ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

Enhancing Path Planning Through Local Motion Predictions

Mr. Gangisetty Naveen Kumar

Assistant Professor, Department of CSE,

Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India

Abstract—

We provide a new strategy for long-range route planning that makes use of a learnt model to anticipate the results of local movements with perhaps incomplete information. What the model is trained with data consisting of unsupervised path acquisition. This part is used by route planners who rely on sampling to determine the quality of potential branch additions to the planning tree. We use two robots—a complicated, simulated quadruped robot (Animal) traversing tough terrains and a basic, actual differential-drive robot (Mighty Thyme) traversing unknown geometry and obstacles—to demonstrate the use of this pipeline. Finally, we demonstrate that planning yields logical routes by quantitatively assessing the model's effectiveness in forecasting the outcome of both local and long-range actions.

I. INTRODUCTION

SAMPLING-BASED PLANNING [1] is an effective and general route planning technique that frames the issue as the search for a series of viable local movements, each of which connects the starting point with the destination. Two states in close proximity to one another, with the first local motion beginning at the source state and the second local motion culminating at the destination state. A local motion estimator takes into consideration robot kinematics, local planners and controllers, and environmental information to estimate the feasibility of local movements.

In most cases, the local motion estimator is a somewhat straightforward component that only verifies that a direct move between two states does not result in a collision with any known obstacles and does not violate any of the robot's kinematic restrictions. But this method won't work when the robot's interactions with its surroundings are complicated and maybe random. Think of a robot with legs that has to map out a lengthy route through rough terrain it is familiar with. Another method of determining whether a local motion is conceivable involves using precise Follow the robot's virtual progress as it traverses the simulated landscape. Given the form of the terrain, the nature of the interaction between the robot and the terrain, the capabilities of the robot's sensors and its low-level controls, and the likelihood that the robot may lose traction. This having only incomplete information about the environment (for instance, we can know the terrain topology but not its softness/friction characteristics at planning time) and a high computing cost (a local motion estimator is queried extremely frequently, and hence has to be quick) are both problems. The major thing we bring to the table is a new way of doing things, in which we: develop an estimator to forecast the result (e.g., success probability, time, energy,...) of movements joining two neighboring states, given the available (potentially imperfect) information about the surrounding environment;

We identify possible local movements and give them a cost to be minimized (for example, finding the route with the least number of turns) using the learnt estimator inside a sampling-based global planning framework. with low risk of failure) given that (1) a local planner and controller are at our disposal, (2) our understanding of the environment is fixed (in particular, at planning time, the map covers source and target locations, and the robot does not acquire new information during trajectory execution), and (3) the robot cannot recover from setbacks and therefore does not require dynamic re-planning. In Section VI, we will quickly go through various options for easing these constraints. When training an estimator, it is helpful to gather examples in the form of input-output pairs, where the input is a set of neighbouring states and any accessible information about the surrounding environment and the output is the result of the action taken. These measurements may be taken either virtually or in the actual world, with the robot performing various man oeuvres in the target area or recording the results. The method may be done in a self-supervised [2] form if the robot can feel the result independently (for instance, by using visual odometer to monitor progress). May gradually adjust the estimator to fit new conditions this method works well for scenarios where the robot's actions will not always have the same result, known as a stochastic scenario. In particular, employing probabilistic machine learning [3] methods ensures that uncertainty in the estimator inputs is correctly transferred to uncertainty in the outputs, which is then dealt with in a systematic manner during planning.

II. RELATED WORK

A. Path Planning

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 18, Issue 1, 2024

Imitation learning from the paths of experts is a typical approach of incorporating machine learning into robotic route planning [4]. According to this line of inquiry, Pfeiffer et al. [5] teach a Given current readings from a laser scanner, a convolutional neural network (CNN) may calculate steering instructions to approach the desired target position; Ollas et al. [6] evaluate expert trajectories to learn properties of travelled map regions, and then compute cost maps in a Bayesian framework; Bagel et al. To get the ideal cost function for the expert trajectories, [7] use inverse optimal planning. In contrast, we use a sampling-based approach to graph search in our study. In order to deal with high-dimensional configuration spaces, sampling-based motion planning algorithms [1] depend on a local planner (or steering function) to establish the connection between nodes of a graph, allowing for control-space searching to account for dynamical or differential constraints. After a target state has been identified, the Rapidly Exploring Random Trees (RRT) [8] method builds a tree by sampling the configuration space to add edges that lead to viable states. Connects two nodes in a tree, the route between them is returned. A different approach is provided by the Probabilistic Road Maps (PRM) [9] programmed, which constructs a network of possible edges within which the optimal route may be determined using methods such as Dijkstra's algorithm. When given enough time to develop the graph, sampling-based planners are able to determine a route from source to destination with probability 1. Asymptotically, optimal versions like RRT* and PRM* converge to the route with the lowest cost [10]. Components of sampling-based planners have been enhanced using machine learning, for example, to automatically choose the appropriate sampler [11] or the optimum expansion strategy depending on the neighborhood of a node [12]. Here, we apply machine learning to a different part of the system—namely, the part that establishes the link between two neighboring states.

B. Search Spaces

Pathways that may be taken, or viable paths, are discovered by the use of certain planning procedures, which may, for instance, search for paths that avoid collisions or investigate the space of all possible states. To maintain a steady walk [13]. In contrast, action space is what planners look for when determining what to do next. Previously, we assessed the practicability of movements on a horizontal grid [14]. Instead, we randomly choose data from a much wider pool in our study.

C. Local Motion Estimation

One important part of global route planning algorithms is estimating the practicability of local movements. Cells in 2D maps, for instance, may be given reversibility scores that are useful for guiding legged robots. to take into consideration the robot's shape and the roughness and slope of the ground [15]; in a more involved technique, Fankhauser et al. [13] first calculate secure footholds, and then body and leg trajectories to traverse steep parts and high steps. When the robot's interaction with its surroundings is intricate, one may train a model to provide reversibility scores:

Before using a convolutional neural network (CNN) to calculate the confidence of retaining balance on prospective footholds, Wellhausen et al. [16] utilize a weakly-supervised technique to segregate terrain classes from a front-facing camera picture. Methods with similar goals have been developed, for instance in the LAGR project [17], which include the classification of terrain before assigning a cost to a grid map. Given a local planner, our proposed method first trains an estimator to predict which local motions are feasible through a large number of random trials on a variety of terrains; then, instead of constructing a fixed resolution grid map, it samples from the entire space of local motions to construct a dense planning tree. Moreover, it doesn't depend on carefully crafted models of the landscape to function, but rather it learns broad characteristics by seeing how they're used in context across different types of terrain. Terrains. Chiang et al. [18] suggest a similar method in their recent work for kin dynamically restricted robots, whereby a local planner taught with reinforcement learning is integrated into a global sampling-based planner. In imitation learning, the expert trajectories for a legged robot come from a complex high-dimensional footholds planner [7] rather than a human pilot, using the same basic concept of employing machine learning to imitate the outputs of a computationally costly planner.

III. MODEL

With the use of machine learning, we analyze the costs and viability of potential global route plans (see below). Fig. 2). Regional Activity (A.) The robot operates through a world about which it has only partial information K modeled at planning time.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

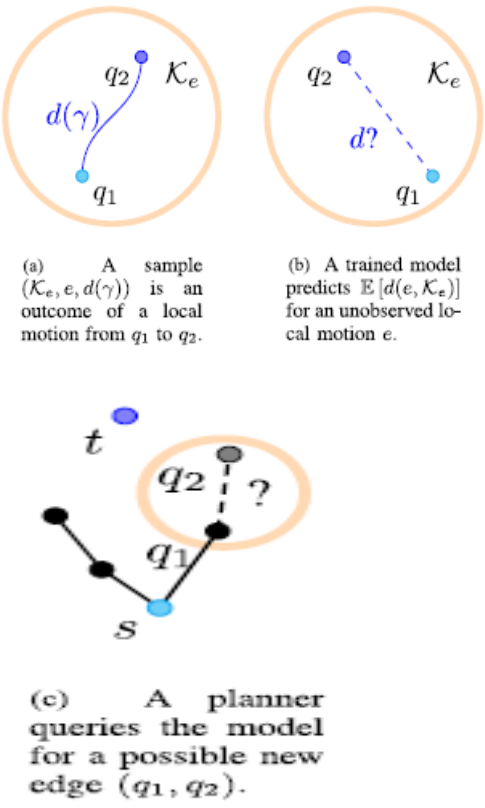


Figure 2: Definitions discussed in Chapter 3: To determine the likelihood of success and cost of candidate edges for a tree, a sampling-based planner (c) queries a model (b) trained using a dataset (a) of local motion outcomes. Establishing a link between s and t as sources and destinations. The model's utilization of expert knowledge, shown by the orange circle, is illustrated. Information. Two neighboring states in the robot's configuration space C create a local motion denoted by the equation e = (q1, q2) C2. Local motion might be represented as a pair of poses in the two-dimensional special Euclidean group SE (2), and knowledge as an obstacle occupancy grid, for a differential-drive wheeled robot navigating a plane with obstacles. Using its local planner and its controller, the robot moves along the time-parameterized trajectory: [0, T] C, where (0) = q1. As a rule, a number of Different trajectories can result from the same local motion being executed due to a number of factors, including: local planners that use random algorithms; incomplete knowledge of the environment (possibly based on the robot's sensors); stochastic interaction with the environment while executing the motion (such as when a robot slips, for example). We thus assume that is taken at random from a distribution that takes into account both the external conditions (environment, e) and the local information (). Key K, where K is the subset containing any information about he's surroundings; in the aforementioned example, local knowledge would be the piece of the occupancy grid map that includes the source and target postures.

B. Acquiring the Ability to Forecast Neighborhood Paths Through the application of supervised machine learning, we are able to foretell the result of a certain local motion (i.e., the predicted value of the trajectory descriptors). To be more specific, we are taught the mapping (e,Ke) y = E[d(e,Ke)] (see Fig. 2(b)). To train a machine learning model, we first amass a large dataset consisting of examples of local trajectories by randomly choosing local movements across contexts and noting the values of their descriptors d().

1) Dataset Collection: If the robot directly extracts the trajectory descriptors from on-board sensors, it may gather the dataset (shown in Fig. 2(a)) in an unsupervised fashion. The first algorithm is a random walk through trees (RRT) based planner that uses a model to forecast the predicted descriptors y of local motion trajectories and information K about the environment to calculate a route from s C to t C. When adding a cost function C to the tree, only edges with a success probability greater than are considered for inclusion.

Initialize $\mathcal{T}$ with a node in s
while $t \notin \mathcal{T}$ do
  $q \leftarrow$ random sample from $\mathcal{C}$ with a small bias toward t
  $q_1 \leftarrow$ vertex in $\mathcal{T}$ nearest to q
  $e \leftarrow$ local motion from q_1 toward q of at most length δ
  $y \leftarrow \mathbb{E}[d(e, \mathcal{K}_e)]$
  if $y_S > \tau$ then
    add edge e with cost $\mathscr{C}(y)$ to $\mathcal{T}$
  end if
end while

That of its interior. We choose a local coordinate frame, centered on q1, since the only information necessary for relevant descriptors is the relative location of q2 with regard to q1, and the local knowledge surrounding q1. Place q1 to stand in for q2 and Key. In the end, the dataset includes several examples of the type (Kq2, q2, d1, d2, . . . , dn). The Second Part Is Studying: The model produces a variety of results, some of which are used for classification and others for regression, since some of the descriptors are discrete (such as S) while others are continuous (such as time and energy). In Fig. 5, we depict the structure of the deep neural network we

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

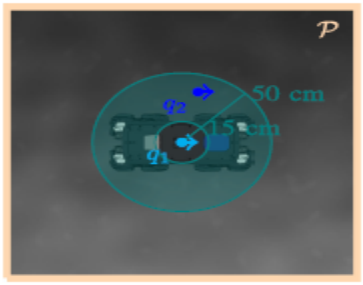
use as a local motion estimator for two distinct planning issues. Path Planning on a Global Scale The global route planner seeks the optimal path for the robot to take from s C to t C, represented as a series of local movements = (e1, e2,. Using the trained model's calculated descriptors, we may differentiate between trajectories. More specifically, we characterize the path's incremental cost as

$$\mathscr{C}(\pi;\mathcal{K}_\pi) = \sum_{e\in\pi} \mathscr{C}(e;\mathcal{K}_e) = \sum_{e\in\pi} \mathscr{C}(\mathbb{E}[d(e,\mathcal{K}_e)]),$$

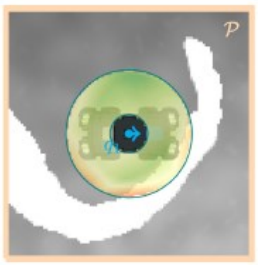
Where the price of the nearby movements is based on the descriptions that have been projected. If we assume, for the sake of argument, that the odds of Due to the fact that the robot's local movements may be completed independently of one another, minimizing the route survival risk R can be a challenging but rewarding task.

$$\mathscr{R}(\pi;\mathcal{K}_\pi) = -\sum_{e\in\pi} \log \mathrm{P}(S(e,\mathcal{K}_e)=1) = -\sum_{e\in\pi} \log y_S(e,\mathcal{K}_e). \quad (1)$$

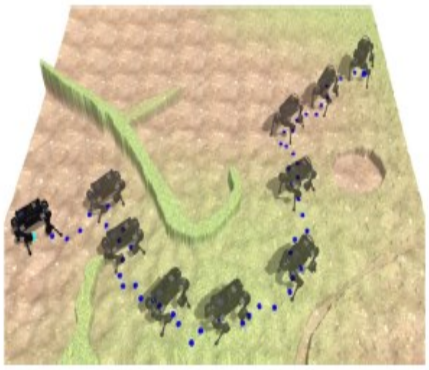
Using a sampling-based planner that repeatedly constructs a network of states linked by edges representing possible local states; we are able to solve the global planning issue. Movements (like local motions that have a good chance of succeeding). Our system utilizes permutations of the Rapid Exploring (Dense) Tree (RDT) [19] method for answering single-use route planning questions. Typically, these algorithms will sample points from the configuration space and iteratively build a treat C that spans the space (with a bias towards t) and



(a) One local motion sample: the target point q_2 is uniformly drawn in an annulus (cyan) from 15 cm to 50 cm around q_1.



(b) Evaluation of the trained model: colors indicate predicted success probability y_S (red: low, green: high) for motions to a target point in the annulus: trying to step over the (white) wall is correctly assigned a low chance of success (orange).



(c) In a challenging terrain, ANYmal successfully executes the planned path of lowest risk (blue dots) between start (left) and target poses (right), computed using the trained model.

FIGURE 3: Animal pipeline (a) by selecting q1 evenly on a map and q2 in an annulus surrounding it, we gather around 90 K random trajectories. In order to utilize RRT* to calculate plans (c), we first use these trajectories to train a model (b). Knowledge of the immediate area around the robot, denoted by the coordinates (q1,q2), is stored in a 2m 2m grid patch P. Height maps are shown in grayscale, while the orange line represents the terrain's elevation. All illustrations are drawn to scale and based on actual scientific research.

As shown in Figure 2(c) and Algorithm 1, the predictions play a crucial role in the planning process, and are used to connect the nodes to the tree via local motions. We cannot evaluate infinitesimally short local motions (otherwise their descriptors would be meaningless), which is the main difference with the original RDT algorithms like RRT, and thus the asymptotic properties are not preserved. RRT* [10] and SST [20] (Stable Sparse RRT) are two ideal variations that employ a cost C (here estimated from

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 18, Issue 1, 2024

the expected descriptors) to create a tree of optimal pathways by connecting them with edges. To the tree may necessitate branch competition or local reorganization (RRT*) (SST).

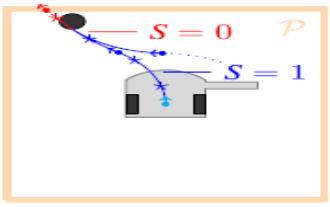
IV. CONDITIONS FOR THE TESTS

Here we provide two examples of how the model described in Section III may be put to use, one for the simulated Animal robot shown in Fig. 1, and the other for a real-world version of the Mighty Thyme robot. The two robots navigate a 2D landscape represented as height maps (i.e., grayscale pictures with altitude values associated with each pixel), while we disregard non-geometric properties (like the surface material). The robot relies on local height maps, which are square patches of height map data centered on the robot's position and aligned with its orientation, to guide its moves in the immediate vicinity. Using the Open Motion Planning Library (OMPL) [21], we deploy a sampling-based planner.

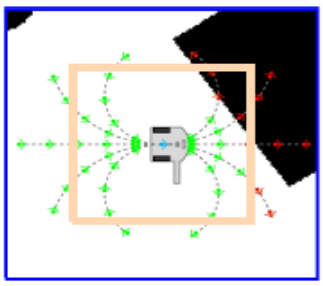
A. Simulated Animal \s1) Robot and Environment: Animal [22] is a state-of-the art\quadruped robot developed at ETHZ for autonomous operation in challenging environments, with the ability to walk on \sough terrain. Gazebo (see Fig. 1(a)) is used to model Animal and its 80 cm 60 cm 70 cm footprint so that we can assess the robot's ability to navigate around various obstacles (such as steps, holes, slopes, and bumps) and design routes that avoid them. For the sake of clarity, we restrict our analysis to orientation-preserving local motions, so that the horizontal position of Animal's centre defines the configuration space C = R2. Local knowledge is represented as a 100 PX 100 PX (2m 2m) patch P of such height map, with a resolution of 2 cm per pixel (see Fig. 3(a)).

To compensate for the unpredictability of the environment, the simulated Animal uses a simple, 1 locomotion planner and a complex closed-loop feedback controller [23]. The only piece of information that is fed into the local planner is the relative target pose, which is represented by the equation q2 = (x, y,). For any sampled trajectory γ, \sd(γ) ∈ {0, 1} × R+ is composed of success S ("has the robot \arrived near enough to q2?") and duration Three, we gather a dataset of about 90 K samples (76% with S = 1), where each sample is a tuple (P, x, y, S, T), by randomly spawning the robot on stable poses. It takes the robot on average 8 steps (about 10 seconds) to complete a local motion (Fig. 3(a)), and it can achieve this by randomly sampling a relative target point at a distance between 15 cm and 50 cm along a random direction. We collect information from 12 unique environments (1,200 square meters in total) that feature difficult obstacles like slopes, bumps, holes, rails, and steps. Height maps of the landscape are generated procedurally using a superposition of simplex noise at varying scales, as we did previously in our work [14].

A deep neural network is trained to predict the success and duration of local motions, as shown in Fig. 4. An estimator consists of two phases: Convolutional layers make up the first type. That function on the height map patch (which is 2D and structured like an image); the second stage processes the resulting features and takes the relative target point as an input (x, y). The model's output is a dense layer with softmax activation, which calculates a success probability yes and an estimated duration yet. Categorical cross-entropy (success) plus mean squared error gives the loss function (duration). All 12 of the dataset's landscapes are unique: the training set has 70K samples from 7 landscapes, the validation set has 10K samples from 2 landscapes, and the evaluation set has 8.7K samples from 3 landscapes. Algorithm 1 is derived from RRT*, and so is Algorithm 5's planner, which uses the trained local motion model to compute global paths.



(a) When the robot collides with an obstacle, it inverts the linear speed (here moving backward) before having reached the target position (red); from the trajectory (blue line), we extract and label many 1 s segments (delimited by star markers).



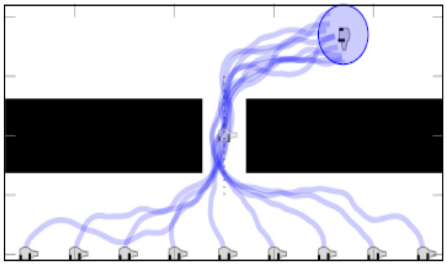
(b) We use the trained model to predict a sequence of five 1 s local motions with constant linear and angular speeds: target poses are colored by success probability y_S (red: low, green high): collisions are correctly identified (red).

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

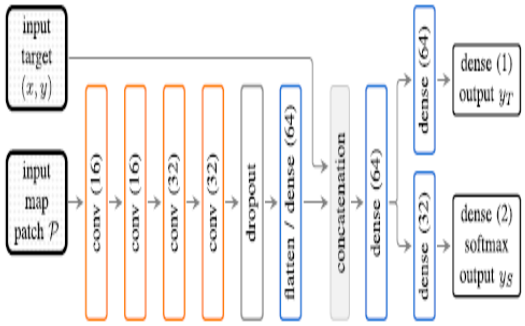


(c) Planning a path through a narrow opening of 44 cm from various starting poses (robots at the bottom) to the same target pose (blue circle) shows that the trained model makes the robot pass on the left side of the corridor to avoid touching with its arm poking out to the right.

Powerful Thyme pipeline, as shown in Fig. We use SST to plan safe trajectories (= 0.98) that minimize duration by (a) randomly changing the robot's angular speed every 2 s in order to collect 1 s training samples labeled according to the occurrence (or not, S 0, 1) of a collision, and (b) using this model to predict the probability of a collision in the next 1 s for given angular and linear speeds. Extensive familiarity with the area the grid patch P (orange stroke) measures 50 cm on all sides and centers on the robot (q1, q2). Anything that involves drawing is to scale and rely on empirical evidence. Please RRT in III.C1. In particular, the planner repeatedly picks a new random starting point q and makes moves to try to make the tree bigger. From q's nearest neighbor along a 45-centimeter-long segment in the direction of q. Candidates are kept if q as long as the distance is greater than 15 cm; otherwise, they are eliminated. Each edge is given a risk of log yes based on the estimator's success score yS, and edges with scores below a threshold of = 0.5 are deemed impossible to construct. Since RRT* is an anytime algorithm that converges towards the optimal solution, we let the tree grow within a given computational time budget in order to find the path with the minimum survival risk, as defined by Eq. (1).

Thyme, the Powerful The Robot and Its Surroundings, Part 1 An improvement upon the original small educational robot Thyme [25], the Mighty Thyme [24] is designed for scientific study. The robot's position and orientation make up its configuration space, denoted by the notation C = SE (2), and are transmitted to its two differentially driven wheels. Figure 1(b) shows a small mobile robot navigating a room with a flat floor and a number of obstacles on it; to make the planning problem more interesting; we have given the robot an extra rigid arm on its right side.

Dimensions for the whole Mighty Thyme are 11 by 11 by 20 centimeters, while the arm is 10 by 2 by 0.5 centimeters. The purpose of this study is to put the model presented in Chapter III to use in a different setting: the pursuit of knowledge. To foresee whether or not the robot (whose geometry is presumed to be unknown) will clash with obstacles during a brief motion, and to utilize this knowledge to design courses that avoid collisions. Local knowledge is represented as robot-centered 80 PX 80 PX (50 cm 50 cm) patches, while environs are represented as height maps with a resolution of 0.625 cm per pixel. Second, nearby motion: In the same way as Animal can conduct arbitrary local movements in free space with the use of a pretty complicated planner/controller combo, the Mighty Thyme can do the same. Instead, we give a simplified model that exemplifies the universality of our method by restricting local movements to constant linear and angular velocities of v = 8 cms1, 8 cms1 respectively.



The Animal Neural Network design, as described in Section IV-A is shown in Figure 5.

The model calculates a probability of attaining the target posture and an estimate of the robot's pose from the given robot-centered height map patch P and the relative target goal (x, y). Time necessary. The architecture of Mighty Thyme is fairly similar to that of the previous game (see Section IV-B). The Third Step Is to Gather the Datasets: We collect information using a controller that chooses an angular velocity at random every two seconds. After every collision, the robot remembers the details and reverses its direction of travel (going backward if it was going ahead and vice versa). There is a motion capture system that keeps track of where the robot and the obstacles are at all times. We operate the robot for 120 minutes over six maps, recording around 460 collisions. After the data is acquired, it is processed to extract 1 second trajectories with constant controls, with S = 1 for trajectories that do not include collisions and S = 0 for those that do (see Fig. 4(a)). We amass a collection of 104 K samples in

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

the format (P, v, S). Four) we use the CNN architecture shown in Fig. 5 for our training, with the exception that our output is limited to the success probability (no-collision) of the motion. The dataset is created by us. 52k samples were utilized for training (3 maps), 20k for validating (1 map), and 30k for judging (all from separate sets of maps) (2 maps). Tracking Your Steps: We use a modification of SST that samples control inputs to attain a local target point in order to account for the differential restrictions of the robot's movements. To be more specific, the planner adds an edge to the tree only if it reduces the local cost-to-come, and SST repeatedly chooses a random posture q SE(2) and multiple control inputs, picking the one that would bring the robot closer to q from its nearest neighbor. Feasible edges are assigned a constant cost of 1, which is proportionate to their duration, and edges with a chance of success below = 0.98 are discarded. SST is a nearly optimum solution converge at any time, therefore we let the tree grow for a predetermined period of computing time.

V. EXPERIMENTAL RESULTS

How accurate are our predictions of the results of local motions?

How helpful are the neighborhood models in determining where to go? Do our presumptions from Section III make sense? And in particular are were we correct to take yes as the chance of success for individual actions and then use it to determine the likelihood of success for a complete trajectory? To begin answering these problems, we now offer experimental data for the two configurations described in Section IV.

The predictions made by the learnt model are consistent with the map, as shown by qualitative testing shown in Figs. 3(b) and 4(b): the Animal model accurately estimates that movements that step over the short wall are dangerous, and the Mighty Thymiomodel appropriately predicts collisions. Similarly, Figs. 3(c) and 4(c) illustrate that the planner computes logical courses between manually selected sites on interesting maps: Animal avoids steep stairs and slopes, while Mighty Thyme (with an arm on the right) remains on the left side of a corridor to prevent collisions. Several examples of planned pathways and their actual implementations may be seen in the film included in the appendices. A. Predicting Motion Close to Its Origin 1) Animal: The degree to which the Animal model's predictions match the test's real values is shown in Fig. 6: left. Data set; the associated performance ratings are shown in the grey assessment columns of Table I (AUC = 0.889) for evaluating single local movements. Both of the model's predictions are highly accurate. Table II and Fig. 6: right show the findings of the Mighty Thyme model predicting success probability (AUC = 0.972), which are similar to those found in the previous section. Inadequate calibration (in comparison to the Animal model) of a model that tends to overstate the success probability may be attributable to the low number of collisions experienced by the robot during data collection (one unsuccessful sample for every 25 successful samples).

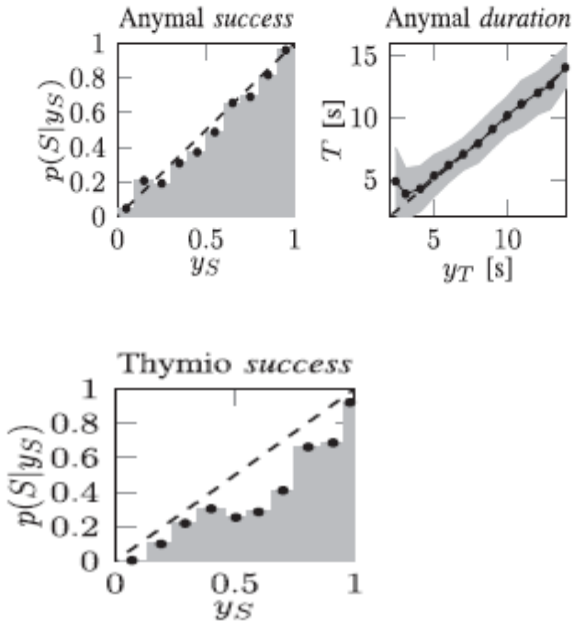


Fig. 6: Animal (8700 samples, left and centre) and Mighty Thyme (30 K samples, right) local motion model assessment (ground truth vs. prediction). The yes values in the success plots are divided up into 0.1-point bins. Throughout the time frame figure, where the dots represent the mean value of T for outputs yT clustered in 1 s width bins and the grey area is bounded by 1 standard deviation. Black dashed lines depict flawless models.

PREDICTION OF SUCCESS USING AN ANIMAL MODEL TABLE I NUMBER OF SAMPLES FOR THE TWO CLASSES AREA UNDER THE ROC (AUC). IN THE COLUMNS ARE THE RESULTS FOR INDIVIDUAL LOCAL MOTIONS e (IN GRAY), PATHS BETWEEN DISCRETE ORIGIN AND DESTINATION POINTS ON CHOSEN MAP LAYOUTS (IN WHITE), AND ALL SUBPATHS sub EXTRACTED FROM SUCH PATHS (BLUE)

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

dataset	# successes			# failures			AUC		
	e	π	$sub\pi$	e	π	$sub\pi$	e	π	$sub\pi$
evaluation	7.30K	-	-	1.40K	-	-	0.889	-	-
rough map	18.4K	562	297K	185	185	49.7K	0.878	0.904	0.912
surf map	22.4K	564	431K	237	237	63.5K	0.962	0.962	0.952

TABLE II
MIGHTY THYMIO MODEL EVALUATION: NUMBER OF SAMPLES FOR THE TWO CLASSES, AREA UNDER THE ROC (AUC), AND *SUCCESS* PROBABILITY FOR SCORES ABOVE $\tau = 0.98$, WHICH IS THE THRESHOLD WE USE TO COMPUTE THE PLANS ACCORDING TO SECTION IV-B

successes	failures	AUC	$P(S = 1 \mid y_S > 0.98)$
28913	1087	0.972	0.997

With a threshold of = 0.98, however, the model produces 99.7% accuracy and 96.1% recall, both of which are sufficient for use in our map design.

Planned Movement of ANY Animal We calculate and then execute several pathways between destination and source locations for the simulated Animal, and publish our findings here to provide light on our main contribution, which is to use the learned local motion model to design a path. To be more specific, we take a random sample of 1000 pairs of source and target states across two maps (rough and surf) of size 10m 10m, as shown at the top of Fig. 8; we then eliminate any pairs that are located outside the green areas (where we anticipate a successful spawning of the simulated robot) and any pairs with a distance shorter than 3 m. Neither map was used to train or assess the model, although both were created by hand to seem like actual outside terrain (rough) and be easily readable (surf, with a smooth, gentle slope).

On a single, state-of-the-art desktop CPU, a single model assessment takes roughly 1 millisecond. Our 120-second RRT* planning window is fine for a granularity of map coverage we've established. Trees with = 0.45m have roughly 10,000 edges and need 100,000 model evaluations.

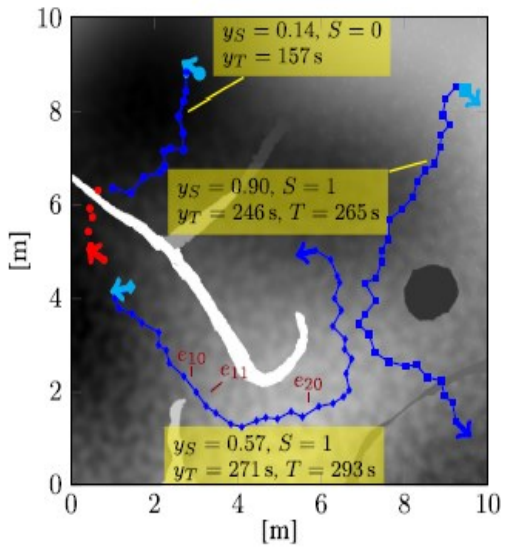


Figure 7: Three examples of routes from origin to destination shown on a crude map, with one unsuccessful route shown in partially red and two successful routes shown in blue (all blue). For each run, we keep track of both the actual results and the predicted outcomes. (Yellow). Multiple runs may yield different results due to random interactions with the landscape. For instance, in 30 attempts along the bottom path, 14 attempts failed while 16 succeeded, yielding a P(S = 1) of 0.53, which is remarkably close to the predicted value yes = 0.57. The failures occurred at a rate of 1 in 3 of the 34 segments (e10 : 7, e11 : 3, e20 : 4). For every desired trajectory, we first create the robot at the starting point and then feed the controller its series of intermediate goal posts, as shown in Fig. 7.

In around 80% of situations, the robot successfully reaches the goal, while in the other 20%; we see a failure to finish one segment of the route due to timeout or other causes. From every given route, we extract all subpaths that begin before a failure (if any): each subpath contains well-defined ground truth labels and predictions for success and duration, greatly expanding the number of (path) samples from 1 K to roughly 400 K. Success rates for predicting the completion of single segments, source-target pathways, and all of their sub paths are compared in Table I. The area under the curve (AUC) is consistent between datasets for both maps. 3 Although it is more challenging to make accurate predictions on the surf map, the rough map's performance is on par with the previously discussed evaluation dataset of single segments. In Fig. 8, we see an example of the prediction's accuracy when applied to sub paths. The model is better calibrated on the rough map than on surf, which is smoother but has steeper slopes, and

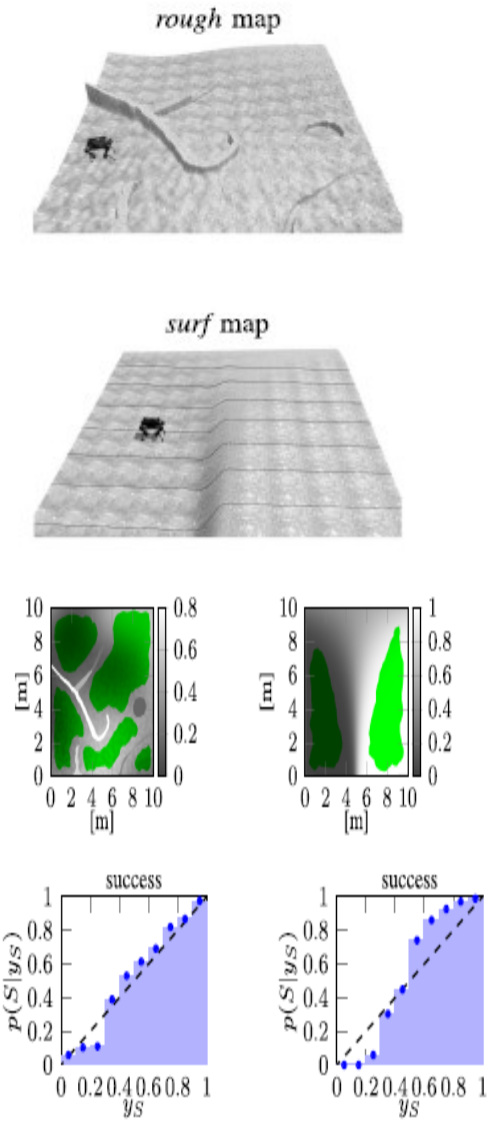
INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

the success score is generally well calibrated, though it tends to overestimate the success chance of risky paths (i.e., those with low score). While we record the time until a single pose is reached during training—which may occur before the robot has fully stopped—during path execution, the controller waits until the robot is still before beginning to move towards the next pose, leading us to believe that the actual duration of sub paths is consistently underestimated. Still, at only about 5%, the discrepancy is hardly noticeable.



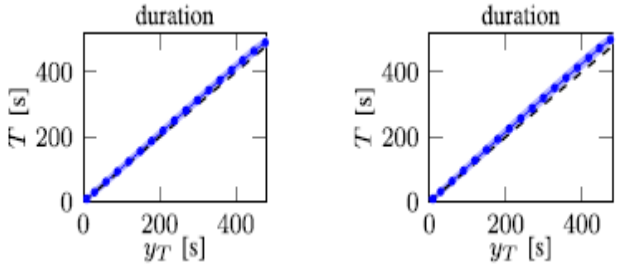


Figure 8: An analysis of two maps' assessment of an Animal model on around 400 K sub paths (rough left and surf right). In the top row, you can see how the landscape might look from above in 3D if a robot were the right size for it. Its height map data is shown in the second row (in meters) and the spawning grounds are shown by the green. The bottom plots show the range of error for the binned prediction outputs, with the area corresponding to 1 standard deviation shown by the dots at the mean.

VI. CONCLUSIONS

To solve the problem of robotic route planning, we used a unique, data-driven strategy: first, we learn to predict the result of short trajectories from local information; subsequently, we utilize this model to plan longer trajectories. Filter out unreliable edges as you construct a thick random tree to link your source and destination states. We used the model to compute control trajectories for a real, non-homonymic, Mighty Thyme robot moving between obstacles with a simple controller and to compute geometrical paths on rough terrain for a simulated legged Animal robot with a sophisticated controller whose performance would be difficult to model by hand. It is worth noting that in the second scenario, it would be simple to analytically model the robot's collision risk with known obstacles; however, it is intriguing that our data-driven approach yields acceptable results, using a small training dataset and requiring no explicit knowledge of the robot geometry.

Figure 8: An analysis of two maps' assessment of an Animal model on around 400 K sub paths (rough left and surf right). In the top row, you can see how the landscape might look from above in 3D if a robot were the right size for it. Its height map data is shown in the second row (in meters) and the spawning grounds are shown by the green. The bottom plots show the range of error for the binned prediction outputs, with the area corresponding to 1 standard deviation shown by the dots at the mean.

ISSN 2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

REFERENCES

[1] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," IEEE Access, vol. 2, pp. 56–77, 2014.

[2] C. Doers, A. Gupta, and A. A. Eros, "Unsupervised visual representation learning by context prediction," in Proc. IEEE Int. Conf. Compute. Vision, 2015, pp. 1422–1430.

[3] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," Nature, vol. 521, pp. 452–459, 2015.

[4] D. González, J. Perez, V. Milanese, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[5] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in Proc. IEEE Int. Conf. Robot. Auto., 2017, pp. 1527–1533.

[6] M.Ollis, W.H.Huang, andM.Happold, "A bayesian approach to imitation learning for robot navigation," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2007, pp. 709–714.

[7] J. Bagnell, J. Chestnutt, D. M. Bradley, and N. D. Ratliff, "Boosting structured prediction for imitation learning," in Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 1153–1160.

[8] S. M. Lavelle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[9] L. E.Kavraki, P. Švestka, J.-C. Lacombe, andM.H.Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Trans. Robot. Auto., vol. 12, no. 4, pp. 566–580, Aug. 1996.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion plan," Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894, 2011.

[11] A. Upadhyay and C. Ekenna, "Investigating heterogeneous planning spaces," in Proc. IEEE Int. Conf. Simul., Model. Program. Auton. Robots, 2018, pp. 108–115.

[12] J. Denny, M. Morales, S. Rodriguez, and N. M. Amato, "Adapting rrt growth for heterogeneous environments," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2013, pp. 1772–1778.