Learning How to Increase the Chance of Human-Robot Engagement

Douglas G. Macharet¹ and Dinei A. Florencio²

Abstract—The increasing use of mobile robots in social contexts makes it important to provide them with the ability to behave in the most socially acceptable way possible. In this paper we investigate the problem of making a robot learn how to approach a person in order to increase the chance of a successful engagement. We propose the use of Gaussian Process Regression (GPR), combined with ideas from reinforcement learning to make sure the space is properly and continuously explored. In the proposed example scenario, this is used by the robot to predict the best decisions in relation to its position in the environment and approach distance, each one accordingly to a certain time of the day. Numerical simulations show a significant performance improvement when compared with a random technique. The robot is able to improve performance after just one day of interaction (a few dozens of trials), and achieves the maximum expected value for the proposed approach within sixty days.

I. INTRODUCTION

It is widely expected that the use of mobile robots in different parts of society will be commonplace in the near future. This change from controlled environments (e.g., factories) to unconstrained environments where people are constantly present (e.g., home, public places, hospitals, etc.) will require robots to behave in “socially acceptable” ways.

This need for socially acceptable behavior crosses many domains (e.g., can I make noise now? how fast can I move and people still feel safe? can I cross in front of someone? behind?). While behavior in a social space co-occupied by humans brings a number of issues, the direct interaction with people is particularly challenging, as the state of mind of the person is hard to estimate — and may change with the interaction itself. Thus, even a simple decision as whether to initiate interaction is challenging. The simplest approach is to be conservative and never initiate interaction. Depending on the role played by the robot this may be acceptable. For example, for a receptionist robot in a building, it may be acceptable to wait for humans to start the interaction. However, it is not clear that result would still hold if we change any of the parameters of the experiment. Changing the robot size, facial expression, the country, the type of mall, or the task, would likely affect the optimum distance. Thus, instead of making the problem more specific, we focus on the learning process itself. If we show how the robot can learn, the same learning process is much more likely to apply, even after any of the conditions of the experiment change.

The scenario we chose to simulate is that of a robot trying to distribute a flyer for a new store opening at a shopping mall (illustrated in Figure 1). The probability of success is influenced by a number of variables, in this work we consider three: time of day, approaching distance, and initial position. All other non-modeled variables are lumped together in the noise component. We also assume the robot will know the outcome of the trial immediately after each approach (success if the customer took the flyer). We then measure the ability of the robot to learn the influence of variables under its control (robot’s position, distance to person before engaging) by themselves and as a function of the variables that are observable, but outside its control (time of day, rate of arrival).

The robot’s objective is to tune the variables under its control to maximize the chance of engagement. Results will, of course, vary widely with the scenario. Over our specific

This work was done while Douglas G. Macharet was with Microsoft Research.

Douglas G. Macharet is with the Computer Vision and Robotic Laboratory (VerLab), Computer Science Department, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil doug@dcc.ufmg.br

Dinei A. Florencio is with Microsoft Research, One Microsoft Way, Redmond, WA, USA dinei@microsoft.com
that exists in human-human interaction can also be applied to human-robot interaction scenarios. Considering this, some works incorporate this personal space model in the path planning step [14], [15], [16]. However, this model is not static and can vary accordingly to different aspects, such as previously experience with the robot [17], or functional noise of the robot [18]. Therefore, it is important to adapt this model, increasing the chance of a successful approach.

A robot is a very dynamic system which demands for techniques that are capable of adapting along time, making learning methods a good choice in this scenario. However, it is also a high dimensional space, and data is very sparse. This creates learning needs which are very specific. As Morimoto et al. put it, “It would be difficult to naively apply the existing machine-learning methods to these robots” [19]. Thus, most of the work in this context has as the main focus the low-level kinematic control of the robot [20], [21], [22], [23], [24]. When dealing with low-level kinematics, data is not as sparse, and can often be further augmented by physical simulation (e.g. “envisioning”). Learning has also been used to train “human-like” trajectories in crowded environments [25], while [26] investigates the influence of the trajectories on pedestrians’ comfort. Some of our own previous work [27], [28] addresses the interaction between people and the robot trajectories, focused on a telepresence scenario. Considering learning in the human-robot interaction scenario [29] presents a reinforcement learning approach which uses discomfort signals from the human as the reward function, making it possible to learn how to approach in socially acceptable distances. Two recent surveys are also relevant, with [30] (and the associated special issue [19]) focusing on several aspects of robot learning, and [31] focusing on the application of reinforcement learning in robotics.

III. METHODOLOGY

A. Problem Formalization

As stated in the previous sections, the machine learning aspect of our problem consists on making a robot learn how to approach a person in order to increase the chance of successful engagement.

The robot is able to choose a certain position \( p \) on the environment from a discrete set of allowed positions \( P \). It can use the time of the day \( t \) to make this decision. When a person is within the range of perception of the robot, it has to decide a distance \( d \) at which it will approach (i.e., start talking to) the person. The result of each approach trial is either successful or unsuccessful. We assume no cost on approaching, thus the robot will approach every person possible. Therefore, we can define the problem at hand more formally as:

\[
\text{Problem 1: The result } r_i \text{ of each approach trial } T_i \in \mathcal{T} \text{ is a binary random variable (success of failure), with the probability of success being a function of the parameter vector } x, \text{ which includes robot’s position } p_i \in \mathcal{P}, \text{ the approach distance } d_i \in \mathcal{D} \text{ and the time of the day } t_i \in \mathcal{T}. \text{ Consider the subset of successful trials } \mathcal{T}_s = \{ T_i \in \mathcal{T} | r_i = \text{success} \}.
\]
Then, we want to optimize the success rate of approaches over the variables under the robot’s control, i.e., :

$$\text{maximize}_{\text{P, D}} \frac{|T|}{|T'|},$$

where $|\cdot|$ represent the cardinality of a set.

**B. Learning Framework**

It is natural to consider a reinforcement learning approach to tackle the proposed problem. However, most standard reinforcement learning techniques rely on discrete states [32]. The common approach of discretizing the variables quickly brings on an explosion of the learning space, making the number samples required for training impractical. The approach of fitting a parametric function requires a prior methodology is based on the Gaussian Process Regression (GPR) technique, which consists of a less ‘parametric’ tool, $x$.

$$(GPR) \text{ technique, which consists of a less 'parametric' tool,}$$

More specifically, each of the likelihood of the observed data as a function of the parameters observed datapoints. This can be done by maximizing the combination of the covariance function of each one of the $K$.

$$\text{likelihood of the observed data as a function of the parameters observed datapoints. This can be done by maximizing the combination of the covariance function of each one of the } K.$$

In the covariance function. More specifically, each of the likelihood of the observed data as a function of the parameters observed datapoints. This can be done by maximizing the combination of the covariance function of each one of the $K$.

$$(K) \text{ is completely specified by its mean function } (m(x)) \text{ and covariance function } (k(x, x')).[33]\text{. Thus, we approximate our random process } f(x), \text{ as a GP:}$$

$$f(x) \sim GP(m(x), k(x, x')). (1)$$

Additionally, we model the covariance function as a linear combination of the covariance function of each one of the state parameters, time of day ($K_t$) and approach distance ($K_d$), i.e.:

$$K = \alpha_1 K_t + \alpha_2 K_d. (2)$$

Note that this does not imply the variables are linearly related, but simply that the covariance of one does not change as a function of the other.

The position variable will be treated separately, therefore this variable will not be incorporated in the covariance matrix.

GPR involves two steps. Initially, we have to fit a covariance matrix related to the process, that best explain the observed datapoints. This can be done by maximizing the likelihood of the observed data as a function of the parameters in the covariance function. More specifically, each of the covariance matrices $K_t$ and $K_d$ corresponding to the specific vectors $x_i$ in the experiments

$$K(x, x) = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} (3)$$

is obtained by modeling $k(x, x)$ as

$$k(x_i, x_j) = \sigma_f^2 \exp \left( -\frac{(x_i - x_j)^2}{2l^2} \right) + \sigma_n^2 \delta_{ij}(x_i, x_j), (4)$$

where $\delta_{ij}(x_i, x_j)$ is a Kronecker delta which is one iff $i = j$ and zero otherwise. The squared exponential function has some characteristic parameters such as the maximum allowable covariance ($\sigma_f^2$), a length parameter related to the separation of the observations ($l$) and a parameter related to the process noise ($\sigma_n^2$).

The next step consists in predicting the mean and variance of the process accordingly to the observations vector $y$ as:

$$\bar{y}_s = K_s K^{-1} y (5)$$

$$\text{var}(y_s) = K_{ss} - K_s K^{-1} K_s^T (6)$$

where $y_s$ is the Gaussian process prediction (random variable) for the test input vector $x_s$, $\bar{y}_s$ is its mean and $\text{var}(y_s)$ its variance. The covariance matrices are $K = K(x, x)$, $K_s = K(x_s, x)$ and $K_{ss} = K(x_s, x_s)$.

Since the method is executed iteratively, it is necessary to use an efficient policy to choose the next state to visit in order to improve the model learned until the moment. Here we borrow from the reinforcement learning theory, and establish a balance between exploration and exploitation.

Considering the mean (Equation 5) and variance (Equation 6) previously presented, we first introduce in Equation 9 the policy regarding the position. Among all the possible positions, we choose the value with the maximum predicted mean with a probability $P_{\text{mean}}$, the value with the highest value on a 95% C.I. with a probability $P_{\text{ci}}$ or a random position otherwise.

$$p_{\text{mean}} = \arg \max_{\forall x \in \mathbb{P}} \max_{\forall i}[\bar{y}_s][i] (7)$$

$$p_{\text{std}} = \arg \max_{\forall x \in \mathbb{P}} \max_{\forall i}[\bar{y}_s + 1.96 \sqrt{\text{var}(y_s)}][i] (8)$$

$$p \sim \begin{cases} p_{\text{mean}}, & \text{if } r \leq P_{\text{mean}} \\ p_{\text{std}}, & \text{if } P_{\text{mean}} < r \leq P_{\text{ci}} \\ U(1, |\mathbb{P}|), & \text{otherwise} \end{cases} (9)$$

where $r \in [0, 1]$ is a random real number with uniform distribution.

We present in Equation 12 the policy regarding the distance. Assuming a position was chosen, we select some test values ($x_s$) and predict the possible values. Similarly to the previous step, with a probability $P_{\text{mean}}$ we select the value with the maximum mean, but then we choose a distance with a Normal distribution considering this value. The value with the highest value on a 95% C.I. with a probability $P_{\text{ci}}$ is selected, and distance is chosen again with a Normal distribution. Otherwise, a value is uniformly randomly chosen in the domain of the set of distances.

$$d_{\text{mean}} = \arg \max_{\forall x \in \mathbb{X}_s} \max_{\forall x \in \mathbb{X}_s}[\bar{y}_s] (10)$$

$$d_{\text{std}} = \arg \max_{\forall x \in \mathbb{X}_s} \max_{\forall x \in \mathbb{X}_s}[\bar{y}_s + 1.96 \sqrt{\text{var}(y_s)}] (11)$$
where \( r \) is a random real number with uniform distribution.

Figure 2 is presented to clarify the main idea behind the policies. The first case on the policies is responsible for choosing the best know value. The second case is used to increase the confidence in an area not yet explored. Finally, the third case is responsible for executing a global search. It is important to notice that choosing the distance based on a Normal distribution it is also executing a local search.

The Gaussian process prediction method has a basic complexity of \( O(n^3) \) due to the inversion of \( K \) (considering standard techniques), which can be prohibitive for large datasets. However, for the specific problem under consideration, datasets are typically small, varying from a few dozens to a few thousands. At this levels, computational complexity is not a problem. For situations where the robot approaches say, over 10,000 people, data consolidation or other methods for controlling the complexity may need to be used.

C. Simulation

The proposed framework was evaluated in a simulation context in order to measure the learning potential, and evaluate the amount of data needed to learn certain characteristics of the interaction.

Our arrival method is based on a non-homogeneous Poisson process, since we consider that it can vary along the day. The frequency of inclusion of new people in the environment is defined by a rate (intensity) function \( \lambda(t) \), which represents an estimate of the amount of new persons that must be created in a given unit of time.

The random length of time that will pass before the next region is inserted into the environment is obtained by generating random numbers based on a sample in the inverse transform in accordance with a cumulative distribution function [34]. The cumulative distribution function of a homogeneous Poisson process rate \( \lambda \) can be represented by an exponential distribution:

\[
F(x) = 1 - e^{-\lambda x}, \quad x \geq 0,
\]

with the inverse transform given by

\[
T = -\frac{\ln U}{\lambda},
\]

where \( U \in [0, 1] \) is a random real number with uniform distribution.

The event schedule of the non-homogeneous Poisson Process for each time slot \( t_i \) are then generated according to a homogeneous Poisson process with rate \( \lambda(t_i) \). Each event has a probability \( p_i \) of being added to the schedule given by

\[
p_i = \frac{\lambda(t_i)}{\max(\lambda(t))}.
\]

Considering an environment with \( P = \{1, 2, 3\} \), we propose for each position a model that represents the probability of successful engagement. The models are based on the time of the day (\( T = [0h, 12h] \) and approach distance (\( D = [0m, 5m] \)). Figure 3 presents the models, the maximum probability (red areas) in all positions is 90%.

Algorithm 1 presents a simplified overview of the execution of the framework. In line 2 an schedule of the arrival times are created using the Poisson process previously described. On each iteration of the loop the robot chooses a new position (line 4) based on the current time (\( t \)) using Equation 9. Line 5 verifies if it is time for a new arrival. When a new arrival happens the robot must initially choose an approach distance (line 6) using Equation 12. The approach tentative is then realized performed considering the position, distance and current time (line 7). The result is obtained selecting an uniform random number and comparing to the corresponded probability of the model. Based on the return the model is updated (line 8) based on Equations 5 and 6.

Algorithm 1 SimulationLoop()

1: \( t \leftarrow 0 \)
2: \( s \leftarrow \text{createArrivalSchedule}() \)
3: while stopping criteria not met do
4: \( p \leftarrow \text{choosePosition}(t) \)
5: if isTimeOfNextArrival(s, t) then
6: \( d \leftarrow \text{chooseDistanceToApproach}(p, t) \)
7: \( r \leftarrow \text{tryToApproach}(p, d, t) \)
8: updateModel(r) \end if
9: \( t \leftarrow t + \Delta t \)
10: end while

IV. NUMERICAL EXPERIMENTS

In this section we describe our experiments and the corresponding statistical analysis, showing the improvement obtained in the success engagement rate with the proposed...
methodology. The simulator was implemented using Matlab and all experiments were executed in a PC with an Intel Xeon 3.60 GHz processor, 8 Gb of RAM, and a 64-bit Windows OS. Table I presents an overview of the specific parameters related to the simulation execution.

**TABLE I**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma^2_f$</td>
<td>0.5</td>
</tr>
<tr>
<td>$l$</td>
<td>1</td>
</tr>
<tr>
<td>$\sigma^2_e$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha_1 = \alpha_2$</td>
<td>1</td>
</tr>
<tr>
<td>$p^\text{d}<em>{\text{mean}} = p^\text{d}</em>{\text{ci}}$</td>
<td>0.8</td>
</tr>
<tr>
<td>$p^\text{s}<em>{\text{mean}} = p^\text{s}</em>{\text{ci}}$</td>
<td>0.9</td>
</tr>
<tr>
<td>$\sigma^2_d$</td>
<td>0.25</td>
</tr>
<tr>
<td>Simulation</td>
<td></td>
</tr>
<tr>
<td>Number of days</td>
<td>60</td>
</tr>
<tr>
<td>Minimum arrival rate</td>
<td>$\frac{1}{30}$</td>
</tr>
<tr>
<td>Maximal arrival rate</td>
<td>$\frac{1}{10}$</td>
</tr>
</tbody>
</table>

Figure 4 presents the behavior of the engagement success rate along the days. The red line are the results given by a random policy (the position and distance are randomly chosen) which has a mean value of $\approx 13\%$. The blue line represents the results of an optimal policy, with a mean success rate of $\approx 61\%$. The black line are the results given by our proposed methodology. After a period of circa 20 days it is possible to observe a convergence of the results to a mean value of $\approx 46\%$, a significant improvement over the random policy.

As the proposed technique is a probabilistic method, in order to perform a thorough statistical analysis, we present next an overall analysis with a significant number of experiments. We run 300 experiments considering the same parameters used in the previous experiment.

Figure 8 presents the average success rate along the simulated period. As can be observed, the methodology achieves a success rate of $50\%$ in less than 30 days, about $4 \times$ more effective than the random policy. If we fit an exponential to the learning curve, we conclude that the methodology
has a “learning time constant” of 7 days. It is important to observe that the mean value obtained by the methodology will be lower than the maximum expected mean using the optimal policy due to the random values that are chosen accordingly to each policy (position and distance). Indeed, in 10% of the time it chooses a random position, and 10% of the time uses a random distance. Thus we estimate that this would reduce the success rate to approximately 52% (i.e., (.81)61% + (.19)13%). The additional 2% sub-optimality is most likely also derived from the random noise added to the distance. In any case, both could easily be removed if we allow the policy to decrease the rate of exploration as it learns.

Note also on Figure 8 the early improvement. We can estimate the success rate at the end of first day at around 18%, an improvement of almost 40% over the random policy after just a day.

Besides number of days, it is also important to evaluate the number of approaches necessary for convergence. Although we consider different arrival rates during the day, we assume they remain unchanged among days. Considering the values used for the minimum and maximum rate we obtained an average of ≈ 80 new approaches each day (Figure 9). Therefore, it has a “learning time constant” of ≈ 560 approaches.

V. CONCLUSIONS AND FUTURE WORK

This work proposed a learning framework for mobile robots in order to increase the chance of successfully initiate engagement in a human-robot interaction.

The proposed methodology is based on Gaussian Process Regression, combined with principles from Reinforcement learning. The use of GPR allows a rather simple model, while still being powerful enough to adequately represent...
the underlying phenomenon. In the simulated experiment, improvements of around 40% (over a random policy) were obtained after just one day (∼80 engagement trials), and nearly optimum results after 60 days. Although suitable for problems with a reduced number of training data (our case), the main drawback of the technique is its computational cost, which can be a problem for long-term executions.

Future research directions include the deployment of a robot in a real world scenario, a better study of the parameters used in the model (probably the use of an optimization process on every learning iteration of the algorithm), and the analysis of different learning policies, particularly regarding the trade-off between exploration and exploitation. The study of techniques with a lower (or even constant) computational complexity is also of interest, particularly for other scenarios, where the number of data points may become excessively large.

ACKNOWLEDGMENTS

The authors would like to thank Dan Bohus, Harsha Kikkeri and Ashish Kapoor for insightful discussions during earlier phases of the project.

REFERENCES