Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

# Qualitative Perception and Control of Mobile Platforms

*Mihir Patil*

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul Plöger
Nico Huebel
Santosh Thoduka

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

| | |
|---|---|
| Date | Mihir Patil |

# Abstract

The objective of this project is to evaluate a qualitative approach for perception and control of a mobile platform, in the context of improving efficiency for the task of robot navigation. Existing techniques for robot navigation try to extract all the information from the environment and represent it in a highly precise numerical manner to aid the design and deployment pre-planned path trajectories. This leads to a prolix of information that the robot will have to process in order to execute a single action, thus highlighting a grossly inefficient manner of representing and utilizing raw environmental data especially in the case of vision based navigation. We present a generalized framework that uses qualitative spatial representations to abstract relative distance and motion data between two objects and further utilizes this to infer the robot's trajectory, thus enabling it to control the motion of the robot via suitable motion updates. We evaluate our approach against a quantitative map based representation to determine the validity of the claims of improved efficiency that are often associated with qualitative representations.

# Acknowledgements

I would like to express my sincere gratitude towards all those who provided me the guidance and motivation to complete this project. I would especially like to thank Nico Heubel and Santosh Thoduka, for their simulating discussions and suggestions that helped me overcome various hurdles associated with the development and completion of this project.

# Glossary

| | | |
|---|---|---|
| **QTC** | - | Qualitative Trajectory Calculus |
| **CDC** | - | Composite Rigid Body Algorithm |
| **QRPC** | - | Recursive Newton-Euler Algorithm |
| **CyCord** | - | Stack of Tasks |
| **RCC** | - | Region Connection Calculus |
| **ARGD** | - | Vector Force Field |
| **OPRA** | - | Whole Body Control |
| **TPCC** | - | Whole-Body Operational Space Control |

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

This project deals with the utilization of qualitative spatial representations for the task of robot navigation in a corridor environment. Qualitative spatial representations refer to human level abstractions [14] of the physical space and the spatial entities that occupy this space, these abstractions may be based on any primitive spatial transforms such as distance, direction, size, shape etc. Depending upon the primitive used, concise and descriptive relations([left-right],[near-far]) between the spatial entities are built which can then be used for reasoning and performing specific actions depending upon the task at hand. Since we explore the utility and efficiency of such representations from a robot navigation perspective our focus lies on two specific modules that are crucial for any navigation task, namely perception and control.

## 1.1 Motivation

While qualitative representations have been used almost extensively in the field of GIS (geographical information systems) [46]for numerous years, their utilization in the field of robotics has been comparatively limited. Some of the reasons that impede the widespread usage of qualitative representations in robotics are:

- The lack of a framework for effective utilization of the qualitative representation in robotics.

- The existence of a vast number of calculi or models that exist for each spatial primitive and the lack of comprehensive details for each makes it difficult to implement them in a generalized manner. [13], [8].
- The abstract nature of these high level representations which obfuscates precise numerical data [35].

The abstract nature of qualitative representations isn't necessarily a drawback as it allows representation to better compensate for vague data [35], this in turn presents a pleasant advantage in the form of being more robust to noisy data. Consider a crowded lobby environment that is constantly changing due to the constant movement of a large number of people, in such as scenario qualitative representations are instantly useful [22], as precise metric maps or precise trajectories may fail due to the constant and often unprecedented change to the environment [41]. Additionally, [35] conclusively states that quantitative representations(precise numerical representations) compute up-to an unnecessary level of accuracy, their interpretation of the physical space. Though this is useful in conditions where highly precise control of the robot's trajectory is required, it becomes an unwanted hindrance elsewhere due to the highly precise and numerical manner in which the environment and the spatial entities are depicted, not only is this unintuitive to a human agent it also requires excessive computational resources (inefficient) [35], [7], [41]to get the representation of every spatial entity exactly right.

Consider the case of a robot tasked with driving along a corridor towards it's end while avoiding collision with the walls. In such a scenario where precise quantitative representations are deemed surplus to requirements,the robot can be controlled using qualitative representations for the perceived physical space. The significant benefit of using a qualitative representation is that we no longer trying to follow a preplanned, precisely controlled trajectory. Therefore eliminating the need to constantly issue control commands. This implies that a qualitative approach would be more efficient [8] (in terms of CPU and battery usage [47]).

Furthermore, humans often communicate basic navigation tasks to each other using approximate spatial relationships to observable landmarks [33] [8] ,without

requiring a precise map (for example, 'walk past the computers and take a left at the elevator') [41]. As qualitative representations also use such high level abstractions, by deploying them we would be emulating a similar communication pattern which in turn facilitates a better human robot interaction [17]. Also this manner of high level robot programming should be carried out without having to refer to numeric data. Ideally, a robot programmer should describe the task to the robot in terms that they would use to describe it when doing it themselves ("task-level" programming). People do not naturally think of physical actions in terms of joint angles or numeric co-ordinates, so high level robot programming should be done in non-numeric terms [7].

## 1.2 Problem Statement

The fundamental definition of our problem statement would be to develop a generalized and modular framework that achieves collision(with respect to the walls) free navigation in a corridor environment using robust features from the surrounding walls to develop qualitative representations that effectively model the common-sense level relationships amongst the features as well as the robot, and further utilize these relationships to evaluate and infer the instantaneous motion of the robot in order to facilitate corrective updates to it's motion while also comparing the efficiency paradigm against a non-qualitative approach to provide tangible proof's that support the claim of better efficiency for qualitative representations.

# 2

# State of the Art

Over the years, research into qualitative spatial representations has led to the development of various models which can be used to represent the visually observable space.From a more scientific perspective these models are known as 'Qualitative calculi', the current state of the art calculi rely on a single spatial primitive such as distance, direction, topology etc., to describe a set of relationships amongst the observable objects. In general each calculi comes with its own set of benefits and drawbacks as each one of them has been tailored to exploit different aspects of space[8], [14].

For the case of robot navigation, current state of the art approaches utilize qualitative calculi that can provide both spatial and temporal information such as the QTC [45] or QRPC [26]. There exist comprehensive surveys that provide detailed information about each of the existing qualitative calculi[12], [8], [13], [14] hence this state of the art aims to provide only a concise overview of the qualitative calculi that are advantageous to our application. We shall look into the relationships afforded by each of these calculi and their classification based on the domain of their utility.

## 2.1 Forms of qualitative spatial representations

### 2.1.1 Topological Representations

: This is the most fundamental spatial representation[12], [8] wherein the observed space is divided into distinctive regions based either on distinctive points in space or on separable objects found in the space. Topological representations draw heavily from the field of "Mereology"(the theory of parthood) to describe relations between the distinctive regions [13], [14]. Qualitative calculi such as RCC, Interval Algebra, n-intersections etc. belong to this category. Such representations deal with the '*invariant properties that are under continuous deformations of objects, including translating, rotating and scaling*', and often include only spatial information while completely disregarding temporal data.

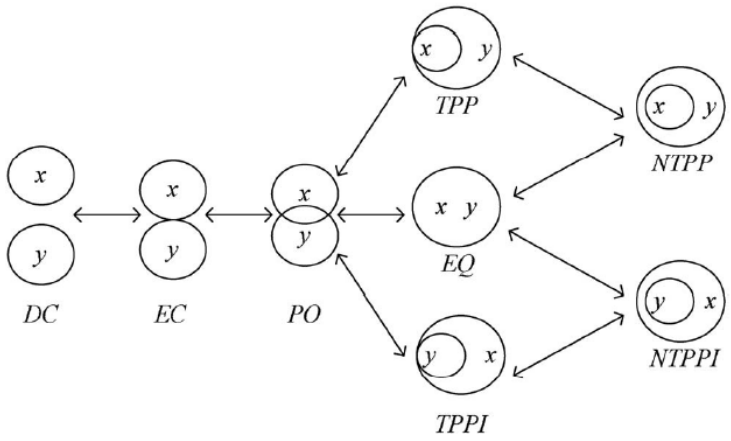
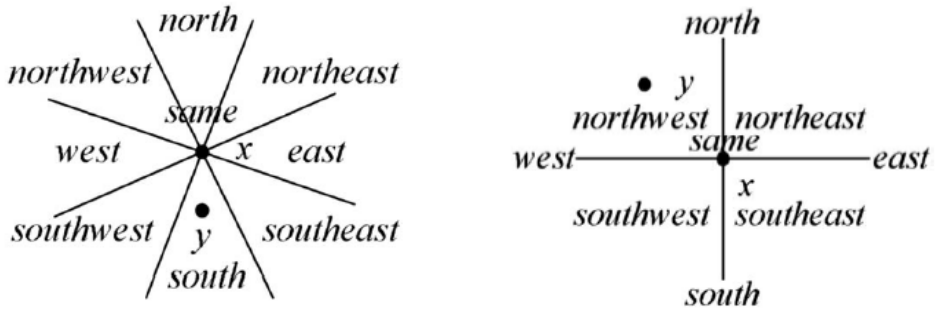



Figure 2.1: The eight jointly exhaustive and pairwise disjoint relations of region connection calculus (RCC8). The arrows show which relation is the next relation a configuration would transit to, assuming the continuous movements or deformations [12], [8], [13], [14].

### 2.1.2 Directional Representations

: The relative direction between two different objects can be represented using directional relations/representations [12], [8], [13], [14]. These representations rely

on three primary elements, a reference object, a reference frame and a target object to define a valid relation between two different objects. Directional representations are widely classified into two categories, point based and projection based with the discerning factor being the dimension of the objects involved and distinction of space using either cone shaped spatial sectors or by using vertical and horizontal lines to create smaller rectangular sectors. Furthermore, the directional calculus isn't restricted to using only cardinal directions, it also allows the use of nominal directional information such as left, right etc. to describe the directional relations. Qualitative calculi such as CDC, OPRA, CyCord etc. utilize the directional representation. Being based of topological representations, directional representations also include only spatial information while disregarding temporal data.




(a) Cone-shaped direction relations [28]

(b) Projection-based direction relations [29]

Figure 2.2: The point based and projection based direction representations [8]

### 2.1.3 Distance Representations

: The qualitative representation of spatial distance can be classified into two groups namely absolute and relative [28], [8], [13], [29]. This classification is made solely on the basis of the presence/absence of an extraneous referential object in the relation between two objects. This distinction can be clearly illustrated by the following example,'the distance between A and B is 8 meters' or 'A is near B', this is a absolute approach as the distance is measured directly between two objects.Whereas saying that 'A is closer to B than that to C' classifies as a relative

approaches as this involves the comparison to a third object. Furthermore, it has been shown that absolute approaches can be qualitative or quantitative, but relative approaches are commonly qualitative [13].Qualitative calculi such as the ARGD (or Delta) and TPCC use the distance representations to describe the observable space. Distance based relations have found to be insufficient by themselves when it comes to the task of robot manipulation/navigation and hence are often used in combination with distance representations to yield a fairly suitable and complete representation of the environment [8]. Like with the direction representations this calculi also lacks temporal data in the encoded relations and is hence unsuitable for applications involving moving objects.

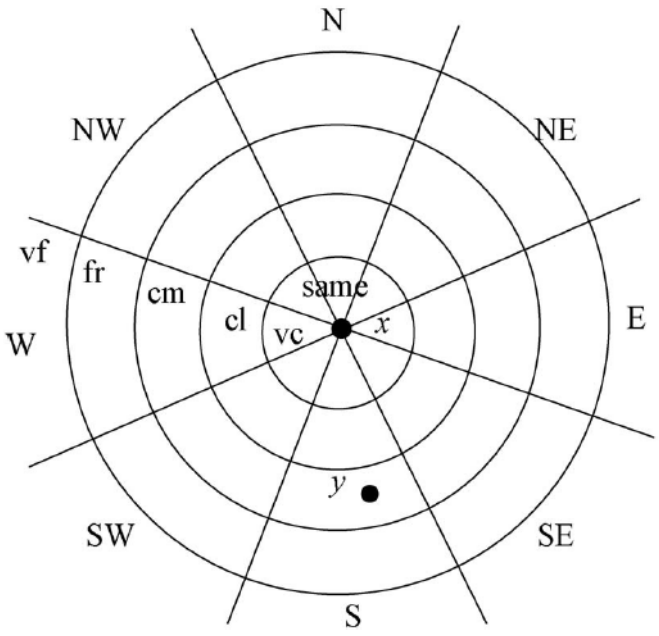



Figure 2.3: An representation of the combination of cone-shaped direction and absolute distance: very close(vc), close(cl), commensurate(cm), far(fr) and very far(vf), [11].

### 2.1.4 Moving object Representations

Topological representations, directional representations and distance representations describe relations between stationary objects. This limitation encouraged the

development of a moving object representation which can qualitatively represent moving objects and their trajectories [12], [8], [13], [14] . These representations effectively deal with both spatial and temporal data to describe valid relations among mobile objects. While these relations include some directional information they mainly describe the relative motion between two objects and not relative direction. The relative motion between two objects is described using oriented line segments which are approximations of the trajectory of the objects in motion. QTC, QRPC are the two prominent calculi that utilize moving object representations. Moving object representations and the calculi using these representations have been proven to have solved the problem of representing moving objects but since these relations lack any distance information, they are still prone to failure and often need a complimentary distance calculi to ensure that a mobile object(robot) can successfully move around in the given environment without collisions.

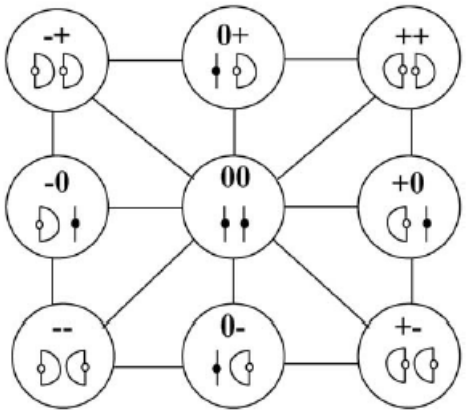



Figure 2.4: Basic relations of basic Qualitative Trajectory Calculus (QTCB) in a conceptual neighborhood diagram. The solid dots represent the stationary objects and the open dots represent the moving objects [**?** ], [**?** ].

| **Domain** | **Model Name** | **Type of objects** | **Description** |
|---|---|---|---|
| Temporal | Interval Algebra [3] | Time Intervals | It does not consider time instants. |
| | Extended Interval Algebra [23] | Time intervals and extreme points of the intervals | It extends IA(Interval Algebra) model by considering new relationships including the extremes of the interval and it introduces the notion of conceptual neighborhood. |
| Spatial | Region Connection Calculus [37] | Sets | It uses the geometric properties associated to the connection between two sets to establish relationships that are no longer linear but planar, and which are invariant to translation, rotation and scaling. |
| | Cardinal Reference System(CRS) [21] | Generic objects | It describes the position of any object by using a cardinal orientation as reference system and by adding also a neutral region |
| | FFC (Flip Flop Calculus) [30] | Points | It is based on the possible positions of a point C with respect to a segment AB defined by other two points, A and B. |
| | SCC (Single Cross Calculus) [23] | Points | Describes the possible positions of a point C with respect to a segment AB and the orthogonal line to segment AB on B. |
| | DCC (Double Cross Calculus) [24] | Points | Describes the possible positions of a point C with respect to a segment AB and two orthogonal lines to segment AB on A and B. |
| | Oriented point based Reasoning [34] | Oriented Points | It is based on the relative orientation between pairs of oriented points in terms of two qualitative spatial dichotomies: the frontback and leftright. |
| | DRA (Dipole Relation Algebra) [18], [19] | Dipoles (or oriented segments) | It is based on the relative position of oriented segments. |
| | OPRA (Oriented Point Relation Algebra) [20] | Oriented points | As DRA model, it is also based on the relative position two oriented points, but it supports different levels of granularity. |
| Spatio-temporal | QTC (Qualitative Trajectory Calculus) [44] | Points | It describes the possible relations among two moving points in terms of the frontback and leftright dichotomies. |
| | QRPC (Qualitative Rectilinear Projection Calculus) [26] | Oriented points | It establishes the possible relations of an object with respect to the trajectory of another object depending on the cross-point of the trajectories and the relative position among them |

Table 2.1: Table(from [26]) describing the key features of the more representative models(calculi) of qualitative representations of spatial or temporal domains in the existing literature.

### 2.1.5 Conclusion

From the above breakdown of the representations and the calculi, it is easy to summarize that distance representations and moving object representations are the most promising representations for our application in mobile robot navigation [8], [12]. Consequently the calculi associated with these representations will be the ones that are further scrutinized in the following section. The reasoning behind this conclusion is fairly simple moving object representations are basically spatio-temporal [13], [14], [48] representations which take into account both the spatial and temporal data to create abstractions of the objects trajectory, this is crucial when dealing with mobile objects as this gives a more concrete representation of the objects in motion [26]. In the case of distance representations although these representations deal only with spatial information, they provide explicit information on how close or far the objects under consideration are. Thus effectively capturing the possibility of a collision between the objects, this sort of information cannot be found in the direction and topological representations hence rendering them unfavorable for our application in mobile robot navigation .

## 2.2 Analyzing qualitative calculi for navigation

This section aims to provide a thorough understanding at a selective group of qualitative calculi based on the conclusions drawn from the previous section. Namely we shall look at the qualitative calculi such as the QTC, QRPC and ARGD which use moving object representations and distance representations and function in the spatio-temporal and spatial domains respectively.

### 2.2.1 Qualitative Trajectory calculus

The QTC calculus was developed to solve the problem of inadequate representation of mobile objects in the spatio-temporal domain, which until then were represented only in the spatial domain using either the RCC or the n-intersections calculi [46], [45]. The major drawbacks of those calculi was their failure to deal with temporal data. Hence mobile objects were often abstracted as stationary

regions using either 'disconnected from' relation (DC) in RCC or 'disjoint' relation in the 9-intersection model, with the exception of a few limiting cases where the two objects meet, such as a collision between two mobile robots [43]. Thus, the limiting factor of these formalisms was that all the DC relations were non-differentiable, due to the ignorance of information regarding relative motion between the two objects .

The QTC calculus constitutes of two major variants the $QTC_B$ and the $QTC_C$, with the major difference between the two being the inclusion of the directional information in the relationships (between two objects) of the $QTC_C$ calculus. Both versions of the QTC are adept at dealing with qualitative movement of objects in one, two and three dimensions. The $QTC_B$ calculus, takes Euclidean distance between two objects as the only constraining dimension.The abstraction of movement is depicted using three qualitative values:

- 0: the object is stable with respect to the other object.
- -: the object is moving towards the other object.
- +: the object is moving away from the other object.

The assigning of these qualitative values to an object's movement depends upon the relative position (constrained by distance) and relative speed of the two objects with respect to each other. This can be illustrated by the following example, consider two objects 'j' and 'k', the possible values for 'j' :

- based on relative position at a time 't' are:
    1. 0 (stable): if there is no change in the relative position with respect to 'k' (no change in relative distance ).
    2. - (moving towards): if there is a change in the relative position with respect to 'k', such that the relative distance between the objects decreases.
    3. + (moving away): if there is a change in the relative position with respect to 'k', such that the relative distance between the objects increases.

- based on relative speed at a time 't' are:

    1. 0 (stable): if speed of 'j' is equal to the speed of 'k'.
    2. - (moving towards): if speed of 'j' is lesser than the speed of 'k'.
    3. + (moving away): if speed of 'j' is greater than the speed of 'k'.

Hence a valid $QTC_B$ relation between qualitative trajectories of two objects is represented as such '-,+,-'(for object 1 with respect to object 2) and '+,-,+' (for object two with respect to object 1). In each individual relationship the value in the first position is the relative position of the object in consideration, the second value is the relative position of the other object and the third value is the relative speed of the current object with respect to the other object. The benefit of having a relation that includes both relative position and speed comes to light when dealing with the movement of objects in higher dimensions (2D, 3D). For instance in 2D, the value '0'(stable) may be interpreted as either the distance being stable or the speed being stable, this confusion during interpretation is avoided by using unique values for both the relative distance and speed and hence the size of the resulting relations is a set of three. Interpretation of these relations is pretty straight-forward when dealing with movements one dimension. The $QTC_B$ being developed to deal with mobile objects when they are exclusively in DC (dis-connected) regions [46] makes it susceptible to inaccuracies when building relations regarding collision conditions.

The $QTC_C$ variation of the QTC calculus is often seen as an extension of the $QTC_B$ calculus, since it uses the same qualitative values to represent the qualitative trajectories of the objects. The only difference between the two versions comes from the integration of directional information which denotes the direction of movement of the current object in relation to a line segment between the two objects. This version of the QTC calculus was inspired in part by the 'Double cross calculus' [49], hence the name. The $QTC_C$ calculus is an improvement on the double cross calculus, as it can define qualitative trajectories and direction for single as well as multiple objects at once, whereas the double cross calculus can only deal with single objects at any given instance of time.

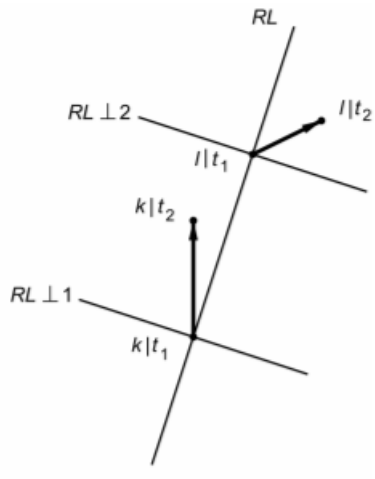



Figure 2.5: A graphical representation of the $QTC_C$ calculus for a relation between two moving objects'k' and 'l', where $t_1$ and $t_2$ are the two distinctive time instances and RL refers to the reference line and the double crosses($RL \perp 1$, $RL \perp 2$) for the two objects [45] .

The relations of the $QTC_C$ calculus, while following the same basic principles of the $QTC_B$, have been tweaked slightly to allow inferencing of the objects position in terms of the relative direction to the reference line. The assignment of the qualitative values to the object's movements can be illustrated using two objects 'k' and 'l', the possible values for 'l' at any given time interval are based on the following reasoning:

- based on movement relative to the respective perpendicular reference line:
    1. 0 (stable): if there is no change in the relative distance with respect to 'k' or with respect to the reference line.
    2. - (moving towards): if there is an increase in the relative distance between the objects.
    3. + (moving away): if there is an increase in the relative distance between the objects.
- based on movement relative to the directed perpendicular reference line(from 'l' to 'k'):

1. 0 (moving along the directed reference line): if the object moves along the line segment or if it was moving along the left/right and continues to do so in the given time interval.
2. - (moving to left side of the directed reference line): if the object moves from the right side of the line segment to the left side in the given time interval.
3. + (moving to right side of the directed reference line): if the object moves from the left side of the line segment to the right side in the given time interval.

Hence a valid $QTC_C$ relation between qualitative trajectories of two objects is represented as such '-,+,-,0'(for object 1 with respect to object 2) and '+,-,0,-'(for object 2 with respect to object 1). Wherein the value in the first position is the relative movement(towards-away) of the object in consideration, the second value is the relative movement(towards-away) of the other object, the third value is the relative direction(left-right) of movement of the first object and the fourth is the relative direction(left-right) of movement of the second object.

**Conclusion** To conclude, the $QTC_B$ calculi is a basic and restrictive representation that encodes only the motion of the object and not its direction of movement. Yet, this restriction allows for a more assured set of relations that do not encode any ambiguity corresponding to the relative motion of the objects in question [43]. The $QTC_C$ on the other hand does not provide distinctive relations to distinguish between the trajectories of a mobile object moving along the directed reference line and continuing to do so and a mobile object moving on either side of the directed reference line and continuing to do so [45]. Therefore leading to a rather ambiguous interpretation of the direction of the object's relative motion, factor into this the fact that we will implement it on a robot in a live environment and it becomes clear that there will be numerous cases of collision that could endanger the safety of the robot as well as the other agents that lie in its vicinity. Hence justifying the selection of $QTC_B$ for our implementation, not only is it safer and simpler but it also allows easier integration with other complimentary calculi such as ARGD, RCC which may be used to deal with possible collision cases.

### 2.2.2 Qualitative Rectilinear Projection calculus

The qualitative rectilinear projection calculus (QRPC) [26], [4], [15] presents an innovative approach to qualitatively representing motion patterns based on planar trajectories. In comparison to the QTC calculus, this representation has a richer description of the motion exhibited by mobile objects. The basis of this claim lies in the oriented rectilinear projection used to represent the trajectories, which allows this calculus to represent rotational motions of an mobile object, something that was not possible in the existing calculi such as the QTC. The QRPC is specifically tailored towards obstacle avoidance and geometric relationships between two objects are generated using the front-back and the left-right dichotomies. The relationships encompassed in the QRPC abstract the objects as points and use their rectilinear projections to make qualitative distinctions such as front-back and left-right, these atomic relations and their base notations are illustrated below:

1. **Notations:**

   - $P_iP_j$: is the relative disposition between the oriented rectilinear projection of the object $O_i$ and the oriented rectilinear projection of $O_j$.
   - $O_iO_j^{LR}$: The relative position of $O_i$ with respect to the left-right dichotomy of $O_j$.
   - $(CO_i{}^{FB})(CO_j{}^{FB})$: The relative position of the point of intersection('C') of the oriented rectilinear projections with respect to the frontback dichotomy of both objects.
   - $O_iO_j^{FB}$: The relative position of $O_i$ with respect to the front-back dichotomy of $O_j$ when the trajectories are superimposed.

     Note: 'i' and 'j' denote the two different objects.

2. **Qualitative relations (atomic) in each of the notations :**

   - The relative disposition between two oriented rectilinear projections($P_iP_j$):
     - $\uparrow\uparrow$ : Parellel in same direction
     - $\uparrow\downarrow$ : Parellel in opposite direction

  - $\uparrow$ : Coincident in same direction
  - $\updownarrow$ : Coincident in opposite direction
  - $X$ : Crossed/Intersecting rectilinear projections

- The relative position of $O_i$ with respect to the left-right dichotomy of $O_j$, $(O_i O_j^{LR})$:
  - '-' : if $O_i$ is on the left of $P_j$.
  - '0' : if $O_i$ is over $P_j$.
  - '+' : if $O_i$ is on the right of $P_j$.

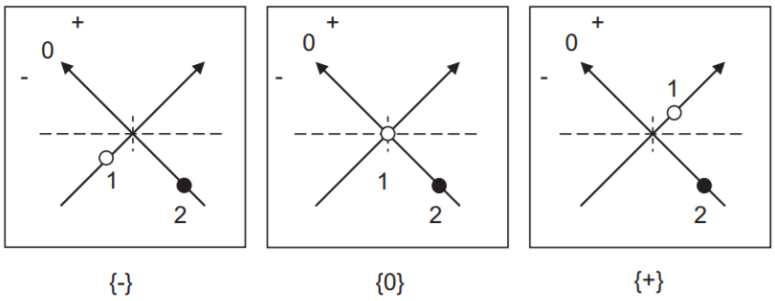



Figure 2.6: Relative position of $O_i$(object 1) w.r.t the left-right dichotomy of $O_j$(object 2) for crossed projections [26].

- The relative position of 'C' with respect to the front-back dichotomy of two objects $(CO_i^{FB})(CO_j^{FB})$:
  - $(+,+)$ : if 'C' is in front of $O_i$ and $O_j$.
  - $(0,+)$ : if 'C' is at same position as $O_i$ and in front of $O_j$.
  - $(+,-)$ : if 'C' is in front of $O_i$ and behind $O_j$.
  - $(0,-)$ : if 'C' is at same position as $O_i$ and behind $O_j$.
  - $(-,+)$ : if 'C' is behind $O_i$ and in front of $O_j$.
  - $(-,0)$ : if 'C' is behind $O_i$ and in same position as $O_j$.
  - $(-,-)$ : if 'C' is behind $O_i$ and $O_j$.
  - $(+,0)$ : if 'C' is in front of $O_i$ and at the same position as $O_j$.

  In cases where the two projections do not intersect with each other($\uparrow\uparrow$ or $\uparrow\downarrow$), this qualitative abstraction presents as conundrum

as the point of intersection 'C' may lie at either positive or negative infinity, hence in such cases the relation is abstracted to either $(+,+)$ or $(-,-)$.

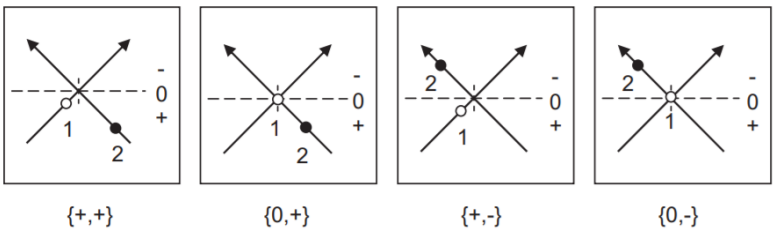
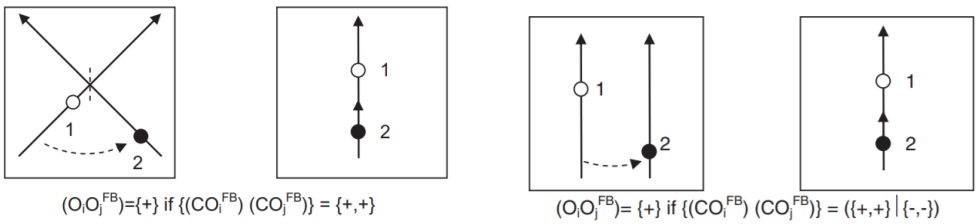



Figure 2.7: Some of the spatial configurations of the intersection point with respect to the front-back dichotomy of two objects for crossed projections $(CO_i^{FB})(CO_j^{FB})$, [26].

- The relative position of $O_i$ with respect to the front-back dichotomy of $O_j$ when the trajectories are superimposed$(O_iO_j^{FB})$:
  - '+' : if $O_i$ is in front of $O_j$ after $P_i$ is superimposed on $P_j$.
  - '0' : if $O_i$ is at the same position as $O_j$ after $P_i$ is superimposed on $P_j$.
  - '-' : if $O_i$ is behind $O_j$ after $P_i$ is superimposed on $P_j$.
    Note: The after the rectilinear projection of $O_i$ is rotated(such that the objects are always facing in opposing directions) along 'C' if it exists, before being superimposed on the rectilinear projection of $O_j$.




(a) The $(O_iO_j^{FB})$ feature for crossed projections. [26]

(b) The $(O_iO_j^{FB})$ feature for parellel projections. [26]

Figure 2.8: The point based and projection based direction representations [**?** ]

**Conslusion** Hence a complete valid QRPC relation comprising of the above mentioned atomic relations can be written as $[(P_iP_j)(O_iO_j^{LR})((CO_i{}^{FB})(CO_j{}^{FB}))(O_iO_j^{FB})]$, with the various values replacing their respective placeholders. Thus proving to be a richer representation of the movements of the mobile objects as it can effectively distinguish between front-back, left-right as well as same or opposing direction of heading. While this calculus does present a compact view of the movements of the objects, like its predecessors it still requires a sequence of qualitative(QRPC) states(individual sets of relations) to effectively represent the movement of an object. Besides, a completely theoretical formulation of the calculi specifically for GIS, limits its utility in robotics as there is no clear indication of the changes that need to be effected to ensure a reliable approach for mobile robot navigation. These drawbacks dissuade us from further pursuing this calculi for our purpose of achieving qualitative perception and control in mobile robots [26].

### 2.2.3 Qualitative Distance Calculus

The qualitative description of distance is based on three primary elements, a primary object, a reference object and a reference frame [11], [49]. When making a statement such as 'A is near B' the qualitative inference is ambiguous without taking into consideration spatial entities such as metric distance between the objects, their respective sizes and shapes, their relative positions, the positions of other objects as well as the frame of reference with respect to the objects itself. Furthermore the distance between two objects varies with perspective, for instance looking at a moving object from two different perspectives leads to two different interpretations of the distance to the object, with respect to observing entity. The distance calculi aims to summarize all these varying spatial quantities into a single coherent qualitative description which can be used to reason about space in terms of qualitative distances.

The ARGD or Delta calculus are some of the common names used for the qualitative distance calculus. These calculi partition the physical space into circular regions of varying granularity. Since there is no restriction on the number of granular divisions that can be applied to a given physical space we may define

this in a manner that is suitable for our application. Commonly the granularity varies from anywhere between 2-5, with the respective labels being very close, close, commensurate, far, and very far.

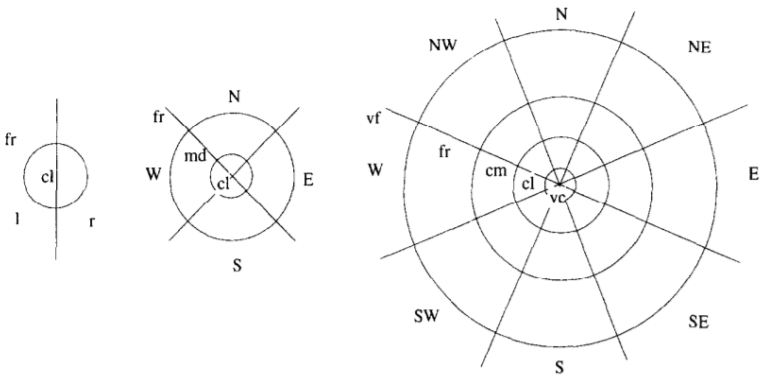



Figure 2.9: Various levels of distance and orientation distinctions. These figures show typical distance/orientation granularity configurations [11].

The various granular regions in the distance calculi are always centered around the object in consideration, these granularities are subject to an inherent 'greater than' order amongst themselves such that the granular region closest to the object has the lowest distance value and the one that is farthest has the highest order. Consider '$q_n$' as the granularity of the distance representation, then the various instances are organized as such $[q_0 < q_1 < q_2 < ... < q_n]$, with $q_0$ being the smallest distance to the object and $q_n$ being the largest. By using this comparative approach for distance magnitudes the calculus effectively maps the granular symbols to one dimensional geometric intervals that represent the distance ranges.

**Conclusion** Since the applicability of the distance approach does not depend upon the number of distance relations [11], it allows the distance calculus to be highly adaptive to the users demands and hence provides a feasible alternative to collision free navigation in mobile objects. Also as mentioned previously the distance representation effectively divides the physical regions around the object into geometric intervals, hence making this approach suitable to define distance thresholds around the robot to avoid collisions with other objects as well as define

a suitable safe region around the robot [31], which in-turn ensures safe movement in the physical space. These benefits of the qualitative distance calculi make it favorable to our application, hence warranting its utilization.

## 2.3 Implementations utilizing qualitative representations for navigation in robots

The goal of this section is to introduce the reader to the existing approaches that use qualitative spatial representations in some capacity to achieve efficient and collision free navigation in mobile robots. The merits and de-merits of each implementation is recognized and a brief summary is constructed based upon the same.

This approach [9], relies heavily on the detection of tangible features in two sets of images one of which is a reference image that is taken during a teach phase (when the robot is taken manually through the desired path and allowed to take reference images through out its path traversal) and a second image that is taken during a replay phase during which the robot is allowed to navigate the desired path by itself, capturing new images all along. The features in both these images are detected and tracked using the Kanade-Lucas-Tomasi [6] feature tracker. The tracked features in the current image (taken during the replay phase) are compared with the features observed in the reference image (taken during the teach phase) and the difference in their positions and distance is used to make qualitative decisions regarding the corrections that have to be made to the robot's path with respect to the desired path. The underlying spatial entity that is compared is the corresponding distance between the tracked features (in both the reference and the current image and the reference image) with respect to a common principal point of the images. Based on this conditions the decisions are made as follows:

- if the robot is moving from point A to B without deviating from the original path, then continue the current path until some deviation occurs.

- if the robot deviates laterally from the desired path and this is indicated in the signed distances between the features of the current and reference image, then:

- if the distance between the reference feature lying on the left/right of the image and the principle point is more than the distance between the current feature lying on the left/right of the image and the principle point then move left or right respectively.
- if the signed distance of the feature lying on the left/right of the image does not match with the sign of the feature lying on the left/right of the reference image then move right or left respectively.

**Conclusion** The merits of such an feature based approach [9], seem to be apparent, yet this approach does not utilize any of the traditionally accepted qualitative calculi. Furthermore its reliance on a teaching phase renders it invalid in cases where prior information about the environment is denied to the robot. The dependence on further factors such as lighting conditions, detection of features and inability to work efficiently in an outdoor environment(due to lack of contrast, the features are not detected) show that this is a highly constrained approach that can work only if a large amount of prerequisites are satisfied. The merit of this approach is still its ability to function exceptionally well when the teach phase is executed properly, with the robot able to follow trajectories upto a 100 meters easily. The authors also mention the apparent improvement in efficiency(in terms of reduced cost and power consumption as well as increased robustness) when using such qualitative approaches, but fail to provide empirical proof for such statements.

The approach presented in [10], also utilizes detection of tangible features to extract pixel coordinates of these features and then compare these coordinates with the corresponding pixel coordinates of the feature in the reference frame. The resulting differences are then used to make decisions regarding the control of the robots trajectory, as previously introduced this approach also makes use of the teach-replay method. This paper extends the approach presented in [9] by combining the pixel coordinates with odometry information to achieve a more robust navigation, also the use of the 'funnel lane' methodology is a novel concept that helps in localizing the robot. The underlying spatial entity used in this approach is the relative positions of the features with respect to the principal point of the image, which yields signed (x,y)co-ordinate values. The distance between

these reference points and the principal points is used to define a region called the 'funnel lane', while the robot lies in this region it is allowed to move directly towards the goal position, but if there are any violations such that the robot moves out of this 'funnel lane' then the controller updates the control commands to ensure the robot re-enters the 'funnel lane'. Since the 'funnel lane' is oriented such that its origin is always at the goal position hence ensuring that the robot zeros in on the goal position.

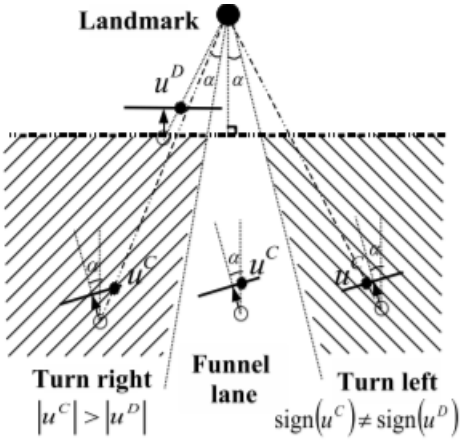



Figure 2.10: The funnel lane created by the two constraints(distance from the principal point and the relative signs of the distances), and when it has turned by an angle $\alpha$, here $\mu^C$ and $\mu^D$ represent the horizontal pixel coordinates of the reference and the current frame [9].

**Conclusion** Being an extension of the approach specified in [9], this implementation [10], suffers from very much the same over-reliance on the teach phase to correctly capture the reference features, while completely failing to utilize any of the traditionally accepted qualitative calculi. Furthermore the authors fail to indicate the maximum number of features that can be considered when constructing this 'funnel lane' and do not provide details about what happens when no feature is detected, making this a promising yet incomplete implementation as not all details are disclosed. The implementation is more robust thanks to the introduction of the funnel lane concept and utilization of odometry information which enable the robot to follow trajectories of upto 380m and at a speed of 1m/s. Again the authors mention the improvement in efficiency in terms of power consumption and

computational requirements yet they fail to provide any conclusive evidence to validate these claims.

The approach followed in [35], is also based on a teach-replay method, but instead of using RGB cameras and images to extract and compare features it uses a sonar array to find the distances from the obstacle. Since this implementation is deployed on a wheelchair in a indoor corridor environment, the sonar array is used to measure the distance from the wall. The primary use of qualitative representations is for human robot interaction and is not used for the navigation task as a suitable update rate for the qualitative representations was not possible. Hence although the use of qualitative calculi based on the directional and distance representations was hypothesized it was not put to test in a factual manner. As mentioned above the proposed approach used a sonar array and dead reckoning to record desired paths and then autonomously replay them. The base assumption being that any path in an indoor environment is basically a combination of following walls and making turns at the corners based on this assumption paths were described as a sequence of distances and angles with specific triggers wherever a corner or an immovable object was encountered so that a suitable maneuver may be performed. Being a teach and replay approach this implementation also compared data obtained during the replay stage with the reference data obtained during the teach stage and made decisions based on the differences so that the wheelchair follows a path that is as close as possible to the original path as possible.

```
<close forward fast>
<close right medium-speed>
<close backward slow>
<very-close left medium-speed>
```

Figure 2.11: The hypothesized qualitative representation with the left most value being the distance, followed by the direction to move in and finally the speed with which to move [35].

**Conclusion** Since this [35], is another teach-replay approach its major drawback is the over-reliance on the teach phase, furthermore being a teach replay approach

that utilizes sonar distances, angles and dead reckoning to obtain trajectories, the system cannot deal with dynamic objects such as humans that may be encountered during autonomous replay as these objects did not exist during the teach phase and hence no information about them exists in the systems knowledge of the environment. The authors also fail to provide concrete information regarding how the control decisions are made instead choosing to only mention that this is done based on the differences in the represented trajectories. One clear merit of this implementation was the hypothesis of how qualitative distance and direction calculi can be combined to achieve navigation in robots, but this is also its drawback as this was only hypothesized and not tested or implemented on the mobile platform. Again the authors assume better efficiency in terms of power consumption and computational requirements due to the usage of qualitative representations yet they fail to provide any conclusive evidence as validation.

The approach presented in [39] makes use of a unique qualitative map based representation, which is inherently based on distance relations. The authors propose the 'Qmap' framework which accepts as an input a line segment representing the desired path. This line segment is then quantized into rectangular cells that form a part of the grid. Since the robot is equipped with a sensor array of 8 sonars each placed at 45 degrees (in a manner similar to the layout of the cardinal directions) and orientation sensors the data received from each of these sensors is stored in a qualitative manner within the cells of the grid. Sensors that are in close proximity of obstacle are marked as active and their respective relations are indicated using either of the three distance primitives [I(increasing), D(decreasing), S(stable)], this is the qualitative data that is stored in each cell during the off-line planning phase. During the run-time the implementation takes as input the generated 'Qmap' and the starting and ending positions. Additionally, data stored in each cell is compared with the real-time data and the control decisions are made based on the differences between the two. In cases where the difference between the reference data (simulated during the planning stage ) and the real-time data is large such that can be caused by the presence of moving obstacles the preference in the control algorithm is given to the real time data such that it ensures that the robot stops moving completely until further intervention by a human controller.

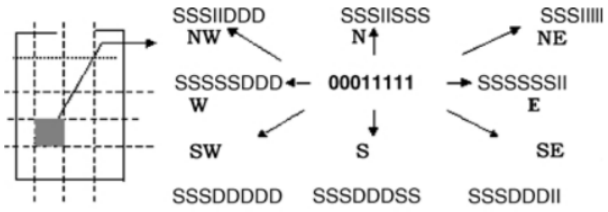
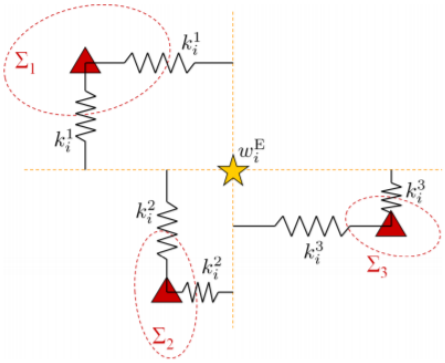



Figure 2.12: figure showing the 'Qmap' for a given line segment, the 0's and 1's indicate the inactive and active sensors while 'S', 'I', 'D' indicate the distance relations of each sensor w.r.t the obstacle(grey cell) [39].

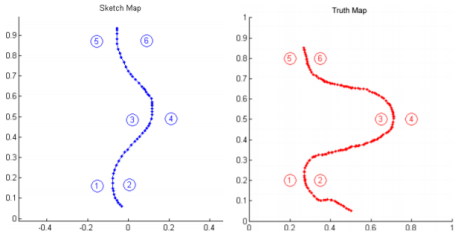
**Conclusion** The authors mention that the implementation [39], deals with obstacles by marking those respective cells as blocked, but do not indicate how or when (planning or run-time) this information is provided to the map also the stopping of the robot when dealing with moving objects is an obvious drawback when considering real world applications. The authors mention that a successful navigation was achieved in 60 percent of the cases out of a possible 15 experimental runs, with the causes of failure being the large size of the robot which in turn caused collisions when turning, the accumulation of errors in the orientation data due to inaccurate orientation sensors and errors in localizing the robot position on the 'Qmap' due to increasing uncertainty, especially when moving in large open spaces. Yet again the authors mention the efficiency based benefits of the qualitative approach but do not provide tangible proof for it and although this is a promising qualitative representation in relation to a global map of the environment it still does not use any of the traditionally accepted qualitative calculi.

This implementation [40], also provides reasonable cause to prove the utility of qualitative representations in global planners. It uses as an input a hand-drawn map containing the desired path to be followed and a set of landmarks that can be used as a reference. This qualitative map does not have to be accurate but should be able to represent the approximate positions of the landmarks in the physical space. The planning module used in this approach utilizes this approximate waypoint positions and the corresponding landmark positions to plan a valid path that can be further fed to a path following module which in-turn executes it on a low

level. The path planning is modeled as a quadratic optimization problem, wherein each way-point is influenced by its surrounding landmarks, such that the closest landmark has a higher bearing on the placement of the way-point rather than a landmark that is farther away. The uncertainty in the position of the landmark is also taken into account when placing a way-point in the final trajectory that is to be executed. Furthermore when planning the final path the relative qualitative relations between the robot and the landmarks are maintained which makes it easier to localize the robot based on the available landmarks and in-turn reduce the uncertainty in the positions of the landmarks themselves. Additionally the EKF-SLAM algorithm is used to make the localization and way-point planning more robust.




(a) Landmarks (shown as red triangles) represented with linear springs to the proposed waypoint location (shown as a star). For each landmark, the spring constant is calculated as a function of the distance between waypoint and landmark, and the uncertainty in the estimated location of the landmark (represented by red ellipses) [41].




(b) The sketched map (blue, on left) includes six landmarks and a trajectory that slaloms between pairs of landmarks. In the true environment (red, on right), the robot must slalom more aggressively to pass between the landmark pairs [40].

**Conclusion** This implementation [40], represents a more reliable approach to the utilization of qualitative representations in a global planning based approach. The preservations of qualitative relations between the way-points and the landmarks combined with a SLAM implementation provide a robust platform on which the robot can securely navigate a given environment. In its current state this approach lacks the knowledge of obstacle avoidance for both stationary and mobile objects,

also the authors do not elaborate on how the robot knows which observed landmark is to be associated with which reference position in the given map, although during implementation on a robot this problem was solved using a overhead camera this does not suggest a feasible real-time implementation. Furthermore the authors suggest the advantage in efficiency that qualitative approaches have over quantitative approaches but do little to prove this as the truth.

This implementation [5], features the use of the qualitative trajectory calculus to create abstractions of human movements and then use these abstractions to reason and execute the corresponding motions of the robot. $QTC_B$ is the selected calculi for creating these abstractions, since it takes into account the relative motion between two mobile objects. The abstractions have been described in detail in the previous section of 'Qualitative trajectory calculus'. Using these previously described abstractions the relative motion between the robot's and the human trajectory is encoded as a qualitative representation. A graph structure is used to build a situation graph tree where each node represents a particular scenario or agent's states and the edges indicate the transition to the next possible states. A depth first approach is used to traverse the whole tree and a fuzzy-logic based interpreter(F-Limette) is used to interpret the states and convert them to a low-level motion command. For the sake of simplicity the authors have only implemented this approach in a restricted limited manner, wherein the robot either follows the human or stops following it based on the described qualitative scenarios.

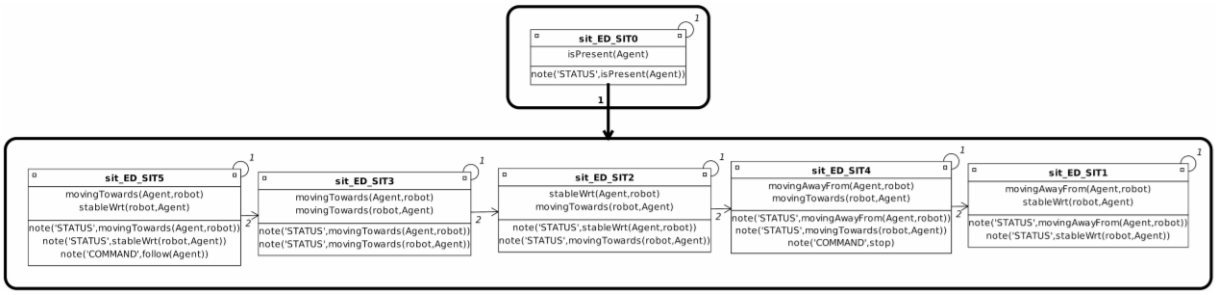



Figure 2.13: Situation Graph Tree of the human motion behaviour and relative robot actions [5].explain the state and transitions

**Conclusion** The above mentioned approach [5], was successfully implemented on the robot platform using laser range finders to track human feet, and create suitable abstractions. Yet it presents only a limited view on the applicability of qualitative representations for robot navigation, furthermore when using such graph structures the space and time complexity increases exponentially with increase in the number of scenarios. Thus contradicting the base assumption that using qualitative representations allow improvement to the computational efficiency for a given task. The authors do suggest the improvement of the current approach by replacing the $QTC_B$ representation with a $QTC_C$ representation that allows a finer control paradigm since it includes directional information(left-right) to be represented in addition to the nature of the existing motion(moving away-towards). Although this implementation shows the utilization of the QTC calculi, it cannot be considered a robust implementation due to the lack of extensive testing and the limited experimentation that was carried out for this purpose.

This approach [16], makes use of the qualitative trajectory calculus to depict relative motion between a human and a robot, in a restricted environment which in this case is a corridor. The use of qualitative representations to achieve robot trajectories that are perceived as safe and intuitive by humans is the main aim of this project and to that effect the 'pass-by'(human and robot cross each other in opposite directions) scenarios are evaluated. The authors use 'velocity costmaps based on the qualitative descriptions to limit the sample space of the dynamic window approach local planner to generate trajectories' [16], that are safe for the robot and human as well as intuitive for the human. The authors combine the $QTC_B$ and $QTC_C$ calculus into a single calculus called the $QTC_{BC}$ which allows the representation to switch between the $QTC_B$ and $QTC_C$ depending upon a distance threshold. This distance threshold is defined such that when the robot is in close proximity to the human agent the representation switches to $QTC_C$ as this provides a more informed representation of the relative trajectories at farther distances since it is sufficient to know whether the two objects are approaching each other or not, the $QTC_B$ representation is used.

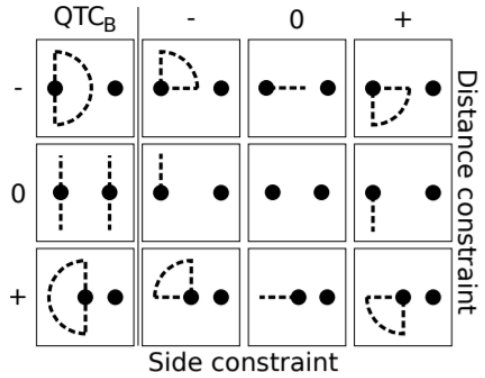



Figure 2.14: The velocity costmap prototypes. The area enclosed by the partial circle represents the low cost area, everything outside is assigned the highest cost value. The black dot on the right represents a human that can have any possible QTC state (except for $QTC_B$) [16].

**Conclusion** The approach [16], was tested by both simulation and application on a robot and the qualitative representation velocity costmaps far outperformed (in terms of minimum number of collisions) the costmaps that utilized Cartesian coordinates as well as a naive dynamic window approach. This approach provides a concrete argument to the utility of qualitative representations in robot navigation as well as human robot interaction, yet it does not clarify the claim stating that the use of qualitative representations is more efficient when compared to a quantitative representation. Also, its evaluation on a single test scenario makes it likely that this approach is highly tuned to achieve greater success on the described task with no indication of its performance on a more generalized task such as navigation through a corridor with or without multiple obstacles.

## 2.4 Limitations of previous work

The previous section brings to light a variety of implementations that utilize qualitative spatial representations in varying capacity to abstract spatial entities and relate them in a semantically cohesive manner. The concern that's highlighted immediately is that many of these approaches do not utilize the traditionally accepted qualitative calculi(based on a set of relations, which have been abstracted from underlying mathematical theories and offer a reliable method to describe spatial configurations [32].) and hence often end up with provisional implementations

that have been developed with reliance on a set of highly specific parameters, which if varied will most likely cause the developed algorithm to fail the given task. Thus showing that there is a lack of a generalized approach to the application of qualitative spatial representations in the domain of mobile robot navigation.

Furthermore all the approaches state the efficiency gain especially in terms of computation and power consumption as the motivation behind using qualitative spatial representations over quantitative ones, but none of them provide tangible proofs to establish the truth of this statement. There exists a lack of effort on this front as there are no comparative studies nor any empirical evidence to support this seemingly obvious claim.

Thus for the purpose of this project we will aim to address the above mentioned limitations, in the proposed approach.

# 3

# Approach

The conclusions drawn in the previous chapter, allow us the liberty of focusing specifically on the two most promising qualitative calculi $QTC_B$ and $ARGD$, for the purpose of developing an algorithm that centers around the utilization of either one or both of them, for the purpose of qualitatively representing physical space in our application of mobile robot navigation. Since the developed approach utilizes the QSRLib to implement the $QTC_B$ and $ARGD$ calculi further details on it will be provided in this chapter, furthermore as the approach can be broken down into several subtasks this chapter is structured in the same manner as these subtasks.

## 3.1 Approach

**Overview** The developed approach is a python implementation in the ROS ecosystem and can be broken down into three sub tasks or modules namely the qualitative perception module, the qualitative relation module and the qualitative control module. Each of these three modules deals with a crucial aspect of navigating using qualitative spatial representations. The perception module uses the RGB video obtained from the camera to detect Aruco markers and decipher their Cartesian coordinates, this data is then forwarded to the qualitative relation module which utilizes it to build qualitative relations of these detected features(markers) with respect to the position of the robot. To build these relations an open source project called the QSRLib is used, this is a library containing python implementations of various qualitative calculi, including the $QTC_B$ calculus and

the *ARGD* calculus. The nature of these relations and the information encoded in them depends upon the calculus used, if the $QTC_B$ calculus is used then the relative trajectory information between the two objects is encoded, whereas if the *ARGD* calculus is used then the relative distance information between the two objects is encoded. As we use a minimum of two markers corresponding markers at any given position on the opposing walls, the qualitative relation module builds the relationships only if both these markers have been detected at similar timestamps. The output from the QSRLib is parsed in a manner such that it will be easier to infer and evaluate in the qualitative control module. The qualitative control module receives the relations of each individual object with respect to the robot these relations are then combined together to better understand the whole picture with regards to the relative movement of the robot in the environment(w.r.t the markers). Once a complete and combined relation is obtained it is analyzed further to decide upon a possible motion command for the robot to continue moving smoothly without any collisions with the walls. These motion commands are then sent to a low level controller, which converts them into a machine readable form and uses this to control the motors and achieve the desired movement.

### 3.1.1 Perception

In mobile robots perception allows the robot to 'see' it's surroundings and safely interact with them, in our developed approach we use a RGB camera to achieve this result. Although other sensors such as range or stereo may be used we use the camera as this provides a more intuitive visual representation of the environment. The perception module implemented in this approach uses RGB video data, captured by a camera mounted on top of the KUKA youbot arm. This video data is assessed in real time to detect ARUCO markers that have been placed beforehand at various positions along the walls of the environment. The use of Aruco markers is warranted by two reasons, primarily this allows the simplification of the task of detecting robust features in the environment, since now the only features we are looking for are the markers. Furthermore these markers make it easy to extract their respective Cartesian coordinates, which is a necessary piece of data that is required by the QSRLib to build qualitative relationships between the

robot and the spatial entities(markers represent abstractions of objects/features on the walls). From a conceptual point of view this allows us to simplify the subtask of perception as we no longer need to detect natural features from the environment and can focus instead on the primary task of creating the qualitative relations and using them effectively.

### 3.1.2 Representation

In the representation module we basically build qualitative relationships between the markers and the robot based on the relative motion between the two. The relations are further used in the control module to infer the robot's allocentric spatial transform(trajectory) and make corresponding control decisions to it's motion. As mentioned previously we do not implement any of the discussed calculi from scratch instead choosing to utilize a library containing python based implementations of each. The implementation uses to separate nodes to work with the $QTC_B$ calculus and the $ARGD$ calculus individually. The data received from the perception module is preprocessed in a similar fashion for both the calculi, first the data is filtered based on the timestamps that both the markers are detected at approximately the same time, then a distinction is made between the marker poses to determine the ones that lie on the left of the robot and ones that lie on the right. Once this is done we begin converting this raw data into the required format for the QSRLib, wherein we associate each marker with a distinctive object name, a common timestamp for all the objects in the relation(the robot, the left marker, and the right marker) and the respective pose of each of the markers. This data is encoded in the form of a time-series of distinctive states for each of the markers and the robot. After this preprocessing the data is finally ready to be packaged into a custom QSRLib request message that contains the selected id of the qualitative calculi in addition to the preprocessed data. Since the QSRLib uses a ROS service-client architecture the computation of the relationships is extremely fast and does not cause any delay in the overall approach.

Upon receiving a response from the QSRLib server the message is sorted for each relation based on the timestamps and then modified to make it easy to read and

understand for a human operator. This modified data allows us to manually verify if the built relations actually reflect the state of the robot and it's environment accurately for that instance of time. A custom topic and message is used to share the data with the control module.

### 3.1.3 Control

The data obtained from the representation module is utilized to make control decisions regarding the robot's movement at that particular instance of time. Since both the qualitative calculi present a different set of relations we use two separate nodes for analyzing the relations obtained from each individual calculi. Each relation is broken down into two parts one is the relationship itself and the other is the set of objects for which the relationship holds, once this is achieved the relationships are passed through a set of if-else rule-sets to find one that matches and the respective control command is issued to the low level controller which in turn controls the motors to move the robot accordingly.

## 3.2 Implementation details

As mentioned previously the entire approach is implemented using python 3 and the ROS environment. The general publisher-subscriber approach is used for all the developed nodes and custom messages and topics are used to exchange data amongst the nodes. The same basic approach is used for both the $QTCB$ and the $ARGD$ calculi wherein the video data is first analyzed in the perception node, the results of which are then shared with the representation node via a common topic using custom messages. The representation filters out unnecessary information using a time filter to ensure that only opposing markers(we assume that there are atleast two markers in the environment ) that were detected at almost similar time instances are used when building the qualitative relations(using the QSRLib) with the robot. These relations are then shared with the control module again via custom messages and a common topic. Once the data has been received by the control node, it sorts out the data so that it becomes easy to run it through a set of if-else conditions that act as rulesets which decide how the robot should move based on what it is seeing. Once a motion update from the rulesets is obtained

the result is published on a 'cmd_vel' topic that is being listened to by a low level controller which then translates it into low level motion commands to the motors. While there are separate nodes for representation and control for both the calculi the perception node is shared among the two, since the same set of data is used to build qualitative relationships amongst the objects irrespective of the calculi used. Finally the developed code is stored in the form of a package named as 'qsr_nav'.

### 3.2.1 QSRLib

**Input Data Structure** To utilize the full capability of the QSRlib [1], we need to convert any raw data into the standard input format('World_Trace' object instance) that is accepted by this library. This 'World_Trace' [1] contains various inbuilt methods that are used to convert the raw data into this format. The primary component of 'World_Trace' is a python dictionary called 'trace', this dictionary uses the timestamps as it's keys while the object instances of the 'World_Trace' class represent the values in this dictionary. Any 'World_State' object instance encodes the spatial objects as it's primary member, this object instance is structured as a dictionary. The keys of this dictionary are represented by unique names of the spatial objects while the values are represented as object instances of the 'Object_State' [1] class. An object instance of the 'Object_State' contains data regarding spatial objects at any particular instance of time.

**Output Data Structure** Like the standard format for input data, the output of the QSRLib also follows a standard output structure, this data structure is an object instance of the type 'World_QSR_Trace' [1]. Similar to the input data structure the primary element of this data structure is also 'trace', implemented in the form of a dictionary whose keys are the timestamps of the QSRs. These keys match the timestamps of the object instance of 'World_Trace'. The values of this dictionary are the object instances the 'World_QSR_State' class. The 'World_QSR_State' object instance uses the 'qsrs' as it's primary element, such that the 'qsrs' is a dictionary which encodes unique combinations of the spatial objects as it's keys and object instances of the QSR class as it's values.

### 3.2.2 Perception

The perception module is built into a single ROS node using the python 3 language and utilizes various libraries to process the video data received from the camera. The video data is published on the 'arm_cam3d/rgb/image_raw' topic in the ROS image format, to utilize this data for detection of the markers we convert it into OpenCV images format using the inbuilt ROS 'cv_bridge' package. To generate and detect the markers we use an inbuilt dictionary from the OpenCV aruco module ('cv2.aruco'), which contains 250 different markers each of the size '6X6'(36 bits). Once the raw data is converted into an OpeCV image format, we can run the 'aruco.detectMarkers()' method to detect markers and extract their respective marker_id, along with their pose(using 'aruco.estimatePoseSingleMarkers()' method) and the time of their detection. Once a marker is detected it's pose in the real world is estimated with the help of the camera matrix (allows the us to map 3d points to the 2D image and vice-versa). The units used for the pose of the markers depends upon the units used to depict the length of one side of the marker, as we have specified this in meters the final pose of the markers is also in meters.

Since we use atleast two markers at corresponding positions on the opposite walls we needed a method to ensure that all the data from both the markers is published at all times and that there is no mix-up of the data, hence to achieve this we make use of two separate publishers for both the markers and to ensure that there is no mix-up we bind each publisher to a single 'marker_id' by using a clever combination of flag variables and if-else conditions. To enable cross checking the marker's ID number is stored until the robot has moved past that particular set of markers. Since we use two publishers we also use two individual topics('extracted_pose1' and 'extracted_pose2') to which these publishers publish their respective marker information. Finally custom messages are used to publish the data to the topics, this message encodes the the time-stamp of the marker detection, the marker id number and a 3D pose vector that contains the Cartesian coordinates for each marker.

### 3.2.3 Representation

Like the perception module the representation module is also implemented using ROS and python 3 within a single node, while we use two different nodes for the two calculi the base approach for both remains the same(as discussed earlier).

**QTCB Implementation :** The marker data published to the topics 'extracted_pose1' and 'extracted_pose2' serves as the raw data for this node. The data is filtered using the 'ApproximateTimeSynchronizer()' method from the ROS 'message_filters' module, this filtering ensures that only marker data that is atmost 0.1 seconds apart will be used for further processing. Upon filtering the data we obtain the marker information for atleast two markers that have been detected at similar instants of time thus ensuring that we always deal with the most recent data. Furthermore the pose vector of both these markers is analyzed to decide which marker lies to the left and which one lies to the right, this is done by checking the values of their X-coordinates(if negative it lies on the left otherwise right) the respective marker ids are then stored as part of the custom message(QsrMsg) that will be published on the 'qtcb_realtionship' topic from where it is picked up by the control module and used as a reference during the decision making process. Since we are using the QSRLib to obtain the $QTC_B$ relations we need to first convert the raw data into the format utilized by the library. To do so we utilize the 'World_Trace' data structure provided by the library, which incidentally also happens to be the primary form to which the raw data needs to be converted.

To begin with we create the 'Object_State' tuple that contains the Cartesian coordinates of the markers, the distinct object name for the marker, a common time stamp for when the marker was detected. For the $QTC_B$ calculus we need atleast two instances of each individual object at consecutive instances of time, as the $QTC_B$ checks the variation in the object's position to determine if the object has moved(further inference is then used to qualitatively represent the objects motion) or not. The next step is to wrap the defined 'Object_State's' in a list and store them in unique variables, this was done to enhance the readability of the code and also because it makes it easier to add them as individual entities to

the 'World_Trace' data structure. The Cartesian coordinates for all the objects and the robot are initialized as (0,0) and get updated when new data is available to the node, for the sake of simplicity the Cartesian coordinates of the robot are always fixed at (0,0) this does not affect the overall functionality of the algorithm in any way and can be physically interpreted as egocentric motion of the robot with respect to it's environment. Finally after defining this time-series of the object's states we can finally add this data to the object of type 'World_Trace' using the inbuilt 'add_object_state_series()' method (this takes as argument the object list of time-series data that was defined earlier). After adding all the 'Object_States' to the 'World_Trace' object instance we can finally wrap it up in the form of a 'QSRlib_Request_Message' which includes the converted data in the from of a 'World_Trace' object instance along with desired calculi (selected from a list of available calculi in the QSRLib) and any runtime arguments that are specific to the calculi in consideration. These runtime arguments are called as 'dynamic_args' and are unique for each calculi. In the case of our implementation of the $QTC_B$ these 'dynamic_args' are the 'no_collapse','quantisation_factor', the no collapse tells the library to avoid collapsing the output amongst timestamps if there is no change in the relative movement/trajectory of the objects, whereas the 'quantisation_factor' represents the minimum change in position that should be considered to assume that the object has moved(0.01 in our case). The completed 'QSRlib_Request_Message' is then sent to the 'qsrlib_ros_server' to obtain the qualitative representation of the relative trajectories of the objects.

Since the relations obtained from the 'qsrlib_ros_server' are mutually exhaustive the relations, complimentary repeated relations exist for the given set of spatial objects this excessive data is often unnecessary, therefor we also specify beforehand the specific set of spatial objects for which we want the qualitative relations, using the 'dynamic_args'. The response obtained from the 'qsrlib_ros_server' is sorted according to the timestamps and looped through using to obtain the objects in the relationships and the respective relations themselves. Once this is done the data is put into a custom message that contains the message time, the marker id's, the id's of the left and right markers, the objects in a relationship and the relationship itself. After filling up the message with all this data it is published

to the 'qtcb_relationship' from where it can be picked up by the next node and utilized for making motion updates to the robot's movement.

**ARGD Implementation :** The implementation of the *ARGD* calculus is the similar to the one described above for $QTC_B$ calculus, with the only difference occurring in specific 'dynamic_args' used for this calculi. Specifically these are 'qsr_relations_and_values' which define the distance thresholds and the corresponding relationship for these thresholds, for our use case we define the thresholds as follows ''Near': 3.0, 'Medium': 3.5 , 'Far': 4.0' in the form of an immutable dictionary. All the values specified for the thresholds are in meters and relate to each other using the 'less than(<)' relational operator(unlike what is mentioned in [25]), hence any distance value below three meters is considered as 'Near' while any distance value above four meters is considered 'Far', for values that lie between any of the defined threshold the assigned relation is always that of the lower threshold, for instance if the distance value is '3.7' meters then the corresponding relation will be 'Medium'.

Another difference from the $QTC_B$ implementation is that for the *ARGD* we use only a single instance of the 'Object_State' tuple for each individual object, this is because unlike the $QTC_B$ calculus which uses distance as a means to approximate the objects trajectory, the *ARGD* uses the distance as the absolute measure of the objects position with respect to other spatial entities, no trajectory or path information is abstracted from the distance data. The reason for this lies in the definition of the distance calculus which states that this calculi is used to qualitatively abstract distances between static objects, hence eliminating the need for any sort of temporally augmented trajectory data. As mentioned in the previous section(QTCB Implementation), the data from this node is published using the custom message 'QsrMsg' to the 'argd_relationship' topic. The composition of this message varies slightly from the one presented in the 'QTCB Implementation', while the 'QsrMsg' for the $QTC_B$ contains the relations between the robot and any one marker at any given instance of time the 'QsrMsg' for the *ARGD* contains the relations between the robot and any both the markers at any given instance of time. The reasons for this will be explained further in the next section of this chapter.

### 3.2.4 Control

The control module also uses two different nodes to deal individually with the $QTC_B$ and $ARGD$ relations, this structure promotes modularity and makes it easier to combine the two pipelines if the need ever arises. Furthermore the control module is divided into two ROS nodes one for interpreting the relations and the other for converting these motion updates into numerical velocity values that can be used to control the robot.

#### QTCB Controller

The control module is implemented as a set of if-else rules that compare the relations obtained from the representation module and present a possible control command that is to be applied to the robot's motion. After obtaining the relations on the 'qtcb_relationship' topic the data is preprocessed using python string operators such as split() and strip() to obtain the individual sets of objects(eg:(object1,robot)) and their corresponding relationships again the relationships are checked to verify that they do actually belong to the $QTC_B$ calculi, since each relationship has this encoded in the form of the calculi name a simple string check suffices for this purpose. Once the relations have been cleaned we use simple if-else conditions to check for possible matches from a set of possible relations, this is feasible as there exists a finite set of relations that are can exist between the two objects, also since the relations between the robot and the two markers are mutually exhaustive it suffices to check the relation of each marker with the robot individually as we know that the corresponding relationship with the other marker is the inverse of the relationship between the current marker and the robot. Furthermore this trick allows us to compare the relationships amongst each set of objects individually and reduces the number of possible states that have to be compared, for a relation between two individual objects there are 9 possible relationships, hence there are 9 possible relations for each object set, now if we decide to combine two object sets then we have a set of $81(9^2)$ possible relationships for only one order of combination of these object sets, if the order is now reversed then we get another set of 81 possible relationships, hence to make this a set of mutually exhaustive relations(for

only two object sets) we have to go through a set of $162(2X81)$ possible relations to obtain one possible update for the robot's motion thus making this an unnecessary complication that can be avoided. Also this number rises exponentially if the number of objects or markers is increased as the object sets are possible combinations of these individual markers this adds an unnecessary dependency on the number of markers. Therefore to avoid such complications we compare the relations for each object set individually and use the simple information of the relative location(left or right) of the object with respect to the robot to decide upon the motion update for the robot. Thus resulting in a simple yet effective control module.

Additionally $\{-, +, 0\} = [towards, away, stable]$, are the actual interpretations of the $QTC_B$ relations when the robot has allocentric spatial transforms with respect to the environment, but in our case since the robot is considered to have egocentric spatial transforms with the environment the relations are inverted. The following code snippet illustrates the rule-sets used and their corresponding motion updates for a robot tasked with moving forward along a corridor, the identifiers 'rel1' and 'rel2' indicate the individual relationships of each object, such that 'rel2' is always reserved for the robot.

```
if relation == "'0,0'":
rel1 = 'Stable'
rel2 = 'Stable'
vel_cmd = 'Move Forward'

if relation == "'0,-'":
rel1 = 'Stable'
rel2 = 'Away'
if obj1 == object_left:
vel_cmd = 'Move Forward'
else:
vel_cmd = 'Move Forward'

if relation == "'0,+'":
```

```
rel1 = 'Stable'
rel2 = 'Towards'
if obj1 == object_left:
vel_cmd = 'Move Right'
else:
vel_cmd = 'Move Left'

if relation == "'+,0'":
rel1 = 'Towards'
rel2 = 'Stable'
if obj1 == object_left:
vel_cmd = 'Move Right'
else:
vel_cmd = 'Move Left'

if relation == "'-,0'":
rel1 = 'Away'
rel2 = 'Stable'
if obj1 == object_left:
vel_cmd = 'Move Forward'
else:
vel_cmd = 'Move Forward'

if relation == "'-,-'":
rel1 = 'Away'
rel2 = 'Away'
vel_cmd = 'Move Forward'

if relation == "'-,+'":
rel1 = 'Away'
rel2 = 'Towards'
if obj1 == object_left:
vel_cmd = 'Move Left'
```

```
else:
vel_cmd = 'Move Right'

if relation == "'+,+'":
rel1 = 'Towards'
rel2 = 'Towards'
vel_cmd = "Move Forward"

if relation == "'+,-'":
rel1 = 'Towards'
rel2 = 'Away'
if obj1 == object_left:
vel_cmd = 'Move Right'
else:
vel_cmd = 'Move Left'
```

### ARGD Controller

Similar to the 'QTCB Controller' the control module for the *ARGD* also uses if-else rule-sets to iterate through a list of possible relations and then select a motion update when a match is found, unlike the $QTC_B$ though we do not compare the relations of each object set individually as the number of relations for each object set is very small(3) and here the combinations of the object sets is necessary to be able to distinguish between different states of the robot with respect to the objects or markers, again the implementation of this rule-set ensures that it is independent of the number of markers and their placements in the environment(as can be seen from the experiments).

Since in this case we use a combination of the relations of the object sets, the data obtained from the 'argd_relationship' topic is processed slightly differently from the manner shown in the 'QTCB Controller' section. Here we apply the python string operators 'split()' and 'strip()' to breakdown the breakdown the raw relationship data into parts so that it is easy to associate them with their

respective object sets. Once this is done we use the if-else conditions to iterate over the various possible combinations of these individual relations to obtain a suitable motion update. A code-snippet of rule-sets and their respective motion updates for a robot moving forward along a corridor is shown below, the identifiers 'rel1' and 'rel2' indicate the individual relationships of each object set.

```
if (rel1 == "'Near'" and rel2 == "'Near'"):
vel_cmd = 'Move Forward'
if (rel1 == "'Near'" and rel2 == "'Medium'"):
vel_cmd = 'Move Right Slowly'
if (rel1 == "'Near'" and rel2 == "'Far'"):
vel_cmd = 'Move Right Quickly'
if (rel1 == "'Medium'" and rel2 == "'Near'"):
vel_cmd = 'Move Left Slowly'
if (rel1 == "'Medium'" and rel2 == "'Medium'"):
vel_cmd = 'Move Forward'
if (rel1 == "'Medium'" and rel2 == "'Far'"):
vel_cmd = 'Move Right Slowly'
if (rel1 == "'Far'" and rel2 == "'Near'"):
vel_cmd = 'Move Left Quickly'
if (rel1 == "'Far'" and rel2 == "'Medium'"):
vel_cmd = 'Move Left Slowly'
if (rel1 == "'Far'" and rel2 == "'Far'"):
vel_cmd = 'Move Forward'
```

## 3.3 Implementation Notes

- The current implementation assumes that there are at least two markers in the environment at any given point of time. This is the core assumption around which the entire implementation is built.

- The approach is heavily dependent on the perception module, the robot moves according to what it sees. If we want to remove the dependency on the use of markers then we need a robust perception module that can track

features such as notice boards etc. this may be achieved b using computer vision techniques such as template matching or optical flow analysis.

- Currently the controllers used in the approach are configured to move the robot forward through the corridor and cannot deal with corner's and dead ends, to allow the robot to deal with such situation additional controller nodes can be implemented that allow such behaviors to be executed safely. In such cases only the controller node needs to be changed.

- Additionally we assume that a minimum of two objects(discounting the robot) are necessary for the robot to build valid relations and successfully navigate through the corridor environment, thus forcing the relationships to be built for only three object sets ([object1, robot], [object2, robot] and [object1, object2]) and enforce this using the 'dynamic_args' argument for the QSRLib.

# 4

# Methodology

This chapter aims to highlight both the hardware and software setup of the developed system. The integration of the software libraries and packages has been implemented and tested extensively on the 'KUKA youBot' which is the aforementioned hardware platform. The following sections briefly illustrate the applied hardware and explain the software packages and their integration with the QSRLib library, which allows us to express physical space qualitatively.

## 4.1 Hardware Setup

The applied robot platform for our application is the 'KUKA youBot'. This is an omni-directional platform, equipped with a 5 Degree of freedom robotic arm, that features a two finger gripper. The robot base is equipped with two 'Hokuyo URG-04LX' laser range finders situated at the front and at the back respectively, this placement allows for robust localization and map-based navigation in known indoor environments [38]. Physically the dimensions of the robot are as follows length 58cm, width 38cm, height 14cm with a ground clearance of 2cm and a minimum and maximum velocity of 0.01m/s and 0.8m/s respectively [2]. The power to this applied platform comes from a 24 volt, 5 Ah lead-acid battery that has an approximate optimal runtime of 90 minutes, but this varies depending upon a multitude of factors such as the robot's velocity, sensors used etc.


Figure 4.1: The youBot robot platform

The robotic arm plays host to a camera mounting which supports the 'ASUS Xtion Pro Live' RDB-D camera, which as the name suggests supports the perception of depth information in addition to the usual RGB or raw image data. This RGB-D camera has a detection range of 80cm - 3.5m with a field of view limited to $58°, 45°, 70°$ horizontally, vertically and diagonally respectively [42]. The image size for this camera is '640x480' at 30frames per second in a VGA format, also being highly compatible with the 'OpenNi development framework' ensures it's easy integration with the ROS packages by the 'b-it-bots RoboCup@Work' team, which has already developed scene segmentation and object detection applications centered around this sensor. The stock internal computer that comes issued with the youBot has been replaced with a 'Intel Core i5' processor that facilitates highly computational perception tasks to be run directly on the platform [38].

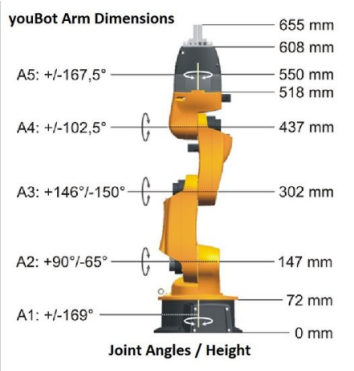



Figure 4.2: youBot arm at full extension

## 4.2 Software Framework

The software framework underpinning the applied robot platform has been developed by the 'b-it bots @Work' team and is based on ROS(Kinetic Kame version) or the robot operating system [36], which provides a modular and distributed structure to design a functional system based on the developer's requirements. This modular architecture provides a rapid and robust communication infrastructure that is based on actions, services and clients to interchange data amongst the various different functional, software components of the applied platform. The framework provides an effective and efficient medium for interfacing various sensors and actuators such as laser scanners, cameras etc.

Another merit of the robot operating system is it's provision of advanced tools for visualizing and testing various types of data and troubleshooting the entire system in cases of failures or errors. One such heavily utilized tool was 'rosbag', used particularly to capture data [27] and evaluate the developed implementation with varying parameters but always on the same set of captured data.

## 4.3 Integration of Qualitative representations

The current software framework, set-up on the robot consists of numerous packages that allow the control of it's base as well as arm actuators while also providing an effective interface to it's various sensors. This modular approach facilitates the use of these small components to develop a higher level task. In the instance of 'Qualitative spatial representations' we need to access the raw RGB image from the camera to detect features or objects and further extract their approximated pose in order to build a qualitative relation with respect to the robot. The nature of this relation depends upon the type of qualitative calculi that is being implemented in order to define these abstractions.

As the objective of this project is to show an generalized and efficient utilization of the existing qualitative calculi, we do not develop any of these calculi from scratch, instead deciding to exploit an existing qualitative spatial representations library called 'QSRLib' [25], this library contains ROS compatible python implementations of the various qualitative calculi, as discussed in the previous chapter.

Although this library can be used either as a standalone python package or a ROS catkin package, we use it primarily with ROS and hence shall focus on it's installation and integration with the same. The 'qsr_lib' package is the one that is being used in our implementation and has system dependencies on 'numpy' and 'matplotlib'. Installing the library is extremely easy as it involves directly cloning the repository(https://github.com/strands-project/strands_qsr_lib.git) from git, and moving the 'qsr_lib' package into the 'src' folder of our catkin workspace [1].

The pre-requisites for using any of the qualitative calculi implemented in the library [1], [25] is input data such as distinctive object id's or names for the various objects amongst which a qualitative relation is desired, a time-series of the states of the perceived objects and the Cartesian coordinates for each of the objects at every instance of the time series. This information is packaged into a custom input data object that is the default input data format of the library. Additional information such as the size of the object may also be included in the input data object, but it is mandatory to include the name of the qualitative calculi for which the qualitative spatial relations are to be computed. This input data structure is sent to a QSRLib service in the form of a request message, the server then computes the relationships and send the output in the form of a response message(client-service architecture in ROS) that details the qualitative relations between the objects constrained by their respective time stamps. The library comes built with the necessary functions that can be used to convert raw data into the data structure format required by the library. The output of the QSRLib can then be further inspected and employed to make decisions regarding the movement or path of the applied mobile platform.

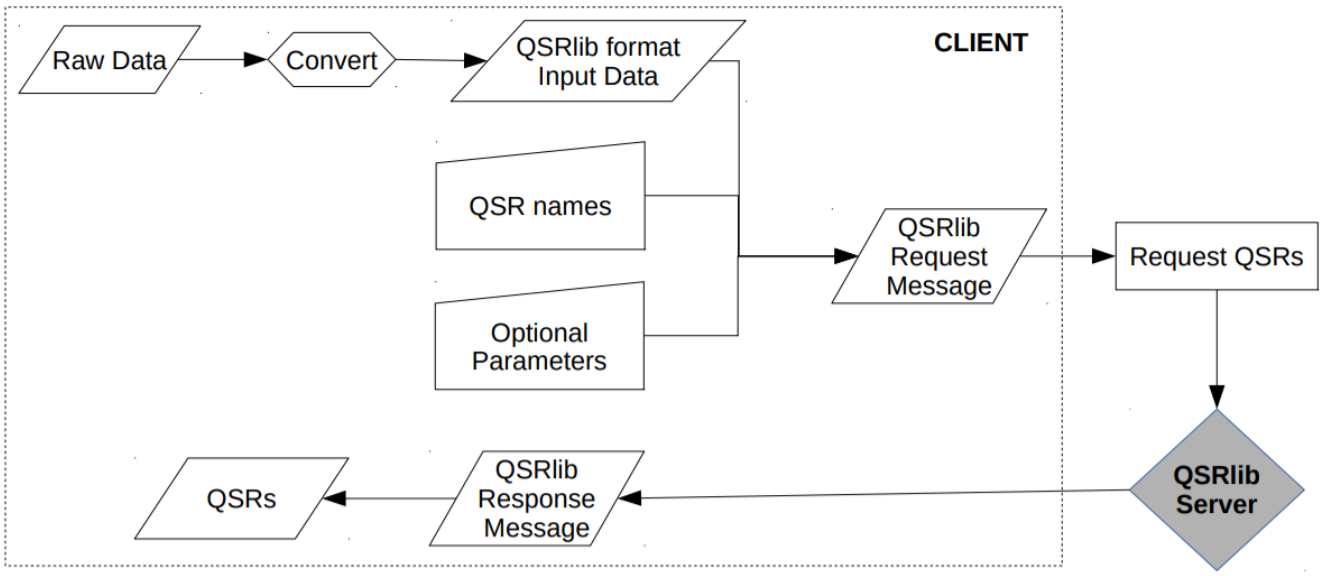



Figure 4.3: A flow chart detailing the inner workings of the QSRLib library [1], [25].

# 5

# Experimental Evaluation

The primary goal of this chapter is to provide vital information regarding the empirical evaluation that was carried out to validate the proposed approach. The following sections will discuss the practical use cases for the developed system and give a brief overview of the general test setup of the performed experiments before diving into the details of the experiments themselves. A logical analysis of the obtained results will be performed to conclude the chapter.

## 5.1 Use Cases

The main aim of this project is to evaluate the efficiency paradigm of qualitative spatial representations for navigation in mobile robots, especially in closed indoor spaces where quantitative representations of the physical space are considered excessive and unnecessary. Therefore keeping in mind these preconditions we define the following possible use cases.

**Use case: Navigating in a corridor environment**

A mobile robot is tasked with navigating from point 'A' to point 'B' in a corridor environment such that it should avoid collision with the walls and any other static or dynamic obstacles if they exist in it's path. Furthermore the robot's movement should be such that it avoids any sudden motions that may seem unsafe or unintuitive to an human agent who might interact with the robot. Also chiefly,

the robot must achieve this path traversal in a manner that is as efficient as possible ,while dealing with imprecise information about the environment or in extreme cases lack of complete information about the environment.

**Resulting Requirements**

Ideally it is desired that any new functionality that is being implemented must be highly generalizable, but keeping in mind the given problem it is unrealistic to expect a 'one size fit's all' implementation that works impeccably for all imaginable situations without any restrictions or compromises. Hence a list of requirements resulting from the problem statement and the above described use case is presented below:

- The starting position should be irrelevant when navigating the corridor.
- The detection of the markers or features should be robust with respect to a reasonable speed of the robot.
- The number of markers or features should not adversely affect the robot's behavior.
- The camera used should have a reasonable field of view so that it can see both the walls and their respective features at any given point of time.
- The camera should be able to capture the features reasonably well, irrespective of the lighting conditions.
- The motion profile of the robot should be a smooth and not disruptive.
- It should be able to navigate any corridor irrespective of it's size or the color of it's walls.

## 5.2 Data collection

During the development stage of the algorithm it was tested continuously against a set of collected data that served as a simulation for corresponding real world situations. This method served as a good test to ensure robustness of the individual

software components as well as the entire system once it was combined together. The data was collected by manually driving around the robot in a corridor and recording the output of the camera video stream(RGB) after markers had been attached to the walls of the corridor. This simulated data served as a backbone for understanding some of the constraints of the developed algorithm such as the minimum number of markers required, the field of view of the camera, the position of the camera, the velocity of the robot and the orientation of the robot to ensure that both the walls and their respective markers are visible. The collected data simulated conditions such as,

- The robot moving along the center of the corridor.
- The robot moving along the left/right wall.
- The robot moving diagonally across the corridor.
- The robot following a zig-zag pattern along the corridor.

In all these collected videos a set of two markers was placed on the opposing walls such that both the markers lie on the same axis that is perpendicular to the walls, also the markers were placed at equidistant intervals.

## 5.3 General Test Setup

The testing of the developed algorithm was carried out in a two step manner wherein during the first step we evaluated the performance of the two implemented calculi for the task of corridor navigation. The performance is evaluated based on three criteria, the effect of the starting positions, the effect of the marker positions and the number of markers. The evaluation was judged based on the number of possible control paradigms that ensued in a crash with either of the walls. This led to the selection of the calculi with the least number of crashes, the selected calculi was then compared against the existing map based navigation approach which uses quantitative spatial representations and has been implemented on the applied robot platform by the 'b-it bots' team. This comparison yields a evaluation of the efficiency paradigms for both the approaches.

A general setup of the performed experiments is presented below, with the specifics of each experiment discussed in a later section.

**Experiment Variables:**

- Initial position and orientation of the robot.
- Field of View of the camera.
- Lighting conditions of the corridor.
- Initial Position of the camera.
- Position of the markers relative to the fixed camera position.
- Position of the markers relative to each other.
- Battery voltage.
- Range of the wireless network.

**Software Requirements:**

- ROS Kinetic
- Python 2.7 or higher
- Qsr_lib package
- Qsr_nav package
- Mir_bringup package
- Mir_moveit_youbot_brsu_4 package
- Moveit_commander package

**Hardware Requirements:**

- Kuka Youbot
- PC capable of runnning ROS
- Remote joystick
- Power supply
- Aruco markers
- Asus Xtion Pro camera

## 5.4 Experiments

### 5.4.1 Experiments for QTCB calculi

**Testing the navigational capabilities of the QTCB calculus in a indoor corridor environment**

The individual experiments and their respective analysis have been presented in a tabular format in the figures 5.1, 5.2 and 5.3.

**Evaluation**

From the experiment tables presented in the following figures(5.1,5.2,5.3) it is clear to see that the main reason for failure of the QTCB calculus is the limited FOV of the camera used. Also the tabular structure makes it clear to understand that the the algorithm is robust enough to function with an arbitrary number and placement of the markers with the only failure occurring when it cannot detect atleast two markers placed on opposite walls. Additionally out of all the test scenarios the only clear failure(irrespective of starting position) of the algorithm was when the markers were placed in a staggered manner along the walls, this was also due to the fact that the camera's FOV is limited and prevented it from detecting both the markers at all instances of time. All the experiments were carried out such that they travel from point 'A' to 'B' and back from 'B' to 'A' this was done to ensure

that the detection component isn't favorable to any one side or travel and is robust enough to function even if the markers on the opposing walls were interchanged. Out of a set of 12 experiments conducted the QTCB representation was able to perform collision free navigation in all but 1 set of experiments giving it a success rate of 83.3 percent.

### 5.4.2 Experiments for ARGD calculi

#### Testing the navigational capabilities of the ARGD calculus in a indoor corridor environment

The individual experiments and their respective analysis have been presented in a tabular format in the figures 5.4, 5.5 and 5.6. For the ARGD calculus the distance thresholds used for discretizing continuous space are an additional experiment variable that affects the representation of physical space.

#### Evaluation

From the experiment tables presented in the following figures(5.4, 5.5, 5.6) it is clear to see that there are two main reasons for failure of the ARGD calculus, the first is the limited FOV of the camera used and the second is the inability of the calculi to distinguish between two objects that lie in the same distance threshold due to this the number of failure cases is higher for the ARGD when compared to the $QTC_B$. Also the tabular structure makes it clear to understand that the the algorithm isn't robust enough to function with any arbitrary number and placement of the markers with collisions occurring regularly in nearly all the test cases.One of the clearest cases of failure was when the markers were placed in a staggered manner along the walls, this was a clear illustration of the ARGD being an insufficient calculi when it comes to dealing with mobile objects, with the need for fine distinctions between the objects lying in the same distance threshold being clearly highlighted in this case. All the experiments were carried out such that they travel from point 'A' to 'B' and back from 'B' to 'A' this was done to ensure that the detection component isn't favorable to any one side or travel and is robust enough to function even if the markers on the opposing walls were interchanged. Out of a set of 12 experiments

conducted the ARGD representation was able to perform collision free navigation in all but 4 sets of experiments giving it a success rate of 66.6 percent.

### 5.4.3 Common Observations

- All the movements observed during the experiments are gradual and there is no sudden change to the robot's velocity or speed.

- The algorithm can be made more robust if the $ARGD$ and the $QTC_B$ calculus can be combined, this is possible with the current implementation but is complicated as the number of rulesets would increase exponentially with respect to the objects in consideration.

- In the current setup the algorithm is inherently dependent on the perception module and any changes to this module would likely warrant changes to the entire setup especially in the case of the distance based $ARGD$ calculus, the $QTC_B$ calculus on the other hand would only require that the number of objects in consideration be changed to the desired number.

- The current setup uses small velocity values to control the robot with the maximum value being 0.1 m/s and the lowest being 0.02 m/s.

- Improvements to the current algorithm can also be made using different setup of the cameras where two cameras facing the opposite wall may be used.

- The current implementation also uses a time based filter to build relations only when two or more markers are detected simultaneously at similar timesteps, in cases where only a single object is used to build the relation with the robot this filtering must be removed to ensure successful navigation.

- Also if more than two objects are to be used when calculating the motion updates then, filtering has to be done to ensure that there is no repetition when building the relations that are consequently used for the decision making process.

### 5.4.4 Conclusion

From the above breakdown of the results it is easy to see that the $QTC_B$ calculi outperforms the $ARGD$ calculi when it comes to the task of collision free robot navigation. As discussed above this is attributed to the fact that the distance calculi lacks the tools to distinguish between markers or features that lie in the same distance thresholds, furthermore due to the limited representation capabilities there exist very few instances where a clear distinction between the markers can be made hence resulting in a fewer number of instances where suitable control commands can be exercised. Beyond this, both the calculi suffered failures due to the limited FOV of the camera as it prevented the detection of the markers lying on the opposing walls whenever the robot moved close to either of the walls(it remains crucial to the algorithm that atleast two markers are detected simultaneously, when placed along the same axis or otherwise). To compliment the validity of the results it is worth mentioning that the $ARGD$ calculus works on abstractions of spatial distances and was developed for representing and reasoning about static objects whereas the $QTC_B$ calculi works on abstractions of relative trajectories between two objects and was developed representing and reasoning about mobile objects, hence it is obvious that it should perform better than the $ARGD$ calculus for the given task. Thus based on this empirical evidence we select the $QTC_B$ calculi to compare and evaluate the efficiency paradigms against a quantitative map based representation.

## 5.5 Evaluation of the efficiency of qualitative representations

Based on the above validation efficiency of the $QTC_B$ calculi was compared against the map based navigation technique that utilized quantitative spatial representations, on the task of navigating from point 'A' to 'B' in the corridor. The starting positions were kept approximately same for the purpose of this testing and the velocity values were ensured to be same. Furthermore since we would like to evaluate the battery consumption a fully charged battery was used during the tests for each representation. The experiment was repeated thrice for each of

the representations so as to obtain any variance in the values and ensure accurate resulting values. For the purpose of this evaluation the $QTC_B$ experiment was setup so that the markers always lay on the same perpendicular axis to the wall and the number of markers on both the walls was the same, the starting position was always along the center of the corridor.

Need to add the battery values here

## 5.6 Conclusion

From the above evaluation it is obvious(or not ?) that the qualitative representations are more efficient in comparison to quantitative representations. This satisfies our initial aim of providing conclusive proof to show the improved efficiency of qualitative representations and evaluating it's utility for the same.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls. | Equal number of markers on both the walls | * Successful navigation of the corridor while avoiding collision with the walls.<br>* A highly discontinuous motion profile with jerky motion.<br>* A video log of the performed experiment. | * The robot managed to successfully navigate the corridor.<br>* The motion profile is smooth and not jerky as previously thought.<br>* The robot maintains a path that is along the center of the corridor.<br>* No collision cases were observed.<br>* It remains crucial to the algorithm that atleast two markers are detected simultaneously, when placed along the same axis. |
| | More number of markers on one of the walls in comparison to the other wall | * Successful navigation of the corridor while avoiding collision with the walls.<br>* A highly discontinuous motion profile with jerky motion.<br>* A video log of the performed experiment.<br>* A collision case due to uneven number of markers are placed on the opposing walls. | * The robot is able to successfully navigate through the corridor despite of the uneven placement of the markers on the opposing walls.<br>* The motion profile is smooth and not jerky as previously thought.<br>* The robot tries to move along the center of the corridor.<br>* Collision occurs when the robot get's too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the walls.<br>*A collision case due to staggered placement of the markers | *The robot almost always fails. The failure can be attributed to the irregular placement of the markers on the wall.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move along the center of the corridor, but due to lack of any motion updates ends up moving diagonally towards either wall and collides with it.<br>*Collision occurs when the robot gets too close to either wall as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly discontinuous motion profile with jerky motion.<br>*A zig-zag pattern to the robot's motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers. | *The robot is able to successfully navigate through the corridor despite of the staggered, uneven placement of the markers on the opposing walls.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The motion pattern is moderately zig-zag as previously assumed.<br>*The robot tries to move along the center of the corridor.<br>*Collision occurs when the robot gets too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |

Figure 5.1: Table of conducted experiments for $QTC_B$ and their respective analysis for a starting position that is along the center of the corridor.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the walls | *The robot managed to successfully navigate the corridor, only if the starting position still allowed the camera to detect both the markers.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the wall. | *The robot managed to successfully navigate the corridor as long as there were atleast two observable markers on the opposite walls.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor in a diagonal fashion.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the walls.<br>*A collision case due to staggered placement of the markers | *The robot almost always fails. The failure can be attributed to the irregular placement of the markers on the wall.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor, but due to lack of any motion updates ends up moving diagonally towards the opposite wall and collides with it.<br>*Collision occurs when the robot is too close to the wall as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers.<br>*A diagonal zig-zag path towards the center of the corridor. | *The robot is able to successfully navigate through the corridor as long as it can see atleast two markers on the opposite walls.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor in zig-zag fashion.<br>*Collision occurs when the robot gets too close to the wall as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |

Figure 5.2: Table of conducted experiments for $QTC_B$ and their respective analysis for a starting position that is along the right wall of the corridor.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls. | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the walls | *The robot managed to successfully navigate the corridor,only if the starting position still allowed the camera to detect both the markers.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the wall | *The robot managed to successfully navigate the corridor as long as there were atleast two observable markers on the opposite walls.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor in a diagonal manner.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving too close to the walls.<br>*A collision case due to staggered placement of the markers | *The robot almost always fails. The failure can be attributed to the irregular placement of the markers on the wall.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor, but due to lack of any motion updates ends up moving diagonally towards the opposite wall and collides with it.<br>*Collision occurs when the robot is too close to the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly discontinuous motion profile with jerky motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers.<br>*A diagonal zig-zag path towards the center of the corridor. | *The robot is able to successfully navigate through the corridor as long as it can see atleast two markers on the opposite walls.<br>*The motion profile is smooth and not jerky as previously thought.<br>*The robot tries to move towards the center of the corridor in zig-zag fashion.<br>*Collision occurs when the robot gets too close to the wall as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |

Figure 5.3: Table of conducted experiments for $QTC_B$ and their respective analysis for a starting position that is along the left wall of the corridor.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls. | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without discontinuous jerky motion.<br>*A video log of the performed experiment. | *When starting from a central position along the walls, the robot managed to successfully navigate the corridor.<br>*The motion profile is smooth as previously theorized.<br>*The robot tries to move along the center of the corridor. |
| | More number of markers on one of the walls in comparison to the other wall | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when robot gets too close to the walls<br>*A collision case when uneven number of markers are placed on the opposing walls. | *When starting along the center of the corridor, the algorithm almost always fails as the numerous markers always lie in the same distance range and hence no tangible update to the robots motion can be made also another factor contributing to the failure is that the markers are not detected as they move out of the FOV of the camera as the robot moves forward or diagonally.<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot gets too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when robot gets too close to the wall<br>*A collision case when markers are placed in a staggered manner on the opposing walls. | *The collision is unavoidable in this scenario, this failure is attributed to the lack of extremely fine values for the distance thresholds and also to the limited FOV of the camera, which often loses track of the markers on the opposing wall as the robot moves along the corridor(diagonally).<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly smooth motion profile without jerky motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers.<br>*A zig-zag path along the center of the corridor. | *When starting from a position along the center of the corridor the robot moves along the center.<br>*The motion profile is smooth as previously thought.<br>*Collision almost never occurs as there is always atleast one marker on the opposite wall that is detected at the same instance of time. |

Figure 5.4: Table of conducted experiments for *ARGD* and their respective analysis for a starting position that is along the center of the corridor.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls. | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving or starting from a position too close to the wall | *When starting from a position close to the right wall the robot moves diagonally towards the opposite wall and will crash as the markers go out of the FOV. In cases where the markers are detected simultaneously along the same axis the robot showed a tendency to move along the center of the corridor and the ability to recover from possible collision cases.<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving or starting from too close to the wall<br>*A collision case when uneven number of markers are placed on the opposing walls. | *When starting from a position close to the right wall the robot moves along the wall or collides, this depends on whether the FOV of the camera if the markers are detected simultaneously in the same frame then collision can be avoided otherwise no motion updates are performed and the robot and will crash as the markers go out of the FOV.<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when starting or moving too close to the wall<br>*A collision case when markers are placed in a staggered manner on the opposing walls. | The collision is unavoidable in this scenario , this failure is attributed to the lack of extremely fine values for the distance thresholds and also to the limited FOV of the camera, which often loses track of the markers on the opposing wall as the robot moves along the corridor(diagonally).<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly smooth motion profile without jerky motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers.<br>*A diagonal zig-zag path towards the center of the corridor. | *When starting from a position close to the right wall the robot moves along the wall or diagonally towards the opposite wall, this depends on the FOV of the camera and on detection of the opposing set of markers.<br>*The motion profile is smooth as previously thought.<br>*Collision almost never occurs as there is always atleast one marker on the opposite wall that is detected at the same instance of time. |

Figure 5.5: Table of conducted experiments for *ARGD* and their respective analysis for a starting position that is along the right wall of the corridor.

| Placement of markers | Pattern of the marker placements | Expected results | Obtained results and analysis |
|---|---|---|---|
| Along a common axis perpendicular to the opposing walls. | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving or starting from a position too close to the wall | *When starting from a position close to the lefft wall the robot moves diagonally towards the opposite wall and will crash as the markers go out of the FOV. In cases where the markers are detected simultaneously along the same axis the robot showed a tendency to move along the center of the corridor and the ability to recover from possible collision cases.<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when moving or starting from too close to the wall<br>*A collision case when uneven number of markers are placed on the opposing walls. | *When starting from a position close to the left wall the robot moves along the wall or collides, this depends on whether the FOV of the camera if the markers are detected simultaneously in the same frame then collision can be avoided otherwise no motion updates are performed and the robot and will crash as the markers go out of the FOV.<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to either of the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| Staggered along the wall | Equal number of markers on both the walls | *Successful navigation of the corridor while avoiding collision with the walls.<br>*A highly smooth motion profile without any jerky motion.<br>*A video log of the performed experiment.<br>*A collision case when starting or moving too close to the wall<br>*A collision case when markers are placed in a staggered manner on the opposing walls. | The collision is unavoidable in this scenario , this failure is attributed to the lack of extremely fine values for the distance thresholds and also to the limited FOV of the camera, which often loses track of the markers on the opposing wall as the robot moves along the corridor(diagonally).<br>*The motion profile is smooth as previously thought.<br>*Collision occurs when the robot is too close to the walls as the opposite set of markers goes out of the FOV of the camera and hence no update to the robot's motion can be made. |
| | More number of markers on one of the walls in comparison to the other wall | *A highly smooth motion profile without jerky motion.<br>*A video log of the performed experiment.<br>*A collision case due to staggered and uneven placement of the markers.<br>*A diagonal zig-zag path towards the center of the corridor. | *When starting from a position close to the left wall the robot moves along the wall or diagonally towards the opposite wall, this depends on the FOV of the camera and on detection of the opposing set of markers.<br>*The motion profile is smooth as previously thought.<br>*Collision almost never occurs as there is always atleast one marker on the opposite wall that is detected at the same instance of time. |

Figure 5.6: Table of conducted experiments for *ARGD* and their respective analysis for a starting position that is along the left wall of the corridor.

# 6

# Conclusions

**6.1 Contributions**

**6.2 Lessons learned**

**6.3 Future work**

# References


[1] QSRLib online documentation documentation page. URL `https://qsrlib.readthedocs.io/en/latest/index.html`.

[2] YouBot Detailed Specifications 3D model hardware description. URL `http://www.youbot-store.com/wiki/index.php/YouBot Detailed Specifications 3D model`.

[3] James F Allen. Maintaining knowledge about temporal intervals. In *Readings in qualitative reasoning about physical systems*, pages 361–372. Elsevier, 1990.

[4] JV Álvarez-Bravo, JC Peris-Broch, JJ Álvarez-Sánchez, and MT Escrig-Monferrer. A guide for blind people using a quantitative+ qualitative spatial representation. In *Proceedings of the first international congress on domotics, robotics and remote-assistance for all*, pages 495–505, 2006.

[5] Nicola Bellotto et al. Robot control based on qualitative representation of human trajectories. 2012.

[6] Stan Birchfield. Klt: An implementation of the kanade-lucas-tomasi feature tracker. *http://www. ces. clemson. edu/˜ stb/klt/*, 2007.

[7] Alan Frank Blackwell. Spatial reasoning for robots: a qualitative approach. 1988.

[8] Juan Chen, Anthony G Cohn, Dayou Liu, Shengsheng Wang, Jihong Ouyang, and Qiangyuan Yu. A survey of qualitative spatial representations. *The Knowledge Engineering Review*, 30(1):106–136, 2015.

[9] Zhichao Chen and Stanley T Birchfield. Qualitative vision-based mobile robot navigation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2686–2692. IEEE, 2006.

[10] Zhichao Chen and Stanley T Birchfield. Qualitative vision-based path following. *IEEE Transactions on Robotics*, 25(3):749–754, 2009.

[11] Eliseo Clementini, Paolino Di Felice, and Daniel Hernandez. Qualitative representation of positional information. *Artificial intelligence*, 95(2):317–356, 1997.

[12] Anthony G Cohn. Qualitative spatial representation and reasoning techniques. In *Annual Conference on Artificial Intelligence*, pages 1–30. Springer, 1997.

[13] Anthony G. Cohn and Shyamanta M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta informaticae*, 46(1-2):1–29, 2001.

[14] Anthony G Cohn and Jochen Renz. Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence*, 3:551–596, 2008.

[15] Matthias Delafontaine, Anthony G Cohn, and Nico Van de Weghe. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*, 38(5):5187–5196, 2011.

[16] Christian Dondrup and Marc Hanheide. Qualitative constraints for human-aware robot navigation using velocity costmaps. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 586–592. IEEE, 2016.

[17] Christian Dondrup, Nicola Bellotto, Marc Hanheide, Kerstin Eder, and Ute Leonards. A computational model of human-robot spatial interactions based on a qualitative trajectory calculus. *Robotics*, 4(1):63–102, 2015.

[18] Frank Dylla and Reinhard Moratz. Empirical complexity issues of practical qualitative spatial reasoning about relative position. In *Workshop on Spatial and Temporal Reasoning at ECAI*, volume 2004, 2004.

[19] Frank Dylla and Reinhard Moratz. Exploiting qualitative spatial neighborhoods in the situation calculus. In *International Conference on Spatial Cognition*, pages 304–322. Springer, 2004.

[20] Frank Dylla and Jan Oliver Wallgrün. On generalizing orientation information in opra. In *Annual Conference on Artificial Intelligence*, pages 274–288. Springer, 2006.

[21] Andrew U Frank. Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages & Computing*, 3(4):343–371, 1992.

[22] Gordon Fraser, Gerald Steinbauer, Franz Wotawa, et al. Application of qualitative reasoning to robotic soccer. In *18th Int. Workshop on Qualitative Reasoning*, 2004.

[23] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial intelligence*, 54(1):199–227, 1992.

[24] Christian Freksa and Kai Zimmermann. On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *Systems, Man and Cybernetics, 1992., IEEE International Conference on*, pages 261–266. IEEE, 1992.

[25] Yiannis Gatsoulis, Muhannad Alomari, Chris Burbridge, Christian Dondrup, Paul Duckworth, Peter Lightbody, Marc Hanheide, Nick Hawes, DC Hogg, AG Cohn, et al. Qsrlib: a software library for online acquisition of qualitative spatial relations from video. 2016.

[26] Francisco J Glez-Cabrera, Jose Vicente ALvarez-Bravo, and Fernando DiAz. Qrpc: A new qualitative model for representing motion patterns. *Expert systems with applications*, 40(11):4547–4561, 2013.

[27] Frederik Hegger. 3D People Detection in Domestic Environments. Technical Report March, Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany, 2012. URL `https://opus.bib.hochschule-bonn-rhein-sieg.de/files/8/brsu_techreport_02_2012_Fred`

[28] Amar Isli and Anthony G Cohn. A new approach to cyclic ordering of 2d orientations using ternary relation algebras. *Artificial Intelligence*, 122(1-2): 137–187, 2000.

[29] Amar Isli, Volker Haarslev, Ralf Möller, et al. *Combining cardinal direction relations and relative orientation relations in qualitative spatial reasoning*. Univ., Bibliothek des Fachbereichs Informatik, 2001.

[30] G É LIGOZAT. Reasoning about cardinal directions. *Journal of Visual Languages & Computing*, 9(1):23–44, 1998.

[31] John Carl Lowe and S Moryadas. *The geography of movement*. Houghton Mifflin Boston, 1975.

[32] Dominik Lücke, Till Mossakowski, and Reinhard Moratz. Streets to the oprafinding your destination with imprecise knowledge. In *IJCAI-2011 Workshop 27*, page 25. Citeseer, 2011.

[33] Pierre-Emmanuel Michon and Michel Denis. When and why are visual landmarks used in giving directions? In *International Conference on Spatial Information Theory*, pages 292–305. Springer, 2001.

[34] Reinhard Moratz. Representing relative direction as a binary relation of oriented points. In *ECAI*, volume 6, pages 407–411, 2006.

[35] Alexandra Musto, Klaus Stein, Andreas Eisenkolb, Thomas Röfer, et al. Qualitative and quantitative representations of locomotion and their application in robot navigation. In *IJCAI*, volume 99, pages 1067–1072. Citeseer, 1999.

[36] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[37] David A Randell, Zhan Cui, and Anthony G Cohn. A spatial logic based on regions and connection. *KR*, 92:165–176, 1992.

[38] M. Roscoe, A. Moriarty, S. Alexandrov, P. Ramanujam, N. Giftsun, F. Hegger, N. Hochgeschwender, J. Paulus, and G. K. Kraetzschmar. The b-it-bots robocup@work 2012 team description paper. In *IROS Competition RoboCup@Work*, Vilamoura, Portugal, 2012. URL `http://mas-group.inf.h-brs.de/wp-content/uploads/2013/07/atworktdpiros2012.p`

[39] Nikitas M Sgouros. Qualitative navigation for autonomous wheelchair robots in indoor environments. *Autonomous Robots*, 12(3):257–266, 2002.

[40] Danelle C Shah and Mark E Campbell. A robust qualitative planner for mobile robot navigation using human-provided maps. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2580–2585. IEEE, 2011.

[41] Danelle C Shah and Mark E Campbell. A qualitative path planner for robot navigation using human-provided maps. *The International Journal of Robotics Research*, 32(13):1517–1535, 2013.

[42] Daniel Maximilian Swoboda. A comprehensive characterization of the asus xtion pro depth sensor. 2014.

[43] Nico Van de Weghe. *Representing and reasoning about moving objects: A qualitative approach*. PhD thesis, Ghent University, 2004.

[44] Nico Van de Weghe, Anthony G Cohn, Philippe De Maeyer, and Frank Witlox. Representing moving objects in computer-based expert systems: the overtake event example. *Expert Systems with Applications*, 29(4):977–983, 2005.

[45] Nico Van de Weghe, Bart Kuijpers, Peter Bogaert, and Philippe De Maeyer. A qualitative trajectory calculus and the composition of its relations. In *International Conference on GeoSpatial Sematics*, pages 60–76. Springer, 2005.

[46] Nico Van de Weghe, Anthony Cohn, Guy De Tre, and Philippe De Maeyer. A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. *Control and Cybernetics*, 35(1):97–119, 2006.

[47] Yujin Wakita, Shigeoki Hirai, and Kazuo Machida. Intelligent monitoring system for limited communication path: Telerobotic task execution over internet. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 2, pages 104–109. IEEE, 1995.

[48] Lv Yan, Peng Wang, and Zonghai Chen. Qualitative representation and reasoning in cognitive maps : A survey from spatial cognition perspective. 2012.

[49] Kai Zimmermann and Christian Freksa. Qualitative spatial reasoning using orientation, distance, and path knowledge. *Applied intelligence*, 6(1):49–58, 1996.