Agent-aware mobile robot navigation in domestic environments

Inês Alexandre Rodrigo Ventura Oscar Lima
ines.afonso@tecnico.ulisboa.pt {rodrigo.ventura, olima}@isr.tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2018

Abstract

Taking into consideration the technological progress and rising connection between human needs and robotics, domestic environments are expected to be increasingly populated by varied robots. Autonomous navigation is an essential competence for mobile service robots deployed in these environments. In fact, researches extensively studied it and, therefore, a considerate amount of tools and algorithms have been and continue to be developed in this field. The work developed focuses on addressing the challenge of autonomous navigation in a dynamic environment populated with humans and robots. Considering in particular humans and non-cooperative robots, *a priori* agent trajectory is unknown. The absence of coordination between agents may result in deadlock or even collision. This work focuses on attaining predictions, with associated uncertainty, to improve robot navigation. Based on previous velocity data and current position, *Brownian Motion (BM)* and *Metropolis Algorithm (MA)* are the applied methods for predicting the next state of the environment. *Kernel Density Estimation (KDE)* is used to compute the probability density function used in MA. A map of predictions is built and sent to the local planner resulting in a faster response to risky situations. The obtained results evaluate not only the computational time but also the prediction's probability success of both approaches. BM was concluded to deliver an overall more conservative, accurate and faster prediction than MA combined with KDE. Considering a prediction map is built to later influence the local planner, the success of the developed work implies an enhanced planner response. Consequently, navigation is improved.

Keywords: Mobile Robot Navigation, Human-Aware Navigation, Robot-Aware Navigation, Agent-Aware Navigation

1. Introduction

As technology progresses, human being's needs are increasingly linked to robotics. Consequently, autonomous mobile service robots are nowadays becoming more and more present around humans in varied environments.

In order to effectively operate, one of the essential aspects to consider is the robots' ability to autonomously navigate.

A large amount of uncertainty is inherent to real-world environments, where robots are expected to be utilized.

When deployed in domestic environments robots are expected to help carry out diverse tasks. For such application, an awareness of humans or other robots that may be, and interact, in the same environment is essential. In a group of interacting humans, each person is found to respect a virtual personal space around every other intervenient. Only by respecting the minimum personal space both comfort and safety perception as well as social acceptance from humans may occur [1].

This work considers the autonomous navigation of a mobile robot in a domestic environment scenario which can be populated by humans as well as non-cooperative robots. The presence of humans and/or robots in the environment involves considering, in real-time, their safety and comfort thereby posing challenges in perception and prediction of behaviour.

1.1. Problem Overview

The problem tackled by this work assumes that the algorithm will only be used in one robot, deployed in a dynamic environment where static and dynamic obstacles can influence real-time navigation.

Autonomous navigation requires awareness of the surroundings. For such, cameras and various sensors can be used. However, parts of the state can be missing from each sensors' data and consequently the environment is only partially observable.

The method developed should ensure that the path chosen does not affect safety and/or comfort of the agents, while still taking into consideration the probability of defective perception. To disregard

the problem of wrongful perception this work considers all detected points individually to later track and predict the overall surrounding change. A correct perception of environment transitions should enable the prediction of next states.

The challenge concerning prediction arises from the stochastic property of the environment. Since there is uncertainty present in the behavior of others, the environment state is not completely determined by the prior state and the actions executed. In addition, non-cooperative agents are considered, there is an absence of communication and *a priori* knowledge about trajectory. Consequently, coordination between agents is inexistent and collision or mutual blocking situations, deadlock, may occur. An accurate prediction of environment changes enables a faster response to undesired situations.

A parametric, *Brownian Motion (BM)*, and non-parametric, *Metropolis Algorithm (MA)*, approach is used considering previous velocity data and current position. The first is a simpler and faster method that is known to successfully simulate human walking behavior. MA is used in combination with *Kernel Density Estimation (KDE)*, composing a promising but more complex method.

Taking into account reaction to a possible collision or deadlock situation is performed on a local level, data solely collected from laser range finders is considered sufficient.

By constructing a map of predictions and sending it to the local planner a faster response to risky situations is obtained.

1.2. Related Work

To the best of our knowledge the field of navigation with regards to general agent awareness hasn't received an extensive treatment by the literature.

Nonetheless, *Human-Aware Navigation (HAN)* has been the focus of many researches. The existing approaches to this field include actions of understanding, deliberating and reacting in order to tackle challenges in producing a natural motion while considering human comfort and social constraints [2]. These actions encompass a group of procedures we found applicable, not only to HAN but also to *Robot-Aware Navigation (RAN)* and more generally, to *Agent-Aware Navigation (AAN)*, Figure 1 where the connections in the diagram show main data flow.

The **Understanding** portion of the schematic collects information about the environment and self-location of the robot.

The **Deliberating** group of functionalities is designed to consider not only the requested action but also the final pose[1] intended to better compute a path plan.

Lastly, the **Reacting** portion of functionalities contemplates not only the action achieve by actuators but also the plan to reacting on a local level i.e. local planning.

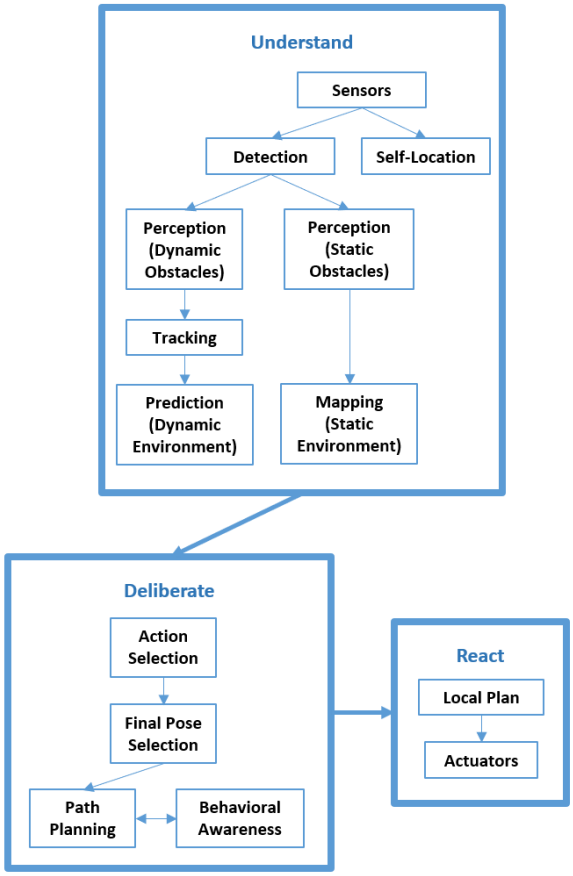


Figure 1: Schematic of relevant functionalities in agent-aware navigation
(adapted from [2])

1.2.1. Human Acceptance

HAN is a well-approached topic in robotic related literature.

Hallway settings are one of the most challenging to human acceptance of robot navigation. The arrangement of a long passage with doors leading to different rooms, a narrow corridor, implies a more careful approach to human-robot interactions as it stresses the importance of motion behavior to ensure humans' safety and comfort [3].

In [3] the acceptance level, of people with limited experience among robots, is determined by manipulating robot's velocity, lateral separation and start of maneuver distance between both agents, called the signaling distance, that shows the robot's intention of performing a maneuver.

A better response was obtained with higher robot speed, bigger lateral and signaling distance. Lateral

[1] The pose of an object combines its position and orientation

separation emerges as the elected crucial parameter.

The conclusions acquired reinforce the relevance of this work. Robot navigation with prediction of agent motion provide a faster response to a close encounter thereby enabling a larger lateral and signaling distance. Consequently, there is an increase in the acceptance level but also in safety, which is crucial not only to humans but all type of agents.

1.2.2. Navigating with dynamic obstacles

MECO research group members deliver in [4] the culmination of several researches done by the referred group regarding a spline-based motion planning method.

Even though the referred work deals with cooperative robots, the communication between agents doesn't give away complete paths. A prediction model is applied using current position, p_k , and velocity, v_k, to linearly compute the prediction in time, $p_k^{pred}(t)$, for obstacle k (equation 1).

$$p_k^{pred}(t) = p_k + t\, v_k \tag{1}$$

In order to avoid contact between agents, the condition obtained is for the robot's shape not to overlay any of the obstacles. In a dynamic environment with $Nobstacles$, consequently $k = 1...N$, the constraint condition is described by equation 2 [4].

$$dist(veh(q(t)), obs_k(p_k^{pred}(t))) \geq \epsilon_t \, , \, \forall_t \in [0, T] \tag{2}$$

Where ϵ is a considered safety factor intended to assure a secure separation between robot and dynamic obstacles. By minimizing T the time-optimal path is found.

Although this provides a good base for an obstacle avoidance method supported by prediction, the safety factor appears unreliable when considering unknown agents' shape. In addition, if the agents change velocity with time, the simple linear prediction will fail.

This work intends to counteract the indicated fault. Computing k obstacle uncertainty for $k = 1...N$ based on set of previous velocities leads to a more accurate and directed safety radius, that will increase for more hesitant agents and decrease for stabler ones consequently aiding the human acceptance level.

2. Background

Detection, perception, planning, tracking and predicting competences are crucial skills, needed to overcome the challenges imposed to a robot deployed in a populated domestic environment scenario.

2.1. Detection

The process of detecting the surroundings is essential to provide the robot with information about the state of the environment and the changes that may occur. Therefore, robots are equipped with a variety of sensors.

Even thought, robots are commonly provided with RGB (Red, Green, Blue) Cameras, laser range finders and scanners, that yield point clouds, this work will only consider data from the front and rear laser range finder. By contemplating only the measures obtained with the lasers, the results obtained can be applied to simple systems, only provided with the type of information attained from this type of sensors.

2.2. Motion Planning

Once the environment is characterized the robot is able to navigate. A plan of intended motion must be determined to lead the robot to the desired location. Generating the action sequence that allows the accomplishment of a given goal is called planning [5].

The environment's state not only depends on previous actions and states but also influences future ones. In order to achieve a feasible and efficient plan, the systems should account for constraint satisfaction, carrying along and reasoning about the variation of world states with time, as different actions take place.

2.2.1. Global Planners

Global planners design a plan to navigate through the environment based on a global model of it and taking into account given constraints and goals [6].

Occupancy grid mapping is a method from which a map of the environment can be built from noisy and ambiguous data, with an established robot pose. The key idea is to partition the world into cells, discrete regions, encode and store in each one the probability of being occupied by an obstacle.

The simplest way to navigate would just comprise mapping the environment and globally plan a path to the goal considering given constraints. Consequently, if an unexpected path obstructing obstacle appeared, sensors would detect it and a new global plan would have to be computed with the new map of the environment. If an obstacle is detected too late, meaning too close for the robot to have time to recompute the plan, stopping may be required.

2.2.2. Local Planners

Since the environment can change while the robot is navigating, it is imperative to have a planner that analyzes if there is any change at the local level that could impair navigation and, if so, adapt.

This reactive property of local planners enables the avoidance of unexpected obstacles while still considering the convergence to the given goal [6].

The **Dynamic Window Approach (DWA)** is amongst various approaches to local planning.

This reactive approach incorporates robot's dynamics, considering only attainable velocities and accelerations within a brief time interval thereby reducing the search space [7].

An optimal solution is selected from the valid search space obtained, the heading and velocity that allows the robot to reach the goal with maximum clearance from obstacles.

With the aid of local planners, the process of navigating is improved. If sensors detect an unexpected path obstructing obstacle, instead of new global plan, a local plan would be computed thereby shortening the computing time and complexity. Once again, the need to stop may be required if the obstacle is detected too late.

2.3. Perception

The ability to become aware of obstacles and perceive them as static or dynamic becomes essential when navigating through crowded environments.

Since the environment this work considers can be populated with humans and robots, and considering that the latest can be presented in almost any shape and size, the perception of such with unknown dimensions implies perceiving any moving obstacle.

Considering this work prediction computation relies on a previous track of the agent, whose shape and size is unknown, the algorithm integration implies detecting, perceiving and tracking undifferentiated moving obstacles. Consequently, and in order to disregard the problem of wrongful perception, all detected points, received from scan data, are considered to later track and predict the overall surrounding change.

2.4. Tracking and Predicting

The environment is expected to suffer changes even while the robot is navigating. This work acknowledges the usefulness of predicting the next state of the environment in the motion planning process. In order to do so, obstacles must be not only perceived but also tracked enabling the study and prediction of their motion.

Among varied methods capable of tracking points, the **K Nearest Neighbor** algorithm determines for each given point the k nearest elements of a group. When utilized between frames the Nearest Neighbor algorithm is capable of returning the smoothest transition possible, minimum feasible distance and velocity [8].

The correspondence between points can also be determined by analyzing the previous behavior and predicting the next point.

When using **Statistical Methods** to predict the next location of an obstacle, the tracking is completed by finding the point matching that maximizes probability. In other cases, the correspondence can be obtained by minimizing the distance between the detections and previous predictions [9] [10].

The challenge concerning prediction arises from the stochastic property of the environment, there is uncertainty in the behavior of a mobile agent.

By discretizing the environment in time, the properties of each state can be analyzed and utilized to predict the following states.

The navigation process is expected to be improved with an accurate prediction of the next states of the environment. While sensors keep scanning the environment, the predictions of the next states are updated. If the a close obstacle is predicted a local plan is computed.

If the prediction algorithm displays good results, the belief is that no obstructive obstacle will ever come as close as it would come with the previous mention motion planning methods. Consequently, by delivering a constructed predictions map to the local planner an early response would be obtained and the robot wouldn't need to stop.

2.5. Stochastic processes

In probability theory, stochastic processes are described as systems evolving with time represented by some variable subject to random fluctuation.

Even with the randomness inherent to some systems, the sequence of random events characterizing them can be described, in different settings, by stochastic processes such as the Bernoulli, Wiener and Poisson process [5].

The Wiener process is a well-known continuous-time stochastic process commonly associated with BM theory as it can be used to study particles' erratic motion analysis [11].

3. Approach and Implementation

Bearing in mind the concepts addressed, this work focuses primarily on developing a prediction method. The intended algorithm is based on agents' current position and velocity data.

A common approach to prediction algorithms is to use the referred required data obtained directly from a tracker. Trackers return the tracked obstacle position in different time intervals, $x(t_1)$, $x(t_2)$, $...\, x(t_n)$. The velocity can be estimated from $V = \frac{\Delta x}{\Delta t} = \frac{x(t_{i+1}) - x(t_i)}{t_{i+1} - t_i}$.

3.1. Prediction Computation

Using obstacles' current position and previous velocity knowledge the prediction of motion can be computed. There are various manners to process data.

A parametric model is defined by a finite set of parameters as opposed to a non-parametric model.

Typically a non-parametric model is based on arbitrary sets of data. More data is expected at every step and consequently this approach appears to be more promising in terms of results but computationally more complex.

Two prediction models are developed, based on parametric and non-parametric approaches. Later on, this work compares both and reasons about performance.

3.1.1. Brownian Motion

BM is the chosen parametric approach in this work. Since BM was originally used to characterize erratic movements, which is not the case of common human motion, applying it to agent positions would just result in a random and not very good prediction of movement.

While considering the constant speed and direction tendency in motion, variations in both are acknowledged. A initially stationary agent has zero velocity that increases when motion is initiated. When stopping at a goal, agent velocity decreases until it reaches zero. The concept behind BM can aid in the characterization of motion fluctuations.

In 2D agent motion BM is applied to describe velocity, v, in x and y *coordinates*, v_x and v_y respectively. Nevertheless, computing a variance based on physical constants becomes an even more complex problem when referring to humans walking. In addition, this work intends to still consider the particularities referred in agent motion.

Therefore, and since there is access to previous data, the Gaussian distribution used to describe the velocities adopts mean, μ, and variance, σ^2 determined by it.

$$V_{bx} \sim \mathcal{N}(\mu_{v_x}, \sigma^2_{v_x}) \tag{3a}$$

$$V_{by} \sim \mathcal{N}(\mu_{v_y}, \sigma^2_{v_y}) \tag{3b}$$

Equation 1 is adapted to (4) in order to obtain a prediction of the next positions, at $t+\Delta t$, based on the previous ones, at t.

$$x(t+\Delta t) = x(t) + \Delta t\, V_{bx} \tag{4a}$$

$$y(t+\Delta t) = y(t) + \Delta t\, V_{by} \tag{4b}$$

In Gaussian distributions, $X \sim \mathcal{N}(\mu, \sigma^2)$, the expected value corresponds to the mean of the distribution, $E(X) = \mu$.

The deviation of a random variable from its mean variance is called the standard deviation, σ. The variance is the square of the standard deviation, σ^2.

When relating two random variables it is useful to account for the relation between them, this measure is called covariance (5).

$$Cov(X,Y) = E(XY) - E(X)E(Y) \tag{5}$$

Considering the advantage of not only analyzing both velocity components but also reasoning about the variation between them, a prediction contemplating covariance, instead of only variance, measures would result in better conclusions.

For this work problem the covariance matrix to be computed is defined as shown in (6).

$$\sum = \begin{bmatrix} \sigma^2_{v_x} & Cov(v_x, v_y) \\ Cov(v_x, v_y) & \sigma^2_{v_y} \end{bmatrix} \tag{6}$$

To improve the solutions (4) would obtain, the prediction of next positions is computed considering the multivariate normal distribution dependent of the covariance matrix, $V_b \sim \mathcal{N}(\mu_v, \Sigma)$ with $\mu_v = \begin{bmatrix} \mu_{v_x} & \mu_{v_y} \end{bmatrix}^T$, equation 7.

$$\begin{bmatrix} x(t+\Delta t) \\ y(t+\Delta t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \Delta t\, V_b \tag{7}$$

The application of this algorithm returns not only one predicted point but a set of possible next positions, Figure 2. The ellipse shaped point cloud formed by this set of points is considered to give safety margins from the most probable predicted point, the center of the ellipse ($x_c(t)$, $y_c(t)$).

$$\begin{bmatrix} x_c(t) \\ y_c(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \Delta t\, \mu_v \tag{8}$$

Since the distribution is computed with the covariance matrix of previous data, the uncertainty inherent to the agent is acknowledged by this margin. With a more uncertain agent, the variance is higher and the points are more scattered. The covariance delivers information about how the data is spread.

3.1.2. Metropolis Algorithm

For the development of a non-parametric approach, MA is chosen algorithm to be used in combination with KDE.

The Metropolis-Hastings algorithm is a MCMC method. This algorithm generates several sample values from a proposed distribution and an initial guess. An evaluation based on a function $f(x)$, proportional to the density of the desired distribution, is performed to determine the acceptance of new samples or the continuity of the previous one. After several iterations, the burn-in phase, the distribution of the accepted values approximates the desired distribution [12].

MA is the particular case of the Metropolis-Hastings algorithm where the proposal distribution is symmetric.

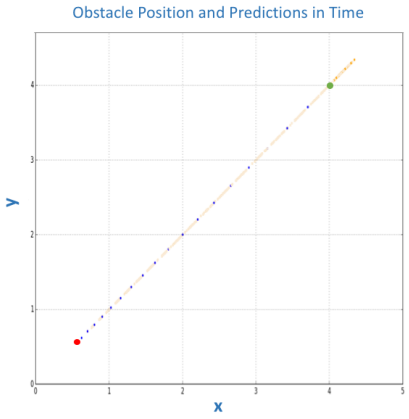


Figure 2: Predictions computed with BM
Red and green dots represent start and end point, respectively.
Blue points represent the remaining obstacle positions.
Yellow dots represent the next expected points.
The strongest yellow being assigned to the last set.

Bearing in mind human motion preference for constant speed and direction, the proposal distribution to apply MA in this work is a normal distribution with mean and variance computed from previous velocity data.

Once again 2D agent motion is considered and the velocity, v, is decomposed in x and y coordinates, v_x and v_y respectively.

The initial guesses, V_{0x} and V_{0y}, for computing the predicted next velocities are the last registered ones.

New possible values, V_{1x} and V_{1y}, are obtained by sampling from a distribution $\mathcal{N}(\mu_v, \sigma_v^2)$.

The acceptability, β, of a new sample is evaluated based on a function proportional to the density of the desired distribution. This work obtains the mentioned function by computing the KDE of previous velocities.

$$\beta = \frac{KDE(V_1)}{KDE(V_0)} \tag{9}$$

A value, γ, is generated from a uniform distribution with minimum and maximum values of 0 and 1, $U(0,1)$. If γ is lower than β the new V_0 becomes V_1, otherwise V_0 value is maintained.

After the burn-in phase, the fluctuations of V_0 values reduce and are registered. When the cycle of iterations ends the average of the registered values, V, is the estimated velocity.

Kernel Density Estimation

Based on provided data samples, KDE estimates the given data underlying function [13].

To compute KDE, distance between data samples, z_i with $i = 1, \dots n$ considering n samples , and desired point, z, is weighted, by function $\hat{f}(z)$ (10), resulting in the assignment of greater density estimation to regions with higher concentration of samples.

$$\hat{f}(z) = \sum_i \alpha_i K(z - z_i) \tag{10}$$

Uniform weights are commonly considered, $\alpha_i = 1/n$.

Different kernel functions, K, can be used to compute the KDE, producing distinct estimates, and avoid the storing of complete data. Bearing in mind all the mentioned considerations about normal distributions and taking into account this is the typically used form of K function, the computed KDE in this work uses a Gaussian form of K.

β is obtained from computing equation 9. The MA and KDE combination creates a set of n probable next velocities, $V_i = [V_{xi}, V_{yi}]^T$ for $i = 1 \dots n$. The predicted positions are obtained with (11).

$$\begin{bmatrix} x(t+\Delta t) \\ y(t+\Delta t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \Delta t\, V_i \tag{11}$$

Similarly to description given for BM, the predictions form an ellipse shape cloud of points, Figure 3. The data spread center, $(x_c(t), y_c(t))$, is computed with an identical equation, (8), considering the velocity values obtained mean.

From the obtained data both variance and covariance are determined and deliver information about orientation and spread of data.

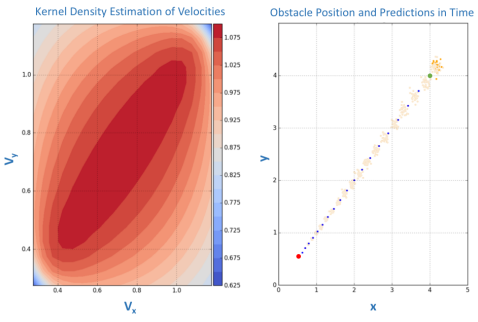


Figure 3: Predictions computed with KDE and MA
The vertical scale bar represents the logarithmic scale applied to the velocities' density function. Regarding obstacle position, red and green dots represent start and end point, respectively.
Blue points represent the remaining positions.
Yellow dots represent the next expected points. The strongest yellow being assigned to the last set.

3.1.3. Uncertainty

The computed set of predicted points gives the idea of an ellipse shaped confidence region of predictions around the center, $(x_c(t), y_c(t))$. Nonetheless, analyzing regions defined by set of points is complex and computationally that is not the best way to determine safe margins.

To directly compute the region of uncertainty it is necessary to analyze the covariance matrix of this problem (6).

Spread and orientation of data is described by the covariance matrix. Eigenvectors of Σ represent the directions by which the data is scattered with variance equal to eigenvalues of the matrix.

The eigenvalues, λ, and corresponding eigenvectors, e, of Σ are computed from equations 12 and 13, respectively.

$$det\left(\begin{bmatrix}\sigma_{v_x}^2 - \lambda & Cov(v_x, v_y)\\ Cov(v_x, v_y) & \sigma_{v_y}^2 - \lambda\end{bmatrix}\right) = 0 \tag{12}$$

$$(\Sigma - \lambda I)\,\mathbf{e} = 0 \tag{13}$$

Confidence Interval

Confidence intervals (CIs) return ranges at which the desired proportion of samples, assigned by setting a confidence level is believed to be.

In a normal distribution, as (3), with known standard deviation, σ, the CI for n observations is computed by firstly manipulating the known distribution to match $\mathcal{N}(0,\,1)$, equation 14.

$$\frac{\overline{V} - \mu}{\frac{\sigma}{\sqrt{n}}} = Z \sim \mathcal{N}(0,\,1) \tag{14}$$

Where $\overline{V}$, called the sample mean, is a point estimator of μ, the population mean.

The intended CI for a confidence level defined by $(1-\alpha)100\%$ can be computed by determining the values of Z_1 and Z_2 that validate equation 15.

$$P(Z_1 \leq Z \leq Z_2) = 1 - \alpha \tag{15}$$

Considering $Z \sim \mathcal{N}(0,\,1)$ it is comprehended that $Z2 = -Z1$, commonly symbolized as $Z_{\alpha/2}$. By manipulating equations 14 and 15 considering the properties of Z, the CI is obtained.

$$P(-z_{\alpha/2} \leq \frac{\overline{V} - \mu}{\frac{\sigma}{\sqrt{n}}} \leq z_{\alpha/2}) = 1 - \alpha \tag{16}$$

Finally, the ellipse is placed centered in $(x_c(t),\, y_c(t))$ and rotated to match the eigenvectors.

Bayesian Credible Interval

The distribution obtained from using MA and KDE is not a normal distribution and the samples extracted from it are not independent, otherwise using Central Limit Theorem an approximation would be possible. Consequently, *Bayesian Credible Intervals (BCI)*, analog to CI, are applied.

BCI, $[a, b]$, is estimated by computing a and b values that validate equation 17.

$$\int_{-\infty}^{a} f(\theta|v^n)d\theta = \int_{b}^{\infty} f(\theta|v^n)d\theta = \frac{\alpha}{2} \tag{17}$$

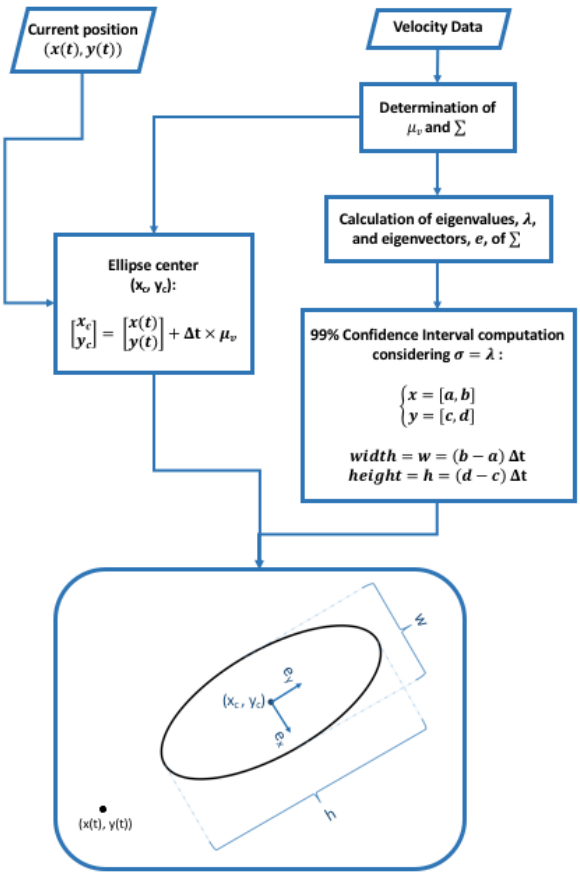


Figure 4: Schematic of BM implementation

By setting the desired credibility level, $(1-\alpha)\,100\%$ analog to the confidence level in CI, the credible interval $[a, b]$ is obtained.

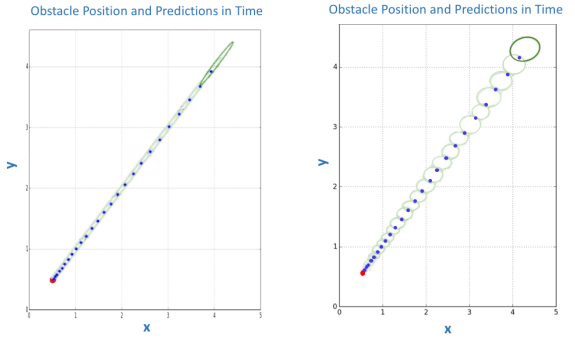
$$P(\theta \in [a, b]\,|v^n) = \int_a^b f(\theta|v^n)d\theta = 1 - \alpha \tag{18}$$

3.2. Simple Algorithm Integration

The developed algorithm is implemented in Python programming and available libraries, integrated in ROS framework.

Even though the developed algorithm is adequate for a wider range of applications, in this work it is applied in a robot equipped with a RGB Camera, a (13.5 cm above) ground sensor, laser range finder, and scanners. A map is previously built and stored using the referred sensors in a preceding scan of the environment.

Bearing in mind that the robot is deployed in a domestic environment, the obstacle avoidance problem can be tackled in 2D, disregarding the *z-coordinate* (perpendicular to the ground). All obstacles are considered to rise from the ground, avoiding collision in 2D, x and y *coordinates*, still guarantees a collision free path.



(a) Predictions computed with BM and CIs

(b) Predictions computed with MA, KDE and BCIs

Figure 5: Computation of predictions with uncertainty ellipses
Red dots represent starting points, and blue represent the remaining obstacle positions.
Green ellipses represent the ellipse where expected points are predicted to be.
The strongest green being assigned to the last computed ellipse.

3.2.1. Coordinate frames

While deployed, robot transmits various types of data that are often relative to different referenced coordinate frames. The tf library is used to track them.

Frame to frame transformations are provided and represented by a position vector and a quaternion. In order to apply the algorithms described, all registered data is adapted to into the same coordinate system. A map fixed frame is advantageous as base frame, it allows a more intuitive perception of the environment state.

3.2.2. Prediction Algorithm

Both prediction algorithms described in receive as input velocity and current position data.

An obstacle prediction model estimates the environment changes in the following time frames.

BM implementation utilizes *stats* from *scipy* Python library, while MA computes the KDE based on *KernelDensity* from *sklearn.neighbors* Python library while also taking advantage of *stats* for computation of BCIs.

3.2.3. Relevant Time Interval

Current positions and previous velocity data are used to predict forthcoming motion.

The extent of data to be processed increases with time resulting in a computationally slow algorithm. In addition, the relevancy of past velocities in prediction calculation has to be taken into consideration.

A relevant time interval of data is defined and the data is excluded accordingly.

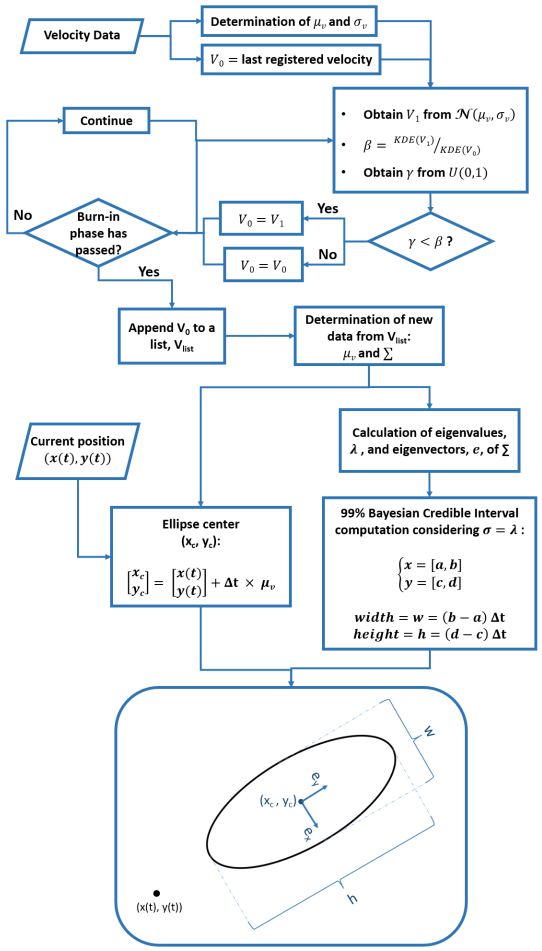


Figure 6: Schematic of MA + KDE implementation

3.2.4. Predictions' Map

A predictions' map is built so that the local planner receives environment estimated predictions. This map describes the occupancy of the space by partitioning the environment into cells and attributing occupancy percentages, 0 or 100% for unoccupied or occupied respectively.

A cell is assumed occupied if the center of the defined cell is contained in any of the ellipses computed by the prediction algorithm.

3.3. Extending the Prediction Approach

Even thought using previously built trackers is an appealing solution for integrating the prediction algorithm, both human and robots are considered discarding most common trackers.

3.3.1. Scan Processing

Scan data is processed considering the required transformations and the position of every unobstructed within range obstacle is registered. Since the position data is transformed to match the *map frame*, the velocities can be directly computed as

well.

3.3.2. Multiple Obstacles

The algorithms used consider all data scanned by the laser range finders. To compute the velocities, obstacle correspondence is required between time frames.

In order to arrange obstacles and their correspondence in time, the *K Nearest Neighbor* method is applied.

Two different groups of points, in separated time frames, are inputed and the distance between them is computed. Each point is labeled with the found corresponding nearest points from the opposite group. Euclidean distance is used as the criteria for finding the closest points.

Clustering

When implemented in a real environment deployed robot the amount of data and detected obstacles, every point detected by the laser range finders, is immense. In order to reduce computational time, a clustering algorithm was utilized.

Each cluster, considered an obstacle, it is tracked and the next position is predicted with adequate uncertainty. The uncertainty margin given to each cluster is then attributed to each of the points composing the corresponding cluster.

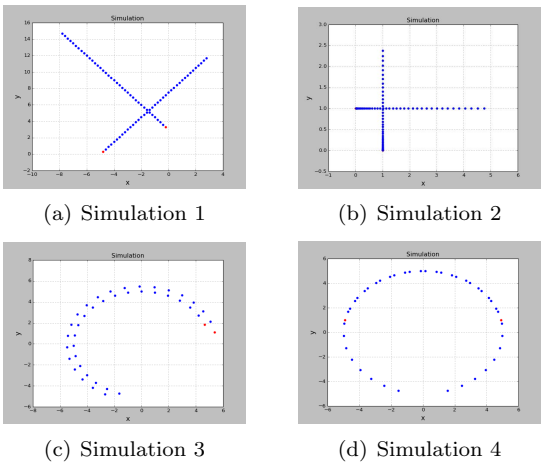
4. Evaluation

The analysis of algorithms performance is based on key factors such as computational time the prediction success. Evaluation of prediction success considers not only the validity of predictions, meaning the predicted points fall inside the predicted ellipse, but also the created ellipses' area. Various simulations, considering single and multiple obstacles, and real setting scenario tests were computed to better evaluate the result of both approaches.

The single obstacle simulations performed consisted of five different motion types: uniformly described, uniformly accelerated, circularly described, uniformly accelerated circular and random motion of an obstacle.

Multiple obstacle simulations intended to replicate 4 possible situations: two crossing moving obstacles with uniformly described motion, two uniformly accelerated obstacles intersecting, two obstacles with uniformly describing circular motion side by side and two obstacles with uniformly accelerated circular motion in a crossing situation, Figure 7 (a) to (d) respectively.

In order to test the performance of the completed algorithms in more probable scenarios, several dynamic environments are replicated. Scenarios such as static robot with agent walking in the surroundings, robot and agent moving in the environment, two agents crossing paths and walking close to each other in the surrounding space of a stopped robot are set.



(a) Simulation 1 (b) Simulation 2

(c) Simulation 3 (d) Simulation 4

Figure 7: Simulations with multiple obstacles
Red dots represent starting points, and blue represent the remaining obstacle positions.

The performance evaluation of both approaches is based on the analysis of both computational time and likelihood. Higher likelihoods, and consequently higher log-likelihoods, imply better data modeling. Considering log-likelihood values depend on data and the tests performed are very diverse, the overall model comparison is achieved by using the *likelihood ratio*, $R_{\mathcal{L}} = \frac{\mathcal{L}_{BM}}{\mathcal{L}_{MA}}$. Since the tests applied are for different time intervals of recorded data, a simple comparison of time will not suffice for applicability conclusions. Instead, $\frac{\Delta t_{model}}{\Delta t_{simulation}}$ is determined.

4.1. Results

The obtained results regarding multiple obstacle simulations are displayed in Table 4.1.

Simulation	1	2	3	4
$\frac{\Delta t_{BM}}{\Delta t_{sim}}$	1,3	1,4	1,9	1,3
$\frac{\Delta t_{MA}}{\Delta t_{sim}}$	837,7	888,3	2608,6	644,5
$R_{\mathcal{L}}$	1,7	0,9	0,9	0,7

Table 1: Obtained Results for Multiple Obstacles Simulations (5 seconds relevant time interval considered)

These results prove that, as expected, BM is a much faster method than MA. Nonetheless, the likelihood ratio indicates similar quality of prediction results for both methods.

The obtained results regarding real scenarios required cluster implementation due to the great amount of data collected by the laser on these kind

of dynamic scenarios. The cluster factor parameter designates the averaged number of points contained in each cluster.

The results obtained went as predicted. In a real scenario with higher amount of data both methods suffer in computational time. Nonetheless, computational time is reduced when higher cluster factors are utilized.

In addition, the results indicate that BM can still be considered a possible approach in real time utilization contrary to MA. Likelihood ratio indicates clear superior quality of prediction results using BM.

4.2. Predicted Costmaps

The obtained costmaps display both static and moving occupied areas, with higher dispersion of points. The static areas are associated with fixed obstacles. Consequently, the prediction has little uncertainty and the computed predictions will point to the real location of the obstacles. Areas with higher dispersion of points are associated with motion, dynamic obstacles and the associated uncertainty.

5. Conclusions

The results obtained take into account not only the computational time but also the prediction's validity and relative quality between approaches. In addition, point prediction validity is considered.

The concluded superior method is the BM. Not only BM is the method that displays superior prediction success but also the faster response.

In real setting tests, MA algorithm takes, in the best case scenario, more than 1000× the simulation time. Consequently, it would not be a useful method in real-time implementation as desired.

Considering a prediction map is built to later influence the local planner, the success of the developed work implies an enhanced planner response. Consequently, navigation is improved.

The major achievements of the present work include the development of an agent motion model that accounts for uncertainty and the formulation of an agent predictor valid for both static and dynamic obstacles.

5.1. Future Work

As possible further development of this work, it is suggested the experimental evaluation of the predicted costmap on the local planner response.

In addition, an optimization of the coded algorithm would possibly result in an improvement on the algorithms' computational time.

An interesting evaluation suggested is to test the developed methods on a robot with higher laser scan range. The agents would be detected earlier, consequently a better and faster response would be possible.

References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[2] A. Sánchez López, R. Zapata, and M. A. Osorio Lama, "Sampling-based motion planning: A survey," *Computación y Sistemas*, vol. 12, no. 1, pp. 5–24, 2008.

[3] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Human-robot embodied interaction in hallway settings: a pilot user study," in *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pp. 164–171, IEEE, 2005.

[4] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, 2017.

[5] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, no. 27, pp. 79–80, 1995.

[6] E. Rimon, I. Kamon, and J. F. Canny, *Local and Global Planning in Sensor Based Navigation of Mobile Robots*. Springer, 1998.

[7] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[8] K. Fukunaga and P. M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," *IEEE transactions on computers*, vol. 100, no. 7, pp. 750–753, 1975.

[9] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[10] D. Reid *et al.*, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[11] M. Kac, "Random walk and the theory of brownian motion," *The American Mathematical Monthly*, vol. 54, no. 7, pp. 369–391, 1947.

[12] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[13] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, 2002.