultaneous registration and segmentation, *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.1905–1912.

[59] **Raguram, R.**, **Frahm, J.M. and Pollefeys, M.**, (2008). A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus, Computer Vision ECCV, Springer, Berlin, Heidelberg, Berlin, Heidelberg, pp.500–513.

[60] **Kuffner, J.J. and LaValle, S.M.** (2000). RRT-connect: An efficient approach to single-query path planning, *IEEE International Conference on Robotics and Automation (ICRA)*, *2*, 995–1001.

[61] **Otte, M. and Frazzoli, E.** (2015). RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning, *The International Journal of Robotics Research*, *35*(7), 797–822.

[62] **Sontag, E.D.** (1983). A Lyapunov-Like Characterization of Asymptotic Controllability, *SIAM Journal on Control and Optimization*, *21*(3), 462–471.

[63] **Quigley, M.**, **Conley, K.**, **Gerkey, B.P.**, **Faust, J.**, **Foote, T.**, **Leibs, J.**, **Wheeler, R. and Ng, A.Y.** (2009). ROS: an open-source Robot Operating System, *ICRA Workshop on Open Source Software*, pp.1–6.

[64] **Kwon, W.H. and Han, S.** (2005). *Receding Horizon Control: Model Predictive Control for State Models*, Springer-Verlag London.

[65] **Dunbar, W.B. and Murray, R.M.** (2002). Model predictive control of coordinated multi-vehicle formations, *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pp.4631–4636 vol.4.

[66] **Li, H.**, **Shi, Y. and Yan, W.** (2016). Distributed receding horizon control of constrained nonlinear vehicle formations with guaranteed gamma-gain stability, *"Automatica"*, *68*, 148 – 154.

[67] **Nair, S. and Kobilarov, M.** (2011). Collision avoidance norms in trajectory planning, *2011 American Control Conference*, 4667–4672.

APPENDICES

APPENDIX A : Vehicle Dynamics and Maneuverability

A mathematical model of Ackermann type steering model for an Autonomous Guided Vehicle (AGV) which is shown in Fig. A.1 is given as a nonlinear function $f()$, subject to state and control variables (or constraints)

$$\dot{x} = f(x, u) \tag{A.1}$$

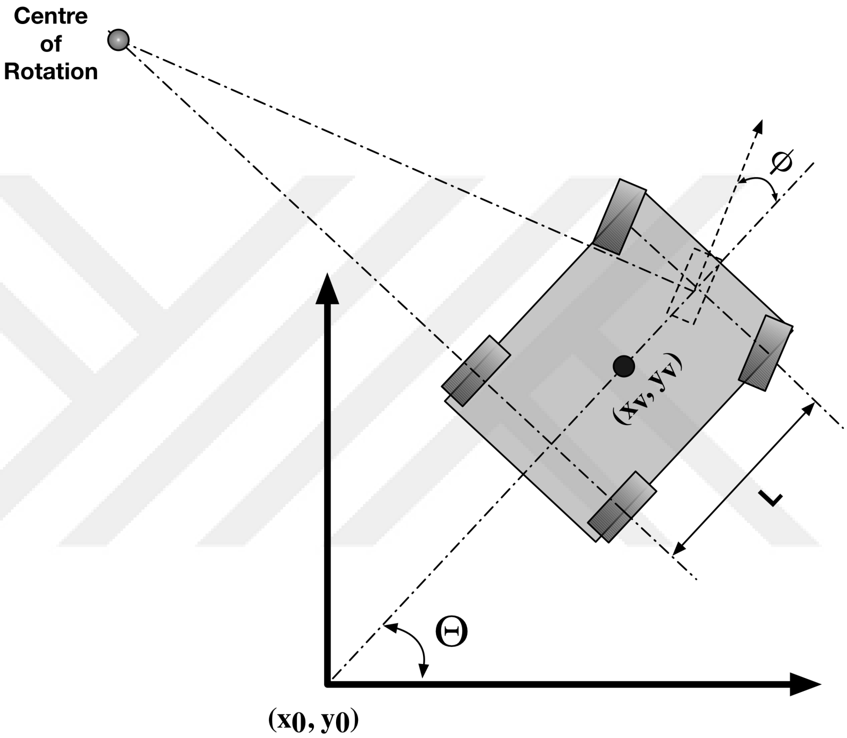


Figure A.1 : Ackermann type steering model.

The vehicle state variables are x, y, the locations in x-y coordinates, θ orientation with linear velocity v and steering angle φ. Control variables of the vehicle are linear acceleration a and angular velocity ω. In vector notation:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \varphi \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} a \\ \omega \end{bmatrix}$$

Eventually, the open form of the nonlinear system dynamics given in Eq. (A.1) can be derived as follows:

$$\dot{\mathbf{x}} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ y \\ \theta \\ v \\ \varphi \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{\theta}\tan(\varphi) \\ a \\ \omega \end{bmatrix} \tag{A.2}$$

Despite the fact that the vehicle model is nonlinear , usually a simple linearized version of this model is highlighted due to the necessity for using the well-known linear control theory. Besides, this notation is also useful to show the quantization effects on the system. Driftless linear model of the vehicle, which is derived from the nonlinear model in Eq. (A.2) is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{A.3}$$

Most of the mobile robotics systems have nonholonomic nature. *Nonholonomic systems* are the systems that have limited configuration space subject to the constraints with position (x) and velocity ($\dot{x}$). The columns of the matrix in Eq. (A.3) are the spannig vectors(b_i, $i = 1,\dots,m$) of the corresponding configuration space (is a subspace of $\mathbb{R}^2$):

$$\mathbf{b}_1 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}, \qquad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.4}$$

B is an rank deficient matrix. i.e.,

$$rank(\mathbf{B}) = m < n \tag{A.5}$$

where, m is 2 and n is 3 in this example. It is evident that the first two rows of **B** cannot generate the third row in a linear combination (Fig. A.2). Thus, one can come up with the result of the system in Eq. (A.3) cannot be asymptotically stabilizable by a continuous feedback law. To determine whether a system is holonomic (i.e., completely integrable) or nonholonomic (i.e., not integrable), *Lie brackets* is the tool one can appeal. Lie bracket for vector fields is defined as:

$$[f_1, f_2](x) = \frac{\partial f_2(x)}{\partial x} f_1(x) - \frac{\partial f_1(x)}{\partial x} f_2(x) \tag{A.6}$$

If the Lie bracket of two vector fields is zero, that means linear combination of system vector fields commute. In other words, for a drift less linear system ($\dot{\mathbf{x}} = \mathbf{Bu}$), in the case of Lie bracket of all combinations of spanning vectors are zero, the system is completely integrable (i.e., holonomic).
The Lie brackets are also an indicator of the controllability of the system, if it satisfies the following condition:

$$rank(\Big[\mathbf{b_1}, \mathbf{b_2}, [\mathbf{b_1}, \mathbf{b_2}]\Big]) = n \tag{A.7}$$

Thus, the system is said to be satisfy the the *Lie Algebra Rank Condition (LARC)* and every state can be freely steered to other state (Chow's theorem [45]). The LARC ensures that the motion of the vehicle is not restricted by a submanifold $\mathbb{R}^n$ (Frobenius

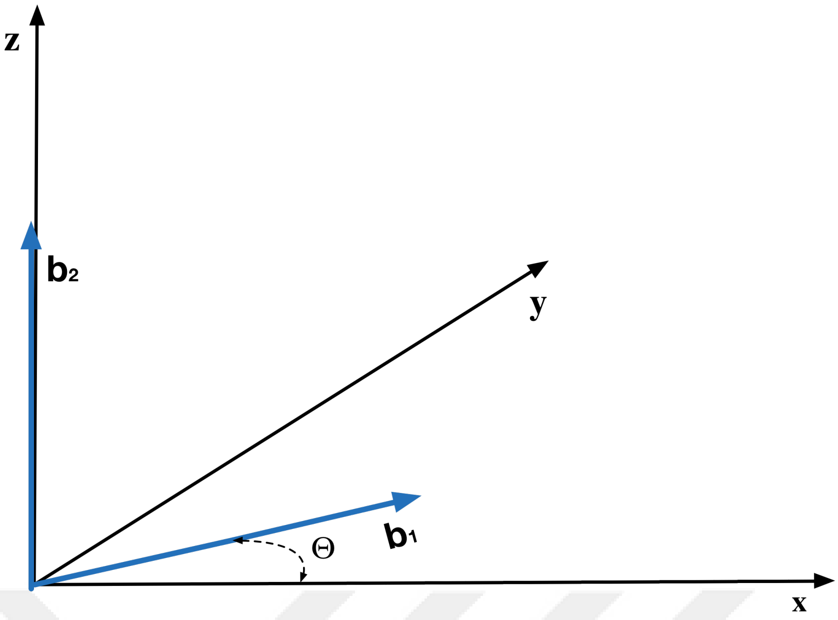


Figure A.2 : Configuration space with spanning vectors.

thorem [45]). For the given nonlinear model in Eq. (A.3) the Lie bracket of two spanning vectors can be found as:

$$\begin{aligned}[b_1, b_2](x) &= \frac{\partial b_2(x)}{\partial x} b_1(x) - \frac{\partial b_1(x)}{\partial x} b_2(x) \\ &= 0. \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin(\theta) \\ 0 & 0 & \cos(\theta) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \sin(\theta) \\ -\cos(\theta) \\ 0 \end{bmatrix} \end{aligned} \tag{A.8}$$

The above result of Lie bracket for the given system cannot be equal to zero. Thus, the system is nonholonomic and the vectors $\mathbf{b_1}, \mathbf{b_2}$ are linearly independent from their Lie brackets. Therefore, the LARC condition is satisfied for this system (completely steerable from one state to another as shown in Fig. A.3).

Checking the steerability of a system is straightforward with testing the LARC condition of the system. However, even with absence of obstacles (external constraints), asymptotic stability is not possible for all configurations (vehicle cannot move sideways) and the steerability condition is satisfied only with *switched control*.

Let's explain this problem with a common situation of parking strategy in parallel parking. By using normalized control signals $v \in [-1, 1]$, $\omega \in [-1, 1]$ moving sideways for parallel parking, there is no a possible configuration of control signals (a specific $[v, \omega]$ pair) that can move the vehicle towards to a goal position. Hence, the solution lies behind a switching strategy between control signal pairs. In this example a simple solution is available in a sequence of motion strategies: go back with steering full right (i.e., $u_1 = [-1, 1]^\top$). In a state (e.g., when rear right wheel comes near an obstacle) go back with steering full left (i.e., $u_2 = [-1, -1]^\top$). Then, go forward with steering full right (i.e., $u_3 = [1, 1]^\top$) and soon go forward with steering full left (i.e., $u_4 = [1, -1]^\top$) to go vertical to the straight motion of the vehicle. This sequence is shown in Fig. A.4.

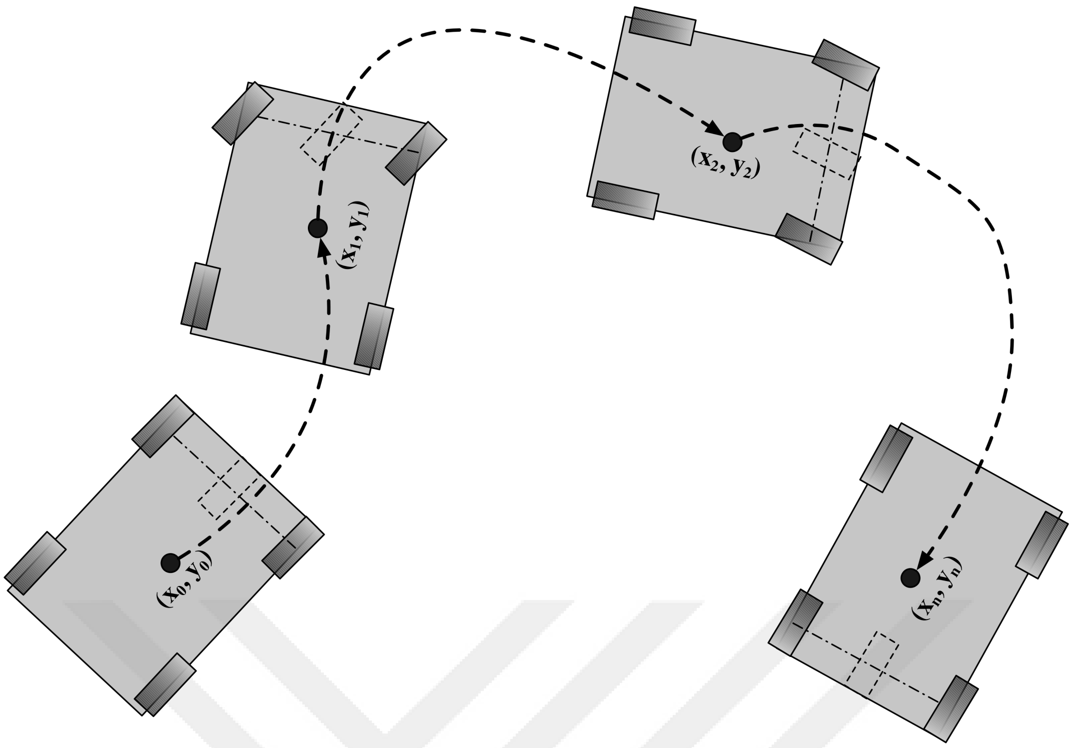


Figure A.3 : Illustration of unrestricted steerability.

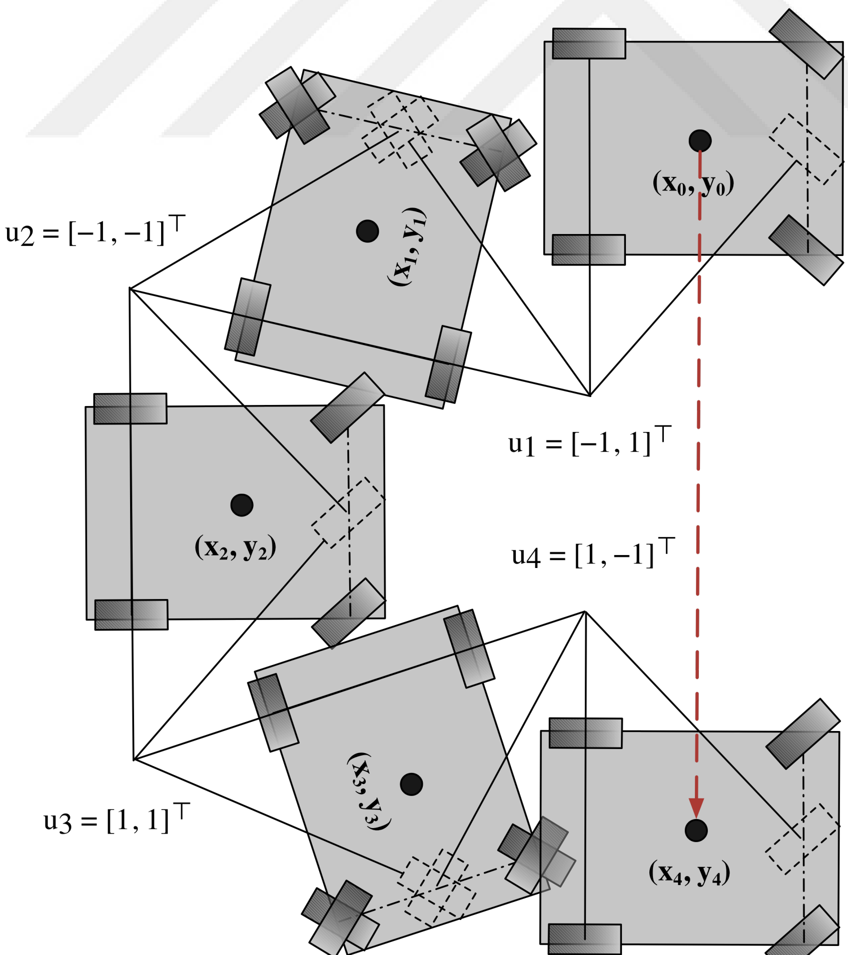


Figure A.4 : Parallel parking strategy: Moving sideways in Lie bracket.

This sideway motion is indicated by the Lie bracket of the system control matrix. Lie bracket of the spanning vectors is calculated in Eq. (A.8), which is perpendicular to straight movement vector of the vehicle. This fact is illustrated in Fig. A.5.

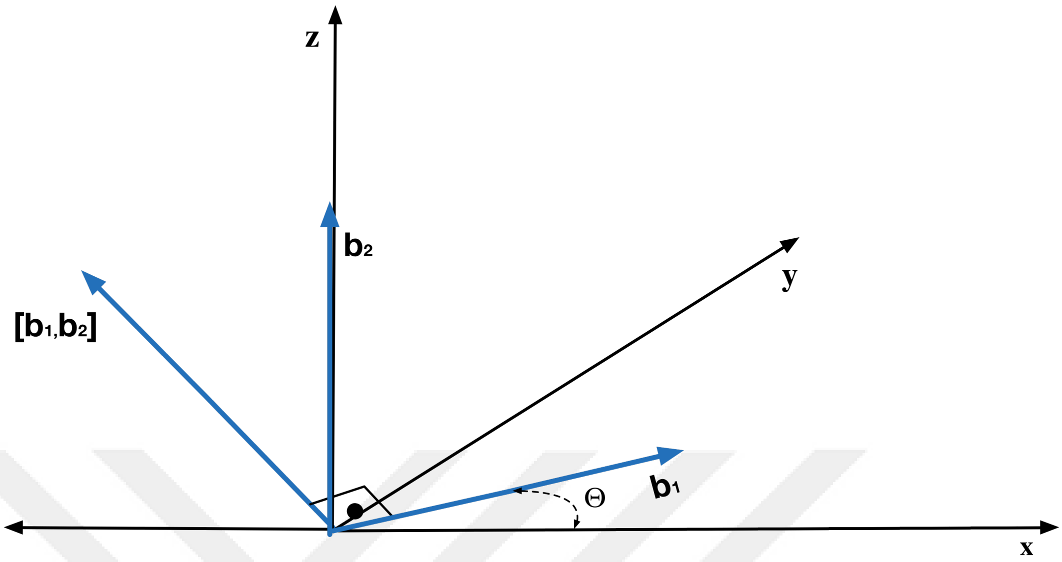


Figure A.5 : Lie brackets of the spanning vectors are perpendicular to straight motion of the vehicle.

APPENDIX B : Obstacle Avoidance

Path planning algorithms should consider the obstacles around the vehicle that has effects on the maneuverability. For the mathematical representation of the obstacles, one way is using *p-norm* to take the advantage of differentiable algebraic function structure. Vector norm of two dimensional coordinates is defined as:

$$\|x,y\|_p := \left(\|x\|^p + \|y\|^p\right)^{\frac{1}{p}} \tag{B.1}$$

This norm definition is used for creating generic shapes like diamonds (L_1 norm), circles (L_2 norm), ellipses (modified L_2 norm) and squares (L_∞ norm). Using this norm notation, an obstacle can be defined as a function:

$$h(x,y) = \left\| \left(\frac{x-x_c}{a}\right), \left(\frac{y-y_c}{b}\right) \right\|_p - c \tag{B.2}$$

where the x_c, y_c is the centre of the obstacle and c is the distance from origin (or measurement point). a, b parameters are indicating ellipse radiuses for $x-y$ coordinates respectively. Any point that is outside of this region returns a positive value from the call of $h()$ function. So, all the x_i, y_i pair that have negative values should be treated as an obstacle in this representation. In mathematically speaking:

$$\forall (x,y) \in X_{obs} \,\Big|\, h(x,y) \leqslant 0 \tag{B.3}$$

If the norm degree p in Eq. (B.2) is taken too large, the value of $h()$ is become too large to handle. To overcome this problem, logarithmic equivalent of this comparison should be used in practical applications:

$$h(x,y) = \log\left(\left\| \left(\frac{x-x_c}{a}\right), \left(\frac{y-y_c}{b}\right) \right\|_p \right) - p\log c \tag{B.4}$$

An extension to this approach is taken place by introducing a surrounding circle radius of vehicle r_s to the Eq. (B.2) and checking the collision distance d_{col} within a compact version of the form as follows:

$$d_{col}(x_v, y_v, x_c, y_c) = \left\| \left(\frac{x_v-x_c}{a+r_s}\right), \left(\frac{y_v-y_c}{b+r_s}\right) \right\|_p - c \tag{B.5}$$

From this perspective, one can derive a test function to avoid a potential collision of two vehicles by checking the L_2 distances with the following inequality test:

$$\left(\frac{x_{v_1}-x_{v_2}}{r_{s_1}+r_{s_2}}\right)^2 + \left(\frac{y_{v_1}-y_{v_2}}{r_{s_1}+r_{s_2}}\right)^2 - 1 > 0 \tag{B.6}$$

There is also a different approach to remark distances to obstacles not only with their spatial properties, but also with their variations in time (i.e., frequency characteristics) using H^k *Sobolev Norms* [67]. The classical H^k norm of a function defined as a sum of k derivatives of x-squared:

$$\|x\|_{H^k} = \sqrt{\int \sum_{i=0}^{k} \left| \frac{\partial^i x}{\partial t^i} \right|^2 \mathrm{d}t} \tag{B.7}$$

One can see that L_2 norm is a special case of H_k norm with $k = 0$. Using Parseval's theorem for time-frequency energy equilibrium, the frequency component of Sobolev norm can be shown:

$$\|x\|_{H^k} = \sqrt{\int_{-\infty}^{\infty} (1+|\xi|^2)^k |X(\xi)|^2 \, \mathrm{d}\xi} \tag{B.8}$$

here $X(\xi)$ is the fourier transform of x(t):

$$X(\xi) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i t \xi} \, \mathrm{d}t \tag{B.9}$$

The world, where the vehicle is navigated is denoted by **W** which is a subspace of $\mathbb{R}^2$ or $\mathbb{R}^3$ depending on the configuration space denoted by **C** is the representation of the vehicle frame including position, orientation and joint angles. Let the configuration of the vehicle as $p(t) \in C$ and the navigation task is taking the vehicle from p_0 to p_T in a time interval $t \in [0, T]$.
Vehicle with its occupying region $A(p)$ should move in a free configuration space C_{free} while avoiding the obstacles $O_i \in W$, $\{i = 1, \ldots, n_{obstacle}\}$. In a mathematical way:

$$C_{free} = \{p \,|A(p) \bigcap O_i \neq \emptyset\} \tag{B.10}$$

Collision boundary is in the borders of this free configuration space where,

$$A(p) \bigcap \partial C_{free} \neq \emptyset \tag{B.11}$$

is satisfied. This section shows how to handle obstacles in this framework has an answer, at least founded on theoretical concepts.

APPENDIX C : Relationship between the Markov Chains and Occupancy Grids

In this section, the Markov property is applied to the state transitions between grids that bounds the occupancy grids to the Markov property and the vehicle control architecture. This section includes a brief introduction to the Markov property and then its equivalence in occupancy grids and zoom-in/out property is given afterwards.

Markov Chain property

Let $[x_1, \ldots, x_k, \ldots x_n]$ are sequences of vehicle states that have been generated by a stochastic process. This stochastic process is called *Markovian process/chain of order s* if the conditional probability of the current state depends only the previous s number of states which is shown in Eq. (C.1). This property is also called as *memorylessness* property of the system.

$$p\big(x_k \mid x_{k-1}, \ldots, x_0\big) = p\big(x_k \mid x_{k-1}, \ldots, x_{k-s}\big) \tag{C.1}$$

By convention, the order s is usually taken as 1 or 2. If $s = 1$, in this situation the process is called as *first order Markov process*. So the Eq. (C.1) becomes;

$$p\big(x_k \mid x_{k-1}, \ldots, x_0\big) = p\big(x_k \mid x_{k-1}\big) \tag{C.2}$$

Likewise, If $s = 2$, the process is called as *second order Markov process* and shown as follows;

$$p\big(x_k \mid x_{k-1}, \ldots, x_0\big) = p\big(x_k \mid x_{k-1}, x_{k-2}\big) \tag{C.3}$$

For the sake of simplicity, lets continue with assuming the system is coherent with first order Markov process. Here, the $p\big(x_k \mid x_{k-1}\big)$ is called *transition probability* with the initialization of the likelihood of first state x_0 as $p(x_0)$. This is called *initial probability distribution*. The *joint probability density function* of this sequence can be calculated as follows in Eq. (C.4);

$$p\big(x_0, x_1, \ldots, x_n\big) = p\big(x_n \mid x_{n-1}\big) \cdots p\big(x_{n-k} \mid x_{n-k-1}\big) \cdots p\big(x_1 \mid x_0\big) p\big(x_0\big) \tag{C.4}$$

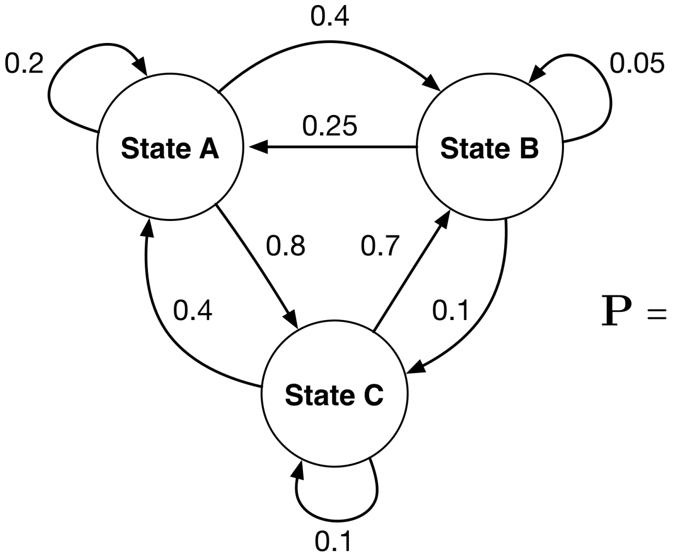
Usually a *state transition matrix* $\mathbf{P}$ is used to define the state transitions between the states. The $\mathbf{P}$ matrices should have the property of all their row sums have to be equal to 1, which is a specification of the *stochastic row matrix*. An example is shown in Fig. C.1 to demonstrate the behaviour of the transition. For any given time, the probability of being in the state x_{k+1} while the current state is x_k is;

$$p\big(x_{k+1}\big) = p\big(x_k\big)\mathbf{P} \tag{C.5}$$

Assume that the initial state is state C (i.e. $x_0 = [0, 0, 1]$) and the probability of the next step can be found as;

$$\begin{aligned} p\big(x_1\big) &= [0,0,1] \cdot \mathbf{P} \\ &= [0.8, 0.1, 0.1] \end{aligned} \tag{C.6}$$

Now the state in one step is most likely (80%) in state A or less likely(10%) in state B or stays in state C (10%) due to the first order Markov assumption. One more step can



$$\mathbf{P} = \begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.25 & 0.05 & 0.7 \\ 0.8 & 0.0.1 & 0.1 \end{bmatrix}$$

Figure C.1 : Transition of the states and corresponding transition matrix P.

be iterated similar to the previous calculation as follows;

$$\begin{aligned} p(x_2) &= p(x_1)\mathbf{P} \\ &= [p(x_0)\mathbf{P}]\cdot\mathbf{P} \\ &= p(x_0)\mathbf{P}^2 \\ &= [0.265, 0.335, 0.4] \end{aligned} \tag{C.7}$$

Using transition matrix, it is possible to calculate the limit probabilities with the assumption of transition matrix is time-invariant using eigenvalue decomposition on **P** matrix;

$$\begin{aligned} \lim_{n\to\infty} \mathbf{P}^n &= \lim_{n\to\infty} \mathbf{W}^{-1}\mathbf{D}\,\mathbf{W} \\ &= \mathbf{W}^{-1} \lim_{n\to\infty} \mathbf{D}\,\mathbf{W} \end{aligned} \tag{C.8}$$

where the **W** is the full matrix, whose rows are left eigenvectors and **D** is a diagonal matrix whose diagonal terms are eigenvalues of the **P**. For given example one can be found the limit of $\mathbf{P}^n$ using Eq. (C.8) as;

$$\lim_{n\to\infty} \mathbf{P}^n = \begin{bmatrix} 0.4255 & 0.2168 & 0.3577 \\ 0.4255 & 0.2168 & 0.3577 \\ 0.4255 & 0.2168 & 0.3577 \end{bmatrix}$$

This limit shows, according to the central limit theorem the state is found in A with the 42.55% probability, 21.68% in state B and 35.77% in state C.

Relationship between the Markov process and the quantization

The process of replacing the sequence of points by a sequence of corresponding sets is called *quantization* [49]. Using the Theorem 5 of the [49], Any first order Markov process can be synthesized in the structure of the first order nonlinear difference equations which in this case is the vehicle dynamics. In this study, the Markov property is applied to the state transitions between occupancy grids, which are also dynamically changeable with respect to the zoom-in/out property of the occupancy graphs.

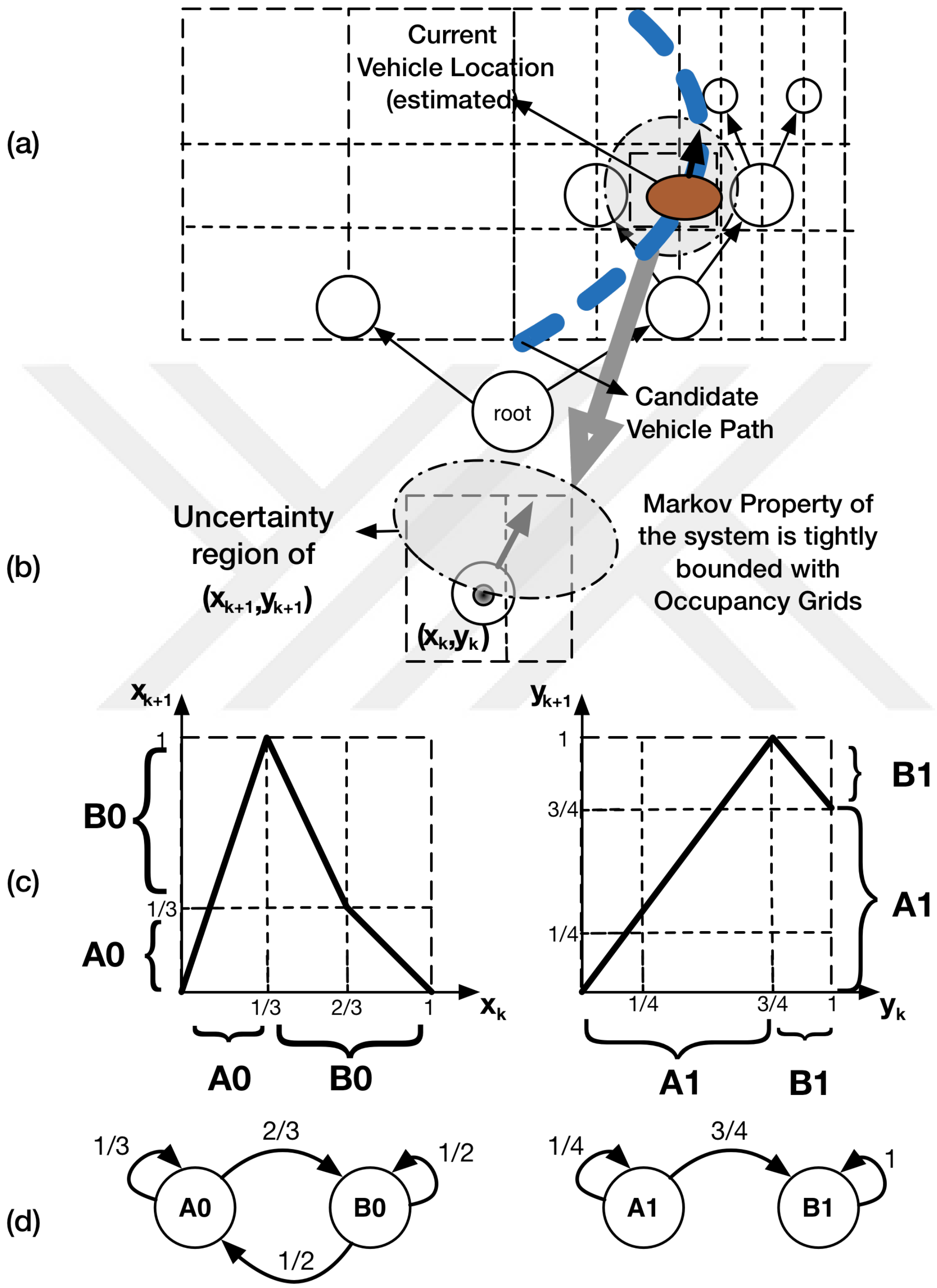


Figure C.2 : Detailed schematic of the relationship between the Markov process and Quantization.

The detailed process between occupancy grids and its equivalence in Markov chains is visualized in Fig. C.2. In Fig. C.2(a), ROI of the vehicle is divided into grid regions due to depth of the corresponding tree branch. The branching occurs with the zoom-in/out property which depends on the vehicle dynamics, obstacle existence in or near candidate path and so on. Achievements on this part of the thesis is set forward to next six months. Here the path with blue dotted line is a Bezier curve and cross through several grid partitions. Red ellipse is the estimated position of the vehicle in a given time where the ellipse symbolizes the uncertainty of the state.

Fig. C.2(b) shows the magnified view of the uncertainty region to remark the current position uncertainty ellipse and its projection to the grids to be occupied. The gray ellipse is symbolized the next predicted states of the vehicle with a growing size of uncertainty from its nature. This prediction step also corresponds to a *possible* state change, which means switching the occupied grids in spatial sense and control mode change (level and step size transition in steering and acceleration, sensor type and the sampling rate of the measurement units etc.)

The switching phenomenon is shown in Fig. C.2(c) and Fig. C.2(d), where $A0, B0$ and $A1, B1$ are candidate states in position update for x and y dimensions respectively. Here in the simplified case the vehicle is in the $A0$ state with the probability of $\frac{1}{3}$ or in the $B0$ state with the probability of $\frac{2}{3}$ along x-dimension. A nonlinear correlation function $F(x_k)$ bounds the current state to the next state, which also sets the probability transition matrix. In this example, if the vehicle is in state $A0$ it stays in the present state with the probability of $\frac{1}{3}$ or switches to the state $B0$ with the probability of $\frac{2}{3}$. On the other hand, if the vehicle is in state $B0$ it stays in this state with the probability of $\frac{1}{2}$ or switches to the state $A0$ with the probability of $\frac{1}{2}$. Likewise, similar interpretation can also be made for y-dimension. The interesting part in here that if the state is in the state $B1$ in initial situation, it definitely stays in this state for the next step. This can be interpreted as the vehicle is going always forward and the relation with the previous states is severed sharply of course harmoniously with the Markovian chain assumptions.

APPENDIX D : Proof for Asymptotic Stability Holds in Zoom-in/out Cases

Proof of Asymptotic stability holds in zoom-in/out cases given in [45]

Theorem D.1. *Assume that the following inequality holds.*

$$\sqrt{\frac{\lambda_{min}(\mathbf{P})}{\lambda_{max}(\mathbf{P})}} > 2\Delta \frac{\|\mathbf{PB}\|\|\mathbf{K}\|}{\lambda_{min}(\mathbf{Q})}$$

Then there exists a quantized feedback control solution that makes the system Eq. (4.17) globally asymptotically stable.

Proof. Zoom-out case: Set $u = 0$, Let $\mu(0) = 1$ And increase μ fast enough to dominate the rate of growth of $\|e^{At}\|$. Then, there will be a time $t_0 \geq 0$ such that,

$$\|x(t_0)\| \leq \sqrt{\frac{\lambda_{min}(\mathbf{P})}{\lambda_{max}(\mathbf{P})}} \frac{R\mu(t_0)}{\|\mathbf{K}\|}$$

which implies that $x(t_0)$ belongs to ellipsoid $R_1(\mu(t_0))$.

Zoom-in case: Let's pick a $\varepsilon > 0$ for $t \geq t_0$ that holds,

$$\sqrt{\lambda_{min}(\mathbf{P})}R\mu > \sqrt{\lambda_{max}(\mathbf{P})}Q_u\|K\|\Delta(1+\varepsilon)$$

where,

$$Q_u := \frac{2\|\mathbf{PB}\|}{\lambda_{min}(\mathbf{Q})}$$

Let $\mu(t) = \mu(_0)$ for $t \in [t_0, t_0 + T)$ where T is given by the Eq. (4.26). Then, one can show that $x(t_0 + T)$ belongs to the ellipsoid $R_2(\mu(t_0))$.

For $t \in [t_0 + T, t_0 + 2T)$, let $\mu(t) = \Omega\mu(t_0)$ where,

$$\Omega := \frac{\sqrt{\lambda_{max}(\mathbf{P})}Q_u\|K\|\Delta(1+\varepsilon)}{\sqrt{\lambda_{min}(\mathbf{P})}R}$$

Here, $\mu(t_0 + T) < \mu(t_0)$ and $R_2(\mu(t_0)) = R_1(\mu(t_0 + T))$. ■

CURRICULUM VITAE

Name Surname: Onur Şencan

Place and Date of Birth: İzmir, 26.12.1985

E-Mail: osencan@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2007, Kocaeli University, Faculty of Engineering, Department of Electrical Engineering
- **B.Sc.:** 2007, Kocaeli University, Faculty of Engineering, Department of Electronic and Communication Engineering
- **M.Sc.:** 2009, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Department of Control Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2008-2016 Research Assistant, İstanbul Technical University at the Robotics Laboratory of Control Engineering.
- 2012-2016 Director of Engineering, Arbon Robot and Control LTD.
- 2016- .. Chief Technology Officer, Vircon Technologies A.S.
- Best Paper Award, Electronic Imaging in 2017.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Sencan O.** and Temeltas H., “A New Motion Planning Framework based on the Quantized LQR Method for Autonomous Robots”, *International Journal of Advanced Computer Science and Applications(IJACSA)*, 9(1), 2018. DOI: http://dx.doi.org/10.14569/IJACSA.2018.090150
- **Sencan O.** and Temeltas H., "A Quantized Approach for Occupancy Grids for Autonomous Vehicles: Q-Trees", *Advanced Robotics*, Volume 32, Issue 11, Pages 1–15 2018. DOI: 10.1080/01691864.2018.1468818
- **Sencan O.** and Temeltas H., "Graphical Model Inference Algorithms for Map Sparsification", *IFAC Proceedings Volumes*, Volume 46, Issue 25, 2013, Pages 205-210, DOI: https://doi.org/10.3182/20130916-2-TR-4042.00021
- **Sencan O.** and Temeltas H., "Nicemlenmiş LQR Yöntemi ile Robot Navigasyonu" in Türkiye Otonom Robotlar Konferansı, (TORK), April 2018.

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Erhan C., Sariyanidi E., **Sencan O.**, Temeltas H., Efficient visual loop closure detection in different times of day, Electronic Imaging, Volume 17, Issue 9, 2017, Pages 5-9
- Erhan C., Sariyanidi E., **Sencan O.**, Temeltas H., "Patterns of approximated localised moments for visual loop closure detection", IET Computer Vision, Volume 11, Issue 3, 2016, Pages 237-245
- Ervan O., **Sencan O.**, Temeltas H. Transversal positioned (TRAP) 2D laser range finder configuration, Pattern Recognition Association of South Africa (PRASA-RobMech), 2016, Pages 1-6
- Erhan C., Sariyanidi E., **Sencan O.**, Temeltas H., "An online visual loop closure detection method for indoor robotic navigation", Proceedings Volume 9406, Intelligent Robots and Computer Vision XXXII: Algorithms and Techniques, 2015; DOI: 10.1117/12.2082532
- Erhan C., Sariyanidi E., **Sencan O.**, Temeltas H., "Loop closure detection using local Zernike moment patterns", Proceedings Volume 8662, Intelligent Robots and Computer Vision XXX: Algorithms and Techniques,2013, DOI: 10.1117/12.2008473
- Sariyanidi E., **Sencan O.**, Temeltas H., "An image-to-image loop-closure detection method based on unsupervised landmark extraction", Intelligent Vehicles Symposium (IV), 2012, Pages 420-425
- **Sencan O.**, Sariyanidi E., Temeltas H.,"Drivable Floor Detection With Monocular Visual Perception", SDU International Journal of Technological Science, Volume 3, Issue 1, 2011, Pages 11-22
- **Sencan O.**, Turanli M., Sariyanidi E., Hakan T., Kurnaz S., Bogosyan S., "A Real-time SLAM Algorithm with Optical Flow based Motion Extraction for Autonomous Robot Navigation", International Unmanned Vehicles Workshop (UVW2010), Volume 1, Issue 1, 2010, Pages 51-57
- **Sencan O.**, Otononom Mobil Robotlarda Optik Akis Tabanli Gorsel Es Zamanli Lokalizasyon Ve Haritalama", Master Thesis, ITU Fen Bilimleri Enstitusu, 2009
- **Sencan O.**, Yildiz A.B., "Asenkron Motorun Simulasyonunun Grafiksel Kullanici Arayuzu Uzerinden Gerçeklestirilmesi, EMO 2007, Pages 1-6