Learning Vision based Mobile Robot Behaviors from Demonstration

Faculty of Electrical Engineering and Information Technology
TU Dortmund University

DISSERTATION

submitted in partial fulfillment
of the requirements for the degree

Doktor Ingenieur
(Doctor of Engineering)

by

M.Sc. Krishna Kumar Narayanan
Dortmund, Germany

Date of submission: 15th February, 2015

First examiner: Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram
Second examiner: Univ.-Prof. Dr.-Ing. Martin Mönnigmann

Date of approval: 2nd November, 2015

Contents

Acknowledgements

This work would not have come to fruition without the guidance and support of my academic peers, family and friends. First of all, I would like to thank Univ.-Prof. Dr.-Ing. Prof.h.c. Dr.h.c. Torsten Bertram for providing me with the opportunity to work in his department and be a part of the RST family. Through his timely advice and guidance at appropriate situations, he has helped me to stay on the right track and allowed me to mature both as a researcher and as a person. I would like to thank apl.Prof. Dr. rer. nat. Frank Hoffmann for his constant support and help through out my tenure at the department. His innumerable ideas and his extensive knowledge has helped me tremendously in formulating this work. For that, i owe my immense gratitude and thanks. I would like to thank all the members of our department who have always been ready to discuss and give suggestions in need. I would like to thank especially Felipe Posada for his inspiring discussions and friendly suggestions. His company has provided me the extra energy, motivation and drive to continue working during the long wee hours at the lab. Special thanks also to Jürgen Limhoff for all the help with the hardware and construction of the mobile robot. Many thanks to my office colleagues and friends, Jörn Mahlzahn and Anh Son Phung for their numerous suggestions and friendly support all through the stay. I would also like to thank the contribution of all the students who through their decision to do master thesis with me have provided numerous insight and understanding for the completion of this work.
My heartfelt thanks to my parents, brother and my partner who have been there in all my moments of good and bad. With out their well wishes and moral support, this research could not have been possible. A big thanks to everyone.

Abstract

Autonomous service robots that are scalable and flexible to learn, accommodate new tasks and thereby assist humans is one of the long term vision of robotics and artificial intelligence. Out of the multitude of skills that we desire from mobile robots, the ability to navigate autonomously is an important task. Behavior based control is an approach to realize this vision where the robot action is divided into independent primitive motion substrates called as behaviors. Manual design and programming of such behaviors require an astute understanding of the environment and its effect on behavioral response. Furthermore, if the perceptual sensor is vision based, the task is more challenging because of the complexity of the visual information. Learning from Demonstration (LfD) is one approach that alleviates this process, where a teacher demonstrates examples of behavioral perceptual states action pairs which are then transferred to a behavioral policy without any explicit programming. Despite the ever progressive research in this area, many design decisions to achieve a successful LfD architecture for learning vision based mobile robot behaviors are yet not completely answered. This thesis addresses these issues and proposes a framework to learn visual robotic indoor behaviors from demonstration examples. Demonstrations are performed by a human teacher to a robot mounted with an omnidirectional camera as its main navigational sensor. Design and analysis of different intuitive demonstration modes to realize an innate, flexible and user-friendly interface are presented. The fidelity of the generated demonstration data from the different teaching modes are evaluated. Situated learning of the robotic behaviors is accomplished by learning scenario specific behaviors. Classifying the environment into three scenarios, behavior fusion within each scenario is achieved by identifying the inherent context of operation. Other behavior modularities such as learning individual behavioral representations are also addressed. The performance and efficiency of the different representations together with their influence on the emerging final behavior are shown. Besides the supervised behavior learning models, an architecture for self-learning vision based robot behaviors within a LfD framework is also proposed. Starting with a limited knowledge, the ability of the framework to update its initial behavioral policy by learning its value function is shown. Successful solutions to challenging problems such as one-shot learning and learning from scratch are accomplished. The goal of the thesis is to present the reader to the potentials of LfD for vision based mobile robot behaviors and propose the necessary steps and design decisions to build such a framework.

Abbreviations

AIC	Akaike Information Criterion
ALVINN	An Autonomous Land Vehicle in a Neural Network
ANN	Artificial Neural Networks
BIC	Bayesian Information Criterion
CCD	Charge-coupled device
DAMN	Distributed Architecture for Mobile Navigation
DE	Context label: Dead end
DTW	Dynamic Time Warping
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
HOG	Histogram of Oriented Gradients
HRI	Human-Robot-Interaction
HSV	Hue-Saturation-Value
IRL	Inverse Reinforcement Learning
L	Context label: Left turn to align in a corridor
LCSS	Longest Common Subsequence
LDA	Linear Discriminant Analysis
LfD	Learning from demonstration
LWR	Locally Weighted Regression
mRMR	minimal-redundancy-maximal-relevance
MSE	Mean Squared Error
NB	Naive Bayes
NMSE	Normalized Mean Squared Error
NRMSE	Normalized Root Mean Squared Error

NUI	Natural User Interface
OL	Context label: Left turn to avoid an obstacle
OR	Context label: Right turn to avoid an obstacle
PCA	Principal Component Analysis
PCR	Principal Component Regression
PMD	Photonic Mixture Device
PNN	Probabilistic Neural Networks
QDA	Quadratic Discriminant Analysis
R	Context label: Right turn to align in a corridor
RANSAC	Random Sample Consensus
RBF	Radial Basis Function
RF	Random Forest
RG	Regression Trees
RGB	Red-Green-Blue
RL	Reinforcement Learning
RMS	Root Mean Squared
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
S	Context label: Carry on straight
SEDS	Stable Estimator of Dynamic Systems
SIFT	Scale Invariant Feature Transforms
SLAM	Simulatenous Localization And Mapping
SNR	signal-to-noise Ratio
SOM	Self-Organizing Map
SSE	Sum of Squared Error
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition

Nomenclature

$A(BB)$	Area of the rectangular bounding box of the shape
A_s	Area of shape
B_c	Bending energy of a curvature
$D\left(x_i, x_q\right)$	Manhattan distance between the query input and neighboring i^{th} input
D_{cs}	Cauchy-Schwarz divergence
$I(x, y)$	Image intensity at pixel locations x and y
$P(r, \theta)$	Similarity P at locations r and θ
$P_{ss'}^{a}$	State transition probability of reaching state s' from state s with an action a
Q^*	Optimal action value function
Q^{π}	Action value function of a policy π
R_t	Long term discounted reward over t time steps
R_{fnf}	Ratio of floor to no floor regions
T_c	Curve temperature
V^*	Optimal state value function
V^{pi}	State value function of a policy π
W_{max}	Maximum width of the free floor region
Δx	Lateral distance to the corridor center
α	Lateral offset angle
α_g	Angular direction of the goal point
β	Orientation angle
β_g	Orientation error at the goal point
$\mathbf{D_b}$	Boundary distances of the shape to the shape centroid
$\mathbf{D_c}$	Mean distance of the boundary points to shape centroid
$\boldsymbol{\kappa_s}$	Curvature of the shape
$\mathbf{Z}_m$	Principal Component number m
γ	Discount factor
γ_1	Skewness
γ_2	Kurtosis
$\hat{yt}_i$	i^{th} predicted target output of the training data
κ	Robot path curvature
$(S_{x,0}, S_{y,0})$	Center of mass of the segmented shape
$\mathcal{F}$	Mapping between the image and its extracted features
$\mathcal{L}$	Log likelihood of the data for the model parameters
x*	Equilibrium attractor state
μ_k	Mean of the k^{th} Gaussian

ω	Rotational velocity
ϕ	Elliptical tilt after an elliptical approximation of the free floor space
ρ_s	Circularity of shape
$\rho_{\max}$	Maximum plausible measurable distance to the goal point
ρ_g	Pixel distance to the goal point
ρ_{max}	Maximum plausible distance to goal point in the image
σ	Standard deviation of the data
σ_d	Standard deviation of the boundary distances to the shape centroid
σ_k	Covariance of the k^{th} Gaussian
σ_{yt}	Standard deviation of the training data output
θ_s	Orientation of the principal axis
θ_c	Critical angle
θ_{min}	Orientation of the closest obstacle
θ_{safe}	Next traversable free floor direction
d_c	Critical distance
d_{min}	Distance to the closest obstacle
d_o	Distance to the nearest obstacle along the current path curvature
e_{hand}	Mean horizontal pixel error of the left and right hand (e_{left}, e_{right})
f_n	False Negatives
f_p	Histogram count of the pixel values
h_s	Perimeter of the shape convex hull
m_{pq}	$(p+q)^{th}$ order image moment
n_k	Number of parameters of the Gaussian Mixture Model
p_s	Perimeter of shape
t_n	False Positives
t_p	True Positives
t_c	Time to collision
ts	Time to collision
v	Translational velocity
x_c	Centroid of image from its moments
xt_i	i^{th} input of the training data
xt_q	Query input
yt_i	i^{th} target output of the training data
$\|\|S\|\|$	Euclidean norm of a shape
$\|\|S_C\|\|$	Centroid size of the shape
$\|\|S_M\|\|$	Mean size of the shape
$\|\|S_{RMS}\|\|$	Root mean square size of the shape
a	Semi major axis
b	Semi minor axis

1
Introduction

> *It is not my aim to surprise or shock you - but the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do things is going to increase rapidly until - in a visible future - the range of problems they can handle will be coextensive with the range to which the human mind has been applied.*
>
> – Herbert Simon, *Heuristic problem solving: the next advance in operations research*

Robot behaviors are the fundamental building block for accomplishing a successful autonomous mobile robot navigation. Behavior based systems takes its inspiration from ecology and biological systems. The reactive behavioral traits found in animals and humans arises from the oldest regions in the brain. Typically called as the reptilian brain, the midbrain section of this stem is concerned with the interpretation of the environmental perception to appropriate reflexive behavioral traits. The modern human brain builds on the instinctive traits of the reptilian brain and moderates with its newer counterparts to generate highly complex behaviors that are based on situation, context and experience. Imbibing similar traits on a machine is a highly challenging process as it needs a profound understanding of the principle behind the workings of the human brain, break them down into rules and transfer it individually to the machine processing unit. Behavior based robotics [Ark98] attempts to build the initial steps towards realizing this biological paradigm by using the software design perspective to modularize the complex behavioral traits to simple behavioral primitives.
Animal behavior in its purest form is reactive and has been the inspiration for behavior based robotics. Reactive control in a robot directly exhibits a very tight coupling between the sensing and action executed in terms of motor action. The stimulus is generated by sensor observation of the environment whose input is interpreted to an appropriate motion action response by the behavior. Conventional mobile robotic research today have had astounding success with autonomous navigation [HBFT03]; [TBF05]; [Alt03]; [MEBFGK10] which work with proximity sensors to sense the environment. Proximity sensors like sonar have a very low resolution and are limited in size whereas laser range sensors provide a better resolution but are expensive and need more power to operate. Computer vision systems on the other hand are getting more cheaper and increasing in performance by the day. Vision is rapidly creating more interest within the robotic community to be considered as the primary navigation sensor either alone or in conjunction with the proximity sensors [DK02]; [BFOO08]. The design of a robust reactive vision based robot behaviors

faces multiple problems in understanding images [NPHB09] such as the lack of depth information to construct 3D scenes, preprocessing large amount of information and thereby requiring high data aggregation, the dynamic nature of indoor environments which vary substantially in geometry, appearance and to lighting conditions and the highly context dependent nature of the different image features such as texture, intensity or optical flow. These problems aggravate the conventional behavior based approach to identify prototypical scenarios and to design context specific albeit general reactive behaviors for them. Multiple works in the area of pure vision based navigation have been done ranging from path following [NFH06] to obstacle avoidance [LBG97]. Despite the relative successes of the works, visual navigation is still trying to establish an acceptable benchmark mainly due to the characteristics and appearance of the environment which irreverently needs a manual parameter tuning from the developer. This is where learning can occur within the framework of behavior based systems. Learning from demonstration (LfD) is a policy development technique with a potential straight forward solution for non-trivial robotic tasks [Arg09]. In LfD, a teacher first demonstrates the policy to the robot and generates a list of perception-action pairs. The robot then generalizes these perception-action pairs to learn a mapping, derive a policy and navigate by itself. It is an indirect or a supervised method of learning behaviors compared to the conventional direct mode of programming. This dissertation addresses these challenges and proposes a framework for learning vision based indoor robotic behaviors from demonstrations. The work focuses on learning the primitive building blocks of robot navigation, where the visual sensorimotor pairs of lower level behaviors are modeled. The behaviors are analyzed and learned in modular fashion so that it can augment and enrich a higher level operation (eg. Simultaneous Localization and Mapping, path planning, exploration) without any interdependency issues. The learned behaviors have no prior information about the environment. The thesis takes inspirations and grazes many interdisciplinary fields such as computer vision, image feature extraction, machine learning and human-robot interaction but always maintaining the spirit of behavior based robotics. Four prototypical behaviors, namely corridor or wall following, obstacle avoidance, door passing and homing or goal point reaching are learned.

1.1. Goal and contribution of the thesis

The principal objective of the thesis is to present the potential of a LfD framework for scalable, flexible and intuitive vision based robotic behaviors. The dissertation tries to answer the following questions:

- How does a human teacher demonstrate navigational skills to a mobile robot in an efficient and intuitive manner?
- How does one transfer these demonstrations from the teacher to the robot working with a different sensor modality?
- What are the different behavioral representations and how do they differ in performance during learning?
- What are the relevant image features that characterize the working principle of different behaviors?

- How to determine scenario, context and behavior specific features?
- Can a learned behavior be improved by introducing new knowledge?

This thesis covers diverse aspects of progressing towards building a behavior learning system. The contributions of the thesis are the following:

Designing intuitive demonstration interfaces
Novel demonstration modes are designed and investigated for flexibility in commanding a mobile robot [NPHB12]. Taking the inspiration from the different kinesthetic modes available to demonstrate a humanoid or a manipulator robots [Cal09]; [SAMB12], this work extends the concept to mobile robots to generate examples that are noise free, flexible and easy to use. Three teleoperation modes for commanding a mobile robot are designed and analyzed. Direct teleoperation with a joystick possess an advantage in that it is extremely fast to generate examples, but exhibits high signal-to-noise ratio with the data [KFD95]; [LD06]. Thus, other alternative modes using graphical and natural user interface are explored. Remote teleoperation is with graphical user interface where the user sits and controls the robot remotely using a steering wheel by looking at the robots camera view projected onto a screen. The user here directly controls from the robots perspective in that there exists no differences in sensor modality. The third mode uses a Natural User Interface (NUI) based framework where motion gestures are used to command the robot from behind. Microsoft Kinect is used to capture naturally occurring teacher gestures and encode them to motion commands. With the Kinect based demonstration mode, the teacher assumes turning a hypothetical steering wheel whose angles are interpreted as turn rates. The translational motion corresponds to the distance between the teacher and the robot. Experiments and a case study is performed with different participants with diverse knowledge of robotics to judge the ease and flexibility of the demonstration modes.

Visual features for robotic behaviors
The first step towards designing a robot behavior is to identify the features that are characteristic of the behavior. An exhaustive overview of different image based features that are relevant for a robotic behavior is presented. An omnidirectional camera pointed towards the floor with a 360° horizontal field of view is used as the primary sensor. The acquired omnidirectional image is first segmented using an ensemble of experts segmentation scheme [PNHB10]; [PNHB11] to obtain floor and non-floor regions. Various features from the segmented image that reflect the shape and moment of the floor together with more behavior specific features such as proximity of the obstacles to the robot are extracted. The extracted features are then reduced to relevant feature subset for different behaviors using wrapper based feature selection approach. The fidelity of the subset is established by cross-validating them against unseen examples.

Scenario and context specific learning
Learning robot behaviors needs a good understanding of the tight coupling between the behavior and the environment. This aspect of the dependency is called as the situatedness of a behavior [Mat01]. We capture this aspect through visual features relevant to individual scenarios which are extracted, identified and learned to generate a multi-

ple model autonomous navigation [NPHB11a]; [NPHB11b]. Scenario corresponds to the robots local environment described through the geometry and the spatial structure of the floor. Supervised examples of three indoor scenarios (corridor, open room and cluttered) are generated manually by driving the robot in these environments and a semantic classification of the scenarios are performed to identify the current scenario the robot is in. We also examine unsupervised semantic classification using self-organizing maps to analyze the significance of the data in determining the scenario. The detected scenarios are compared with the hand labeled semantic labels to see if manual intuition concurs with that of the data. Once the scenario is identified, the context of operation within the scenario is also learned. A context is expressed through a short-term perceptual trace of the robots recent history. The correct context of operation is found by matching the memory trace of the robots recent history with demonstrated examples recorded in the different contexts. Two time series trajectory matching techniques, namely Dynamic Time Warping (DTW) and Longest Common Subsequence (LCSS) are tested and their individual pros and cons discussed. As an alternative, behavior fusion within an entire scenario are modeled and trained with Artificial Neural Networks. The generalization of these models against unseen data and scenarios are exhaustively validated and tested.

Supervised learning of behaviors
In contrast to learning scenario specific behaviors, learning individual behaviors are explored. This is an another method of modularizing behavior fusion, where instead of learning the environmental dependence, the perception-action mapping of every behavior is learned. The learned behaviors thus represent the smallest subunit of the visuo-motor mapping architecture. Behaviors such as corridor following, obstacle avoidance and homing are learned through static and dynamic mapping architectures. Feed forward and recurrent neural networks are trained to learn the different behaviors. The performance of the behaviors using static and dynamic mapping are exhaustively tested and validated in unseen environments for robust performance. Another interesting behavioral representation is discussed in [SDE95]; [Bic00]; [FWTPK03]; [War06], where they represent robotic behaviors as dynamic systems emerging from their interaction with the environment. The behavioral dynamics are expressed by a vector field with attractors and repellers in the cross-space of perceptual variables and robot action. Attractors correspond to regions that correspond to stable states and repellers correspond to regions that need to be avoided. The dynamics of corridor following, obstacle avoidance and homing are learned using Gaussian mixture models by ensuring asymptotic stability using the Stable Estimator of Dynamic Systems algorithm [KZB11]. The acquired behavior dynamics are tested exhaustively in different environments and their differences to a neural representation are discussed. The learned behaviors are also tested experimentally on the robot in an indoor office environment. Results of behavior coordination through arbitration and command fusion as well a behavior classification architecture is proposed [NPHB13].

Learning behavior fusion
The key challenges involved in learning a monolithic model of behavior fusion across all scenarios in contrast the other presented representations are further analyzed. The difficulty involved in selecting the relevant feature set that expresses all the characteristics of the behaviors is discussed. A feed forward neural network [NPHB10] and an instance

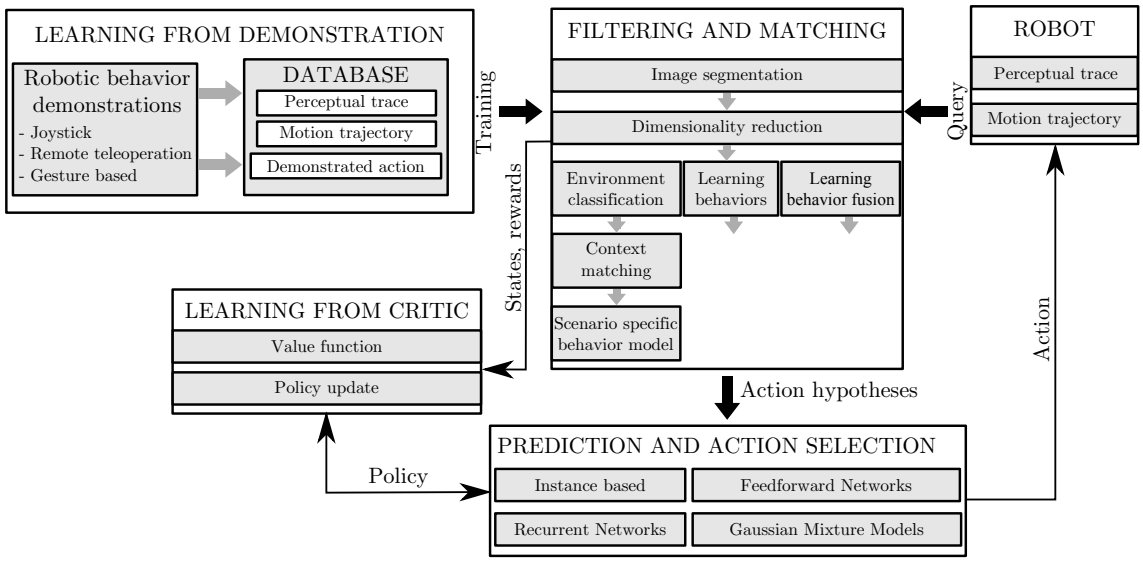


Figure 1.1.: System architecture of learning from demonstration framework for visual behaviors.

based model [NPHB09] are trained to learn the behavior fusion. The core differences in its approach compared to learning individual behaviors are presented and their results correspondingly analyzed.

Reinforcement learning of visual behaviors
Finally, Reinforcement Learning (RL) within the framework of LfD for learning visual robot behaviors are presented. Starting from basic knowledge about the task provided by the teacher, we present a framework where a visual corridor following behavior improves its policy by learning and optimizing the value function. Behavior dynamics learned from Gaussian Mixture Model (GMM) is used as the initial model which is iteratively improved and optimized. Two different reward functions, namely dense and spare rewards are designed and tested. The framework is tested for potentially challenging problems such as one shot learning and learning from scratch. The iterative policy improvement for the different problems together with the final model performance both in simulation and on the real robot are tested.

1.2. Thesis outline

The system architecture of this thesis follows a bottom-up architecture where we focus on the individual ideas through the chapters to generate a bigger vision (Fig. 1.1). Each chapter of the thesis can be independently read which is introduced with the necessary background, motivation and its corresponding state of the art. Chapter 2 gives a brief overview of the different fundamental concepts presented in the thesis ranging from behavior based robotic architecture to the principles of learning from demonstration. This is followed by Chapter 3, where the extraction of different visual features are explained. The chapter also explains how to select the optimal feature set for a behavior and presents an application example at the end. Chapter 4 presents three demonstration modes for instructing a mobile robot. Direct and remote teleoperation together with NUI based demonstration method are presented and their performance are compared against each

other. Scenario and context based behavior modeling is addressed in detail in Chapter 5. Learning individual scenarios using artificial neural network representation is discussed in Chapter 6. Static and dynamic neural network representation with manually extracted visual features and automatically generated principal components are explained. Chapter 7 explores learning behaviors expressed as dynamic systems using Gaussian mixture models. In the second part of the chapter, learning behavior fusion across all scenarios and their disadvantages to the modularized learning are presented. In Chapter 8, the integration of reinforcement learning with LfD framework to learn visual corridor following behavior is dealt in detail. Chapter 9 closes the dissertation with a summary and gives an outlook into future works in this field.

2
Background

Behavior based robotics proposed by Brooks in [Bro86], was a revolutionary concept at its time to describe robot motion behaviors. According to behavior based control, the whole sensor-motor relationship can be broken down into simple primitive substrates called behaviors which are then further sequenced and coordinated to solve complex problems. This methodology paved the way to realize the fundamental necessities for intelligent robotics, namely situatedness and modularity [Alt03]. Situatedness means that the sensors used to perceive the environment and the motors that execute an action on the robot are tightly coupled. A brilliant example of such a system is illustrated with the Braitenberg vehicle [Bra86]. Functional modularity on the other hand defines the ability of accomplishing complex movements through simple and independent modules. The significance and origin of functional modularity in biological systems and its application in robotics is discussed in [CNPW98]. Following a *think the way you act* biological paradigm, behavior based system follows a bottom up approach, where the survival behavioral primitives are placed in the lower level hierarchy with complex behaviors placed above them. The higher behaviors then typically influence the lower level behaviors to subsume and override their inputs. Also called as subsumption architecture [Bro86], it allows scalability in adding more behaviors thereby achieving complex capabilities. A broad review of these concepts is presented in [Bro86]; [Ark98]. The behavior based control left many desirable properties such as a planning module. The planning module carries a disadvantage in that it comes at the price of the fast execution time realized by a pure behavior based control but allows designing complex movements. Hence, with the reactive control as the basis, a hybrid deliberative-reactive architecture is developed with a planning or a reasoning module to steer the robot [Ark98]. This type of organization called as the Plan-Sense-Act approach [Mur00], allowed complex tasks such as localization, path planning to be accomplished with reactive behaviors. An impressive implementation of this architecture is the museum tour-guide robot developed by [BCFHLSST99]. Behavior based systems differ from hybrid deliberative-reactive systems in that they do not need a centralized planner to make a decision. Behavior based systems can store different representations in a distributed fashion, thereby requiring no centralized arbiter [Mat01]. The network of intercommunicating behaviors thereby achieves the same performance with less computational overhead as required by a centralized planner.

A robotic behavior is nothing but the functional mapping β of a stimulus or a percept $s \in S$, where S is the stimulus domain to a response $r \in R$, where R is all the range of possible responses. A behavior is thus indicated by the triplet (S, R, β) with the mapping

$\beta : S \rightarrow R$ [Ark98]. For every behavioral mapping β, one can establish the desired stimulus variables that trigger a specific action. For example, the presence or the distance of an obstacle is the required stimulus for an obstacle avoidance behavior whereas the location of the docking station is the stimulus for a homing behavior. In a system with multiple behaviors each with different stimulus strengths, the appropriate action to be acquired is determined by coordinating the behaviors.

Behavior Coordination

Behavior coordination problem directly translates to an action selection problem [Pir99]. [Mae89] defined the action selection problem as following:

> How can an agent select *the most appropriate* or *the most relevant* next action to take at a particular moment, when facing a particular situation?

In order to choose the most appropriate action, a behavior coordinator or an arbiter module is required which works similar to the planning module in a hybrid architecture. Given multiple behaviors with different stimulus strengths, the arbiter has to decide the right action to be applied to the robot actuators. Given a stimuli $\mathbf{s}_i$ for each behavior i at a given time t, g_i indicating the relative strength of each active behavior, r_i denoting the response of the behavior and the coordination function C [Ark98], the final response ρ is given by $\rho = C(\mathbf{G} * \mathbf{R})$ where, $*$ indicates the scaling operation of every behavioral gain g_i belonging to the set of behavioral gains $\mathbf{G}$ and the corresponding behavioral response r_i in the behavioral response set $\mathbf{R}$.

The coordination schemes are divided into two types, namely arbitration and command fusion. Arbitration corresponds to a competitive method where the behavior with the highest gain is given control (See Fig. 2.1(a) for a suppression network for three indoor behaviors). It is a *winner take all* approach where the single response of the winning behavior is executed on the robot. Subsumption architecture is a typical example of an arbitration approach. Each higher level behavior carry larger priority towards the final motor action. The higher behaviors can suppress or inhibit the lower level behaviors depending on the set sensor conditions. In contrast to the priority based arbitration, [Ros97] proposed a more democratic alternative called as the Distributed Architecture for Mobile Navigation (DAMN) where each behavior votes for a particular actions thereby the action that receives most votes is chosen. For such a system, the robot velocity actions are discretized to predefined responses and the behaviors vote for a corresponding steering action. Another formalism in contrast to the priority based arbitration is the state based arbitration using finite state machines proposed by [KCB97]. Behavior selection is triggered using a state transition where a certain event changes the current state of the robot thereby requiring a new behavior. One interesting example of the use of finite state machines is the work of [KIOIN02] who use it for development and evaluation of an interactive Human-Robot communication.

Command fusion is a cooperative approach to action selection where the final output is a fusion of the behaviors. The fusion methods typically fall into three major categories, namely voting, fuzzy and superposition. Action voting proposed by [HB95] follows a similar approach as with arbitration only that here the behaviors in addition to voting for one action also possess the possibility of inhibiting conflicting actions. The votes and inhibition are summed for each action and the action with the highest cumulative vote wins

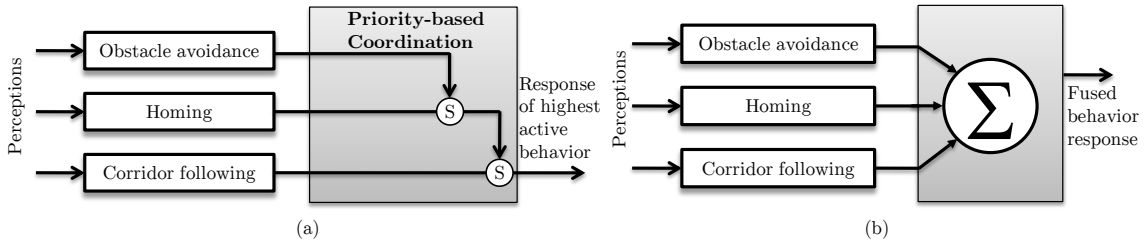


Figure 2.1.: Behavior coordination schemes. (a) Priority based coordination. The **S** indicates the suppression or inhibition of lower level behaviors by higher priority behaviors. (b) Fusion with vector summation [Ark98].

[Pir99]. With fuzzy approaches, each behavior defines a membership function over the possible set of actions, whose outputs are then combined using an appropriate operator (e.g. max operator). The defuzzification is then performed on the fuzzy set to obtain the final crisp action for robot control [Saf97]. The third method is the superposition principle, which is by far the most straight forward method to find an appropriate output. Each behaviors relative strength or gain is directly used and multiplied with its behavioral action whose outputs are then summed. Fig. 2.1(b) illustrates this process. This method has a disadvantage in that two behaviors with equal gain but with diametrically opposite outputs tend to cancel out. This is overcome by the inhibition voting schemes of [HB95] explained before.

Dynamic System Approach to Behavior Based Robotics

Behavioral dynamics proposed by [SDE95] is based on the theory of non-linear dynamical systems. Any robot behavior because of its situated nature exhibits a tight coupling with the environment and through this interaction behavioral patterns
emerge [Mat01]. The behavior and the environment resonate together as a mutually paired dynamical system with a stable attractor in the space of its behavioral variables. Behavioral variables are the variables that can describe a particular behavior and typically define the dimensions along which the behavior can change [Bic00]. Thus, a behavioral state is defined by a point in the cross-space of the behavioral variables. A behavior expressed in terms of differential equations, the persistent behavior patterns correspond to attractor equilibrium states of the behavior dynamics. Starting from any initial configuration in the behavioral state space, the resulting trajectories of the behavior dynamics converge them to their attractors. Attractors are constant solutions of the dynamical systems where the system does not change in time. Conversely repellers states correspond to unstable state spaces which repel the behavior trajectory away. The design requirements for behavioral variables are given [Bic00]; [Alt03] as follows,

- at any point, the behavior must be completely expressible with its behavioral variables,
- the values of the behavioral variable must be independent of the current value,
- it must be possible to express the same values using another on board sensor or some internal world model and
- it must be possible to derive the dynamics of these variables thereby impose on the actuator system of the robot.

The examples in [SDE95] illustrate the design reasons behind the selection of behavioral variables by considering a movement of an autonomous vehicle in a plane. The movement of such a robot must be controlled in that the location of obstacles typically need to be avoided and desired target locations (if any) must be reached. In this case, the heading direction ϕ and the translational velocity v are deemed to be sufficient behavioral variables. The heading direction at any point can be used to define the angular location of the obstacle and the target, the value of these locations are independent of the current heading of the robot and the time derivative $\dot{\phi}$ can be directly controlled on most of the platforms. Application of behavioral dynamics to mobile robot navigation has been studied in [Bic00]; [Alt03]. In [Alt03], the individual behavioral dynamics are designed based on sonar sensor information and the coordination between the different behaviors is done through competitive dynamics by assigning weights for each behavior. Performed in indoor environment they discuss go to, obstacle avoidance, corridor following, wall avoidance and door passing. In this thesis, the dynamic representation for a visual robotic behavior is discussed in chapter 7.

2.1. Learning in robotics

Machines that can learn is one area that has been vigorously explored during the last few years. Starting from the iconic first line of Alan Turing in [Tur50] where he asks, *Can machines think?* and following through history with a computer (Deep Blue) beating the world chess champion [CHJH02] to natural language processing wonder called Watson [Fer11], machine intelligence has been on an exponential increase in awe and attention throughout the world. Robots that can learn, be taught concepts, motor skills and teach how to use their sensors are all excellent footsteps towards artificial intelligence. [Ark98] presents a definition of learning from the perspective of robotics;

> Learning produces changes within an agent that over time enable it to perform more effectively within its environment.

Within this operational definition of learning and adaptation, the hardest task is to choose the right techniques to learn. The mode of learning and the type of task to perform dictates the learning algorithm. For an industrial robot performing only repetitive motions, it is sufficient to learn this action as accurately as possible. It may never need any new information regarding the motion. But for a navigating mobile robot, learning new information is paramount to the execution of a task. Learning can also occur in other situations where it is impossible to provide precise information about the environment [Gro01]. These are typically scenarios where robots are used where a human cannot venture. In these cases, learning in robots as with humans can happen over the entire life time and is duly named as Lifelong learning [Thr96]. In such environments where the characteristics constantly change, determining the relevance of sensor information and composing them to behavioral action is the foremost aspect to be figured out. Some of the mechanisms where a robot system can be made to learn behaviors are through supervised learning, reinforcement learning, evolutionary learning and learning from demonstration.

Supervised Learning
Supervised learning refers to learning the mapping function from a set of labeled training data. Each example is given as a pair of input and its corresponding output. Artificial Neural Networks (ANN) are one such suited supervised algorithm, where the learning is achieved as a result of alteration in the synaptic weights of the networks. The output of such a network is first evaluated against potentially unseen data and if unsatisfactory, the weights are correspondingly updated and trained again. [Pom89] presented one of the earliest examples of neural network for autonomous navigation. [Thr96] introduces life-long learning using explanation based neural network through knowledge transfer across multiple learning tasks. ANN models used today are just rudimentary products of the inspiration acquired from human brain. One main disadvantages of the method is that the structure of the ANN is fixed before learning. The ability of ANN to extend and adapt to newer data remains a challenge. New attempts to learn layered models of inputs are the deep neural networks [Sch92]; [HOT06], where multiple layers of the neural network can be first pre-trained one layer at a time in an unsupervised manner. The weights are then used to further train a supervised neural network for fine-tuning. Powerful artificial reasoning systems like Watson [Fer11] use deep learning to achieve highly complex inferences. In this dissertation, ANN as tool for supervised learning of visual behaviors are examined in detail in chapter 6.

Reinforcement Learning
RL refers to techniques that uses a critic to guide the development of a policy by giving rewards and penalties for an action. Also called as learning from critic [SB98], the approach gets a feedback from the critic for every state within the environment. This allows the robot to start without any prior knowledge of the environment. The goal of RL is to device a mapping from perceptual states to appropriate actions such that the policy maximizes the long term reward. Thus, the reward function determines the quality of the behavioral response. The challenging part of the design is the reward or the credit assignment problem [Ark98]. The authors of [SPK02] use dense and sparse rewards to learn robotic behaviors on a mobile robot. The work puts a few open questions regarding the complexity of behaviors that can be achieved when working with only sparse reward function and how long does the training take in such cases. We inspect these issues in Chapter 8, where using a sparse reward function design for a corridor following behavior, the behavioral policy is iteratively improved.

Evolutionary Learning
Evolutionary learning refers to unsupervised learning methods where a behavior is evolved over generations from a population of candidate controllers [Ark98]. Drawing inspiration from natural evolution, evolutionary learning uses genetic algorithms where, the initial population consists of a set of solutions. By performing a cross-over to select the best fit individuals from the population, they are further combined to produce new population of controllers for the next generation. At every generation, the quality of the population is evaluated to make sure that the current generation is fitter than the previous generation. [Hof97] presents a genetic algorithm method to learn a mobile robot wall following behavior. The problems with evolutionary learning is that it is computationally expensive to evaluate entire populations and to mutate. In addition, the design of the fitness function

also plays a big role in deciding the quality of the final solution.

2.2. Learning from demonstration

LfD or imitation learning is one approach which is gaining increasing attention in the robotics community [ACVB09]; [BCDS]. Typically speaking, all of the different learning algorithms presented above can be integrated into the LfD framework. It involves a human demonstrating a behavior to the robot during which the robot observes and records the corresponding perceptions and actions. From the observations, any of the appropriate machine learning algorithms can be utilized to generate a policy mapping between the perceptions and motor actions. This allows a behavior demonstrated in one sensor modality to be transferred to a potentially different sensor modality. The key issues in LfD are [ND01] as following,

- Which aspect of demonstration must be imitated?
- What is the evaluation metric? or How the skill should be encoded?
- How to compute correspondence between the teachers and the robot?
- How to identify the teacher and accomplish knowledge transfer through an intuitively designed demonstration interface?

LfD has been in extensive research field for a range of robotic tasks ranging from using motion primitives to learn trajectories [SMI07]; [Cal09], gesture learning in humanoid robots [LY96]; [Cal09], object recognition [RSGB09], robotic navigation [LZ00]; [NCJOF08] to learning body schema and percept for imitation [ACH05]. In all of the above mentioned cases except autonomous robotic navigation, the configuration or the joint space is known to the robot manipulator. For a mobile robot assumed to be in horizontal plane, the task space is the group of all possible positions and orientations in the plane. This space is a Lie Group in two dimensions [Nof99]. The presence of obstacles create blocks or holes in the task space thereby inversely generating a set of configurations for a collision free path for the robot. In the event of no knowledge of the current traversing environment, the generation of the list of configurations for the robot becomes impossible as the number of task space configurations go to infinite. The authors of [WZ98] back in 1998, used Fuzzy Associative Matrix to acquire a mapping between the sonar sensor readings to the plausible trajectories of the robot to learn trajectory velocities. This enabled the robot to acquire object avoidance, wall following and goal seeking behaviors simultaneously. They repeatedly select new trajectories at random and imitate them until a collision is detected. The downside of this approach is that it prolongs the learning time and has limited generalization ability in narrow passages. Another development is the use of neural approaches to behavior learning introduced by [TN99]. An online learning scheme using a mixture of recurrent neural networks as experts are used to account for the different sensory-motor flow. The network because of its recurrent nature also learns the associated context. For behavior LfD, the goal is to generate a controller that enables the robot to repeat a demonstrated behavior from a set of demonstrated examples. Nevertheless, a conception of *what is a satisfactory repetition?* is not clear. [BH10] present a formalism

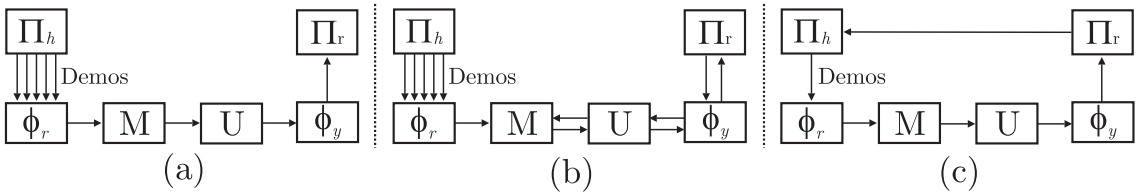


Figure 2.2.: Modes of learning in LfD [Bil13]. (a) Batch learning, (b) self-improvement learning and (c) interactive learning. Human demonstrates examples of a behavioral policy Π_h whose correspondences are solved and encoded in the task space coordinates of the robot using the operator ϕ_r. These representative features are evaluated to improve the performance metric (M) to update the model (U). Using again the operator ϕ_y the final learned behavior policy Π_r is deployed on the robot.

of LfD where two interpretations to this question are provided. One form of imitation is the so called *action-level imitation* where the job of the robot is to imitate exactly the actions shown during demonstration. Typical example here is a pick and place robot in a controlled industrial environment. Squared error as an evaluation metric here captures the accuracy of the learned policy quite robustly. The second case is called as *functional imitation*, where the capability of the robot to generalize situations unseen using only demonstration knowledge is evaluated. Here, the imitation trajectory path becomes secondary and the perceptual state-action pairs are judged. In addition to squared error, a metric of similarity of the trajectory between the demonstrated and the learned policy is important [Bil13]. A further distinction of generalization is achieved by the mode of learning achieved. [Bil13] presents batch learning, self-improvement learning and interactive learning as the three modes of LfD architectures (Fig. 2.2). In batch learning, the demonstration data collected are learned offline and the policy is then used during reproduction. This invariantly involves the teacher to take care in generating sufficient samples with a lot of initial positions. A major part of this thesis from Chapter 5 - 7 performs batch learning of behaviors. Self-improvement learning builds from the policy learned from batch learning with a possibility to improve the policy on acquiring new data. The new samples generated from the batch learning process is then iteratively updated using RL [GHCB07]. Chapter 8 of this thesis explores this aspect of learning for a corridor following behavior with RL. In Interactive learning [Che09], the teacher is actively involved in the self-improvement cycle. The teacher provides examples to the robot as and when it is needed. This means, a flexible and fast demonstration interface between the teacher and the robot is an utmost necessity. The demonstration modes explained in Chapter 4 can be considered for an eventual interactive-learning of navigational behaviors.

3

Visual Features for Robot Navigation

The same meaning can be expressed in many different ways, and the same expression can express many different meanings.

– Halevy, Norvig and Pereira, *The Unreasonable Effectiveness of Data*

Given a sequence of perceived images, the first step towards designing visual navigational behaviors is to understand the scene captured. Images offer us a large amount of information about the environment, nevertheless the challenge remains in interpreting this enormous amount of data into effective usable information. Following an initial preprocessing step to enhance the clarity of the image, extraction of generic features that reflect the information numerically is done. The role of feature extraction is to express the image through a set of relevant and appropriate cues that are both rational and computationally efficient. By expressing the pixel intensity of an image at location (x, y) as $I(x, y)$, [Cor96] offers a description of the visual features as image functionals given by,

$$f = \int \int_{Image} \mathcal{F}\left(x, y, I(x, y)\right) \mathrm{d}x\mathrm{d}y \tag{3.0.1}$$

where, $\mathcal{F}$ is a linear or non-linear mapping between the image and its extracted features. Two classes of features are addressed. First, the features are extracted through human intuition deciding on the apparent relevance of them to successfully describe the scene captured. This is accomplished using a blend of existing computational methods of image understanding. The advantage of the first approach is that the features arising from human input carry relevance and information that are rational and easier to interpret in relation to the task demonstrated. The disadvantage is that it needs an expert input to carefully focus on the imagery and decipher their correlation for a better functional description. This is a time consuming task and tends to miss some aspects that may be hidden or not obvious to the human eye. Very often, features originating in the latent space of the image also carry substantial information about the scene. Principal Component Analysis (PCA) and Principal Component Regression (PCR) are methods to extract latent space feature descriptors. We term the second set of features as automatically generated features because of the mode of extraction. This chapter discusses the identification and extraction of both the family of features relevant for the purpose of indoor robotic visual navigation. It introduces the fundamental step in building behavior models for LfD with mobile robots where the robots perception is broken down into small behavioral descriptors.

3.1. Free floor segmentation

Feature extraction begins with whole scene understanding from an image by partitioning it into relevant and meaningful segments. Design of vision based robotic behaviors requires a scene understanding that is robust to the variations of geometry and appearance of the environment. The performance of existing indoor segmentation schemes suffers from lack of depth information, augmented by substantial variations in geometry, texture and visual appearances of an indoor environment. One approach is to construct a 3D model of the environment using laser scanners and then perform a semantic classification [MMSB05]. However, it is prone to ignore information in the image that might be useful to improve the navigational capabilities of the robot as a vision system in contrast to proximity sensors is able to distinguish between objects such as obstacles, walls, doors or a docking station. Furthermore, the ultimate objective of vision based navigation is to achieve autonomous behaviors that rely purely on image rather than the geometric reconstruction from range sensors. This motivates our experimental setup where a Photonic Mixture Device (PMD) camera [Pla06], delivering 3D range data is fused with an omnidirectional camera. The PMD camera has a 204×204 pixel resolution across a $40^\circ \times 40^\circ$ field of view and the omnidirectional camera consists of a Charge-coupled devive (CCD) camera with a hyperbolic mirror with a vertical field of view of 75° of which 15° is above the horizon. The mirror is directed towards the floor thereby capturing the 360° view of the robots local environment. The PMD camera provides depth information in addition to an intensity and amplitude image, thus does not require texture or contrast information. The depth information is first used to detect the different semantic entities to the front of the robot. An initial 3D scan of the frontal region by the camera is segmented into different planar surfaces that belong to walls, floor and obstacles by means of Random Sample Consensus (RANSAC) [FB81] algorithm. One of the drawbacks of the 3D camera is the limited field of view and a low resolution making it unusable for robust navigational schemes. Hence, in order to overcome this limitation an omnidirectional camera is utilized together with the 3D camera. This setup allows it to fuse depth and high resolution intensity images of the environment. Fig. 3.1 shows the experimental setup of an omnidirectional camera and a 3D PMD camera mounted on the pioneer 3DX mobile robot. The advantage of this integrated 2.5D camera system is exploited by transferring the identified planar segment points onto the omnidirectional camera by a homography obtained from the intrinsic camera parameters. The projected points on the omnidirectional camera act as seeds or markers for multiple 2D segmentation algorithms such as watershed, region growing, histogram backprojection and graphcut [CJSW01]; [SB91]. The quality of marker based segmentation depends on the initial seeds, the texture and illumination of the environment. No segmentation algorithm performs robust across all conditions and situations. Every algorithm exhibits contrasting performance in different scenarios. Hence, an ensemble of experts [Pol06] image segmentation scheme is performed where multiple naive Bayes classifiers are trained on features extracted from the outputs of the different 2D segmentation algorithms together with other topological image features. By maintaining different appearance models and seeds for floor and obstacle regions derived from the ground truth 3D data, the final segmentation is obtained by fusing the heterogeneous classifier outputs to label an individual image pixel as either floor or obstacle. Fig. 3.2 shows the system architecture of the ensemble of experts segmentation scheme. One has

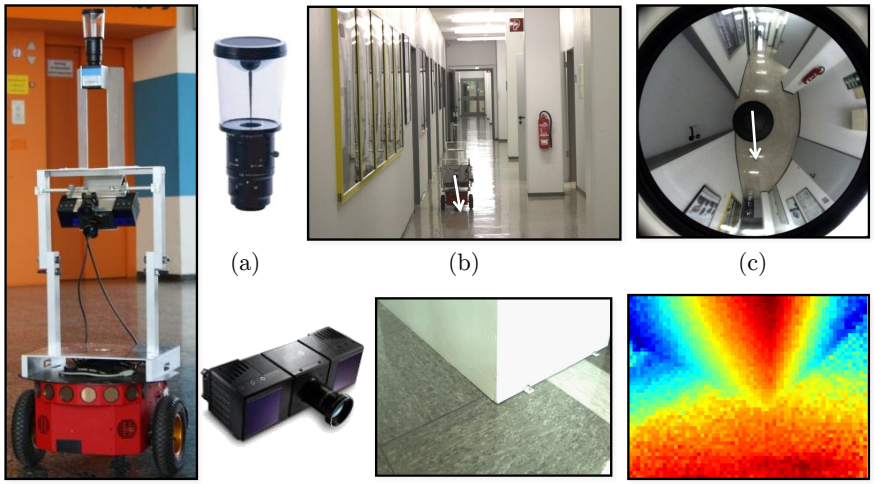


Figure 3.1.: Omnidirectional and PMD camera mounted on Pioneer 3DX mobile robot. (a) Omnidirectional camera with a hyperbolic mirror, (b) robot mounted with an omnidirectional camera in a corridor, (c) corresponding omnidirectional camera image, (d) PMD camera, (e) monocular image of a scene viewed by the PMD camera and (f) the corresponding depth image from PMD camera.

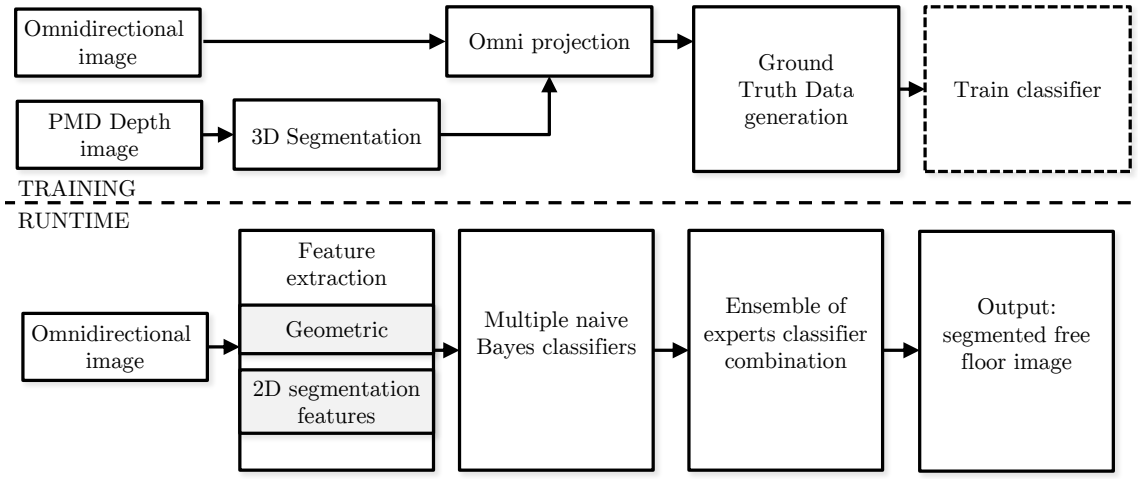
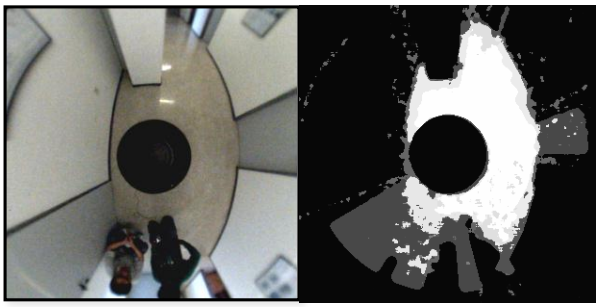


Figure 3.2.: System architecture: Ensemble of experts based free floor segmentation [PNHB11].

to notice here that the depth information from the PMD camera is used only for training the floor and obstacle models and are not used during runtime. Fig. 3.3 shows the input omnidirectional camera image and the corresponding segmentation output. Ensemble of experts is currently an active area of research and for more details refer to the works of [PNHB10]; [PNHB11].

3.2. Visual features

Training examples are generated by demonstrating different behaviors to the robot during which, the camera image and the action are recorded. This section explains the identification and extraction of visual features from the segmentation examples.

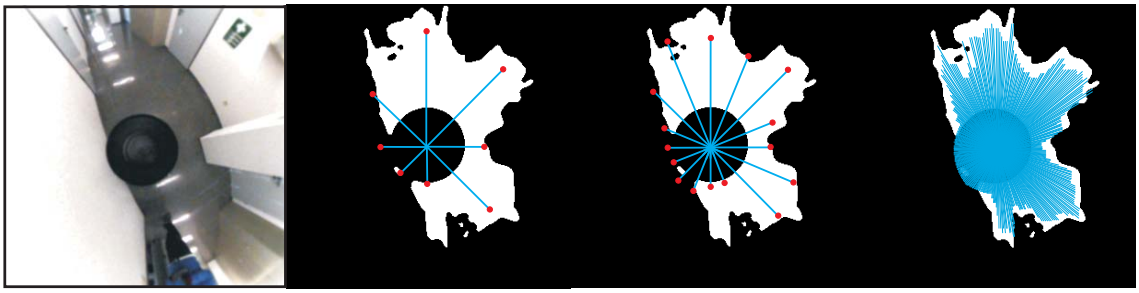


(a) (b)

Figure 3.3.: Segmentation Images. (a) Omnidirectional camera image and (b) Ensemble of experts segmentation image.

3.2.1. Spatial features

The segmented image presents itself as a similarity map P where each pixel value represents floor similarity. The location of the pixel in the image is described in cylindrical coordinates by a radial distance r and angular orientation θ. This similarity map $P(r, \theta)$ is filtered to remove all the pixels that are below the threshold of 0.7 to generate a binary segmentation image. Thus, the pixels with value 1 denote floor and 0 denotes obstacles or walls. The binary image is discretized into sectors defined by an angular range and radial intervals. The sector partition is uniform with 16 bins in angular direction and each partition is further divided to segments using 20 uniform bins along radial direction. A segment is labeled as occupied in case the fraction of non-floor pixels exceeds a given threshold. In each sector, the closest obstacle distance (d_{min}) is given by the radius of the first non floor segment in radial direction. Thereby, we extract a list of visual range features that directly correlate to the nearest obstacles in the immediate vicinity of the robot. Fig. 3.4 shows the mapping between the segmentation and the distance to obstacles for different angular resolutions. By limiting only to the front of the robot in the angular range of $[-90°, 90°]$, the proximity and direction of the closest obstacle is expressed with the inverse of the closest distance $\frac{1}{d_{min}}$ and its angular location (θ_{min}). Additionally, a critical zone of $[-30°, 30°]$ is defined in the immediate front to identify obstacles that predominantly trigger obstacle evasion. The inverse of the critical distance (d_c) and critical orientation (θ_c) are also registered. Furthermore, the bearing of the safe direction (θ_{safe}) with a minimum separation of D that is closest to the current robot heading is also computed. θ_{safe} is computed such that the robot always maintains a sufficient separation to the obstacles on either side. In the event of the nearest obstacle emerging within the robots critical zone, both critical point and the nearest obstacle point contain the same information. Fig. 3.5 shows the aforementioned parameters on the omnidirectional and the segmented image. To ensure continuity of avoiding motions in corners or dead end situations and to promote continuation of turns, a flag that indicates the previous turning direction is also recorded. It attains a value of -1 and 1 for left and right turns and 0 in the event of no turn respectively. The distribution of the obstacle distances around the robot is analyzed by projecting them down to the perspective panoramic view of the omnidirectional image. Fig. 3.6 shows the projection of 8 sector partition on one such panoramic image in which the height of the shaded columns indicates the extension of free space along that direction. Furthermore, the relative difference of floor area in opposite



(a) (b) (c) (d)

Figure 3.4.: Scan lines represent the distance to the closest obstacle in a considered sector. (a) Omni-directional image and (b),(c) and (d) free floor scan with a resolution of 8 sectors, 16 sectors and 360 sectors.

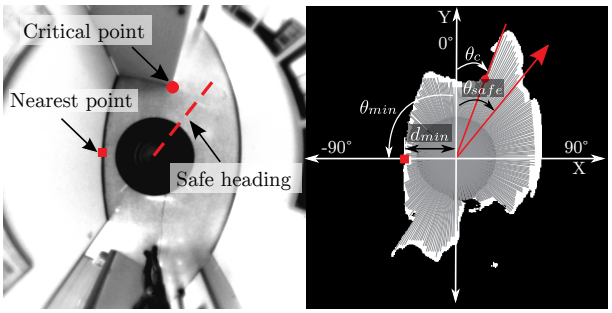
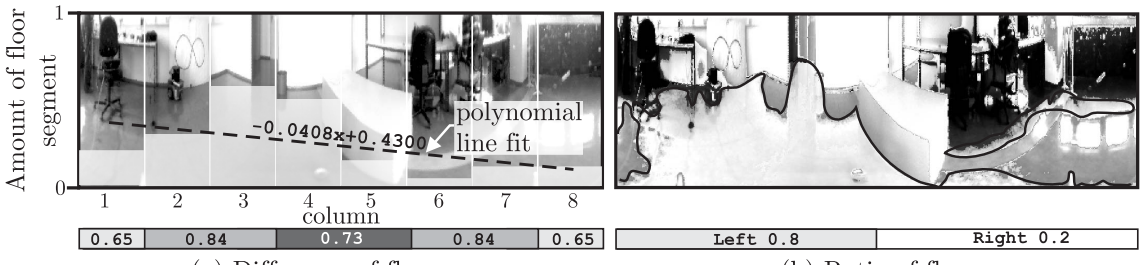


Figure 3.5.: Nearest point, critical point and safe heading features.

column pairs (1-8, 2-7, 3-6, 4-5) which indicate the lateral alignment and orientation with respect to the floor are also registered. Similarly global features that capture the total area of the floor, obstacles, ratio of floor to non-floor (R_{fnf}), maximum width of free floor region (W_{max}) in the binary segmented image and the slope and intercept of the line fitting the two extreme columns in the panoramic image are computed.

3.2.2. Image moments

Image moments use statistical image features to express the spatial characteristics of a segmented image. For a binary segmentation image, where the intensity of the pixels take either a 0 or 1 as its value, the $(p+q)^{th}$ order moment for a digital image (m_{pq}) can be



(a) Difference of floor area (b) Ratio of floor

Figure 3.6.: Panoramic image features. (a) Height and amount of floor segment in every column, polynomial line fit, ratio of the floor area between column pairs 1-8, 2-7, 3-6, 4-5 and (b) free floor segmentation on panoramic image and the ratio of the floor area between the right and left sides.

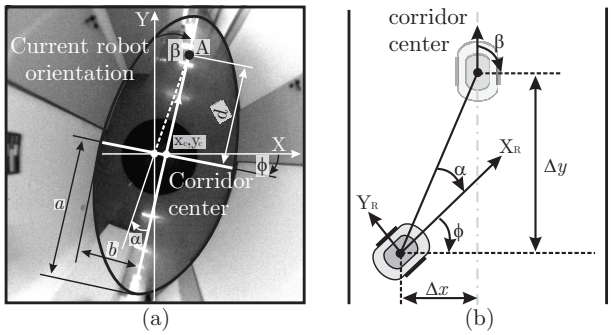


Figure 3.7.: (a) Image moment features in the omnidirectional view of a corridor and (b) the feature interpretation from the robot pose.

expressed using the formal description given in 3.0.1 as,

$$m_{pq} = \sum_x \sum_y I\,(x, y) x^p y^q. \tag{3.2.1}$$

The summation is done over all the x and y image pixels. The zeroth moment (m_{00}) of the binary image effectively computes the area of the segmented area. Correspondingly from the first moments (m_{01}, m_{10}), the centroid of the image (x_c) is computed directly.

$$\text{Centroid of the region: } x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}. \tag{3.2.2}$$

Furthermore, by calculating the perimeter (p_s) derived from the zeroth moment, a simple metric about the circularity (ρ_s) of the floor is determined using 3.2.3.

$$\rho_s = \frac{4\pi m_{00}}{p_s^2}. \tag{3.2.3}$$

The zeroth and first order moments are variants to translation and thus are not able to determine the orientation of the shape. The second order moments or the geometric central moments are invariant to translation and rotation [FZS09], and can be used to determine the principal axis orientation θ_s by,

$$\tan\,(2\theta_s) = \frac{2m_{11}}{m_{20} - m_{02}}. \tag{3.2.4}$$

To get the second order moments on the segmented free-floor region on the omnidirectional camera, the distribution of the free space is approximated with an ellipse with an elliptical tilt of ϕ. Considering a look ahead point A on the major axis at a distance (2m) from the centroid, the angles α and β are extracted. In a corridor, α represents the lateral offset angle or a measure of robot alignment within the corridor and β represents the alignment of the robot to the corridor center. Fig. 3.7 shows these features on the omnidirectional image of a corridor and the corresponding robot pose within the corridor.

3.2.3. Shape features

Computational shape analysis on the segmented image allows to derive pure geometry or shape based descriptors. These descriptors are either boundary based or region based

features. The requirement is to uniquely characterize the shape from the descriptor vector that is invariant to translation, scale and rotation [CC00]. It is legit to note here that some of the spatial and image moment features described in the previous section already explain away some geometrical characteristics of the shape. These are typically the features arising from shape statistics namely mean, median, maximum and minimum width of the floor, their ratios and metric descriptors such as perimeter, area, centroid, maximum-, minimum- and mean- distance of the boundary points to the centroid and diameter. The following shape features dig a little deeper into the realm of shape features to extract more abstract descriptors.

Curvature-based descriptors

While the curvature by itself qualifies as an image feature, typically these vectors are too long, redundant and are computationally expensive to process them [CC00]. This is overcome by breaking down the curvature to specific descriptors listed below.
Curvature statistics: The mean, median, variance, standard deviation, maxima, minima and the corresponding inflection points of the curvature.
Bending energy: Bending energy of a curvature (B_c) is defined as the integral of the squared curvature over the length of the curve given by [CC00],

$$\text{bending energy } B_c = \frac{1}{P_c} \int \kappa\left(t\right)^2 \mathrm{dt}. \tag{3.2.5}$$

P_c is the perimeter of the curve and $\kappa\left(t\right)$ is the curvature of a parametric curve $c(t) = (x(t), y(t))$. x and y are the coordinates of the point evolving over time t to obtain the curve $c(t)$. The curvature $\kappa_s\left(t\right)$ is given by,

$$\boldsymbol{\kappa_s} = \frac{x'y'' - y'x''}{\left(x'^2 + y'^2\right)^{3/2}} \tag{3.2.6}$$

differentiated with respect to time t. In other words, bending energy expresses the amount of energy needed to reshape the closed curvature of the shape into a circle without changing the perimeter of the shape [MIJ08].

Shape statistics

From the shape statistics descriptors such as mean and median distances of the boundary points to the centroid of the shape; higher order moment features such as skewness and kurtosis of this distribution is analyzed. Skewness of the distribution measures the asymmetry of the distribution around the sample mean. The skewness (γ_1) of the boundary distances $\mathbf{D_b}$ to the centroid is the third standardized moment given by,

$$\gamma_1 = \Sigma \left[\frac{\mathbf{D_b} - \mathbf{D_c}}{\sigma_d^3}\right] \tag{3.2.7}$$

where, $\mathbf{D_c}$ is the mean distance of the boundary points to the centroid and σ_d is the standard deviation of $\mathbf{D_b}$. The skewness of the shape is denoted by a negative value, when the spread of the distances is to the left of the centroid and vice versa. For a perfectly symmetric distribution, the skewness is zero.

Kurtosis (γ_2) is a measure of how outlier-prone the boundary distances of the shape to the mean distance to the centroid are. This is computed by measuring the degree of peakedness of the distribution, given by the fourth standardized moment:

$$\gamma_2 = \frac{\Sigma(D_b - D_c)^4}{(\Sigma(D_b - D_c)^2)^2}. \tag{3.2.8}$$

Complexity descriptors

Different shape descriptors [CC00] are formulated from perimeter p_s and area A_s as following:

- Thinness ratio defined as $4\pi\frac{A_s}{p_s^2}$.
- Area to perimeter ratio defined as $\frac{A_s}{p_s}$.
- Rectangularity of the shape given by $\frac{A_s}{A(BB)}$, where $A(BB)$ is the area of the rectangular bounding box of the shape.
- Temperature of the contour as a measure of the smoothness of the contour [DKM86]. The authors extend this thermodynamic formalism to study plane curves and define it as $\left(\log_2\left(\frac{2p_s}{p_s - h_s}\right)\right)^{-1}$, where h_s is the perimeter of the shape convex hull.
- Entropy of the segmented image in the front half plane given by $-\sum(f_p \log_2(f_p))$, where f_p is the histogram count of the pixel values. Thus, the entropy tells if the floor to the front is clear or cluttered with obstacles.

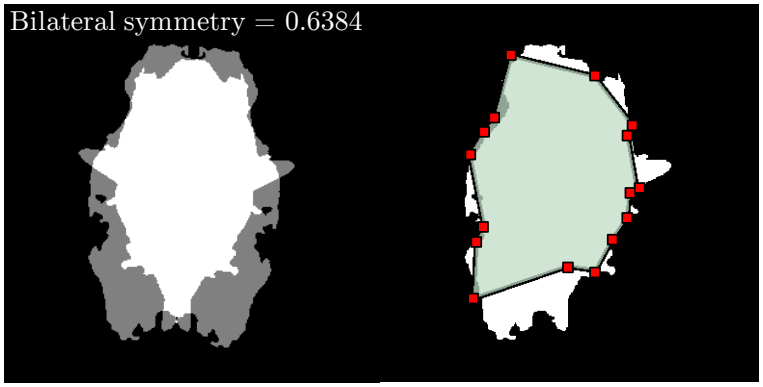
Shape metrics

Bilateral symmetry is a measure of the shape symmetry with respect to its principle major axis. Fig. 3.8(a) shows the bilateral symmetry of segmented corridor image. To compute the symmetry, first the original segmented image is superimposed with its reflected image to generate a grayscale image. If n_f denotes the number of foreground pixels (pixels with value of 1 or 2) and m_o denotes the overlapping pixels (pixels with value of 1), then the bilateral symmetry of the segmented geometry is given by m_o/n.

Norm features: By using n_l landmark points to approximate the shape of the segmented floor, the Euclidean norm $||S||$ is computed as,

$$\|S\| = \sqrt{\sum_{i=1}^{n_l} {S_{x,i}}^2 + \sum_{i=1}^{n_l} {S_{y,i}}^2}. \tag{3.2.9}$$

$\|S\|$ is the *2-n Euclidean norm* of the shape and $(S_{x,i}, S_{y,i})$ are the landmark coordinates. Fig. 3.8(b) shows one instance of the landmark points. Considering an angular resolution of 22.5°, the minimum distance point in each segment is computed as described in section 3.2.1. These extremum points are set as landmarks and the *2-n euclidean norm* thus computes the shape metric for the polygon constructed by connecting the landmark points. Since the coordinates in 3.2.9 are the absolute coordinate values, the measure varies with respect to translation within the image. In order to make this translation invariant, the center of mass of the shape is shifted to the origin $(S_{x,0}, S_{y,0})$ first and then the *2-n*



(a) (b)

Figure 3.8.: (a) Bilateral symmetry of a segmented image of a corridor. Shown are the original and the mirrored image overlaid on one another. (b) Landmark points and the enclosing polygon used for computing shape metrics.

Euclidean norm is computed. Other similar metrics are root mean square size $||S_{RMS}||$, mean size $||S_M||$ and centroid size $||S_C||$ of the shape given by,

$$\|S_{RMS}\| = \sqrt{\frac{\sum_{i=1}^{n_l} \left(S_{x,i} - S_{x,0}\right)^2 + \sum_{i=1}^{n_l} \left({S_{y,i}}^2 - S_{y,0}\right)^2}{2n_l}}, \tag{3.2.10}$$

$$\|S_M\| = \sum_{i=1}^{n_l} \sqrt{\frac{\left(S_{x,i} - S_{x,0}\right)^2 + \left(S_{y,i} - S_{y,0}\right)^2}{n_l}} \text{ and} \tag{3.2.11}$$

$$\|S_C\| = \sqrt{\sum_{i=1}^{n_l} \left(\left(S_{x,i} - S_{x,0}\right)^2 + \left(S_{y,i} - S_{y,0}\right)^2\right)}. \tag{3.2.12}$$

3.3. Feature generation

PCA is a statistical technique predominantly used for dimensionality reduction and feature extraction [Jol02]. In PCA, the higher dimensional data points are projected onto a lower dimensional subspace called as the *principal subspace* such that the diversity of information in the original dataset is preserved. The dimensionality of the data set is thereby reduced where a set of observations in the original data set that are possibly correlated are converted to a set of values in a lower dimensional latent space that are linearly independent. These set of values in lower dimensional subspace are called the *principal components*. The principal components are selected in a way such that the first component explains the largest variance of the original data set, the second component the next largest and so on. Conventional computation of the principal components starts with the assumption of a zero mean data set availability and computing its covariance matrix. The corresponding eigenvector and eigenvalues of the covariance matrix are computed from which the eigenvector with the highest eigenvalue is chosen as the first principal component. Our segmented free floor image denoted by A has a resolution of 310×310 pixels, the covariance matrix of which is of dimension 96100×96100, making it

computationally infeasible to compute the principal components. One easier workaround is to use Singular Value Decomposition (SVD) applied to eigenvalue decomposition of the training images. Eigenvalue decomposition of the image makes a key assumption that the eigenvectors are not linearly independent such that there exists $\mathbf{X}^{-1}$ to express the image $\mathbf{A}$ as,

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}. \tag{3.3.1}$$

$\mathbf{X}$ is a $n \times n$ matrix whose j^{th} column is the eigenvector x_j of the image $\mathbf{A}$ and $\mathbf{\Lambda}$ is a $n \times n$ diagonal matrix with the corresponding eigenvalues λ_j in the diagonal. For a non-square image matrix $(m \times n)$, $\mathbf{X}$ is not invertible for which we compute the SVD given by,

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T.$$

$\mathbf{U}$ is a $m \times m$ and $\mathbf{V}$ is a $n \times n$ orthogonal left and right singular matrices. $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix with non-negative singular values. This implies,

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T.$$

$\mathbf{\Sigma}^T\mathbf{\Sigma}$ being a diagonal matrix now holds the squared singular values σ_j^2. Comparing this decomposition with the eigenvalue decomposition shown in 3.3.1 we can conclude that

$$\mathbf{\Sigma}^T\mathbf{\Sigma} = \mathbf{\Lambda} \text{ and } \mathbf{V} = \mathbf{X}.$$

The singular values of the SVD represent now the square root of the eigenvalues λ. For more details of the derivation refer to the works of [WRR03]. Fig. 3.9 shows an instance of the mean image and the first four eigenvectors (Eigenimages) generated from five segmentation images in a corridor. The training images are generated from a single corridor following navigation behavior demonstration and taking five representative images spanning the length of the demonstration. The eigenimages explain the percentage variation in the training images compared to its mean image. Fig. 3.10(a) shows the percentage of variance explained by the first four components. In this case, just four components are enough to explain 100% of the original training data. For a larger training data set, four components is typically not enough to reflect all the characteristics of the input set. In such cases, more components need to be extracted and correspondingly a component selection is to be carried out to determine the number of relevant components that reflect the task appropriately without running into the curse of dimensionality [Bis06]. One strategy to analyze the number of relevant components is by examining the singular values for the principal components using the so called *scree plot* as shown in Fig. 3.10(b). The plot directly correlates to the relative magnitude of the singular values, the analysis of which provides a direct insight on the number of relevant characteristic components required to approximate the original data. Scree test provides a rule of thumb to determine the number of components to be retained for analysis. The test proposes to stop the analysis at the point where singular value stops descending precipitously. This point normally corresponds to the largest change in slope of the singular values or the percentage explained creating an elbow in the plot. For this example, all the four components deem as relevant. Fig. 3.11 presents a much larger data set generated from fourteen corridor following behavior demonstrations. Five representative images for every demonstration are considered thereby resulting in a total of seventy images. The subset of images contain initial configurations with a large heading error as well the final configurations in

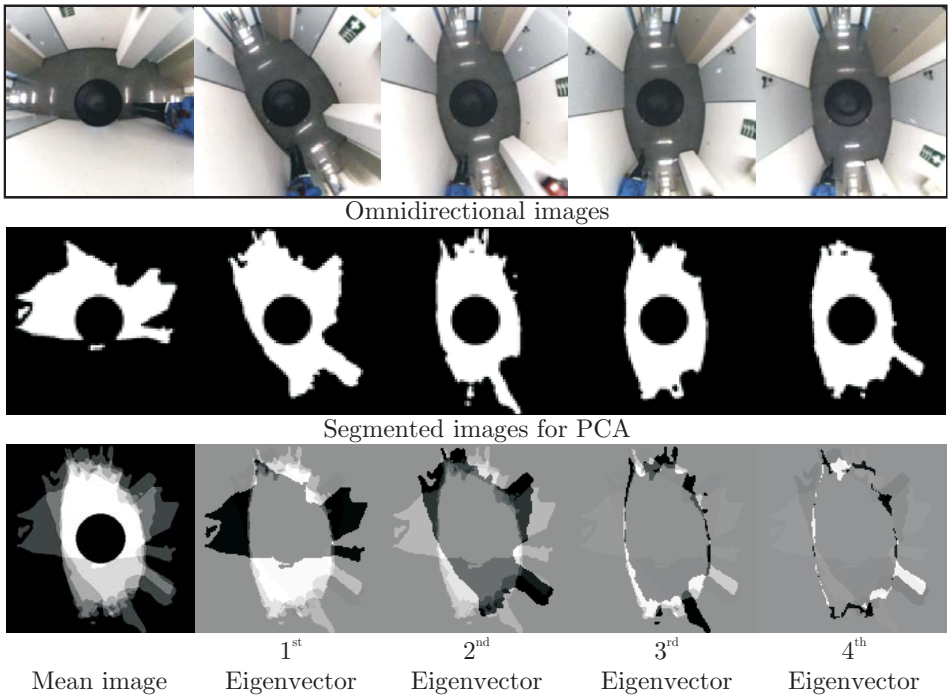


Figure 3.9.: (Top) Five omnidirectional images spanning the length of a single corridor following demonstration; (Middle) the corresponding free-floor segmentation and (Bottom) mean vector along with the four dominant PCA eigenvectors.

which the robot is centered and aligned with the corridor. The corresponding percentage variance and the scree plot for the components are shown in Fig. 3.12. From the plot, it can be observed that the curve tends to flatten out between ten and sixteen components. The scree plot thereby provides only a guideline to the number of appropriate components but not necessarily the most optimal number of components. It is true that by choosing more components, the performance of the PCA based model can be improved but only at the cost of additional and diverse data spanning the entire input space. The trade-off between the number of components and required training data depends on the desired performance metric of the corresponding task.

Principal Component Regression

PCR is an extension of PCA where a regression analysis on the principal components against the output variable is performed to determine the individual component importance. Given a training data set $\{xt_{iC}, yt_i\}_{i=1}^{N}$, where N is the number of input examples, C is the number of data points in one input example, yt_i are the elements in the vector of target outputs $\mathbf{YT}$ for inputs $\mathbf{XT} = \{xt_{i1}, xt_{i2}, \ldots, xt_{iC}\}$; PCA reduces this to m principal components denoted by $\mathbf{Z}_m$. The top principal components are listed starting from the most informative principal component (corresponding to the largest singular value) to the least informative principal component. The approach does not consider the relevance of the individual principal component with respect to the output y. PCR attempts to solve this by regressing the target y on the principal components $\mathbf{Z_m}$. The output of this regression $\hat{yt_i}^{pcr}$ is the sum of the univariate regressions [HTF01],

$$\hat{yt_i}^{pcr} = r_0 + \sum_{j=1}^{m} r_j Z_{ij} + \varepsilon_i,\ i = 1, 2 \ldots N \tag{3.3.2}$$

where, $r_{0,1\ldots m}$ are the regression parameters and ε_i is the error term which captures all

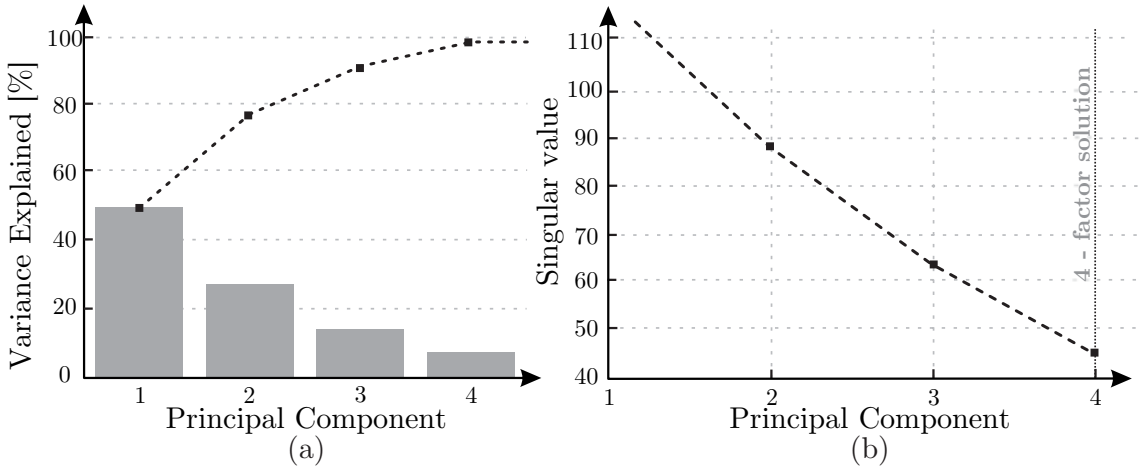


Figure 3.10.: (a) Percentage variance explained by the individual principal components and (b) Scree plot of the singular values for the first 4 components.

other influences on the output not interpreted by the regression coefficients. Z_{ij} are the principal components projections of the training data. This method goes by the assumption that low variance principal component may also be important in defining the behavioral action of a robot. Fig. 3.13 shows the correlation coefficient of the principal components against the target variable. The target variable in our example is the curvature traversed by the robot expressed as a ratio of rotational to translational velocity. The figure shows that the top correlating components are not necessarily the components that reflect high variance in the data. The components that are collinear or redundant are eliminated using PCR thereby retaining only the relevant components. This raises the obvious question, *what are the number of relevant components and which are them?*

3.4. Feature selection

Feature selection selects a subset of characteristic relevant features and eliminates the rest from building a model. Feature selection (also known as *input variable selection*, *subset selection*, *pattern discovery*) aims to address three aspects to model building: improve the prediction accuracy of the model, provide a better understanding of the underlying working principle of the model in that influence of different factors to the model are comprehensible and to have cost- and computationally efficient task predictors. Feature selection is broadly classified into three types of algorithms namely wrapper, embedded and filter algorithms [KJ97]. Wrapper approach takes the feature selection problem to an optimization of the model performance. This requires searching through all the possible combinations of features to select the one which possesses the best generalization capability to data that are not seen by the model during training. Embedded approach is directly incorporated in the model where the weights of the model parameters are modified for optimal generalization performance. Standard shrinkage methods for regression algorithms and weight decay for neural network models [HTF01] are typical examples of embedded feature selection approach. Both, wrapper and embedded approach are model based or in other words the algorithm to build the model is already known. Filter based method on the other hand, is a model free feature selection which utilizes statistical inference of the

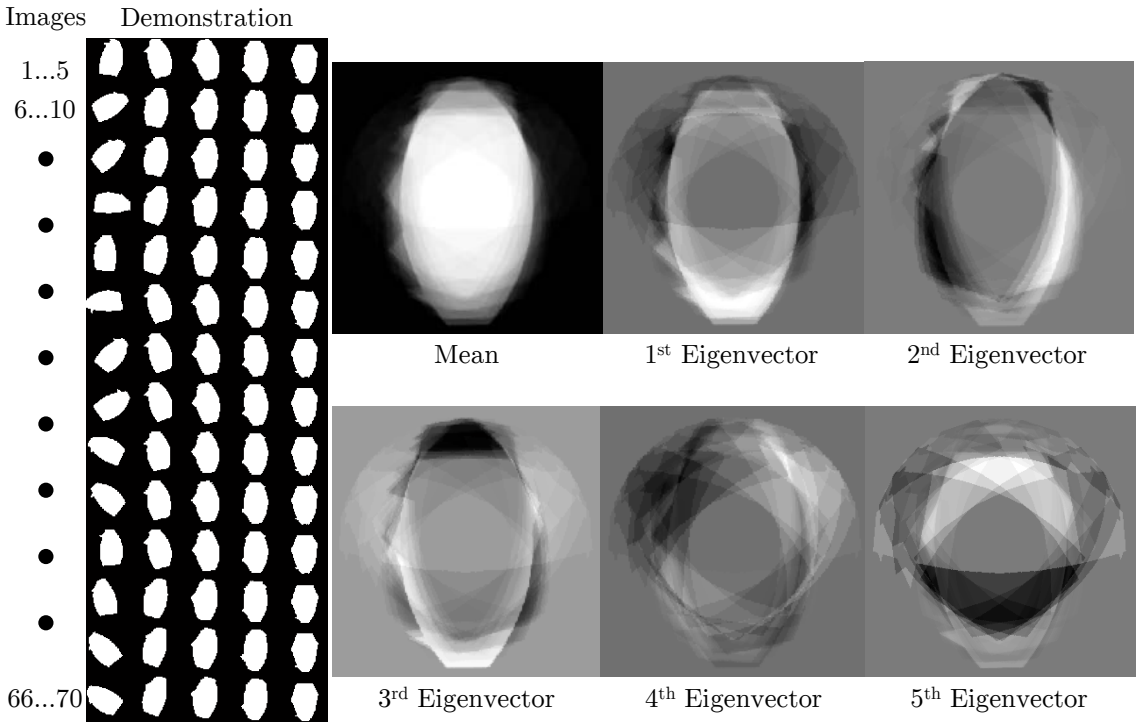


Figure 3.11.: Five images per demonstration spanning the entire length for fourteen demonstrations. Thus, a total of seventy training images are used for computing the principal components. The mean vector along with the five dominant PCA eigenvectors are shown.

input data to select the most relevant features. This is typically done either by checking the correlation of the input variables to the target or by minimizing the information entropy or using mutual information [KJ97]; [RGH11]. The influence of the input feature set is judged using a metric of performance which reflects the measure of fit of the model.

3.4.1. Performance metric

Given N training examples with j input features XT_j and target yt, the most common performance measure is the Mean Squared Error (MSE) given by,

$$\text{MSE} = \frac{\sum_{i=1}^{N} \left(yt_i - \hat{yt}_i\right)^2}{N} \tag{3.4.1}$$

where, $\hat{\mathbf{YT}}$ is the predicted output of the model. In spite of the simplicity and intuition of MSE, it still carries a few imperfections. MSE is independent of temporal and spatial relationships between the original and the predicted data-set such that reordering the data does not change the MSE [WB09]. The interpretation of MSE to different scaling of data also throws in some challenges. One way to alleviate this is by computing the Normalized Mean Squared Error (NMSE) between the prediction and the actual target. NMSE interprets the residual variance of the predicted output by dividing the MSE by the variance $({\sigma_{yt}}^2)$ of the target given by,

$$\text{NMSE} = \frac{\text{MSE}}{{\sigma_{yt}}^2}. \tag{3.4.2}$$

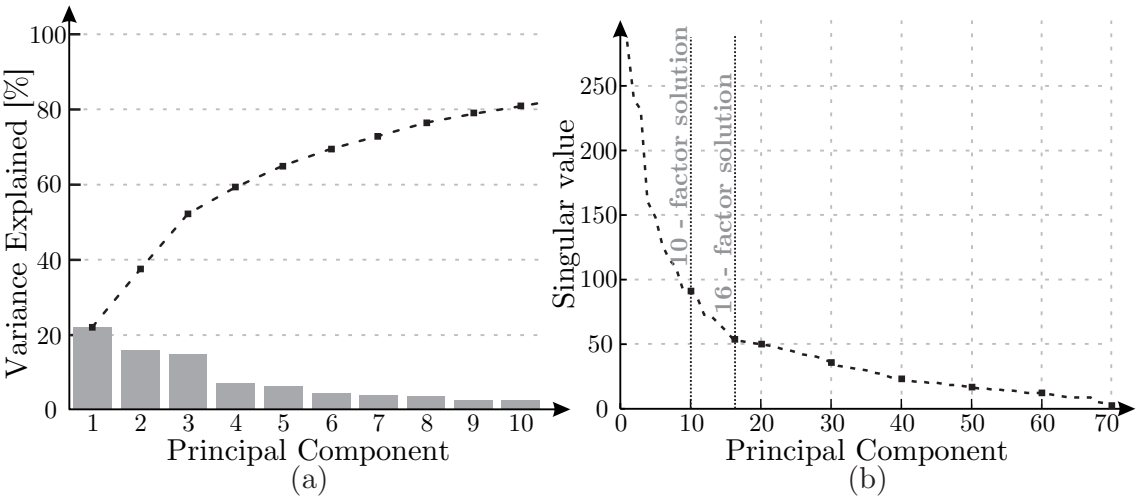


Figure 3.12.: (a) Percentage variance explained by the individual principal components. Only 95% of the cumulative distribution is shown and (b) scree plot of the singular values for the first 70 components.

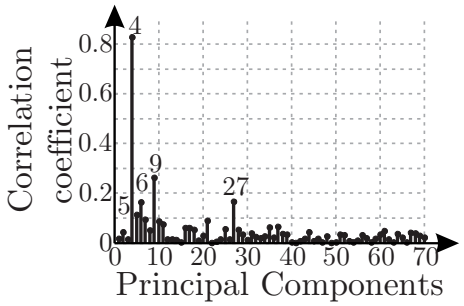


Figure 3.13.: Correlation coefficients of principal components against the behavior outputs.

3.4.2. Cross-validation

Cross-validation is the statistical method to test the generalization ability of a model. It is done by partitioning the data set into two subsets, a training set and a test set to validate the performance of the model. The most common cross-validation technique is the *k-fold cross-validation* where the available data set is split into k roughly equally sized parts first. The cross-validation is then carried out k times where for each iteration $k-1$ data groups are used to train a model and then evaluated on the remaining one. The final cross-validation error is then computed by averaging the generalization error registered at every fold or iteration. A simpler version of this technique is the *hold-out* cross-validation where the data set is split into only two sets namely one for training and the other for testing. In spite of the simplicity and ease in performing *hold-out* cross-validation, it is ambiguous if the validation set is small. Hence, it is preferable to perform both the cross-validation methods for arriving at a quantitative conclusion about the model. Another approach used typically when data is plentiful is to split the data set into three sets, where the first data set is used as *training set* and is validated against the second set called the *validation set*. Based on the model performance on the validation set, the complexity model parameters are optimized until a satisfactory generalization error on the validation set is obtained. The final performance is then determined by testing the model against the third data set called the *testing set*. While performing a feature selection routine, every feature subset based model need to be thoroughly cross-validated before being selected.

3.4.3. Wrapper approach

Wrapper approach is used to select the relevant subset of features from the input training data set that improves the overall generalization performance of the model. The subset selection algorithm exists as a wrapper around the machine learning algorithm, hence the name [KJ97]. In this approach, the precision and accuracy of every feature subset is checked on the induction algorithm using the different performance metrics mentioned before. The method then searches for the most practical set for which the accuracy is the highest. The generalization accuracy of the different subsets is computed with either a *k-fold* or a *hold-out cross-validation* on the test set. There are two selection techniques to arrive at the feature subset namely, forward chaining and backward chaining. Forward chaining counters the selection procedure by starting with an empty set of features and start adding features to the subset until the generalization error no more improves. Backward chaining or elimination starts with the full set of features and iteratively eliminates the least useful feature. Forward chaining enjoys the advantage that it is computationally efficient as it is fast to build models with fewer number of features whereas backward chaining captures the interactive nature of the individual features but still suffers with the computational inefficiency [KJ97]. Fig. 3.14 shows the evolution of the MSE and NMSE on unseen test set output for the example dataset presented in Fig. 3.11. The top seventy principal components are extracted from the entire demonstration data to obtain a training data-set of dimension 2100×70. Using forward chaining the top principal components for PCR that best generalize the unseen test-data is identified. Cross-validation is performed by both hold-out cross-validation where 60% of the data set is used for training and the remaining 40% is used for testing and a *5-fold* cross-validation.

3.5. Application example

We illustrate the use of feature selection for identifying the relevant principal components for modeling a visual navigation behavior using the corridor following example presented in Fig. 3.11. Fourteen demonstrations each of length 150 sample units are generated. For every demonstration examples, the top 70 principal components are extracted thereby generating a total of 2100×70 training examples. The demonstrated robot action is transcoded into path curvatures in terms of the ratio between rotational and translational velocity. Fig. 3.15 shows a pictogram of some of the demonstrated corridor following behavior. Wrapper approach with forward chaining is performed on the demonstration examples to find the optimal subset that minimizes the generalization error. Fig. 3.14 shows the evolution of the generalization error with increasing number of features. The generalization errors: MSE and NMSE are plotted for both *hold-out* and *5-fold cross-validation*. *Hold-out cross-validation* is done by training on 60% and testing on the remaining 40% of the data. Conventionally in machine learning, cross-validation is performed by randomly selecting the training and test examples. In our case, the examples are related trajectories of a behavior execution, hence the training is performed on 60% of the fourteen trajectories (8 trajectories) and testing done on the remaining 40% (6 trajectories). For a *5-fold cross-validation*, each fold has 3 trajectories, thereby training is done on 11 trajectories and tested on the remaining 3 trajectories for every fold. By setting a feasible tolerance on the allowable increase in the generalization

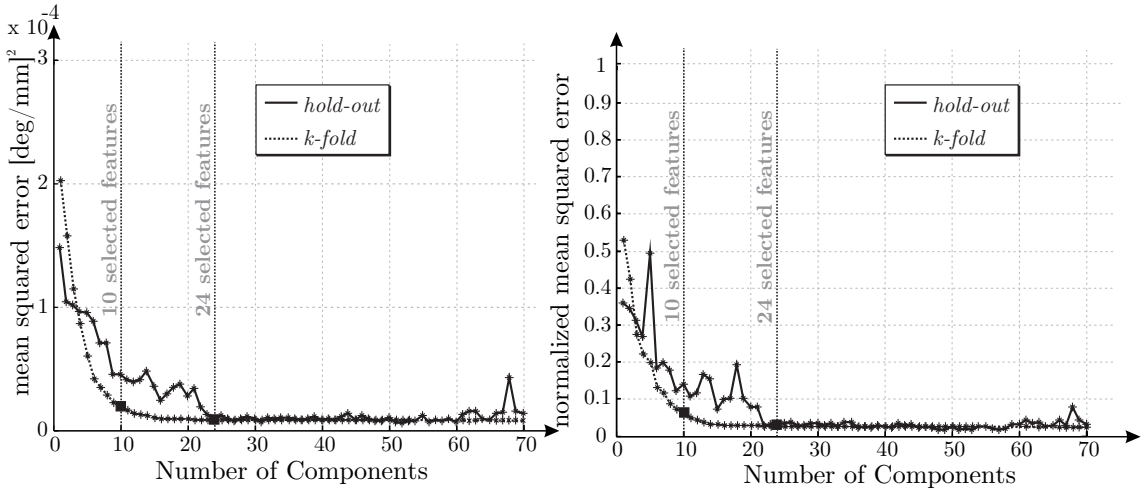


Figure 3.14.: Forward chaining error evolution for PCR with *hold-out* and *k-fold* cross-validation. (Left) Mean squared error and (Right) normalized mean squared error. The cross-validation of the two subsets of features are shown in Table 3.2.

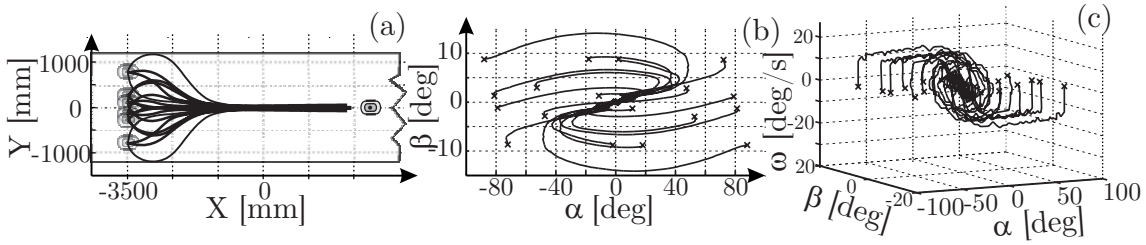


Figure 3.15.: Pictographic representation of corridor following behavior demonstration.

error between two subsequent subspace to prevent a local optimum, one ends up with two different subsets. From the plot, one can observe that with *5-fold cross-validation* the error does not show any significant improvement by adding more than 10 features, whereas for *hold-out cross-validation*, the error is rather fluctuating and flattens out with 24 features. The figure also shows the pros of *k-fold* cross-validation against *hold-out* in having a more accurate and consistent tend in generalization error. The bobbing error from *hold-out cross-validation* points at the drawback in that the error heavily relies on the kind of split performed for selecting the training and testing trajectories. Table 3.1 shows the selected features generated from hold-out and 5-fold cross-validation. Table 3.2 tabulates the cross-validation MSE and NMSE for the two subsets of features. Please note that the values in Table 3.2 only partially correspond to the error values from Fig. 3.14. During forward chaining, we cross-validate on seventy components using *hold-out* and *k-fold* to arrive at 24 and 10 relevant components. Table 3.2 shows the generalization error of cross-validating the two selected subsets using *hold-out* and *5-fold*.
Fig. 3.16 shows the performance of PCR for both the subsets on the target κ for selected

Table 3.1.: Selected features

Cross-validation	Component number
hold-out	4,12,7,25,54,8,42,6,49,19,21,60,31,10,2,67,34,68,58,20,9,11,35,23
5-fold	4,8,6,2,9,11,5,7,30,10

Table 3.2.: Cross-validation error for the selected set of features

Cross-validation	10 feature set		24 feature set	
	MSE $\times 10^{-5}[\text{deg/mm}]^2$	NMSE	MSE $\times 10^{-5}[\text{deg/mm}]^2$	NMSE
hold-out	1.690	0.040	0.867	0.0268
5-fold	1.901	0.0614	5.464	0.047

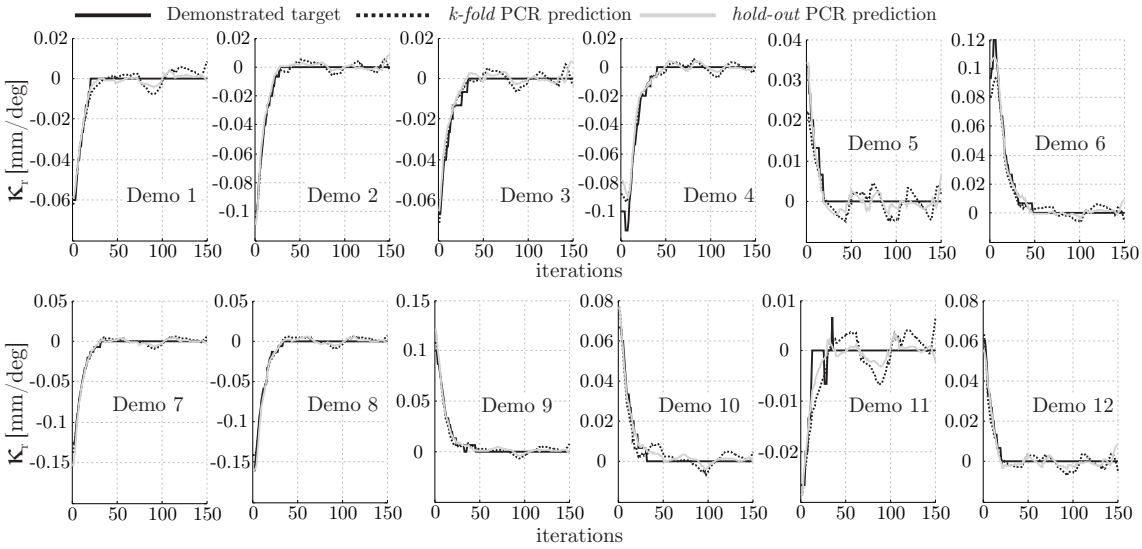


Figure 3.16.: PCR prediction with the different feature subset for all the demonstration trajectories. *k-fold* and *hold-out* predictions are performed using 10 and 24 components respectively.

demonstration trajectories. The figure shows that both the subsets approximates the target robot curvature pretty good for all the demonstrations. Interestingly, regression of 10 selected principal components performs equally good to regression over 24 components. This validates the terminating condition of forward chaining and the reasoning behind that conclusion (Fig. 3.14). Hence, for this application example of learning visual corridor following behavior, ten principal components shown in Table 3.1 deem as characteristic features to model the behavior.

3.6. Related work

The potential of using vision for mobile robot navigation has grown leaps and bounds over the last decade [BFOO08]; [DK02]. The advent of faster computational capability has allowed the exploration of the realms of image understanding using multitude of features to capture various associated aspects such as motion detection [Cam95] and objects recognition [Low99]. Image features such as Scale Invariant Feature Transforms (SIFT) [Low99], Speeded Up Robust Features (SURF) [BTVG06] and more recently Histogram of Oriented Gradients (HOG) [DT05] are popular local descriptors that extracts interesting cue points about the environment. By computing these local descriptors on a single reference image, they can be queried visually with any other target image for recognition. The authors of [SLL01] use a trinocular stereo system on whose images SIFT features are extracted and matched to identify robust 3D landmarks. The landmarks are then further

used for generating a 3D map of the environment. In [CC13], a visual path following using a monocular camera is performed by matching the current robot perception with the key images of the environment used earlier. They use visual keypoint based localization shown by [RLDL07], who match Harris corners detected between the query and target image. With the visual path following in the upper layer, a laser based collision avoidance works as the primitive navigational behavior. These approaches require a domain knowledge of the environment in that the key images are extracted first during the training phase and are matched during runtime for localizing the position and the underlying robotic behaviors are based proximity sensors. Our method attempts to learn a purely vision based robotic behavior without any a priori information about the environment. Other similar feature matching based navigation schemes have been proposed by [JAC97]; [RC01].
The works of [Pom89]; [Kni02] show different approaches to achieve purely vision based navigation. The famous ALVINN land vehicle of [Pom89] predicts the heading of the vehicle to follow the road based on a list of local and global road features, whereas [Kni02] uses stereo vision for robot navigation. In our case, the robot must learn different robotic behaviors whose characteristics depend strongly on the environment it traverses. Hence both the geometrical and textural characteristics together with features that are inherent to robot navigation need to be extracted. Global descriptors that analyze the shape and the contour [ZL01]; [Fre10] of the segmented patch need to be extracted to acquire further knowledge about the image. Shape and contour analysis has been historically used for diverse areas of research involving optics ranging from medical imaging [NS02] to biometric gender classification [DW11]. All these features require expert inputs from the programmer and draw conclusions on its relevance. PCA presents an insight into the latent feature space of the image which is generally overseen by the human. In [VBK01], panoramic omnidirectional image is projected onto its principal components to extract features for localization. The works of [AMW05] use Hidden Markov Models to extract indoor office features to identify scenarios associated. PCR in the context of environment modeling is used by [VK99]. The principal components of a supervised set of robot positions are regressed against high dimensional sensor measurements to perform robot localization.

3.7. Summary

This chapter presented the methodology to tackle the problem of generating and aggregating meaningful representative visual behavior features to map relevant relationships between the perception and action. The perception is a fusion of 3D PMD and a 2D omnidirectional camera system mounted on a robot. Given a set of diverse training examples of a behavior execution, the perception is first segmented for floor and non-floor region. A mixture of experts segmentation scheme fusing heterogeneous classifier outputs trained on multiple 2D segmentation algorithms is utilized to obtain a segmentation image of floor and non-floor pixels. Using standard image processing algorithm together with manual intervention, several features that express a range of aspects such as relative proximity and location of obstacles with respect to the robot, geometry and shape characteristics of the floor through image moments and computational shape analysis are extracted. A second approach explores the latent space of the recorded demonstration

images to determine if there are hidden variations that can be mapped to the demonstrated behavior output. Using PCA, the eigenvectors of the different training images are identified and are listed down in decreasing order of the data variance explained in its eigenspace. Conventionally, the top principal components are directly selected as the most relevant features for the data presented. From a machine learning perspective, they fail to reflect the influence of the different individual components to the presented target output. We address the problem, where the recorded target robot action is also considered into the training process by regressing the extracted principal components to it. All the principal components irrespective of their order of variance explained are checked for relevance. Once the feature extraction is done, the most characteristic subset for mapping is selected through a feature selection algorithm. Using a forward chaining wrapper approach, the list of relevant features are increased sequentially by carefully observing their cross-validation generalization error against both seen and unseen examples. This aspect is examined in detail through an application example for selecting the most relevant subset of principal components for an indoor corridor following behavior. It was shown that based on the chosen mode of cross-validation and the desired performance metric, different sets of characteristic features are obtained. The generalization error of the different subsets to unseen data set was tabulated. This chapter outlined the different approaches to breakdown enormous amount of image information into representative visual cues that could describe the demonstration. The choice of the number of features for a descriptive subset can become highly subjective depending on the mode of feature selection and more importantly on the learning algorithm used to model these subsets. In other words, a feature that presents itself as highly relevant working with a certain modeling algorithm might completely fall out of contention when dealing with a different learning algorithm. Hence, a careful selection of the subset with pretext to the behavior modeling technique is imperative.

4

Teaching modalities for mobile robot demonstrations

The object of teaching is to enable the student to get along without a teacher.
– Elbert Hubbard

Personal service robots capable of performing multiple tasks being accessible to the larger population is becoming a realistic vision. Mobile robotics here is one such important sub genre. The mobile robots available today in the market and also in research rely mainly on manual programming to decipher the perception-action mapping. The programmer is typically an expert who requires a profound understanding of the task and its interaction within the confines of the environment. For a vision based system, this corresponds to mapping images to actions. Images are broken down to multiple features which can reflect the different changes within the environment [War06]. A major hurdle is the difference in the perception modality between the human and the robot. LfD [ACVB09] solves this challenge where human based demonstrations are observed and generalized to a skill. Machine learning algorithms are used to identify patterns and learn the ulterior perception-action mapping exhibited by the human. The approach does not restrict the robot instruction to only experts, but also allows non-experts to interact with the robot in an intuitive manner. The bottleneck of this process lies in the fidelity of the data recorded from demonstration of the teacher. Mostly, therein lies a correlation between the quality of the data and the generalization of the task learned. Data quality irrespective of the learning method used is of ultimate importance. Cross-validation, feature and data selection are some of the methods used to compensate this deficiency. In the context of LfD, this amounts to the quality of observing or interpreting the human demonstrations. One of the ways to assure good data interpretation is to ensure efficient and comfortable demonstration modes to record the perception-action pair. In this chapter, three interfaces for demonstrating mobile robot navigation skills are investigated [NPHB12]. The pros and cons of a demonstration mode, its complexity and the user friendliness to both experts and non-experts are surveyed. Behavioral tasks such as line following, corridor centering, obstacle avoidance, door passing and homing are chosen for evaluating the performance of the demonstrations. Performance evaluation metrics include the average time and number of commands taken for a demonstration together with the smoothness of the recorded actions across multiple demonstrations.

4.1. Mapping problem

Flexible handling of industrial manipulators have been the driving force behind the development of intuitive user interfaces over the years. Augmenting from textual programming, teach-pendants are hand-held control/programming units which allow to move and teach the robot. The robot is manually sent to goal position through a set of desired intermediate positions which are recorded and reproduced in the same way during imitation. The desired motion is specific to the workpiece under observation and lacks generalization. Teaching attributed in LfD, attempts to teach the essence of the task with regards to its sensor perception and executed action. Conventional teaching method is the kinesthetic mode where the human directly moves the arm of a manipulator or a humanoid robot and performs an example demonstration of the task [Cal09]; [INS02]. The sensor measurements and the joint positions are recorded during the demonstration. Adept transfer of information between the teacher and the robot depends in matching the sensing and interpretation of the environment by both. This is called as the correspondence or the mapping problem [DN02]. The authors in [ACVB09], partition the correspondences within LfD into two types namely, record and embodiment mapping which are drawn according to the differences in the perception of the states experienced by the teacher and the robot. Fig. 4.1 shows the intersections of record and embodiment mappings. Record mapping refers to the mapping between the perception of the teacher during demonstration and the data recorded. Embodiment mapping on the other hand refers to the mapping between the recorded data to a format understandable and executable by the learner. The former is generally associated with the demonstration phase while generating data and the latter performed during model building and imitation phase. For a mobile robot, an equivalent kinesthetic teaching is to drive the robot manually similar to an automobile. [ACG10] present a similar idea of a robot driving interface for children. The scalability of this design for all users in an indoor environment is a big challenge. In this chapter, we address this challenge by presenting different pure vision based robot command interfaces. The following conditions are carefully considered before their design, namely

1. establishing the perceptual ability of the task to capture task relevant information,
2. design of an intuitive interface for the teacher and
3. user friendliness of the interface.

Three modes of demonstration interfaces are discussed namely: joystick control, gesture based command and remote teleoperation with a steering wheel. In joystick and gesture based control, the human leads from behind and walks the robot through the environment. The commanded turn rate is based on the humans own vision. With the steering wheel control, the teacher controls the velocities of the robot remotely by perceiving the environment through robots vision. With the first two approaches, the instructed human actions might depend on perceptual information that is seen by the human and not by the robots camera. This ambiguity of mapping is avoided in the third mode, where the robots view functions as the only perceptual input to instruct. On the contrary to the advantage offered, the live streaming of robot camera lacks depth information making it more difficult for the human to estimate distances and velocities. The robot in our case

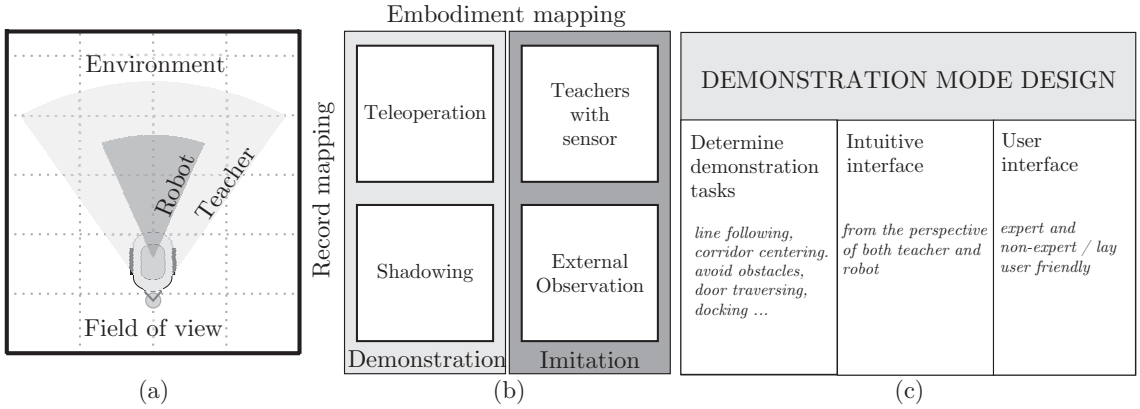


Figure 4.1.: The mapping problem. (a) A coarse example of the difference in the perception and field of view of the teacher and robot, (b) representation of the different record and embodiment mapping together with their expression in demonstration and imitation phase [ACVB09] and (c) three building blocks for a sound design of a robot demonstration mode.

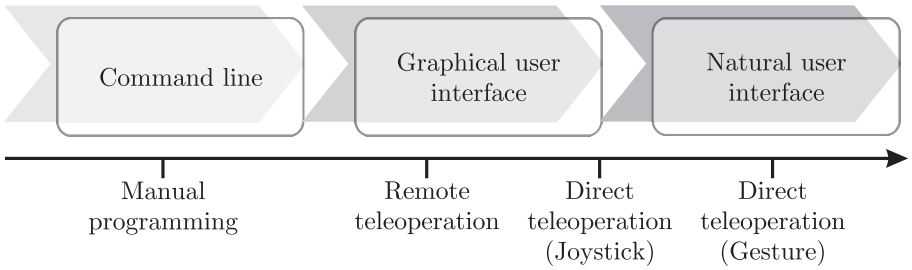


Figure 4.2.: The different user interfaces for human-machine interaction.

is equipped with an omnidirectional camera. The human is provided with two visualizations of the environment: the original omnidirectional image of the camera or a mapped birds-eye view. The aim of this chapter is to analyze the teaching modes which enable a non-expert user to guide the robot through the environment with little or no training. We study this through the above mentioned three modes, starting from command line interface through graphical user interface to natural user interface. The distinction of the three interfaces is shown in Fig. 4.2.

4.2. Teaching modalities and Human-Machine interfaces

The experimental setup consists of a Pioneer 3DX mobile robot mounted with an omnidirectional camera and a 3D Kinect camera facing the rear side. The Kinect offers an advantage against the PMD camera in that the resolution of the acquired depth image allowing robust tracking of the teacher. As for the perceptual sensor, the difference between the sensor modality of human and the robot is mainly the field-of-view. An omnidirectional camera with a 360° horizontal field-of-view can close this gap and is thus used to reduce this difference. The availability of depth information from the Kinect allows it to track skeletons thus can be extended to track the teachers gestures. In essence, all the three demonstration modes are different forms of teleoperation. The modes and the mobile robot with the cameras used for the demonstration is shown in Fig. 4.3.

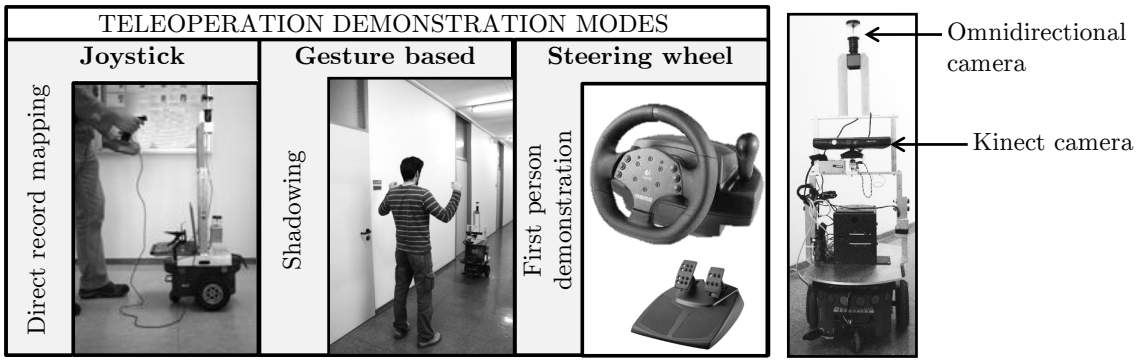


Figure 4.3.: Overview of the different teleoperation based demonstration modes and the mobile robot experimental setup.

4.2.1. Remote teleoperation

An automobile accessory set comprising a steering wheel together with an accelerator and break pedal is used to control the robot remotely by perceiving the environment through an omnidirectional camera. The catadioptric image is generally conceived inconvenient for humans used to perspective projection. One option would be to perform the demonstration with a monocular camera. Nevertheless, monocular cameras have a very limited field of view thereby rendering them less appropriate to demonstration navigation behaviors. This being said, the lack of depth perception in the omnidirectional image accentuates the uncertainty in executing the navigation decisions. To help the user with a sense of robots proximity to nearby obstacles, the sonar readings from the robot are also displayed. Additionally, a birds-eye view image obtained through radial correction around the image center [WGLSv00] is also provided as an alternative vantage point. The birds-eye view is obtained from the omnidirectional image corrected for distortion and then unwrapped. The geometric consistency between the image and the real world eases the task (for example, corridors appear as image bands of constant width). The GUI allows the user to switch between omnidirectional and birds-eye view at any time. The heading is controlled by turning the steering wheel with the translational velocity set by pushing the accelerator pedal. A 60° steering angle is transformed to maximum rotational velocity and a minimum of 5° steering angle is required to start turning. A sonar based stop behavior works as a safety measure when the robot gets too close to obstacles. In the case of imminent collision this behavior overrides the user commanded velocity. Information exchange between the teacher and robot is established using TCP/IP protocol with a client-server communication. In the event of communication loss, the robot stops. Fig. 4.4 shows the technical setup of the steering wheel demonstration mode.

4.2.2. Joystick control

Joystick control provides a direct interface to the robot. It provides an identity record mapping between the teacher and robot. The teacher passively controls while the robot records the data. We use an USB connected Logitech wingman joystick with a throttle trigger to indicate start and stop of demonstration activity. Since no fixed translational speed or turn rate can be set, both need to be explicitly controlled by the teacher. As an alternative, one can also use the joystick only for controlling the turn rate whereas

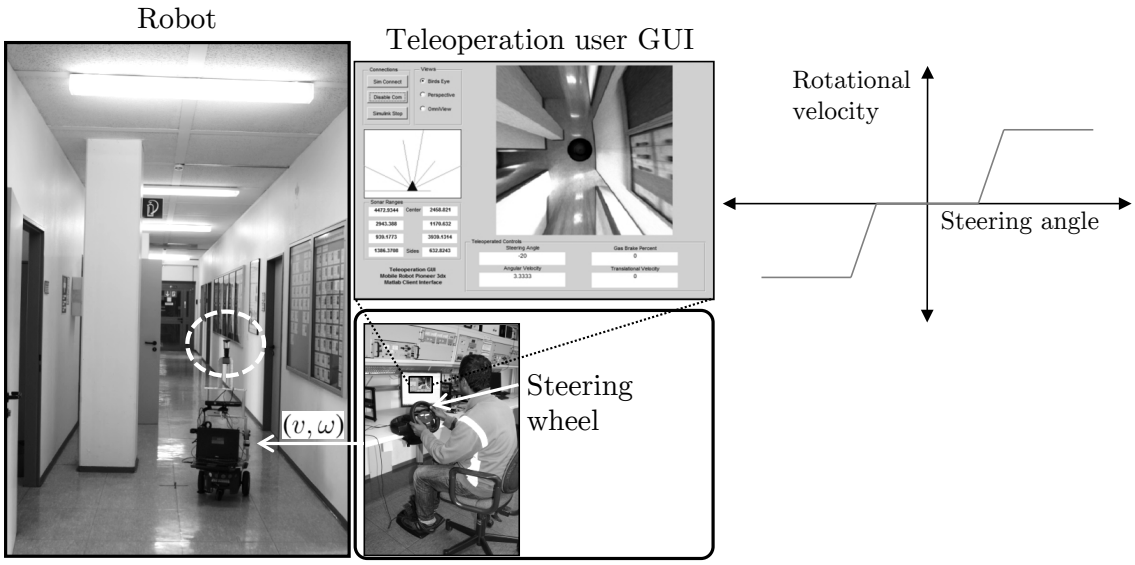


Figure 4.4.: Steering wheel based teleoperation.

the translational speed is set via a simple sonar based stop behavior. To maintain the flexibility of this mode to be used with different joysticks and hand held controllers, a direct transfer of the command from the joystick to the robot is performed. Thus the controller commands are unfiltered and react directly on the robot motors.

4.2.3. Gesture based control

Gesture based control interprets the teachers instructional gestures as motion commands. The method exploits the advantages of a Natural User Interface (NUI) by using body gestures for instructions and thereby eliminating intermediate control devices. NUI allows even a non-expert user to intuitively interact with the robot. Fig. 4.5 shows the working principle of the designed gesture based interface. The Kinect camera is integrated such that it looks to the rear of the robot. The teacher demonstrates from behind using gestures similar to driving a hypothetical automobile steering wheel. Open source framework for natural interfaces `openni-tracker` [Ope]; [SSKFFBCM13] is used within a `ROS` [QCGFFLWN09] framework to track the skeletal joints (Head, neck, torso, left and right hand) of the teacher. The obtained Cartesian coordinates $(x_{joint}, y_{joint}, z_{joint})$ of the joints are transformed to its corresponding image coordinates (u_{joint}, v_{joint}) using the Kinect camera model. The camera gaze is controlled to place the teacher always directly in the center of its field of view. A desired region of interest along the center of the image is defined whose deviation is aptly corrected through camera gaze. The gaze is controlled using a pan servo motor over which the Kinect camera rests. The horizontal pixel error of both the hands (e_{left} and e_{right}) beyond the demarcation zone is determined.

$$e_{left} = \begin{cases} du_{left} - u_{left}, & \text{if } u_{left} < du_{left} | u_{left} > du_{right} \\ 0, & \text{otherwise} \end{cases}$$

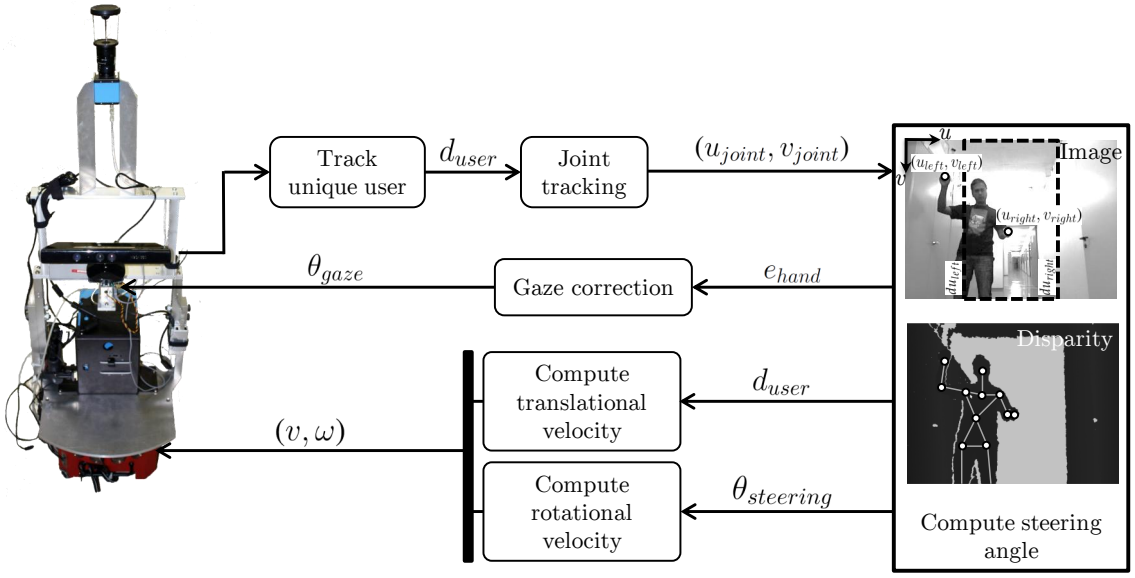


Figure 4.5.: Architecture of gesture based demonstration

Figure 4.6.: Pan and tilt correction. (a) Camera gaze correction with a pan servo motor in action during a door traversing maneuver and (b) user image before and after tilt adjustment done during calibration.

$$e_{right} = \begin{cases} du_{right} - u_{right}, & \text{if } u_{right} > du_{right} | u_{right} < du_{left} \\ 0, & \text{otherwise} \end{cases}$$

$$e_{hand} = \frac{e_{left} + e_{right}}{2} \tag{4.2.1}$$

The error of the left and right hand (e_{hand}) beyond the demarcation of the region of interest (du_{left},du_{right}) is used to control the gaze. An image based visual servoing scheme is performed to minimize the average of the right and left hand deviations (e_{hand}) to zero. Fig. 4.6 (a) shows this active during a trial to demonstrate a door passing behavior from a corridor.

To trigger forward motion of the robot, the teacher starts moving closer to the robot at normal walking speed. On recognizing the teacher, the initial distance between the robot and teacher (d_{user}) is set as reference and translational velocity is controlled to maintain this distance. A proportional feedback control loop where the error between the current user distance and the reference user distance is fed as the control variable. The controller determines the change in velocity to the current translational velocity to

Figure 4.7.: *psi* pose and the steering angle poses for left and right turn

adapt to the moving user. Experiments show the optimal operating range for robust tracking and control for the Kinect is between $1.5 - 3.5$ meters. The robot stops when the distance is too close (below 30% of the control distance) or when the tracking loses the user. The steering angle of the hypothetical steering wheel rotation gesture is transformed to rotational velocity. The angle is computed between the up-down line connecting the head, neck and torso and the horizontal line connecting the left and right hand. The user needs a minimum steering angle to trigger the robot turn rate. Typically, set to 5° it avoids the robot to be over sensitive to small unintentional changes in the steering angle. Similarly, the steering angle also has an upper bound. Angles greater than this upper bound are correspondingly interpreted as maximum rotational velocity. A running average of the last five instructions determines the final rotational velocity. The initial calibration performed through a *psi-pose* [Ope] serves as the reference pose relative to which the steering angles are determined. Fig. 4.7 shows the initial pose and the relative steering angle gestures for a left and right turn. During the initial calibration, the tilt of the Kinect is also calibrated to have the optimal field of view of the body skeleton of the user. This is achieved by computing the ratio of the teachers body or the teachers height during calibration. By setting a desired ratio of the users height in the cameras field of view, the tilt angle is set to have maximum coverage of the users skeleton. From the distance of the teacher and the ratio of teachers skeleton in the field-of-view of the camera, the tilt is computed through triangulation. Fig. 4.6 (b) shows the calibration of the user before and after tilt correction. The sonar sensor readings delivered are merged and used for a base-line safety mechanism by reducing the speed when getting closer to near by obstacles. This is to prevent any deliberate or accidental attempt to collide. From the distance d_o to the nearest obstacle along the current path curvature of the robot, an equivalent time to collision t_c is computed. The current velocity (v_c) is not subsumed until the time to collision gets less than $t_s = 2$ seconds (see (4.2.2)).

$$v = \min \left\{ v_c, \frac{d_o}{t_s} \right\}. \tag{4.2.2}$$

4.3. Demonstration tasks

The different demonstration modalities are compared and tested for precision and the level of difficulty in executing a demonstration. Behavioral tasks namely, driving the robot along straight and curved line, avoiding an isolated obstacle, passing a door from

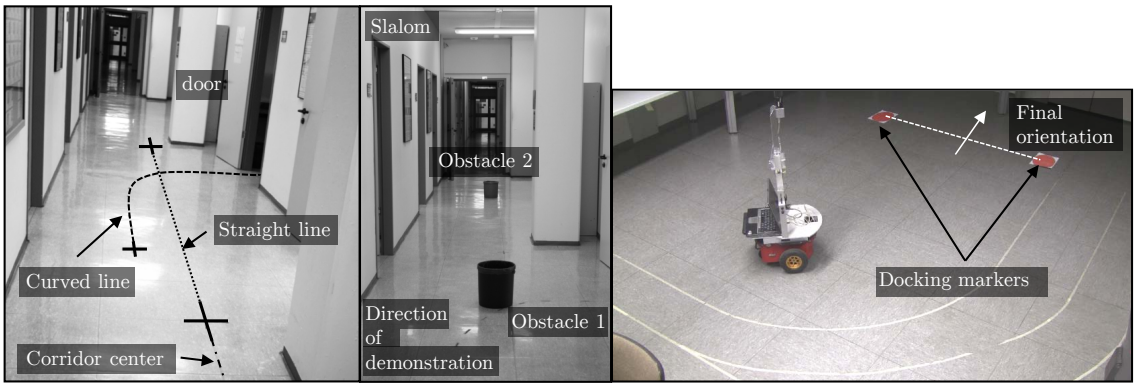


Figure 4.8.: Overview of the scenarios used for line following (straight and curved line), navigational behaviors (corridor following, obstacle avoidance, slalom, door passing and homing).

different angles of approach, guiding the robot to a docking point and a slalom motion are experimented. The experiments are designed to cover a variety of robot maneuvers. The trials start with line following experiments, where the start and the end point of the trial is predetermined. This is followed by behavior executions, where the aim is to demonstrate a motion policy and not a pre-determined trajectory. Corridor following demonstrations start from different misaligned robot configurations with the aim to drive the robot to the center of the corridor. For obstacle avoidance, distinct obstacles are placed in front of the robot to be evaded. The goal of door passing behavior is to guide the robot through a door such that the robot passes between the door sills perpendicularly. Slalom behavior fuses the obstacle avoidance and corridor following together. Two obstacles are placed along the center of the corridor. The robot, also positioned along the corridor center is to be guided from one end of the obstacle course to the other end of the obstacle course. The goal of the task is to maintain the robot orientation along the corridor center and divert the straight line motion only on encountering obstacles. Goal point reaching or homing is performed by guiding the robot to a specified target area indicated either by two red or green colored markers placed one meter apart. The goal point is the center point of the line connecting the two markers. Fig. 4.8 shows the scenarios associated with the mentioned experiments.

4.3.1. Experimental setup

A user study with fifteen participants with varying associations with robotics is carried out by asking them to perform the above mentioned tasks. An introduction of the task together with an example task demonstration is performed by an expert and each participant is allowed ten minutes to get acquainted with the three modes. Failure in finishing a task is also recorded and with an opportunity try again upon request and curiosity. A cautious preview of the test conditions such as day-light, number of doors open or frequency of people moving in the corridor are done by the expert to maintain data consistency across all users.

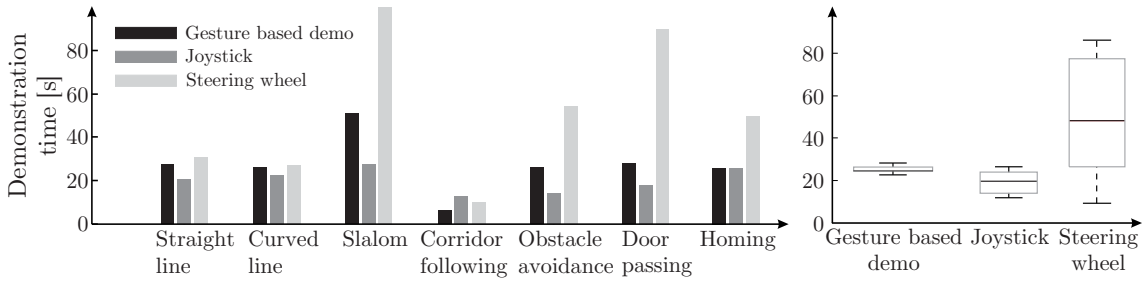


Figure 4.9.: Bar and box plot of the average demonstration times of the three modes across all scenarios. The box plot shows the median, 25 and 75 percentile edges.

4.3.2. Performance measures

The performance of a demonstration in different modality is measured using diverse indicators namely; duration of the task, total number of velocity commands used, average translational velocity of the robot, average curvature of the executed trajectory and the variance of the distribution of curvature. The total time of demonstration is computed from the first command until the final command issued by the user. With a joystick; pressing and release of the trigger switch indicates the start and end of a demonstration. In gesture based mode, the explicit stop demonstration gesture is used whereas in steering wheel based remote teleoperation the user presses the *stop demo* push button in the GUI. The number of velocity gradient changes are also recorded from the number of different rotational velocity commands. The number of commands is an useful measure to interpret the user difficulty together with the sensitivity of the demonstration mode. The number of extraneous and corresponding correctional commands is also be determined for more insight. Curvature is computed as the ratio of commanded rotational velocity to translational velocity. The variance of these curvatures indicate the degree of continuity during the demonstration. Smoothness of the demonstration is computed from the measure of dispersion of the curvature through signal-to-noise Ratio (SNR). SNR is computed by taking the ratio of mean of the recorded curvature to its standard deviation, thus giving a measure of dispersion of the data. This interpretation decouples the robot motion from the dynamics thereby allowing to compare slow and fast motions against each other. The pedagogical aspect of instructive learning of a robot is collected with a questionnaire filled by the users at the end where the different demonstration modes are rated with respect to their relative degree of difficulty, fun and the amount of time needed to expert the mode (See Appendix A).

4.4. Validation and user study

The results from the user studies indicate that joystick based demonstration which required the least preparation time has also the fastest average execution time with a median value of 19.16 seconds followed by gesture based demonstration with 25.95 seconds. Remote teleoperation with steering wheel required longest acquaintance time (median of 48.14 seconds) mainly due to the apparent lack of visual feedback about the robots position within the environment. Fig. 4.9 shows the bar plot of the demonstration time required for different tasks together with a box plot of the average demonstration time for the three different modes. The figure shows that for medium complexity tasks such as

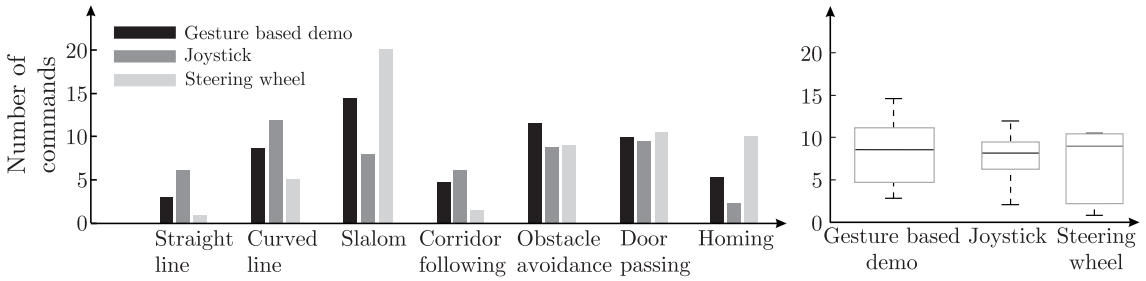


Figure 4.10.: Bar and box plot of the Number of commands used for the three modes across all scenarios. The box plot shows the median, 25 and 75 percentile edges.

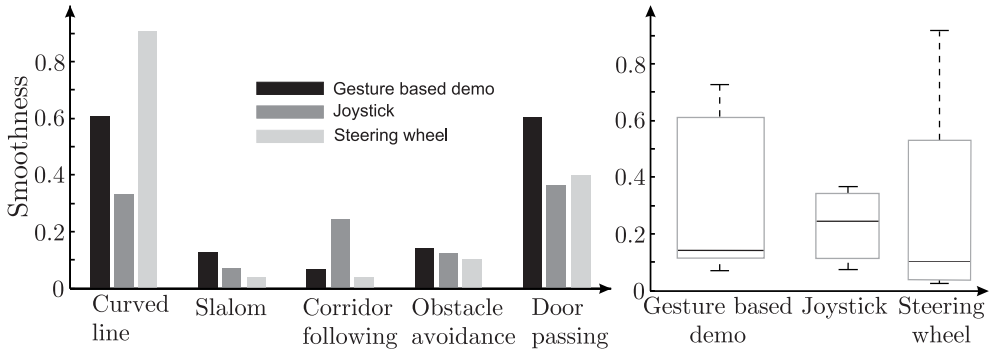


Figure 4.11.: Bar and box plot of the smoothness of the recorded curvatures expressed as SNR of the data. The box plot shows the median, 25 and 75 percentile edges.

obstacle avoidance, homing and door passing, remote based teleoperation requires more time to complete the task compared to the other modes. Demonstrating a slalom task remotely presented as the most challenging of the tasks. Despite the distance measurement given by the sonar readings, the participants were very careful of the robots safety which eventually reflected in a slower demonstration time and with relatively higher corrective actions. On an average, participants needed 240 seconds to complete the slalom tasks remotely compared to 50 seconds with gesture based mode and a mere 26 seconds with the joystick. Similar trend is also seen with door passing behavior demonstration remotely (joystick - 16 seconds, gesture based - 26 seconds and steering wheel - 85 seconds) and homing (joystick and gesture based - 24 seconds and steering wheel - 48 seconds). The number of commands needed to demonstrate a slalom behavior correlates with the total demonstration time needed. On an average of 50 commands were needed for steering wheel based remote demonstration whereas joystick and gesture based demonstration required only 8 and 14.5 commands for slalom. Tasks with relatively lesser complexity such as obstacle avoidance (joystick - 9.5, gesture based - 10.0 and steering wheel - 10.5) and corridor following (joystick - 8.8, gesture based - 11.5 and steering wheel - 9) needed relatively same number of commands. From the experiments, one could notice that gesture based control performs consistently well across all scenarios and also with non-experts. All the tasks experimented had a specific demonstration start and end location except for corridor following where the users decide themselves to end the demonstration once they deem the robot is centered and aligned in the corridor. This difference is reflected in the relatively short completion time of the corridor following behavior demonstration (joystick - 11.9 sec, gesture based - 6 sec and steering wheel - 9.4 sec). Fig. 4.11 shows the smoothness of the tasks except for straight line following and homing determined from

the SNR of the curvature. SNR interprets the dispersion of the curvature data along a demonstration. Thus, a smaller value indicates higher dispersion and larger value indicates a lower dispersion or a smoother trajectory. From the figure, one can notice that gesture based demonstration mode generates less noisy demonstrations compared to the other schemes. Joystick based demonstration data on the other hand has more noisy nature attributed mainly to the haptic properties while controlling both translational and rotational velocities together. Even though, handling joystick to control a robot is the most easiest to get accustomed to, the sensitivity of the movement and maintenance of a coherent relationship between the translational velocity and turn rate needs a larger preparation time.
After completing the tasks, the participants are asked to survey the difficulty and the ease of the different modes (See Appendix A). The user opinions confirm the intuition in that joystick is the easiest and fastest to get acquainted whereas gesture based demonstration is the most intuitive to generate relevant and smoother examples coupled with a better fun factor to keep the user engaged. The steering wheel based teleoperation on the other hand needed less preparation time than gesture based mode, nevertheless the lack of depth perception from an image makes the demonstration relatively harder to master. With a longer preparation time, the users are able to generate more smoother trajectories. The findings of the results suggest that NUI based demonstration is more intuitive, less noisy and reaches out to all users of diverse backgrounds. Joysticks are extremely fast and easy to get started but the nature of demonstration tends to produce more outliers. From the perspective of the data, this falls behind the other two but starts a clear favorite from a user's perspective. Steering wheel based teleoperation based on mounted omnidirectional camera performs less satisfactorily on comfort. Nevertheless, the potential of this mode of teaching increases the mobile robot flexibility to learn new tasks even when the teacher and the robot are very far apart.

4.5. Related work

The rationale to have easier and intuitive interface for both experts and non-experts to handle and teach navigational skills to a mobile robot has been the basis for many research in LfD and Human-Robot-Interaction (HRI). The approaches distinguish themselves mainly with respect to the demonstration platform and the mode of interaction between the teacher and the robot for information transfer. Kinesthetic teaching by demonstrating directly on a manipulator is the most popular mode of generating examples [Cal09]; [KNCC11]. For a mobile robot, this modulation needs extra sensory cues to generate relevant and situation-aware training data. In [NM01], the robot learns the skills by observing and following a teacher. This modality also allows transfer of underlying behaviors between two robots. To extend this to learn navigational behaviors, one way is to observe the skill through an external camera and follow the same example to generate its own data. But this poses considerable challenge in that, accurate positioning and localization of the human together with the interpretation of the skill set performed first needs to be done even before the data could be changed to an understandable format for the robot. [DH02] provide an insight into an active mode of learning by letting the robot follow the teacher. Both the teacher and robot share identical platform thereby overcoming

the embodiment mapping. For learning visual navigational behaviors, following a teacher is also not a practical approach as it needs relevant features that capture the local frontal environment. Thus in our approach, the teacher walks behind the robot and instructs the robot with a joystick or a specific gesture. Other sensory inputs such as auditor cues [KTM10] and with corrective feedback from the robot [Che09] have been discussed for improving the quality of the demonstration. In [KTM10], a mutual exchange between the teacher and the robot is carried out using audio messages. Other mode of people tracking with a potential for robot instruction can be seen [CD04]. The authors use a thermal camera and fuse it with color images captured by a pan-tilt monocular camera to detect people in the images. With the advent of 3D cameras, the need for multiple cameras to acquire the depth and location of the human is made easy. [CBB03] present another interesting aspect of teaching where the people movement trajectories are observed and modeled on the robot. The data is recorded using laser-range finders and similar trajectories that characterize human motions are clustered. In [Che09], the confidence on the quality of the policy deems the robot to request for new corrective demonstration to augment the low confidence policy. Proper interpretation of the teachers instructions depends on the transfer of the modality differences and thereby solve the correspondence problem to map human onto relevant robot actions. Natural body movements for robot instruction and guiding are presented in [MSG10], where the user points a goal location with a finger. We utilize this aspect in a similar NUI fashion using body gestures to command velocities. Teleoperating a robot using a live video stream is also presented in [GL96]. In this contribution, this mode is also considered as one alternative by providing video sequences of omnidirectional camera mounted on the robot and using a steering wheel and accelerator pedal to control the robot velocity and turn rate.

4.6. Summary

This chapter presented a comparative analysis of three modes of demonstrating navigational behaviors to a mobile robot in an indoor environment. The approach tries to tackle the problem of generating outlier free and relevant training data in the context of LfD for mobile robot. Further motivation stems from the need to design an easy and intuitive approach for all kinds of users with different knowledge of robotics. Three teleoperation modes with different modality are designed and tested. Joystick represented the haptic mode of direct mapping between the robot and the user. There is an identity record mapping as the actions are transferred by directly controlling the motor velocities of the robot. The depth information of Kinect camera is used in the second mode of teleoperated demonstration, where user gestures are interpreted as robot motion commands. The NUI based framework used body postures to extract turn rate whereas the rate of change of distance between the user and the robot presents the translational velocity. Remote teleoperation using a steering wheel and accelerator pedal is considered as the third mode where a continuous video stream of the mounted omnidirectional camera is used by the user to guide the robot through the environment. To enhance the information, additional information such as nearest obstacles detected by sonar and a birds eye view generated from the omnidirectional camera is also provided. To test the flexibility of the three approaches, user studies are performed with volunteers from diverse backgrounds. The

studies show that using gestures to command a robot allows even a lay user to generate smooth trajectories and at the same time providing an intuitive interface. Joystick is the easiest and fastest to master, but suffers from the haptic sensitivity which requires additional preparation time. Remote teleoperation presents difficulty to most of the users mainly due to the unfamiliar first person view of the environment. The lack of depth information aggravates the difficulty and the loss of perspective view of the environment makes the mode very challenging. The demonstration time is generally longer operating remote, but presents a high potential to generate new demonstration examples especially in situations where it is difficult for a human teacher to interact with the robot physically. With adequate acquaintance time, the survey shows that the performance of the demonstration and the fidelity of the training data improves. Suggestive improvements for this approach are using alternative teleoperation modes such as a gamepad control where individual buttons for specific motion commands. This might improve the fidelity of the data compared to a joystick in that constant velocities can be directly controlled from the user interface and allows a fast interchange between semi-automatic and fully manual control. With the advent of short range depth cameras such as PMD nano [Nan13], more advanced natural motions such as finger motions can be used to direct the robot more faster and with more control. The presented demonstration methodologies here, give an insight into identifying and designing individual components required for building a comfortable and complete human-robot interaction.

5

Scenario and context specific visual behavior learning

> *Our senses were made for the environment, there is nothing else of which they can be aware. So the problem of environmental awareness is the general problem of sensory awareness.*
>
> – Mark Terry, *The Abundance of Environmental Educators*

Programming navigational behaviors from demonstrations needs to counter two important design decisions, namely *how to perceive an environment and what decisions are appropriate for those perceptions?* The first question deals with understanding the scenario the robot finds itself in. In other words, it requires capturing and understanding the characteristic aspects of the world around the robot through image features that are similar to demonstration examples. The second question addresses the execution of right action within this scenario.
A reactive robotic behavior is highly situated in nature in that it correlates directly to the immediate nature of the environment. A heuristic for design and development of such a system is presented in [Ark98], who proposes to modularize the global behavior policy into sub-behavior control agents. The essence is to have a network of task specific behaviors communicating with each other thereby emerging a global control policy. Such a policy requires an agent to arbitrate or fuse the individual behavior policies. One way to avoid a centralized planner is to modularize the behavior emergence in accord to the environmental scenario the robot encounters. An anti-thesis to this heuristic is to attempt and build a general, monolithic model of the relation between individual perceptions and actions across arbitrary scenarios. Such a model is difficult to design due to the immense variance in the appearance of the environment and correspondingly the need of an enormous demonstration data to cover this diversity.
This chapter addresses the above issue by introducing a two level matching scheme where both the scenario and context of operation are considered. Formally, we define the two terms as following,

- A **scenario** describes the robots local surrounding based on the type of geometry by giving it a semantic class. Scenario is not to be confused with place recognition where a specific location is recognized but identifying similar geometry of environments seen before. For example, a robot in building A corridor perceives a similar scenario as a robot in building B corridor.

- A **context** describes the current situation of the robot based on its recent perceptual history. For example a robot turning right to avoid a static obstacle to its left has the same context as a robot turning right to avoid a person standing to the robot immediate left.

In the proposed navigation scheme, first a classification of the traversing scenario based on the distribution and shape of free floor space segmented on the omnidirectional image captured by the robot is done. Upon the classification of the current scenario, a second level identifies the current context within the scenario similar to the examples presented during demonstrations. Scenario classification is vision based matching whereas context identification is akin to matching time series trajectories. As an alternative to identifying the context, all the perception-action pairs within a certain scenario is learned using an artificial neural network. Decisions in individual scenarios depend on different aspects, e.g. obstacle avoidance relies on the angular distribution of the local free space, whereas corridor following relies on the robots lateral and angular displacement with respect to the corridor walls. Within each scenario the relevant subset of features that best generalizes unseen data are extracted and selected. This mapping between the geometric feature set and the robot action is learned and cross-validated for both bias and unseen data. Exhaustive tests on the generalization capability of both the schemes against trained and untrained environment across all different scenarios are performed. The acquired policies are validated on the robot to assess the performance. The robot behavior is learned from a partially teleoperated behavior demonstration. In the demonstrated behavior, the human operator controls the robots turn rate with a joystick whereas similar to cruise control the translational velocity is automatically controlled based on the proximity to the sonar detected nearest obstacle. The framework is equally applicable to the other demonstration modes presented in chapter 4.

5.1. Scenario classification and validation

The proposed matching architecture shown in Fig. 5.1 accounts for the situatedness of the behavior by first classifying the scenario and then select action on the basis of the context or of its specific model. Identifying a scenario deals with the classification of an image in an indoor environment into a topological scenario class. A semantic classification of the environment enables the mobile robot to efficiently utilize only the features that are relevant for the specific scenario. To illustrate this, we consider a robot in two different indoor environments namely a corridor and a foyer. Assuming the robot has no goal and is only wandering the environment, the robot in a corridor needs to know about its lateral orientation within the corridor with respect to the walls to position itself along the center of the pathway, whereas the robot in a spacious foyer needs to keep going straight until it encounters an obstacle or an eventual change in the environment which triggers another action. The lateral orientation of the robot in a foyer brings little to no information about the next action to be executed thus lacks relevance. Thus, it becomes highly important to first identify the scenario to arrive at correct motion commands. Fig. 5.2 shows a sample geometric map of an indoor office environment and the corresponding omnidirectional images. A typical indoor environment can be coarsely distinguished into three locations, namely a pathway or a corridor, an open space or room and a dynamic or cluttered envi-

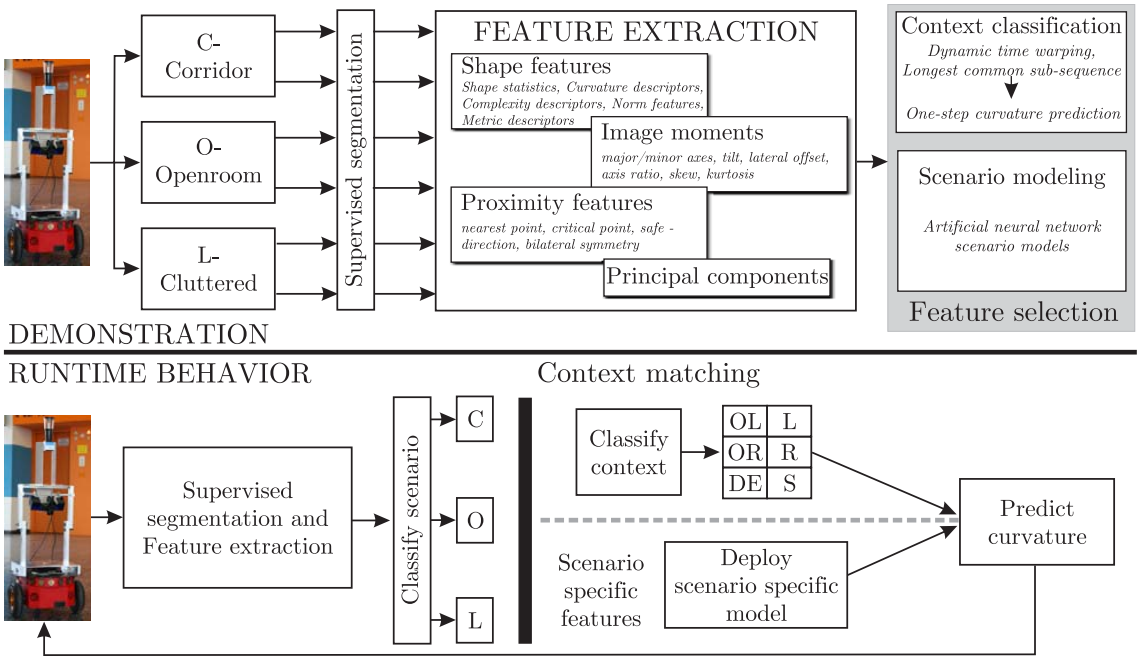


Figure 5.1.: The proposed two-level feature matching architecture

ronment. Supervised classification of the scenarios is performed by labeling them as *C*, *O* and *L* respectively. A total of 1831 examples are collected, of which 617 are from corridor, 584 from open spaces and 630 from cluttered environment. The corridors are approximately 2.5 m wide with pillars partially blocking the corridor. The images are recorded at different environments which differ in texture and appearance of the floor as well as the geometry and locations of walls and obstacles. Fig. 5.3 shows some snapshots of the environments used for training. Open room examples are performed in a hall or foyer with an isolated obstacle in the middle to trigger an obstacle avoidance behavior. The aim here is to keep going straight until the robot encounters an obstacle. The evading direction depends on the location of the obstacle with respect to the approaching direction of the robot. For instances where the obstacles are exactly in front, the robot is turned in the direction of maximum free floor distribution. The inherent situational awareness of humans to look ahead and foresee obstacles in executing motor decisions play a predominant role in deciding these direction of evasion. The segmentation image presents itself as a binary image of the robots omnidirectional camera perspective of the environment. From the segmentation image, fifty-four visual features introduced in chapter 3 are extracted. The relevant feature subset that classifies the three labeled scenario classes are identified using a feature selection routine. Classification is then performed with different learning algorithms such as Naive Bayes (NB), Regression Trees (RG), Probabilistic Neural Networks (PNN) and an ensemble learning with Random Forest (RF). Besides the fifty-four visual features, classification using the principal components of the segmented images is also analyzed. Cross-validation and generalization with PCA are compared with the previous result. The scenarios conceived relevant for robot navigation are labeled manually and arise from logical regression of possible environments that a robot might encounter. We establish this manual hypothesis of the number of relevant scenarios by performing unsupervised classification of the acquired training data with Self-Organizing Map (SOM). Fig. 5.4 shows the system architecture of the scenario classification framework.

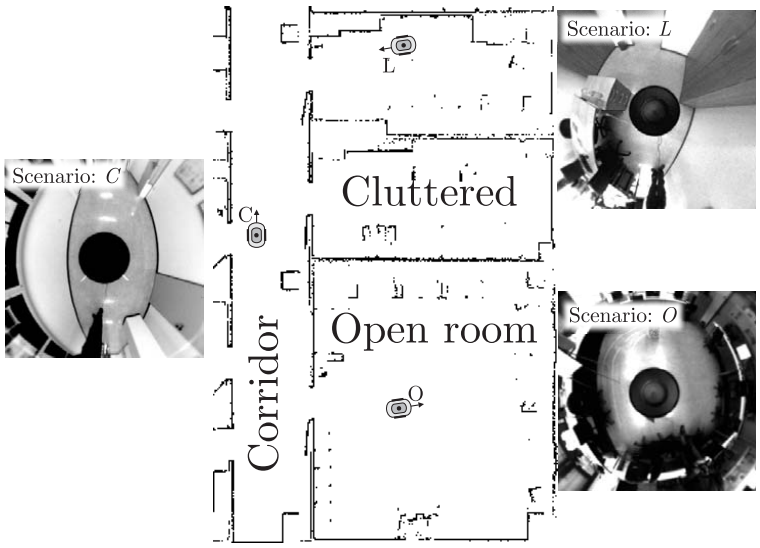


Figure 5.2.: Geometric map of an indoor environment with corridor, open spaces and cluttered offices. The corresponding omnidirectional images are shown together with their semantic label.

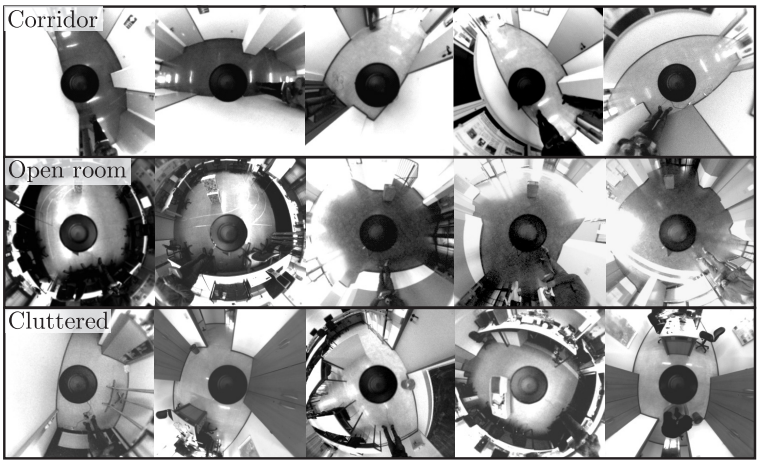


Figure 5.3.: Examples of training scenarios used for demonstration

5.1.1. Visual features

As a prerogative to any machine learning algorithm, the most characteristic features for the classification task are identified through a wrapper approach by convolving it with different classifiers. Following the paradigm, that the best m features are not the m best features, a minimal-redundancy-maximal-relevance (mRMR) feature selection technique [PLD05] is employed to identify features that demonstrate a strong dependency on the target class and at the same time exhibit minimal redundancy. mRMR is filter based where the features are selected based on preset conditions such as mutual information and correlation between the input and target class. The optimal set of features are then determined by convolving the features with different classification algorithms such as NB, RG and PNN. Table 5.1 shows the top 15 features identified by mRMR. Fig. 5.5 shows the nature of distribution of the two such selected features across the three scenarios. The scatter plot shows that there exists a good distinction between the three scenarios, nevertheless with an ambiguity at the transition region between corridor and open room. The 15 proposed features from mRMR are filter based and does not give

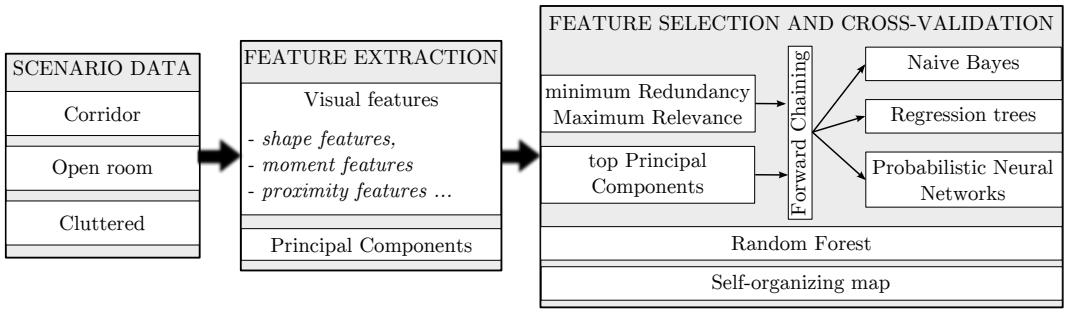


Figure 5.4.: Scenario classification architecture

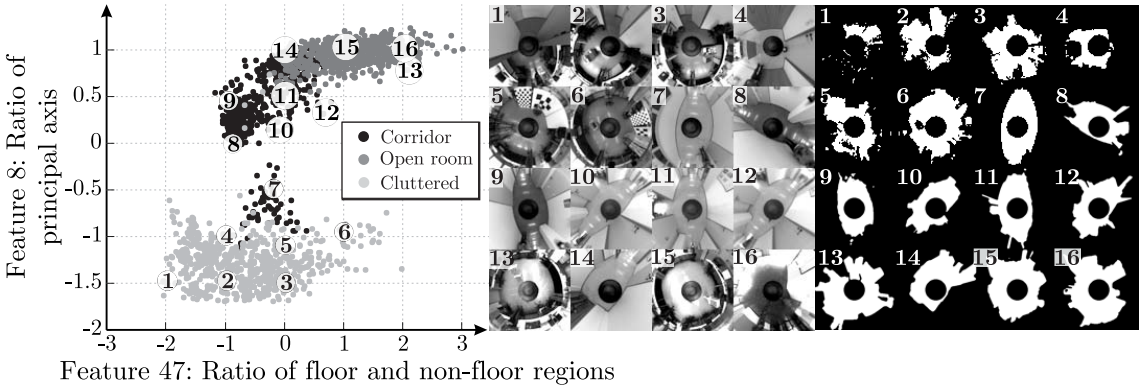


Figure 5.5.: (Left) Features: Ratio of floor and non-floor vs ratio of the principal axis of the scenario data. The numbered points are the vertices of the closest relevant grid points of the point scatter. (Right) The corresponding omnidirectional scenario training images and the segmented free floor images.

the optimal set of features based on classification accuracy. This is found by a wrapper approach. A 10-fold cross-validation is performed to compute the classification accuracy of the feature set in every iteration. Fig. 5.6 shows the classification error rates of the different classifiers using forward chaining. The figure shows that RG outperforms the others by almost 2%. PNN has relatively higher classification error and the error tends to saturate with 12 features whereas NB and RG needs 15 features. Table 5.2 tabulates 10-fold cross-validation classification errors. The confusion matrix together with the bias precision and recall errors for the three classifiers are tabulated in Table 5.3 and 5.4. In a classification algorithm, bias precision error gives the proportion of true positives among the predicted values, whereas recall error gives the proportion of actual true positives among the total available positive predictions. Recall error is the statistical equivalent of measuring the sensitivity of a classification algorithm. By visualizing the performance of the classification algorithm with a confusion matrix, where the columns of the matrix represent the predicted classes and the rows represent the actual class, the bias precision and recall errors are [OD08],

$$\text{Precision} = \frac{t_p}{t_p + f_p}; \text{Recall} = \frac{t_p}{t_p + f_n}, \tag{5.1.1}$$

where t_p and f_n correspond to true positives and false negatives respectively. From the cross-validation and the confusion matrix, we can see that both RG and PNN perform equally good and exhibit a better performance against NB. The performance of PNN strongly depends on the standard deviation value or the spread of the Radial Basis Function (RBF) used to determine the distances in the hidden layer of the neural net. The

Table 5.1.: Top 15 features for classification using mRMR feature selection technique.

Feature number	Feature name
27	Area of shape
47	Ratio of floor and non-floor regions
53	Image entropy
54	Entropy of robot front half plane
37	Multiscale Statistics (Standard deviation of second level scaled down image)
17	Skewness of distribution of boundary point distances to the shape centroid
36	Multiscale Statistics (Standard deviation of first level scaled down image)
8	Ratio of the principle major and minor axis
33	Temperature of shape contour
21	Mean size of shape
25	Bilateral symmetry
29	Thinness ratio
18	Kurtosis of the distribution of boundary point distances to the shape centroid
30	Area to perimeter ratio
32	Rectangularity of the shape

Table 5.2.: 10 fold cross-validation error using the feature subset selected from forward chaining. See Table 5.1 for the features used.

Classifier	Number of Features	Misclassification rate
Naive Bayes	15	0.04
Regression Trees	15	0.03
Probabilistic Neural Networks	12	0.06

optimal value of the Gaussian standard deviation is determined by cross-validating the PNN for all values between zero and one. Tests show that a spread of 0.5 generalizes best to unseen data and therefore is selected for future validations (See Appendix C.1). In addition to the mRMR based feature subset used for classification, an ensemble learning method of classification using RF is also tested. RF operate by constructing multitude of classification decision trees. Each tree is built using random m features. The training of the tree is done by cross-validating the data by bootstrapping two-thirds of the examples. The out of bag or the cross-validation error is computed on the remaining one-thirds of data. Thus, RF has an advantage in that the cross-validation is inherent to the algorithm itself. RF can also be used to perform feature selection in a natural way. The feature importance of every input variable can be computed by determining the increase in the MSE by omitting the feature to compute the decision tree. Thus the features those indicate a higher increase in the MSE correspond to features with higher importance for the classification. A total of 100 random trees are generated and the out-of-bag classification error for these trees together with their feature importance are shown in Fig. 5.7. It shows the loss curve for the generalization error for the number of trees grown. The

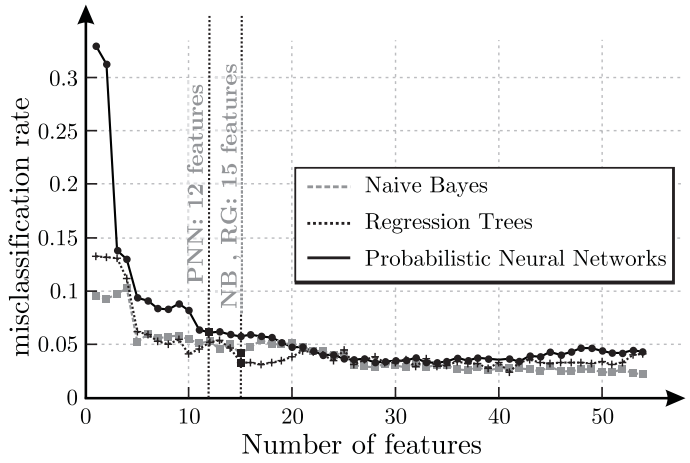


Figure 5.6.: 10 fold cross-validation error during forward chaining on mRMR selected features.

Table 5.3.: Confusion matrix of bias test with (left) Naive Bayes, (middle) Regression Trees and (right) Probabilistic Neural Networks using the selected feature sets. See Table 5.1 and 5.2 for the feature details.

Naive Bayes			
	C	O	L
C	563	35	19
O	12	568	0
L	9	0	625

Regression Trees			
	C	O	L
C	614	2	1
O	7	573	0
L	3	1	630

Probabilistic Neural Networks			
	C	O	L
C	616	1	0
O	0	580	0
L	1	0	633

classification error saturates to approximately 4% for twenty and more trees. The plot on the right shows the importance of the features obtained by omitting them and computing the corresponding increase in the generalization error. By setting cut-off at 0.8, the 10 most relevant features are identified as shown in Table 5.5.

5.1.2. Principal components

One of the advantages of PCA is its ability to capture the direction of variation in a data set. This direction defined by the Principal Components allow to describe large data sets with few components thereby implicitly identifying the latent features describing this variability. The segmentation images of the three scenarios shows that there is a visually recognizable variation in the spread and shape of the free floor across them. PCA captures this variation and allows to be presented with lesser dimensions. Forty five images each from corridor, open room and cluttered environment examples are selected as the training set. Taking the entire 1831 training examples for computing the principal components with SVD is computationally expensive. Hence forty five examples from each scenario

Table 5.4.: Bias errors: Precision and recall errors for the scenario classifiers.

Classifier	Precision			Recall		
	C	O	L	C	O	L
Naive Bayes	0.964	0.941	0.97	0.912	0.979	0.985
Regression Trees	0.983	0.994	0.998	0.995	0.987	0.993
Probabilistic Neural Networks	0.998	0.998	1	0.998	1	0.998

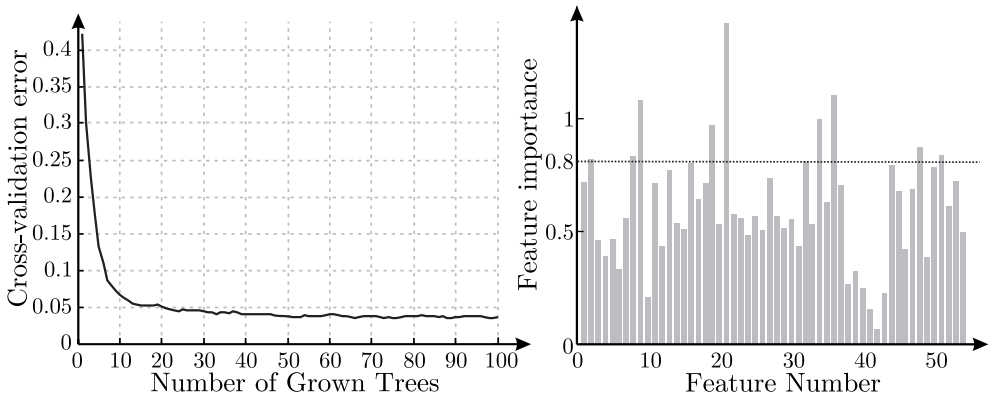


Figure 5.7.: Random forest based scenario classification. (Left) Out-Of-bag cross-validation error and (Right) Feature importance.

Table 5.5.: Top 10 features obtained by RF

Feature number	Feature name
2	Y-coordinate of the shape centroid
8	Ratio of the principle major and minor axis
9	Maximum distance of shape boundary to the centroid
19	2-n Euclidean norm
21	Mean size of the shape
32	Rectangularity of the shape
34	Multiscale statistic (Entropy of first level scaled down image)
36	Multiscale statistic (Standard deviation of first level scaled down image)
48	Distance of the critical point to the robot
51	Ratio of the floor to the right and left of the robot

that capture the diverse texture, geometry and the essence of the scenario are used. Fig. 5.8 shows the segmented images used for extracting the principal components together with the mean and the top five principal components registered. Fig. 5.9 shows the scree plot of the computed components and the variance explained by each of the components. The figure shows that with twenty components the singular values flatten out but does not provide an exact number of sufficient relevant components. Although there are 135 possible components to explain 100% of the variance; it would require a huge training data set to cover this high dimensionality. Thus, the relevant components are identified by incrementally checking for the improvement in classification accuracy on unseen testing data by convolving it with different classification algorithms. The entire 1831 training examples are projected down to 135 components and are cross-validated to find the best component subset for classification accuracy. The nature of the principal components in distinguishing the scenarios on the collected data is shown in Fig. 5.10. From the figure, one can observe that the first component captures the circularity of the floor region and the second component captures the entropy or the total area of the segmented floor to the front of the robot. Fig. 5.11 shows the misclassification rates of the different classifiers using forward chaining. The *10-fold cross-validation error* for the components show that both NB and PNN achieve a 94% classification accuracy (See Table 5.6). Nevertheless,

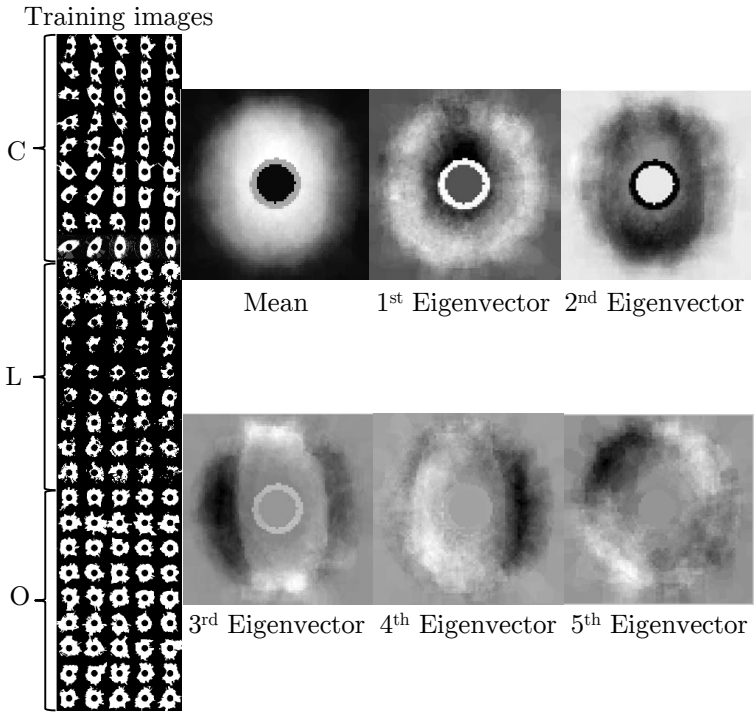


Figure 5.8.: Training images used for extracting the principal components and the mean vector together with the top five components.

NB needs 15 components to achieve this performance whereas PNN captures the distinction with only 6 components. RG in contrast achieves only a maximum classification accuracy of 90.08% with 5 components. Table 5.7 also shows the bias performance of the algorithms and one can observe that majority of the misclassification occurs because of the ambiguity between few corridor and open room images (example: image 20 and 21 in Fig. 5.10). The precision and recall errors of the three algorithms are shown in Table 5.8. All the three classification algorithms show a classification accuracy of over ninety percent with PNN exhibiting the best performance across all the three scenarios. Nevertheless, the performance of the classification lies in the generalization ability to unseen and completely unknown test examples. In addition to the cross-validation

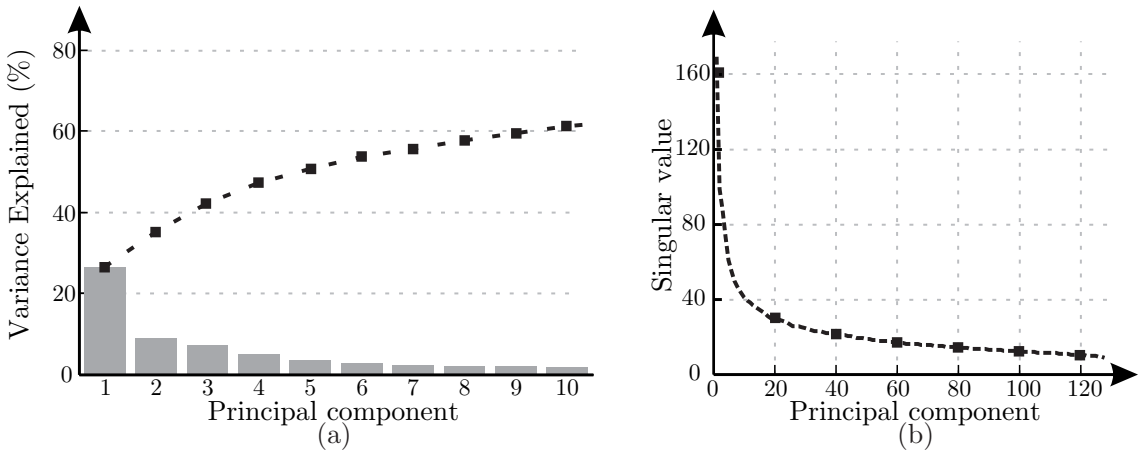


Figure 5.9.: (a) Percentage variance explained by the first 10 principal components and (b) Scree plot of the singular values for the first 130 components.

Table 5.6.: 10-fold cross-validation error using principal components for scenario classification.

Classifier	Number of Features	Misclassification rate
Naive Bayes	15	0.06
Regression Trees	5	0.09
Probabilistic Neural Networks	6	0.06

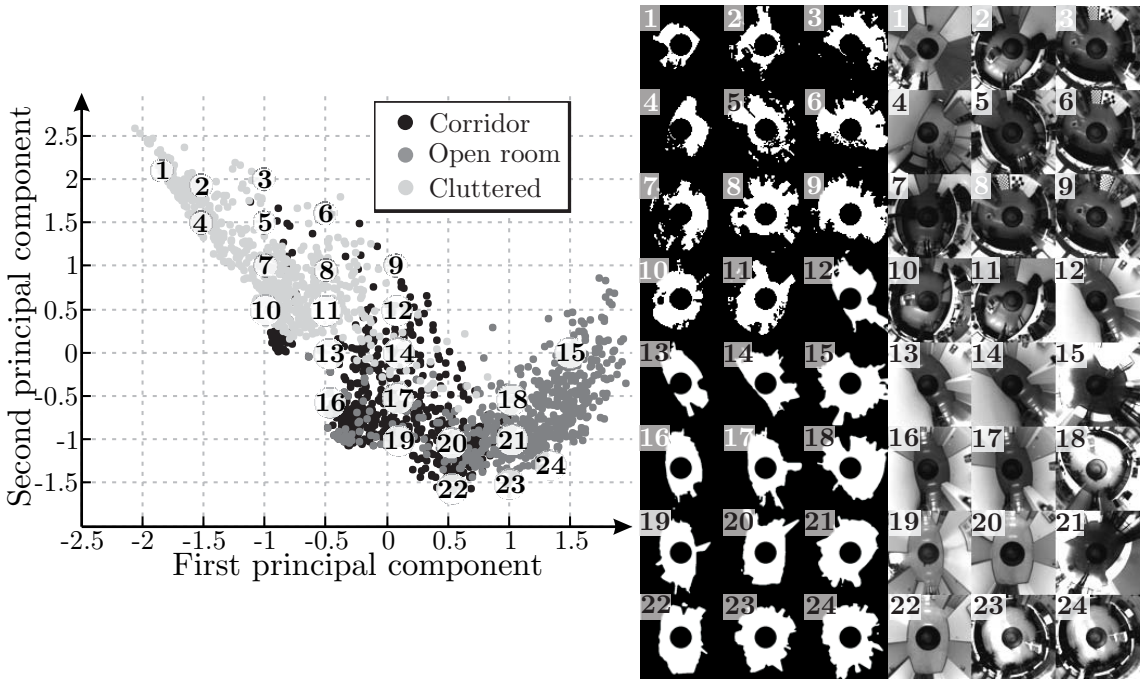


Figure 5.10.: (Left) The first two principal components of the scenario data recorded. The numbered points are the vertices of the closest relevant grid points of the point scatter. (Right) The corresponding omnidirectional training images and the segmented free floor images.

error shown in Table 5.6, experimental results on different environments are described later to establish the robustness and the performance of all the algorithms. Similar to the classification with visual features, principal component based data is also classified with RF. The ensemble of experts approach of RF should augment the performance further compared to individual classifiers. Fig. 5.12 shows the out-of-bag cross-validation error registered while growing the number of trees to learn the classification boundary. RF shows an improved performance compared to the previous three classifiers however does not better the classification accuracy achieved with visual features. The figure also shows the top eight components whose feature importance is higher than the cut-off of 0.8

Table 5.7.: Confusion matrix of bias test with (left) Naive Bayes, (middle) Regression Trees and (right) Probabilistic Neural Networks using principal components.

Naive Bayes			
	C	O	L
C	585	19	13
O	51	530	3
L	4	21	605

Regression Trees			
	C	O	L
C	597	13	7
O	53	532	0
L	6	0	624

Probabilistic Neural Networks			
	C	O	L
C	602	14	1
O	50	534	0
L	0	1	629

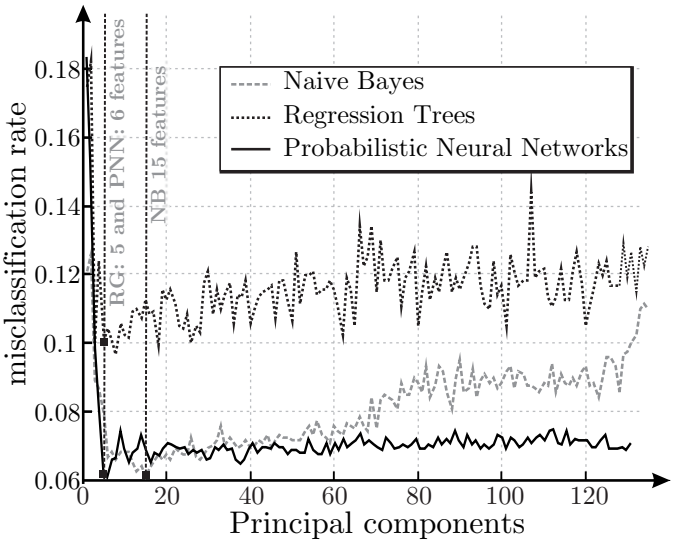


Figure 5.11.: 10 fold cross-validation error on the number of principal components for scenario classification

Table 5.8.: Bias errors: Precision and recall errors for the PCA based scenario classifier

Classifier	Precision			Recall		
	C	O	L	C	O	L
Naive Bayes	0.914	0.929	0.97	0.948	0.907	0.96
Regression Trees	0.911	0.976	0.988	0.967	0.910	0.990
Probabilistic Neural Networks	0.923	0.972	0.998	0.975	0.914	0.998

(See Table 5.9). A bias generalization accuracy of 96.4% is obtained using the top eight components. Table 5.9 also shows the registered confusion matrix for bias prediction. In summary, both visual features based classification and the principal component based classification acquire very high classification performance between the three hypothesized scenarios deemed relevant to learn navigation. Nevertheless, only the false positive rate of the classifiers in unseen environments would establish the robustness of the method. This is addressed in the section 5.1.3 where the discussed classifiers are tested experimentally in new indoor environments.

Unsupervised Self-Organizing Maps

SOM is a data visualization technique and an unsupervised classification algorithm which works by reducing a higher dimensional data and representing it in lower dimensional

Table 5.9.: (Left) Components with feature importance of more than 0.8 and (Right) Confusion matrix of bias test with RF on principal components

Top components
1 2 3 4 5 6 10 15

Random forest			
	C	O	L
C	584	33	0
O	31	553	0
L	0	0	630

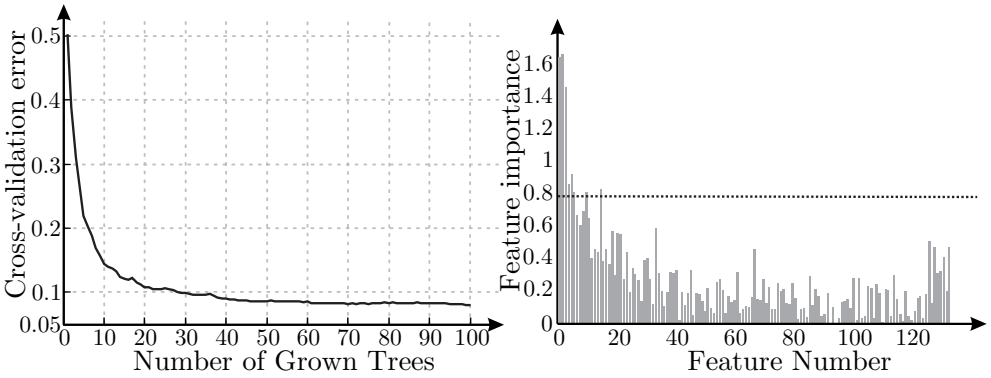


Figure 5.12.: Random forest based scenario classification using principal components. (Left) Out-Of-bag cross-validation error and (Right) component importance.

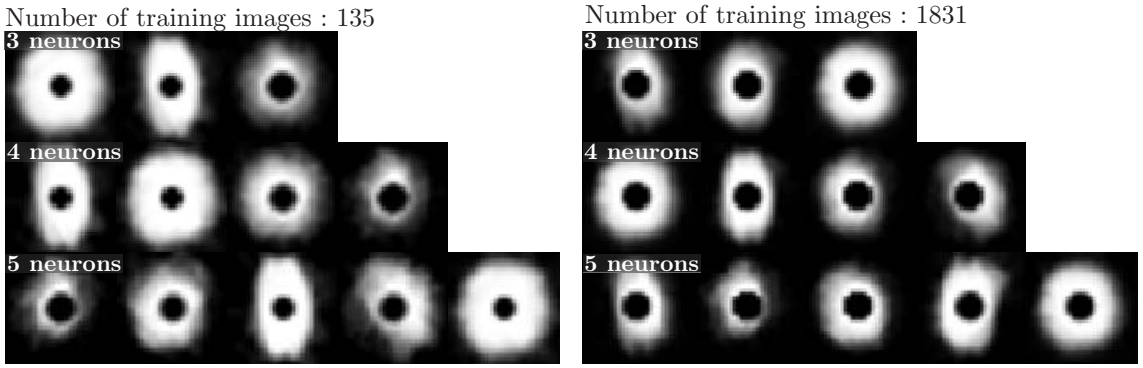


Figure 5.13.: Learned weights of SOM network with 3,4 and 5 neurons for both 135 and 1831 training images.

map, typically of 2 or 3 dimensions. SOM takes its inspiration from the organization and the evolution of neurons in the human brain where, particular sensory events triggers one or more neurons in a specific neural location [BB11] associated with similar or related events. A higher dimensional visual or sensory input is in theory mapped down to two or three dimensional neural location in the brain [BB11]. SOM also called as Kohonen map attempts this realization using i prototypes or neurons (P) lying in a one or two dimensional manifold to self-organize and associate itself to specific patterns in the data. The association here is done by finding the closest neuron to the given input and then move the neuron and its neighbors towards the input via an update. The similarity measure between the input and the neuron is usually a Euclidean distance measure. The best matching prototype (P_c) for a given input x is the closest unit given by [KH07],

$$\|x - P_c\| = \min_i \{\|x - P_i\|\} \tag{5.1.2}$$

The weight of a prototype (m_i) is then updated given by,

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{5.1.3}$$

where, t is the time and h_{ci} is the neighborhood function around the closest prototype m_c. For scenario classification, we use an one-dimensional manifold with three to five neurons, each to represent prospective scenarios. The goal of this approach is for the data to self-discover different scenarios. The neurons in SOM are placed evenly on the principal component plane of the input vectors and correspondingly bends the plane to

approximate the data points. The segmented free-floor images from demonstration are used as input vectors after sampling them down to 20×20 pixels. We utilize two separate sets of input namely, one with all the 1831 segmentation images and other only with the 135 segmentation images used to compute the principal components based supervised classification used in the previous section. Fig. 5.13 shows the learned SOM neuron weights as images for 3,4 and 5 neurons. The neuron weights thus represents the different identified scenarios. From the images, one can see that for three neurons with only 135 training images the clustered classes concur to the manual scenario labels of open room, corridor and cluttered. With 1831 training images, the open room scenario clearly distinguishes itself whereas the other two classes represent sub-scenarios or different contextual situations. Notice the three classes identified by training SOM with 1831 images. The third cluster corresponds to an open room, whereas the first two classes express the two obstacle evasion contexts where the obstacle is on the left and the free floor extends to the right and vice versa. Looking at the segmentation images one can also infer them to be two sub-scenarios outside of an open room environment. With four or more neurons, open room and corridor environments form a class by themselves and the different contexts form the remaining classes. For SOM with only 135 examples, the separation of the scenarios happens within the cluttered environment. Training with 135 images with 4 neurons show that apart from corridor and open room examples, the cluttered environment is further split into *approaching obstacle* and *very close to obstacle* scenarios. With more neurons, more complex sub-scenarios begin to emerge such as misaligned in the corridor on the left or right side, in the corridor center or evading obstacles on the left or right side in a cluttered environment etc. Fig. 5.14 shows the classified scenarios by SOM using a scatter plot of the first two principal components. Note here that for training SOM, the raw segmentation image is used as input. The principal components of the training set is used only for visualization. The figure also shows omnidirectional images and its corresponding segmentation images for selected points in the component space. Fig. 5.15 shows the scatter plot of the classified classes trained with 1831 input data.

5.1.3. Experimental validation

The scenario classification models are tested on the robot in different unseen environments. The robot mounted with an omnidirectional camera is allowed to wander around the environment for more than sixty minutes using a simple sonar based wandering behavior. During the trials, omnidirectional camera images are recorded together with the robots global position. Sonar based localization using particle filter is used to track the robots position in the map. Fig. 5.16 shows the path followed by the robot in the corridor of the department together with the classifier results trained with visual features. Multiple runs along the corridor from different starting positions are carried out along with separate trials to enter three rooms along the corridor. From the figure one can see that NB is more sensitive to perturbations in the environment such as open doors in a corridor. PNN and RG exhibit similar performance and consistency in classifying places. RF on the other hand is more intuitive than the rest in classifying the open room and cluttered environments. This can be specifically observed at nodes A, B and C detailed in Fig. 5.17. Node A corresponds to entering a computer lab with free space along the center,

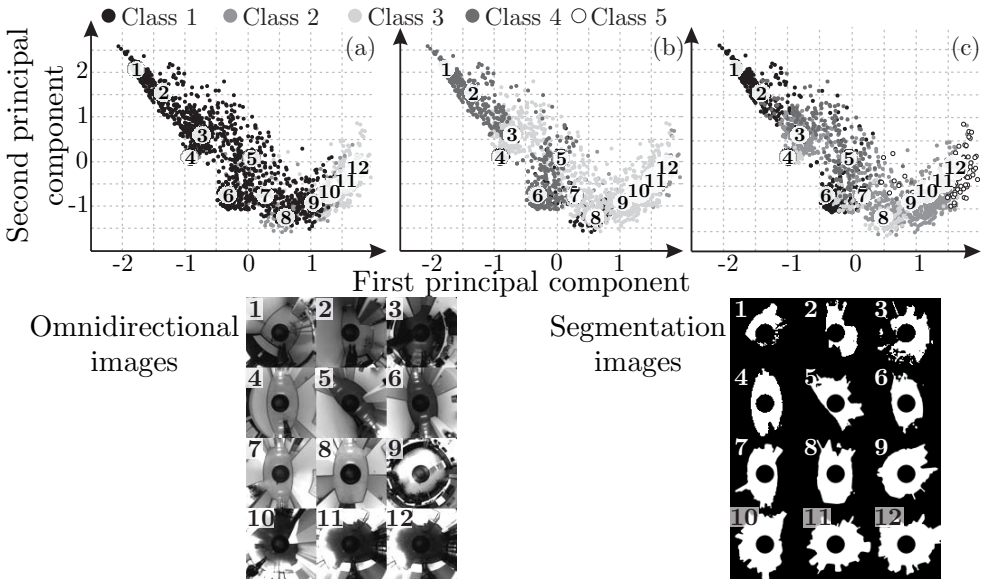


Figure 5.14.: Scatter plot of first two principal components projected down to all the demonstration data. The scenario labels are classified with SOM with (a) 3 neurons, (b) 4 neurons and (c) 5 neurons. Number of training data used: 135.

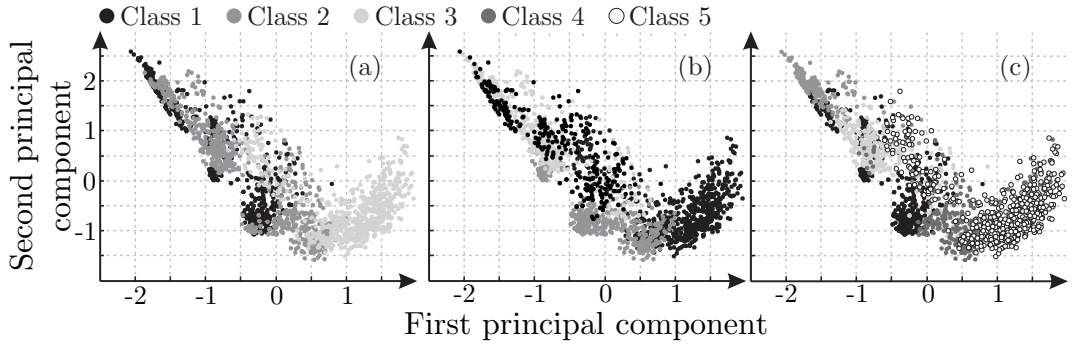


Figure 5.15.: Scatter plot of first two principal components projected down to all the demonstration data. The scenario labels are classified using SOM with (a) 3 neurons, (b) 4 neurons and (c) 5 neurons. Number of training data used: 1831.

node B corresponds to entering a narrow floor way to enter the kitchen and the node C is where the robot encounters the corridor end. At node A and B, while entering the room, NB, PNN and RG tend to classify the scenario as corridor until the door post is seen. From the context of semantic classification for map building, this is appropriate but for arriving at a decision as to which scenario model is appropriate for navigation, it can be argued that the classification is probably not optimal. In other words, NB, PNN and RG are rotationally invariant to the corridor alignment whereas RF is rotationally sensitive in that the orientation of the corridor plays a role during classification. The figure also shows approaching corridor dead end where RF identifies the dead end earlier and classifies it as cluttered compared to the others, which classify relatively late into the approach. Fig. 5.18 shows the classified scenarios predicted by the PCA based classifiers. In contrast to classifiers trained with visual features, PCA based classifiers are sensitive to image rotations. The image shows that all the four classifiers discriminate entering a room or a doorway and going along the corridor as two distinct scenarios. Visual features extracted from the segmented free floor image in the previous experiment are invariant to rotation

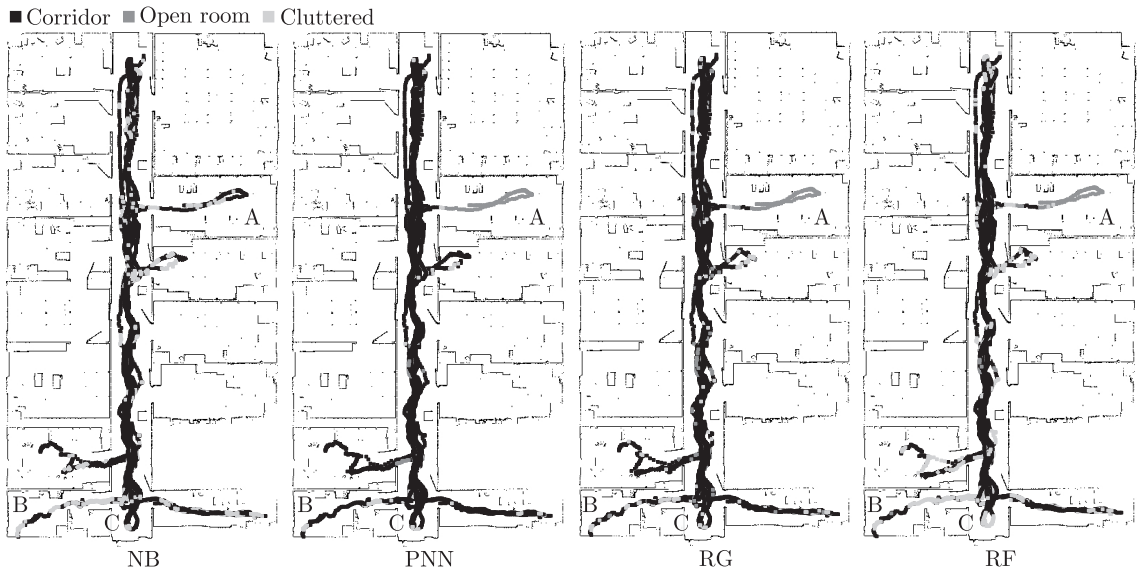


Figure 5.16.: Path traced by the robot wandering an office corridor environment and the corresponding scenario classified by the different algorithms trained on visual features.

in that they describe the geometry or shape of the extracted free floor with only a few features that capture the orientation of the robot within the environment. PCA on the other hand computes the principal components directly from the segmentation images. Conversely, visual features possess an advantage in that they are robust and consistent against image and environment noise, like opening a door or a dynamic obstacle.
Finally, the experiment is validated with the three and four class SOM trained on both 135 and 1831 images. Fig. C.2 shows the traced path along with the corresponding class. Trained with only few examples, the SOM generalizes even slight variations of environment along the corridor as scenario 3, which corresponds to cluttered environment. Trained with four classes, the cluttered classification is further divided into two subclasses corresponding to approaching and at critical distance to an obstacle. Trained over 1831 examples, both 3 and 4 class SOM distinguishes contextual scenarios rather than geometrical variations. As seen in Fig. C.2 with 4 classes, the corridor classification is more robust and consistent than its counter part. The intuitive nature of this classification and others are further validated in two unseen and untrained foyer and open space environment in Appendix C.
In summary, both visual features and principal components based classifiers generalize unknown and unseen scenarios good with little variations between them. Visual feature based classifiers are invariant to rotations, or in other words to the orientation of the robot within specified scenario whereas, principal component based classifiers are not. It is an objective decision to conclude what is more preferable as, one is more relevant with reference to navigational behaviors in contrast to classification of the scenario geometry irrespective of how the robot is positioned. Unsupervised classification of the recorded demonstration data establish that the logic in selecting relevant indoor scenarios is satisfactory with a search for more clusters revealing interesting contextual aspects within the environment. Among the classifiers PNN and RF exhibit consistency and perform robust against noise and perturbations in the environment with RG.

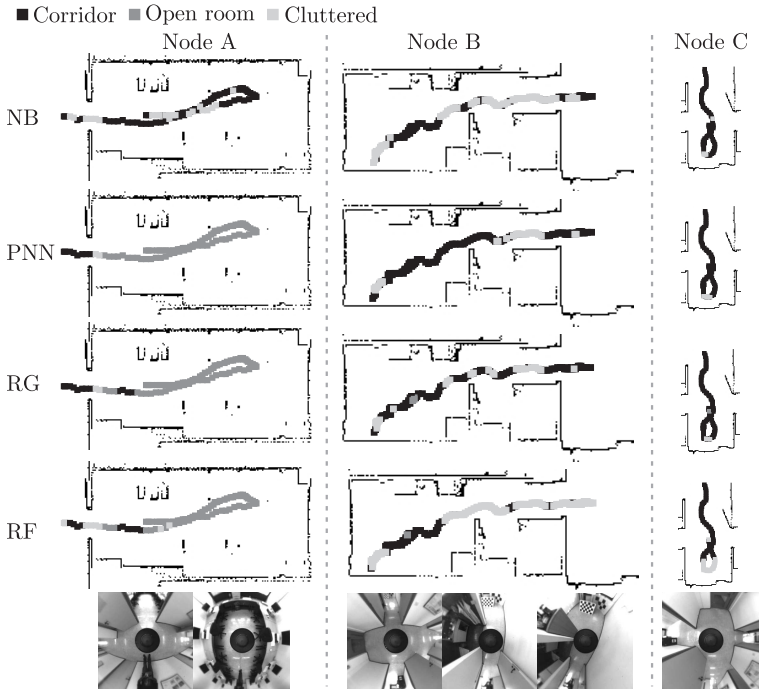


Figure 5.17.: (Left) Node A: entering the lab from corridor, (middle) node B: entering into kitchen from corridor and (right) node C: encountering a dead end in a corridor.

5.2. Context classification using perceptual trace

Post classifying the scenario, the demonstration context most similar to the current is identified and the corresponding action is imitated. A context within a scenario is defined by a perceptual trace of the features current history. This perceptual trace contains more information than a potentially ambiguous single perception and thereby constitutes a fingerprint of the context as well as the robots current state. In every scenario, different maneuvers behavioral contexts within multiple indoor scenarios are demonstrated and the examples are labeled accordingly. The labels are *OR/OL* for a right/left turn to avoid an obstacle, *R/L* for a right/left turn to center or align the robot, *S* for straight motion or stop action and *DE* for a turn within a dead end situation. The recent robots path curvature is also registered thereby attaining a pseudo-time series data that characterizes sub paths of persistent robot maneuvers. The path curvatures are transcoded as the ratio between rotational and translational velocities. Table 5.10 lists the features recorded for different contexts in each scenario. It is worth noting that the orientation error (β) which is considered for corridor scenario is omitted in the other two scenarios as it predominantly influences the corridor centering behavior and bears no relation with sequences of motion open room or cluttered environments. A total of 49 trajectories (10 OL, 11 OR, 10 L, 7 R, 8 S and 3 DE) in corridor, 29 trajectories (9 OL, 10 OR, 5 S, 5 DE) in open room and 25 trajectories (9 OL, 9 OR, 2 S , 5 DE) in cluttered environment are generated using manually controlling the robot with a joystick. The trajectory sequences are matched with two algorithms namely DTW and LCSS. Given two sequences of trajectory, DTW [BC94]; [SC90] measures similarity between the two and allows elastic shifting of the time axis or allows extension and contraction of the trajectories along the time axis. The algorithm takes account of the difference in lengths of the two trajectories and takes

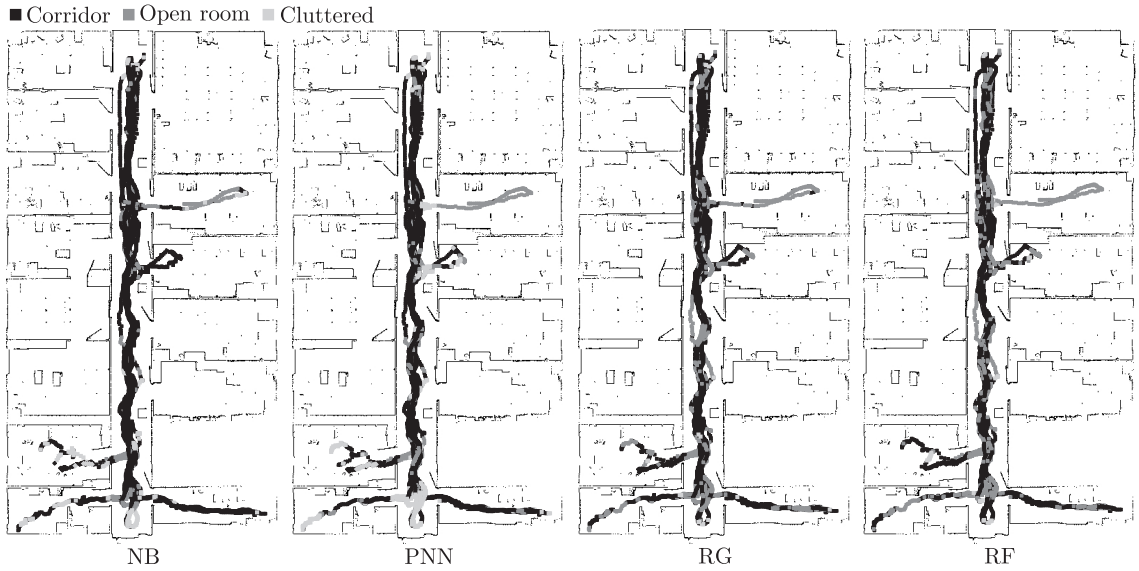


Figure 5.18.: Path traced by the robot wandering an office corridor environment and the corresponding scenario classified by the different algorithms trained on principal components.

Table 5.10.: Selected features for trajectory matching

Feature #	Feature	C	O	L
17	Critical distance $1/r_c$	x	x	x
18	Critical angle θ_c	x	x	x
7	Orientation error β	x	-	-
-	Curvature	x	x	x

account of the nonlinear nature of them. Alignment of the two trajectories is performed by arranging the two trajectories on either sides of a grid with each grid cell containing a distance measure (typically an euclidean distance measure) between the two corresponding elements of the sequences. The best match is then computed by searching for the optimal path through the grid that minimizes the total distances between them. LCSS takes its inspiration from DTW which finds the longest common subsequence matching the two trajectories. The idea is similar to DTW in which both allow stretching and contraction of the trajectories along the time axis with an exception which allows some elements to be unmatched [VKG05]. This makes LCSS more robust to outliers and exhibits efficient computation of the optimal path. Fig. 5.19 shows one such working principle result obtained by comparing two perceptual traces of β recorded within a *OL* context. Each element of the context examples are characterized by the perceptual feature vector as well as the path curvature at the specific time instance. Cross-validation of the trajectory matching between a given query trajectory with the reference trajectory is carried out by a novel *add one more* cross-validation method. At first a single instance of the query trajectory is matched to the reference trajectories. This situation is equivalent to the start of a run where the robot has no memory of any previous traversals. New elements of the query sequence are added and matched until the entire query sequence is mapped onto a reference trajectory. Two such cross-validation results are shown in Fig. 5.20 and 5.21. A left turn in a corridor (27 instances) to avoid an obstacle and a dead end (80 instances). The figures show the selected context together with a box plot of the results obtained

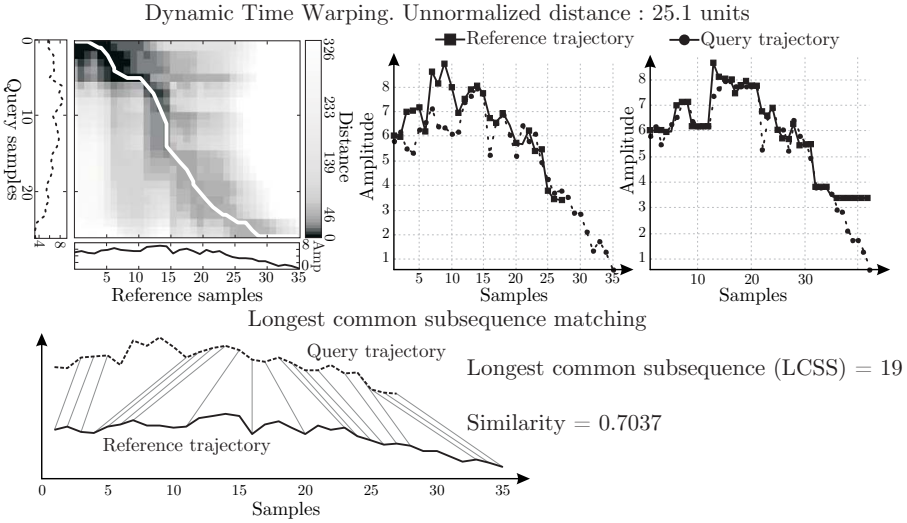


Figure 5.19.: Working principal behind DTW and LCSS on two perceptual traces of orientation angle (β) within OL context.

by other trajectory context comparisons. In our experiments, we restrict the length of maximum perceptual trace memory to 20 elements. With LCSS, the first few instances are matched incorrectly, as the robot possess no memory of the previous trajectory elements. However by adding more instances the trajectory matching results converge to the correct context. For trajectory matching with DTW, we utilize the minimum distance between the trajectories to classify the context. In the experiments, we observe that with fewer memory traces typically in the order of 10-15, DTW performs good but with traces longer than 15, the context classification tends to ameliorate. Fig. 5.21 visualizes the best matched results with the two experiments performed with a perceptual trace of 20. LCSS searches for a common sub-sequence and does not require a complete match of trajectory sequences. Thus even with a longer perceptual trace, LCSS converges to the right context. The results shown here only visualizes the matched context label whereas the process also predicts the exact matched trajectory and a history of its point-to-point correspondence. Once the point correspondence is established, the next single step curvature is predicted. By considering the different context examples across the three scenarios as a pseudo-time series sequences, a Recurrent Neural Network (RNN) [Elm90] is trained for the one step prediction. Recurrent neural network with hidden and context units provide an internal representation of the previous states being visited. The inputs are provided to the network like a standard feedforward network and the weights of the hidden layers are updated in a single backpropagation cycle. Upon the introduction of further elements of the input sequence, the recurrent connection between the hidden and the context layer allow the context unit to preserve the previous state or values of the hidden unit. This information exchange between the hidden and the context units allow the network to maintain the previous state of the hidden unit forming a kind of memory unit. Even though the RNN possess the capability to represent the complete dynamics of the perception variables to its action and thus provide a multi-step prognosis, we limit ourselves to a simplistic requirement of only a single step prediction.

Separate networks are trained for the different contexts across all the three scenarios. The context data set is fed as input to the neural network with the curvature as the

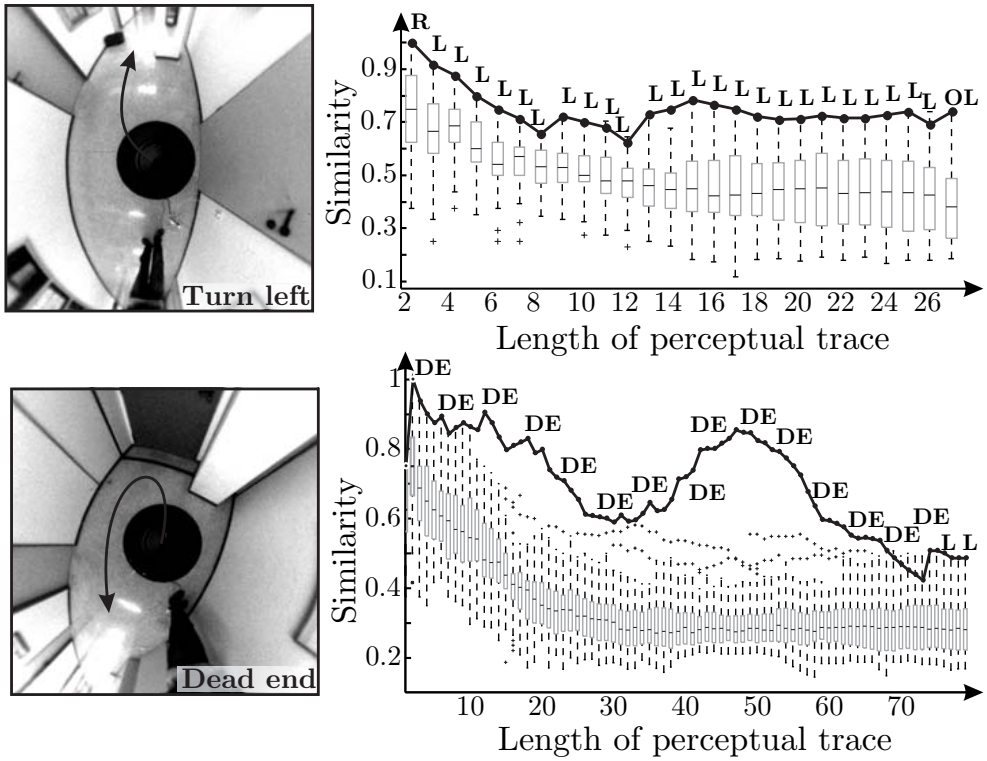


Figure 5.20.: Trajectory matching cross-validation with LCSS. Turn left (OL) in corridor (top) and dead end (DE) in corridor (bottom).

Table 5.11.: Cross-validation: MSE on validating on the test trajectories

Context	Test MSE $\times 10^{-3}[\text{deg/mm}]^2$
OL	5.2
L	1.8
OR	2.7
R	1.1
S	0.27
DE	1.2

output to be predicted. Recurrent neural network is trained with Levenberg-Marquardt backpropagation algorithm. The training and validation of the network rests upon hold-out cross-validation with 60% of the presented trajectories are used for training, 20% for validation and the remaining 20% for testing. Table 5.11 shows the test MSE for the contexts. From the table, one can observe that straight line motion (S) is a trivial context to identify and predict followed by dead end (DE) and obstacle evading actions (L and R). Fig. 5.22 shows the original and predicted values for some of the test trajectories whose validation errors were shown in Table 5.11. The plots present a different perspective into the generalization. The generalization errors of OR and OL contexts from the table might tend to present a mediocre performance but in reality the essence of the motion is correctly predicted. Presenting RNN with multiple and different sequences of executing in a context approximates the different motions to generate a behavior instead of accurately imitating every demonstration example presented. The advantage of learning the sequence with RNN is that it captures the continuation of the previously matched sequence or

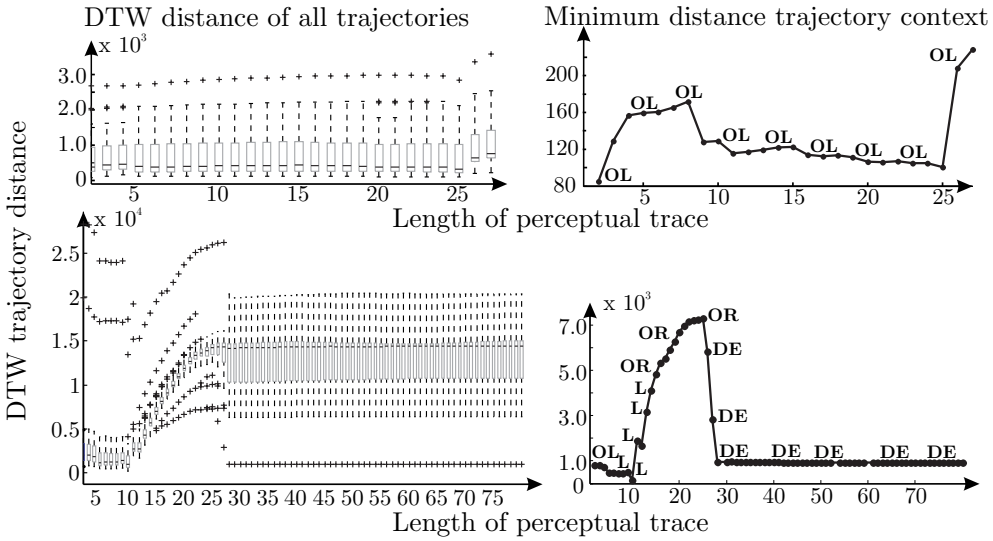


Figure 5.21.: Trajectory matching cross-validation with DTW. Turn left (OL) in corridor (top) and dead end (DE) in corridor (bottom)

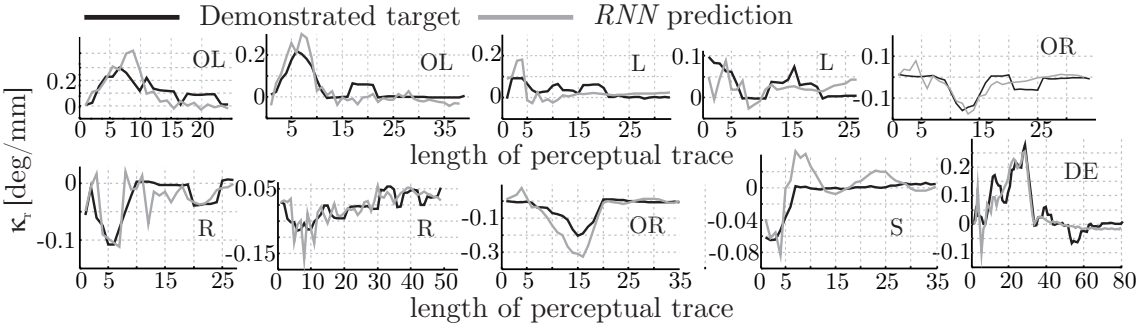


Figure 5.22.: RNN based one-step curvature prediction for test trajectories in different contexts.

subsequence to predict the next best curvature. This predicted curvature in conjunction with current translational velocity is then mapped into its appropriate turn rate.

5.3. Scenario modeling

As an alternative to perceptual tracing and prediction, we also investigate an approach in which the perception action relation is represented by a set of multiple ANNs trained on data from separate scenarios. Similar to the previous approach, a wrapper method with forward chaining identifies the subset of features that achieves the best generalization. Notice, that in contrast to the previous approach the problem is no longer a classification task but a regression problem. Feature selection and 5-fold cross-validation validation is performed with respect to multilayer feedforward network trained with Levenberg-Marquardt backpropogation. With forward chaining, candidate features are incrementally included in the input representation, until the MSE of the trained model on unseen test data no longer decreases with additional features. For every feature subset analyzed by forward chaining, tests are performed with different number of neurons and the one with the minimum MSE is registered. Table 5.12 shows the selected features for corridor, open room and cluttered scenarios. The training and validation rests upon hold-out cross

validation with 60% of the data used for training, 20% for validation and the remaining 20% for testing. The training, validation, testing and the NMSE between prediction and the true curvature for ANN are listed in Table 5.13. From the table we can observe that corridor and open room model generalize the best achieving a considerable improvement in the residual variance of the prediction by almost 35% and 27%. The generalization of the cluttered room examples is marginally inferior to corridor or open room mainly attributed to the large variance of the data. The generalization capability of the ANN models across different scenarios are also validated as listed in Table 5.14. The inter-scenario validation by definition, tests the ability of the learner to generalize training in one scenario onto a fundamentally different scenario. From the table, one can observe that the generalization error between potentially different scenarios is relatively larger than the intra-scenario validation (Table 5.13) predominantly due to the lack of similar examples. The increasing error trend in generalization between scenarios can be intuitively coupled to the complexity of the traversing and the testing environment. Open room being the least complex of the three scenarios as it encounters only isolated obstacles, finds hard to generalize with cluttered or corridor examples which necessitates a persistent motor correction to either avoid an obstacle or center in an corridor. Similarly, generalization error by testing open room examples with corridor and cluttered environment model is marginally inferior to their counterparts, nevertheless one has to consider that a right or a left turn in an open room instead of a go straight motor action might be detrimental to the generalization error but is still deemed acceptable as long as it avoids the imminent obstacles. In addition to the generalization capability, the false positive rate in predicting the correct curvature would establish a foolproof robustness of the model. This aspect is addressed in the next section where the robot is tested on seen and unseen scenarios and environments.

5.4. Experimental results

The scenario and context based visual navigation scheme is tested on the mobile robot in different indoor environments. Different combination of classifiers and trajectory matching schemes are tested. Experiments are run on a pioneer 3DX mobile robot in multiple indoor environments with different texture and geometry. In this section, we highlight the interesting results that establish and provide a proof of concept for the navigation architecture. The robustness of the schemes are determined by allowing the robot to wander for longer duration with frequent start with different initial configurations. The experimental validation of scenario classification in section 5.1.3 showed the performance of the different classifiers to unseen indoor environments. This section deploys the entire scenario based visual navigation scheme for test. Fig. 5.23 shows a prototypical scenario within a corridor. Starting from a misaligned position, the robot is tested with RF based scenario classifier working with LCSS based trajectory matching. The next action within the matched context is predicted by the context specific RNN. Starting from a similar position, the navigation scheme is also tested with the ANN scenario models. Both the schemes identify the scenario quite robustly and traverse along the center of the corridor. The classified scenarios in the experiment are shown with corresponding color code as seen in the figure. The scenario initially is classified correctly as corridor with a cluttered

Table 5.12.: Selected features for individual scenario modeling with ANN.

Feature	C	O	L
X-coordinate of elliptical centroid	x	-	-
Y-coordinate of elliptical centroid	-	-	x
Orientation error (beta)	x	-	-
Floor-no-floor ratio	x	x	-
Nearest distance	x	-	-
Nearest angle	-	x	-
Bilateral symmetry	x	x	x
Next Safe free floor direction	x	-	-
Ratio of the principle major and minor axis	x	x	x
Critical distance	x	-	-
Critical angle	x	-	x
Semi major axis	-	-	x
Semi minor axis	x	x	-
Multiscale entropy	-	x	-
X-coordinate of the mid pt of W_{max}	-	x	x
Kurtosis	-	-	x
Area-perimeter ratio	-	-	x
Thinness circularity ratio	-	-	-
Rectangularity	-	x	x

Table 5.13.: Intra-scenario cross-validation with scenario specific ANN

Scenario	Training MSE $\times 10^{-3}[\text{deg/cm}]^2$	Validation MSE $\times 10^{-3}[\text{deg/cm}]^2$	Testing MSE $\times 10^{-3}[\text{deg/cm}]^2$	NMSE	Bias MSE
C	6.7	8.3	6.1	0.65	6.9
O	12	14	10	0.73	14.3
L	9.7	7.7	7.6	0.85	8.6
Monolithic	11.1	11.4	8.5	0.87	10.7

classification at the end mainly due to the open room seen on either side of the robot. The figure also shows the context classified by LCSS. At the beginning the context is classified as OL, set to turn left to avoid the door posts at the starting configuration (See experiment (a) in Fig. 5.23), followed by smaller corrective actions to keep the robot aligned along the corridor center. Scenario specific ANN is also tested along the same corridor but from the adjacent side. The scenario classification is quite robust and the corresponding ANN model aligns the robot smoothly concurring well to the cross-validation performance. The second experiment is an extended scenario of the first experiment where the robot needs a smart interplay between corridor and cluttered scenario. Fig. 5.24(a) shows the scenario within a corridor environment. For this experiment we perform scenario classification using the RG and the robot curvature predicted by the ANN scenario model. The robot in Fig. 5.24(a) starts in the corridor little misaligned and has two door blocking the path way further down the corridor. The robot circumnavigates the obstacle course successfully. The two open doors at the end change the nature of the segmented floor there by changing the scenario classification to adapt. The figure also shows the recorded omnidirectional

Table 5.14.: Inter-scenario cross-validation with scenario specific ANN

Training	C	C	L	L	O	O
Testing	O	L	C	O	L	C
MSE $\times 10^{-3}[\mathrm{deg/cm}]^2$	13	14.4	13.5	11.2	10.7	19.7

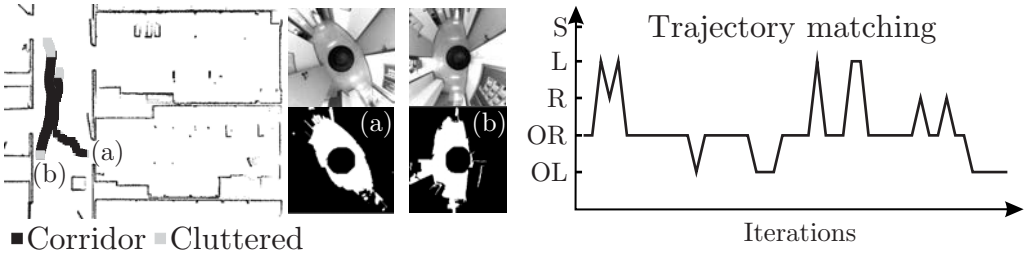


Figure 5.23.: Scenario: Untrained corridor. Starting from misaligned configurations, scenario classification is performed using RF and one-step curvature predicted using (a) context based RNN based on the trajectory matching results from LCSS and (b) scenario specific ANN. The context identified from matching the trajectories using LCSS is shown on the right.

image and the corresponding segmentation image at particular nodes. The classifier at node 1 identifies the scenario to be corridor but because of the open door to the left, the segmentation changes between open room and cluttered before steadying down. After getting past node 2, the segmentation image captures the door and changes the scenario classification to open room. The performance of the model within a cluttered environment is shown in Fig. 5.24(b) where the robot starts in a room cluttered with boards and obstacles. The robot classifies the environment to be open room until enough obstacles are segmented in the omnidirectional image. The correct classification of the impending scenario helps the robot to evade the obstacles and traverse the obstacle course smoothly. Furthermore experiments are performed in different environment with equally successful results. In summary, the experiments establish the performance the proposed architecture for navigation by adapting the behavior with respect to the traversing scenario. The test runs are validated with classifiers trained on visual features whose output determine the necessary model to deploy. The experiments can be equally tested with classifiers trained on principal components and also the model arising from unsupervised SOM network. The results also show that classifying the context within the scenario and using the information for making a decision is successful nevertheless, the scenario specific ANN exhibiting more robustness to noise in the environment. For more experiments and extended results check Appendix C.2.1

5.5. Related work

Augmenting geometric maps with semantic information has been an active research in computer vision and robotics for quite some time. The authors of [BS02] use virtual sensor as a combination of sonar and odometry information to identify rooms within an indoor environment. They perceive the identification of geometrical entities within an indoor environment as an open problem to be solved. This was alleviated by the works of [MMB06]; [MRTJB07] where topological maps from geometric maps are obtained using laser range sensor. Typical patterns of range sensor readings in indoor environments

are learned using Adaboost thereby classifying every point to a semantic class. Typical classification of the environment was done between a corridor, room and a doorway. Another similar work is from [NH08] where the authors use a combination of RANSAC and Iterative closest point algorithm to extract planes from 3D laser range data and label them. Our work broadens the spectrum by using only visual information to classify the environment. By segmenting the free floor in the immediate surroundings of the robot captured using an omnidirectional camera, patterns within different indoor scenarios are learned. The works mentioned above deal generally with Simulatenous Localization and Mapping (SLAM) or mapping task and are not with the behavioral navigation of the robot. In our case, the scenario classification is done from the perspective of the robot to navigate and wander within an unknown environment. Self-identification of places from images is described in [ZBK07], where the captured images are clustered on the fly to identify the different distinctive scenarios. Similar approach is done here with SOM with different classes to identify classes from the recorded data. The challenge in learning appropriate robot actions for different indoor scenarios is first to identify the underlying scenario and next to understand the context to execute the most appropriate action within the scenario. One approach involves matching previous memory traces with the current ones and identifying the contexts. DTW [BC94] has been the state of the art in matching time series data over the years. Nevertheless, when the two time series are translated, skewed and have missing data between the two trajectories then DTW performs poorly. One alternative is taken from string matching in computer science where the longest common subsequence is searched between the two trajectories. In [VKG05], a non-metric similarity function between two trajectory sequences are identified from the length of the common subsequences between them. This allows stretching, contraction of the trajectories and also handles missing data robustly. [HWSK09] use it for a long term vehicle motion prediction. The rotationally invariant metric allows a combined trajectory classification and a particle filter framework for a 3 seconds trajectory prediction into the future. Navigation for a robot within a highly dynamic environment without any a priori knowledge of the map does not need a long term prediction but a more deeper understanding of the context. For more experimental results with different classifiers in more diverse environments refer to Appendix C.2.1.

5.6. Summary

This chapter presented a new framework for acquiring visual navigational behaviors from demonstrations. The approach tackles the problem by identifying features that classifying the underlying indoor scenario the robot currently traverses and makes the appropriate motion decision pertaining to the scenario. Omnidirectional image acquired by the robot during demonstration are segmented for free floor regions. The binary segmentation image thus forms the starting point towards scene understanding. Diverse set of classifiers such as NB, RG, PNN and RF are tested for their accuracy against different representation of the segmented image. Computational image features that reflect the proximity of the obstacles around the robot, shape features that extract the geometry of the environment and statistical image moment features are all extracted and the most optimal combination for classifying an indoor scenario is tested. As an alternative, principal com-

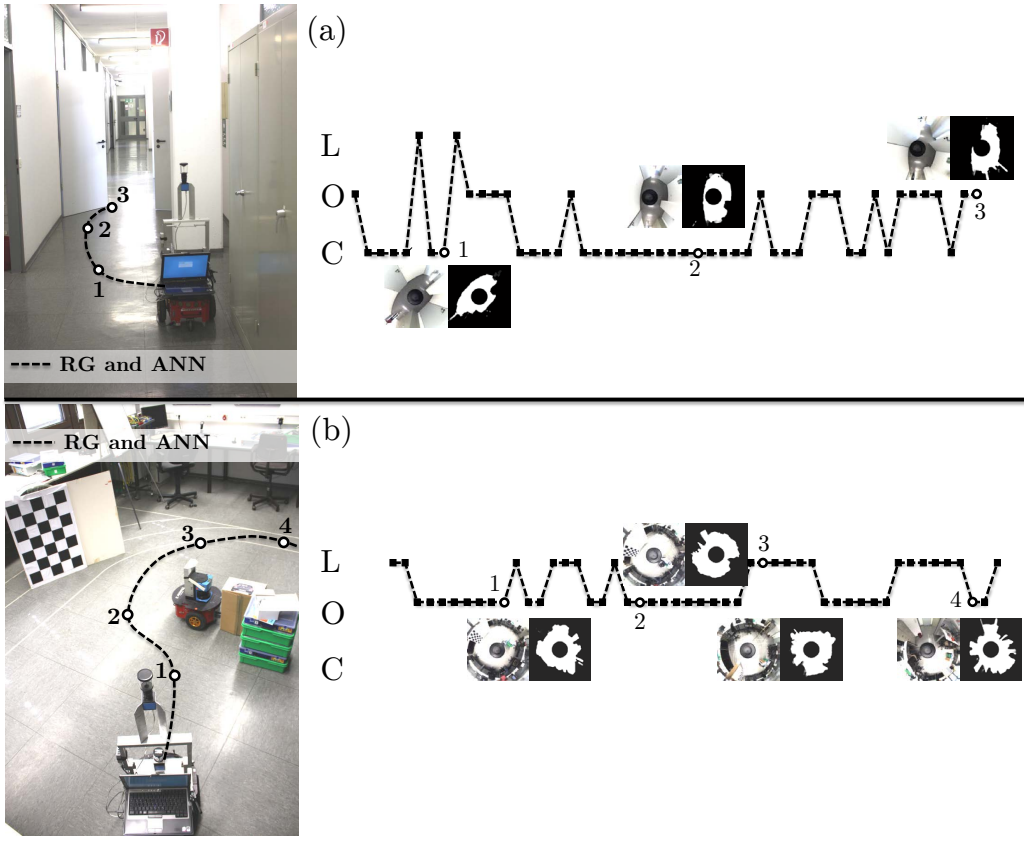


Figure 5.24.: Scenario: Untrained corridor and cluttered environment. (a) Blocked corridor scenario and (b) cluttered scenario. In both the experiments, scenario classification is done with RG with scenario specific ANN for predicting the robot curvature. Exhibited on the marked nodes are the captured omnidirectional image and the corresponding segmentation image.

ponents of the segmentation images are also modeled and tested. The experiments give an intuitive insight into the performance difference between the different classifiers and features. Furthermore, similar contexts within a scenario are also identified by comparing the visual perceptual trace of the robot with the demonstrated trajectories. Artificial neural networks to model individual scenarios are also analyzed whose generalization ability to seen and unseen environments are exhaustively tested and compared to trajectory matching. Experimental results demonstrate the ability of both trajectory matching and scenario based neural network model in unknown scenarios and transition between two environments.

The ultimate aim of mobile robots in service environments is to achieve a vibrant interactive environment for both human and the robots. Human-robot interaction is an active domain [FSM09] in modern robotics and human-aware motion-planning of mobile robots is one area which promises a lot of potential [SMUAS07]; [KPAK13]. The ideas from a situation aware navigation can serve as a foundation for further proximic robotic behaviors [MM11]. In addition to their basic capability to move autonomously in dynamic environments, socially competent robot should demonstrate a navigational behavior that appears natural to humans.

6

Supervised learning of behaviors with artificial neural networks

The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves.
– Herbert A. Simon, *The Sciences of the Artificial*

The previous chapter addressed learning robot behaviors by modularizing robot motion to different indoor scenarios and the context of operation. The emerging behaviors learn a fusion of multiple behaviors within the scenarios and context. This chapter focuses on the alternative aspect of modularizing robot motions by decoupling the scenario dependence and learn the behaviors directly. From the perspective of behavior learning with demonstrations, we attempt to answer two quintessential questions namely,

1. What is an efficient policy? and
2. How many demonstrations suffice to generate an efficient policy?

The first question tries to acquire an understanding to how to define an efficient policy? This is an open ended question. The answers typically tend to be highly subjective and opinionated. For example, a path following robot is considered efficient if it imitates a demonstrated trajectory exactly as shown. Precision and accuracy of imitation are of paramount importance for such tasks. In a behavior based framework, an efficient policy is one which can handle noise and unknown or unseen data sturdily. Generalization and robustness of imitation are of paramount importance in these tasks. We use the machine learning tools for policy evaluation together with the spirit of behavior based robotics to answer this question. The second question shapes itself from the first, where the number and type of demonstration needed for an efficient policy are confronted.

6.1. Visual behavioral features

Three behaviors namely corridor following, obstacle avoidance and homing are demonstrated. They are generated by guiding the robot from different starting positions using a joystick. Data acquisition and model validation are first performed in simulation before testing it on the real environment. The simulation framework offers a flexible platform to test the robot across diverse environmental configurations at ease. For details about

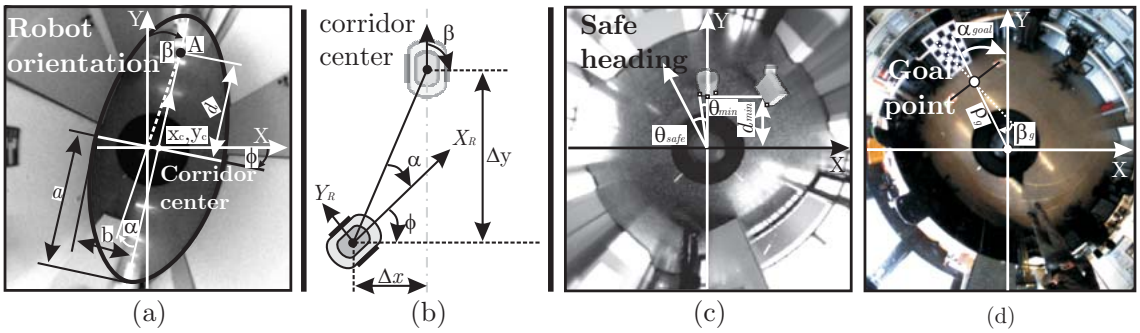


Figure 6.1.: Visual behaviors. (a) Corridor following features from image moments, (b) robot pose within a corridor, (c) obstacle avoidance features and (d) homing features

the simulation environment check appendix B.0.1. For training individual behaviors, the behavior features are manually selected. By carefully considering the nature of the behaviors, the relevant features are identified through human intuition. Fig. 6.1 shows the typical scenarios associated with the three behaviors from the perspective of the robot. The objective of corridor following is to drive the robot to the corridor center. It goes by the rationale that within a corridor environment in the event of no obstacles, the most optimal path for a robot is along the corridor center. This is similar to a wall following behavior where the robot needs to maintain a constant distance to the right or left walls. For a corridor following behavior, the objective is thus to maintain a constant distance to both the adjacent walls thereby driving the robot to the corridor center. Visualized from a birds eye perspective as seen in Fig. 6.1 (b), the behavior agent needs to control the rotational velocity of the robot such that the lateral distance to corridor center (Δx) and the orientation angle (β) is zero. On an omnidirectional segmented image, these variables are extracted by determining the image moments of the free floor region as described in chapter 3.2.2. The lateral offset angle (α) gives an equivalent measure of Δx and the final orientation is equivalently obtained from the measure β. Thus, the perceptual state $[\alpha, \beta] = [0°, 0°]$ constitutes the desired goal state with the ω at the goal state also converging to zero.

For an obstacle avoidance behavior, three features to the immediate front of the robot namely; nearest obstacle distance (d_{min}), nearest obstacle bearing (θ_{min}) and next safe traversable direction (θ_{safe}) are extracted. θ_{safe} is computed by searching radially outward from the robot front for a free floor direction whose traversable area extends a specified threshold D. The sine- and cosine- functions of these variables are utilized to fulfill the periodicity requirements for the heading direction thereby obtaining the perceptual features $\sin(\theta_{safe})$ and $\cos(\theta_{min})\frac{1}{d_{min}}$. The goal of the behavior is to converge the perceptual variables to zero. From a behavioral perspective this means to maintain a safe straight direction ($\theta_{safe} = 0°$) and keep the obstacles away from the robots path ($\theta_{min} = 90°$). The inverse of the minimum distance is taken to give more emphasis to obstacles that are closer. The rotational velocity ω, thus at $[\sin(\theta_{safe}), \cos(\theta_{min})\frac{1}{d_{min}}] = [0, 0]$ is also zero.

A typical visual homing scenario is depicted in Fig. 6.1(d). The homing landmarks are indicated by two distinctive red and green markers, whose mid point of the straight line connecting them is the final goal location. The two markers are placed on the floor approximately 2 meters apart. The homing behavior is a pure vision based behavior and the robot possesses no prior knowledge of the map or the location of the markers in the

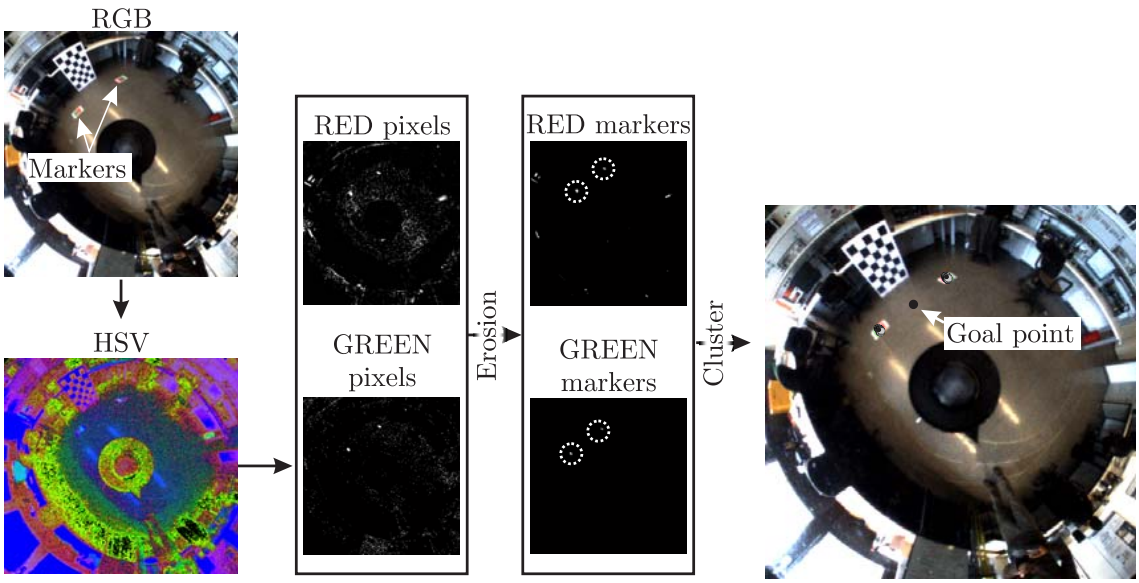


Figure 6.2.: Illustration of the steps involved with the goal point extraction.

map. At every time instance, the omnidirectional image is searched for landmarks and on identifying them, the angular direction of the goal point α_g is computed. Considering a line perpendicular to the one connecting the two markers, the angle between this perpendicular line and the one connecting the robot to the goal point β_g is computed. β_g is computed assuming that the final goal orientation is perpendicular to the line connecting the goal markers.
Additionally, the distance to the goal (ρ_g) is also considered. The aim of the behavior is to reduce ρ_g and α_g initially and then correct the orientation error β_g on close approach. The angles are transformed proportional to the normalized value of the distance measure to get two features namely,

$$\begin{aligned} \alpha_{gt} &= \alpha_g \frac{\rho_g}{\rho_{\max}} \\ \beta_{gt} &= \beta_g \frac{\rho_g}{\rho_{\max}} \end{aligned} \quad ,where \quad \rho_{\max} = \sqrt{{x_c}^2 + {y_c}^2}$$

where x_c and y_c are image center points and $\rho_{\max}$ is the maximum plausible distance of a goal point that can be detected. Prior to the feature extraction, the goal markers from the omnidirectional image are extracted. The Red-Green-Blue (RGB) image is first converted to Hue-Saturation-Value (HSV) color space. By setting thresholds for the red and green blobs, two binary images with segmented red and green marker are obtained. Unwanted pixels in the image are then morphologically removed with an eroding process. The rest of the available pixels are then clustered to find the final position of the markers. In order to avoid false correspondences between the two color clusters, a distance constraint is set on the detected red and green clusters such that only the closest cluster group is considered. Fig. 6.2 illustrates this process. Within a simulation, the position of the markers are known which are correspondingly back-projected on the corresponding omnidirectional image of the scene.

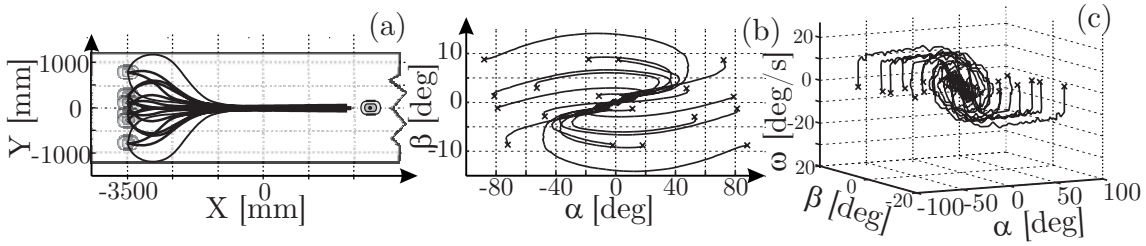


Figure 6.3.: Corridor following demonstrations in simulation. (a) Demonstration trajectories in a prototypical corridor, (b) the recorded perception variables (α vs β) and (c) perception variables vs recorded rotational velocity ω. The starting point of the demonstrations are indicated by a cross.

6.2. Static mapping

The outputs of a static or direct mapping depends only on the input presented and possess no memory. For a robotic behavior, this means that every state has one fixed action. A feedforward neural network with backpropagation (ANN) is one example of a static network.

6.2.1. Corridor following

A set of twenty demonstrations are generated using a joystick in simulation. The robot is placed at different misaligned configurations along the corridor and then driven to the corridor center. The configurations are set such that they cover majority of the state space spanned by the three variables (ω, α, β). Typically the demonstrations are performed on one side of the corridor and the examples mirrored to generate identical examples on either side of the corridor. For mirroring, all the three perceptual variables are simply multiplied with -1. Fig. 6.3 shows the pictographic representation of the behavior demonstration together with the registered variables. The performance validation of the ANN are performed by checking its training, validation and testing errors (MSE and NMSE). The generalization ability of the ANN against known and unknown data is done to establish the robustness of the model. Different variations of training data are used for designing the ANN. Various situations with different number of training trajectories are trained (See Appendix D.1). Tests show that by training on two examples with each from either side and the periphery of the corridor, the learned model generalizes well in driving the robot away from the walls but deteriorates when starting misaligned but close to the corridor center (See Fig. D.3).
This is alleviated a little bit in the next experiment, where four demonstrations; two from either side of the corridor are presented for training. Fig. 6.4 shows the selected trajectories for training. Training is again performed for different number of hidden neurons and it was found that with three neurons a NMSE of 0.44 on the training and 0.66 on the testing data is achieved. The selected training trajectories here reflect the two extremes robot misalignments. Starting at an orientation of $-80°$ facing the wall and an orientation of $+80°$ facing towards the corridor center. The generalization error on the testing set shows this, where a 34% improvement in the residual variance of the predicted test data is gained. Fig. 6.5 visualizes the generalization performance of the model against all the training and testing trajectories. Experiments show that more trajectories brings a pronounced improvement. But it does not answer the question to how many

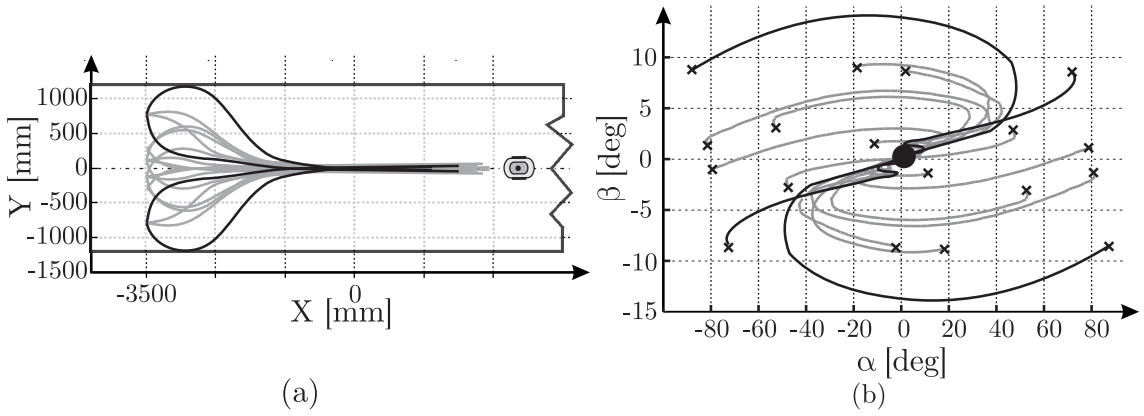


Figure 6.4.: (a) Four training trajectory demonstrations used and (b) the corresponding perceptual trace. Dark lines correspond to the training data.

demonstrations are adequate to learn the policy in a robust fashion. Hence, a one shot learning of the problem is addressed where only a single trajectory is used for modeling the policy. Fig. 6.6 shows the trajectory of the demonstration used for demonstration together with its perceptual trace. A single demonstration of traversing to the corridor center from the periphery is used. The reasoning behind the selection of this particular trajectory pertains to the logic that learning actions that avoids the corridor boundaries are more sensible and safer than the precision of reaching the corridor center. In other words, the essence of the behavior should repel the robot away from the corridor walls as soon as possible and drive the robot to the center. Out of the different number of hidden neurons, the model with two neurons exhibited the minimum NMSE of 0.822 against the unseen nineteen examples and 0.0669 against the single demonstration example. The MSE on the untrained examples show 6.67 deg/sec deviation from the target rotational velocity. Fig. 6.7 shows the trajectory predicted for some of the demonstrations. In terms of generalization of the model against unseen data, the one shot model exhibited similar testing performance as the model trained with two trajectories from the same side of the corridor (See Appendix D.1). Interesting to note here is that the generalization NMSE against unseen data is 3% better than the model with two training trajectories. This goes to show that theoretically one demonstration is enough to capture the essence of the corridor following behavior. But, the generalization ability of the model against novel unseen data suffers. Table 6.1 summarizes the training and testing errors of the three models. Important to note here is that the generalization errors arise from the MSE recorded between the predicted rotational velocity for the recorded perceptual state from demonstration. The influence of an error at one time step does not cumulate forward to the next step prediction. This is shown in Fig. 6.8, where the three models are deployed in the simulation for exact starting configurations recorded during the demonstration. The translational velocity is set constant (150 mm/s). The simulation results shows that all the three models mostly align with the corridor center quite well. The one shot learned model counters most of the presented situations very well except for the starting configuration positioned very close to the untrained section of the corridor. Interesting to note is that the ANN model trained with four demonstration examples minimizes the lateral offset error to the corridor center quite rapidly compared to the other model, but does not align perfectly to the center of the corridor. Both the one shot and 2 trajectory model, despite

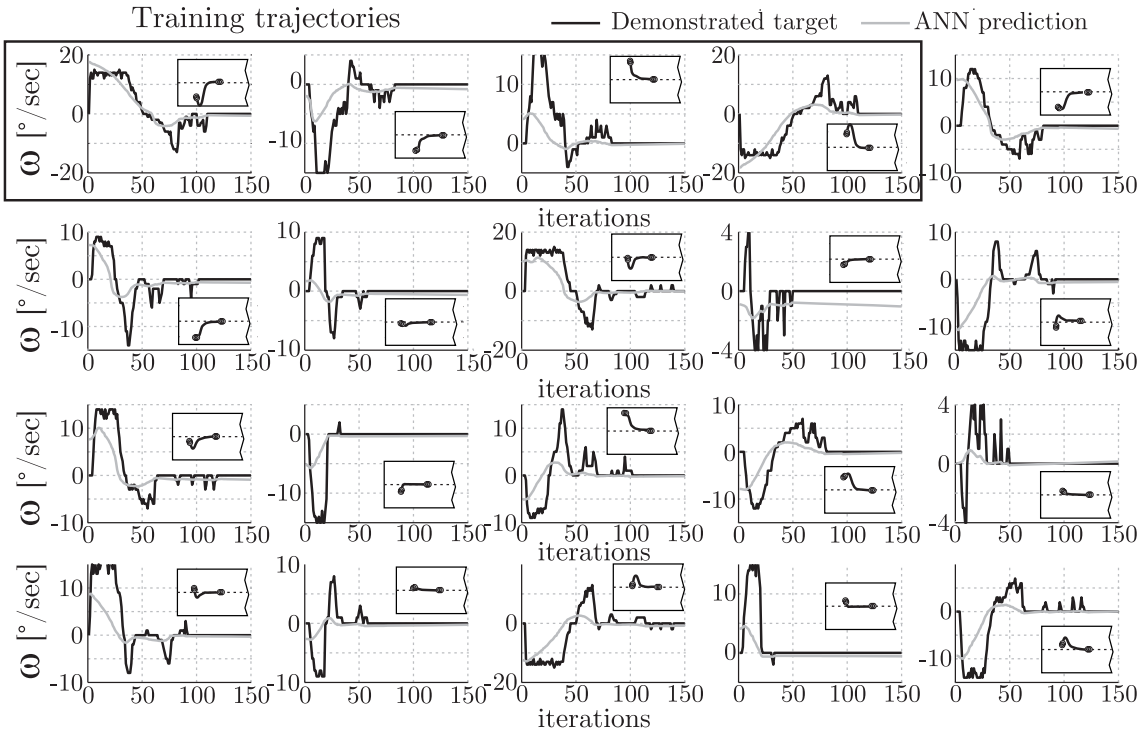


Figure 6.5.: ANN prediction for all the corridor following demonstration trajectories. The boxed trajectories are the training trajectories and the remaining sixteen trajectories are used for testing generalization.

Table 6.1.: Training and testing errors for the three different corridor following ANN models

Number of Training trajectories	Number of Hidden Neurons	Training MSE $[deg/sec]^2$	Training NMSE	Testing MSE $[deg/sec]^2$	Testing NMSE
Two	Five	2.54	0.05	6.37	0.89
Four	Three	10.1	0.44	8.04	0.66
One	Two	3.25	0.066	6.67	0.85

the periphery data provided as one training example, still gets precariously close to the walls before correcting itself. This behavior can be referred with a Hinton diagram of the network weights show in Fig. 6.9. Hinton diagrams helps to visualize the weights of the input and hidden layer of the network thereby giving an insight into the relative importance given to the perceptrons. The size of the blocks represent the magnitude of the perceptron weight and the color represents the sign of the weight. Positive weight is represented with a black block and negative weight with a gray box. The ANN trained with four trajectories interprets the input by giving very large weights to the orientation error relative to the lateral offset error, nevertheless the very low magnitude of the layer weights reverse the importance towards the lateral offset angle. Given the four trajectories for training, the network powered with the backpropagation algorithm settled down to weights, where accuracy in the lateral offset angle carries more importance than the final orientation angle. This configuration achieves the lowest MSE during training and model validation. The network weights of the other two models show a relatively larger weight given to the orientation angle both in layer 1 and layer 2. The simulation results of the experiment shown in Fig. 6.8 validated the model against unseen starting configurations

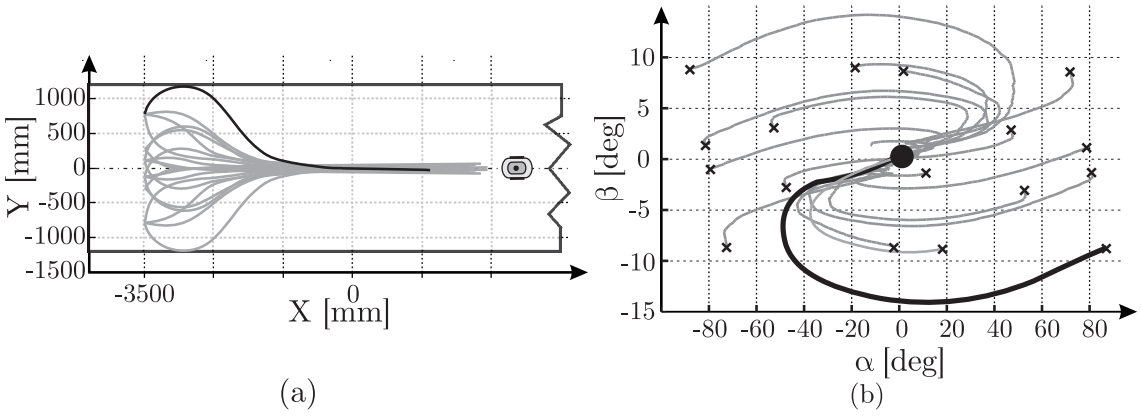


Figure 6.6.: (a) One shot learning training trajectory and (b) the corresponding perceptual trace. Dark lines correspond to the training data.

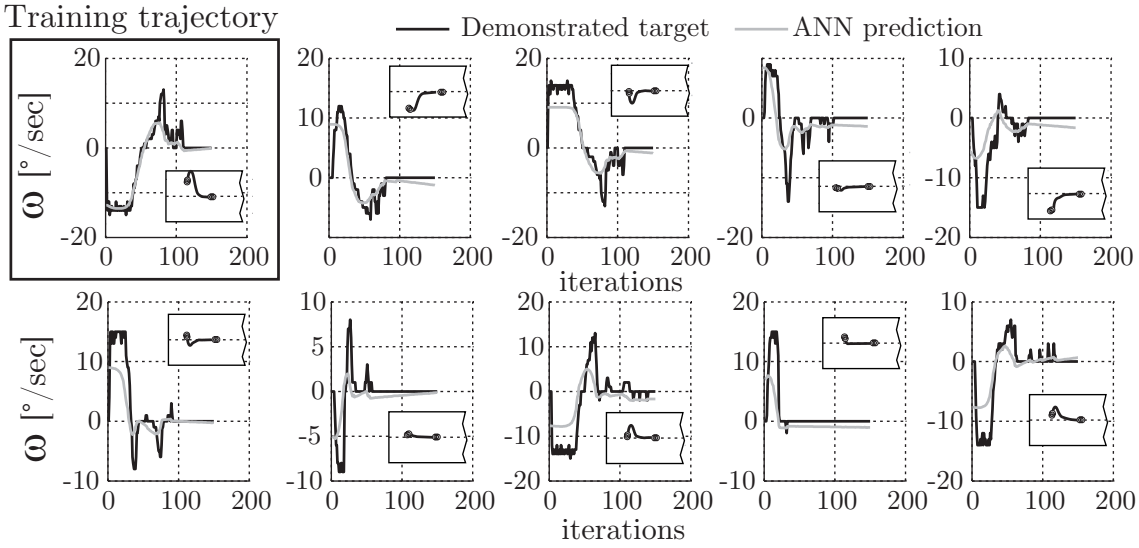


Figure 6.7.: ANN prediction shown for selected demonstrated corridor following trajectories. The boxed trajectory is used for training and the remaining trajectories are used for testing generalization.

albeit in a similar environment. The generalization of the model in completely different corridor scenarios is shown in Fig. 6.10. In the first scenario the hallway has varying width and orientation. The second scenario is a square path hallway with perpendicular corners. The third environment attempts to simulate the real world by using a distorted corridor. In the first two scenarios, the robot starts with an orientation of 0° whereas in the third scenario the initial orientation of the robot is 45° facing away from the corridor center. The results show that all the three models perform quite robustly to variations in the environment with slight difference in the motion pattern. Models trained with only one or two trajectories tend to react quick to a change in the environment. This is specially seen along the wall corners where the alignment of the corridor changes suddenly. The path traversed by the robot deployed with the neural network trained with only one or two trajectories drives too close to the corner whereas the model with more training data stays along the corridor center longer and maintains a safe distance during 90° turns. In a rugged corridor shown in Fig. 6.10(c), all the three models traverse the corridor in robust fashion; again with the neural network model trained with less trajectories reaching faster than the other. In summary, ANN generalizes the corridor following in quite a robust

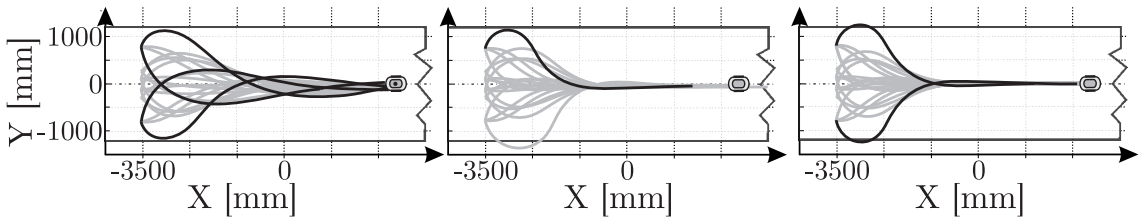


Figure 6.8.: Simulation results of the three models from the demonstrated configurations. The dark trajectory indicates the configuration on which the model is trained.

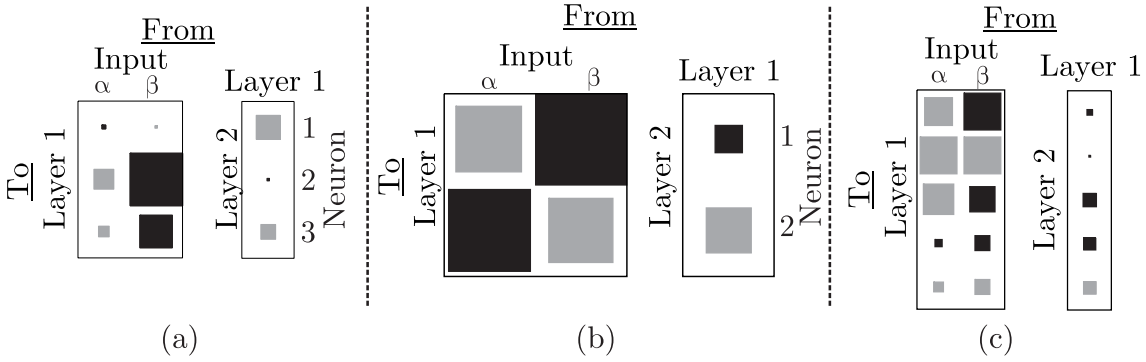


Figure 6.9.: Hinton diagram of the network weights. Shown are the weights of ANN trained with (a) four , (b) one and (c) two trajectories.

fashion. Because of the symmetric nature of the behavior, few demonstrations suffice to capture the essence of the behavior. Experiments show that single demonstration is enough to generate a corridor following policy. The downside of policies generated with very few trajectories tend to be very sensitive to changes in the environment. Training with more example trajectories drives the robot along the corridor center longer and keeps a safe distance to the wall all through out the experiments. The experimental results of the behaviors in real indoor corridor environment is presented in section 6.4.

6.2.2. Obstacle avoidance

Two sets of demonstration are performed in conjunction to obstacle avoidance. The first set of demonstrations focuses on the rotational velocity of the robot while evading obstacles. Different configurations of obstacles are placed in front of the robot and the user deviates it away from the hindrance. Eight demonstrations of obstacle avoidance behavior are generated. The second second of demonstrations correspond to a stop behavior also performed with a joystick. Fig. 6.11 shows the obstacle avoidance demonstrations. The recorded robots rotational velocity are also transcoded to their corresponding path curvatures κ. The advantage of learning curvature over rotational velocity is that for different maximum velocities, the turn rate from curvature does not under or overshoot the intended path. The stop behavior demonstrations are done with the robot starting at a safe distance to the obstacle in front and moving forward to slowly reduces the speed on closer proximity. The slowing down distance and the stopping distance is not previously set. Two demonstrations of the stopping behavior is thus generated.
We focus on two scenarios used to learn the turn rate of the robot. Two training trajectories one for left and right turn to evade an obstacle are presented first followed by, four

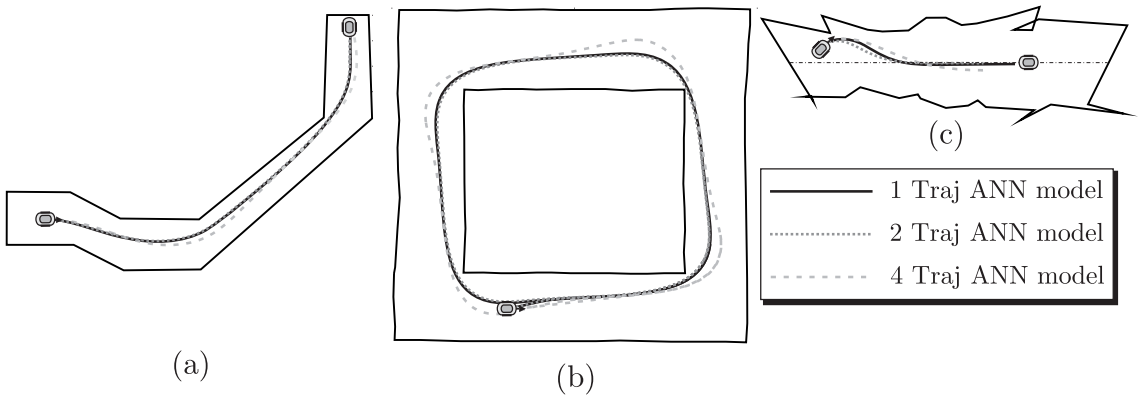


Figure 6.10.: Model reproductions in unknown and noisy corridor scenarios.

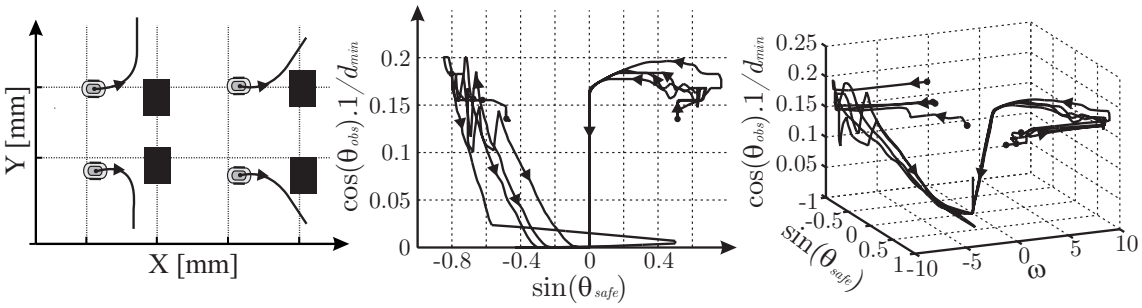


Figure 6.11.: Obstacle avoidance demonstrations. (a) Pictorial representation of some of the demonstrations shown, (b) the recorded perception variables $\cos(\theta_{min})\frac{1}{d_{min}}$ and $\sin(\theta_{safe})$ and (c) perception variables vs recorded rotational velocity ω. The starting point of the demonstrations are indicated by a dark circle and the arrows show the direction of the trajectory.

training demonstrations of turns in only one direction. For each training set, different number of neurons ranging from one to five are used build the model and are tested for generalization against the rest of unseen trajectories. The one with the minimum test NMSE is picked as the best. The ANN is trained for both output cases of predicting ω and κ. Table 6.2 shows the bias and the test errors of the network. For details into the selection of the appropriate number of neurons for the model, refer to the Appendix D.1. The table shows that given two trajectories, the network is able to generalize well with the other unseen starting configurations. The generalization error of the model trained with four trajectories with all performing a right turn, is very poor on demonstrations that require a left turn. Fig. 6.12 shows the rotational velocity prediction on both seen and unseen demonstration trajectory with the first model. From the figure one can see that

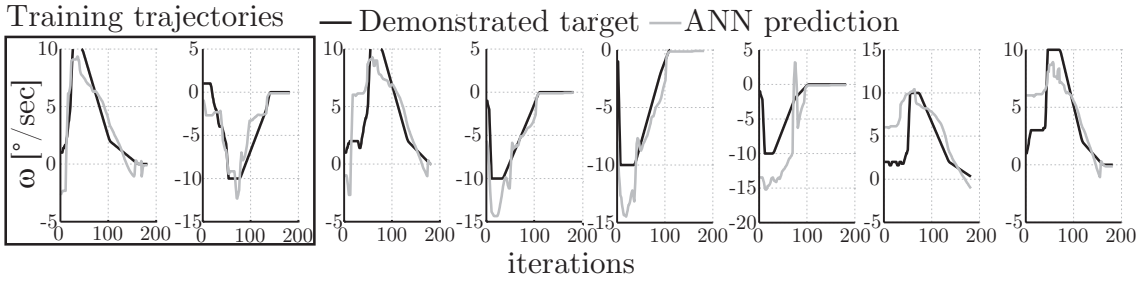


Figure 6.12.: ANN prediction of the rotational velocity for all the obstacle avoidance demonstration trajectories. The boxed trajectories are used for training and the remaining are used for testing generalization.

Table 6.2.: Obstacle avoidance: Training and testing errors for the different ANN models

Predicted output: rotational velocity ω					
Number of Training trajectories	Number of Hidden Neurons	Training MSE $[deg/sec]^2$	NMSE	Testing MSE $[deg/sec]^2$	NMSE
Two	Three	2.46	0.09	7.33	0.16
Four	Four	2.12	0.15	21.3	4.01
Predicted output: curvature κ					
Number of Training trajectories	Number of Hidden Neurons	Training MSE$\times 10^{-5}$ $[deg/mm]^2$	NMSE	Testing MSE$\times 10^{-5}$ $[deg/mm]^2$	NMSE
Two	Four	5.89	0.048	32.5	0.16
Four	Five	11.1	0.20	1620	5.56

with the knowledge of one right and one left turn suffices to generalize the other scenarios used in the demonstrations quite robustly. For learning the translational velocity of a stop behavior, the critical distance (d_c) to the obstacle is used. The translational velocity recorded during the demonstration is normalized to its maximum velocity ($v_{op} = \frac{v}{v_{max}}$) and used as the output of the model. From the two demonstrations, one is used for training and the other for testing. The relationship between the critical distance and the normalized translational velocity can be easily inferred and programmed as linear. This is a relatively simple problem, in that with only one explanatory variable a linear regression or a perceptron network would suffice. Nevertheless, when controlling a stop behavior with a joystick, humans do not maintain a well laid out linear pattern. The braking distance and the stopping distance differs between the users. Fig. 6.13 shows the recorded and the predicted velocity output for both perceptron network and a linear regression model. Table 6.3 shows the training and the testing error for both the models. From the errors one can infer that both the methods approximate very similarly and thus either can be used. Hence, for further experiments we choose to work with the ANN model. Fig. 6.14 shows the architecture of the obstacle avoidance behavior and trajectory traversed by the robot starting from the same initial configuration as the demonstrations. The experiments are performed with two maximum translational velocities of 150 mm/s and 60 mm/s. By maintaining the same maximum translational velocity limit of 150 mm/s used in demonstrations, the ANN model predicting the rotational velocity evades the obstacle very fast and robust. Nevertheless, when the velocity constraints are changed, there remains a disconnect between the speed at which the robot traverses and the rate at which the robot turns. This is overcome when the model outputs curvature, where the current and the maximum translational velocity constraints can be directly used to compute the appropriate rotational velocity. Fig. 6.15 shows different ad novel environments used for validating the robustness of obstacle avoidance. The first scenario is modeled in a corridor where an obstacle and a dead end scenario is simulated. Both the curvature and rotational velocity models trained on simple evading maneuvers in an open space environment counter the obstacles quite robustly with the model predicting rotational velocity slightly faster than the one predicting the path curvature. The second scenario is a room cluttered with obstacles. The size and shape of the obstacles are intentionally made

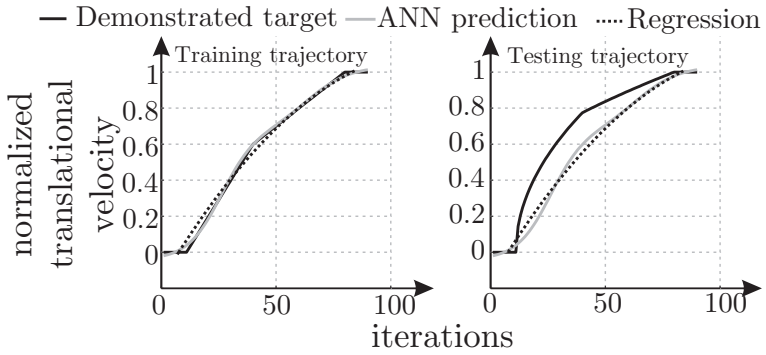


Figure 6.13.: Obstacle avoidance translational velocity predictions on the training and testing demonstrations.

Table 6.3.: Translational velocity training and testing errors of ANN and regression model

Fitting method	Training		Testing	
	MSE $\times 10^{-3}$	NMSE	MSE	NMSE
ANN	0.14	0.0012	0.0193	0.16
Regression	1.3	0.0108	0.0197	0.16

much smaller than that of demonstration. In spite of the good performance by both the models, there is one marked difference with the curvature based ANN model operating with 150 mm/s maximum velocity. After the first successful evasion of the obstacle, the model takes relatively longer time to counter sharp turns around the obstacles around the corner (especially the top right, bottom left and top left obstacles). This is due to the relatively higher approach velocity of the robot to the corner during which the predicted curvature is still a straight line path. Getting closer, the ANN model evades the obstacle with a sharp turn which takes longer duration to complete. The experimental results of the behaviors on the real robot within dynamic indoor scenarios are shown in section 6.4.

6.2.3. Homing

Eighteen demonstrations of different lengths are used for demonstrating homing behavior. Starting from different configurations, the teacher guides the robot to the center of the

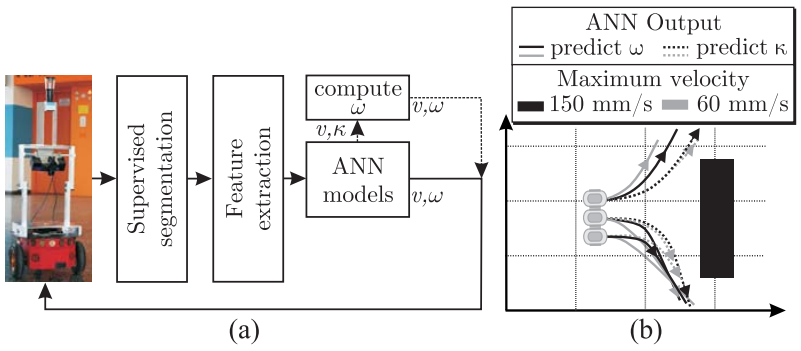


Figure 6.14.: (a) Obstacle avoidance architecture and (b) trajectory traced during imitation starting from the same initial configuration as demonstration.

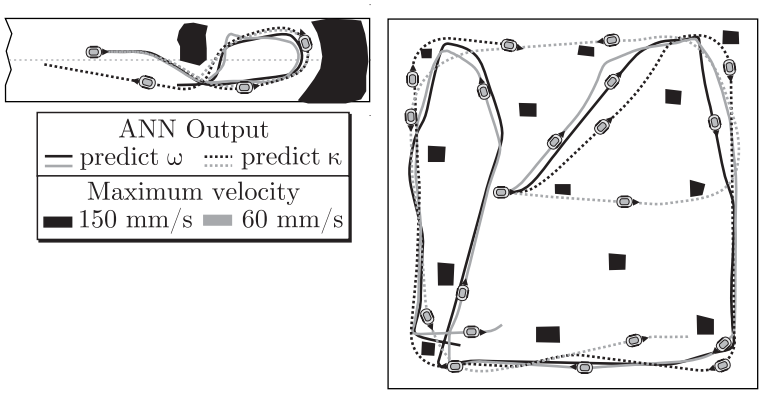


Figure 6.15.: Obstacle avoidance: Cross validation in an unseen environments.

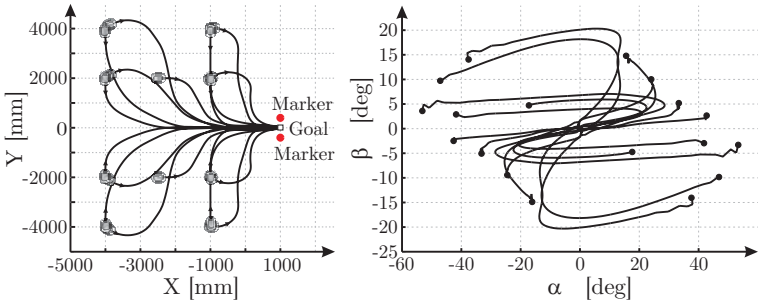


Figure 6.16.: Homing demonstrations in simulation. (Left) trajectories demonstration and (Right) the corresponding perceptual variables

line connecting the docking markers. Fig. 6.16 shows the recorded trajectories and the corresponding perceptual angles α_g and β_g. The behavior needs to capture two modes of approach to the homing area. First, the initial approach to the markers and then the final approach to align correctly. The first phase typically minimizes the distance to the goal (ρ_g) and the aligning the goal point direction (α_g) to zero. Closer to the homing area, the final alignment error is adjusted. The translational velocity of the robot is reduced linearly to ρ_g. The translational velocity is executed with the same model as in obstacle avoidance except that the critical distance to the obstacle is replaced with the distance to the goal. As with the previous behaviors, different sets of training trajectories are analyzed for generalization. Two models are discussed here. The rest of the different combinations of the trajectories used for training can be found in appendix D.2. The first set consists of taking all the input trajectories for training. The second set is with demonstrations starting from the farthest point to the goal. Different number of neurons ranging from one to nine are tested on the training data and the ANN with seven neurons was found to have the least NMSE validation error. Similarly, for the second data set the best generalization is achieved with five neurons. Table 6.4 shows the recorded MSE and NMSE obtained during training. The table shows that training MSE with all the trajectories is higher than the one trained on only two trajectories. This is because of the larger number of training examples with the first model compared to very few examples with the second one. Nevertheless, the test NMSE with the two trajectory model shows a poorer generalization. This is seen in Fig. 6.17 where both the models are tested in simulation from the demonstration starting configurations. The imitation trajectories show that both the model drive the robot to the goal with the path taken clearly indicative

Table 6.4.: Training and testing errors of ANN

Number of Training trajectories	Number of neurons	Training MSE $[deg/sec]^2$	Training NMSE	Testing MSE $[deg/sec]^2$	Testing NMSE
18	7	8.93	0.42	-	-
2	5	0.4093	0.024	19.85	1.3

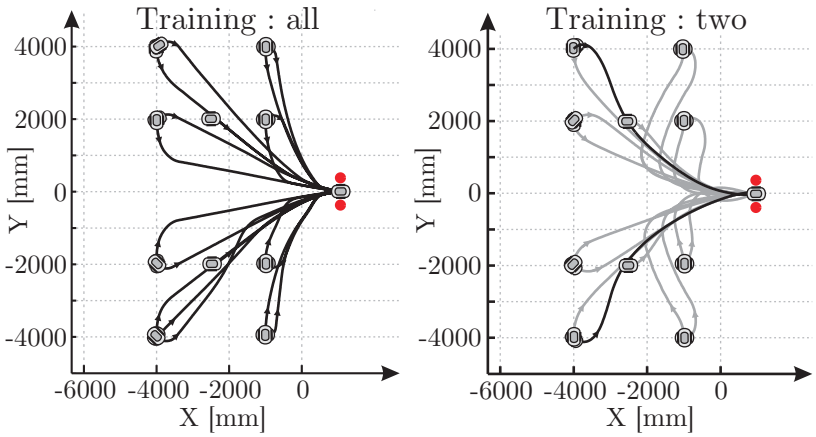


Figure 6.17.: Homing imitation results of the two models from all the initial configuration shown during demonstrations. The dark trajectory indicates the configuration on which the model is trained.

of the knowledge acquired from training. Training with all the trajectory, the first model quickly compensates the bearing angle α_g and then corrects the orientation error β_g. This is quite clearly learned from the abundant training trajectories provided. The second model does not possess this knowledge in that only two trajectories starting farthest from the goal are demonstrated. This is evident from the trajectories traced where the robot orientation error is first corrected and then driven to the goal. Each of the simulation is executed for longer duration to allow enough time for the model to reach the goal. Table 6.5 shows the average feature errors at the end of the imitation together with the absolute angular errors recorded. The model trained with all the trajectories reaches the target location with approximately one degree precision in bearing and orientation and clearly outperforms the model trained with only two trajectories. In spite of the performance differences, the performance of the two trajectory model still can be deemed satisfactory. The robustness of the models is tested further in the next experiment where the robot is along the periphery of a circle around the goal point. The initial configuration of all the positions are set to zero degrees. The positions to the right of the goal point specifically here correspond to scenarios not shown in the demonstration to either of the two models in that they simulate situations where the goal point is behind the robot. Fig. 6.18 shows

Table 6.5.: Average feature and orientation errors of the imitation

Number of Training trajectories	Average angular errors α_g [deg]	Average angular errors β_g [deg]	Average feature errors $\alpha_g \frac{\rho}{\rho_{max}}$ [deg]	Average feature errors $\beta_g \frac{\rho}{\rho_{max}}$ [deg]
all	-1.02	0.95	-0.06	0.04
2	2.78	-3.24	0.17	-0.20

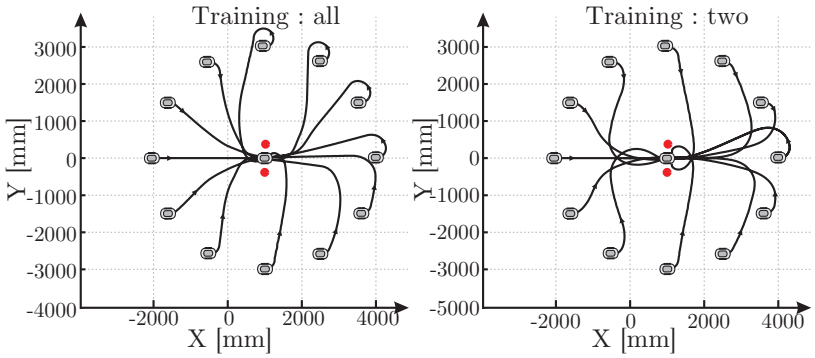


Figure 6.18.: Homing Imitation: Path traced by the robot when starting from positions placed along a circle around the goal point.

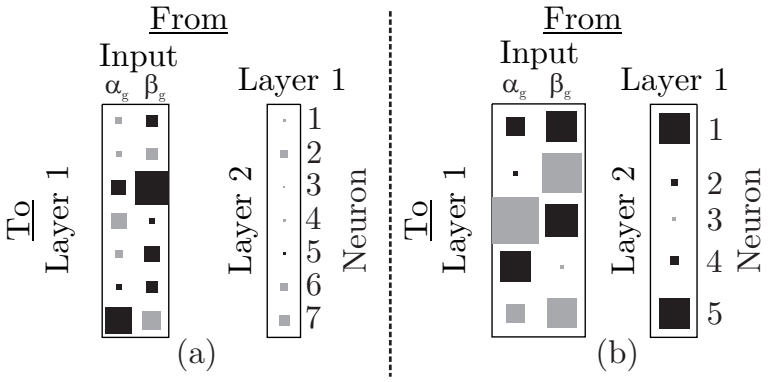


Figure 6.19.: Hinton diagram of the network weights. (a) ANN model trained with all demonstration trajectories and (b) ANN model trained with two demonstration trajectories.

the path traced by both the models. The path show that all the trajectories go towards the goal in a smooth fashion. Some of the interesting characteristics of the results here are; starting from a very flat angle, the model trained on only two trajectories typically overshoots the center line before aligning itself to the center, whereas the fully trained model does an about turn and approaches the goal robustly. The initial turning direction of both the models when starting from the right side of the goal is also curious in the sense, the model trained on all the trajectory always turns left to approach towards the goal. This is mainly because of the model characteristic where the bearing angle is first reduced followed by the orientation angle. This apparent difference between the two models can be seen in the Hinton diagram of the network weights (Fig. 6.19). Please note that the proportion of input weights assigned by each neuron to the feature is either subsumed or shot up with the layer weights. For the second model on the right side, one can see that neuron three and four has big positive and negative weights on α_g, nevertheless the corresponding layer weights are very small that it subsumes the high impact of these neurons. Correspondingly, the layer weights of neuron one and five are big whose input neurons give higher weights to β_g explaining this behavior.

6.3. Dynamic mapping

In dynamic mapping, the model output depends on the input presented while also preserving a memory of the previous internal states visited. Recurrent neural networks are

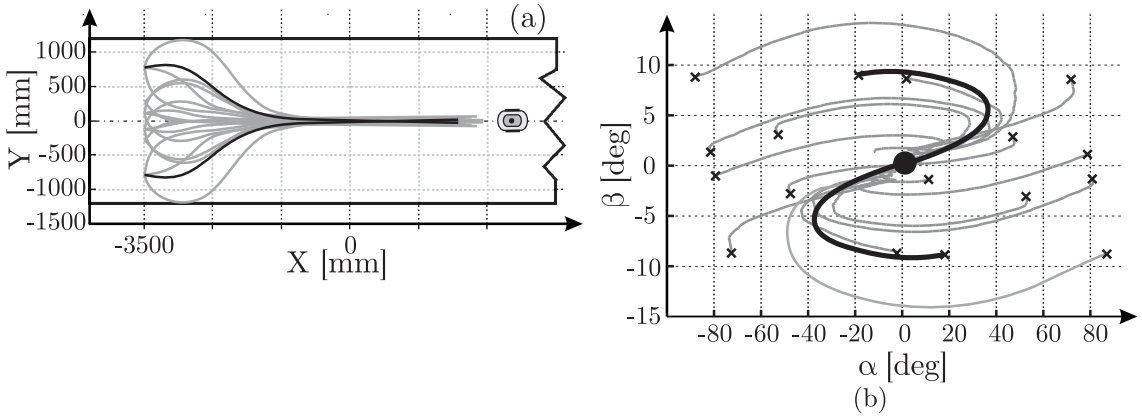


Figure 6.20.: Demonstration set used for corridor following. (a) The demonstrated trajectory used for training is shown in dark and (b) the corresponding perceptual trace.

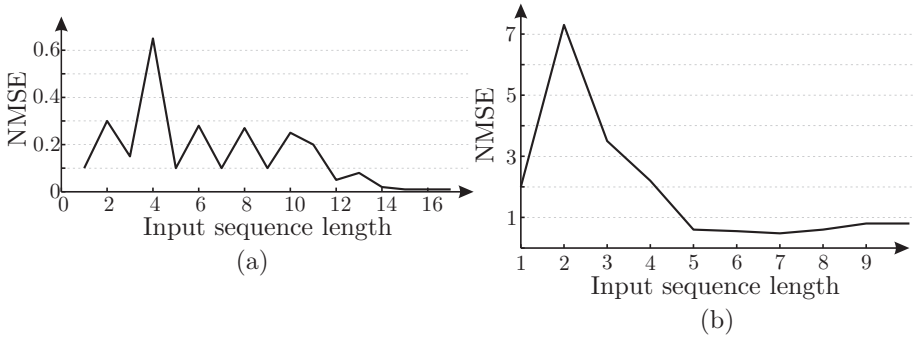


Figure 6.21.: Testing errors recorded for RNN trained with different input sequence lengths. (a) Corridor following and (b) Obstacle avoidance.

a type of dynamic networks where the neurons in hidden layer are connected with each other thereby creating an internal state which preserves the dynamic temporal behavior of the data presented. The training data are given as a multiple pseudo time-series data and the sequences are fed to the network concurrently at each time step. The RNN consists of two layers with a tangent sigmoid activation function in the first hidden layer and a linear activation function for the output layer.

Corridor following

We use the same demonstrations as in feedforward ANN. Two demonstration trajectories presented from either sides of the corridor are used for training. Fig. 6.20 shows the demonstration trajectories used together with the perceptual trace. RNN is setup with one hidden layer with 20 neurons and trained using conjugate gradient backpropagation algorithm. The network is tested for different input sequence lengths and the most optimal is chosen based on the cross-validation error registered on unseen examples. Fig. 6.21 shows the registered NMSE for different sequence lengths ranging from 1 to 20. From the figure we can see that for a sequence length of 16, the NMSE is the lowest. By providing an input sequence the context of operation of reaching the current state is also considered during training. The trained model is cross-validated for generalization against trajectories not shown during demonstration. Table 6.6 shows the mean MSE and NMSE recorded for unseen data. From the errors one can notice an improvement in the test

Table 6.6.: RNN training and testing errors recorded.

Behavior	Training		Testing	
	MSE	NMSE	MSE	NMSE
CF	0.011 $[\text{deg/sec}]^2$	0.003	0.27 $[\text{deg/sec}]^2$	0.25
OA	2.1 $\times 10^{-4}[\text{deg/mm}]^2$	0.02	0.0017 $[\text{deg/mm}]^2$	0.48

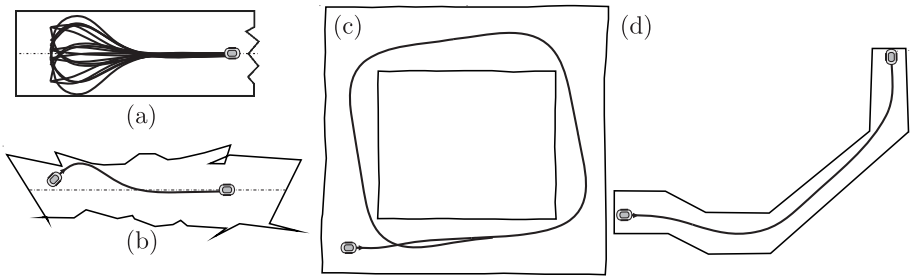


Figure 6.22.: RNN model reproduction in unknown and noisy corridor scenarios

data generalization compared to static mapping with feed forward network. With the addition of more trajectories into training, this performance further improves. Fig. 6.22 shows the successful imitation trajectory in unseen corridor scenarios.

Obstacle avoidance

Three demonstration trajectories are used for training obstacle avoidance behavior. The demonstrations correspond to two left turns and one right turn. For stopping behavior demonstrations, we use the same model as ANN. Based on the observations before, we learn the path curvature for determining the turn rate of the behavior. Similar to corridor following, the optimal sequence length to train the network is found by testing the network at every iteration against the untrained examples. The NMSE of the process is shown in Fig. 6.21(b). From the figure one can see that for a sequence length of 5, the network generalizes the unseen trajectories with a NMSE of 0.48. The small sequence length needed is understandable since obstacle avoidance is a highly dynamic behavior where only the immediate past is of importance. Table 6.6 shows the training and the testing errors for RNN with a sequence input length of 5. Fig. 6.23 shows the trajectory traced by the obstacle avoidance RNN in unseen environments. The model generally performs well to

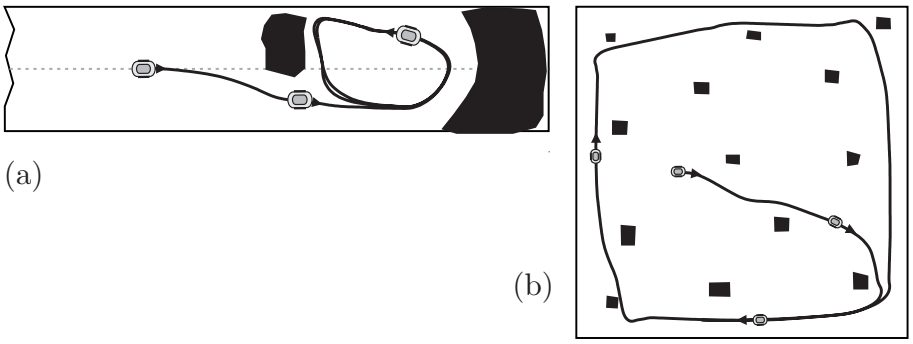


Figure 6.23.: RNN model reproduction in unknown obstacle avoidance scenarios in (a) corridor and (b) cluttered environment.

avoid obstacles in a robust fashion but suffers at narrow passages. This can be seen in Fig. 6.23 where the robot does not come out of the passage and loops around. The direction of approach and the next safe direction detected by the robot coming out of the dead end always points away from the passage exit thereby looping the robot back to the same place it visited. The set of demonstrations shown for training did not contain situations replicating this exact situation thereby making the network ignorant of such maneuvers. A static network such as ANN learns a direct mapping of a single feature set to the output and does not care about the previous perceptual trace, whereas a recurrent dynamic network places an attractor at places that concur to the perceptual traces shown during demonstration. In a more complex cluttered scenario but devoid of such corners, the network wanders quite robustly by avoiding all the obstacles encountered. Experimental results of the learned recurrent network is discussed later in the chapter.

6.4. Experimental results

Inferring from the simulation results, realistic experiments for the different behavior representative models are performed on the Pioneer 3DX mobile robot. Sixteen demonstrations of corridor following, fourteen demonstrations of homing and twelve demonstrations of obstacle avoidance are recorded.

6.4.1. Artificial neural network

Fig. 6.24 shows a corridor scenario where the robot is starts with a small lateral offset and a larger orientation error. The translational velocity is set constant here to 60 mm/sec. The maximum rotational velocity is limited to 10 deg/sec. The figure shows that the model drives the robot quite robustly to the corridor center. The evolution of the perceptual variables together with the predicted action is also shown where one can see the values going to zero indicating the robot has reached the center of the corridor. Two scenarios for obstacle avoidance are shown in the figure. The first situation is a corridor dead end scenario and the second scenario is a typical narrow corridor scenario. In the second scenario, the path way of the robot is blocked by an open door on the left followed by a pillar immediately thereby needing the robot to avoid both as a slalom obstacle avoidance. In both the cases, the robot successfully evades the obstacles in a robust fashion. The associated omnidirectional images are also shown in the figure to visualize the perception of the robot more clearly. In the first case, the robot proceeds straight until the dead end is detected via the segmentation result. The translational velocity is correspondingly reduced and the robot performs an about turn to avoid and proceed further. The second experiment similarly detects the door first triggering a large turn to the right and on detecting the wall turns left to the center of the floor.
Homing behavior is tested from different configurations from the goal point. The markers for the experiment are placed in an elevated position compared to the demonstration where they are placed on the floor. For the learned behavior this change should not make any difference. The behavior can be extended to any different kind of markers as long as there exists a method to extract the position of the homing markers on the image. The experiments exhibit a good performance of the behavior to approach the goal point. The figure shows two such instances of experiments. The first experiment starts from

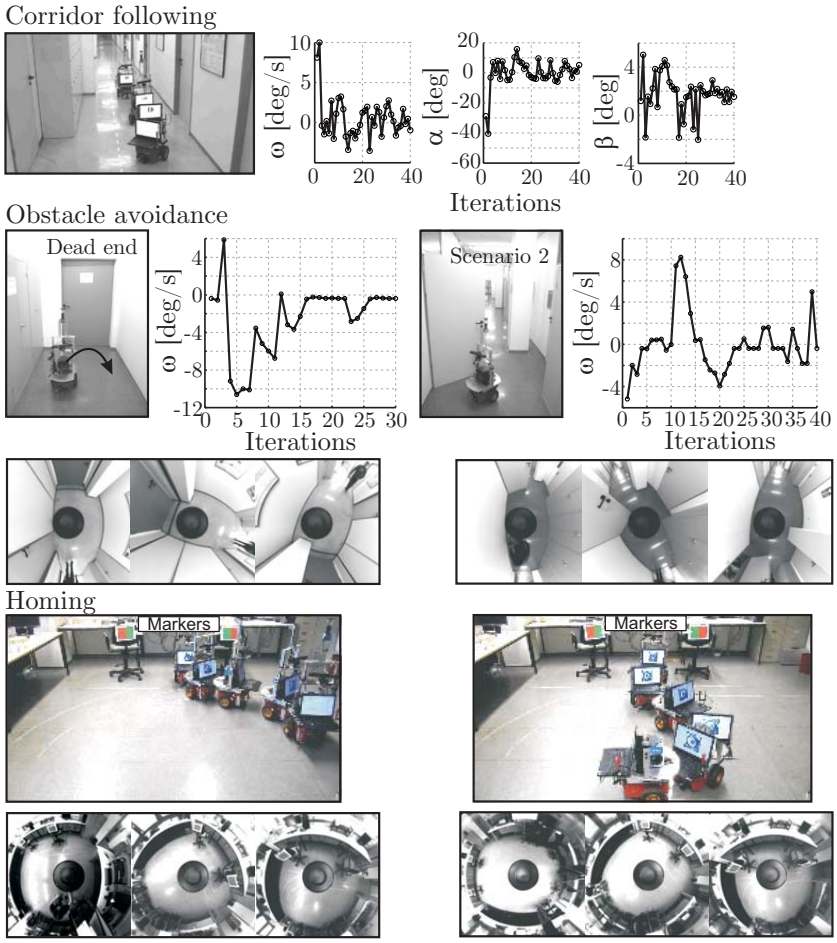


Figure 6.24.: Experimental validation of learned visual behavior using feed forward ANN. (Top) Corridor following, (middle) obstacle avoidance and (bottom) homing.

a very flat angle from the left of the goal and the second experiment starts straight in front of the goal but misaligned by almost 90°. In both the cases, the bearing angle is first compensated before the final orientation angle is adjusted closer to the goal point. The translational velocity of the robot is controlled in accordance to the distance to the goal point. Note, that all the features here are extracted purely from the images thereby annulling any requirements of the Cartesian position of the robot in the environment. The experimental results performed with learned ANN models show that robot is able to generalize to dynamic and diverse scenarios not presented during training. With any of the conventional or learned behavior coordination schemes, the three behaviors can be robustly coordinated for autonomous navigation. Some of the coordination experiments are explained later in the chapter.

6.4.2. Recurrent neural network

Corridor following is performed at a constant translational velocity of 60mm/sec. Fig. 6.25 shows the two experiments performed with the corridor following RNN. Corridor following is performed with a total input sequence length of 16 as explained in section 6.3. The experiments are started with the initial state offset angle and orientation error of $[\alpha, \beta] = [-42°, 10°]$ and $[53°, -6°]$. The figure also shows the evolution of the

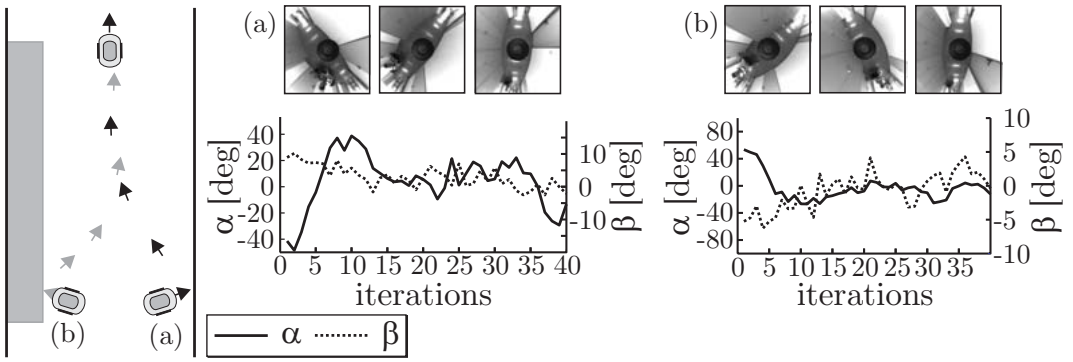


Figure 6.25.: Experimental validation of learned corridor following using RNN. (a) Initial configuration of the robot with $[\alpha, \beta] = [-42°, 10°]$ and (b) with $[\alpha, \beta] = [53°, -6°]$. Shown are also the evolution of the perceptual variables during the experiment.

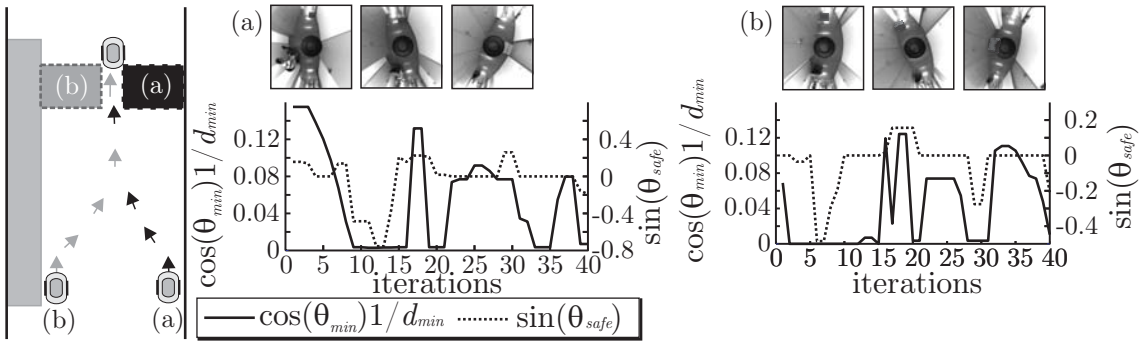


Figure 6.26.: Experimental validation of learned obstacle avoidance using RNN. Shown are also the evolution of the perceptual variables during the experiment.

perceptual variables α and β during the experiment and can see them being driven to zero corresponding to the corridor center. Obstacle avoidance is tested on the same scenario but this time with big box placed along the corridor acting as an obstacle. The obstacle is positioned directly in front of the robots path thereby making a forced evasion to the right and pass the narrow way leading further to the corridor. Fig. 6.26 shows the obstacle avoidance experimental result together with the evolution of the perceptual variables $\sin(\theta_{safe})$ and $\cos(\theta_{min})1/d_{min}$. Omnidirectional image of the scenario is also shown in the figure. The translational velocity of the robot is controlled directly by the corresponding RNN whose output decides the turn rate based on the path curvature predicted by the second RNN. Multiple experiments are conducted in different corridors and configurations of obstacles for a longer duration and successful results are recorded. The curvature output of the model depends highly on the segmentation input provided. If the data obtained from the segmentation is noisy, the generalization performance of the model tends to deteriorate irrespective of the prowess of the algorithm. The presence of a context layer and an internal memory overcomes this in that the sensitivity to abrupt changes or noise in the data is handled better compared to static networks. The experimental verification with the real robot in an actual dynamic environment establishes thus the capability of a dynamic RNN in learning visual navigation behaviors with no a priori knowledge of the environment.

6.5. Automatic feature selection

The previous approach used the visual features which are manually identified and rationalized from a single snapshot. These features typically arise out of human intuition examining the scene and uses the teachers expertise to identify typical patterns in navigational behaviors. Crudely speaking, the realm of learning for demonstration is where the human is only used to generate demonstrations and the features that relate to the exhibited behaviors are extracted automatically. This way the human is further detached from the reasoning about the feature relevance during the feature extraction process. We address this aspect in the following section where features are automatically extracted using PCA. However, PCA only finds the independent features most relevant from the input data presented but does not take the executed robot action into account. Hence, through PCR the principal components with high correlation to the output robot action are extracted. From the identified components, the perception-action mapping is then learned with a ANN.

6.5.1. Principal component extraction

The principal components of the demonstration images are determined as described in chapter 3.3. In order to handle to the computational complexity of PCA, the total number of training instances are sub-sampled to five representative images per demonstration. For corridor following, this subset of images covers both initial misaligned configuration and the final aligned configurations. The corridor following demonstration images and the principal components correspond to the application example presented in chapter 3.5. Fig. 3.11 shows the recorded images along with the five dominant eigenvectors. Obstacle avoidance, on the other hand does not depend on the entire local environment but mostly on the proximity of the obstacles ahead. Thus, only a rectangular region in the front half of the image is analyzed and projected onto its principal components. The demonstration data spans the complete scenario starting from the variations of different obstacle positions in front of the robot to homogeneous floor regions at the end of the demonstration. Fig. 6.27 shows the segmented images used for computing the principal components along with the five dominant eigenvectors for obstacle avoidance. The obstacle avoidance demonstrations are projected onto 75 principal components. Thus, a total of 2100×70 training examples for corridor following and 1885×75 training examples of obstacle avoidance are generated. The percentage of variance explained by the individual components together with their singular value for the two behaviors are visualized with a scree plot shown in Fig. 6.28. From the figure, one can see that the graph tends to flatten out at around four components for corridor following and five components for obstacle avoidance. The perception-action mapping between the selected components and the recorded robot action is trained with an ANN. Network training and validation is performed on a multi-layered feedforward network with Levenberg-Marquardt backpropagation. Using 60% of the data used for training, 20% used for validation and the remaining 20% for testing, the generalization errors are recorded for both the behaviors (Table 6.7). The errors clearly show that corridor following is easier to generalize than obstacle avoidance and the selection of components are performed unsupervised independent of the desired output.

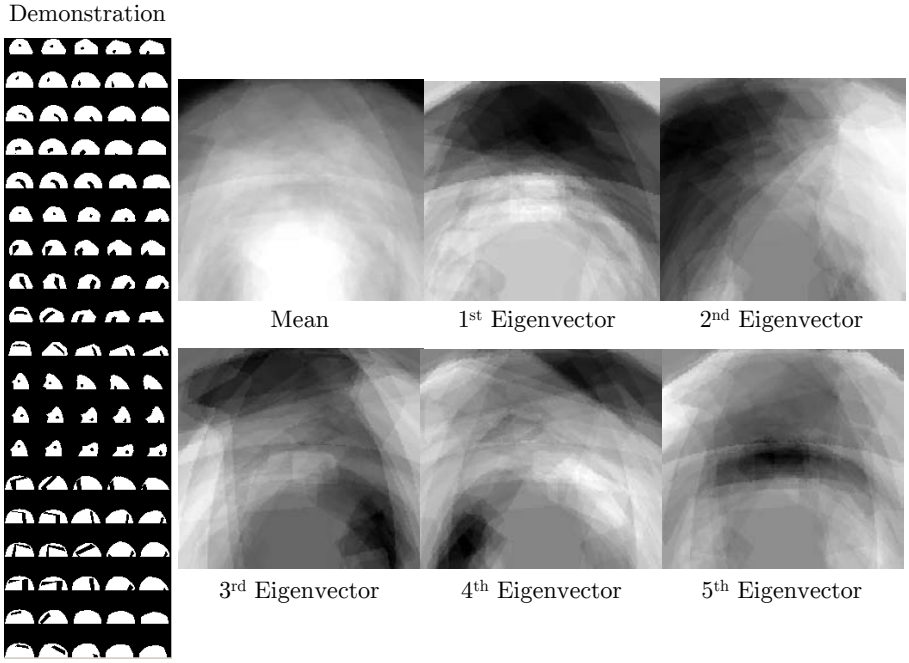


Figure 6.27.: Five images per demonstration spanning the entire length of obstacle avoidance demonstrations. A total of seventy five training images are used for computing the principal components. The mean vector along with the five dominant PCA eigenvectors are shown.

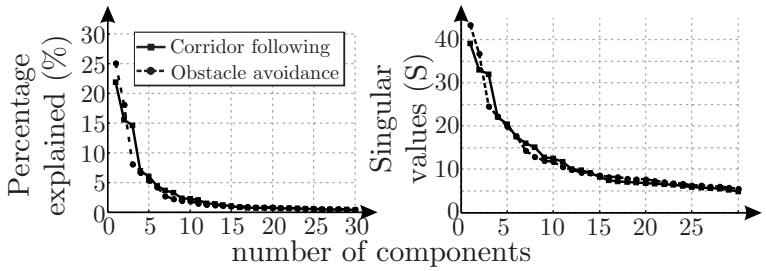


Figure 6.28.: Percentage variance explained by each component and the scree plot of the singular values for the top components.

This is overcome with PCR which is an extension of PCA, where the correlation of the projected components with the output variable is also considered. Fig. 6.29 shows the top individual feature correlations against the target. The coefficients provide an immediate inference in that the components of corridor following which are influenced by the shape of the environment have a predominantly a high correlation to the robot motion, whereas for obstacle avoidance, the top correlation coefficient is only 0.35. Nevertheless, the individual component correlation does not capture the mutual interaction between the different latent variables to help emerge a policy. Hence, a wrapper approach with forward chaining is then performed on the principal components to identify the best subset of correlating components. During forward chaining for every feature subset, PCR finds the subset by regressing the principal components against the target. The generalization errors: MSE and NMSE are recorded for a *5-fold cross-validation* within the feature selection process (Fig. 6.29). Thus, for every fold of cross-validation the training is done on 11 trajectories and tested on the remaining 3 trajectories. The evolution of the feature selection errors in the figure show that for corridor following, addition of more principal

Table 6.7.: Training, validation and testing errors for ANN trained with principal components. Cf - Corridor following, Oa - Obstacle avoidance.

Behavior	Training		Validation		Testing	
	MSE $[\text{deg/mm}]^2$	NMSE	MSE $[\text{deg/mm}]^2$	NMSE	MSE $[\text{deg/mm}]^2$	NMSE
Cf	0.11×10^{-4}	0.019	0.08×10^{-4}	0.02	0.18×10^{-4}	0.3
Oa	0.007	0.44	0.009	0.57	0.008	0.49

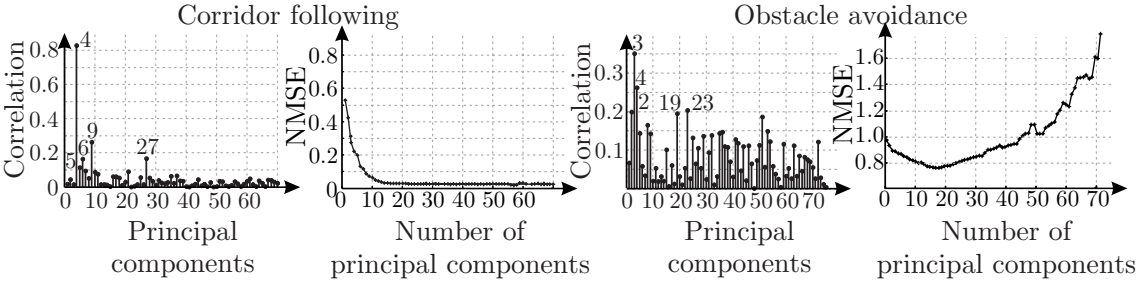


Figure 6.29.: Correlation coefficients and the forward chaining error evolution with PCR with *5-fold* cross-validation shown for both the behaviors.

components reduces the errors and tends to saturate around 10 features. For obstacle avoidance, the cross-validation error is the lowest for 18 components (See Table 6.8 for the features). Addition of more features tends to worsen the generalization. This is mainly due to the fact that obstacle avoidance is mostly correlated with the context of situation rather than the geometry of the scenario. Addition of more principal components brings in more details about the scenario geometry which, considering the variety of different scenarios generates ambiguity and correspondingly poorer generalization. Table 6.9 also tabulates the cross-validation errors of both the behaviors.

The PCR generalization of obstacle avoidance shows that there exists a non-linear relationship which is difficult to model. Hence, the selected components from PCR are trained with a feedforward ANN. The generalization with ANN shows a huge improvement compared to a direct regression (See Table 6.10). Corridor following on the other hand exhibited good generalization even with PCR, hence we avoid ANN. Fig. 6.30 shows the performance of ANN trained on PCR based features selected for all the obstacle avoidance demonstration trajectories. The performance of PCR on all corridor following demonstration trajectories can be referred in Chapter 3 (Fig. 3.16).

Fig. 6.31 shows the corridor following reproduction trajectories registered in different corridor environments. The top principal component based ANN model and the top correlating PCR model are compared in the same environment and starting configurations. Both the models traverse the scenarios quite robustly with the PCR based model handling the narrower corridors better than PCA based model. With broader corridors as shown in

Table 6.8.: Components selected using PCR based forward chaining.

Behavior	Selected components
Corridor following	4,8,6,2,9,11,5,7,30,10
Obstacle avoidance	4,3,38,5,41,35,36,42,37,29,1,44,13,39,28,17,18,57

Table 6.9.: *5-fold Cross-validation error* for the selected set of principal components for learning the behaviors

Behavior	Testing error	
	MSE $[\text{deg/mm}]^2$	NMSE
Corridor following	1.901×10^{-5}	0.0614
Obstacle avoidance	0.0107	0.76

Table 6.10.: Training, validation and testing errors for obstacle avoidance ANN

Training		Validation		Testing	
MSE$\times 10^{-4}$ $[\text{deg/mm}]^2$	NMSE	MSE$\times 10^{-4}$ $[\text{deg/mm}]^2$	NMSE	MSE$\times 10^{-4}$ $[\text{deg/mm}]^2$	NMSE
1.47	0.009	3.76	0.02	4.61	0.03

Fig. 6.31(c), the PCA based model performs marginally better than the other counterpart. For obstacle avoidance, ANN modeled on PCA and PCR are compared in unseen and novel environments as shown in Fig. 6.32.

6.5.2. Experimental results

Realistic experiments with the Pioneer 3DX mobile robot are carried out with both the models. Fig. 6.33 shows the two corridor following experimental scenarios. The figure also shows the segmented omnidirectional image and the principal component reconstruction using the top four components for PCA and the selected ten components for PCR. Both the models perform good. The principal components of corridor examples capture the misalignment and generally correlate well with the output. The challenge with obstacle avoidance is more where the principal components now become independent of the geometry of the environment but on the position of the obstacles. Fig. 6.34 shows two runs of obstacle avoidance carried out on a corridor. Similar to the previous setups, the robot starts in front of a pillar in a corridor followed by a door by the far end blocking the way. The result show that both evade the obstacles with different trajectories. With the first pillar as an obstacle both the models evade them well, with the PCR model making a larger turn. This happens at every obstacle where sharper and prolonged evasions are done. The PCA based model is relatively smoother and sometimes gets close to the obstacles but does not crash. The top 5 components from obstacle avoidance even if they do not have a high correlation with the output, they still manage to capture the essence of the behavior quite well.

6.5.3. Behavior classification

In order to accomplish navigation we require an arbitration mechanism in which switching among behaviors is also learned from data. In our case, the local environment and behavior are classified according to its appearance. The classifier rests on the training data acquired for the two behaviors and labeled as C for corridor following and O for obstacle avoidance. The relevant features for classification are identified using the mRMR

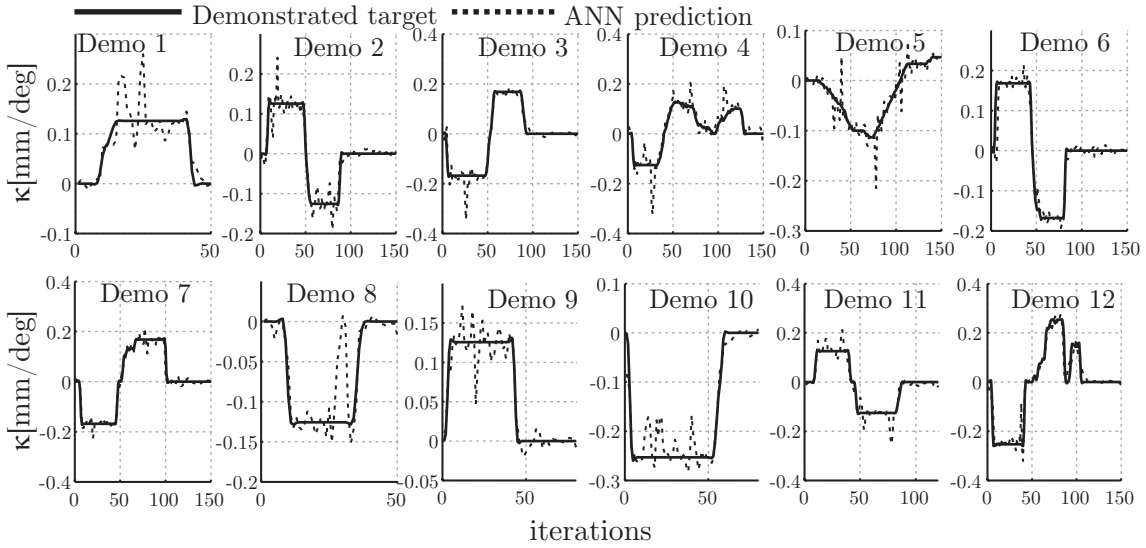


Figure 6.30.: ANN prediction with the features selected using PCR for obstacle avoidance. Curvature prediction shown for some of the demonstration trajectories.

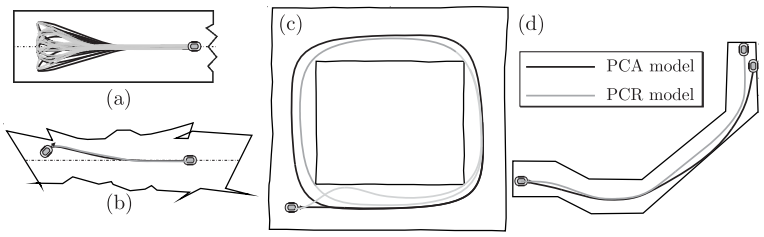


Figure 6.31.: Corridor following reproduction with principal components as features. (a) Trained corridor, (b) noisy corridor scenario, (c) unknown square corridor scenario and (d) corridor with changing widths.

[PLD05] by convolving the principal components with different classifiers such as NB, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and RG. Forward chaining is performed on these features and the feature set is expanded incrementally until the classification error no longer improves. Table 6.11 tabulates the *5-fold cross validation* performance of the different algorithms along with their optimal number of features to achieve the best classification accuracy. The table reveals that all the classification algorithms achieve a high rate of accuracy ranging from 90% for QDA to 99% accuracy with RG. By considering both the classification performance and the number of features it is clear that RG outperforms others by approximately 9% and utilizes considerable less number of features thereby reducing the complexity of the model. The selected features for RG based behavior classification are components 1,2,7 and 11. Fig. 6.35 shows the confusion matrix of RG validated on 40% of unseen data and also the performance on all the data together.

The complete navigational framework with PCA based ANN model for both the behaviors together with the behavior classification scheme is tested in several environments. Fig. 6.36 shows the robot path for one such experiment with PCA based ANN as it navigates the corridor. The RG based behavior classification adapts well to the variations and disturbances caused by the processing of real world images compared to the training data except for certain areas in which the incorrect obstacle avoidance behavior is selected instead of corridor following (see A1 in the figure). This is largely attributed to the

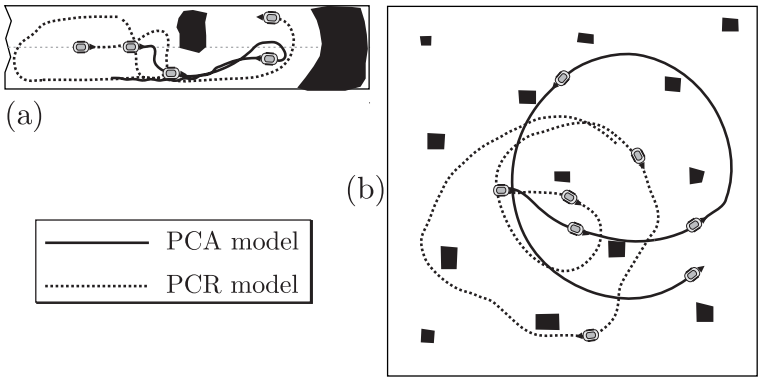


Figure 6.32.: Obstacle avoidance reproduction in unseen environments. Robot action is predicted by ANN model based on the principal components of the demonstration images. Shown are the reproduction trajectories of both PCA based ANN model and PCR based ANN model.

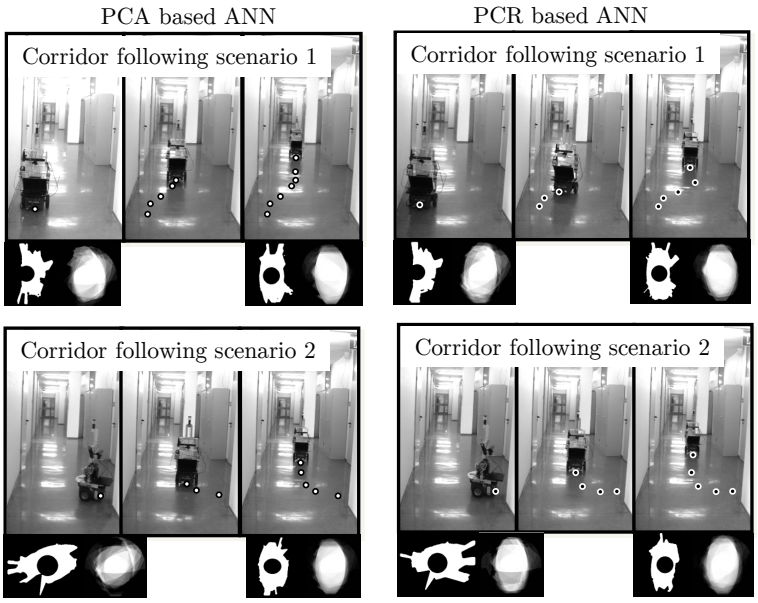


Figure 6.33.: Corridor following with PCA and PCR based ANN model. Shown are also the segmented free-floor image and its corresponding principal component based reconstruction.

false positives generated by the segmentation algorithm which requires an update of the internal floor model on a regular basis. Once the model is adapted to the new conditions the quality of the segmentation image improves thereby improving the performance of the behavior as well. At A2, the robot encounters a narrow passage due to an open door blocking the corridor on one side. The arbitration switches to obstacle avoidance due to the change in the corridor geometry and thereby avoids the obstacle and traverses the passage. At the dead end of the corridor, the robot turns 180° until the floor to the front of the robot is again free of obstacles.

6.6. Summary

Throughout this chapter, the individual behaviors are modeled using neural networks to answer two inherent questions that we asked at the start of the chapter namely, what is an efficient policy and how many demonstrations suffice for generating an efficient policy?

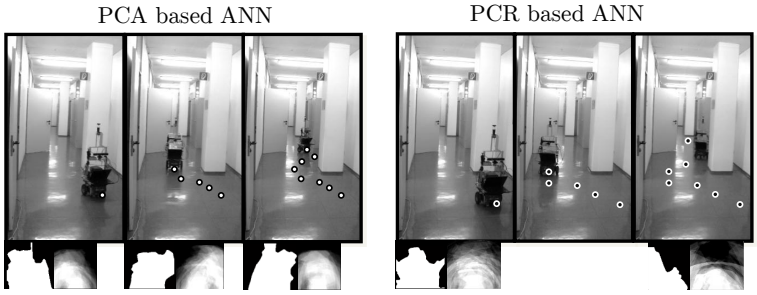


Figure 6.34.: Obstacle avoidance with PCA and PCR based ANN model. Shown are also the segmented free-floor image and its corresponding principal component based reconstruction.

Table 6.11.: Behavior classification: *5-fold cross validation*

Classifier	Number of features	Misclassification rate
NB	13	0.075
LDA	12	0.112
QDA	6	0.1015
RG	4	0.011

The networks were trained with diverse set of demonstrations and their performances are compared in both seen and unseen environments. The performance efficiency of every model is checked two fold. First, the training and the testing errors of the model against seen and unseen demonstration data are computed. Second, the model is deployed in simulation in ad novel environments where the robustness and sensitivity of behaviors are further tested. All the way through the training process, we aimed at learning the essence of the behavioral policy rather than attempt to replicate the demonstration trajectories accurately. From a conventional machine learning point of view, this typically indicates a higher generalization error during cross-validation and considered sub-optimal but from the behaviors perspective it can be termed acceptable as long as the corrective action executed proceeds to a successful completion of the task at hand. Static mapping with feed forward neural networks possess no memory and learns a direct mapping of perceptions to its corresponding actions. Experiments with corridor following and homing show that the static network is capable of learning the policy even with one or two examples. For a more complex behavior such as obstacle avoidance, the quality of the demonstration is of paramount importance than the quantity of demonstrations. Results showed that even with very few demonstrations, but performed on either side of the obstacle is sufficient for the network to learn policy mapping that eventually can traverse much complex unseen cluttered environments. Recurrent neural networks are examples of dynamic networks with feed back connections between the layers thereby possessing an internal memory allowing them to handle input sequences. Perceptual traces of inputs are fed to RNN. With corridor following, the model shows very similar performance to feed forward network managing to learn the behavior with just two demonstrations. In spite of the successful training of obstacle avoidance with RNN, the generalization error was inferior than feed forward neural network. This is counter intuitive in the sense that it is natural to expect a system with memory to perform better than a reactive snapshot model. More evident to notice is the performance of the model in unseen environments where the robot

Tested on 40% unseen data

TRUE CLASSES	PREDICTED CLASSES		
		Corridor following	Obstacle avoidance
	Corridor following	667	9
	Obstacle avoidance	8	744

Tested on entire data

TRUE CLASSES	PREDICTED CLASSES		
		Corridor following	Obstacle avoidance
	Corridor following	1679	11
	Obstacle avoidance	8	1872

Figure 6.35.: Confusion matrix of RG behavior classification with *60-40 hold-out cross-validation* and on the entire training data.

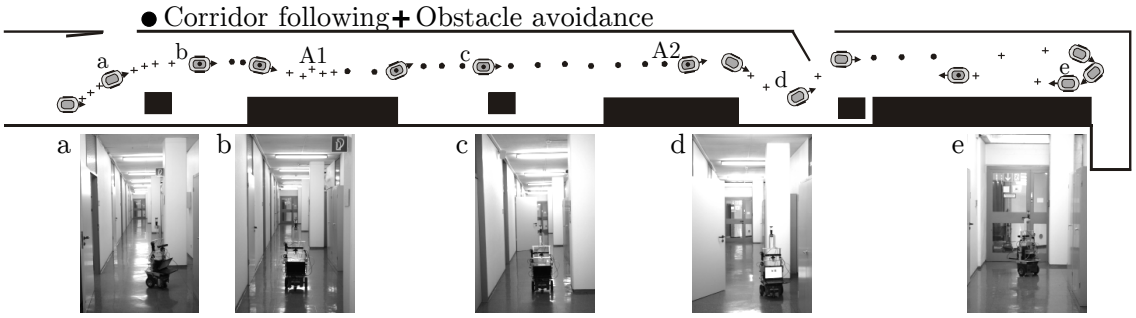


Figure 6.36.: Complete framework with behavior classification and PCA based ANN models.

falls into a local minimum. This problem directly correlates to the difficulty involved in training a RNN, where a network trained on one set of temporal dependencies of input features finds it very hard to generalize when presented with another set that are slightly different. Arriving to this conclusion raises another question: *is memory really required for robotic behaviors?* The characteristics of RNN to preserve context of operation is of high promise for mobile robots as already presented for context based operation in chapter 5, but for behaviors which can spread over different geometry and appearance of environments, this necessitates a thorough study. The author in [Sut13] discuss the inherent difficult of training RNN and present a wide range of solutions to counter this issue. Another dynamic network worth testing is Echo State Network [Jae01]; [Jae07], where the hidden layer neurons are only sparsely connected and provides a linear output. As an alternative approach to using visual features, automatic feature extraction through the principal components of behavior specific perceptions are also tested. Behavior representative features are identified and selected using PCA and PCR and provide the foundation to supervised learning of a behavior in terms of an ANN. With conclusions derived from model validation and testing, experimental validation is carried out on the Pioneer 3DX robot mounted with an omnidirectional camera. The results establish the robustness of the different models along with the fundamental differences in interpreting and executing a behavior.

7

Supervised learning of behavioral dynamics and behavior fusion

This chapter is divided into two subsections. In the first part, learning behavior representation emerging from the dynamics between perception and action is presented. The authors of [SDE95] presented a perspective of the interaction between behavior and the environment as a mutual pair of dynamical systems. For a mobile robot behavior, the perception-action cycle describes this interaction. The features that sense the environment are called as *behavior variables* and they evolve according to their own *behavioral dynamics*. In this chapter, the visual behavioral dynamics are learned from demonstrated data. Considering the behavioral variables $\mathbf{x}$ as state variables, the dynamics are then described by a set of time continuous difference equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \psi) + \eta \tag{7.0.1}$$

where, f is a non-linear differentiable function with parameters ψ and additive zero mean Gaussian noise η. As per the representation, the behavioral dynamics thereby possess an attractor at the equilibrium state $\mathbf{x}^*$ with $f(\mathbf{x}^*) = 0$. We define the mapping function f here in terms of GMM, whose parameters ψ consists of the Gaussian priors, means and covariances. The demonstration examples modeled with GMM is further enhanced by the asymptotic stability at the equilibrium state criteria of Stable Estimator of Dynamic Systems (SEDS) proposed by [KZB11].

The second part of the chapter delves into learning a monolithic behavior fusion model across all scenarios. Throughout the chapters, the emergent behavior is distinguished either at the environmental perception level (scenario) or at the behavior level. Its been consistently argumented that learning a monolithic model that learns the behavior fusion across all the scenarios is a difficult task. We address the key challenges involved, namely the matching of visual perceptions, the complexity reduction of visual information and the generalization of actions from demonstrations onto novel scenarios. A feedforward neural network and an instance based model [Mit97] is trained on the subset of the recorded perception action pairs.

7.1. Stable estimator of dynamical systems

When $x_{t=0,n=1}^{T_n,N}$ denote the action and perceptual features of dimension D recorded during N distinct demonstrations of the task of length T_n, the demonstrated trajectories approximated by a GMM with K Gaussians represented by the prior probabilities $(\boldsymbol{\pi}_k)$, mean $(\boldsymbol{\mu}_k)$ and covariance matrices $(\boldsymbol{\sigma}_k)$ is shown in equation (7.1.1).

$$P(\mathbf{x}) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \frac{1}{\sqrt{(2\boldsymbol{\pi})^D |\boldsymbol{\sigma}_k|}} e^{-\frac{1}{2}\left((\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\sigma}_k^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)} \tag{7.1.1}$$

The mean and the covariance matrix of the k^{th} Gaussian is represented as,

$$\boldsymbol{\mu}_k = \begin{pmatrix} \boldsymbol{\mu}_{\mathbf{x},k} \\ \boldsymbol{\mu}_{\dot{\mathbf{x}},k} \end{pmatrix} \quad , \quad \boldsymbol{\sigma}_k = \begin{pmatrix} \boldsymbol{\sigma}_{\mathbf{x},k} & \boldsymbol{\sigma}_{\mathbf{x}\dot{\mathbf{x}},k} \\ \boldsymbol{\sigma}_{\dot{\mathbf{x}}\mathbf{x},k} & \boldsymbol{\sigma}_{\dot{\mathbf{x}},k} \end{pmatrix}. \tag{7.1.2}$$

Taking the posterior mean estimate of $P(\dot{\mathbf{x}}|\mathbf{x})$, the estimate of the function f in (7.1.1) is expressed as a non-linear sum of linear dynamical systems [KZB11]:

$$\dot{\mathbf{x}} = \sum_{k=1}^{K} h_k(\mathbf{x}) (\mathbf{A}_k \mathbf{x} + \mathbf{B}_k) \tag{7.1.3}$$

where, $\mathbf{A}_k = \boldsymbol{\sigma}_{\dot{\mathbf{x}}\mathbf{x},k} (\boldsymbol{\sigma}_{\mathbf{x},k})^{-1}, \mathbf{B}_k = \boldsymbol{\mu}_{\dot{\mathbf{x}},k} - \mathbf{A}_k \boldsymbol{\mu}_{\mathbf{x},k}$ and $h_k(\mathbf{x}) = \frac{P(\mathbf{x};\boldsymbol{\mu}_{\mathbf{x},k},\boldsymbol{\sigma}_{\mathbf{x},k})}{\sum_{k=1}^{K} P(\mathbf{x};\boldsymbol{\mu}_{\mathbf{x},k},\boldsymbol{\sigma}_{\mathbf{x},k})}$. Under sufficient conditions, SEDS learns parameters of the GMM such that it guarantees asymptotic stability at the equilibrium $\mathbf{x}^* = 0$. Imitation trajectories are sampled from the learned GMM via Gaussian Mixture Regression (GMR) [Cal09]. GMR samples the controls $\dot{\mathbf{x}}$ from the current input state $\mathbf{x}$. Given a current state $\mathbf{x}_t$ at time t , GMR predicts $\dot{\mathbf{x}}_t$, from which the next action to be executed is computed using the sample time dt and the previous state. The number of appropriate Gaussians for the mixture model is determined by computing the Bayesian Information Criterion (BIC). The BIC is given by,

$$\text{BIC} = -2\mathcal{L} + (\log N) n_p \tag{7.1.4}$$

where $\mathcal{L}$ is the log likelihood of the data for the model parameters, N is the number of data points and n_p denotes the number of parameters of the GMM. The first term describes the accuracy of the model and the second term the complexity of the model in relationship to the amount of data available. Thus, the model with minimum BIC exhibits the highest posterior probability. The following section analyzes the results of learning the dynamics of the three visual behaviors using GMM. The behavioral variables are the same as presented before. The goal state or the potential equilibrium state for all the three behaviors is at $(0, 0, 0)$. Thus, the behavior triplet used for corridor following is (ω, α, β) and for obstacle avoidance is $(\omega, \sin(\theta_{safe}), \cos(\theta_{min}) 1/d_{min})$ and $(\omega, \alpha_g, \beta_g)$ for homing.

7.1.1. Behavior learning

The optimal number of Gaussians for the demonstrated data are obtained by computing BIC by varying the number of Gaussians between one and ten (see Appendix D.3.1).

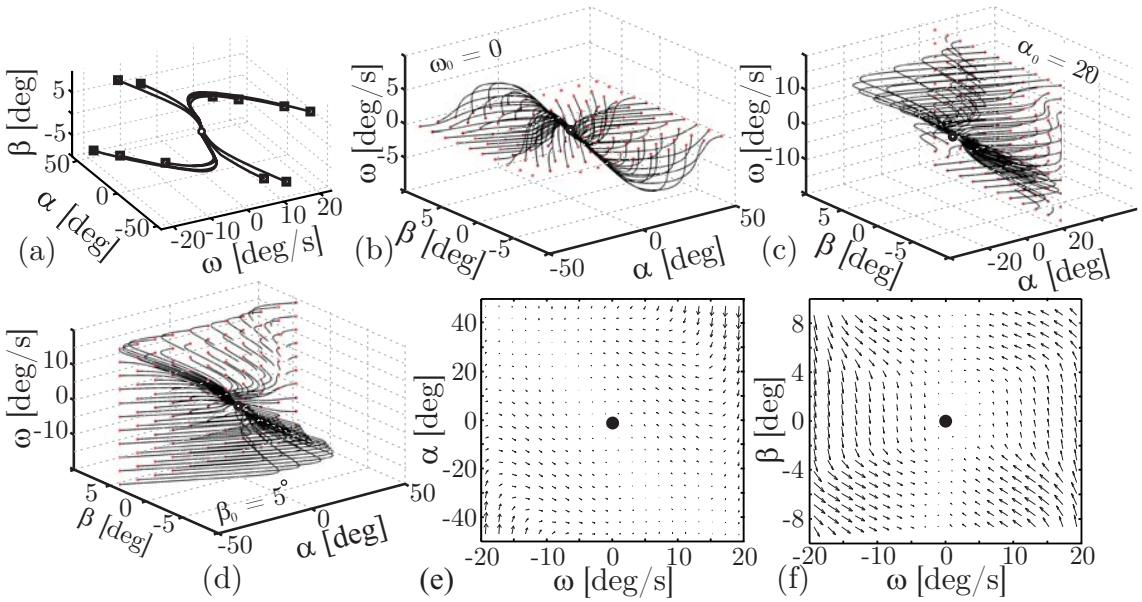


Figure 7.1.: Corridor following behavioral dynamics. (a) Reproduction of the demonstrated trajectories, (b) stream lines of novel trajectories with initial rotational velocity $\omega_0 = 0$, (c) stream lines of novel trajectories with initial lateral offset to the corridor $\alpha_0 = 20°$, (d) stream lines of the novel trajectory with initial orientation error to the center of the corridor $\beta_0 = 5°$, (e) and (f) projection of the vector $\dot{\mathbf{x}}$.

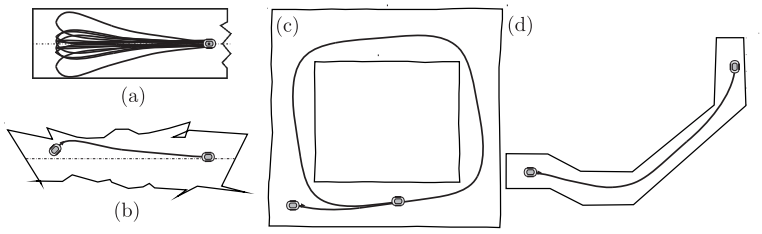


Figure 7.2.: Corridor following reproductions of the GMM. (a) Trained corridor, (b) noisy corridor scenario, (c) unknown square corridor scenario and (d) corridor with changing widths.

All the demonstrated data are used to arrive at the mixture model. Fig. 7.1(a) shows the reproduction of the GMM starting from the same initial configuration as demonstration used for corridor following. The figure also shows the behavioral space trajectories recorded for different starting conditions. Fig. 7.1 (b)-(d) correlates to different starting configurations spanning most of the state space and the corresponding generalization abilities. The trajectories show that even from state spaces not presented during training, the GMM model drives the robot robustly to the single attractor state at $(0, 0, 0)$. Fig. 7.1 (e)-(f) shows this basins of attraction where the vector field $\dot{x}(\alpha, \beta, \omega)$ is projected on a two dimensional subspace. The streamlines show that with higher misalignment and lateral offset angular error, the magnitude of $\dot{\omega}$ is larger. Fig. 7.2 shows the cross-validation of the model from both seen and unseen environments. The first experiment starts from the same initial configurations within the corridor as demonstrations and the second experiments puts the robot in a completely unseen scenario. The testing scenarios are the same as introduced in the previous sections. Starting from the same starting positions as in the demonstrations, the learned GMM model works good in driving the robot to the center of the corridor. The robustness of the model is further validated against unseen scenarios (Fig. 7.2(b)-(d)). Noisiness in the data is replicated by distorting the

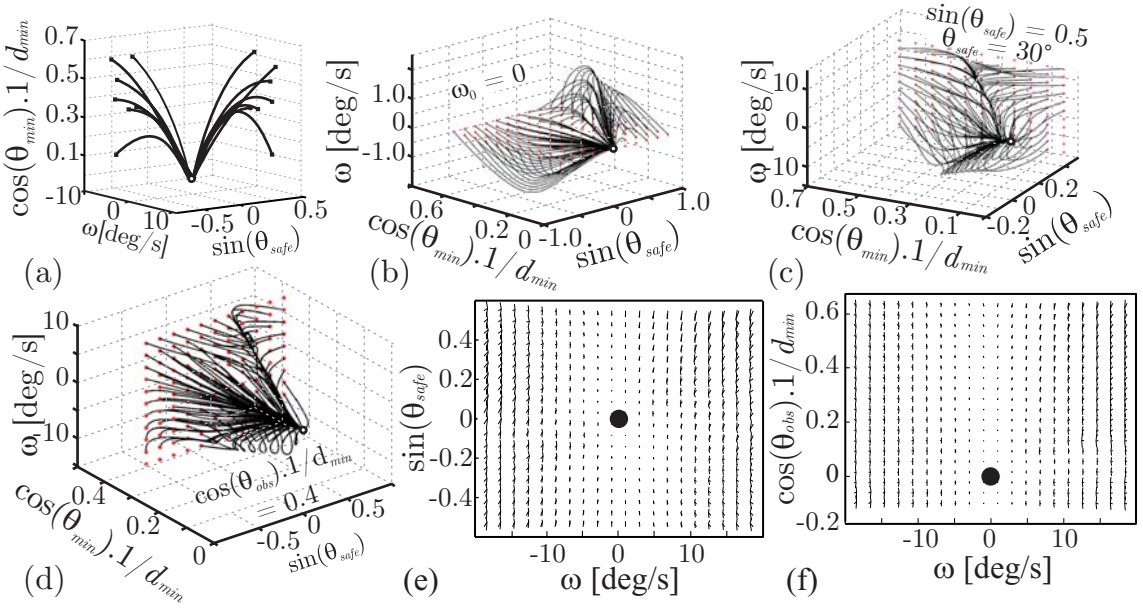


Figure 7.3.: Obstacle avoidance behavioral dynamics. (a) Reproduction of the demonstrated trajectories, (b) stream lines of novel trajectories with initial rotational velocity $\omega_0 = 0$, (c) stream lines of novel trajectories with initial safe direction at 30°, (d) stream lines of the novel trajectory with $\cos(\theta_{min}).1/d_{min}$, (e) and (f) projection of the vector $\dot{x}$.

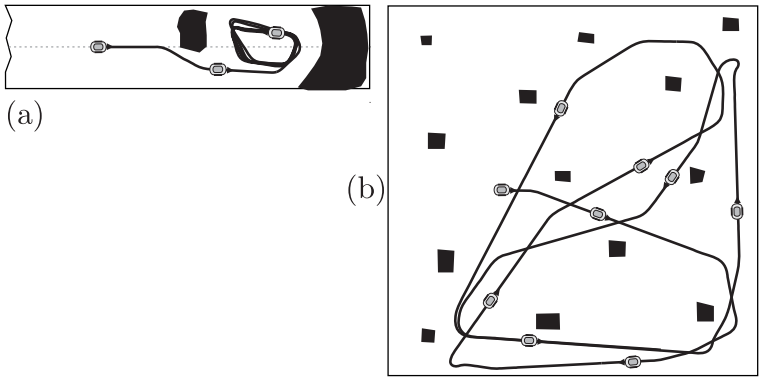


Figure 7.4.: Cross-validation in unseen environments with GMM

environment followed by two fundamentally different hallway environments. In all the cases, the GMM model encounters the environment in a robust fashion thereby further confirming the presence of asymptotically stable attractor along the corridor center.
The rotational velocity (ω), of obstacle avoidance is modeled by capturing the safe direction (θ_{safe}) and the closest obstacle (d_{min}, θ_{min}) to the robot. We take as previously $\sin(\theta_{safe})$ and $\cos(\theta_{min}).1/d_{min}$ as perceptual variables, thereby positioning the attractor at the origin which corresponds to the safe direction lying straight ahead and the closest obstacle is at an lateral angle of 90° to the robot. The dynamics of the learned obstacle avoidance GMM models is shown in Fig. 7.3. The figure shows that starting from different configurations, the behavioral dynamics converge to the equilibrium state. For completeness, the translational velocity of the behavior is also modeled from the demonstrated data. The linear relationship between the deceleration and the critical distance makes it quite straight forward to model. Please refer to the appendix D.3.2 to check for the details. Validation of the model to unseen scenarios is shown in Fig. 7.4. The biased

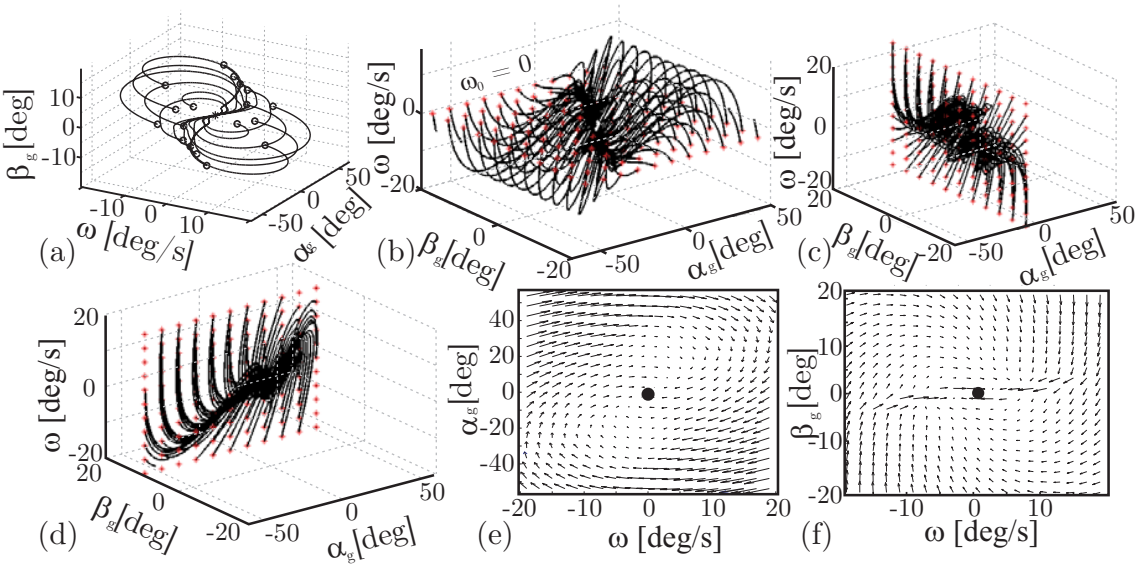


Figure 7.5.: Homing behavioral dynamics. (a) Reproduction of the demonstrated trajectories, (b) stream lines of novel trajectories with initial rotational velocity $\omega_0 = 0$ deg/s, (c) stream lines of novel trajectories with starting $\alpha_g = -10$ deg, (d) stream lines of the novel trajectory with starting $\beta_g = 10$ deg, (e) and (f) projection of the vector $\dot{\mathbf{x}}$.

nature and fundamental different in the representation of GMM to ANN can be seen in Fig. 7.4(a). The obstacle avoidance dynamics learned from simple evasion maneuvers encounters a narrow passage whose perceptual readings do not correspond to any of the examples shown. The robot thereby gets stuck in a local minima not able to get out of the passage. In a cluttered environment this is also partially visible nevertheless not as explicit as the previous experiment.

Homing behavior is modeled along the same lines as the two behavior mentioned before. The translational velocity is controlled with respect to the distance of the goal point to the robot and the rotational velocity is predicted with bearing angle and the final orientation angular error as features. Fig. 7.5 shows the learned behavior dynamics of the homing behavior. The single attractor of the dynamics is clearly seen in the figure, with the policy converging to its equilibrium point $[\alpha_g, \beta_g] = [0, 0]$ from all over the state space. Fig. 7.6 shows the imitation trajectories of the homing behavior in the simulation using both the translational and rotational velocity models. Starting from the same initial configurations as in demonstration, the learned dynamics successfully drives the robot to the goal point. The average absolute final bearing and orientation errors are 10° and 5°. The figure also shows the path traversed by the robot starting from different point along a circle around the goal point. As in the previous experiment, the initial orientation is always set to zero. In both the cases, the model drives the robot to the center. Nevertheless, one can notice a wobble in the path at the beginning which also clearly correlates to the streamlines shown in Fig. 7.5 (e) and (f). This is because of the fact that usually only one or very few Gaussians are used to model all the variations of motions recorded when starting from different initial configurations during demonstration, potentially revealing the inherent biased nature of GMM compared to ANN. Nevertheless, the learned dynamics after the initial curved path drives to the behavior equilibrium point also from unseen state spaces.

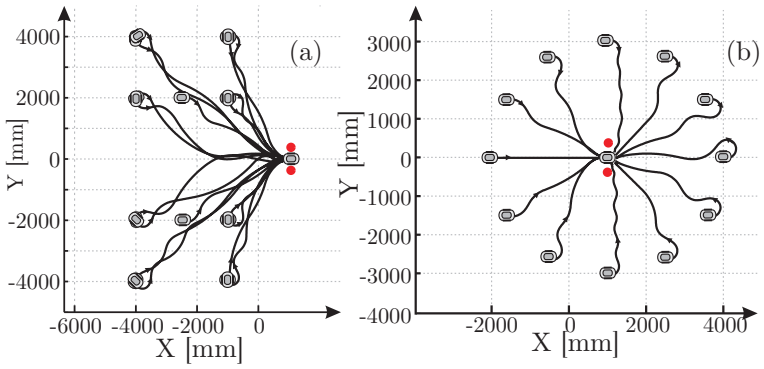


Figure 7.6.: Cross-validation in (a) from known configurations as shown in demonstrations and (b) starting along a circle around the goal point.

7.1.2. Experimental results

The reproduction of the modeled behavior are shown in Fig. 7.7. The translational velocity for corridor following is kept constant. In the case of obstacle avoidance the velocity is linearly reduced according to the learned velocity model and similarly for homing, the translational velocity is linearly reduced with respect to the model based on the distance to the obstacle. Corridor following is first tested by starting the robot along the corridor at different initial configurations of $(\alpha, \beta) = (51°, 5°), (-56°, 2°)$ and $(51°, 12°)$. In all the three cases, the learned behavior successfully returns and aligns the robot to the corridor center. Obstacle avoidance is tested on three prototypical scenarios. The first scene is a typical avoid an obstacle on the left scenario; the second scene contains two obstacles next to each other followed by a small gap to traverse. The third scenario mimics a dead end scenario by blocking the corridor with a large obstacle. The robot turns around until the free space is again found which extends beyond the critical distance. In all the cases, the interaction of the behavioral variables with the environment results in a successful passage of the gap. Fig. 7.8 shows the recorded perception variables for the experimental run together with the executed rotational velocity. The homing behavior is activated whenever the markers are detected in the image. Thus the behavior only triggers in the perceptual vicinity of the homing markers. Three such approaches of the homing behaviors are shown in the figure. One of the advantage of modularizing behaviors is the scalability of them. This is illustrated in the homing scenario II, where the acquired behavior is transferred and reused for a door traversing behavior. In this case, the homing markers are placed at the door posts to indicate the presence of a door. Given any door detection methodology and with the knowledge of the door posts in the omnidirectional image, the learned behavior can be directly transferred and extended. One such proposed door detection method is described in Appendix E for future works.

Behavior coordination

The overall behavior dynamics is obtained by coordinating the behaviors via arbitration or alternatively from the weighted fusion of the individual behaviors. For coordination, we use here the conventional coordination approach. The main objective of the experimental verification is so illustrate the ability of the learned behaviors to traverse and cooperate complex and dynamic novel situations. The behavioral gains are ascertained through two

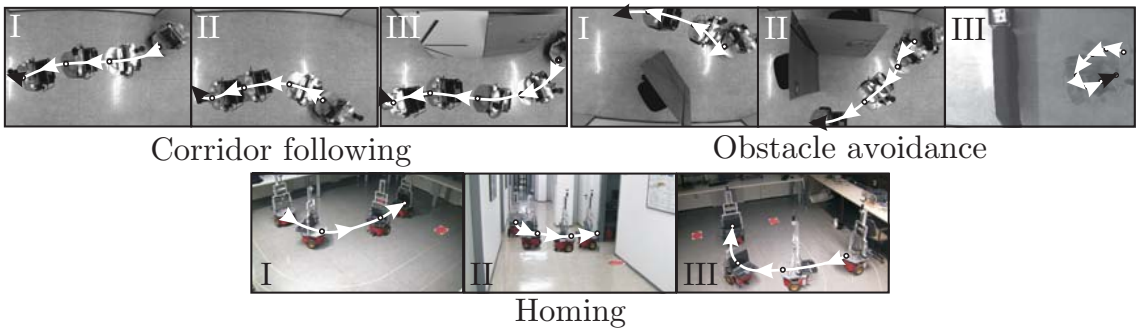


Figure 7.7.: Scenes from the experimental validation of corridor following, obstacle avoidance and homing with learned GMM behavior dynamics.

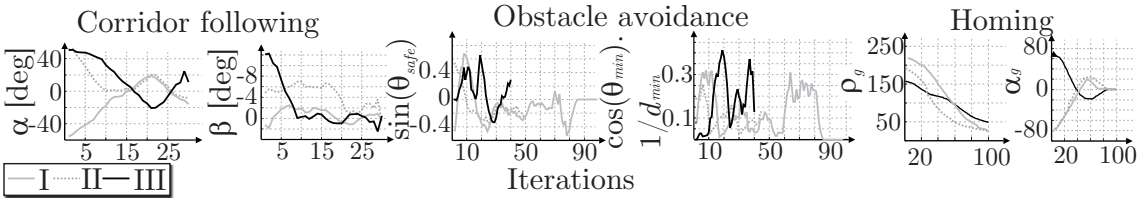


Figure 7.8.: Evolution of the perception variables recorded during the experiments shown in Figure 7.7.

behavioral variables namely, the distance to the closest obstacle (d_{min}) and the distance to the goal (ρ_g). Two thresholds for safe and critical distances to obstacle are set to change the gains of the behavior. The coordination of homing behavior relies on three thresholds namely *far*, *close* and *near*. The thresholds *close* and *near* are set so that the homing behavior is not subsumed by the other behaviors when the robot is close to the goal point but still in the vicinity of an obstacle. Such typical situations are encountered when the goal is close to a wall or an object to trigger an obstacle avoidance behavior. Fig. 7.9 shows the designed gains. The weights of the individual behaviors are computed by taking the product of the two gains and then normalizing it with the gains of the other behaviors. Both competitive (subsumption architecture) and a cooperative coordination (weighted summation) schemes are tested. With subsumption architecture, the most appropriate behavior is determined from the behavior with the highest gain:

$$\omega_o = \omega_b \text{ with weight } w_b = \max(w_c, w_{ob}, w_h)$$

where, ω_o is the commanded rotational velocity, b is the behavior with the maximum gain among the three behaviors (c,ob and h) and ω_b is the corresponding behavioral rotational velocity. For the cooperative behavior fusion, the overall behavioral dynamics is obtained from the weighted aggregation of the individual behavioral responses:

$$\dot{\omega} = \sum_b w_b f_b(\mathbf{xb})$$

where **xb** is the behavioral triplet.
The three behaviors are coordinated and tested in an office environment where the robot is supposed to navigate a corridor cluttered with obstacles, traverse a door indicated by homing landmarks and reach the final target pose. In theory, the robot is allowed to wander in the environment with the homing behavior getting activated on the first identification of homing markers in the omnidirectional image. Fig. 7.10 shows the path traced

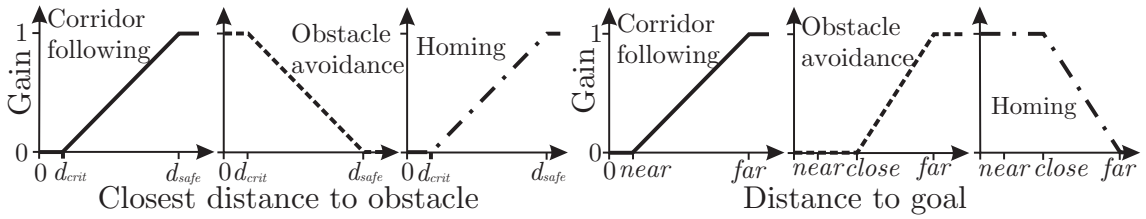


Figure 7.9.: Behavior gains based on perceptual variables

with both arbitration and command fusion behavior coordination. The interesting regions of the experiment are indicated by A1, A2 and A3. The registered behavioral gains for behavior arbitration is shown in the middle column of the figure. The omnidirectional image of the scene and the corresponding free floor segmentation of the free space at locations denoted by node 1 to 5 are also shown. Situation 1 shows the interplay between corridor following and obstacle avoidance behavior. The design of the gains are more apparent here where the robot on encountering the obstacle, corridor following is subsumed by obstacle avoidance. This allows a successful circumnavigation of the obstacle. Situation A2 exhibits a similar switching between the two behaviors where the corridor is narrowed down by two obstacles. The corridor is first partially blocked by an open door (node 2) and then two obstacles obstruct the passage further at node 3. A3 shows the execution of homing as a door traversal behavior. Even though the approach to the target pose in between the door posts occurs from a at angle the homing behavior successfully guides the robot through the open door (nodes 4 and 5) by a coordinated interplay of homing and obstacle avoidance. Once entering the room, the omnidirectional camera again picks up the target markers thereby executing the homing behavioral dynamics to successfully maneuver the robot to the target pose. The entire experiment is also performed using command fusion behavior coordination starting from the same initial configuration as in arbitration. Both the coordination schemes circumnavigate the obstacles and reach the goal point.

7.2. Learning behavior fusion

In the second part of this chapter we focus on learning behavior fusion of corridor following and obstacle avoidance algorithm. Demonstrations for behavior fusion are generated from a sonar based behavior coordination thus enabling to fuse emergent behaviors learning from a completely different sensor modality. The realization of the proposed integrated learning framework needs to counter two important challenges namely, the extraction of the relevant subset of features that capture the fusion of both the behaviors appropriately and how the model derived from the features can generalize to novel environments. Training examples are generated in five different indoor scenarios, two scenarios with different corridors and three scenarios in an open room and foyer environments with isolated obstacles to emphasize obstacle avoidance. Thus a total of 8876 training examples are generated. In each scenario, the examples are gathered while the robot powered by the sonar based wandering behavior is executed for a time span of 10-15 minutes. The diversity of the data is increased by starting demonstrations at regular time intervals from different random heading directions. In other words, the wandering behavior is temporar-

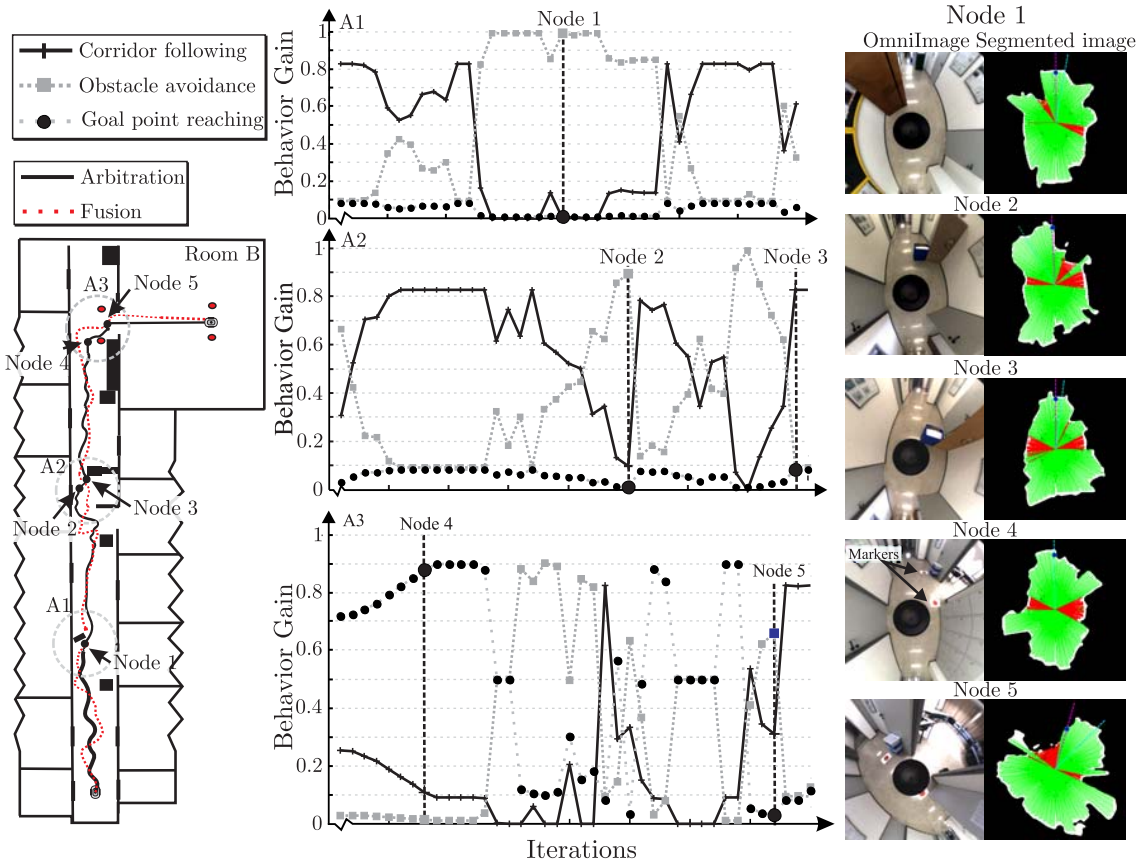


Figure 7.10.: Behavior coordination with GMM. Shown are the trajectories traced by both arbitration and command fusion schemes. The evolution of the gains of the three behaviors at three nodes A1, A2 and A3 are shown in the middle together with omnidirectional camera image and segmentation image at five node locations Node 1 through Node 5.

ily interrupted, a random heading direction is executed before the wandering behavior resumes control over the robot. Without these disturbances the training set would be dominated by perceptual states in which the robot is already aligned and centered with respect to the corridor. The experimental setup here is further augmented by using a monocular camera in conjunction with the PMD camera calibrated with each other. A floor segmentation on the monocular image is performed triggered by floor points identified by the PMD camera in addition to the omnidirectional image segmentation. Similar to the panoramic features as shown in Fig. 3.6, 12 additional column features from the monocular image are also computed. Instance based model and feed forward ANN are trained separately and their performance analyzed.

7.2.1. Instance based learner

From every demonstration instance, 53 visual features are extracted. The best subset of the 53 features are determined using an wrapper approach and forward chaining in conjunction with an instance based learning algorithm. Instance based learner is a memory based technique, where once queried with an input data, the model predicts the output by locally interpolating the neighbors of the query corresponding to a distance measure [BBB99]. This is often termed as a lazy learning technique, where the generalization beyond the training data is delayed until the query is asked. In other words, generalization

occurs only at query time. This carries an inherent disadvantage in that the training data should be saved and is always used during prediction. The local neighbors of the query input are determined either through Manhattan distance given by,

$$D\left(xt_i, xt_q\right) = \frac{\sum_{j=1}^{m} W_{(j)} \left| xt_{i(j)} - xt_{q(j)} \right|}{\sum_{j=1}^{m} W_{(j)}}, \tag{7.2.1}$$

where W_j, $xtt_{i(j)}$ and $xt_{q(j)}$ denote the j^{th} components of vectors $W\left[m, 1\right]$ weights, i^{th} input example xt_i and query point xt_q. The scaling factor indicates the relative weight of the features to the distance function. The scaling factors are set inversely proportional to the squared distance between the maximum and minimum feature value. Thus, the distribution of feature values in the training set is also considered. Forward chaining is performed by validating the training data in different scenarios against each other to identify the best feature subset. We refer to these validations as inter- and intra- scenario validation. Fig. 7.11 shows the recorded MSE of the selected features for both scaled and unscaled model during forward chaining. From the graph it is apparent that the constant model for unscaled distance outperforms the linear instance based model. The reason for the inferior performance of the linear model is due to the relatively sparse density of training examples within a high dimensional feature space. The prediction yt_q for the query point with a constant model is given by,

$$\mathbf{yt}_q = \frac{\sum_{1}^{k} D\left(\mathbf{xt}_i, \mathbf{xt}_q\right) y_i}{\sum_{1}^{k} D\left(\mathbf{xt}_i, \mathbf{xt}_q\right)},$$

where k is the number of nearest neighbors for the query point $\mathbf{x}_q$, $\mathbf{x}_i$ is the i^{th} input example, $\mathbf{y}_i$ is the corresponding output and $D\left(\mathbf{xt}_i, \mathbf{xt}_q\right)$ is the Manhattan distance shown in Eq. 7.2.1. With a set of twenty features and only about few thousands of training instances a local linear model is subject to overfitting, in particular as the training instances within a demonstration tend to be correlated. Hence, further feature selection is only performed with respect to the constant instance based model. From the figure it is also apparent that the distance scaled model presents is more robust with respect to inclusion of additional features compared to the unscaled constant model for which the generalization error is less robust. The lower generalization error is attributed to the fact that features with a large data range contribute possess more influence than features with low variation. Hence, the results of the scaled model are used for forward chained feature selection. The inclusion of additional features terminates once the generalization error increases by more than 0.001 deg/cm or if the number of selected features exceeds twenty. Training on open room data and validating on corridor following data exhibits substantially larger generalization error and terminates within only 3-4 features. This failure to generalize from one scenario onto another is explained by the fundamentally different structure or open rooms and corridors. In the former the robot circumnavigates isolated obstacles, thus avoiding proximity to objects, whereas in the later the robot balances proximity by alignment and centering with respect to the corridor. The futile attempt to generalize isolated obstacle avoidance to corridor following results in an overly aggressive behavior that merely bounces the robot between left and right wall. Fig. 7.12 illustrates the selected features for the three aforementioned scenarios. Those features that are selected at least in two out of three scenarios are included in the final feature set for

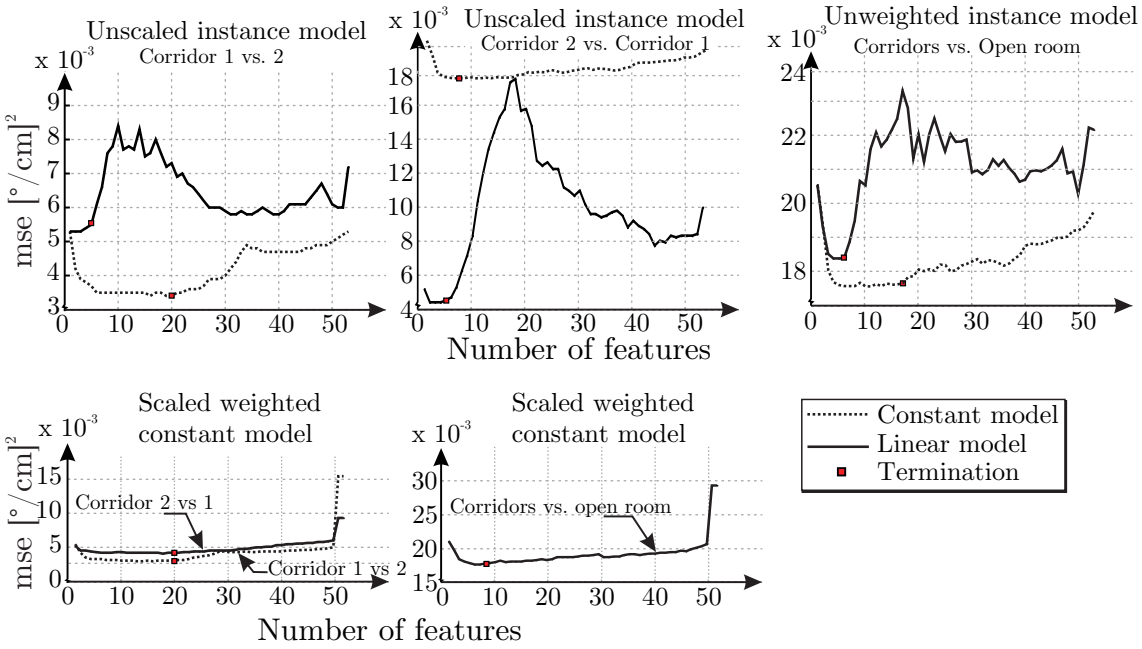


Figure 7.11.: Forward chaining error evolution. (Top left) Unscaled instance based model. Training: corridor 1, validation: corridor 2, (Top middle) Training: corridor 2, validation: corridor 1, (Top right) Training: corridor 1 and 2, validation: open room. (Bottom left) scaled constant model. Training and validation between corridor 1 and 2 and vice versa and (Bottom right) Training: corridor 1 and 2, validation: open room. Total number of training instances: 8876. Corridor 1: 3312 instances, corridor 2: 2302 instances and open room: 3326 instances.

learning. The black squares in the plot correspond to the thirteen finally selected features listed in Table 7.1. The aggregated image features (1,2,10,11,13) that provide information about the robots global perceptual space represent almost half of all the features. The remaining features capture local properties such as direct or relative proximity between symmetrically opposite headings. A subset of diverse, but carefully selected features are sufficient to generalize a visual control from the demonstrated behavior even though it relies on completely different sensor modality. The selected feature set represents a balanced mixture of monocular and panoramic image features emphasizing the importance of both coarse and detailed image features. The generalized behavior selected is validated for its robustness in the following section.

The generalization capability of the learned behavior is validated using *k-fold* validation and an inter-scenario testing. The number of nearest neighbors are varied between one and fifteen. The validation of the training set is performed on the unseen subset data. The inter-scenario validation tests the ability to generalize from training in one corridor onto performance in the second unseen and geometrically similar corridor with albeit different texture. The scenario analysis also investigates as to whether the learner is able to generalize from the two corridor scenarios onto the fundamentally different open room scenario. Since the final features for training are in fact generated using generalization error with forward chaining, the combination of scenario specific image features for the entire data set would further establish the final robustness of the training set.

Table 7.2 shows the validation results of *k-fold*, inter-scenario and inter-behavioral testing. The generalization Root Mean Squared Error (RMSE) between corridor 1 and 2 (C1 and C2 in Table 7.2) is robust, due to the similarity of training and test instances. This char-

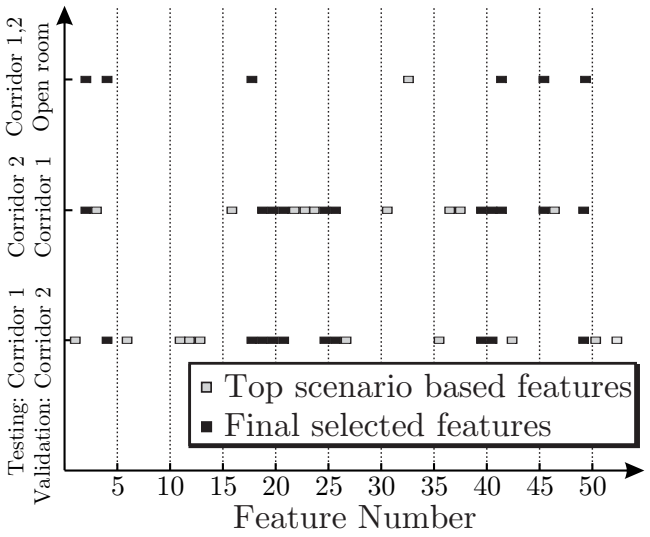


Figure 7.12.: Table of selected features in individual inter- and intra- scenario validation and the final selection of features.

acteristic robustness of the training data is also seen from the *4-fold* validation between the two corridor data where the k^{th} block of C1 is tested against the entire C2 dataset and vice versa. The validation error between the corridor following and obstacle avoidance (OA in Table 7.2) and vice versa is substantially larger mainly attributing to the lack of similar examples. In LfD, the training examples exhibit a substantial amount of ambiguity of the actions as well as the perceptions. This is particularly true for mobile robots with complex visual perceptions in unstructured environments. This perceptual aliasing is first compensated by clustering the nearest neighbors of the output curvature. Regression is then limited to those subsets of neighbors that form the largest cluster in order to achieve robustness. It also helps to resolve problems arising from ambiguous training instances and noise. This clustering in output space improves the robustness of the acquired behaviors even though it increases the mean regression error from a machine learning point of view. However, from the perspective of robotic behaviors, the compromise action is often inadequate. With the clustering, matching of queries with training instances occurs in two stages. Clustering identifies the general similar scenario, e.g. passing left or right, and regression the most similar instances within the same situation.

Experimental results

A variety of experiments are performed to validate the performance and robustness of acquired behaviors. The experimental platform is a Pioneer 3DX mobile robot equipped with a 3D-2D camera rig and an omnidirectional camera. Fig. 7.13(a) and (b) show a prototypical experimental scenario. The robots ability to avoid obstacles within a corridor is tested in corridors C1 and C2 with different floor texture and illumination. In both cases the robot is positioned in front of a pillar alongside the left wall of the corridor. The depicted trajectories show that the robot successfully evades the obstacle and centers itself in the corridor. The experiment is repeated in the second corridor. The general emerging behavior is similar and confirms the ability to generalize behaviors across different visual appearances of geometrically similar environment. Fig. 7.14 shows another test scenario in which the robot starts in a confined space with a frontal obstacle

Table 7.1.: Relevant features selected for training with scaled constant instance based model

S.No	Category	Feature
1	Monocular	Percentage of wall
2	Monocular	Height of column 1
3	Monocular	Ratio of floor to the left and right
4,5,6	Monocular	Difference of floor area column pair 1-12,2-11,3-10
7,8	Panoramic	Height of column 1,2
9	Panoramic	Amount of floor in column 8
10	Panoramic	Slope of the line
11	Panoramic	Intercept of the line
12	Panoramic	Difference of floor area column pair 2-7
13	Robot centric	Previous turn flag

Table 7.2.: Inter-scenario and inter-behavioral validation

Training scenario	Testing scenario	RMSE [deg/cm]	Maximum squared error $[\text{deg/cm}]^2$
k^{th} block set	rest $k-1$ sets	0.0858	0.6028
k-fold C1	C2	0.0695	0.2350
k-fold C2	C1	0.0845	0.2422
C1	C2	0.0736	0.2785
C2	C1	0.0728	0.3183
C1&C2	OA	0.1407	1.07
OA	C1&C2	0.0958	0.5264

adjacent to the wall. This configuration represents a dead end or a narrow passage situation. The traversed trajectory shows that the learned behavior is robust enough to cope with corrupted segmentation that results from the occlusion of floor seed points by the nearby wall and obstacle. The robustness of the acquired behavior is validated by having the robot wandering in the environment over a prolonged period of about one hour. During that period the robot successfully avoided static obstacles and people that intruded the corridor. The narrow passages and corridor dead end situations are resolved by maintaining the initial turning direction until again the robot is heading towards the free corridor space.

7.2.2. Artificial neural networks

For the following examples we use only the moment and the proximity features for learning the behavior fusion. Table 7.3 shows the top correlating features between the input and the curvature output of the robot. This provides an insight into the relative importance

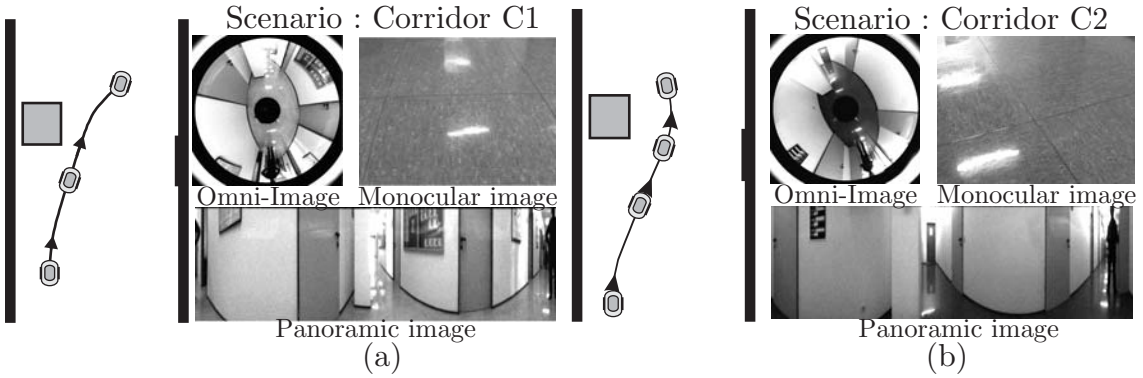


Figure 7.13.: Behavior fusion experimental scenario 1.

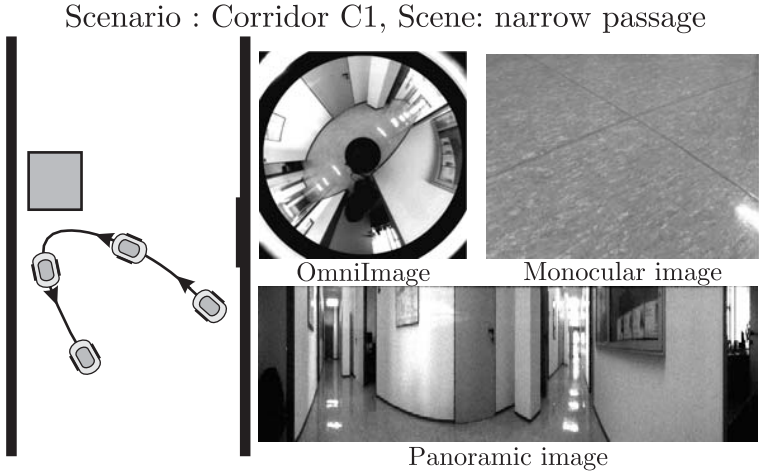


Figure 7.14.: Behavior fusion: Corner scenario

of the individual features but unfortunately does not provide any more information about the optimal feature set. This is identified using the wrapper approach. For forward chaining, we use a combination of Locally Weighted Regression (LWR) and ANN to determine the feature set. LWR [AMS97] is a nonlinear function approximation model in higher dimensions. At the core of the approximation lies piecewise local linear models in a computed region of validity defined by a weighting kernel. Given a query point x_q, we employ a Gaussian kernel centered at x_q and perform regression. The identified features are then modeled with ANN for predicting the curvature. With an eager learning method such as ANN, the target function is approximated globally during training, thus requiring less memory and runtime for prediction than lazy learning methods. Table 7.3 also shown the selected features from LWR. It is interesting to note here that the centroid feature which possess high correlation to the output has fallen out during forward chaining whereas the rest of the features are selected. In order to confirm this confounding factor, we try combinations of these four variables with ANN and check for their generalization ability. The training, validation and testing errors (Normalized Root Mean Squared Error (NRMSE)) for ANN are shown in Table 7.4.
From the table one can see that the combination $15, 17$, $15, 16, 17$ and $3, 17$ generalize better than the remaining combinations by showing an improvement in the residual variance of the prediction by almost $25 - 30\%$. The errors also show that feature 16 has little or no improvement in the error compared to the other two. Hence, we restrict the subsequent

Table 7.3.: Top correlating features and selected features

Feature#	Feature	Regression coefficient	Correlating feature	Selected feature
17	Previous turn direction	0.71	x	x
3	X-coordinate of the elliptical centroid	0.40	x	-
15	Angle between major axis and current robot heading(β)	0.35	x	x
16	Ratio of the elliptical axis	-	-	x

Table 7.4.: Feature combination and validation using ANN

Features	Training RMSE [deg/cm]	Testing RMSE [deg/cm]	NRMSE
3 15 16 17	0.027	0.038	0.84
3 15	0.041	0.043	0.95
3 15 16	0.042	0.041	0.91
3 15 17	0.029	0.034	0.76
15 16	0.040	0.040	0.97
15 17	0.030	0.033	0.74
15 16 17	0.031	0.034	0.76
3 17	0.034	0.027	0.60

analysis to $3, 17$ and $15, 17$. With the selected feature set, we test their combinations with the closest obstacle features $(1/d_{min}, \theta_{min})$ obtained through different frontal half partitioning of two, four or eight uniform segments (See Fig. 3.4). The generalization of ANN in inter- and intra- scenario cross-validation is shown in Table 7.5. From the table one can infer that the generalization between the two corridors is robust. This is mainly because of the similarity of the training and testing examples albeit different corridor geometries. With the visual sonar proximity information, the model performs better.

Experimental results

Two simple experiments are performed with the feature set $15, 17$ together with four frontal proximity information governed by curvature predicting ANN. Two scenarios for corridor following and obstacle avoidance are shown here. Experiments are performed for a prolonged duration in the corridor with random starting configurations. Sometimes in the event of bad segmentation or navigating through narrow environment the behavior fails. Fig. 7.15 shows the overlaid runs of two experiments. In the first experiment the robot is positioned in front of a pillar and misaligned thereby triggering a fusion of obstacle avoidance at the beginning followed corridor following. The second scenario places the robot at completely misaligned configuration along a corridor with a wall to the front to trigger an obstacle avoidance maneuver. As a comparison, the trajectories from sonar based wandering behavior is also shown. As one can see, the learned behavior

Table 7.5.: Feature sets 15, 17 and 3, 17. Locally weighted regression (LWR) and Artificial Neural Networks (ANN)

Feature set	Training	Testing	# sector	Training RMSE [deg/cm]	Testing RMSE [deg/cm]
15, 17	CO1	CO2	-	0.0318	0.0285
15, 17	CO1	CO2	2	0.0303	0.0295
15, 17	CO1	CO2	4	0.0266	0.0335
15, 17	CO1	CO2	8	0.0288	0.0264
15, 17	CO	OR	-	0.0315	0.0409
15, 17	CO	OR	2	0.0308	0.0440
15, 17	CO	OR	4	0.0272	0.0436
15, 17	CO	OR	8	0.0233	0.0440
15, 17	Full	Full	-	0.0416	0.0449
15, 17	Full	Full	2	0.0321	0.0398
15, 17	Full	Full	4	0.0313	0.0352
15, 17	Full	Full	8	0.0313	0.0302
3, 17	CO1	CO2	-	0.0320	0.0322
3, 17	CO1	CO2	2	0.0326	0.0266
3, 17	CO1	CO2	4	0.0254	0.0350
3, 17	CO1	CO2	8	0.0257	0.0224
3, 17	CO	OR	-	0.0349	0.0427
3, 17	CO	OR	2	0.0284	0.0435
3, 17	CO	OR	4	0.0294	0.0441
3, 17	CO	OR	8	0.0236	0.0442
3, 17	Full	Full	-	0.0326	0.0355
3, 17	Full	Full	2	0.0332	0.0344
3, 17	Full	Full	4	0.0350	0.0315
3, 17	Full	Full	8	0.0278	0.0345

initiates a turn earlier than the demonstrated behavior. In the first experiment the learned behavior turns right much earlier compared to the sonar behavior which turns only on detecting an obstacle. Similarly in the second experiment, a similar trend can be observed where the learned ANN robustly aligns itself to the corridor center much quickly than the demonstrated behavior.

7.2.3. Discussion

The experimental results show that both with instance based learning and ANN, the learned model counters prototypical scenarios quite robustly. Nevertheless, they run into problems especially when the environment is dynamic or noisy. Furthermore training a robust monolithic model requires a lot of expertise and training data to cover the diversity. By adding more behaviors to the system, the model complexity grows exponentially requires lots of training data to prevent under fit. A mobile robot behavior is highly situated, in that there is a strong coupling between the behaviors and the environment.

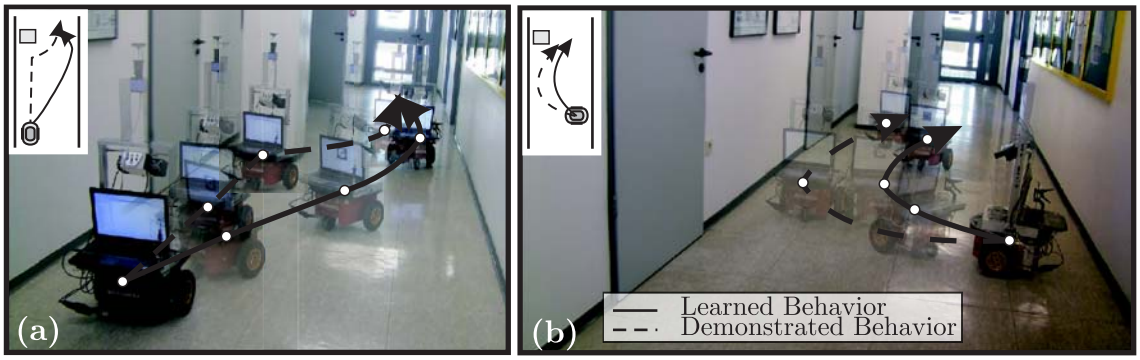


Figure 7.15.: Demonstrated vs. learned behavior. (a) Obstacle avoidance scenario and (b) misalignment in a corridor.

With a monolithic model, the individual behavioral interactions with the environment is either forgotten or lost in the curse of dimensionality. An effective alternative is to identify the most appropriate modularization of the navigation policy. Two of the modularization are discussed in Chapter 5, 6 and in the first half of this chapter. ANN models typically requires loads of data and proper parametrization of the neuron weights to learn multiple behaviors at the same time. Hinton and co in [HOT06], propose a deep neural network where the network pre-trains itself through unsupervised learning and then using backpropagation learning for fine tuning the weights. Nevertheless, pre-training such a network needs abundant information to learn the variations in the data. With increasing number of behaviors, this becomes extremely complex to train and manage. Another alternative is to design an adaptive behavior which learns on-line with exploration. Reinforcement learning within the framework of LfD promises a lot of potential in that the learning can start with reasonable a priori knowledge. This is discussed in detail in the next chapter where a proof of concept RL framework is presented to learn a corridor following behavior using minimal demonstrations.

8
Reinforcement learning of behaviors

Bellmann's Principle of Optimality: An optimal policy has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
– Richard Bellmann, *Dynamic Programming, 1957.*

The supervised learning of behavior approaches discussed until now, learns a policy solely from demonstration examples. This goes with the assumption that the teacher is an expert who is well versed with the task and its successful execution. In the previous chapters we also saw that in the event of erroneous demonstrations or few demonstrations, generalizing a policy to diverse scenarios becomes a challenging task. In most cases, the essence of the learned policy succeeds in executing the demonstrated tasks but the optimality of task execution is not always unquestioned. Reinforcement learning (RL) [SB98] provides an approach to learn without an external teacher, deal with not so deterministic scenarios and rewards the most optimal state-action pair. This chapter introduces a reinforcement learning approach to learn situated robot behavior within the framework of learning from demonstration. RL for robotic behavior presents a clear challenge in that the state and action pairs are continuous and the features number of such pairs are very large. For every state, searching through all possible actions to find the optimal long term policy is an expensive process computationally. This problem is partially alleviated using an initial behavior model derived from demonstration examples. Using this initial model, an optimal behavior model is derived from it using RL. As a proof of concept for this approach we present RL of corridor following behavior in this chapter. As an initial behavior model, we use the corridor following policy learned with GMM. Using the learned behavior dynamics, the state-value function is computed and the policy then iteratively improved using RL. The parameters of the GMM are adapted through RL trying to search for the policy with the highest long term reward. The potential of the framework for challenging problems such as one-shot learning and learning from scratch are analyzed, tested and finally validated on the robot. Fig. 8.1 shows the system architecture of the proposed RL scheme for learning corridor following.

8.1. Reinforcement learning using a model

Reinforcement learning is a machine learning paradigm that finds the optimal policy for a task by searching for the prescribed set of states S the optimal action from the available set

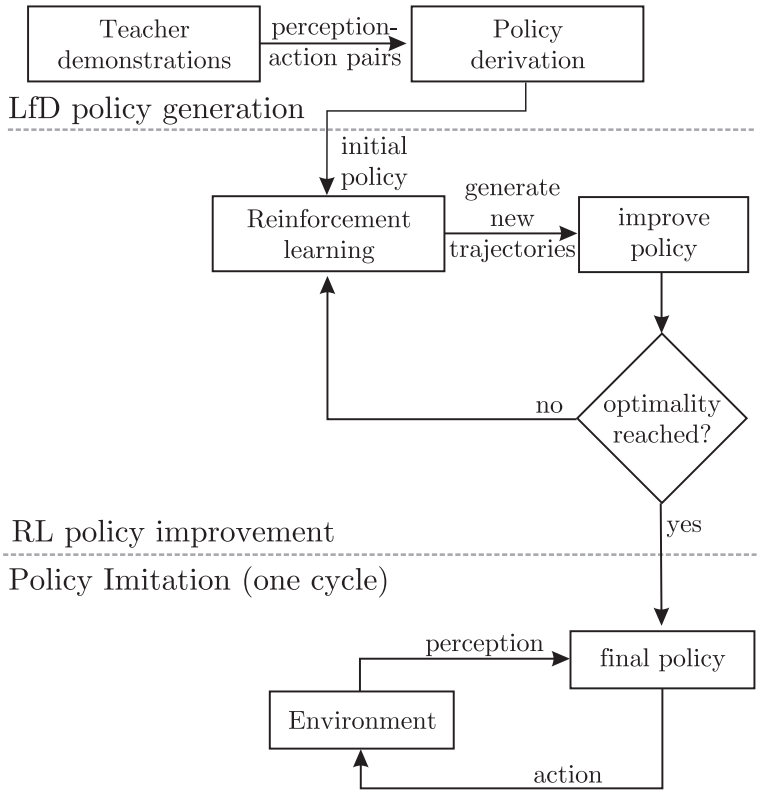


Figure 8.1.: System architecture of the proposed reinforcement learning of corridor following within the framework of learning from demonstration.

of actions A. For every state s_t at time t, the RL agent considers different actions a_t that can be executed. For every such state-action pair, the agent receives a reward $r_{t+1} \in \Re$ indicating how good the action is with regards to the task. For a visual corridor following behavior, two types of reward functions are used, namely a dense reward function and a sparse reward function. Dense reward function as the name suggests, assigns a non-zero reward for every state-action pair available. In the context of corridor following, the state variables considered are the offset angle (α) and the orientation error to the corridor center (β) with the rotational velocity (ω) acting as the action variable. The design of the reward function is thus designed such that the agent receives a maximum reward at the corridor center where $[\alpha, \beta, \omega]$ =[0 deg/s, 0 deg, 0 deg] and reduces correspondingly when moving away. For designing the reward function, the perceptual variables are first normalized $(\alpha/\alpha_{\max}, \beta/\beta_{\max}, \omega/\omega_{\max})$ where $\alpha_{\max} = 90$ deg, $\beta_{\max} = 10$ deg and $\omega_{\max} = 15$ deg/s, so that the total reward given never exceeds 1. The reward is then computed as,

$$R = \mathrm{K}_\omega \frac{1}{\omega + 1} + \mathrm{K}_\alpha \frac{1}{\alpha + 1} + \mathrm{K}_\beta \frac{1}{\beta + 1}, \tag{8.1.1}$$

where $\mathrm{K}_\omega = 0.1, \mathrm{K}_\alpha = 0.2$ and $\mathrm{K}_\beta = 0.7$ are the weights associated with ω, α and β. Sparse reward is highly goal oriented and easier to design in that only the goal state is rewarded and all other states are equally penalized. For corridor following, the goal area is around the corridor center $(\mathrm{abs}(\alpha) \leq 5 \text{ deg}, \mathrm{abs}(\beta) \leq 0.5 \text{ deg}, \mathrm{abs}(\omega) \leq 1 \text{ deg/s})$ which receives a reward of 0 and all other state-action combinations get a penalty of -1. Fig. 8.2 shows both the reward design for corridor following behavior. The goal of the agent is to find the policy that maximizes the long term discounted reward. The long term reward

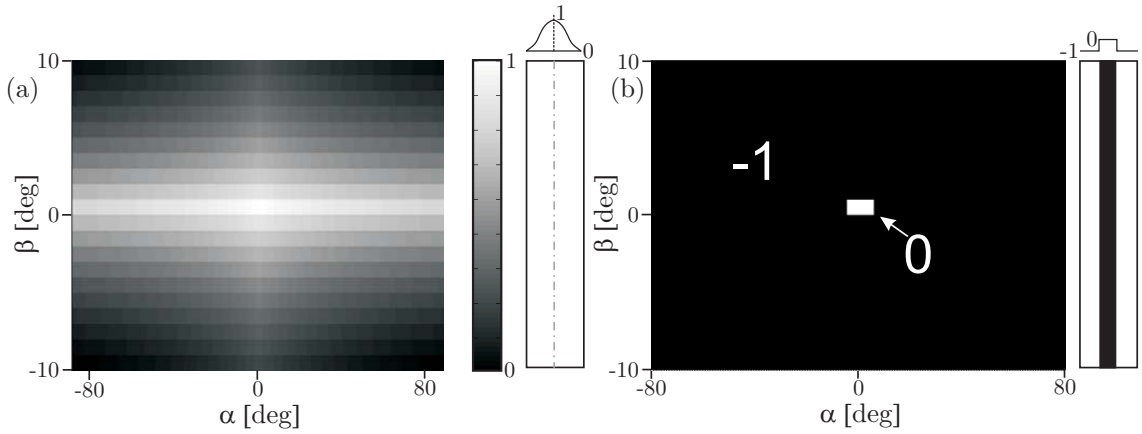


Figure 8.2.: Reward functions for different perceptual states with the action ω set to 0. (a) Dense reward and (b) sparse reward. Shown are also the pictograph of the reward with respect to the corridor.

is computed by summing the reward of T future time steps given by,

$$\mathrm{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots + \gamma^{T-1} r_{t+T} = \sum_{k=1}^{T} \gamma^{k-1} r_{k+1}. \tag{8.1.2}$$

γ is the discount factor and is set to 0.99.
The state value function (V^{Π}) of a policy Π starting from a state s is given by,

$$V^{\Pi}(s) = \sum_{a} \Pi(s,a) \sum_{s'} P^{a}_{ss'} \left[\mathrm{R}^{a}_{ss'} + \gamma V^{\Pi}\left(s'\right) \right]. \tag{8.1.3}$$

Equation 8.1.3 is often referred to as the Bellman equation [Bel57] which gives the value of the optimal policy in terms of the optimal action taken at the current state and then proceed to take optimal actions for the foreseeable future. The optimal policy is then required to learn the optimal state value V^* given by,

$$V^*(s) = \max_{\Pi} V^{\Pi}(s). \tag{8.1.4}$$

The action value function computes the expected future rewards based on the action to be executed while at a certain state s. The action value or the Q optimal value function is given by [SB98]; [SPK02],

$$Q^{\Pi}(s,a) = E_{\Pi}\left[\mathrm{R}(s,a) + \gamma \max_{a'} Q^*\left(s', a'\right) \right] \tag{8.1.5}$$

where s' is the next state reached on performing an action a at state s. The above equation gives the value function of taking an action a at state s and then acting optimally further on for T future time steps. Here, the selection of the action is not implicit. To learn the optimal policy, the optimal action value function Q^* need to be learned, given by,

$$Q^*(s,a) = \max_{\Pi} Q^{\Pi}(s,a). \tag{8.1.6}$$

Bellman optimality equation says that the value of being in a state and following an optimal policy must be the same as the value of choosing the best action while being in that state [SB98]. This allows to write the optimal value V^* as,

$$V^*(s) = \max_{a} \sum_{s'} P^{a}_{ss'} \left[\mathrm{R}^{a}_{ss'} + \gamma V^*\left(s'\right) \right] \tag{8.1.7}$$

where, $P^a_{ss'}$ gives the state transition probability of reaching state s' from state s with an action a. Equation 8.1.7 can also be written as,

$$V^*(s) = \max_a \left(\mathrm{R}(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V^* \left(s' \right) \right). \tag{8.1.8}$$

Fig. 8.3 shows one example of the value function computed for the rewards acquired by following an initial corridor following GMM. The figure shows the value function for different values of the perceptual variables α and β when the action variable $\omega = 0$. This corresponds to the robot always going straight for all misalignment and orientation errors along a corridor. The following section gives an insight into building the initial corridor following GMM.

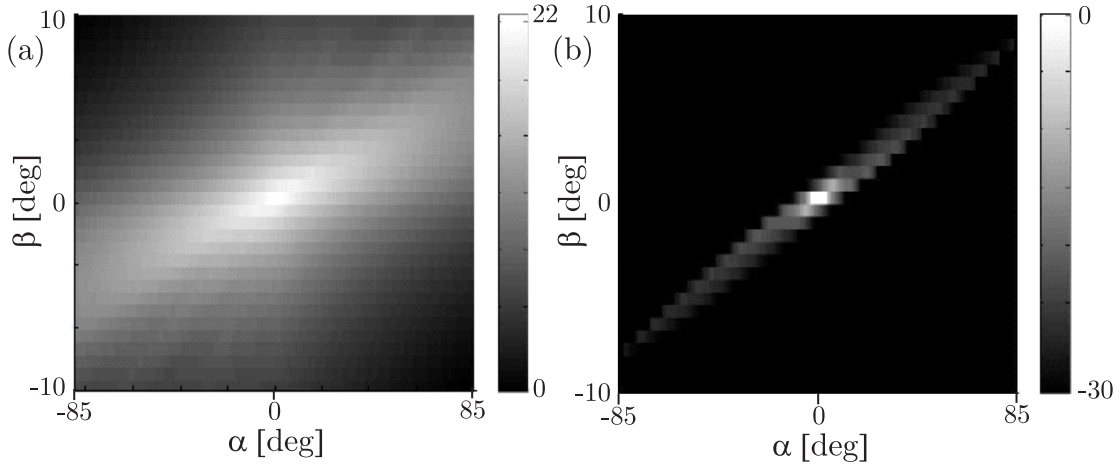


Figure 8.3.: Value function computed with (a) dense reward and (b) sparse reward.

8.1.1. Building up the initial behavior model

Visual corridor following dynamics is modeled using GMM as mentioned in chapter 7.1.1. In the previous chapter, we saw that optimizing the parameters of GMM using SEDS showed promising results in that it ensures the presence of one global attractor at the goal position. The non-linearity of behavioral dynamics is expressed as a sum of linear dynamical systems (See 7.1.3). During imitation, the prediction is done using GMR by regressing the GMM on the inputs. The limitations of GMM are the bias of the learned dynamics with the demonstration data and the corresponding necessity to present diverse set of demonstrations covering all the possible perceptual state space to learn the complete behavior dynamics. This aspect can be seen in Fig. 8.4 where training and reproduction of two GMM's, one trained with four demonstration examples and an another trained on a single demonstration example (dashed trajectory) are shown. From the figure, one can see that not all the demonstrations are perfectly executed. The end of the demonstration does not necessarily correspond to the corridor center. This aspect is typically seen in the reproduction of the GMM trained on four trajectories where the model in theory learns the essence of the behavior and moves towards the corridor center but not with the shortest and smoothest path. For model trained with only one trajectory, this is very

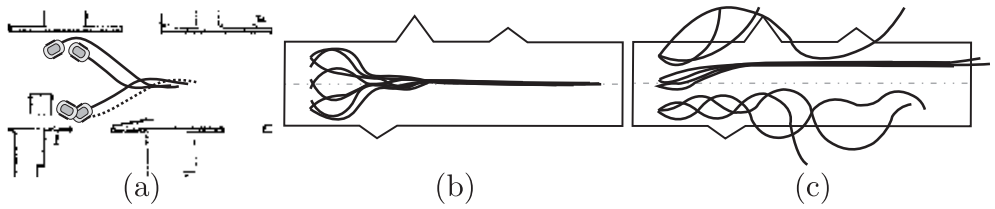


Figure 8.4.: Corridor following demonstrations and reproductions. (a) Trajectories used for demonstration, (b) reproduction from unknown starting positions with GMM trained on four demonstrations and (c) reproduction from unknown starting positions with GMM trained on single (dashed) demonstration.

apparent and the number to demonstrations does not suffice for generalizing the behavior from different unseen starting configurations.

8.1.2. Value iteration algorithm with Gaussian mixture model

The optimal policy with the GMM is determined by finding the optimal value function using value iteration algorithm [SPK02]. The value iteration algorithm for policy improvement is shown in Algorithm 1.

Data: Generate an initial GMM from demonstration examples.
while *Cauchy-Schwarz divergence between current GMM and previous GMM < 1* **do**
  for *all starting test configurations* **do**
    for *all possible actions* $a \in A$ *at current state* s *compute* $Q(s,a)$ *for* n *time steps* ($n \times$ dt *seconds) using current GMM behavior dynamics* **do**
      Q-value function: $Q(s,a) := \mathrm{R}(s,a) + \gamma \sum_{s \in S} P^{a}_{ss'} V_{k^{\Pi}}(s')$
    end
    $V_{k+1^{\Pi}}(s) := max_a Q(s,a)$;
    if *Final reward of policy* $\Pi > 0.55$ **then**
      Add trajectory from policy Π from state s for future training;
    else
      Reject trajectory and continue;
    end
  end
  Improve policy with new training trajectories;
end

Algorithm 1: Value Iteration Algorithm using GMM

Assuming the robot starts from state $\mathrm{S}_1 = [\omega_1, \alpha, \beta]$, the breakdown of the steps involved in arriving at the best appropriate action for the state using RL is shown in Fig. 8.5. Details of the individual steps are explained below.

Step 1 : Looping over a discrete set of actions

For the current perceptual state (α, β) and the action ω executed to arrive at the state, vary the next executable action around the current value to determine the optimal action for the state. Note, that the number of rotational velocities that can be executed at any specific state for a mobile robot is continuous and a very large value. It becomes computationally very expensive to check the optimal value function for all possible actions for every state. Hence, only the neighborhood of the current rotational velocity is checked to

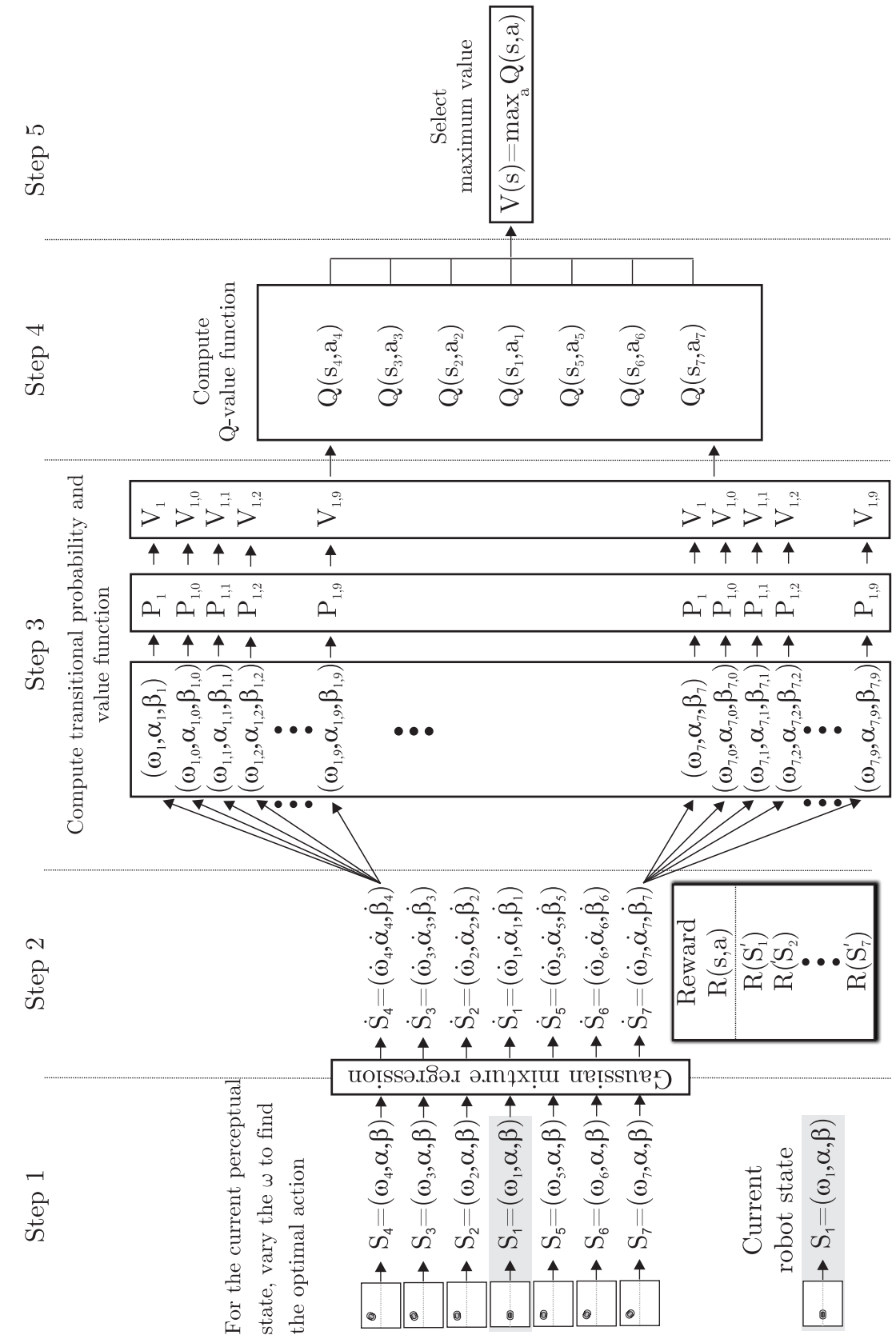


Figure 8.5.: Schematic of value iteration algorithm with an initial GMM for learning corridor following behavior.

find the optimal action. Anything outside the neighborhood indicates an evasive behavior which is fundamentally not desirable for a mobile robot behavior. Therefore, rotational velocity states within 3 [deg/sec] of the current state with a resolution of 1 [deg/sec] are searched. Varying ω around the current states gives us,

$$\begin{bmatrix}\omega_1 & \omega_2 & \omega_3 & \omega_4 & \omega_5 & \omega_6 & \omega_7\end{bmatrix} = \begin{bmatrix}\omega & \omega-3 & \omega-2 & \omega-1 & \omega+3 & \omega+2 & \omega+1\end{bmatrix}$$

corresponding to the set of states $S_1 \dots S_7 \in S$.

$$\mathrm{S} = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3 & \omega_4 & \omega_5 & \omega_6 & \omega_7 \\ \alpha & \alpha & \alpha & \alpha & \alpha & \alpha & \alpha \\ \beta & \beta & \beta & \beta & \beta & \beta & \beta\end{bmatrix}.$$

Please note that one can vary the number of neighborhood actions depending on the specificity of task and the computational complexity of the search and the method would be equally applicable.

Step 2 : Determine the next state

For every state, apply GMR to obtain the next state. The learned dynamic model predicts the next state dynamics in terms of angular acceleration $\dot{\omega}$ and the rate of change of perceptual variables $\dot{\alpha}$ and $\dot{\beta}$. The conditional expectation of the output $\dot{\omega}$ from the perceptual input states $(\dot{\alpha}, \dot{\beta})$ can be approximated by a single Gaussian distribution $\mathcal{N}(\hat{\xi}, \hat{\Sigma})$ with a mean centered around the output $\hat{\xi}$ and covariance $\hat{\Sigma}$ given by [Cal09],

$$\hat{\xi} = \sum_{k=1}^{K} h_k \hat{\xi}_k, \quad \hat{\Sigma} = \sum_{k=1}^{K} h_k^2 \hat{\Sigma}_k \tag{8.1.9}$$

where h_k is the probability that the Gaussian k is responsible for the input $\xi^{\mathcal{I}}$. For the computed dynamic states $\dot{S}_1 \dots \dot{S}_7$, the next state is computed as,

$$S_i' = \dot{S}_i \mathrm{d}t + S_i, \text{where} \quad i = 1 \dots 7.$$

Here, dt is the time period used for recording the behavioral dynamics. For a state action pair the corresponding sparse or dense reward $\mathrm{R}(s, a)$ is computed here.

Step 3 : Compute transitional probability

Transitional probability $P_{ss'}^{a}$ involves computing the probability of reaching the state s' from current state s on performing an action a. This probability reflects the stochastic nature of performing an action. Transitional probability hence in a way describes the dynamics of the world. It is quite difficult to give a specific value to the transitional probability, but since the dynamics is described in terms of Gaussian, we sample 10 random samples from the distribution to represent the stochasticity. Thus for every state, one gets additional states thereby increasing the total number of states to 77. From the computed transitional probabilities, the corresponding value function can now be computed directly.

Step 4: Compute the Q-value

This is continued into the future for n time steps. We limit ourselves to 40 future time steps for sparse reward and 25 future time steps for dense reward. Working with a dt of 0.3 seconds, this corresponds to looking 12 seconds into the future for sparse reward and 7.5 seconds for dense reward. Experiments reveal that with dense reward function

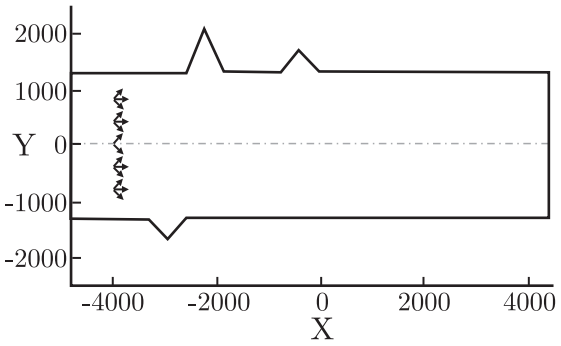


Figure 8.6.: The fourteen different configurations used for RL.

fewer steps suffice compared to working sparse rewards. Thus, the Q-value for each of the initial states are computed as in 8.1.10.

$$Q(s,a) = \mathrm{R}(s,a) + \gamma \sum_{s' \in S} P^{a}_{ss'} V^{\Pi}_{k}(s'). \tag{8.1.10}$$

Step 5 : Select best action
The best action for the current state is thus determined from the highest Q-value.

$$V^{\Pi}_{k+1}(s) = \max_{a} Q(s,a).$$

The above mentioned steps described the methodology used to select the most appropriate action for one state s. We perform this for 14 different starting configurations in the corridor. The configurations are ,

$$\begin{bmatrix} \omega \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 45 & -45 & 0 & 45 & -45 & 0 & 45 & -45 & 45 & -45 & 0 & 45 & -45 & 0 \\ -8 & -8 & -6 & -4 & -4 & -4 & 0 & 0 & 4 & 4 & 4 & 8 & 8 & 8 \end{bmatrix}.$$

From every starting configuration, value iteration is performed to collect the optimal trajectory data in simulation for 60 time steps. We agree that there exists many starting configurations that can be used to generate the data to test RL, but we choose the above 14 manually since they cover the majority of the behavioral state space. Fig. 8.6 visualizes these starting configuration within a standard corridor design used in simulation. The trajectories generated from RL based on the learned GMM is then used to update the policy. Two approaches are used to update the policy.

Policy Update and Improvement

The trajectory data generated from RL value iteration starting from the 14 configurations are used for improving the model. Two alternatives for policy improvement are tested:

1. **New model:** Build a completely new model from the collected data or
2. **Optimized model:** The mean and covariance of the previous model is used to initialize and optimize GMM model with new trajectories.

The difference between the two is in the initialization of the mean and covariance of the Gaussian. With a new model, the mean and covariance are computed through clustering the training data to set the Gaussian parameters using Expectation Maximization (EM) algorithm [Cal09] and learn the dynamics from SEDS. For optimizing the model, the first step is skipped and the parameters of the already existing GMM is used to initialize and update.

Stopping condition

The termination criteria to stop updating the GMM is done by computing the Cauchy Schwarz divergence (D_{cs}) between the two mixture of Gaussians [KHP11]. D_{cs} computes the difference between two probability distribution functions given by,

$$D_{cs}(q,p) = -\log\left(\frac{\int q(x)p(x)\mathrm{d}x}{\sqrt{\int q(x)^2\mathrm{d}x \int p(x)^2\mathrm{d}x}}\right) \tag{8.1.11}$$

where, p and q are two mixture of Gaussian distributions. This measurement is symmetric and obtains a value of zero only if $q(x) = p(x)$. The divergence value is similar to the established Kullback-Leibler divergence [KL51] but carries an advantage that it is computationally efficient and can handle high dimensions easily. By comparing the two adjacent GMMs emerging from RL the relative change in the model is detected. We set a D_{cs} threshold of 1 to establish no further significant change in the models and correspondingly no improvements in the model reproduction.

We establish the principle of RL on the model shown in Fig. 8.4(b). Trained on four demonstrations on a corridor, the imitation trajectories are not optimal and overshoot the corridor center initially. This GMM is subject to RL value-iteration algorithm on the fourteen test configurations with dense rewards. The trajectories generated from the value iteration algorithm are then used as training data for updating the policy. Imitation trajectories of the policy in simulation are shown compared to their preceding model imitation in Fig. 8.7. At every iteration, the D_{cs} value between the current GMM and the previous GMM is computed to check for the termination condition. Table 8.1 tabulates the D_{cs} values at every iteration of RL. The values and the imitation show that optimized policy is slower in driving the robot the corridor center. With a new model, the convergence is very quick mainly due to the elimination of the parameter bias of the previous GMM to the initial training data. For policy development with a new model, the parameters of the initial corridor following model is used only for the first iteration whereas for an optimized policy update, the bias of the initial model is still retained to adapt to the new data. The Q-value of the new model and optimized model is compared with the original model for a trajectory (Initial configuration $[\omega, \alpha, \beta]$ =[0 deg/s, 45 deg, -0.5 deg]) shown during training and iteration can be seen in Fig. 8.8. From the figure one can observe that the Q-value of the updated model rises up quite fast indicating an execution of a policy that promises a long time positive reward.

The effect of RL with a sparse reward function where only the goal point is rewarded with 0 and the rest of the states are penalized with a -1 is tested next. Fig. 8.9 shows the traced trajectories in simulation working with sparse reward on an initial model trained on the same four trajectories. The figure shows the trajectories traced for a new GMR model. From the figure one can notice, that with a sparse trajectory the total time needed to reach the corridor center might be a little longer than the model learned with dense reward, nevertheless still generates very smooth trajectories. Only three iterations

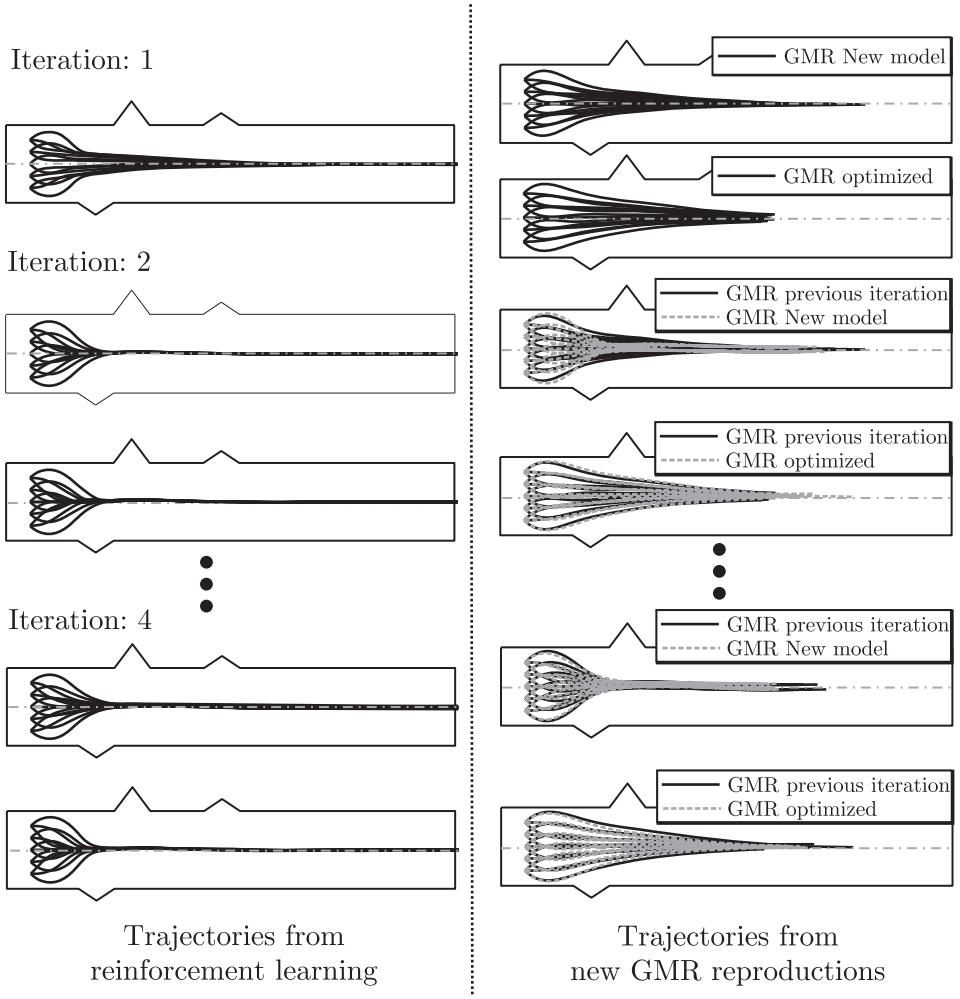


Figure 8.7.: Number of demonstrations: 4, reward function: dense. (Left) Trajectories traced during value-iteration algorithm and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

suffice here to learn the behavior (See Table 8.2) and the Q-values for the initial and the final GMM is shown in Fig. 8.10. The Q-values in the figure clearly show the lack of the initial GMM model to execute the most optimal action. With a sparse reward, the path which is the shortest and fastest to the corridor center gets the highest long time reward. The improvement in the Q-value attained is immediately apparent during the first iteration of RL where the maximum Q-value is attained in approximately 17 seconds. The final model after the third iteration shows the best policy performance and attains the maximum Q-value in 12 seconds. The generalization ability of the final model is tested in two unknown environments. Fig. 8.11 shows two new environments not used either for demonstration or RL. The first scenario changes the corridor width and the second corridor has different width as that of demonstration corridor and changes the direction of alignment in the midway. The original GMM model together with the reinforcement learned GMM model for both sparse and dense trajectory is shown. Both the models perform much smoother than the original model at the beginning, with the optimized model taking relatively longer to reach the corridor center. This confirms and lays the conceptual basis in making RL work for visual corridor following behavior. The limits of this algorithm are pushed a little further with one shot learning where only a single

Table 8.1.: Number of demonstrations: 4, reward function: dense. D_{cs} values between the GMMs at every iteration. Initially 4 trajectories are used for training and value-iteration is done with a dense reward function.

Iteration	D_{cs}	
	New model	Optimized model
0-1	8.7403	8.8897
1-2	1.2985	1.8624
2-3	1.1634	2.0187
3-4	0.4650	0.8582

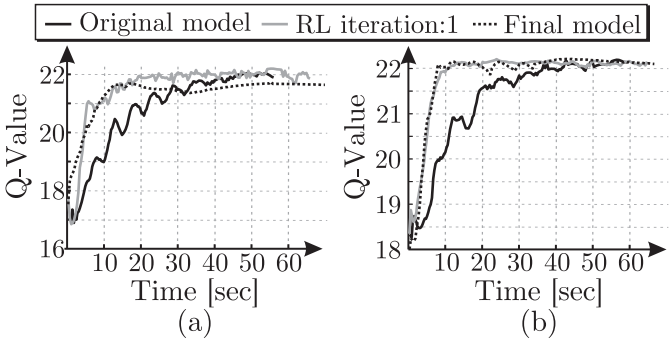


Figure 8.8.: Q-value recorded for the trajectory starting with an initial perceptual state and action of $\alpha = 45$ deg, $\beta = -0.5$ deg and $\omega = 0$ deg/sec.The Q-values for 60 sec imitation done with the initial GMM, RL of the initial GMM and the final GMM. The final GMM in (a) is a new GMM and in (b) is the optimized GMM.

demonstration is used to teach the behavior.

8.1.3. One shot learning

One shot learning focuses on learning the behavioral essence with just a single demonstration. The demonstration trajectory shown in Fig. 8.4(c) is used here. Taking this model as the initial policy for corridor following, RL is performed with both dense and sparse reward function to check for its feasibility. The lack of information in the initial model does not guarantee a successful RL trial. In other words, the sub-optimal nature of the initial model more often tends to predict the next state which inevitably leads the robot out of the corridor. Hence, we perform a two level filter of trajectories where the quality of the trajectories are tested. At every iteration, if the attained final reward of the trajectory is less than a threshold of 0.55, we neglect it. In the event where none of the trajectories from the fourteen starting configurations qualify for the next iteration,

Table 8.2.: Number of demonstrations: 4, reward function: sparse. D_{cs} values between the GMMs at every iteration. Initially 4 trajectories are used for training and value-iteration is done with a sparse reward function.

Iteration	D_{cs} New model
0-1	5.4289
1-2	1.1465
2-3	0.6193

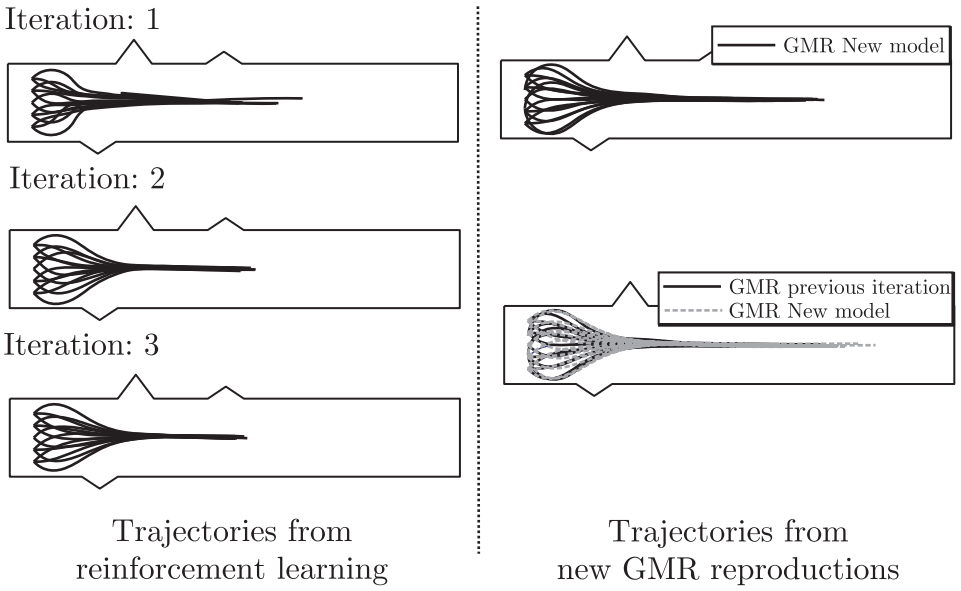


Figure 8.9.: Number of demonstrations: 4, reward function: sparse. (Left) Trajectories traced during value-iteration algorithm using a sparse reward function and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

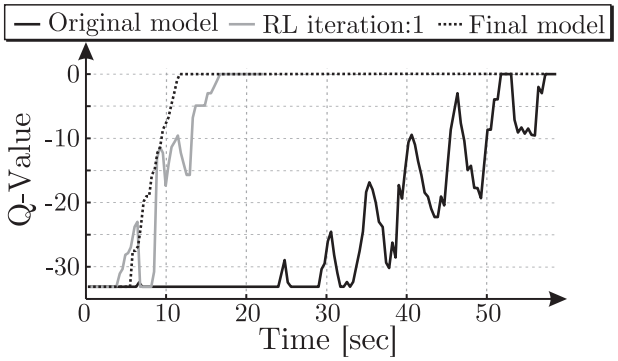


Figure 8.10.: Q-value recorded for the trajectory starting with an initial perceptual state and action of $\alpha = 45$ deg, $\beta = -0.5$ deg and $\omega = 0$ deg/sec.The Q-values for 60 seconds imitation done with the initial GMM, RL of the initial GMM and the final GMM. The final GMM is a new GMM modeled on the trajectories from RL.

ten new configuration which are intentionally set closer to the corridor center are tested. Being set closer to the corridor center means at least one or two trajectories would end up closer to the corridor center. This work around ensures that we do not run out of trajectory training data to bring knowledge to the model. The ten new starting configurations are,

$$\begin{bmatrix} \omega \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & -10 & 20 & -20 & 30 & -30 & 20 & -20 & 10 & -10 \\ -4 & -4 & -1 & -1 & 0 & 0 & 1 & 1 & 4 & 4 \end{bmatrix}.$$

We show here the results of policy improvement carried out by generating a new model from the RL trajectories. The methodology is equally applicable for policy update through optimized parameters. From Fig. 8.12, one can immediately infer that not all the configuration starting positions perform good and thereby considered for the next iteration. With the initial model, seven out of the fourteen standard tests fail and are further tested

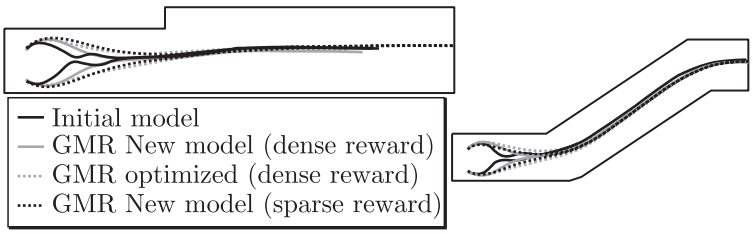


Figure 8.11.: Number of demonstrations: 4. Trajectory reproductions in unknown environments using the different models trained on four corridor following demonstrations.

Table 8.3.: One shot learning with dense rewards. D_{cs} values between the GMMs at every iteration. Also shown are the number of successful RL trajectories at every iteration.

Iteration	D_{cs}(old, new)	Successful trajectories out of 14 configurations	Successful trajectories out of 10 additional configurations
0-1	9.0475	7	7
1-2	2.8127	3	8
2-3	3.6857	14	0
3-4	1.1079	14	0
4-5	1.0308	14	0
5-6	0.9998	14	0

on the ten new configuration on which again only seven out of the ten pass. The second iteration again rejects eleven trajectories nevertheless the remaining data collected possess good quality in that the third iteration almost immediately achieves a high generalization. Table 8.3 tabulates the D_{cs} and the number of qualified trajectories in every iteration. A total of six iterations are done to generate the final model. Fig. 8.13 shows the policy improvement reflected through the Q-value of the reproduction. Q-values of two experimental runs, one starting from a known configuration $[\omega, \alpha, \beta] = [0 \text{ deg/s}, 45 \text{ deg}, 4 \text{ deg}]$ and the second one from unknown configuration with a very large misalignment angle $[\omega, \alpha, \beta] = [0 \text{ deg/s}, 85 \text{ deg}, 3 \text{ deg}]$ are shown. The Q-value for the initial model remains between 16 and 18 without any big improvement over the course of time. The model after just one iteration already shows a better performance but still exhibits oscillations. The final model outperforms the other two courtesy RL and the exhibits a stable behavior in terms of the optimal action chosen for the visited perceptual states. The difference is more apparent for the second test starting from an initial configuration not shown during demonstration. The value function of the initial model drops precipitously due to the lack of training data. The final RL model confirms the optimality with a steady increase in the Q-value and exhibits stability in performance.

With a sparse reward function, learning takes longer and in that not all the starting configurations are considered to improve the model until the fourth iteration. Fig. 8.14 shows the qualified trajectories used for training at the first, second and final iteration. Check Table 8.4 for the corresponding D_{cs} for the new model in every iteration. The figure shows that with just a sparse reward function and with only one demonstration of the behavior, the proposed learning architecture with RL learns the essence of the behavior quite robustly. Fig. 8.15 shows the corresponding Q-values registered for a simulation run within a corridor with one known and one unknown starting configuration similar to the one with dense reward. With the initial model, the value function shows

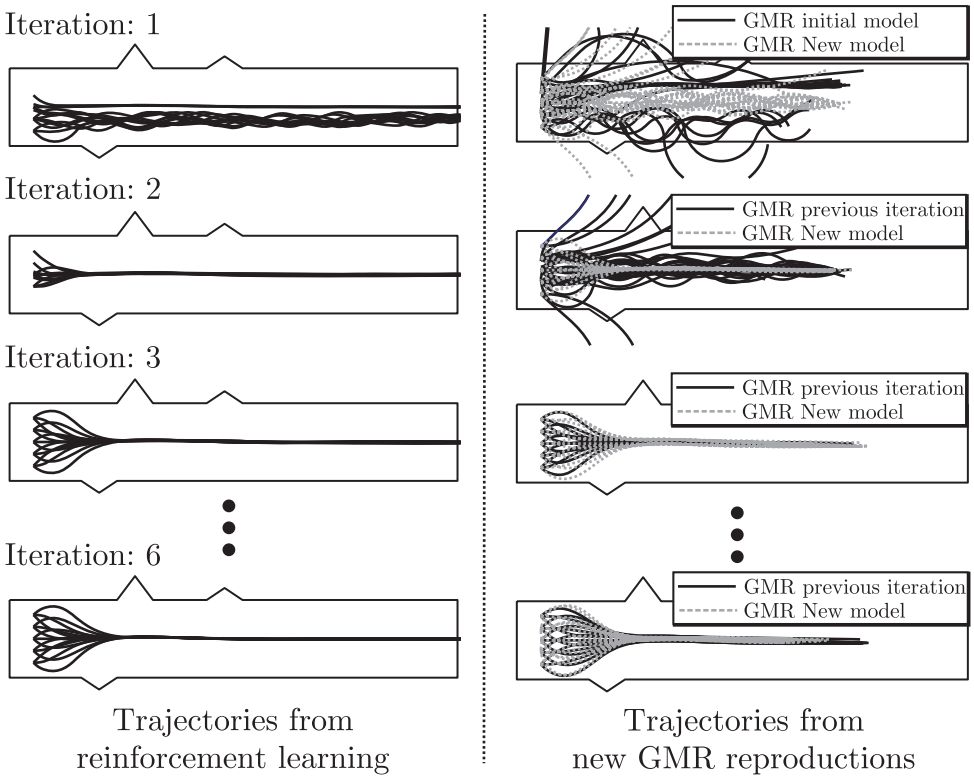


Figure 8.12.: The trajectories traced for the different initial configuration during one-shot RL with dense reward function. (Left) Trajectories traced during value-iteration algorithm using a dense reward function and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

little to no improvement for both the starting configurations. This is obvious in that the reproductions of the initial model never drive the robot to the corridor center thereby only accumulating more penalties. With the RL model, the Q-value is quickly improved and reduces to zero indicating no long term penalties. The one-shot learned models are tested in two unseen environment as shown in Fig. 8.16. The performance of the original model is as predicted very poor with the model exhibiting no characteristic of the behavior. The nature of the reward function used to learn the behavior is also seen in that the GMR model learned with sparse rewards requires longer time to reach the corridor center than the model learned with dense reward. Nevertheless, both the models traverse smooth trajectories and exhibit robustness in novel environments.

Table 8.4.: One shot learning with sparse rewards. D_{cs} values between the GMMs at every iteration. Also shown are the number of successful RL trajectories at every iteration.

Iteration	$D_{cs}(\text{old}, \text{new})$	Successful trajectories out of 14 configurations	Successful trajectories out of 10 additional configurations
0-1	8.9556	7	6
1-2	1.2355	9	8
2-3	1.8160	9	8
3-4	1.6793	14	0
4-5	0.4150	14	0

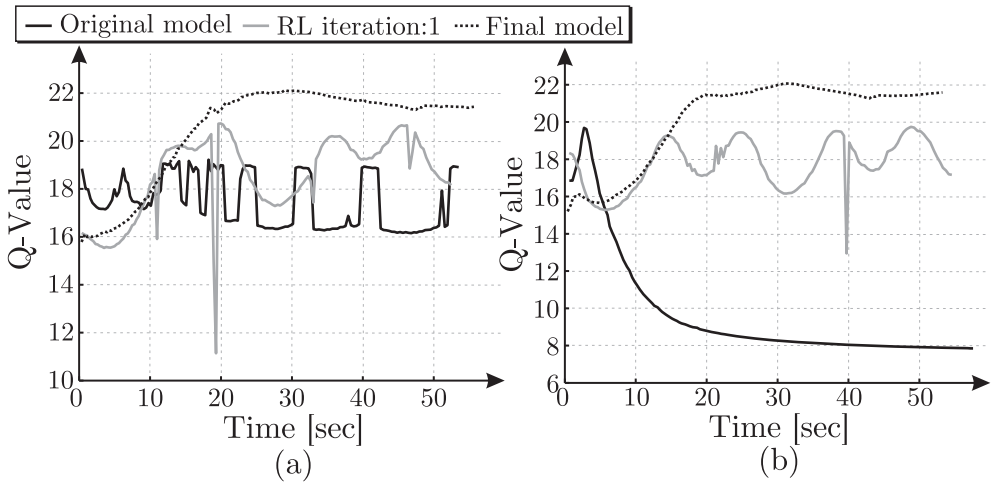


Figure 8.13.: One shot learning with dense rewards. Q-value recorded for the trajectory starting with an initial perceptual state and action of (a) $\alpha = -30$ deg, $\beta = -3$ deg and $\omega = 0$ deg/sec and (b) $\alpha = -85$ deg, $\beta = 3$ deg and $\omega = 0$ deg/sec. The Q-values for 60 seconds imitation done with the initial GMM, RL of the initial GMM and the final GMM.

8.2. Learning from scratch

Learning from scratch deals with the problem of acquiring knowledge about a task with no prior information available. This is a daunting task as there are no example demonstrations of the task provided. Nevertheless, the available reward information makes this problem a trial and error learning of the value function. A dummy initial GMM of corridor following behavior is generated using a blind demonstration. The model as such does not contain any meaningful information about the behavior. The trajectory used and the corresponding reproduction along with the Gaussians are shown in Fig. 8.17. The RL iterations are shown in Fig. 8.18. From the figure one can see that the initial model drives the robot simply straight irrespective of the orientation of the robot. The RL trajectories in the first iteration thus only slightly varies and thus only the trajectories that are within the corridor premises are considered for training and updating the policy. With more iterations, more trajectories move towards the corridor center with the sixth iteration successfully converging all the fourteen trajectories. The termination conditions at every iteration are shown in Table 8.5. The D_{cs} value during the third iteration fall below the threshold of 1, but the training is continued since not all of the fourteen test configurations qualified for policy update.

Learning a policy with no prior knowledge and with only a sparse reward presents the toughest challenge until now. The RL trajectories confirm this (See Fig. 8.19). In the first iteration, all the trajectories fail and have a low Q-value of -33.1028. Since none of the tested trajectories pass the qualification, we are forced to take all the failed trajectories to generate a new model. At this instance, the model is literally presented only with failed examples. The second iteration also does not succeed with any of the trajectories but the policy drives three trajectories forward which get a little better Q-value. After this point, every iteration improves from the previous one and the behavior is successfully learned during the seventh iteration where all the starting configurations drive to the corridor center (See also Table 8.6). Starting with absolutely no knowledge of the behavior and starting from almost inconsequential demonstration, RL exhibits significant success in

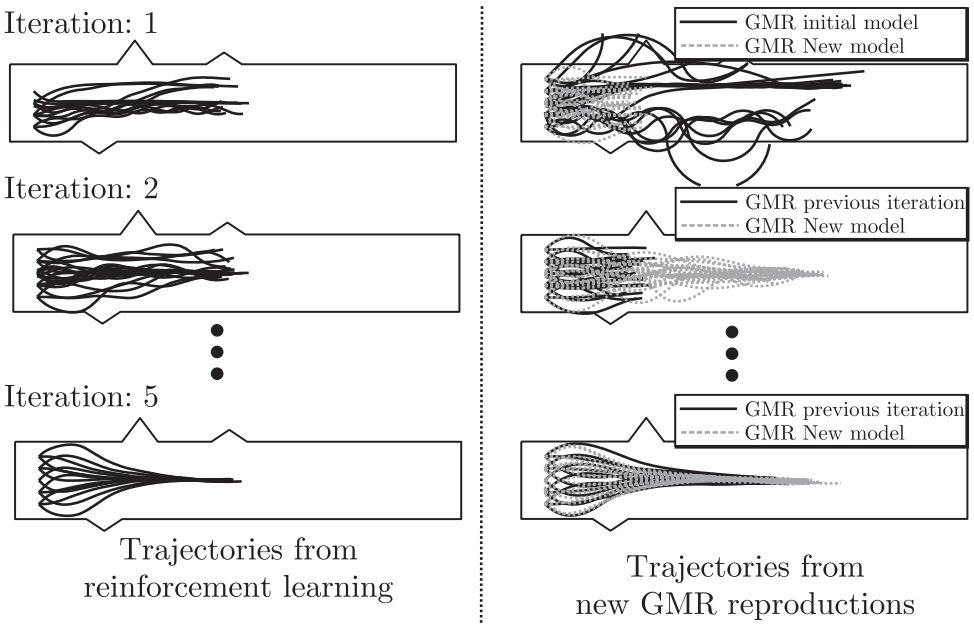


Figure 8.14.: The trajectories traced for the different initial configuration during one-shot RL with sparse reward function. (Left) Trajectories traced during value-iteration algorithm using a sparse reward function and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

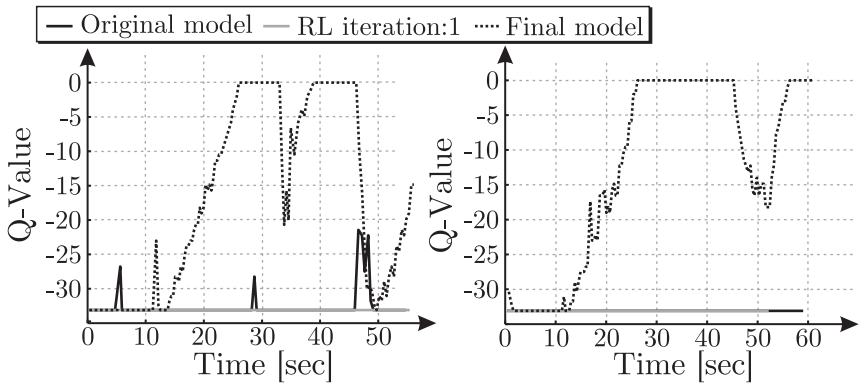


Figure 8.15.: One shot learning with sparse rewards. Q-value recorded for the trajectory starting with an initial perceptual state and action of (a) $\alpha = -30$ deg, $\beta = -3$ deg and $\omega = 0$ deg/sec and (b) $\alpha = -85$ deg, $\beta = 3$ deg and $\omega = 0$ deg/sec. The Q-values for 60 seconds imitation done with the initial GMM, RL of the initial GMM and the final GMM.

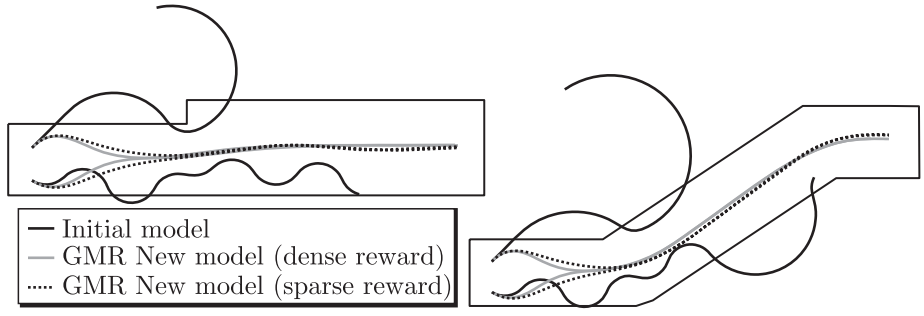


Figure 8.16.: One shot RL reproductions in unknown corridor environments.

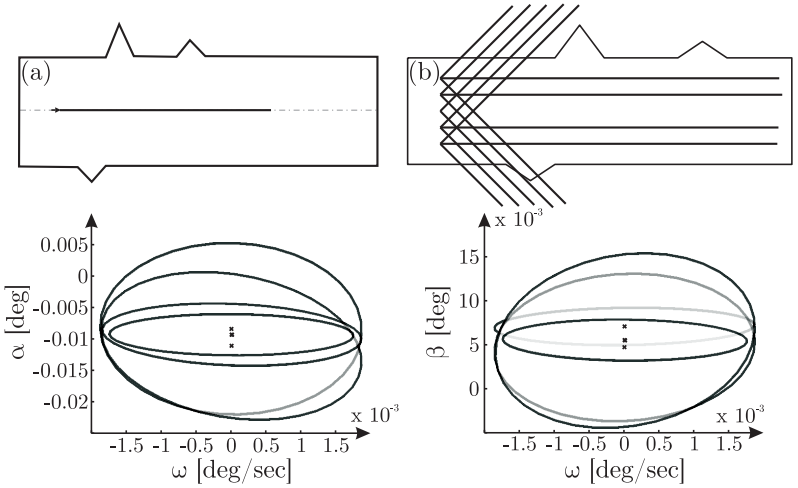


Figure 8.17.: The demonstration and the initial model used for the first iteration of learning from scratch. (a) Demonstration trajectory and (b) reproductions from different initial configurations.

Table 8.5.: Learning from scratch with dense rewards. D_{cs} values between the GMMs at every iteration. Also shown are the number of successful RL trajectories at every iteration.

Iteration	D_{cs}(old, new)	Successful trajectories out of 14 configurations	Successful trajectories out of 10 additional configurations
0-1	7.9473	3	5
1-2	6.8063	6	8
2-3	0.8722	10	10
3-4	1.6151	14	0
4-5	1.2892	14	0
5-6	0.9957	14	0

learning the corridor following behavior. Fig. 8.20 establishes the robustness of the new model further through the two successful tests in unknown environments.

8.3. Learning time

Value-iteration and policy update is performed offline with Matlab. Irrespective of the software specific performance measures, the training times presented here give a general outlook into the complexity of the learning process. Table 8.7 gives the list of durations recorded for training different corridor following models. As expected, the nature of reward function and the number of demonstrations used for the initial model plays a significant role in determining the total training time. One shot learning and learning from scratch required the highest amount of training time. Learning from scratch with a sparse reward function took the most training time of around 93 minutes. The presented table are the time registered to build a new model at every iteration rather than optimizing the model.

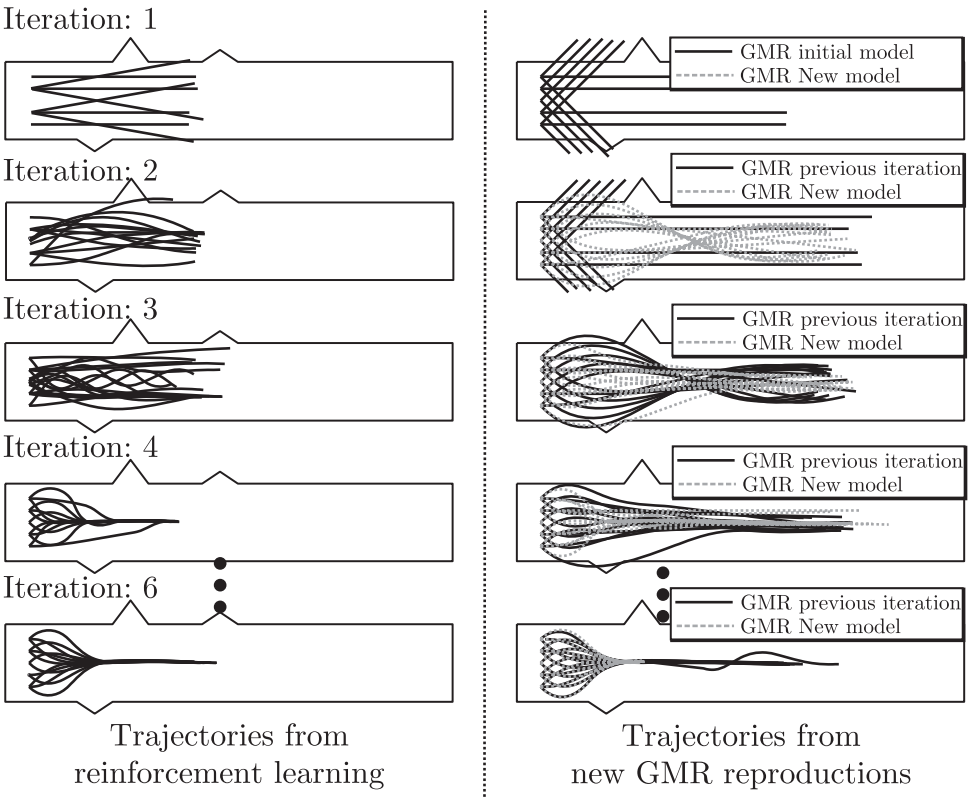


Figure 8.18.: The trajectories traced for the different initial configuration during learning from scratch RL with dense reward function. (Left) Trajectories traced during value-iteration algorithm using a dense reward function and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

8.4. Experimental results

The models discussed above are tested on the real robot in a corridor environment. The translational velocity for the experiments are kept constant at 60 mm/s. The demonstrated trajectories for these experiments are the same one as shown in Fig. 8.4. Experiments are first performed for different starting configurations on the policy trained initially with four demonstrations with optimized GMM and a new GMM. Fig. 8.21 shows the overlay of the traced robot trajectory for the different models with two initial configurations $[\omega, \alpha, \beta] = [0 \text{ deg/s}, 45 \text{ deg}, 5 \text{ deg}]$ and $[0 \text{ deg/s}, -45 \text{ deg}, -5 \text{ deg}]$. Working with the initial GMM, the robot oscillates around the corridor center by alternatively predicting the maximum and minimum angular velocity (See Fig. 8.21). The RL models with both dense and sparse rewards drive the robot quite robustly to the center. The optimized GMM takes a longer path to reach the corridor center than the new model. A significant difference in the performance is with the next experiments with one shot and learning from scratch model. Fig. 8.22 shows the reproduction of the original GMM trained on only one trajectory which fails to counter the environment and crashes on the adjacent walls. The reproduction of the final model using both dense and sparse reward show a very good performance and drives the robot robustly to the corridor center. Shown in the figure are also the registered state-action variables. Similar results are also seen with learning from scratch models. Starting from the same initial configuration (See Fig. 8.23), the original GMM model does not possess any prior knowledge about angular ve-

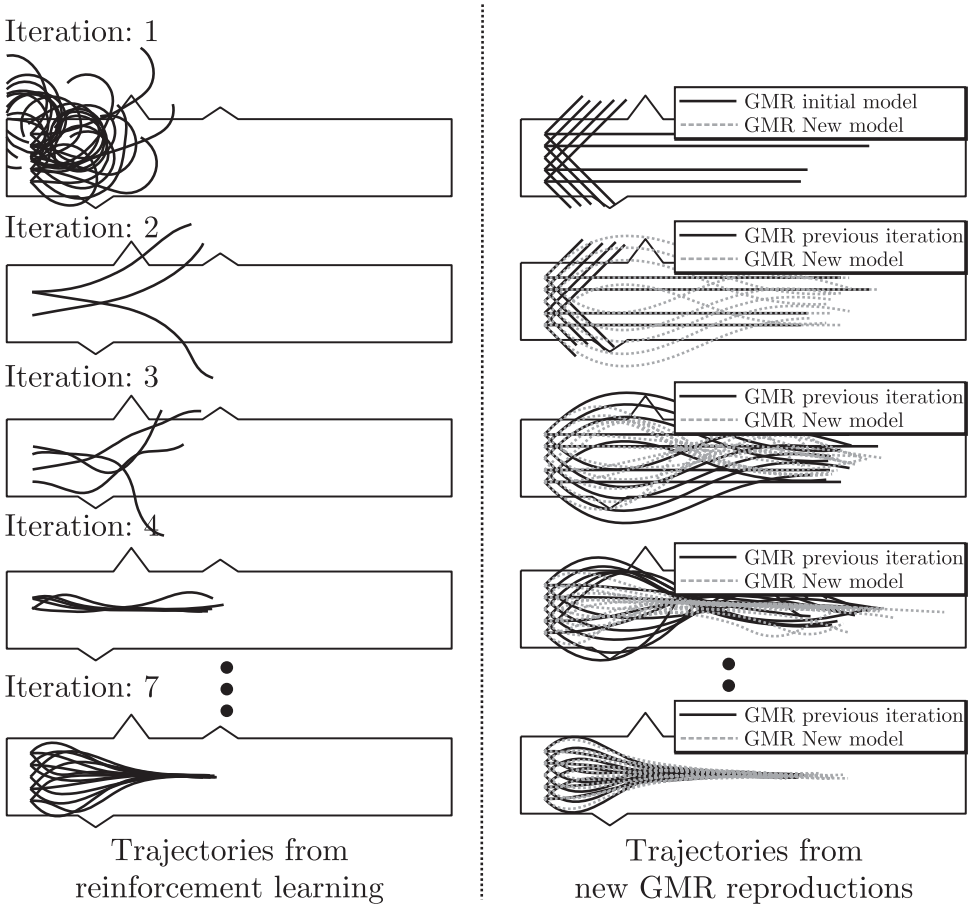


Figure 8.19.: The trajectories traced for the different initial configuration during learning from scratch RL with sparse reward function. (Left) Trajectories traced during value-iteration algorithm using a sparse reward function and (Right) trajectories traced by the updated corridor following model trained on the trajectories generated from value-iteration algorithm.

locity resulting in crash. The sparse and the dense model on the other hand traverse the corridor quite successfully.

8.5. Related work

Reinforcement learning for mobile robotics and autonomous vehicles has been of constant interest over the last decade [For02]; [RPB09]; [Kob12]; [KBP13]. The most famous work in this area is published by [SPK02], where two robotic behavior policies namely corridor following and obstacle avoidance are learned through value function approximation. The learning is divided into two phases. In the first phase, while the robot is being driven with a controller or a joystick, the learning system passively observes the states, actions and rewards and uses them to approximate the value function. The second phase starts when the value function is complete enough to control the robot effectively. The authors use an instance based algorithm based on Locally Weighted Regression algorithm [SPK00] to approximate the value function. Another related work in this area is the work of [CP07], where the authors use RL to determine an optimal path from a given initial position to a goal position in an indoor environment. Using a supervisor table, human demonstrations are recorded for action-state pairs. With just four actions: stop, move forward, turn left

Table 8.6.: Learning from scratch with sparse rewards. D_{cs} values between the GMMs at every iteration. Also shown are the number of successful RL trajectories at every iteration.

Iteration	D_{cs}(old, new)	Successful trajectories out of 14 configurations	Successful trajectories out of 10 additional configurations
0-1	21.097	14	10
1-2	5.2371	1	2
2-3	4.9908	1	3
3-4	5.5152	1	4
4-5	2.2305	6	0
5-6	1.2413	14	0
6-7	0.9906	14	0

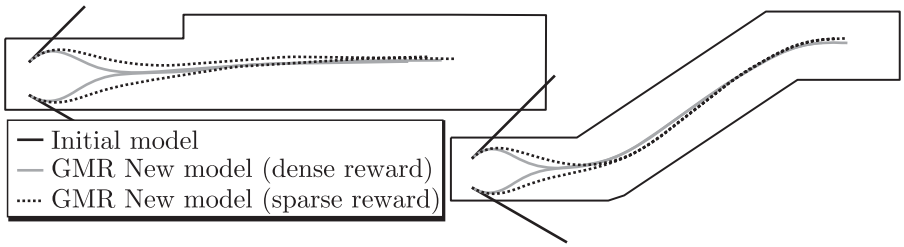


Figure 8.20.: Learning from scratch with RL reproductions in unknown corridor environments.

and turn right, the Q-value of the demonstrations are saved. Further work in RL used in mobile robotics are [MMD05]; [JLMMD06] who use it for an image based visual servoing framework working on a docking behavior. For using RL in a continuous state space as in a perception-action state space for a mobile robot, solutions such as classical grid methods to save the Q-value function at every step becomes impossible [SB98]. We alleviate this problem by using a non-parametric probabilistic method such as GMM to build a policy. The general challenge of learning with function approximation with Gaussian kernels is the estimation of the number of mean centers and widths of the Gaussians [KBP13]. By using LfD, these data can be determined using the demonstration examples collected prior to the RL. This approach has been used by [GHCB07] for an open-loop reaching movement and for a closed-loop cart-pole swingup task by [DR11]. Our approach is similar to the works of [GHCB07] in that both learn a Gaussian model of the task dynamics from the

Table 8.7.: Training times when using different demonstration trajectories.

Number of demonstrations	Reward function	Number of iterations	Total training time [min]
14	Dense	2	12
6	Dense	5	30
4	Dense	4	24
4	Sparse	3	27
1	Dense	6	44
1	Sparse	5	63
scratch	Dense	6	48
scratch	Sparse	7	93

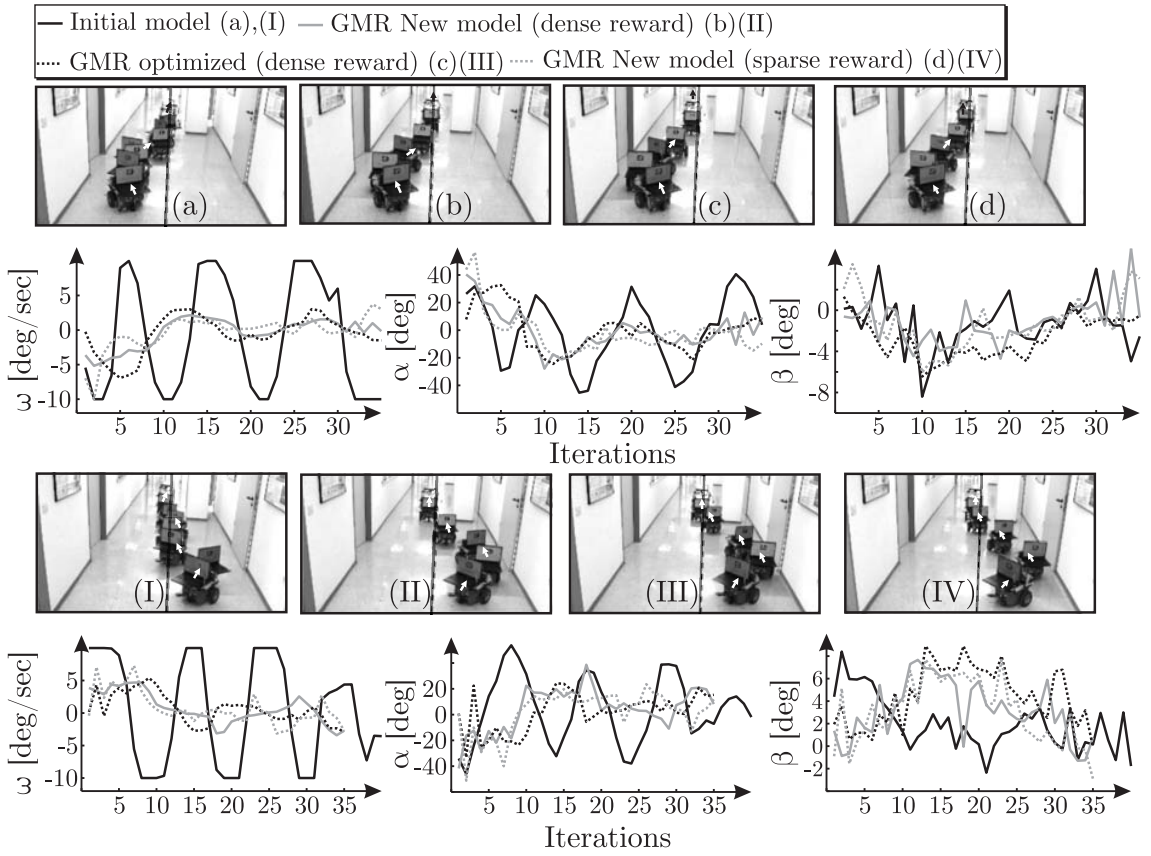


Figure 8.21.: GMM trained with four demonstrations. (a),(I) Reproduction of the initial model starting from $[\omega, \alpha, \beta]$ =[0 deg/s, 45 deg, 5 deg] and [0 deg/s, -45 deg, -5 deg], (b),(II) Reproduction of new GMM trained with dense reward, (c),(III) reproduction of optimized GMM trained with dense reward and (d), (IV) reproduction of new GMM trained with sparse reward.

demonstration data. In [GHCB07], the trajectory is generated using a dynamical system modulated by a velocity profile from demonstrations on a humanoid robot. We express the robot motion patterns through behavioral dynamics, where the relationship between the dynamics of the perceptual variable and the robot velocity is modeled using a GMM. The authors of [DR11] propose a PILCO (Probabilistic Interference for Learning Control) algorithm which uses Gaussian Processes to successfully learn from scratch. In [RK03], the learning task with RL and GMM in a continuous state space is accomplished using the policy iteration algorithm for the mountain car problem. They model the value function is modeled with a GMM and it is believed the generalization properties and the ease in manipulating the GMM parameters make them ideal for the corresponding task.
Another influential aspect affecting the performance of RL is the design of reward function. [TB06] analyze this aspect where a human feedback is used to add more information into learning not only from past executed actions but also to provide rewards directed to future subsequent actions that can be taken from the current state. Human feedback within a learning from demonstration framework for policy update is employed by [ABV07], where the robot on encountering novel environments with low confidence values turns towards the teacher to provide extra demonstrations. This way the policy is updated in areas of state spaces not seen during demonstration. For a mobile robot performing a corridor following behavior, we design the reward function in two ways. With a dense reward, a high human input is necessitated in that a non-zero reward is given to all state-action

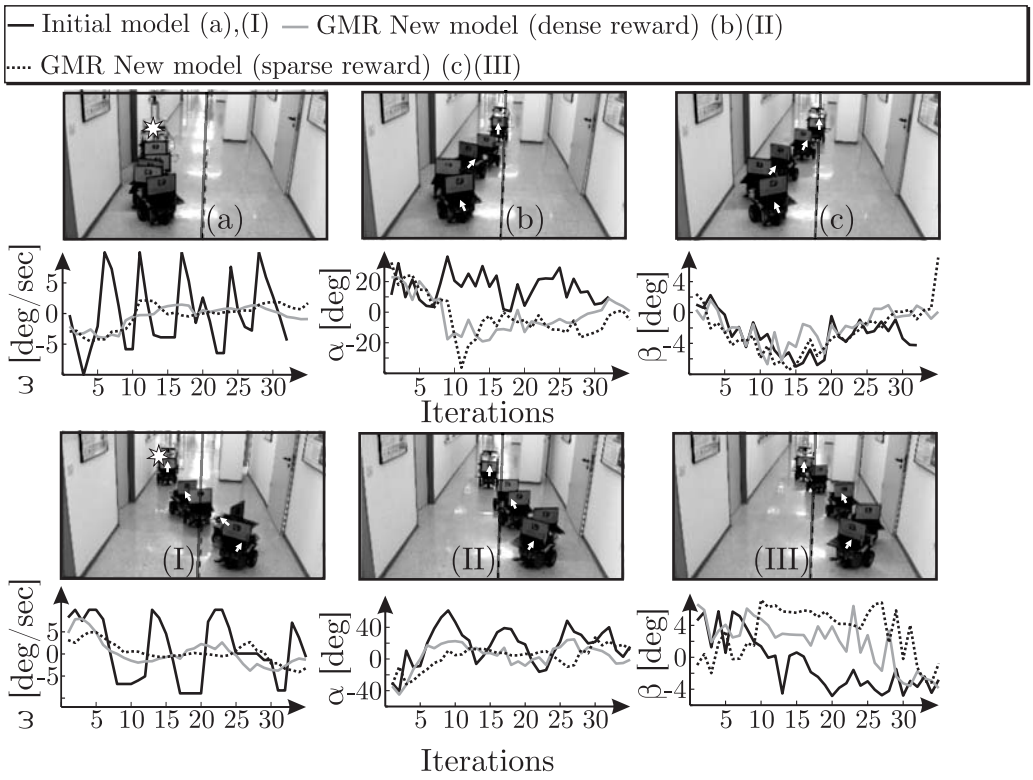


Figure 8.22.: One shot learning. (a),(I) Reproduction of the initial model starting from $[\omega, \alpha, \beta]$ =[0 deg/s, 45 deg, 5 deg] and [0 deg/s, -45 deg, -5 deg], (b),(II) Reproduction of new GMM trained with dense reward, and (c), (III) reproduction of new GMM trained with sparse reward.

combinations. The alternative is the sparse reward where only the goal locations are awarded with a reward of 0 and rest of the state space is penalized with -1. Working with a sparse reward function with no prior knowledge of the task is extremely challenging task. The above works mentioned counter this problem using the LfD framework to gather knowledge through examples. In this chapter, we countered this problem by generating a dummy model with no behavioral knowledge and use this template model to explore and exploit the reward function to learn the behavior essence.

8.6. Summary

In this chapter, we have presented a reinforcement learning framework for learning corridor following behavior dynamics. Prior knowledge about the behavior is provided via GMM learned from teacher demonstrations. Demonstrations are performed on the real robot controlled with a joystick. Two modes of reward function $\mathrm{R}(s, a)$ are designed. Dense reward functions gives more information and are manually tuned. Sparse reward functions corresponds to a reward only at the goal point, which for corridor following is around the corridor center. A reward of 0 is given at states closer and at corridor center and a penalty of -1 is given to other regions. The Q-value $Q(s, a)$ of the demonstrated policy is learned using Value-iteration algorithm with the GMM updated at every iteration. The value function of the policy is evaluated on fourteen different starting configurations in simulation from which the successful trajectories are used for model update. The framework is first evaluated with a corridor following policy modeled on four demonstrations. The

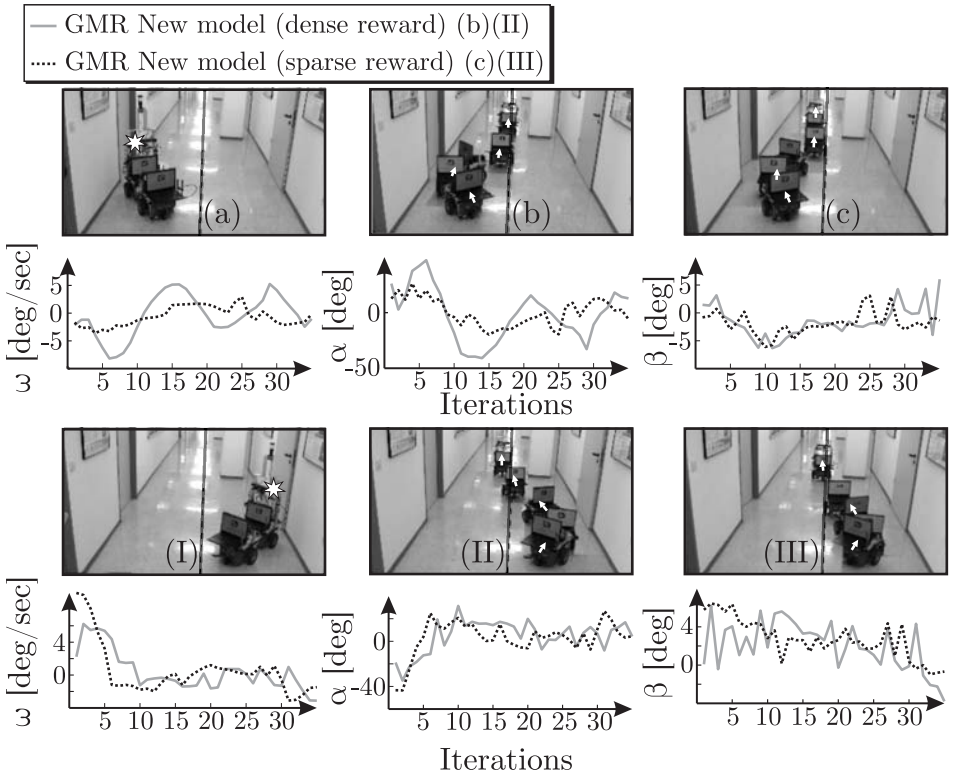


Figure 8.23.: Learning from scratch. (a),(I) Reproduction of the initial model starting from $[\omega, \alpha, \beta]$ =[0 deg/s, 45 deg, 5 deg] and [0 deg/s,-45 deg,-5 deg], (b),(II) Reproduction of new GMM trained with dense reward, and (c), (III) reproduction of new GMM trained with sparse reward.

model emerging from final iteration of value iteration achieved very good generalization and presents no instabilities and traces a smooth trajectory. The generalization capability of the model is further established successfully by testing it in two unseen simulation environments where the model always converges the robot to the corridor center fast and smooth. More challenging tasks such as one shot learning and learning from scratch are also successfully learned and their results show that the robot starting with no information about the behavior is able to successfully learn the inherent value function using a sparse reward. The proposed framework was finally tested successfully on the real robot in a dynamic corridor environment.

From this chapter, it is clear that reinforcement learning for mobile robot navigation is not necessarily a straight forward autonomous task. Many design decisions such as the structure of reward functions, appropriate feature representation of the behavior and the number of demonstrations needed to build prior information need to be determined. The design of reward functions is the fulcrum around which the complete performance of value-iteration algorithm depends. For higher level behaviors such as obstacle avoidance, homing and door traversing, it is imperative to arrive at a reasonable reward function. This in turn requires identifying the right feature representation of the behavior. A promising alternative is the inverse reinforcement learning algorithm [AN04]. Also known as inverse optimal control, Inverse Reinforcement Learning (IRL) identifies the hidden reward functions associated directly from expert demonstrations and is further used to derive the policy. One inspiring research along this direction is the work of [HVFF10], who use IRL to learn navigation rewards to observe human motions in a crowded environment.

9
Conclusions

This thesis contributed a framework for learning vision based robotic behaviors from demonstration examples. It tackles several core issues of using LfD for robotic behavior such as flexibility of demonstration modes, the appropriate behavioral representation for learning and the potential of self-learning behaviors. The main contributions of this thesis are the following,

1. **Natural user interface demonstrations show promising results for demonstrating a mobile robot:** Comparative analysis of three modes of demonstration are designed and tested for demonstrating motion patterns to a mobile robot. Joystick represented the haptic mode which had a direct mapping between the user and the robot. A remote teleoperation with a steering wheel is considered for a demonstration mode performed directly with the perception of the robot. It provided a continuous stream of omnidirectional camera images mounted on the robot and the user uses the image to guide the robot through the monitor. Natural user interface is considered as the third mode where body gestures are used to command robot motions. User studies are performed with participants with diverse backgrounds and the results showed that using natural user interface for demonstrating a robot allowed even a non-expert user to generate smoother and intuitive trajectories. Joystick being the direct command tool is the fastest to generate demonstrations but the haptic nature of the command mode has a relatively higher signal to noise ratio. Typical difficulty arises from remote demonstrations since they do not provide any depth information. The loss of perspective view also accentuated the difficulty, making it a highly challenging interface. Nevertheless, the ability to remotely demonstrate a robot new capabilities definitely is of high promise for future research. The study showed that given adequate time, the performance of the demonstration and correspondingly the fidelity of the recorded data improved significantly.

2. **Visual feature extraction and selection:** After segmenting the omnidirectional image into floor and non floor regions, several manually extracted visual features appropriate for robotic behaviors that reflect the geometry of the environment and the proximity of obstacles around the robot are extracted. In addition to the manual features, the principal components of the segmented image in different indoor environments are also found. The selection of the most relevant features for the behavior to be learned are determined using a wrapper approach of forward chaining. Feature selection was a recurrent topic in this thesis where forward chaining was

used for multiple behavior learning tasks. Selection of an appropriate feature set is always performed through cross-validation on unseen demonstration examples.

3. **Learning behaviors that are specific to a scenario and context is an effective approach to model situatedness:** Robotic behaviors are highly situated. Their performance and action are directly dictated by the characteristics of the environment. A new two tier framework is proposed here, where in the first level the current indoor scenario is first classified and in the second level the context within the scenario is matched and the next action predicted. Visual features pertaining to individual scenarios that exhibit robust geometric and scenario distinction are selected and trained on a variety of different classifiers. Three scenarios, namely corridor, open room and cluttered are considered for classification. The accuracy of the classification scheme is exhaustively validated against unseen data achieving sometime more than 98% accuracy. The successful models are subject to rigorous tests in different indoor environments where their classification results are further analyzed. Contexts within a scenario correspond to the current situation of the robot using the recent perceptual memory trace of the behavior variables. The perceptual trace is compared and matched for similarity using time series trajectory matching techniques. Alternatively, artificial neural networks which are used to model entire scenarios are also presented. The experimental results of the complete framework show that both the schemes perform good in unseen environments with context based model little sensitive to outliers. ANN based scenario model on the other hand exhibits robust performance to noise. By modeling situatedness through scenario and context, the behavior coordination problem is inherently integrated into the learning framework.

4. **Learning neural and dynamic representations of a behavior:** Instead of modularizing behavior fusion in terms of the structure of the environment, learning individual behaviors using demonstration examples are presented. Three behaviors, namely corridor following, obstacle avoidance and homing are used consistently for different schemes. A neural network representation and a dynamic system representation of the behaviors are considered. Two core issues of behavior learning are discussed, namely *how to define a good learned behavior policy?* and *how many demonstration suffice to generate a good behavior policy?* With neural approaches, we find that the quality of the demonstration play a huge role in policy building than the quantity of demonstrations. The performance of every learned model is tested in simulation first on ad novel scenarios. Experiments show that learned with only one or two examples, which capture the core of the behavior, suffice to generate a model that can traverse complex environments. Naturally, this arises the question: how does the teacher determine which demonstrations are the best. One rule of thumb here is to generate examples that are diverse and quite possibly use demonstrations that start from extremities of the behavioral state space. Both static and dynamic neural networks are trained and tested on the real robot. Apart from the behavioral level distinction, feature level difference with the learned behaviors are also presented. Both, manually extracted visual features and the principal components are used to train the neural network.

   Compared to neural network representation, the thesis also shows that a coordi-

nated behavior based system can be learned by representing individual behaviors as attractor dynamical systems. The behavior variables are selected on the image space of the robots perception. Attractors that represent stable behavior solutions are modeled using GMM and their dynamics learned. The results show a robust performance of the learned dynamics to deal with unseen environments. Compared to neural approaches, GMM needs relatively more demonstrations to shake off the biased nature of reproductions. By using conventional coordination scheme, three behaviors are evaluated on the real robot to execute challenging pure vision based wandering.

5. **Monolithic behavior fusion models are impractical:** Learning a monolithic behavior fusion model across all the scenarios is a highly difficult learning problem. The diversity of the training data in such cases need to be large and the learning algorithm should be capable of handling such diversity. Neural network and nearest neighbor approaches are tested to learn behavior fusion. Experiments show that the high model complexity necessitates a large number of features thus leading to dimensionality problem. The lack of modularization of the policy is also reflected in experimental results where the learned behavior was sensitive to noise and perturbations.

6. **Learning with a critic exhibits promising future:** A new framework for self-learning a robotic behavior using RL is presented. As a proof of concept, visual corridor following behavior is learned. A prior knowledge model, the behavior is first learned with GMM whose parameters are iteratively optimized and thereby the policy improved. The successful performance of the framework to learn behaviors with both a dense and sparse reward function is presented. The robustness of the framework is tested further with one-shot learning and learning from scratch. Provided with just one demonstration example and with a sparse reward function, the framework exhibited the capability of recovering the optimal value function of a behavior. Reinforcement learning with scratch presents a problem where no reasonable demonstration knowledge is provided. Working with only a sparse reward function, the RL framework is able to recover the value function and generate an optimal corridor following policy. The final policies are tested on the real environment and their successful evaluation were documented.

9.1. Outlook

The work presented in this dissertation provided an insight into the potential of learning within a behavior based framework. The behavioral performance of the robot working on purely vision based input has a direct correlation to the quality of the features provided. As discussed throughout the thesis, the selection of the relevant features for the appropriate behaviors is a time consuming and sometimes frustrating process. Evolutionary feature selection and deep architectures offer a lot of promise in this area. Evolutionary robotics is population based artificial evolution algorithms [Hol92], where the autonomous robot controllers evolve to acquire some kind of intelligent decision making. With the handful of image features, evolutionary robotics selects a random subset of candidate controllers

as the initial population. Every individual controller can be any of the different behavior representations discussed such as ANN or RNN or GMM. During every generation, the evolutionary algorithms is supposed to design the vision based system by extracting the relevant features for the behavior controllers. The advantage of such a system is that the algorithm can produce controllers which might be capable of handling and functioning in environments that even humans might find it complex. The drawback of evolutionary algorithms is the computational complexity involved in repeating and evaluating the fitness of every individual controller in the population for multiple generations till the optimal controller is found. Deep learning architectures here provide an alternative methodology which models a higher level abstraction of data using multiple non-linear transformations. A deep learning architecture for a neural network would here involve training multiple hidden layers of neurons achieve complex non-linear relationship between input and output. Using a combination of unsupervised and supervised learning, the weights of the hidden layers are trained to find patterns within the input in a fast and effective manner. Learning behavioral representation using deep neural networks can be done by first presenting all the demonstration example images to the network to assign the weights and thereby bias the neurons to the data in an unsupervised manner. With these weights, the network is then further trained with the corresponding label in a supervised fashion.
Another promising area is in the field of robot proxemics. It deals with the design and implementation of robot behaviors that not only makes the robot move collision free but also execute a legitimate and socially acceptable trajectory. By observing the patterns exhibited by humans dealing complex indoor environments, the inherent behavioral coordination can be learned. Typical coordination tasks here would be the observation of humans performing lateral passing within a narrow environment and the astuteness shown in handling intersection scenarios. Typical extension here would be also to learn human-machine interactive formations and response.
Outside of robotics, these principles are equally applicable to design and development of autonomous driving cars. Autonomous cars or driver less cars are pushing the capabilities of a traditional car to sense the environment and make navigation decisions without any human input. The pioneering goal of autonomous vehicles is to develop a system that exhibits the capabilities of a human driver. Extending LfD to autonomous driving can provide very useful insight into human driving characteristics thereby augmenting important driving controls such as lane keeping and navigation in safety critical situations.

A

Questionnaire on demonstration modes

Please use this questionnaire to provide feedback about the demonstration modes and your experiments with it.
Enter 1 - closest, 2 - somewhat close, 3 - very far.

Q.No	Question	Joystick	Gesture based	Steering wheel
1	Which one of the mode did you find it the easiest to demonstrate to the robot?			
2	Which one is the most fun to operate and demonstrate?			
3	Which one required a longer preparation time?			

Additional comments

B

Software architecture

The software architecture of the navigation framework is designed in a hierarchical fashion as shown in Fig. B.1. The bottom layer represents the driver level which represents the different sensors used. This layer is concerned with the acquisition of sensor data, synchronization of multiple camera systems which are then passed to the pre-processing interface. The omnidirectional camera uses IEEE 1394 bus standard to provide the raw images at the rate of 15 frames per second. The monocular, PMD camera and laser have an USB interface and have a frame-rate of 30 frames per second. Sonar data are acquired either through a serial or an USB interface. The data from the sensors are processed in the platform layer which contains implementation based on the configuration of the camera and the robot. The raw sensor data is thus pre-processed and down-sampled which is then used for the segmentation routine. The segmented image is thus fed used as input to extract meaningful image based features. Thus the raw image from the sensor is converted to visual features upon which higher level algorithms are executed. The algorithmic layer consists of all the higher level decision and control algorithms for the robotic system. This consists of the different behavior learning algorithms which are further used for behavior execution at runtime. This layer also consists of the localization module which tracks the robot location in the map. The localization module is used only for validation and visualization of the behavior performance and does not share any information with the behavior or other modules in the system. The user interface layer provides all the interaction between the robot and the teacher. This includes interface hardware such as joystick, Microsoft Kinect and an external PC for remote teleoperation.

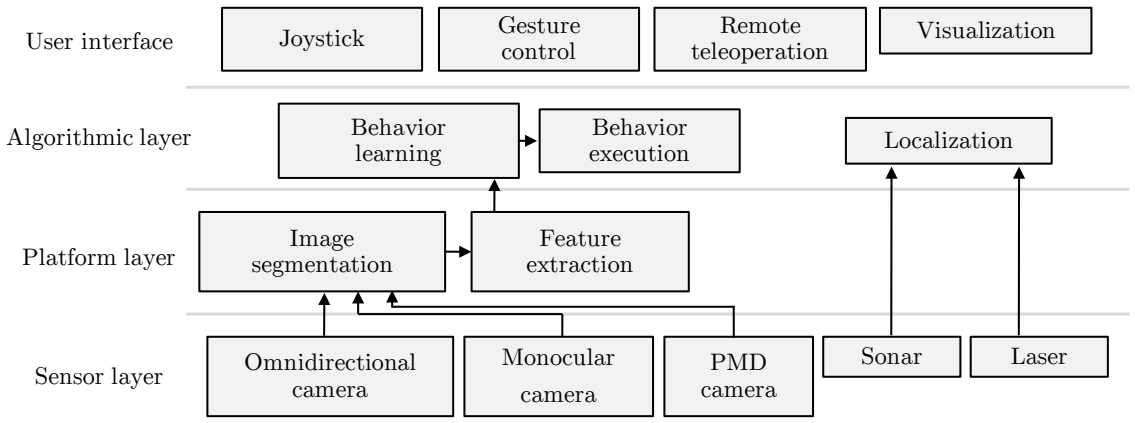


Figure B.1.: Software architecture used for the robotic navigation framework.

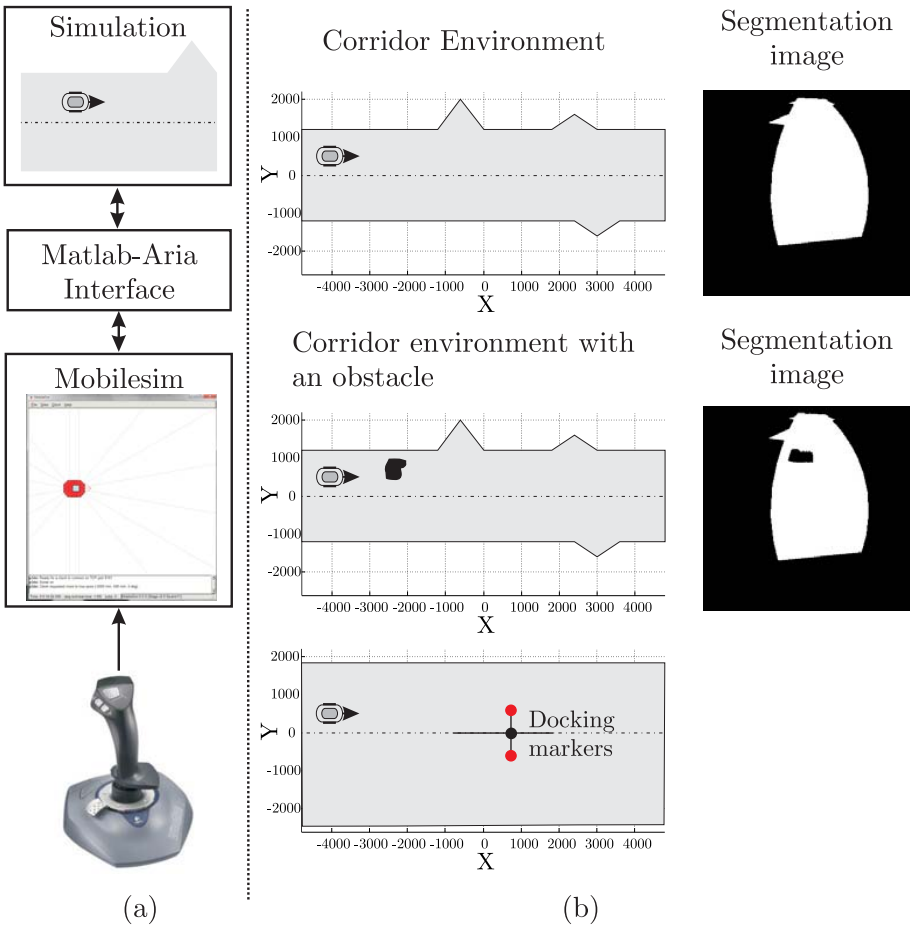


Figure B.2.: (a) Overview of the simulation architecture and (b) robot in a corridor environment with and without obstacle together with its corresponding segmentation image and robot with homing area in the vicinity indicated by two red markers 1m apart.

The user interface is predominantly used during the demonstration phase and is decoupled during the autonomous navigation phase.

B.0.1. Robot simulation

The simulation environment is created with MATLAB and is used to visualize and build personalized environment configurations with multiple obstacles. The kinematics of Pioneer 3DX mobile robot is determined by directly using its Proprietary simulation software MobileSim distributed with the development kit ARIA [Mob]. ARIA is a C++ library to dynamically control the robot and access the integrated proximity sensors. Matlab-Aria interface developed by [Pos], uses MEX binaries to access the robot functionality via MATLAB. The simulation starts with the user positioning the robot within the designed environment, along with MobileSim. The velocity or motion commands from Matlab are first sent to Mobilesim. The updated new position from Mobilesim is then read and is correspondingly used to set the next robot position in the simulation. The simulation is visualized with a birds eye view of the robot and its environment to the user. A segmented omnidirectional image of the virtual world is provided by projecting the floor and the obstacle points from the map onto camera coordinates using the omnidirectional camera model. Calibration and camera parameter estimation is done using the calibration tool-

box of [SMS06]. The simulation facilitates changes to the geometry of the environment and addition and reconfiguration of obstacles. The reprojected segmented image provides the basis on which the robot local environment is segmented into floor and obstacles. The motion of the simulated robot is based on the kinematic model of a Pioneer 3DX mobile robot equipped with an omnidirectional camera with a 75° field of view directed towards the floor. Figure B.2 shows overview of the simulation architecture along with the different environmental configurations and manual design of obstacles within these configurations.

C

Scenario classification

C.1. Scenario classification with probabilistic neural networks

In the hidden layer; centered on every training instance, a Gaussian is used to determine the influence of the neighboring points on the determination of the class in the pattern/summation layer. Presented with an input, the distances of the input to the training inputs are computed in the first layer and applies a radial basis function kernel using the spread or the standard deviation. For scenario classification using PNN, the optimal value of the Gaussian standard deviation is done by varying it from 0.1 to 1. For every spread values, a 10-fold cross-validation is performed and one with the lowest test misclassification error is identified as the optimal value. Fig. C.1 shows the cross-validation error recorded for different values of standard deviation.

C.2. Extended experimental validation

Section 5.1.3 presented the classification performance of the different scenario classification models in an indoor corridor environment. The extended experimental validation of the scenario classifiers in other diverse indoor environments are presented here. Fig. C.3 shows the results of scenario classified while wandering an open foyer. The central foyer is rightly classified as open room by PNN, RG and RF, with the connecting pathways as corridors. At every pathways and its dead ends, the scenario classification follows the

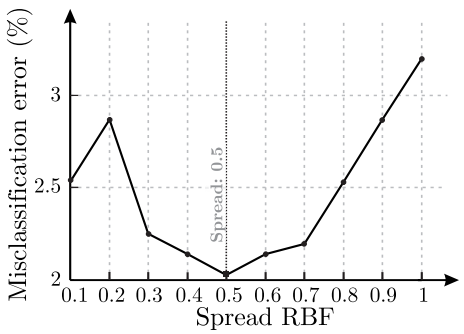


Figure C.1.: 10-fold cross-validation error on different values of spread (Gaussian standard deviation) with a PNN.

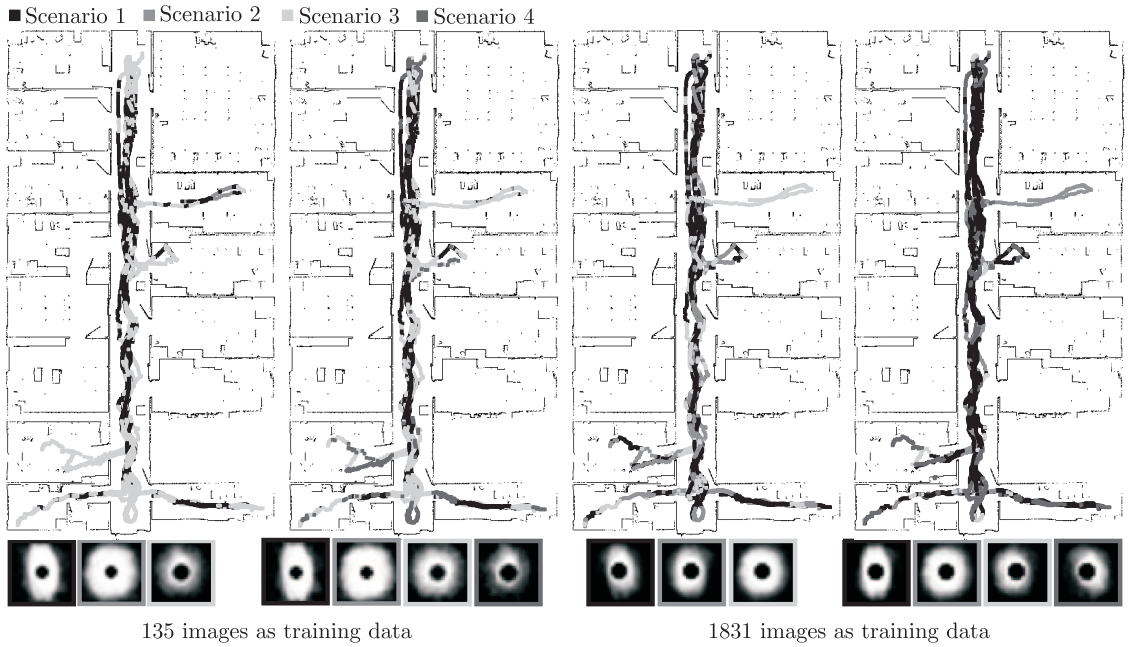


Figure C.2.: Path traced by the robot wandering an office corridor environment and the corresponding scenario classified by SOM.

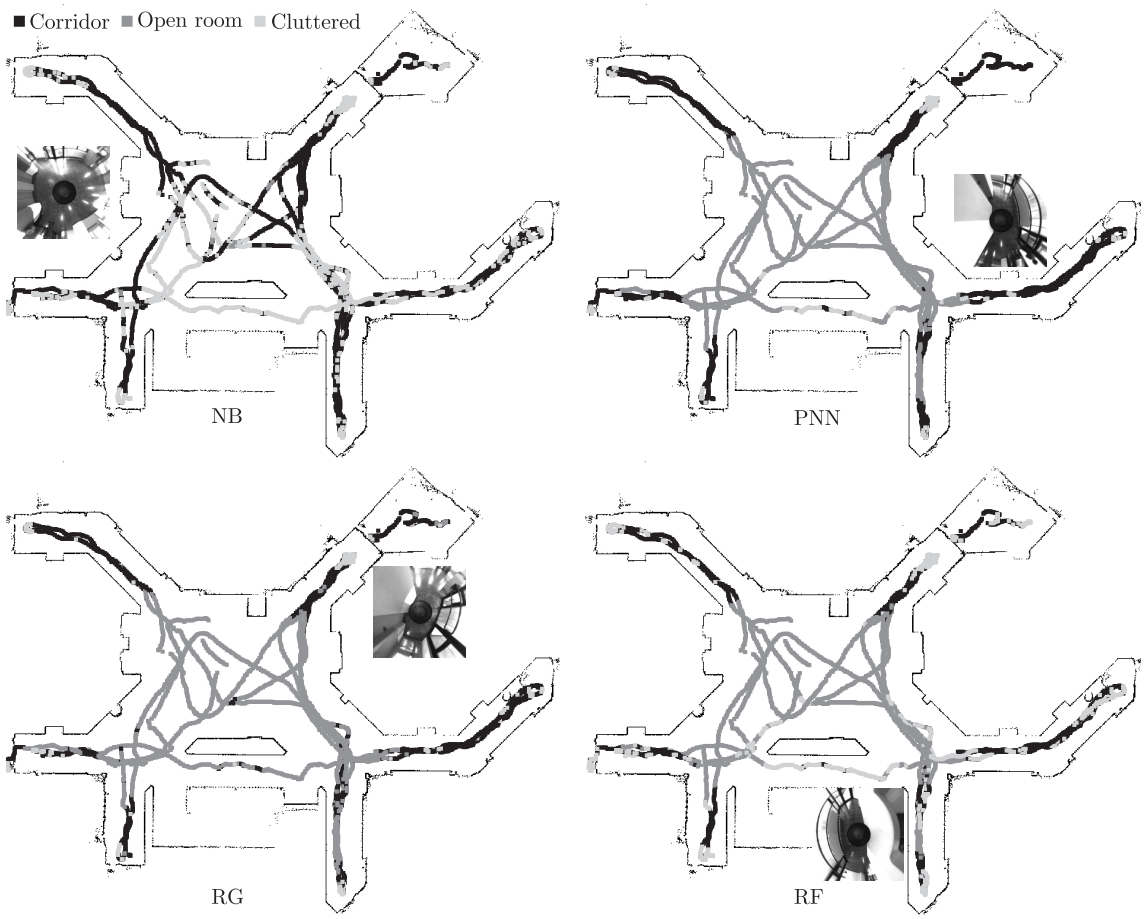


Figure C.3.: Path traced by the robot wandering an open space foyer environment and the corresponding scenario classified by the different algorithms trained on visual features.

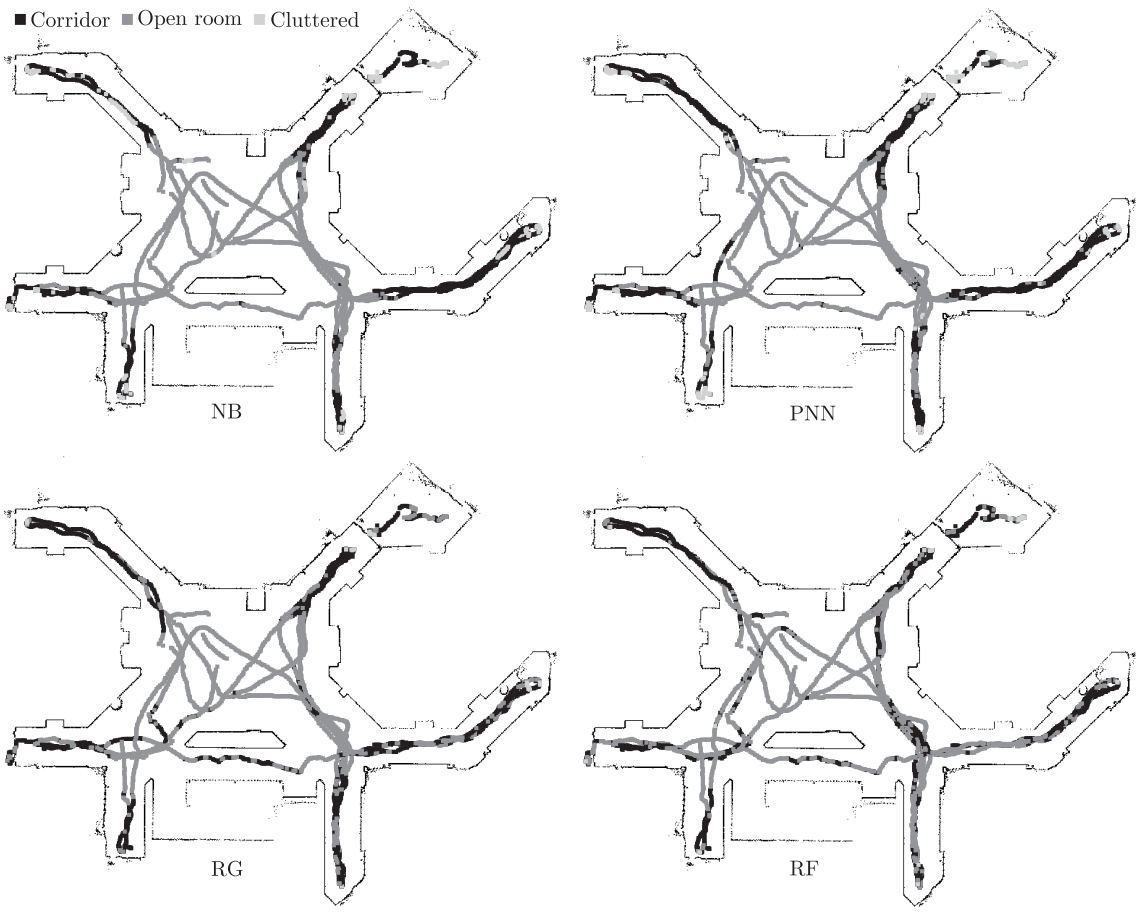


Figure C.4.: Path traced by the robot wandering the an open space foyer environment and the corresponding scenario classified by the different algorithms trained on principal components.

same pattern as with the corridor environment, thus exhibiting consistency across diverse indoor environment. The experiment also shows the robustness of the classifiers against different texture and lighting conditions as foyer in contrast to the corridor environment is illuminated with sunlight. Similar to the performance in the corridor, PNN and RF show the best performance followed by RG and NB. Fig. C.4 shows the scenario classified by the second group of classifiers trained with their principal components. In an open environment, where the variations in geometry are not frequent, PCA based scenario classification outperforms the visual features based classifiers. In contrast to the performance in corridor environment, a foyer environment does not present scenarios like going through a door way where the orientation of the robot plays a role in scenario identification. All the four classifiers are quite consistent with principal components in a foyer environment. Scenario classification with SOM exhibits good performance with both three and four premonitions. Clustering with four classes is more evident here, where entering and exiting the path way are considered as different scenarios. From the image one can see that by training on all the available demonstration data, the classification accuracy is more consistent and robust than its other counterpart. A final run is evaluated in a relatively smaller foyer environment. The environment is less illuminated and has presents a different texture and appearance to the other two. The experimental results with visual features, principal components and SOM are shown in Fig. C.6 and C.7. The performance of the classifiers follow the same trend as in other scenarios with

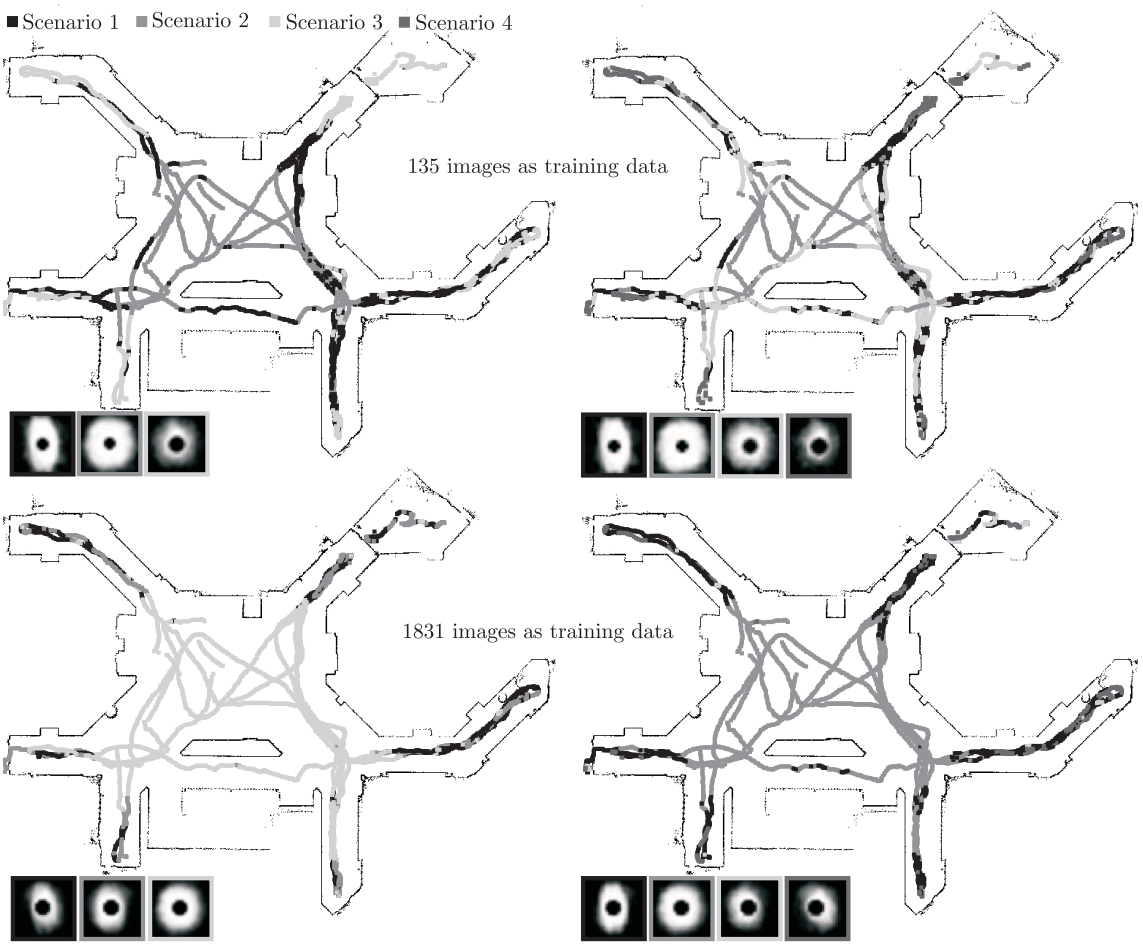


Figure C.5.: Path traced by the robot wandering the an open space foyer environment and the corresponding scenario classified by SOM.

PNN and RF at the top. For completeness, Fig. C.7 shows the classification with three and four classes identified from the data by SOM. Interesting to note is that with four class SOM trained on 135 images, the foyer is predominantly classified as scenario three and four and not scenario two which resembles the open room scenario. Compared with the classification results of a larger foyer, we can notice that SOM is able to distinguish the difference in the lack of geometry against a standard open room scenario. Evaluation results with classes 3 and 4 for SOM trained with all the data correspond well to the scenario clustering.

C.2.1. Extended experimental results

This section presents more extended experimental results carried out using the scenario and context specific models [NPHB11a]. Fig. C.8(I) shows an instance of a prototypical obstacle avoidance scenario in the corridor. Two schemes are shown. In both the cases, the first level scenario classification is performed by RG whereas the action selection is first tested with a scenario specific ANN and then with LCSS based context matching. Starting from an identical initial pose, both schemes successfully circumnavigate the obstacle with minor variations in the exact trajectory. Executing with ANN, the robot turns sharply right at the beginning and overshoots the center line because of the detected free

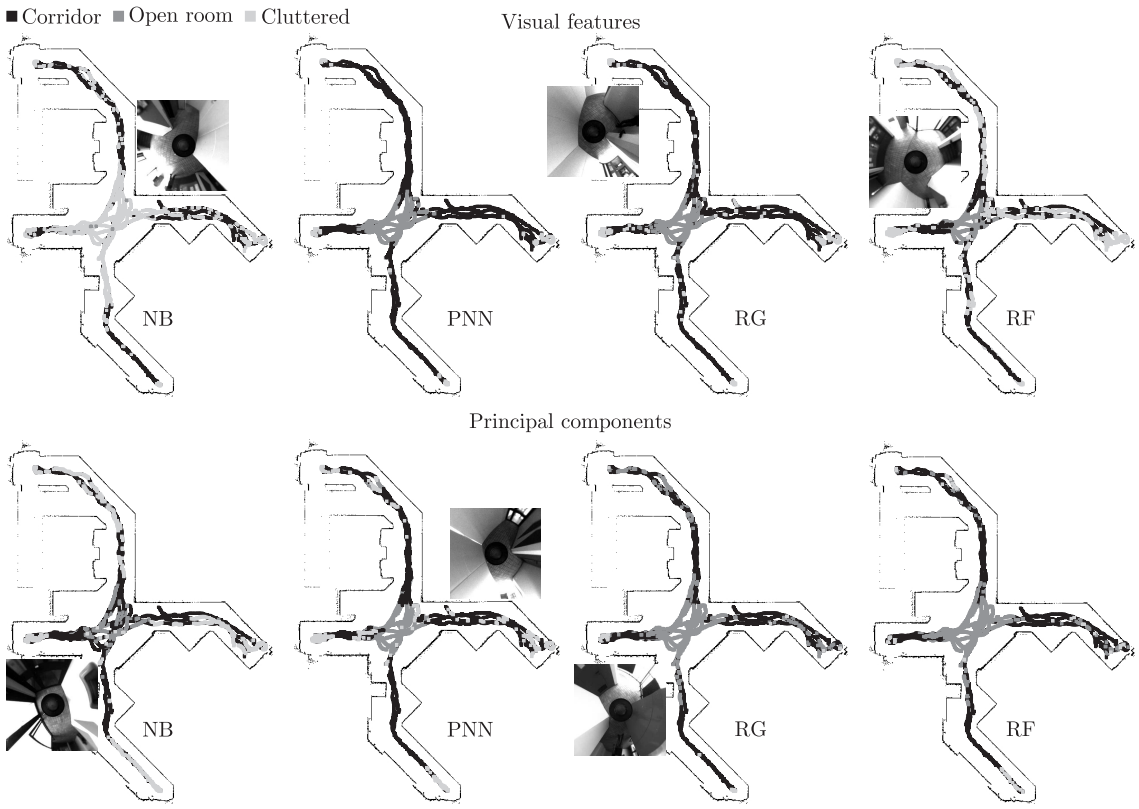


Figure C.6.: Path traced by the robot wandering a narrower foyer environment and the corresponding scenarios classified based on visual features and principal components.

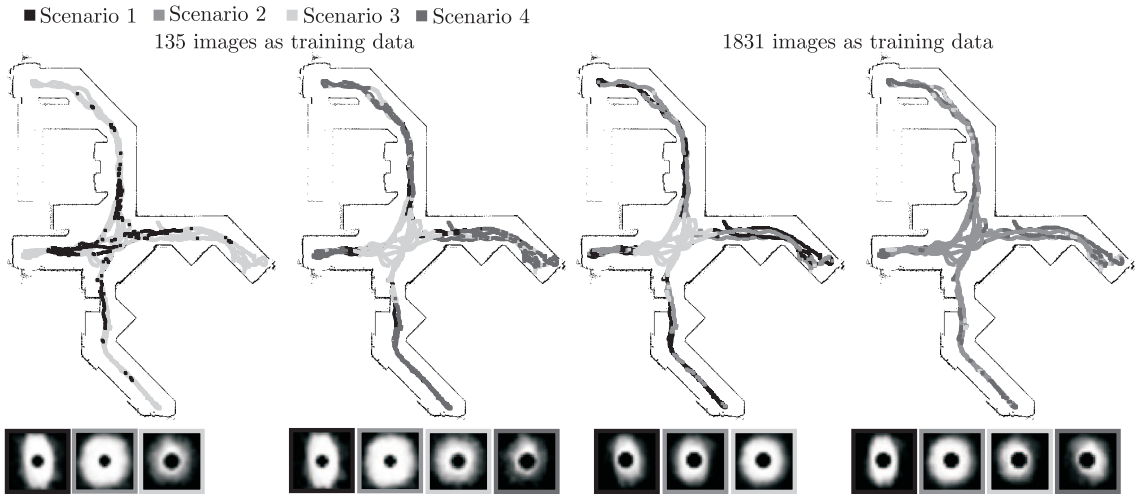


Figure C.7.: Path traced by the robot wandering a narrower foyer environment and the corresponding scenarios clustered by SOM.

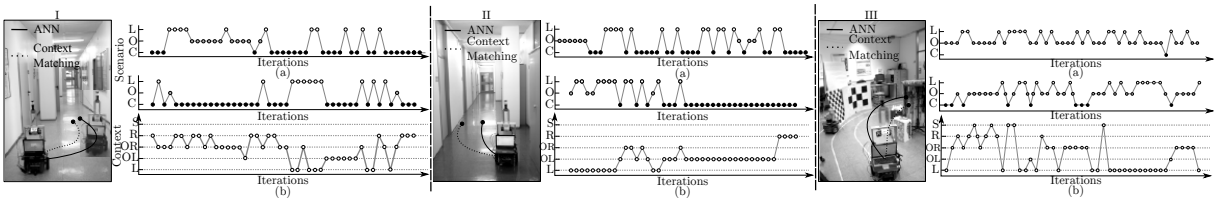


Figure C.8.: (I) Scenario 1: Known corridor. (a) Artificial neural networks and (b) context matching and prediction, (II) Scenario 2: Unknown corridor. (a) Artificial neural networks and (b) context matching and prediction, (III) Scenario 4: Cluttered environment. (a) Artificial neural networks and (b) context matching and prediction

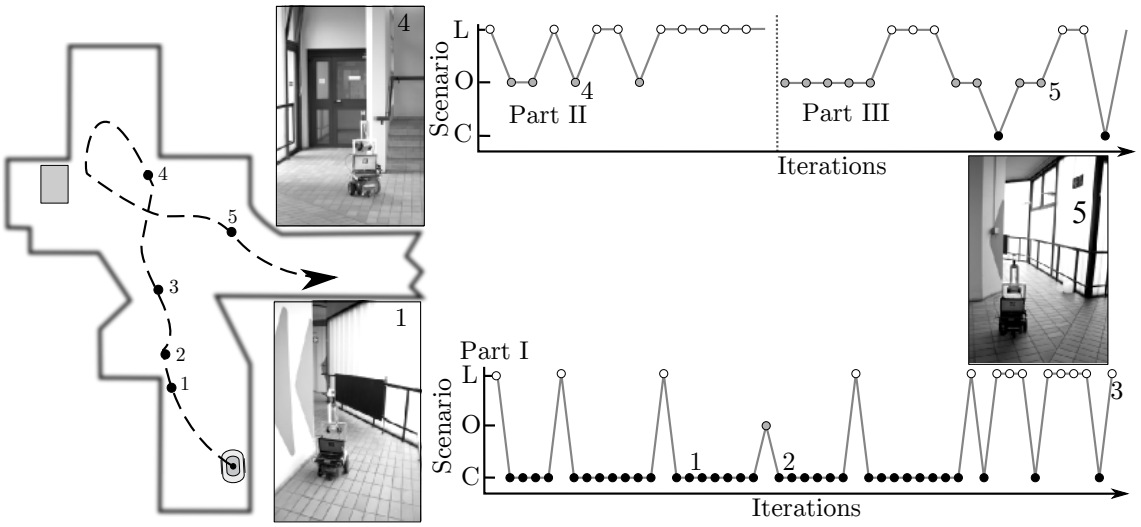


Figure C.9.: Scenario 3: Unknown foyer scenario with artificial neural networks

space on the right hand side courtesy of the open door. This is directly correlated to the initial incorrect and misleading scenario classification as open room or cluttered environment. Nevertheless the associated trained behaviors in spite of the proposed suboptimal actions still avoid collisions. With trajectory matching and prediction, the robot traverses relatively smoothly in avoiding the obstacle with a consistently correct classified scenario and thus correct selection of the best matching subsequence of reference trajectories. Similar performance is seen when testing it in an unseen corridor where no training examples exist (Fig. C.8(II)). From the figure it is apparent that both methods are able to generalize successfully to the new corridor but exhibit a higher scenario misclassification rate. This misclassification is due to the similarity in color and intensity between the floor and the obstacles resulting in less accurate free space segmentation. Nevertheless the scheme is robust enough to avoid the obstacle. Fig. C.9 shows one such successful traversal in a foyer environment not presented during training under control of the ANN. The foyer is insofar challenging for action prediction as it contains a mixture of corridor, open room and cluttered geometries. From the figure it is apparent that the starting pose is classified as a corridor (Part I) and the associated acquired behavior guides the robot towards the corridor center. Upon reaching the center of the foyer (position 3), the classification switches between cluttered and open room environment (Part II). On encountering a corner between position 4 and 5, an obstacle avoidance behavior dominates that returns the robot to the foyer center.

To further ascertain the performance of the learned behaviors, two more experimental re-

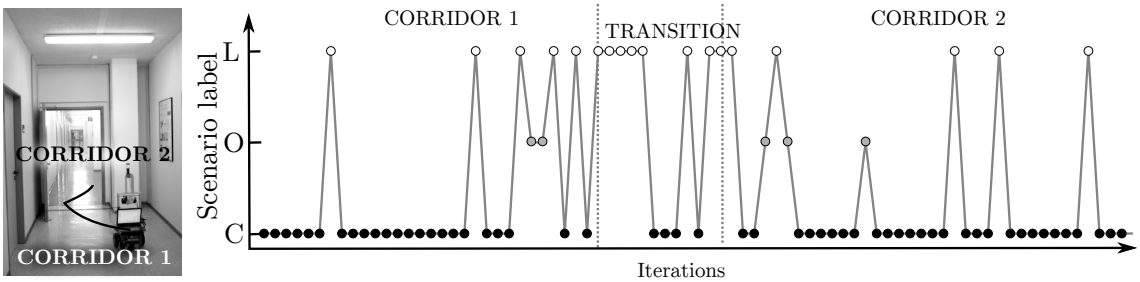


Figure C.10.: Scenario 5: Crossing corridors with artificial neural networks

sults are analyzed. Fig. C.8(III) shows the responses of the ANN and trajectory matching scheme in an environment cluttered with boards and obstacles. Both methods successfully circumnavigate the obstacles, nevertheless with trajectory matching the robot passes the obstacles at close range whereas the ANN based behavior maintains a safer distance. The stability of the ANN based model is further tested by traversing from one corridor to another corridor. Generally because of the abrupt changes in texture and appearance among the corridor, the segmentation first deteriorates in the transition zone as it requires an update of the internal floor model for segmentation. Once the model histogram is adapted to the new texture and color the segmentation error drops. Fig. C.10 shows one such traversal between two corridors.

D

Supervised learning of visual behaviors

D.1. Artificial neural networks

Corridor Following

Chapter 6 presented a detailed analysis of modeling corridor following using ANN. Of the twenty available corridor following demonstrations, different subsets of trajectories are used to generate a policy. To illustrate the procedure used to train and test a corridor following, we consider the situation where only two trajectories; one from either side of the corridor are available (Fig. D.1). The demonstration examples only partially cover the perceptual state space and thus permits a detailed testing on unseen data. ANN is trained with different number of hidden neurons ranging from 1 to 9. Every model trained is further tested against the remaining eighteen unseen trajectories. Both the training and test MSE and NMSE are shown in Fig. D.2. The plot shows that training errors incrementally improve with more neurons. The low training NMSE suggests that the model has clearly overfitted the training data. This can be seen from the poorer testing errors. With five neurons, the ANN achieves the minimum MSE and second lowest NMSE on the test data. Nevertheless, a NMSE of 0.85 and 0.89 are still high which indicates only a moderate reproduction of the demonstrated policy indicated by the $10-15\%$ improvement in the residual variance of the prediction. Fig. D.3 shows the predicted rotational velocity against the actual targets for all the twenty demonstrations.

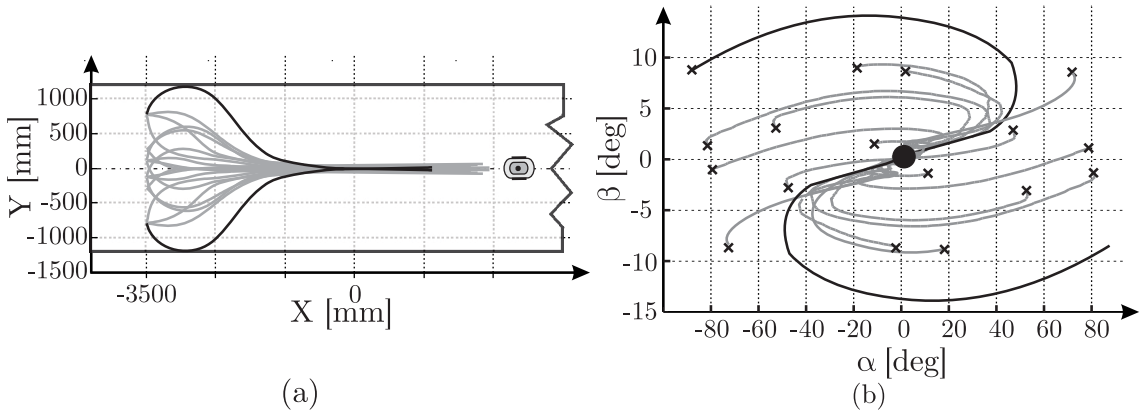


Figure D.1.: (a) Two training trajectory demonstrations recorded from either side of the corridor and (b) the corresponding perceptual trace. Dark lines correspond to the training data.

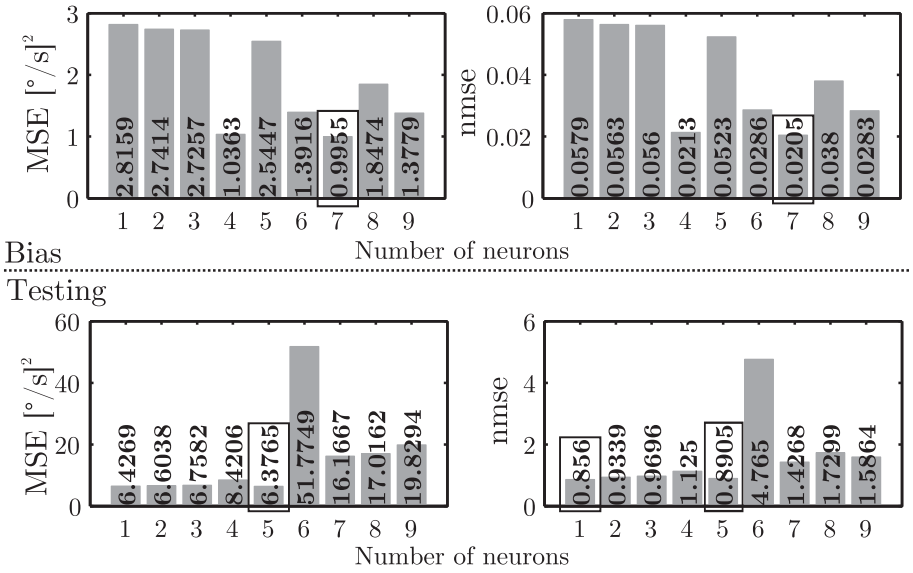


Figure D.2.: Two training trajectories: Training and testing errors recorded for different number of hidden neurons. (a) Training MSE and NMSE and (b) testing MSE and NMSE. Boxed errors correspond to the lowest errors.

For each of the trajectories the corresponding robot trajectory recorded in the simulation are also shown. The plots show that the generalization deteriorates for the trajectories for starting poses very close to the corridor center and with orientation error β. This is mainly attributed to the lack of information in the training data set.
Figure D.4 shows the different sets of training data used. Each training data is assigned a label which is further cross-referenced for the remaining of this chapter. Every set of training data is checked for the appropriate number of neurons starting from 1 till 9. The NMSE recorded on the unseen trajectories is used to gauge the best model. Table D.1 shows the testing errors recorded for the different number of neurons across all the trained models. One can notice from the errors that a low MSE does not mean that the policy reproduction is better. This can arise from the type and the number of trajectory used for testing. Demonstrations starting close to the corridor center need only a small rotational velocity to get to the center whereas the ones starting at the periphery of the corridor need a larger rotational velocity at the beginning to reach the goal faster. These differences can give MSE that can be easily misinterpreted. Hence, we use NMSE to determine the quality of the model against unseen data. Minimum NMSE on the test data set is used to pick the appropriate number of neurons for the corresponding training data set used.
Figure D.5 shows the trajectory traversed by the different ANN models. The starting configuration on which the model is trained is shown with a darker trajectory. The figure shows that models trained with trajectories that provide knowledge about both hard and soft turn to the corridor center perform the best. Models that learn either one of the turn maneuvers perform poorly to the unseen examples. This shows that it is not the number of trajectories that is important rather the quality in the diversity of the training trajectories that define the robustness of the model and thereby the behavior policy.

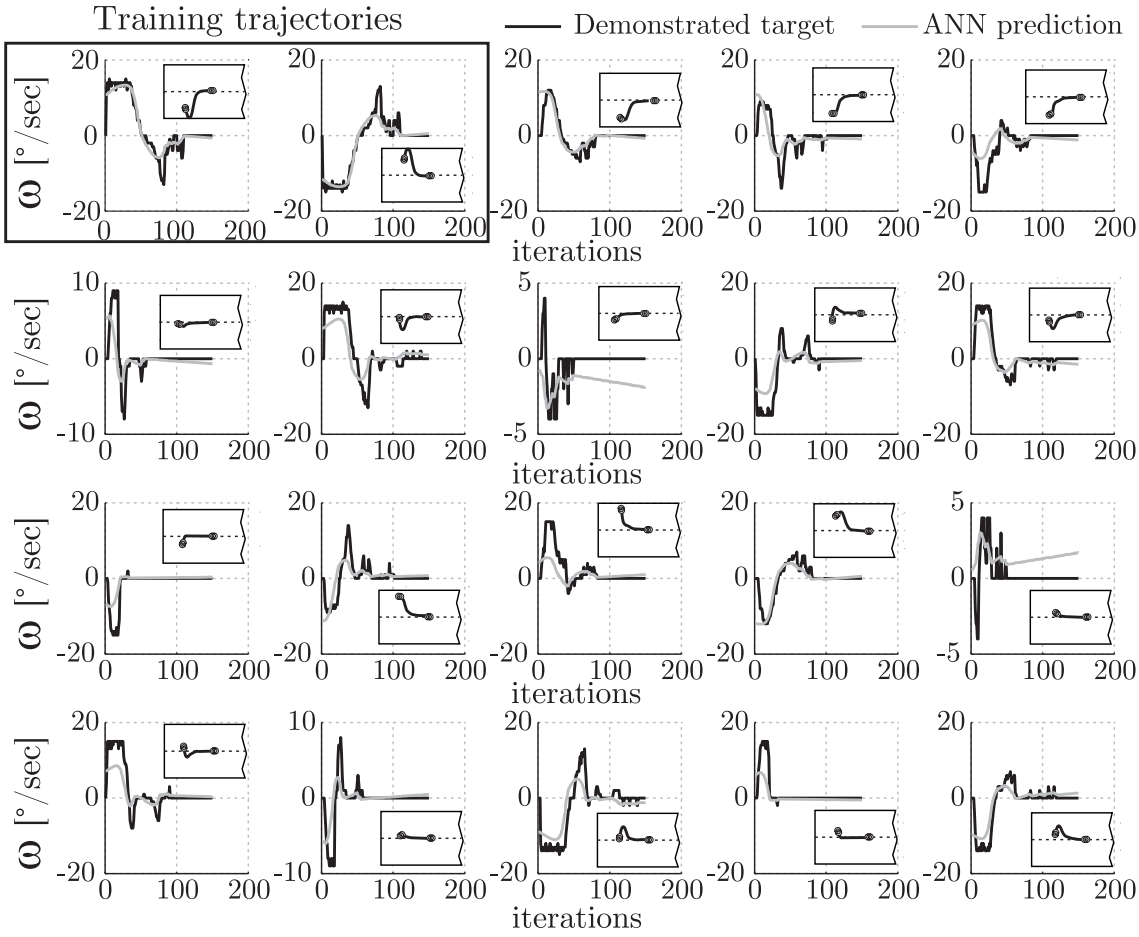


Figure D.3.: ANN prediction for all the corridor following demonstrations. The boxed trajectories are used for training and the remaining for testing generalization.

Obstacle avoidance

Obstacle avoidance is accomplished by controlling both the translational and rotational velocity. The translational velocity is demonstrated by slowing the robot approaching the obstacle. ANN is trained on the data with different set of neurons ranging from one and five to pick the model with the lowest NMSE against the unseen trajectory. The input to the model is the recorded critical distance d_{crit} and the output is the velocity of the robot normalized to its maximum $v_{op} = v/v_{max}$. Table D.2 shows the recorded training and test MSE and NMSE. In addition to the neural network, a linear regression fit of the training data is also analyzed. The table also shows the generalization training and testing errors. A linear second order polynomial is fit to the data to compute the three parameters. The input data is standardized for zero mean first and then fitted. For an input x, the fitted result takes the form: $f(x) = p_1.x^2 + p_2.x + p_3$, where $p_1 = -0.0688, p_2 = 0.3453$ and $p_3 = 0.6291$ are the regression coefficients. The turn rate of the robot is computed for both model predicting the rotational velocity and conversely predicting the path curvature. As in the case of corridor following, different sets of trajectories are used for training. Four left turns (2 soft and 2 hard left turns) and four right turn maneuvers (2 soft and 2 hard right turns) for obstacle avoidance are demonstrated. The different training sets are compiled together for training are,

- I : Two training trajectories: one hard left and one hard right turns,

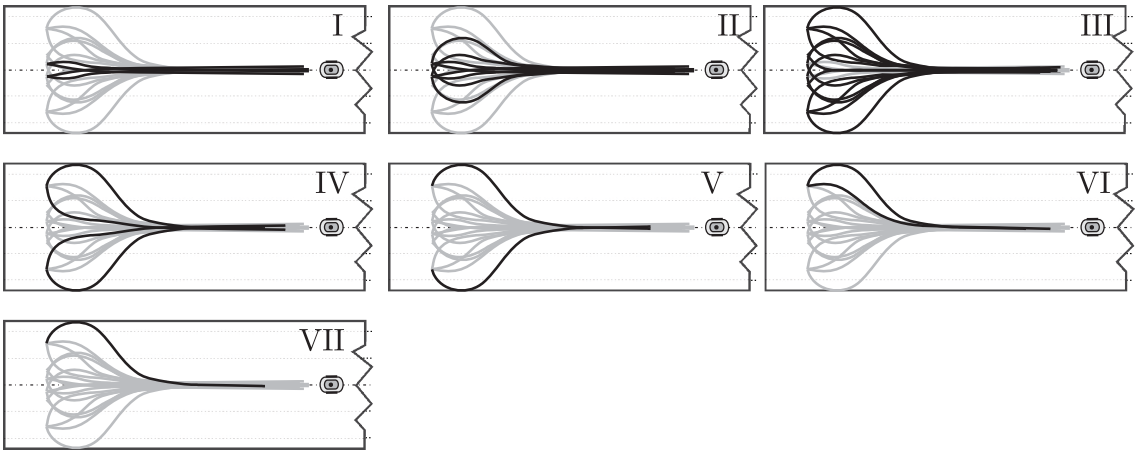


Figure D.4.: Summary of the different demonstration sets used for learning corridor following

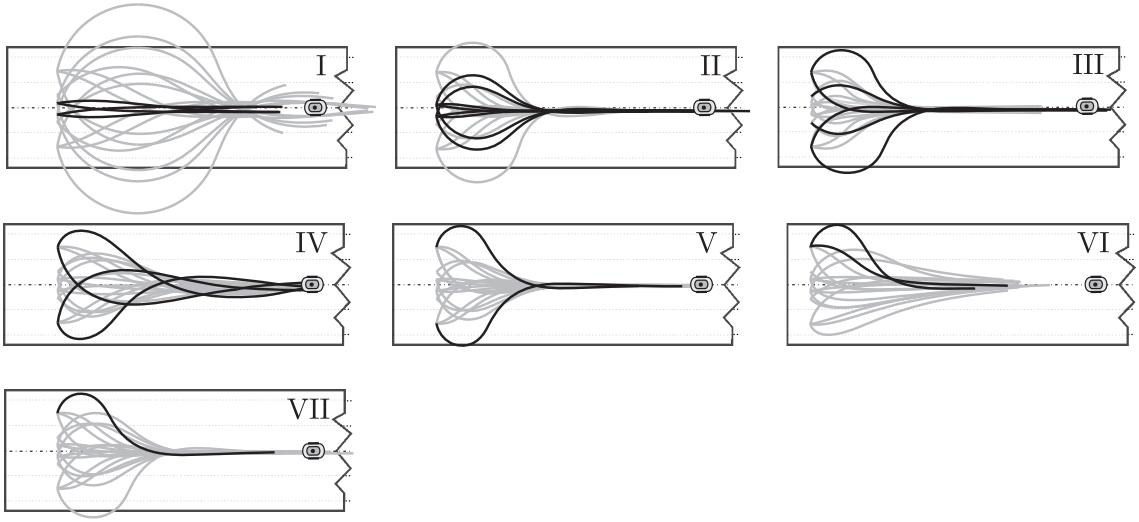


Figure D.5.: Reproductions of all the models. The training configurations are indicated with darker color. The labels correspond to the demonstration subset introduced in Figure D.4

- II: Four training trajectories: all left turns,
- III: Four training trajectories: all right turns and
- IV: Two training trajectories: one soft left and one soft right turn.

The following table D.3 shows the recorded error encountered for the different number of neurons during training. The MSE and NMSE on the training and the test data are correspondingly tabulated. The errors show that with only two trajectories with each an example of a left and right side obstacle avoidance, the model is quite good in generalization. We achieve an improvement in the error residual variance of almost $84 - 85\%$ for data sets I and II. Similar trend is also seen in models predicting path curvature instead of the rotational velocity. For models that are only trained on a single sided turn, behave poorly on the unseen data.

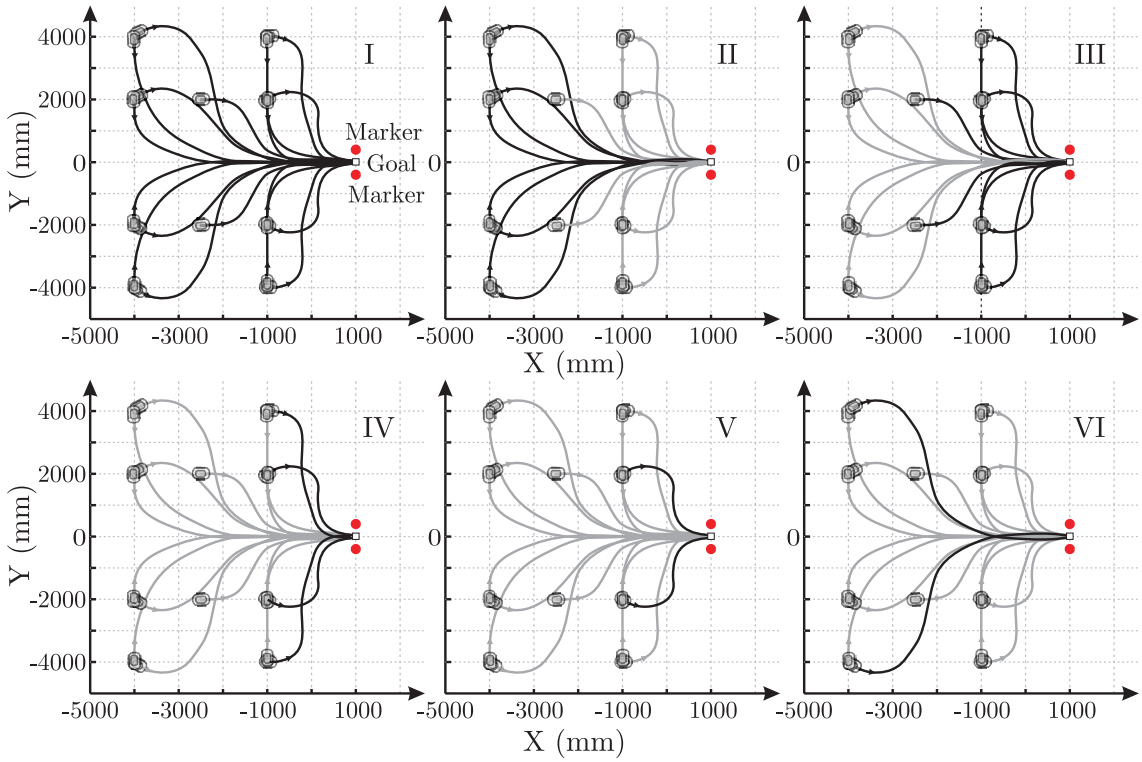


Figure D.6.: Summary of the different demonstration sets used for learning homing

D.2. Homing

The goal of homing behavior is to approach the goal or the target location. In a priori known environment, this would account to searching the state space for an optimal path with minimum cost. In our case, there is no a priori information neither about the environment the robot traverses nor about the location of the goal point. The goal point behavior is activated only on detecting the presence of the goal point markers. Thus the behavioral action to reach the goal point is done purely based on the perceptual information. A total of eighteen demonstration are generated to test the homing behavior (See Figure 6.16). Different sets of training trajectories used for learning the homing behavior is shown in the figure D.6. The idea here is to find out how many and what examples suffice for capturing the essence of the behavior. Table D.4 shows the training and test errors recorded for different number of neurons for all the training subsets. Compared to previous two behaviors, the NMSE on the test set is quite high for almost all the different subsets used for training. This is mainly due to the dynamics nature of the policy in taking the corrective action very path thereby taking the shortest route to the goal rather than follow the trend of paths shown in demonstration. Human demonstrations of goal point behaviors tend to drive the robot straight and slowly increase the path curvature to approach the goal. The policy on the other hand reacts to the error in the perceptual variables immediately to move towards the goal. This is seen in from the validation of the learned model on unseen starting positions or typically starting from an untrained state space location. Figure D.7 shows the reproduction trajectory of all the trained models. This concurs to the previous argument in that all the model drive the robot to the goal quite robustly. The high generalization error is mainly attributed to the

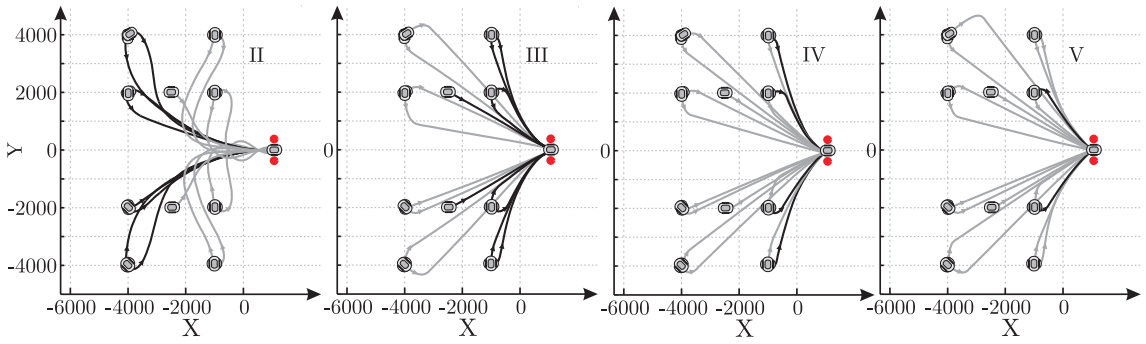


Figure D.7.: Reproduction of the different ANN homing models.

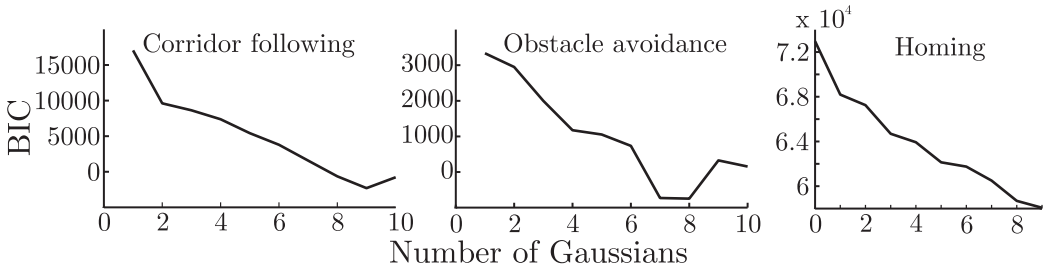


Figure D.8.: BIC values for different number of Gaussians for the learning the behavioral dynamics of corridor following, obstacle avoidance and homing using GMM and SEDS.

biased path taken by the robot starting from unknown starting configurations.

D.3. Gaussian mixture models

D.3.1. Number of Gaussians

The disadvantage of Gaussian mixture models and Expectation Maximization is that the number of Gaussians needed to model the data is not known. BIC computes the log likelihood of the data for the model parameters and thereby describes the accuracy and the complexity of the model. BIC is given by,

$$BIC = -2\mathcal{L} + (\log N)\, n_p$$

where $\mathcal{L}$ is the log likelihood of the data for the model parameters, N is the number of data points and n_p denotes the number of parameters of the GMM. For the three visual behaviors the number of Gaussians are varied between one and ten and the optimal number of Gaussians are determined. Fig. D.8 shows the BIC values for the different number of Gaussians for the three behaviors. The figure shows, nine Gaussians for corridor following, eight for obstacle avoidance and ten for homing have the lowest BIC value.

D.3.2. Modeling translational velocity

The translational velocity for obstacle avoidance behavior is demonstrated through a stop behavior. The robot starting at a safe distance from the obstacle is moved forward while the teacher slows the speed of the robot when reaching closer. The robot is then eventually stopped at a critical distance in front of the obstacle. The minimum distance

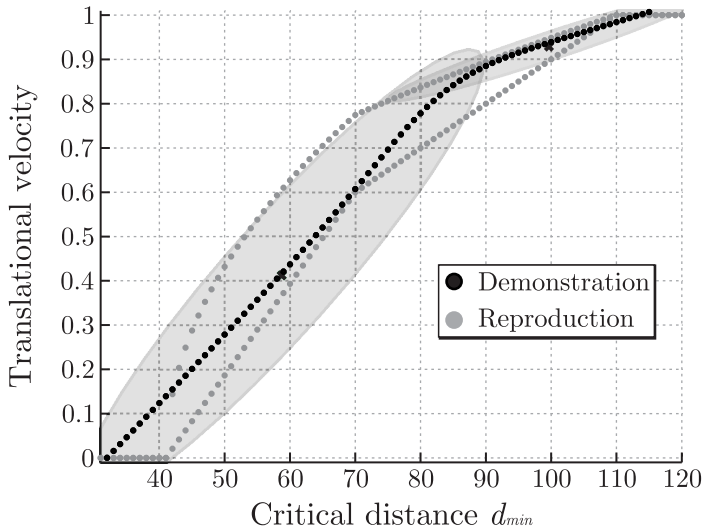


Figure D.9.: GMM modeled on the translational velocity demonstrations. Shown are also the reproductions generated using GMR

to the obstacle d_{min} computed from the segmented image is used to learn the translational velocity. The original translational velocity is normalized with the maximum and modeled with GMM. Two such demonstrations are used as training data and the corresponding Gaussian fit is shown in Fig. D.9. The reproduction trajectory is generated by regressing the GMM by using d_{min} as the input variable.

Table D.1.: Training and testing errors recorded for the different number of neurons for all the training subsets of corridor following behavior

Demo Set	Training MSE $[°/sec]^2$ Number of neurons									Testing MSE $[°/sec]^2$ Number of neurons								
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
I	1.43	1.1	0.29	0.90	0.39	0.26	0.17	0.15	0.18	15.3	25.9	92.7	37.0	194.7	59.2	161.3	196.7	39.8
II	4.03	3.51	2.26	2.50	2.18	1.54	2.22	2.21	2.04	7.74	14.06	29.48	11.2	151.6	128.6	44.4	24.6	14.5
III	6.47	6.46	6.38	5.82	5.66	5.64	5.04	4.97	5.48	2.92	2.73	3.07	3.35	4.25	3.78	4.41	3.25	2.97
IV	5.64	3.05	10.1	5.42	4.02	2.95	2.82	2.86	4.57	6.18	11.3	8.04	6.60	7.48	11.14	12.33	15.59	9.69
V	2.81	2.94	2.72	1.03	2.54	1.39	0.99	1.84	1.37	6.42	6.60	6.75	8.42	6.37	51.7	16.16	17.01	19.82
VI	4.31	2.48	2.17	1.18	0.78	0.78	1.40	0.70	0.64	305	12.7	426	15.7	18.7	50.8	901	90.8	158
VII	5.41	3.25	1.82	1.09	1.17	0.75	0.81	0.78	0.78	21	6.6	77.5	14.96	8.50	244	28.1	47.9	36.9
Demo Set	Training NMSE Number of neurons									Testing NMSE Number of neurons								
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
I	0.82	0.77	0.29	0.48	0.36	0.25	0.13	0.13	0.21	0.66	1.19	4.97	1.65	9.91	3.03	8.36	10.2	2.0
II	1.6	1.12	0.73	0.66	0.86	0.54	0.82	0.91	0.69	0.35	0.70	1.20	0.63	6.99	4.94	2.27	0.91	0.79
III	0.35	0.34	0.34	0.31	0.31	0.29	0.27	0.28	0.29	3.81	3.54	4.1	4.3	5.6	4.56	5.5	3.73	3.54
IV	0.30	0.14	0.44	0.27	0.19	0.13	0.13	0.14	0.23	0.94	0.98	0.66	0.90	0.98	0.94	1.03	1.16	1.04
V	0.05	0.05	0.05	0.02	0.05	0.02	0.02	0.03	0.02	0.85	0.93	0.96	1.12	0.89	4.76	1.42	1.72	1.58
VI	0.21	0.10	0.09	0.04	0.02	0.02	0.04	0.02	0.01	8.2	0.82	31	1.19	1.17	3.77	35	5.39	7.63
VII	0.11	0.06	0.03	0.02	0.02	0.01	0.01	0.01	0.01	2.5	0.85	4.4	1.1	1.04	12	1.7	2.6	2.13

Table D.2.: Training and testing errors recorded for the different number of neurons for all the training subsets of stop behavior and the regression errors.

Model	Training MSE $\times 10^{-4}$ Number of neurons					Testing MSE Number of neurons				
	1	2	3	4	5	1	2	3	4	5
ANN	7.04	1.43	0.46	0.26	0.09	0.019	0.019	0.02	0.02	0.02
Regression	0.0013					0.0197				
Model	Training NMSE Number of neurons					Testing NMSE Number of neurons				
	1	2	3	4	5	1	2	3	4	5
ANN	0.0059	0.0012	0.0004	0.0002	0.0001	0.161	0.159	0.167	0.167	0.167
Regression	0.0108					0.163				

Table D.3.: Training and testing errors recorded for the different number of neurons for all the training subsets. Both models predicting the rotational velocity and curvature are shown.

Predicted output: rotational velocity $\omega[^\circ/\text{sec}]$										
Demo Set	Training MSE $[^\circ/\text{mm}]^2$ Number of neurons					Testing MSE $[^\circ/\text{mm}]^2$ Number of neurons				
	1	2	3	4	5	1	2	3	4	5
I	3.66	1.45	2.46	1.32	1.17	6.62	6.80	7.33	8.09	8.71
II	2.39	2.15	2.01	1.62	1.87	35.8	38.2	33.8	22.6	39.8
III	2.7	2.75	2.60	2.12	1.53	29.0	46.9	59.7	21.3	151
IV	2.98	0.91	3.14	0.68	2.45	4.77	4.91	5.17	4.76	4.35
Demo Set	Training NMSE Number of neurons					Testing NMSE Number of neurons				
	1	2	3	4	5	1	2	3	4	5
I	0.14	0.05	0.09	0.05	0.04	0.18	0.17	0.16	0.18	0.19
II	0.25	0.22	0.19	0.15	0.19	> 1	> 1	> 1	> 1	> 1
III	0.21	0.23	0.2	0.16	0.11	> 1	> 1	> 1	> 1	> 1
IV	0.11	0.03	0.12	0.02	0.09	0.18	0.13	0.20	0.14	0.15
Predicted output: curvature $\kappa[^\circ/\text{mm}]$										
Demo Set	Training MSE $\times 10^{-5}[^\circ/\text{mm}]^2$ Number of neurons					Testing MSE $\times 10^{-5}[^\circ/\text{mm}]^2$ Number of neurons				
	1	2	3	4	5	1	2	3	4	5
I	17	5.8	12	5.8	5.6	25	35	70	32	45
II	10	9.7	14	8.07	9.17	160	170	170	140	160
III	12	11	11	9.3	11	130	250	310	160	162
IV	12	4.2	7.3	4.2	4.8	22	22	17	40	30
Demo Set	Training NMSE Number of neurons					Testing NMSE Number of neurons				
	1	2	3	4	5	1	2	3	4	5
I	0.15	0.04	0.10	0.04	0.04	0.17	0.18	0.26	0.16	0.19
II	0.27	0.23	0.70	0.18	0.22	> 1	> 1	> 1	> 1	> 1
III	0.22	0.2	0.23	0.15	0.20	> 1	> 1	> 1	> 1	> 1
IV	0.10	0.03	0.06	0.03	0.03	0.19	0.14	0.13	0.20	0.16

Table D.4.: Training and testing errors recorded for the different number of neurons for all the training subsets of homing behavior

Demo Set	Training MSE $[^\circ/\text{sec}]^2$ Number of neurons									Testing MSE $[^\circ/\text{sec}]^2$ Number of neurons								
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
I	15.5	15.3	10.2	9.4	9.9	9.7	8.9	9.94	12.1	-	-	-	-	-	-	-	-	-
II	4.24	3.38	3.15	1.93	1.85	1.60	1.77	2.14	1.72	52.1	64.1	52.3	34.8	34.8	34.3	44.04	88.0	72.6
III	17.2	10.6	17.2	4.52	4.3	4.4	4.3	4.04	3.99	16.95	23.1	20.0	25.8	30.2	27.2	30.4	76.4	103.7
IV	13.7	13.2	9.8	9.32	6.82	8.60	8.73	11.91	9.17	44.4	26.19	129.6	31.5	41.3	397	114	22.2	169
V	14.4	15.7	14.8	7.7	7.4	3.3	6.1	10.7	4.1	49	54	>1k	30	38	63	60	70	109
VI	2.25	1.62	1.07	0.66	0.4	1.04	0.36	0.6	0.28	39	76	23	50	19.8	25.4	109	24.4	35.1
Demo Set	Training NMSE Number of neurons									Testing NMSE Number of neurons								
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
I	0.99	0.94	0.55	0.57	0.53	0.54	0.42	0.56	0.60	-	-	-	-	-	-	-	-	-
II	0.49	0.43	0.38	0.22	0.21	0.18	0.21	0.25	0.18	2.6	2.9	2.5	1.5	1.8	1.9	2.06	4.0	2.48
III	0.80	0.43	0.91	0.17	0.12	0.11	0.13	0.13	0.10	2.33	2.23	2.8	2.14	2.63	2.30	2.68	10.7	12.7
IV	0.22	0.21	0.16	0.15	0.11	0.14	0.15	0.19	0.11	5.7	3.0	20	2.90	4.2	79	18	2.3	29
V	0.18	0.20	0.19	0.10	0.09	0.04	0.07	0.13	0.05	6.1	6.4	36	2.6	3.7	6.1	4.7	7.4	13.54
VI	0.08	0.05	0.03	0.02	0.01	0.03	0.01	0.02	0.01	2.5	4.06	1.72	2.44	1.3	1.9	5.2	1.55	2.2

E

Image based door detection

We present one approach to detect and model doors within an indoor environment. There exists many approaches [AKPT04]; [HBB09] in literature over the recent years where the authors fuse the information from a laser and a panoramic camera to build models of doors. Trained on visual and proximity features of positive and negative examples of a door, the model is used to predict the presence of a door within an image. Another pure visual approach can be read in [PNHB09], where templates of typical door panels within an omnidirectional image are matched to detect the door. The extracted door panels are then used to execute a door passing behavior. The presented method here is a mixture of both the above works. Our experimental setup consists of only an omnidirectional camera facing upwards with a 360 ° perspective. The omnidirectional image is converted into a panoramic view to obtain a perspective, undistorted image. Positive and negative door examples are collected from panoramic images and global HOG descriptors are extracted. Fig. E.1 shows one omnidirectional image with its corresponding panoramic image. Shown are also two prototypical positive and one negative door examples.
We use HOG descriptors to extract door features from the training data. HOG [DT05] is a feature extractor that counts the number of gradient orientations in localised portions of an image. The method involves decomposing the image into small square segments and compute the histogram of oriented gradients in each cell. Depending on the resolution of the descriptors needed, the cell size can be varied. With higher cell resolution, HOG captures finer details and presents accurate information about the image. Nevertheless, this backfires when we match two histograms even with smallest amount of noise. Large

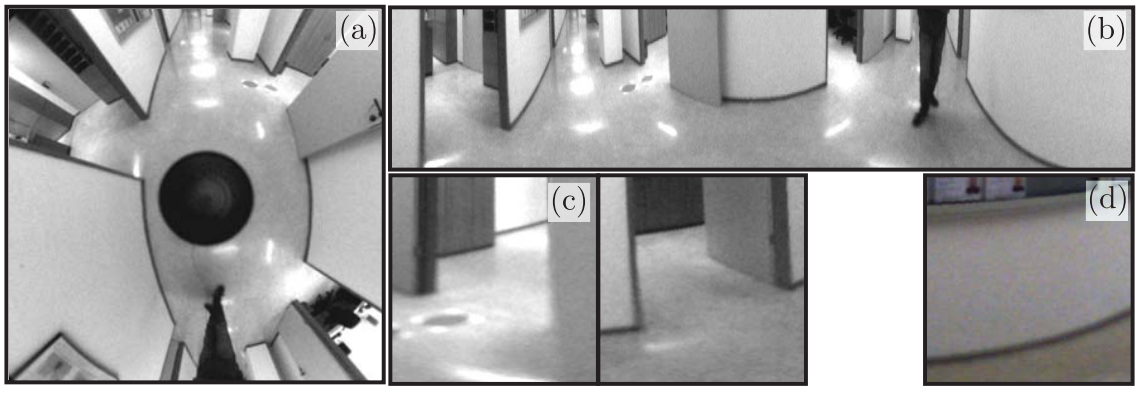


Figure E.1.: (a) Omnidirectional image of a corridor, (b) the corresponding panoramic view of the view, (c) positive examples of door and (d) negative example of a door.

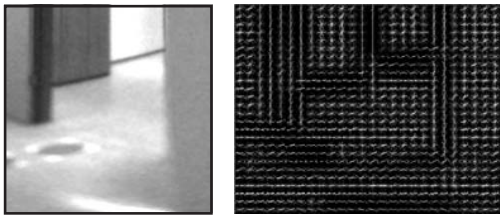


Figure E.2.: (a) A door image and (b) the corresponding HOG descriptor with cell size of 8.

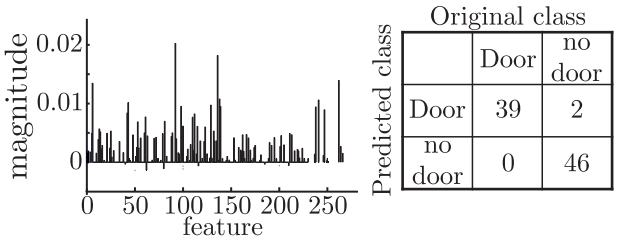


Figure E.3.: (Left) Feature importance of RF door classifier and (Right) bias confusion matrix of the classifier.

cell resolution also looses far too many details thereby making detection unstable. A cell size of 8 is used to compute the histogram. Fig. E.2 shows a door image and visualizes the standard extracted HOG on them. A total of 46 door and 41 non-door examples used to generate the training data for learning a RF classifier. With 100 trees to learn the classification, RF achieves a misclassification rate of 0.022%. Fig. E.3 plots the individual feature importance of the HOG descriptors together with the bias confusion matrix.

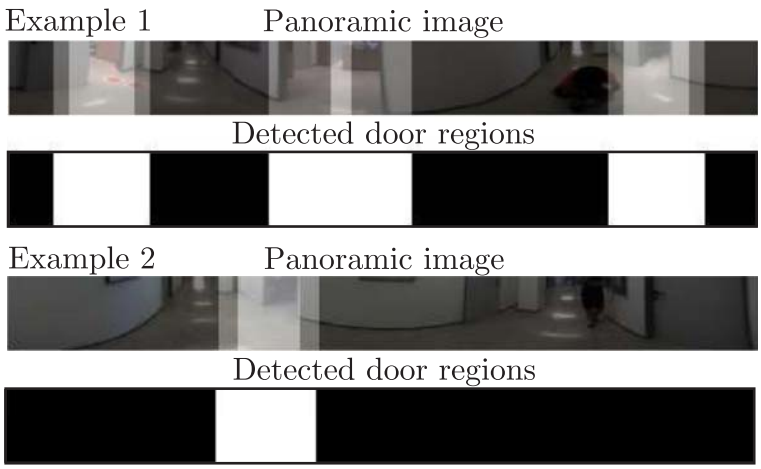


Figure E.4.: Two examples of door detection. Shown are the panoramic image overlaid with the cumulative door probability predicted by RF and the final door regions obtained after thresholding.

The learned model is tested on the omnidirectional image to test for the door detector's performance. The model is tested on several images taken of different doors. The omnidirectional image is first converted to panoramic image and a mask of the same size as training door image is sled over it. The probability of the image window containing the door is cumulatively added and all regions with a probability of higher than 0.7 are considered to contain the floor. Fig. E.4 shows one example of the door detection performance. The successful performance of the classifier makes it a strong candidate to be

implemented for a door passing behavior in the future. With in the framework of LfD, the mentioned methods in this thesis can be successfully extended to a door passing behavior as well.

Bibliography

References

Parts of the material presented in this work has been originally published in conferences and journals. These publications as well as the resources by other researchers are summarized in the following list:

[ABV07] B. D. Argall, B. Browning, and M. Veloso. "Learning by demonstration with critique from a human teacher". In: *ACM/IEEE international conference on Human-robot interaction*. ACM. 2007, pp. 57–64.

[ACG10] S. K. Agrawal, X. Chen, and J. C. Galloway. "Training special needs infants to drive mobile robots using force-feedback joystick". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 4797–4802.

[ACH05] C. A. Acosta-Calderon and H. Hu. "Robot imitation: Body schema and body percept". In: *Applied Bionics and Biomechanics* 2.3 (2005), pp. 131–148.

[ACVB09] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. "A survey of robot learning from demonstration". In: *Robot. Auton. Syst.* 57.5 (2009), pp. 469–483.

[AKPT04] D. Anguelov, D. Koller, E. Parker, and S. Thrun. "Detecting and Modeling Doors with Mobile Robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2004, pp. 3777–3784.

[Alt03] P. Althaus. "Indoor navigation for mobile robots: Control and representations". PhD thesis. KTH, 2003.

[AMS97] C. G. Atkeson, A. W. Moore, and S. Schaal. "Locally weighted learning for control". In: *Artificial intelligence review* 11.1-5 (1997), pp. 75–113.

[AMW05] O. Aycard, J-F. Mari, and R. Washington. "Learning to automatically detect features for mobile robots using second-order Hidden Markov Models". In: *arXiv preprint cs/0501068* (2005).

[AN04] P. Abbeel and A. Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *International conference on Machine learning (ICML)*. ICML '04. Banff, Alberta, Canada, 2004.

[Arg09] B. D. Argall. “Learning mobile robot motion control from demonstration and corrective feedback”. PhD thesis. University of Southern California, 2009.

[Ark98] R.C. Arkin. *Behavior-based robotics*. 1998.

[BB11] R. Battiti and M. Brunato. *Reactive Business Intelligence. From Data to Models to Insight.* Via della Stazione 27, I-38123 Trento - Italy: Reactive Search Srl, Feb. 2011.

[BBB99] G. Bontempi, M. Birattari, and H. Bersini. “Lazy learning for local modelling and control design”. In: *International Journal of Control* 72.7-8 (1999), pp. 643–658.

[BC94] D. J. Berndt and J. Clifford. “Using Dynamic Time Warping to Find Patterns in Time Series”. In: *KDD Workshop'94*. 1994, pp. 359–370.

[BCDS] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Handbook of Robotics Chapter 59: Robot Programming by Demonstration*.

[BCFHLSST99] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. “Experiences with an interactive museum tour-guide robot”. In: *Artificial intelligence* 114.1 (1999), pp. 3–55.

[Bel57] R. Bellman. *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[BFOO08] F. Bonin-Font, A. Ortiz, and G. Oliver. “Visual Navigation for Mobile Robots: A Survey”. In: *J. Intell. Robotics Syst.* 53.3 (Nov. 2008), pp. 263–296.

[BH10] E. A. Billing and T. Hellström. “A formalism for learning from demonstration”. In: *Paladyn* 1.1 (2010), pp. 1–13.

[Bic00] E. Bicho. “Dynamic Approach to Behavior-Based Robotics: design, specification, analysis, simulation and implementation”. PhD thesis. 2000.

[Bil13] A. Billard. “Formalism for Learning from Demonstration”. In: *Scholarpedia* (2013).

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[Bra86] V. Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.

[Bro86] R. A. Brooks. “A Robust Layered Control System for a Mobile Robot”. In: *IEEE Journal of Robotics and Automation* 2.1 (1986), pp. 14–23.

[BS02] P. Buschka and A. Saffiotti. “A virtual sensor for room detection”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 1. 2002, 637–642 vol.1.

[BTVG06] H. Bay, T. Tuytelaars, and L. Van Gool. “Surf: Speeded up robust features”. In: *Computer Vision, (ECCV)*. Springer, 2006, pp. 404–417.

[Cal09] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.

[Cam95] T. Camus. *Real-Time Optical Flow*. Tech. rep. MINNEAPOLIS MINNESOTA, 1995.

[CBB03] G. Cielniak, M. Bennewitz, and W. Burgard. “Where is...? learning and utilizing motion patterns of persons with mobile robots”. In: *IJCAI*. 2003, pp. 909–914.

[CC00] L. F. Costa and R. M. Cesar Jr. *Shape Analysis and Classification: Theory and Practice*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 2000.

[CC13] A. Cherubini and F. Chaumette. “Visual navigation of a mobile robot with laser-based collision avoidance”. In: *The International Journal of Robotics Research* 32.2 (2013), pp. 189–205.

[CD04] G. Cielniak and T. Duckett. “People recognition by mobile robots”. In: *Journal of Intelligent and Fuzzy Systems* 15.1 (2004), pp. 21–27.

[Che09] S. Chernova. “Confidence-based robot policy learning from demonstration”. AAI3358060. PhD thesis. Pittsburgh, PA, USA, 2009.

[CHJH02] M. Campbell, A. J. Hoane Jr, and F. Hsu. “Deep blue”. In: *Artificial intelligence* 134.1 (2002), pp. 57–83.

[CJSW01] H. D. Cheng, X. H. Jiang, Y. Sun, and J. Wang. “Color image segmentation: advances and prospects”. In: *Pattern Recognition* 34.12 (Dec. 2001), pp. 2259–2281.

[CNPW98] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner. “Emergence of functional modularity in robots”. In: *From animals to animats* 5 (1998), pp. 497–504.

[Cor96] P. Corke. *Visual Control of Robots: High-Performance visual servoing*. Vol. 2. Mechatronics. Research Studies Press (John Wiley), 1996.

[CP07] K. Conn and R. A. Peters. “Reinforcement Learning with a Supervisor for a Mobile Robot in a Real-world Environment”. In: *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 2007, pp. 73–78.

[DH02] J. Demiris and G. M. Hayes. “Imitation in animals and artifacts”. In: ed. by K. Dautenhahn and C. L. Nehaniv. Cambridge, MA, USA: MIT Press, 2002. Chap. Imitation as a dual-route process featuring predictive and learning components: a biologically plausible computational model, pp. 327–361.

[DK02] G. N. DeSouza and A. C. Kak. “Vision for mobile robot navigation: A survey”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.2 (2002), pp. 237–267.

[DKM86] Y. Dupain, T. Kamae, and M. Mendés. "Can one measure the temperature of a curve?" In: *Archive for Rational Mechanics and Analysis* 94 (2 1986), pp. 155–163.

[DN02] K. Dautenhahn and C. L. Nehaniv. "The correspondence problem". In: *Imitation in Animals and Artifacts*. Cambridge, MA, USA: MIT Press, 2002.

[DR11] M. Deisenroth and C. E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *International Conference on Machine Learning (ICML)*. 2011, pp. 465–472.

[DT05] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2005, pp. 886–893.

[DW11] Y. Dong and D. L. Woodard. "Eyebrow shape-based features for biometric recognition and gender classification: A feasibility study". In: *International Joint Conference on Biometrics (IJCB)*. IEEE. 2011, pp. 1–8.

[Elm90] J. L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211.

[FB81] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commununications of the ACM* 24.6 (June 1981), pp. 381–395.

[Fer11] D. A. Ferrucci. "IBM's Watson/DeepQA". In: *SIGARCH Comput. Archit. News* 39.3 (June 2011).

[For02] J. R. N. Forbes. "Reinforcement learning for autonomous vehicles". PhD thesis. UNIVERSITY of CALIFORNIA, 2002.

[Fre10] D. Frejlichowski. "An experimental comparison of seven shape descriptors in the general shape analysis problem". In: *Image Analysis and Recognition*. Springer, 2010, pp. 294–305.

[FSM09] D. J. Feil-Seifer and M. J. Matarić. "Human-Robot Interaction". In: ed. by Robert A. Meyers. Springer New York, 2009, pp. 4643–4659.

[FWTPK03] B. R. Fajen, W. H. Warren, S. Temizer, and L. Pack Kaelbling. "A dynamical model of visually-guided steering, obstacle avoidance, and route selection". In: *International Journal of Computer Vision* 54.1-3 (2003), pp. 13–34.

[FZS09] J. Flusser, B. Zitova, and T. Suk. *Moments and Moment Invariants in Pattern Recognition*. Wiley Publishing, 2009. ISBN: 0470699876, 9780470699874.

[GHCB07] F. Guenter, M. Hersch, S. Calinon, and A. Billard. "Reinforcement learning for imitating constrained reaching movements". In: *Advanced Robotics* 21.13 (2007), pp. 1521–1544.

[GL96] G. Z. Grudic and P. D. Lawrence. “Human-to-robot skill transfer using the SPORE approximation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. Apr. 1996, 2962 –2967 vol.4.

[Gro01] A. Großmann. “Continual learning for mobile robots”. PhD thesis. PhD thesis, School, 2001.

[HB95] J. Hoff and G. Bekey. “An architecture for behaviour coordination learning”. In: *IEEE International Conference on Neural Networks*. Vol. 5. IEEE. 1995, pp. 2375–2380.

[HBB09] J. Hensler, M. Blaich, and O. Bittel. “Real-Time Door Detection Based on AdaBoost Learning Algorithm.” In: *Eurobot Conference*. Ed. by A. Gottscheber, D. ObdrzÃąlek, and C. T. Schmidt. Vol. 82. Communications in Computer and Information Science. Springer, 2009, pp. 61–73.

[HBFT03] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. “A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2003.

[Hof97] F. Hoffmann. “Evolutionary learning of mobile robot behaviors”. In: *the First Workshop on Frontiers in Evolutionary Algorithms FEA*. Vol. 97. Citeseer. 1997.

[Hol92] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[HOT06] G. E. Hinton, S. Osindero, and Y-W. Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[HTF01] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001, p. 533.

[HVFF10] P. Henry, C. Vollmer, B. Ferris, and D. Fox. “Learning to navigate through crowded environments”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 981–986.

[HWSK09] C. Hermes, C. Wohler, K. Schenk, and F. Kummert. “Long-term vehicle motion prediction”. In: *IEEE Intelligent Vehicles Symposium*. 2009, pp. 652–657.

[INS02] A. K. Ijspeert, J. Nakanishi, and S. Schaal. “Learning Attractor Landscapes for Learning Motor Primitives”. In: *NIPS*. 2002, pp. 1523–1530.

[JAC97] S. D Jones, C. Andresen, and J. L. Crowley. “Appearance based process for visual navigation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2. IEEE. 1997, pp. 551–557.

[Jae01] H. Jaeger. “The" echo state" approach to analysing and training recurrent neural networks-with an erratum note’”. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (2001).

[Jae07] H. Jaeger. “Echo state network”. In: *Scholarpedia* 2.9 (2007), p. 2330.

[JLMMD06] L. Jun, A. Lilienthal, T. Martinez-Marin, and T. Duckett. “Q-RAN: A Constructive Reinforcement Learning Approach for Robot Behavior Learning”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 2656–2662.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. Second. Springer, Oct. 2002.

[KBP13] J. Kober, D. Bagnell, and J. Peters. “Reinforcement Learning in Robotics: A Survey”. In: (2013).

[KCB97] J. Košecká, H. I. Christensen, and R. Bajcsy. “Experiments in behavior composition”. In: *Robotics and Autonomous systems* 19.3 (1997), pp. 287–298.

[KFD95] M. Kaiser, H. Friedrich, and R. Dillmann. “Obtaining Good Performance From A Bad Teacher”. In: *International Conference On Machine Learning, Workshop On Programming By Demonstration*. 1995.

[KH07] T. Kohonen and T. Honkela. “Kohonen network”. In: *Scholarpedia* 1 (2007), p. 1568.

[KHP11] K. Kampa, E. Hasanbelliu, and J. C. Principe. “Closed-form cauchy-schwarz PDF divergence for mixture of Gaussians”. In: *International Joint Conference on Neural Networks (IJCNN)*. 2011, pp. 2578–2585.

[KIOIN02] R. Kanda, H. Ishiguro, T. Ono, M. Imai, and R. Nakatsu. “Development and evaluation of an interactive humanoid robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. IEEE. 2002, pp. 1848–1855.

[KJ97] R. Kohavi and G. H. John. “Wrappers for Feature Subset Selection”. In: *Artificial Intelligence* 97.1-2 (1997), pp. 273–324.

[KL51] S. Kullback and R. A. Leibler. “On information and sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[KNCC11] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell. “Upper-body Kinesthetic Teaching of a Free-standing Humanoid Robot”. In: *IEEE Conf. on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 3970–3975.

[Kni02] J. Knight. “Towards fully autonomous visual navigation”. PhD thesis. Department of Engineering Science, University of Oxford, 2002.

[Kob12] J. Kober. “Learning Motor Skills: From Algorithms to Robot Experiments”. In: (2012).

[KPAK13] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. "Human-aware robot navigation: A survey". In: *Robotics and Autonomous Systems* (2013). ISSN: 0921-8890.

[KTM10] N. P. Koenig, L. Takayama, and M. J. Mataric. "Communication and knowledge sharing in human-robot interaction and learning from demonstration". In: *Neural Networks* 23.8-9 (2010), pp. 1104–1112.

[KZB11] S. M. Khansari-Zadeh and A. Billard. "Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models". In: *IEEE Transaction on Robotics* 27.5 (2011), pp. 943–957.

[LBG97] L. M. Lorigo, R. Brooks, and W. Grimsou. "Visually-guided obstacle avoidance in unstructured environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 1. IEEE. 1997, pp. 373–379.

[LD06] J. Li and T. Duckett. "Growing RBF networks for learning reactive behaviours in mobile robotics". In: *International journal of vehicle autonomous systems* 4.2 (2006), pp. 285–307.

[Low99] D. G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *International Conference on Computer Vision*. Vol. 2. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–1157.

[LY96] C. Lee and X. Yangsheng. "Online, interactive learning of gestures for human/robot interfaces". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 1996, 2982–2987 vol.4.

[LZ00] K-J. Lee and H-T. Zhang. "Learning robot behaviors by evolving genetic programs". In: *26th Annual Confjerence of the IEEE Industrial Electronics Society (IECON)*. Vol. 4. IEEE. 2000, pp. 2867–2872.

[Mae89] P. Maes. "How to do the right thing". In: *Connection Science Journal* 1.3 (1989), pp. 291–323.

[Mat01] M. J. Matarić. "Learning in behavior-based multi-robot systems: policies, models, and other agents". In: *Cognitive Systems Research* 2.1 (Apr. 2001), pp. 81–93. ISSN: 1389-0417.

[MEBFGK10] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. "The Office Marathon: Robust Navigation in an Indoor Office Environment". In: *International Conference on Robotics and Automation (ICRA)*. 2010.

[MIJ08] Y. Mingqiang, K. K. Idiyo, and R. Joseph. "A Survey of Shape Feature Extraction Techniques". In: *Pattern Recognition, Peng-Yeng Yin (Ed.) (2008) 43-90* (Nov. 2008). Ed. by Peng-Yeng Yin, pp. 43–90.

[Mit97] T. M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

[MM11] J. Mumm and B. Mutlu. “Human-robot proxemics: physical and psychological distancing in human-robot interaction”. In: *International conference on Human-robot interaction*. HRI ’11. Lausanne, Switzerland: ACM, 2011, pp. 331–338. ISBN: 978-1-4503-0561-7.

[MMB06] O. Martínez Mozos and W. Burgard. “Supervised learning of topological maps using semantic information extracted from range data”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 2772–2777.

[MMD05] T. Martinez-Marin and T. Duckett. “Fast Reinforcement Learning for Vision-guided Mobile Robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2005, pp. 4170–4175.

[MMSB05] O. Martínez Mozos, C. Stachniss, and W. Burgard. “Supervised Learning of Places from Range Data using Adaboost”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2005, pp. 1742–1747.

[Mob] Adept MobileRobots. *Pioneer Robots Software Development Kit*. URL: `http://www.mobilerobots.com/Software.aspx`.

[MRTJB07] O. M. Mozos, A. Rottmann, R. Triebel, P. Jensfelt, and W. Burgard. “Supervised semantic labeling of places using information extracted from sensor data”. In: *Robotics and Autonomous Systems* 55 (2007), pp. 391–402.

[MSG10] C. Martin, F. F. Steege, and H. M. Gross. “Estimation of pointing poses for visually instructing mobile robots under real world conditions”. In: *Robotics and Autonomous Systems* 58.2 (2010), pp. 174–185.

[Mur00] R. R. Murphy. *Introduction to AI Robotics*. 1st. Cambridge, MA, USA: MIT Press, 2000.

[Nan13] PMD[Vision] CamBoard Nano. *Spec Sheet CamBoardNano*. 2013. URL: `http://www.pmdtec.com/html/pdf/flyer_camboard_nano.pdf`.

[NCJOF08] M. Nicolescu, O. Chadwicke Jenkins, A. Olenderski, and E. Fritzinger. “Learning behavior fusion from demonstration”. In: *Interaction Studies* 9.2 (2008), pp. 319–352.

[ND01] C. L. Nehaniv and K. Dautenhahn. “Like me?-measures of correspondence and imitation”. In: *Cybernetics & Systems* 32.1-2 (2001), pp. 11–51.

[NFH06] T. Nierobisch, W. Fischer, and F. Hoffmann. “Large View Visual Servoing of a Mobile Robot with a Pan-Tilt Camera”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 3307–3312.

[NH08] A. Nüchter and J. Hertzberg. “Towards semantic maps for mobile robots”. In: *Robot. Auton. Syst.* 56.11 (Nov. 2008), pp. 915–926. ISSN: 0921-8890.

[NM01] M. Nicolescu and M. J. Matarić. “Experience-based representation construction: learning from human and robot teachers”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2001, pp. 740 –745.

[Nof99] S. Y. Nof. *Handbook of industrial robotics*. Vol. 1. John Wiley & Sons, 1999.

[NPHB09] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Imitation Learning for Visual Robotic Behaviors”. In: *Proceedings of the 19. Workshop Computational Intelligence*. Dortmund, 2009, pp. 221–236.

[NPHB10] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Robot programming by demonstration”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 288–299.

[NPHB11a] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Scenario and context specific visual robot behavior learning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 1180–1185.

[NPHB11b] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Situated learning of visual robot behaviors”. In: *Intelligent Robotics and Applications*. Springer, 2011, pp. 172–182.

[NPHB12] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Human-Machine Interfaces for Intuitive and Effective Demonstrations of Mobile Robot Behaviors”. In: *Proceedings of the 22. Workshop Computational Intelligence*. Dortmund, 2012, pp. 427–441.

[NPHB13] K. K. Narayanan, L. F. Posada, F. Hoffmann, and T. Bertram. “Acquisition of Behavioral Dynamics for Vision Based Mobile Robot Navigation from Demonstrations”. In: *Mechatronic Systems*. 1. 2013, pp. 37–44.

[NS02] R. Nafe and W. Schlote. “Methods for shape analysis of two-dimensional closed contoursâĂŤa biologically important, but widely neglected field in histopathology”. In: *Electronic Journal of Pathology and Histology* 8.2 (2002).

[OD08] D. L. Olson and D. Delen. *Advanced Data Mining Techniques*. 1st. Springer Publishing Company, Incorporated, 2008.

[Ope] *OpenNI User Guide*. OpenNI organization. Nov. 2010. URL: `http://www.openni.org/documentation`.

[Pir99] P. Pirjanian. *Behavior Coordination Mechanisms - State-of-the-art*. Tech. rep. IRIS-99-375. Institute for Robotics and Intelligent Systems, 1999.

[Pla06] M. Plaue. *Technical Report: Analysis of the PMD Imaging System*. Tech. rep. Interdisciplinary Center for Scientific Computing,University of Heidelberg, Dec. 2006.

[PLD05] H. Peng, F. Long, and C. Ding. “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (Aug. 2005), pp. 1226–1238.

[PNHB09] L. F. Posada, T. Nierobisch, F. Hoffmann, and T. Bertram. “Image Signal Processing for Visual Door Passing with an Omnidirectional Camera”. In: *VISAPP (1)*. 2009, pp. 472–479.

[PNHB10] L. F. Posada, K. K. Narayanan, F. Hoffmann, and T. Bertram. “Floor segmentation of omnidirectional images for mobile robot visual navigation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 804 –809.

[PNHB11] L. F. Posada, K. K. Narayanan, F. Hoffmann, and T. Bertram. “Ensemble of experts for robust floor-obstacle segmentation of omnidirectional images for mobile robot visual navigation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2011, pp. 439 –444.

[Pol06] R. Polikar. “Ensemble Based Systems in Decision Making”. In: *IEEE Circuits and Systems Magazine* 6.3 (2006), pp. 21–45.

[Pom89] D. A. Pomerleau. *Alvinn: An autonomous land vehicle in a neural network*. Tech. rep. DTIC Document, 1989.

[Pos] L. F. Posada. *Matlab-Aria Interface*. Tech. rep. Institute of Control Theory and Systems Engineering, Technische Universität Dortmund, Germany.

[QCGFFLWN09] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. “ROS: an open-source Robot Operating System”. In: *ICRA Workshop on Open Source Software*. 2009.

[RC01] A. Rizzi and R. Cassinis. “A robot self-localization system based on omnidirectional color images”. In: *Robotics and Autonomous Systems* 34.1 (2001), pp. 23–38.

[RGH11] May R., Dandy G., and Maier H. “Review of Input Variable Selection Methods for Artificial Neural Networks”. In: *Artificial Neural Networks-Methodological Advances and Biomedical Applications*. Ed. by K. Suzuki. Intech, 2011.

[RK03] C. E. Rasmussen and M. Kuss. “Gaussian Processes in Reinforcement Learning”. In: *NIPS*. 2003, pp. 751–759.

[RLDL07] E. Royer, M. Lhuillier, M. Dhome, and J-M. Lavest. “Monocular vision for mobile robot localization and autonomous navigation”. In: *International Journal of Computer Vision* 74.3 (2007), pp. 237–260.

[Ros97] J. K. Rosenblatt. “DAMN: A distributed architecture for mobile navigation”. In: *Journal of Experimental & Theoretical Artificial Intelligence* 9.2-3 (1997), pp. 339–360.

[RPB09] R. Roberts, C. Pippin, and T. Balch. “Learning outdoor mobile robot behaviors by example”. In: *Journal of Field Robotics* 26.2 (2009), pp. 176–195.

[RSGB09] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard. “Unsupervised learning of 3d object models from partial views”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2009, pp. 801–806.

[Saf97] A. Saffiotti. “The uses of fuzzy logic in autonomous robot navigation”. In: *Soft Computing* 1.4 (1997), pp. 180–197.

[SAMB12] E. L. Sauser, B. D. Argall, G. Metta, and A. G Billard. “Iterative learning of grasp adaptation through human corrections”. In: *Robotics and Autonomous Systems* 60.1 (2012), pp. 55–71.

[SB91] M. J. Swain and D. H. Ballard. “Color indexing”. In: *International Journal of Computer Vision* 7.1 (Nov. 1991), pp. 11–32.

[SB98] Richard S. S. and Andrew G. B. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[SC90] H. Sakoe and S. Chiba. “Readings in speech recognition”. In: ed. by Alex Waibel and Kai-Fu Lee. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. Chap. Dynamic programming algorithm optimization for spoken word recognition, pp. 159–165.

[Sch92] J. Schmidhuber. “Learning complex, extended sequences using the principle of history compression”. In: *Neural Computation* 4.2 (1992), pp. 234–242.

[SDE95] G. Schöner, M. Dose, and C. Engels. “Dynamics of behavior: Theory and applications for autonomous robot architectures”. In: *Robot. Auton. Syst.* 16.2-4 (1995), pp. 213–245.

[SLL01] S. Se, D. G. Lowe, and J. Little. “Vision-based mobile robot localization and mapping using scale-invariant features”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. IEEE. 2001, pp. 2051–2058.

[SMI07] S. Schaal, P. Mohajerian, and A. K. Ijspeert. “Dynamics systems vs. optimal controlâĂŤa unifying view”. In: *Progress in brain research* 165 (2007), pp. 425–445.

[SMS06] D. Scaramuzza, A. Martinelli, and R. Siegwart. “A Toolbox for Easy Calibrating Omnidirectional Cameras”. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2006.

[SMUAS07] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon. “A Human Aware Mobile Robot Motion Planner”. In: *IEEE Transactions on Robotics* 23.5 (2007), pp. 874–883. ISSN: 1552-3098.

[SPK00] W. D. Smart and L. Pack Kaelbling. “Practical Reinforcement Learning in Continuous Spaces”. In: Morgan Kaufmann, 2000, pp. 903–910.

[SPK02] W. D. Smart and L. Pack Kaelbling. “Effective reinforcement learning for mobile robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. IEEE. 2002, pp. 3404–3410.

[SSKFFBCM13] J. Shotton, R. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. “Real-time human pose recognition in parts from single depth images”. In: *Commun. ACM* 56.1 (Jan. 2013), pp. 116–124. ISSN: 0001-0782.

[Sut13] I. Sutskever. “Training Recurrent Neural Networks”. PhD thesis. University of Toronto, 2013.

[TB06] A. L. Thomaz and C. Breazeal. “Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance”. In: *AAAI*. Vol. 8. 1. 2006, pp. 8–6.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Vol. 1. MIT press Cambridge, 2005.

[Thr96] S. Thrun. *Explanation-Based Neural Network Learning*. Springer, 1996.

[TN99] J. Tani and S. Nolfi. “Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems”. In: *Neural Networks* 12.7 (1999), pp. 1131–1141.

[Tur50] A. M. Turing. “Computing machinery and intelligence”. In: *Mind* 59.236 (1950), pp. 433–460.

[VBK01] N. Vlassis, R. Bunschoten, and B. Krose. “Learning task-relevant features from robot data”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2001, 499–504 vol.1.

[VK99] N. A. Vlassis and B. J. A. Kröse. “Robot environment modeling via principal component regression”. In: *IROS*. 1999, pp. 677–682.

[VKG05] M. Vlachos, G. Kollios, and D. Gunopulos. “Elastic Translation Invariant Matching of Trajectories”. In: *Machine Learning* 58.2-3 (2005), pp. 301–334.

[War06] W. H. Warren. “The dynamics of perception and action.” In: *Psychological Review* 113.2 (2006), pp. 358–389.

[WB09] Z. Wang and A. C. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (Jan. 2009), pp. 98 –117.

[WGLSv00] N. Winters, J. Gaspar, G. Lacey, and J. Santos-victor. “Omni-directional vision for robot navigation”. In: *IEEE Workshop on Omnidirectional Vision*. 2000, pp. 21–28.

[WRR03] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. “Singular value decomposition and principal component analysis”. In: *A Practical Approach to Microarray Data Analysis* (2003), pp. 91–109.

[WZ98] K. Ward and A. Zelinsky. "Acquiring mobile robot behaviors by learning trajectory velocities with multiple FAM matrices". In: *IEEE International Conference on Robotics and Automation (ICRA).* Vol. 1. IEEE. 1998, pp. 668–673.

[ZBK07] Z. Zivkovic, O. Booij, and B. Kröse. "From images to rooms". In: *Robot. Auton. Syst.* 55.5 (May 2007), pp. 411–418. ISSN: 0921-8890.

[ZL01] D. Zhang and G. Lu. "A comparative study on shape retrieval using Fourier descriptors with different shape signatures". In: *International conference on intelligent multimedia and distance education (ICIMADE).* 2001, pp. 1–9.