Learning High-Dimensional Critical Regions for Efficient Robot Planning

by

Abhyudaya Srinet

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2020 by the
Graduate Supervisory Committee:

Siddharth Srivastava, Chair
Yu Zhang
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2020

ABSTRACT

Robot motion planning requires computing a sequence of waypoints from an initial configuration of the robot to the goal configuration. Solving a motion planning problem optimally is proven to be NP-Complete. Sampling-based motion planners efficiently compute an approximation of the optimal solution. They sample the configuration space uniformly and hence fail to sample regions of the environment that have narrow passages or pinch points. These critical regions are analogous to landmarks from planning literature as the robot is required to pass through them to reach the goal.

This work proposes a deep learning approach that identifies critical regions in the environment and learns a sampling distribution to effectively sample them in high dimensional configuration spaces. A classification-based approach is used to learn the distributions. The robot degrees of freedom (DOF) limits are binned and a distribution is generated from sampling motion plan solutions. Conditional information like goal configuration and robot location encoded in the network inputs showcase the network learning to bias the identified critical regions towards the goal configuration. Empirical evaluations are performed against the state of the art sampling-based motion planners on a variety of tasks requiring the robot to pass through critical regions. An empirical analysis of robotic systems with three to eight degrees of freedom indicates that this approach effectively improves planning performance.

To my family and love, thank you for everything

ACKNOWLEDGMENTS

I would like to thank Dr. Siddharth Srivastava for his support and guidance and giving me the opportunity to work with him. I would like to thank my committee members Dr. Yu Zhang and Dr. Yezhou Yang for their time and assistance. I would like to thank the members of Autonomous Agents and Intelligence Robots Lab for their help and support. I would like to thank Naman Shah and Vatsal Sodha for their help and ideas with the approaches presented in this thesis.

I would like to thank my mother, father and brother for their immense support. I would also like to thank Simran Sundriyal for her constant encouragement and support. Lastly, I want to thank everyone who has helped me get here.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Sampling Based Motion Planning

Motion planning is the process of finding a sequence of collision-free configurations that lead the robot from an initial configuration to a goal configuration. Sampling based planners such as Rapidly exploring Random Trees (RRT) (LaValle and Kuffner Jr, 2001) solve the motion planning problem by sampling the configuration space and then connecting samples by reachability. They are probabilistically complete as the number of samples approaches infinity. However, environments have regions which are critical such that the robot must pass through these critical region in order to reach the goal. In cases where these regions comprise of narrow passageways, the uniform sampler requires a much longer time to establish connectivity. The probability of such regions getting sampled reduces or they may be sampled at a much later stage thereby increasing planning time. These critical regions are formally defined in Chapter 2.

The efficiency of sampling based motion planners can be improved by augmenting the sampling distribution to sample from such regions. Learn and Link Planners (LLP) (Molina *et al.*, 2019) expand subgraphs rooted at samples belonging to the identified critical regions in the configuration space. This enables them to build a biased roadmap that spreads across the environment. An example of such a region would be a long narrow passage connecting two rooms. Figure 1.1 depicts such an environment with a long narrow passage connecting 2 rooms. The depicted samples are generated by RRT and LLP when the planner is given a time limit of 10s. It

can be observed that LLP achieves better connectivity across the environment as compared to RRT which seems to have trouble sampling the passageway. Learn and Link provides 2 modes of operation:

1. Learn and Link Planner (LLP): A single query mode planner useful for planning from a start to goal configuration. The planner builds a biased roadmap using subgraphs rooted at the start and goal configurations as well the critical regions identified by the network.

2. Learn and Link Roadmap (LL-RM): A multi query mode planner that builds a roadmap of the environment using the generated samples and random samples from the environment. The random samples are sampled uniformly from the configuration space. This helps the environment achieve better connectivity across the environment for general querying.

Once the graph is constructed and the start and goal configurations are reachable, the planner uses Dijkstra's algorithm to find a path.

Molina *et al.* (2019) use a network that generates a distribution of the critical regions for base navigation i.e. 2 Degrees of freedom (DOFs) and append a collision free configuration to the sampled configuration for remaining DOFs. We extend this work further by building a network that generates a sampling distribution for higher dimensional configuration spaces. Providing encoded information regarding goal configuration and robot location showcase that the network learns to bias the identified critical regions towards the goal configuration and focuses the sampling distribution to sample configurations belonging to these regions. We use encoder-decoder architectures for building the neural network. The methodology is tested on 3 domains comprising of 11 unique environments: 7 SE(2) environments and 4 3D environments. We leverage the Learn and Link planners as described in (Molina

et al., 2019) to work with the samples generated by our network.

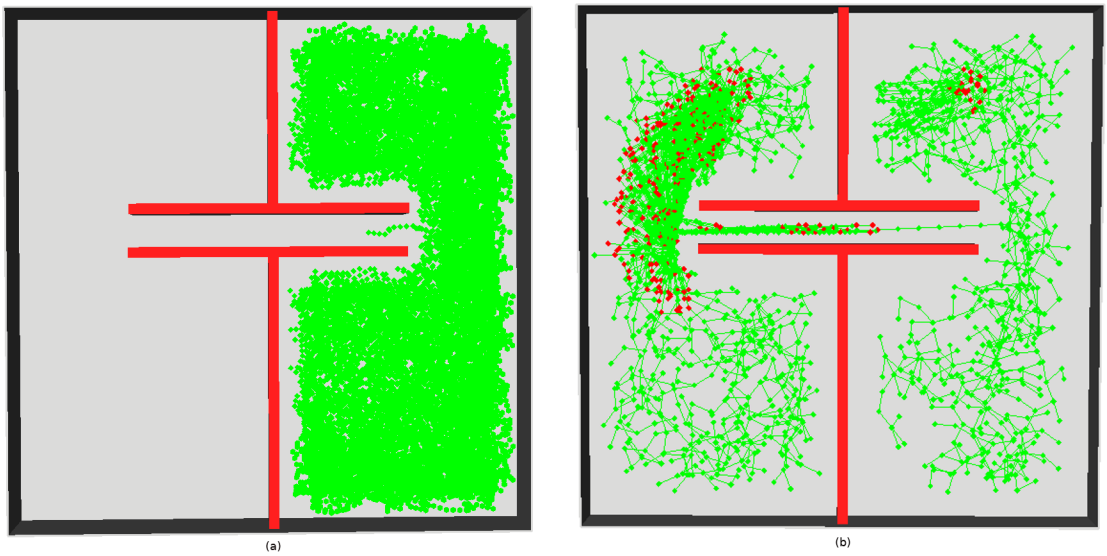

Figure 1.1: (a) Sampled states by RRT planner when given a planning time of 10s. Each green point is a randomly sampled state (b) Sampled states by the LLP planner when given a planning time of 10s. Each green point is a randomly sampled state while red points are states predicted by our network

1.2 Related Works

Our work aims to improve sampling based motion planners by generating a sampling distribution that biases the samples towards identified critical regions. Previous works on improving sampling based motion planners include using better heuristics for sampling. Urmson and Simmons (2003) improves the exploration of RRTs by biasing their growth based on costs discovered from exploration. BIT* (Gammell *et al.*, 2015) uses heuristics to efficiently search a series of increasingly dense implicit random geometric graphs while reusing previous information. In adaptive sampling techniques, Burns and Brock (2005) develop a utility guided strategy for sampling that approximates the model of the configuration space and its connectivity. The sampler then selects samples based on utility and the planning task. Learning based methods have also been used to improve motion planning. Zhang *et al.* (2018) use a policy-search

based method to learn implicit sampling distributions for different environments from past solutions and generalize it to novel environments. Havoutis and Ramamoorthy (2009) learn a non-linear manifold subspace of the configuration space (C-space) to improve distance measures and sample the C-space efficiently. Pan *et al.* (2013) use instance based methods by storing previous colliding and non-colliding samples. This past information is then leveraged to improve planning. Our network aims to encode this information into the sampling distribution itself. Zucker *et al.* (2008) use reinforcement learning to learn the sampling distribution. Lehner and Albu-Schäffer (2017) fits Gaussian Mixture Models to previous solutions to learn a better sampling distribution. Kumar *et al.* (2019) provide a framework for training a CVAE with biased samples towards diverse regions to build a good reaching roadmap. Ichter *et al.* (2018) learn sampling distributions through a Conditional Variational AutoEncoder conditioned on environment data encoded as an occupancy grid. However they are difficult to generalize due to the complex input requirements and also require high inference computation times. Ichter *et al.* (2019) use graph theoretic techniques to learn and predict the criticality of a sample from local environment features. They learn the criticality of a sample by learning from shortest path problems. Their approach requires predicting the criticality of a generated sample and then drawing samples proportional to their criticality. Our approach can work with simple raster scans of the environment and encodes conditional information along with the input while the predicted distribution can be directly used for sampling.

Chapter 2

METHODOLOGY

Definition 1. *Given a robot R, an environment E, and a class of motion planning (MP) problems M, the measure of criticality of a Lebesgue-measurable open set $r \subseteq \mathbb{R}^n$, $\mu(r)$, is defined as $\lim_{s_n \to +_r} \frac{f(r)}{v(s_n)}$, where $f(s_n)$ is the fraction of observed motion plans solving tasks from M that pass through s_n, $v(s_n)$ is the measure of s_n under a reference (usually uniform) density, and $\to^+$ denotes the limit from above along any sequence s_n of sets containing r ($r \subseteq s_n$ for all n). (Molina* et al.*, 2019)*

Intuitively, these regions can be thought of as regions that are crucial to solve motion planning problems in M however have low probability of being sampled uniformly. We aim to learn a distribution which samples these critical regions efficiently by training our network on a train set D_{train}. D_{train} consists of N_{train} samples, where each sample N_i is generated by solving p MP problems $\tau_1 \ldots \tau_p$. We use OpenRAVE (Diankov and Kuffner, 2008) to simulate all p motion planning problems and solve them using OpenRAVE's implementation of the OMPL RRTConnect planner with a time limit of 180s.

The input to the network is a raster scan of the environment along with encoded conditional information about the goal configuration and the robot location. The output of the network is the distribution that is further used for sampling configurations from the critical regions. We discretize the configuration space of the robot by binning the range of the respective DOFs. We then sample the p motion plan solutions and for each bin assign the count of how many samples in the solution belong to a given bin. We then normalize these bin values across each DOF to represent a

distribution.

We demonstrate this approach on 3 domains:

1. 3 DOF Robot: A rectangular shaped robot with 3 DOFs in SE(2) environments
2. 4 DOF Robot: A L-shaped robot with 4 DOFs in SE(2) environments
3. 8 DOF Robot: A stationary fetch robot with 8 DOFs in 3D environments

Each domain involves 2 stages:

1. Data Generation
2. Network Training

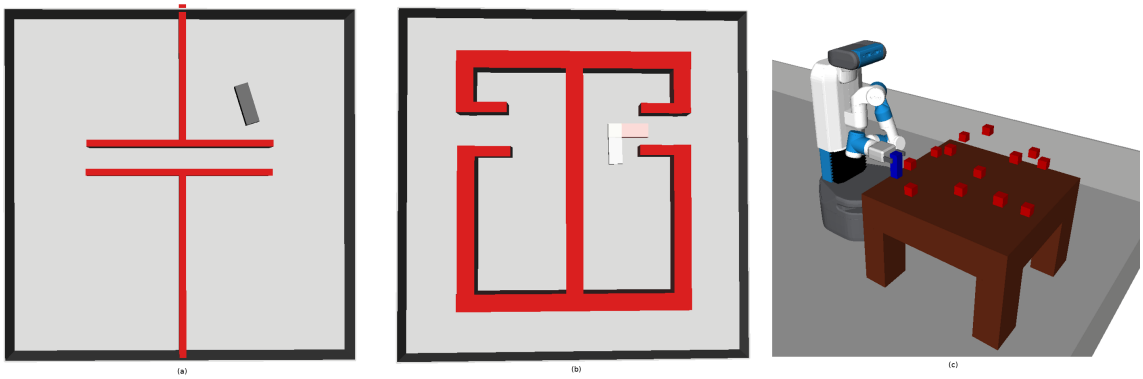

Figure 2.1: (a) Rectangular 3 DOF robot (b) L-shaped 4 DOF robot (c) Fetch robot with 8 DOFs

2.1 3 DOF Robot

We aim to identify the critical regions of the environment and build a distribution that can be used to generate samples in these regions from the configuration space of the robot. We perform multi-label classification to build the sampling distributions.

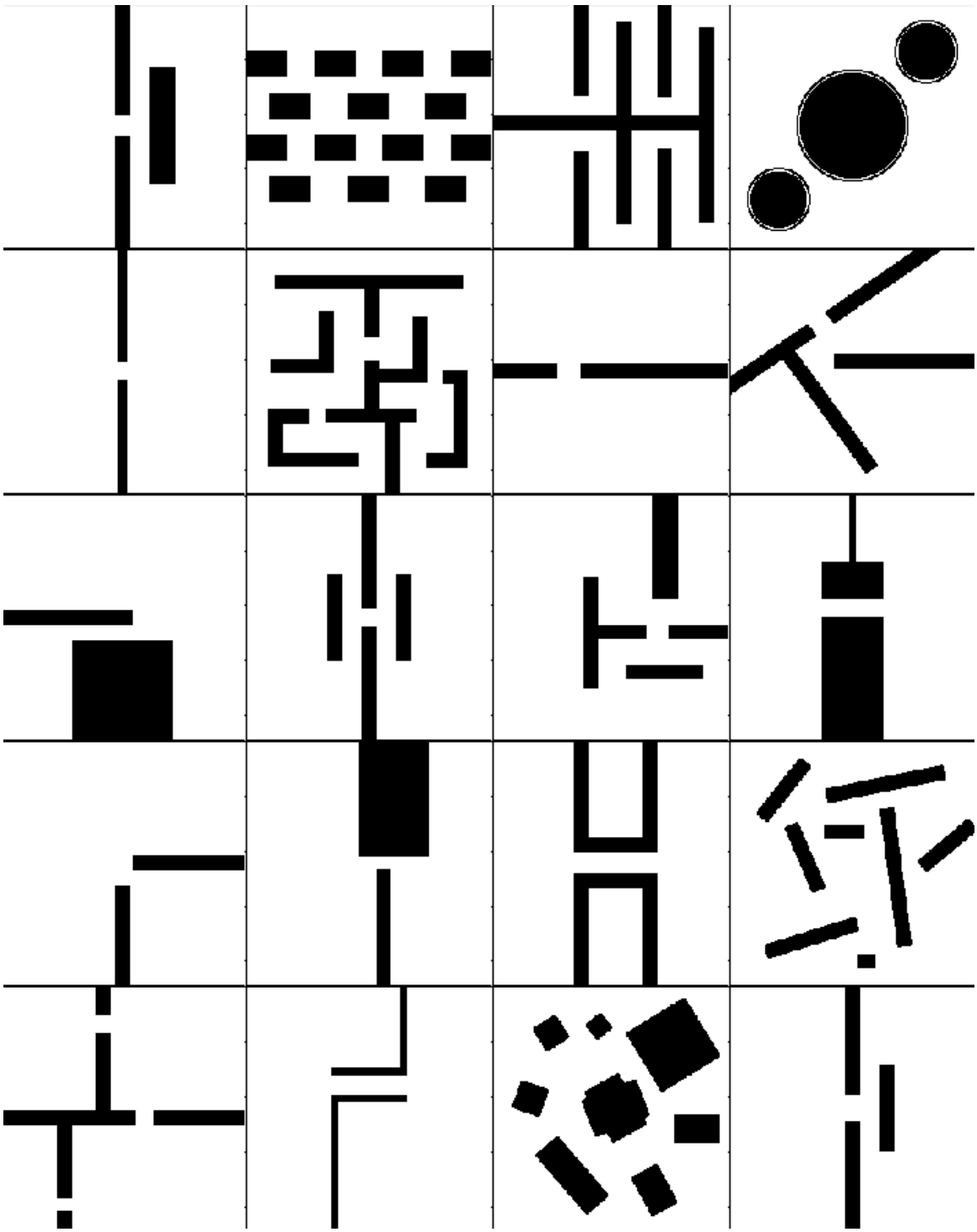

Figure 2.2: Training environments used in the 3 DOF and 4 DOF robot domains

2.1.1 Data Generation

We hand-create the SE(2) environments used for training (Figure 2.2). For each environment, we generate $N_{train} = 100$ data samples. For each N_i, we solve $P = 50$ motion planning problems from a random start configuration to a random goal configuration.

The input is generated by performing a 2D raster scan of the environment at $z = 0$ to generate a 224×224 grayscale image shown in Figure 2.3. This is done by scanning a pixel-sized body across the environment and assigning the image's pixel a value of 0 if an obstacle collides with the pixel body. This generates a black and white image with pixel values equal to 0(black) for obstacles and 1(white) for free space.

The labels are distributions constructed as 224×224 images with 5 channels. The first 2 DOFs represent the x and y values of the robot in the SE(2) environment. We divide these DOF ranges into bins equal to the number of pixels. Since a 2D image's pixel indices are in a row and column format similar to the xy axes in an SE(2) environment, the first 2 DOFs are represented jointly in the first channel. The third DOF represents the orientation of the robot. The range of the orientation is [-π, π] which is divided into 4 equal sized bins such that the bins correspond to an orientation of North, East, West, and South respectively. Each of the 4 remaining channels correspond to one of the bins. Each pixel in a channel takes the value equal to the fraction of the P motion plans that pass through it.

We generate motion traces of all P motion plans as seen in Figure 2.3 (b). Each colored point in the image represents a sample from the p MP solutions for the sample. The different colors imply different orientations in the sampled configurations. It can be observed that the vertical passageway requires samples where the robot needs to be oriented vertically and the same can be observed for the horizontal passageway.

The first row in Figure 2.4 depicts the raw channels for a data sample. We can observe that channels 2 and 5 have highlighted the passage where the robot must be oriented vertically facing north/south while channels 3 and 4 highlight the passage where the robot must be oriented horizontally facing east/west.

We then perform saliency mapping (Itti and Koch, 2000) on the individual channels to smoothen the salient regions as shown in the second row of Figure 2.4. We then threshold the generated saliency maps and assign a value of 1 for values above the threshold and 0 for values below the threshold. We experiment with various thresholds as discussed in Appendex A and found a threshold of 27% to be optimal. The final image obtained after thresholding can be seen in the third row of Figure 2.4. Finally, we also perform data augmentation by rotating each data sample by 90, 180, 270 degrees and flipping the image vertically and horizontally.

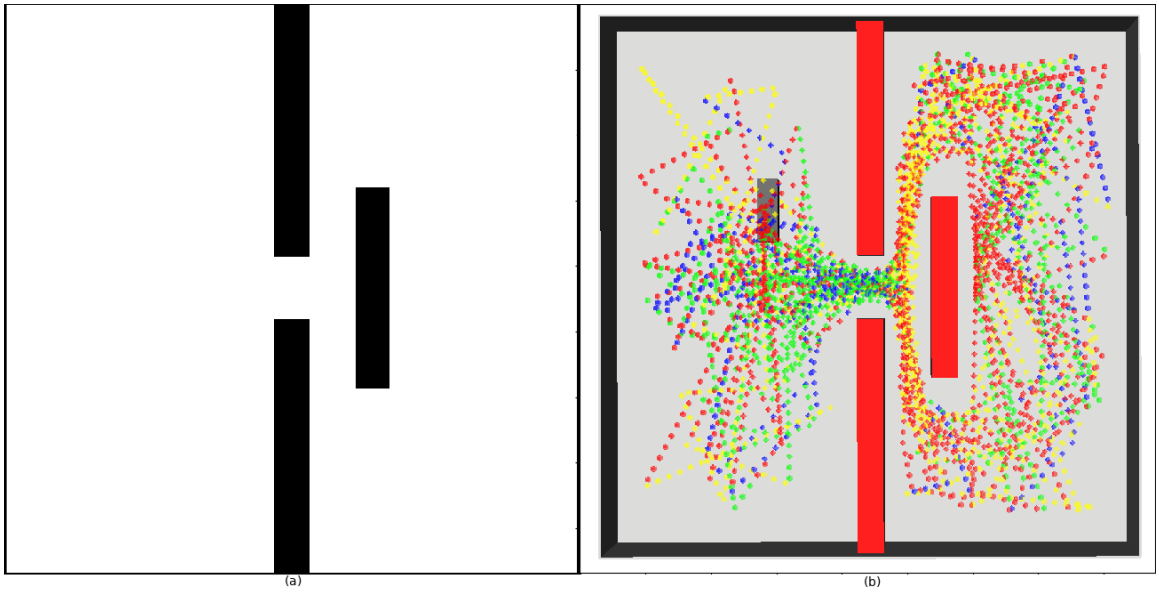

Figure 2.3: (a)A sample input image created after raster scanning the environment (b) Motion traces of 50 motion plans. The different colored points imply the 4 different orientations of the rectangular robot. Red pixels imply oriented facing north, yellow pixels imply oriented south, green pixels imply oriented west, blue pixels imply oriented east. We can observe vertical passages are concentrated with red and yellow pixels while the horizontal passageway has green and blue pixels

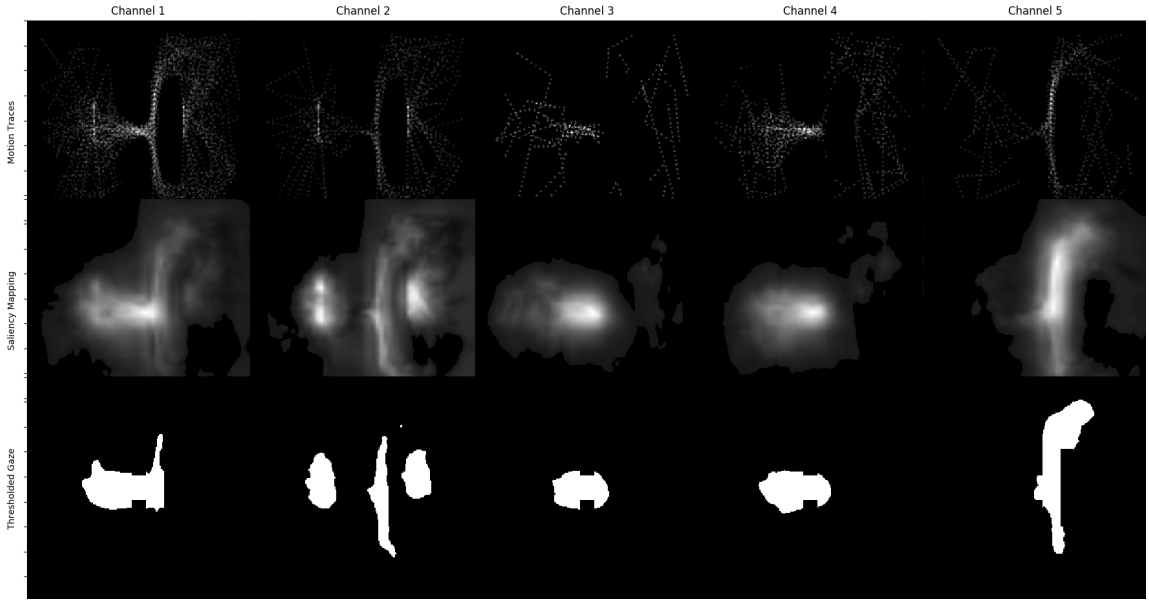

Figure 2.4: The three rows depict the 3 phases of data generation for the environment shown in Figure 2.2. First row depicts the Motion traces plotted for the 4 channels for 50 motion plans. Second row depicts the channels after applying saliency mapping on them. Third row depicts the channels after thresholding the saliency maps by a threshold of 27%.

2.1.2 Network Architecture

Our network is an Autoencoder architecture to perform pixel wise classification on the $224 \times 224 \times 5$ output image. As depicted in Figure 2.5, we use the same architecture used in (Molina *et al.*, 2019). The encoder consists of 3 encoding blocks. The first and second block consists of 2 convolutional layers with 64 and 128 filters respectively followed by a max pool layer. The third encoding block has 3 convolutional layers with 256 filters each followed by a max pooling layer. The decoder consists of corresponding 3 decoding blocks equivalent of the encoder blocks. The first block consists of 3 deconvolution layers with 256 filters followed by an upsampling layer. The second and third decoder blocks have 2 deconvolution layers of 128 and 64 filters respectively followed by an upsampling layer. All layers use a kernel of shape [3,3]. The convolution and deconvolution layers have a stride of 1 while the

maxpool and upsampling layers have stride 2. The output of each convolution and deconvolution layer is batch normalized (Ioffe and Szegedy, 2015). Each layer uses ReLU as it's activation function. The loss function is weighted sigmoid cross entropy with a weight of 2 for the positive class. This is due to class imbalance since the number of pixels identifying critical regions are low as compared to the non-critical regions. We experiment with weights of 2, 5, 10 and found 2 to be optimal.

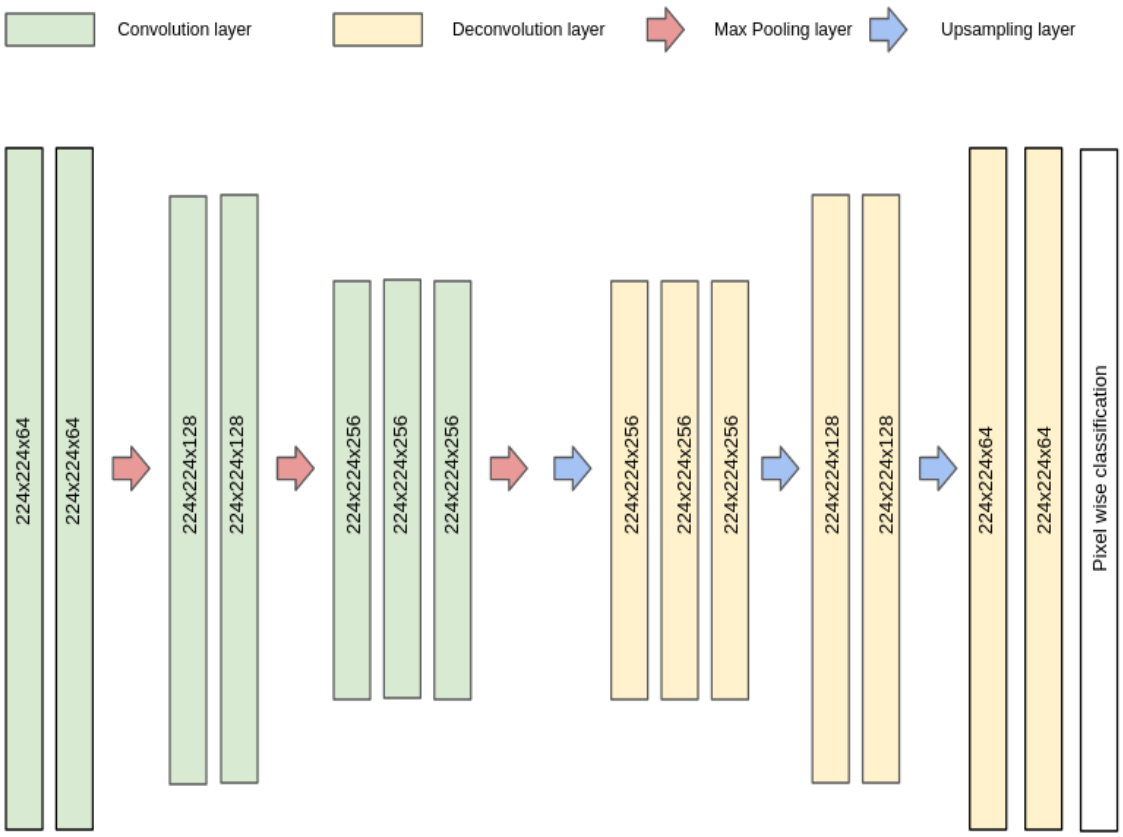

Figure 2.5: Network architecture for the model used in the 3 DOF robot domain

2.1.3 Training

Our train set comprises a total 12000 samples. We use a batch size of 10 with random shuffling applied. We use Adam (Kingma and Ba, 2014) for the optimizer with a learning rate of 0.0001 and train for 10000 iterations. We build upon the data

pipeline implemented using Tensorflow by (Azzini, 2017).

2.2 4 DOF Robot

In this domain, we include the goal configuration as part of our input to make the network learn to plan towards the goal configuration. As seen in Figure 2.6, the robot is an L-shaped robot where the primary rectangular body can rotate between angles of range [-π, π]. The secondary rectangular body or the tail can rotate between angles of range [-$\pi/2$, $\pi/2$] hinged at the end of the primary body. The first 2 DOFs represent the x and y coordinate values in the SE(2) world and the 3rd and 4th DOFs represent the orientation of the primary and secondary body respectively. We perform multi-class classification to build the sampling distributions.

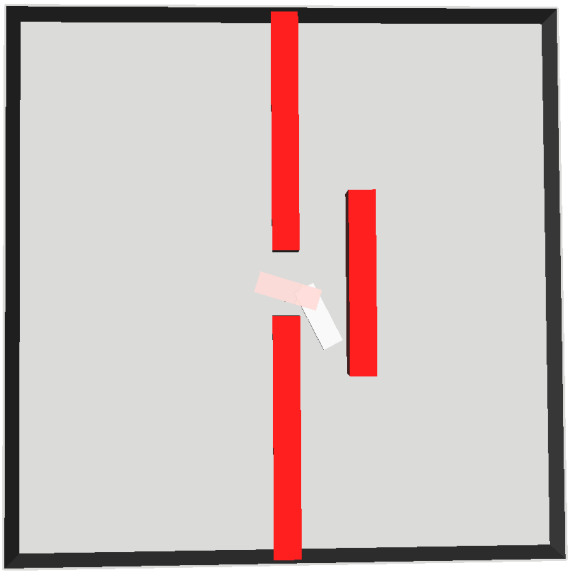

Figure 2.6: The L-shaped robot in a training environment

2.2.1 Data Generation

We use the same environments as shown in Figure 2.2. For each data sample N_i, we solve $P = 50$ motion planning problems from a random start state to a fixed goal

state.

The input is structured as 224×224 images with 7 channels. The first 3 channels are a grayscale image of the input generated by performing a 2D raster scan of the environment as described in the Section 2.1. The next 4 channels encode the goal configuration. The goal configuration here consists of a tuple of the 4 DOF values(x, y, θ_1, θ_2) which are normalized to a range of [0, 1] based on the DOF bounds. We then perform depthwise concatenation as demonstrated in (Choi *et al.*, 2018) to encode conditional information within the input. Each goal DOF configuration value is repeated across every pixel in its respective channel. These channels are concatenated to the input image channels.

The labels are distributions structured as 224×224 images with 3 channels. The first channel is created as described in the Section 2.1.1 and encodes the first 2 DOFs. The remaining 2 channels encode the remaining 2 DOFs of the robot. The second channel describes the orientation (θ_1) of the primary rectangle body of the L shaped bot (3rd DOF). This DOF has a range of [$-\pi$, π] which is divided into 8 bins. Each pixel in this channel takes the value equal to the index number of the bin (ranging 0-7) with the most motion plans passing through it such that the DOF value lies in the bin range. For pixels where no motion plans pass through, we set a value of 8 implying an unknown class. The third channel describes orientation(θ_2) of the secondary rectangle of the L-shaped robot (4th DOF) and has a range of [$-\pi/2$, $\pi/2$]. The DOF range is divided into 9 bins. Similar to the previous DOF, each pixel in this channel takes a value equal to the index of the bin (ranging from 0-8) with the most motion plans passing through it such that the DOF value lies in the bin range. For pixels where no motion plans pass through them, we set a value of 9 implying an unknown class. Each pixel in these channels represents the bin corresponding to the critical region in the C-space for the respective channel's DOF. We then perform

saliency mapping (Itti and Koch, 2000) to smoothen the salient regions. We then threshold the generated saliency regions and assign a value of 1 for values above the threshold and 0 for values below the threshold. The second and third channel are masked using the first channel to have consistent identified critical regions across all channels. Remaining pixels are set to their respective unknown class values. The generated labels can be seen in the ground truth columns in Figure 3.5. Finally, we perform data augmentation by rotating each data sample by 90, 180, 270 degrees and flipping the image vertically.

2.2.2 Network Architecture

Our network is inspired from the U-Net architecture (Ronneberger *et al.*, 2015) depicted in Figure 2.7. The encoder consists of 5 encoding blocks with 64, 128, 256, 512, and 1024 filters respectively. Each encoding block consists of 2 convolutional layers, the first layer has a stride of 1 while the second layer has a stride of 2. Springenberg *et al.* (2014) showcases the advantage of using convolutional layers in place of pooling layers for better learning albeit at the cost of more training parameters. The decoder consists of corresponding 5 decoding blocks. We concatenate the output of the corresponding encoder block to the input of each decoding block except the first one. All layers use a kernel of shape [3,3]. The output of each convolution and deconvolution layer is batch normalized. We use ReLU as the activation function for each layer. The loss function is a sum of 3 losses based on the predicted pixels for the different channels. For the first channel, we calculate sigmoid cross entropy since it performs binary classification. For the remaining 2 channels, we calculate the softmax cross entropy loss since it performs multi-class classification.

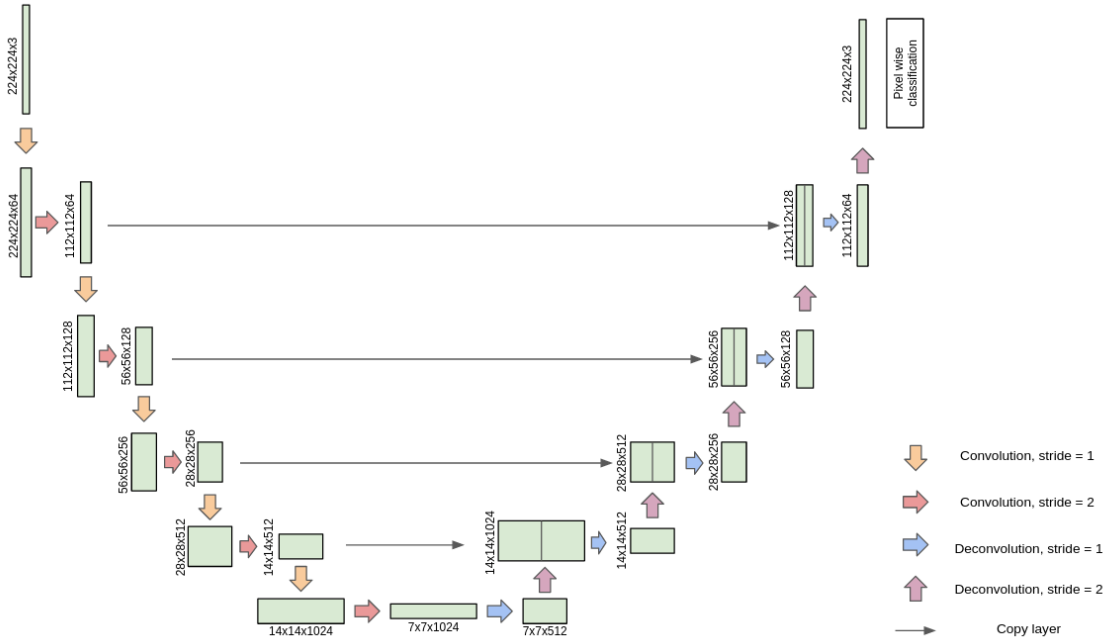

Figure 2.7: The network architecture for the model used in the 4 DOF robot domain

2.2.3 Training

Our training set comprises a total of 10000 samples. We use a batch size of 10 with random shuffling. We use Adam (Kingma and Ba, 2014) for the optimizer with a learning rate of 0.0001 and train for 10000 iterations.

2.3 8 DOF Robot

In this domain, we include the robot's location and the goal configuration as part of our input so our network is conditioned to plan for a goal given the robot's current location in the environment. Since we plan for the robot's arm in a 3D environment, the location of the robot is also important to learn the sampling distribution based on the obstacles around the robot at any given location. We use the fetch robot (Wise *et al.*, 2016) as our robot in this domain. The robot's arm and torso makeup an 8 DOF configuration space.

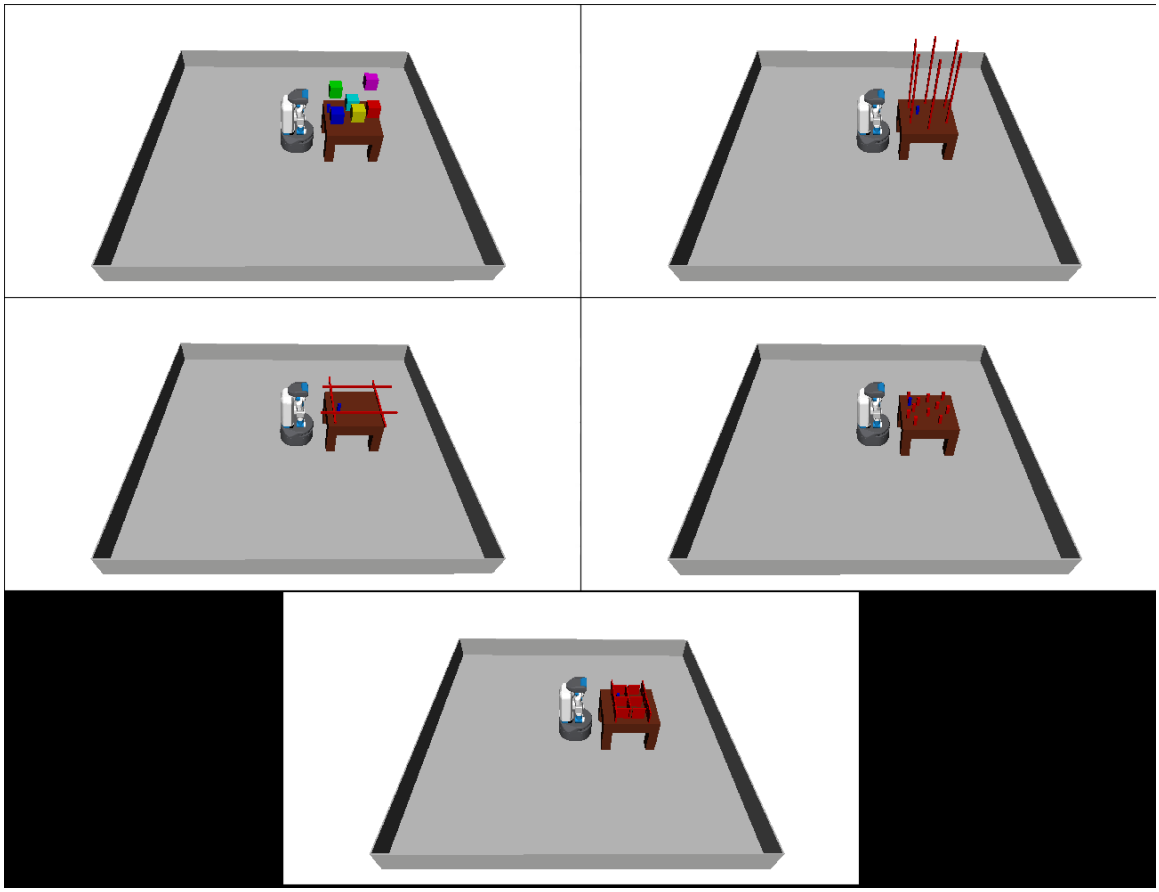

Figure 2.8: The training environments for the 8 DOF robot domain

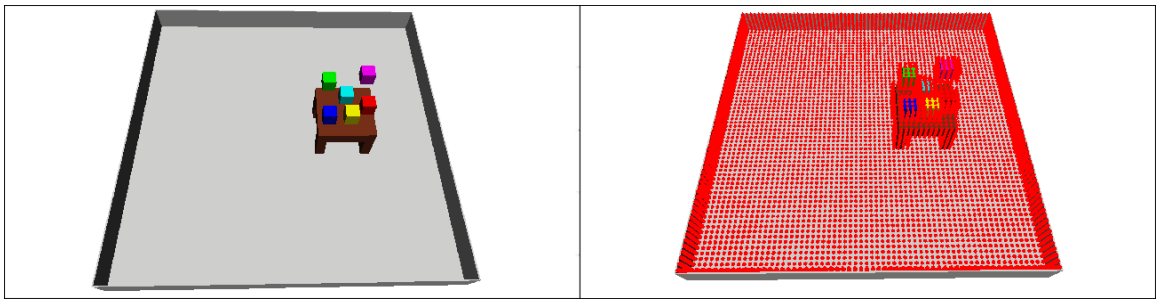

Figure 2.9: (Left) The training environment (Right) The voxelized version of the training environment shown on theleft. Each red pixel represents an obstacle

2.3.1 Data Generation

We generate 5 environments consisting of a table cluttered with immovable obstructions as shown in Figure 2.8. For each data sample, we solve for $P = 50$ MP problems. The goal for a MP problem in this domain is for the robot to be able

to grasp the blue rectangle object placed on a table in front of it where the table is cluttered with immovable obstructions as seen in Figure 2.8. We randomly place the object to be picked up on the table within reachable distance of the robot arm and solve from a random start configuration to a randomly chosen pickup pose out of 8 possible pickup poses.

The input is structured as a $64 \times 64 \times 64$ 3D tensor with 17 channels ($64 \times 64 \times 64 \times 17$). The first 3 channels represent the 3D equivalent of a grayscale image. The 3D environment is voxelized by scanning a voxel sized obstacle across the environment. If a collision occurs, the voxel takes a value of zero, else a value of one for free space. Figure 2.9 depicts a 3D environment and it's voxelized version. We can observe that the voxelization captures the floor, walls, table and obstructions in the environment. The next 8 channels represent the goal configuration values of the robot's DOF. As described in Section 2.2, we perform depthwise concatenation (Choi *et al.*, 2018) to encode this information in the input tensor where each value is repeated across each voxel in the respective channel. The last 6 channels represent the robot's location as its x, y, z coordinates and it's 3 axis angles. The encoded goal configuration and robot location values are normalized to [0,1] based on the DOF and environment bounds.

The labels represent a distribution of the robot's DOF. Each DOF is divided into 10 equal sized bins. Since the fetch robot has 8 DOFs, our labels are shaped as 8x10 arrays. To build the distribution, we sample the path of each motion plan solution τ_i and count how many samples lie in a given bin for each DOF. We then normalize these counts to get a distribution.

We also perform data augmentation by rotating each input environment by 90, 180, 270 degrees.

2.3.2 *Network Architecture*

The network consists of an encoder followed by a fully connected layer as shown in Figure 2.10. The encoder consists of 6 encoding blocks. Each block consists of a 3D convolution layer followed by a max pooling layer. The convolution layers in each encoding block consist of 32, 64, 128, 256, 512, 1024 filters respectively. Each convolution layer has a kernel of shape [3,3,3] and stride of 1. The max pooling layers have a window of size 2 and stride of 2. The output of each convolution layer is batch normalized. We use ReLu as the activation function for the convolutional layers. The final layer outputs a 1024 vector encoding of the input which is connected to a fully connected layer outputting a 80 length vector which is reshaped into a 8×10 matrix and a softmax layer is applied to this to get the final predicted probability distribution. The loss function is the KL Divergence between our predicted distribution and the ground truth.

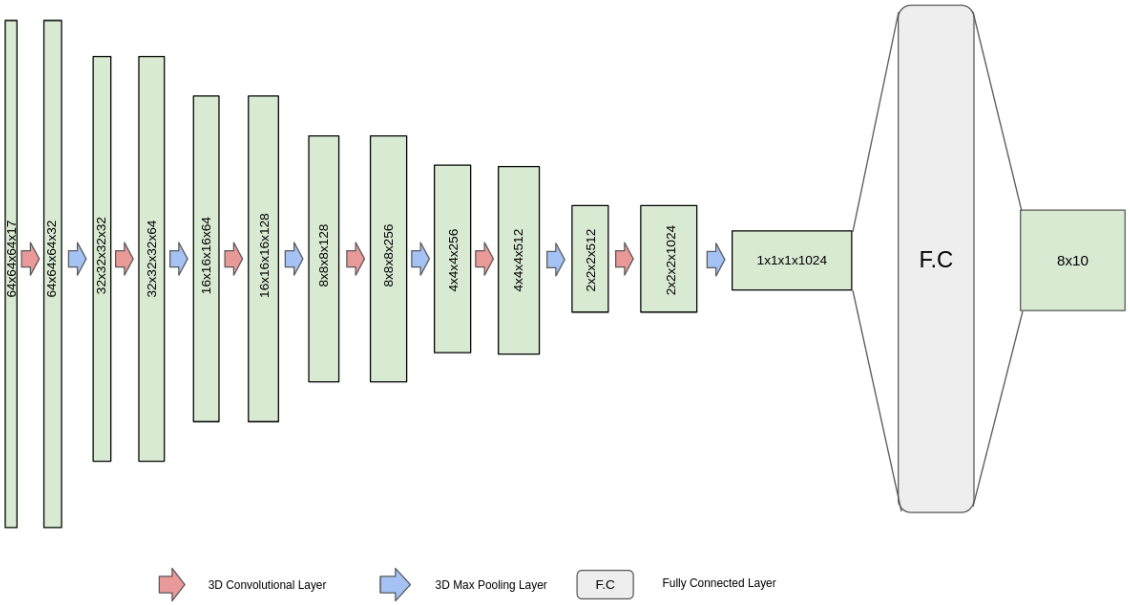

Figure 2.10: Network architecture of the model used in the 8 DOF robot domain

2.3.3 Training

Our training set comprises a total 6000 samples. We use a batch size of 5 with random shuffling. We use Adam for the optimizer with a learning rate of 0.00001 and train for 10000 iterations.

Chapter 3

RESULTS

For evaluating the predictions, we run the LLP and LL-RM planners on various test environments. LLP and LL-RM are set to use 5% of the samples generated by the network. We use RRT, RRTConnect and RRT* as the baseline for our comparisons. We use the implementations of baseline planners provided by (Şucan *et al.*, 2012). All planners are given a step size of 0.2. Each motion planning problem consists of planning from a fixed start configuration to a fixed goal configuration within a given time limit. We report the success rate out of 100 runs for each test environment.

3.1 3 DOF Robot

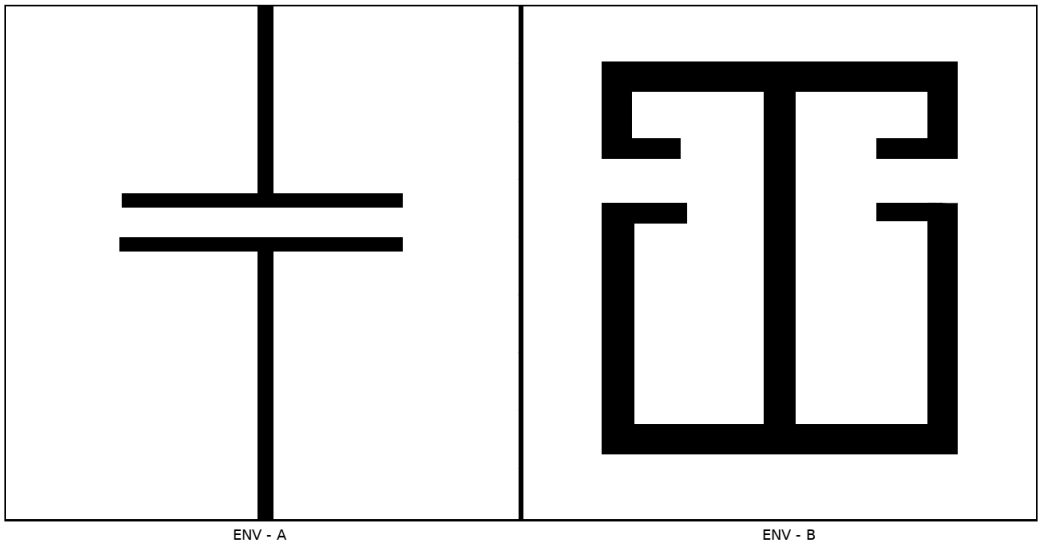

Figure 3.1: The test environments used for evaluation in the 3DOF robot domain. (left) ENV - A (right) ENV - B

Figure 3.2 shows the predictions after training the model from the training set. Figure 3.3 shows the predictions of the model on the 2 test environments shown in

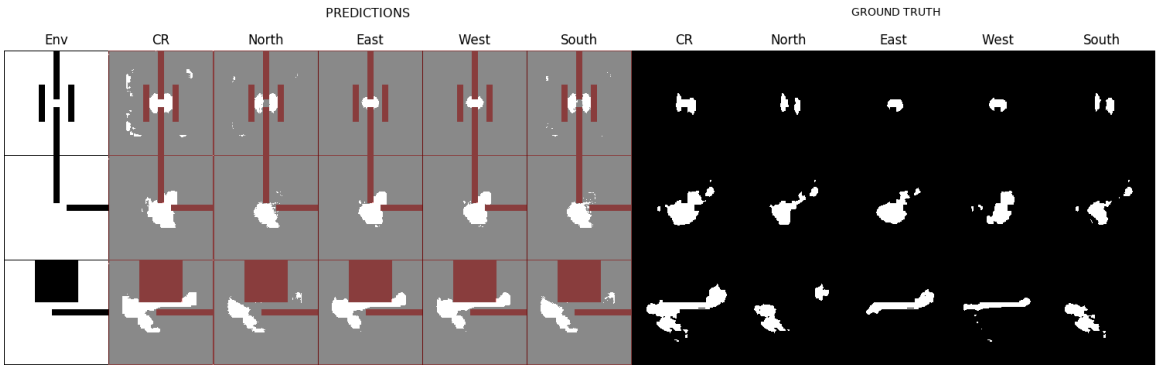

Figure 3.2: Predictions and ground truth for 3 training environments in the 3DOF robot domain

Figure 3.1. We can observe in Figure that the network is able to correctly identify the different orientations of the robot in the narrow passages of the test environments. In ENV-A, for a planning time of 20s, RRT, RRTConnect and RRT* achieve 51%, 3% and 1% success rates while LLP and LL-RM achieve 97% and 91% success rates respectively. In ENV-B, for a planning time of 4s, RRT, RRTConnect and RRT* achieve 4%, 13% and 0% success rates respectively while LLP and LL-RM both achieve 100% success rates. For both environments, LL-RM is given 1 second to build the roadmap. Figure 3.4 illustrates the planning time vs success rate for the four planners.

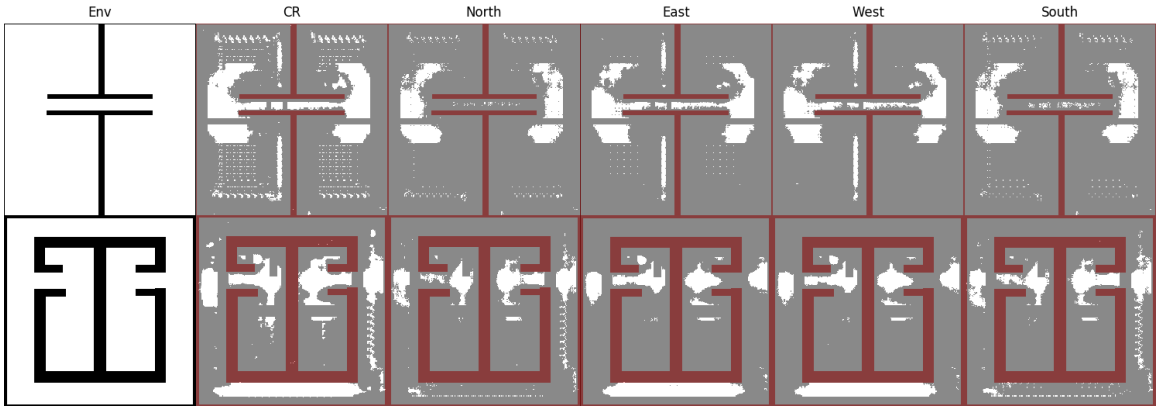

Figure 3.3: Predictions on the 2 test environments depicted in Figure 3.1

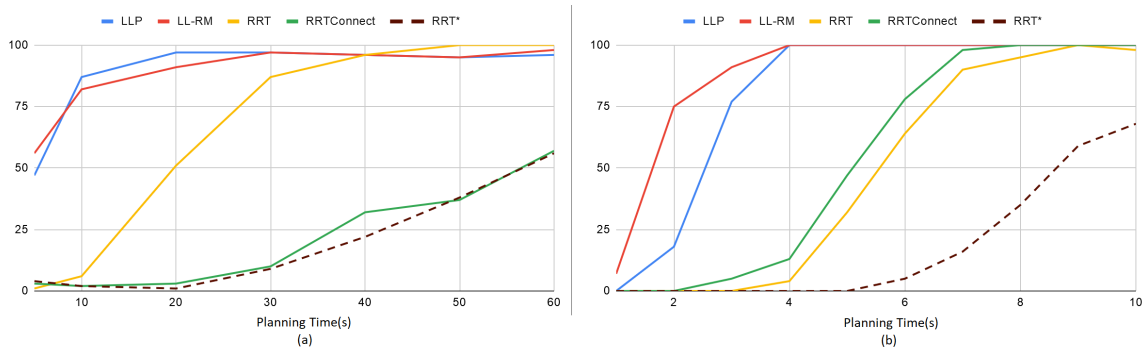

Figure 3.4: (a) Planning Time(s) vs Success Rate(%) on ENV-A (b) Planning Time(s) vs Success Rate(%) on ENV-B

3.2 4 DOF Robot

This domain consists of the goal configuration included as part of the input. Figure 3.5 shows network predictions on the train set. Figure 3.6 shows network predictions on the test environments. Different patches of grayscale colors in the ground truth and predictions imply the different DOF bins for the channel at the respective pixels. We also demonstrate results for different goal configurations and show that the network is able to learn the concept of a goal region and directs sampling towards it. ENV-C, ENV-D, ENV-E, and ENV-F are test environments consisting of the same obstacle space but differ in the goal configuration. Figure 3.6 depicts the results of the network for each test environment. We can observe that the identified critical regions are directed towards the goal depicted by the green pixels. On ENV-E, for a fixed planning time of 300 seconds, RRT, RRTConnect and RRT* achieve 7%, 2% and 1% success rates respectively while LLP and LL-RM achieve 100% success rates. For ENV-G, for a fixed planning time of 60 seconds, RRT, RRTConnect and RRT* achieve 25%, 82% and 25% success rates while LLP and LL-RM achieve 95% and 97% success rates respectively. For both environments LL-RM is given 1 second to build the roadmap. Figure 3.7 illustrates the planning time vs success rate for ENV-E and ENV-G.

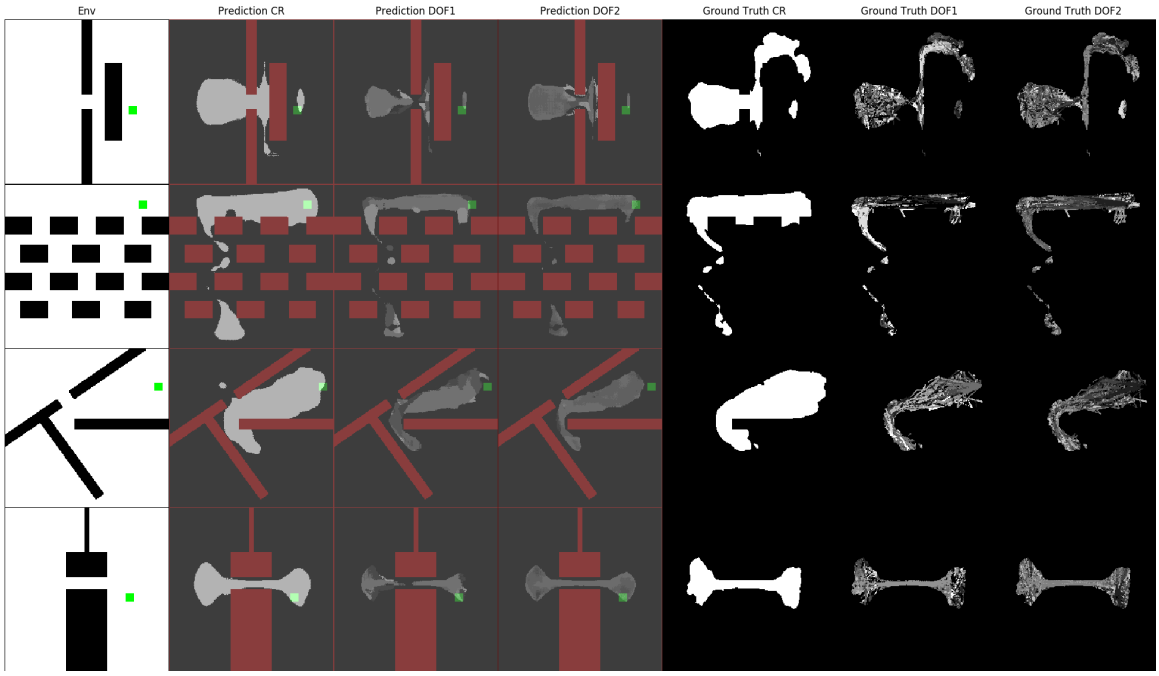

Figure 3.5: Predictions and ground truth for 4 training environments in the 4 DOF robot domain. The green square denotes the goal region for the given input

3.3 8 DOF Robot

This domain consists of the goal configuration and robot location included as part of the input. The four test environments are shown in Figure 3.8. ENV-H consists of a table with obstructions and the robot needs to reach the pose depicted in Figure 3.8. For a fixed planning time of 30 seconds, RRT, RRTConnect and RRT* achieve 23%, 83% and 40% success rates while LLP and LL-RM both achieve 100% success rates. ENV-I is a kitchen environment where the robot needs to pick up the rectangle object from the sink in front of it. For a fixed planning time of 30 seconds, RRT, RRTConnect and RRT* achieve 0%, 4% and 21% success rates respectively while LLP and LL-RM achieve 100% success rates. ENV-J consists of a common cutting machine used in manufacturing environments and the robot needs to pick up the rectangle object kept on the machine. For a fixed planning time of 30 seconds, RRT, RRTConnect and RRT* achieve 0%, 1% and 49% success rates respectively

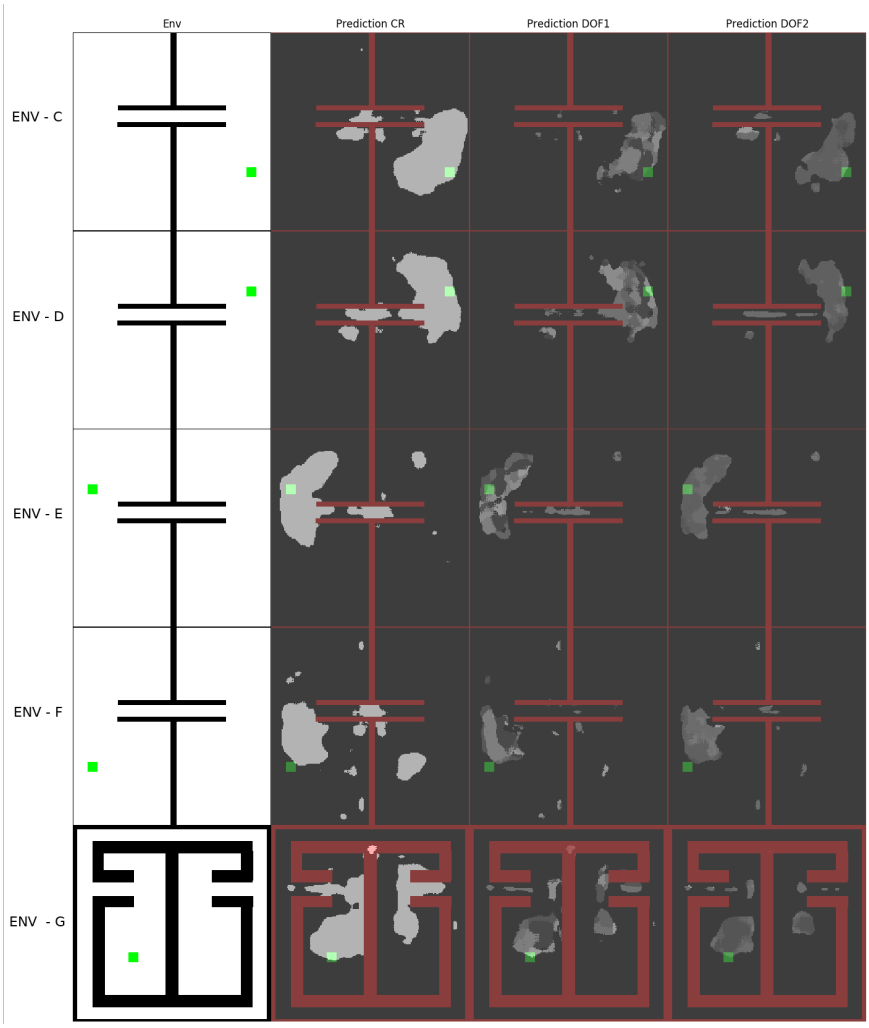

Figure 3.6: Predictions on 2 test environments. The first 4 rows depict different goal regions for the same environment and their corresponding predictions. The green square denotes the goal region for the given input

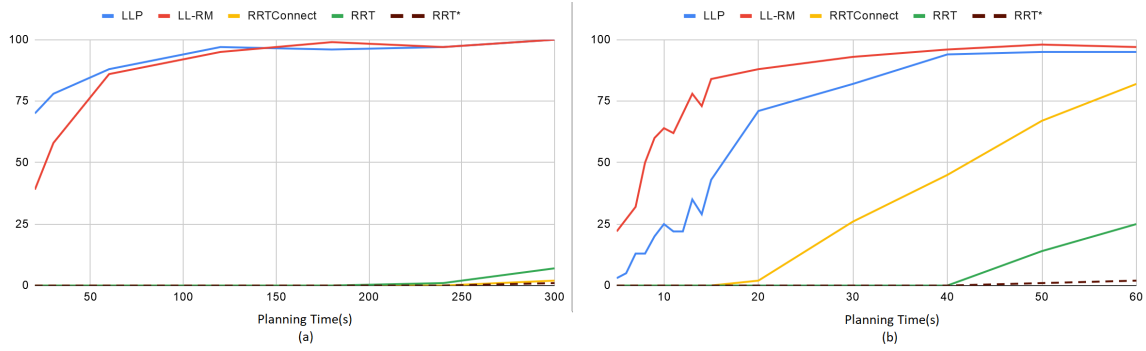

Figure 3.7: (a) Planning Time(s) vs Success Rate(%) on ENV-E (b) Planning Time(s) vs Success Rate(%) on ENV-G

while LLP and LL-RM achieve 100% success rates. ENV-K consists of an autoclave machine also used in manufacturing environments and the robot needs to pick up the rectangle object kept inside the machine. For a fixed planning time of 30 seconds, RRT, RRTConnect and RRT* achieve 0%, 79% and 2% success rates respectively while LLP and LL-RM achieve 100% success rates. For both environments, LL-RM is given 2 seconds to build the roadmap. We increase the time limit for building the roadmap since the higher number of dimensions in the configuration space increases complexity of the problem and time required for building a sufficiently space covering roadmap. Figure 3.9 illustrates the planning time (seconds) vs success rate (%) for the four environments.

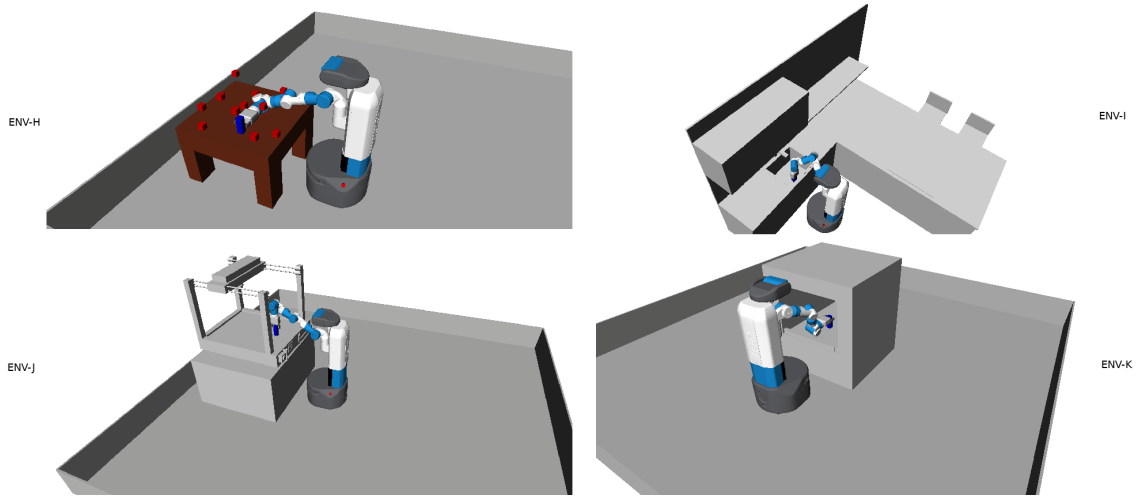

Figure 3.8: The four test environments for the 8 DOF robot domain. The robot needs to reach the shown arm and torso configuration to be able to pick up the blue rectangular object

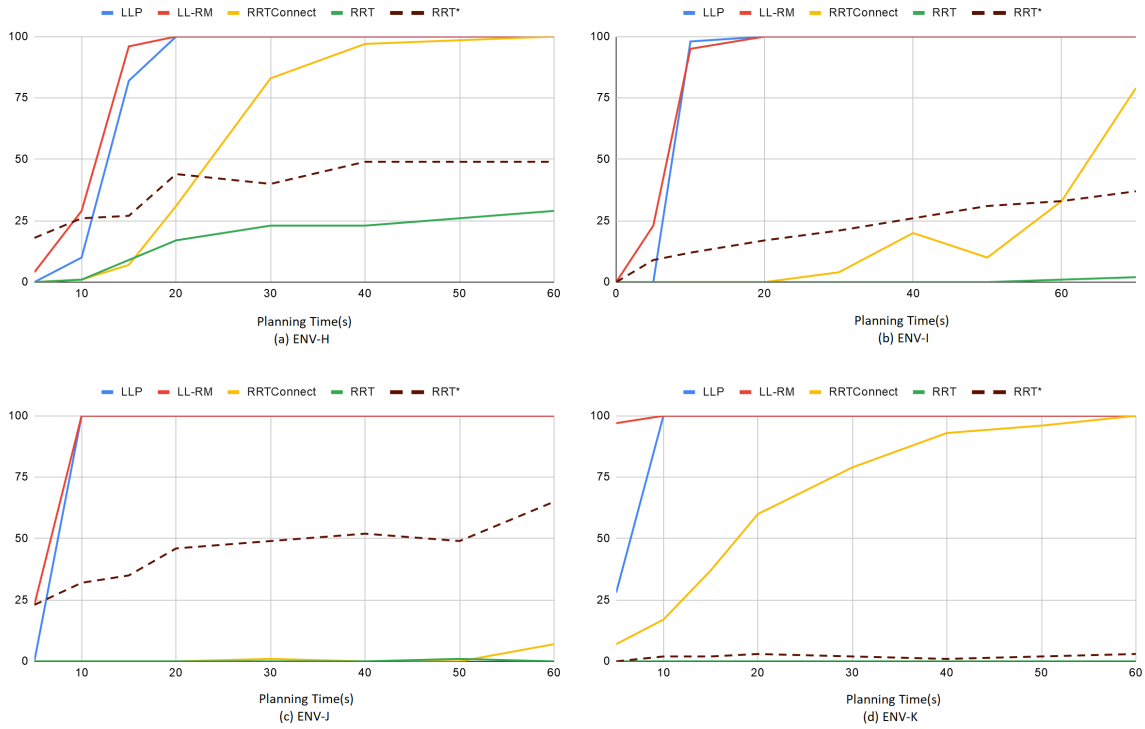

Figure 3.9: (a) Planning Time(s) vs Success Rate(%) on ENV-H (b) Planning Time(s) vs Success Rate(%) on ENV-I (c) Planning Time(s) vs Success Rate(%) on ENV-J (d) Planning Time(s) vs Success Rate(%) on ENV-K

Chapter 4

CONCLUSIONS AND FUTURE WORK

We developed and evaluated an approach to learn critical regions in the configuration space of the robot. These regions have low probability of being sampled, however most solutions require the robot pass through these regions. We use convolutional encoder decoder networks to learn a distribution that can be used for sampling the configuration space of the robot more efficiently. We used the Link and Learn planners to leverage the samples generated from the predicted distributions. Empirical evaluations show that the generated samples help speed up motion planning and provide better success rates at finding a solution than state of the art planners.

For future work, an ablation study can be performed to reduce the number of parameters and network size from the proposed architectures. This can help achieve faster inference and training time. The proposed architectures vary in different domains so a unified architecture can be established for the provided approach. The predicted critical regions for the 8 DOF domain albeit having improved the planning time have scope of more refinement and structure to improve identification of the critical regions. The ability of neural networks to identify critical regions for a motion planning problem can be generalized to a task and motion planning problem to identify abstract states for performing the task.

REFERENCES

Azzini, `https://github.com/andreaazzini` (2017).

Burns, B. and O. Brock, "Toward optimal configuration space sampling.", in "Robotics: Science and Systems", pp. 105–112 (Cambridge, USA, 2005).

Choi, Y., M. Choi, M. Kim, J.-W. Ha, S. Kim and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 8789–8797 (2018).

Diankov, R. and J. Kuffner, "Openrave: A planning architecture for autonomous robotics", Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 79 (2008).

Gammell, J. D., S. S. Srinivasa and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs", in "2015 IEEE international conference on robotics and automation (ICRA)", pp. 3067–3074 (IEEE, 2015).

Havoutis, I. and S. Ramamoorthy, "Motion synthesis through randomized exploration on submanifolds of configuration space", in "Robot Soccer World Cup", pp. 92–103 (Springer, 2009).

Ichter, B., J. Harrison and M. Pavone, "Learning sampling distributions for robot motion planning", in "2018 IEEE International Conference on Robotics and Automation (ICRA)", pp. 7087–7094 (IEEE, 2018).

Ichter, B., E. Schmerling, T.-W. E. Lee and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning", arXiv preprint arXiv:1910.03701 (2019).

Ioffe, S. and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", arXiv preprint arXiv:1502.03167 (2015).

Itti, L. and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention", Vision research 40, 10-12, 1489–1506 (2000).

Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980 (2014).

Kumar, R., A. Mandalika, S. Choudhury and S. S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning", arXiv preprint arXiv:1907.09574 (2019).

LaValle, S. M. and J. J. Kuffner Jr, "Randomized kinodynamic planning", The international journal of robotics research 20, 5, 378–400 (2001).

Lehner, P. and A. Albu-Schäffer, "Repetition sampling for efficiently planning similar constrained manipulation tasks", in "2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 2851–2856 (IEEE, 2017).

Molina, D., K. Kumar and S. Srivastava, "Learn and link: Learning critical regions for efficient planning (extended version)", Arizona State University, School of Computing, Informatics, and Decision System Engineering, Tech. Rep. ASUCISE-2019-002 (2019).

Pan, J., S. Chitta and D. Manocha, "Faster sample-based motion planning using instance-based learning", in "Algorithmic Foundations of Robotics X", pp. 381–396 (Springer, 2013).

Radford, A., L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", arXiv preprint arXiv:1511.06434 (2015).

Ronneberger, O., P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in "International Conference on Medical image computing and computer-assisted intervention", pp. 234–241 (Springer, 2015).

Springenberg, J. T., A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving for simplicity: The all convolutional net", arXiv preprint arXiv:1412.6806 (2014).

Şucan, I. A., M. Moll and L. E. Kavraki, "The Open Motion Planning Library", IEEE Robotics & Automation Magazine 19, 4, 72–82, `https://ompl.kavrakilab.org` (2012).

Urmson, C. and R. Simmons, "Approaches for heuristically biasing rrt growth", in "Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)", vol. 2, pp. 1178–1183 (IEEE, 2003).

Wise, M., M. Ferguson, D. King, E. Diehr and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications", in "Workshop on autonomous mobile service robots", (2016).

Zhang, C., J. Huh and D. D. Lee, "Learning implicit sampling distributions for motion planning", in "2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 3654–3661 (IEEE, 2018).

Zucker, M., J. Kuffner and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners", in "2008 IEEE International Conference on Robotics and Automation", pp. 3757–3762 (IEEE, 2008).

APPENDIX A

EXPERIMENTS WITH THRESHOLDING SALIENCY MAPPING OUTPUT

For the 4 DOF domain, we also experiment with different thresholds on the outputs of the saliency mapping performed using (Itti and Koch, 2000). Figure A.1 depicts 3 different outputs after thresholding the given saliency mapping output. For the experiments described in this thesis we use a threshold value of 70. However, we also test our planners by training the model on data generated using a threshold value of 40. Figure A.2 shows the plots for planning time in seconds vs the success rate for the two test environments in the 4 DOF domain namely ENV-E and ENV-G using this dataset. On Env-E, for a fixed planning time of 180 seconds, LLP and LL-RM achieve 100% success rates when using a threshold of 40. For a threshold of 70, LLP and LL-RM achieve 96% and 99% success rates on Env-E. For Env-G, for a fixed planning time of 20 seconds, LLP and LL-RM achieve 46% and 50% success rates respectively when using a threshold of 40. For a threshold of 70, LLP and LL-RM achieve 71% and 88% success rates respectively. On visual inspection of the thresholded outputs, we observe that as threshold values decrease, the critical regions expand and tend to become more noisy.

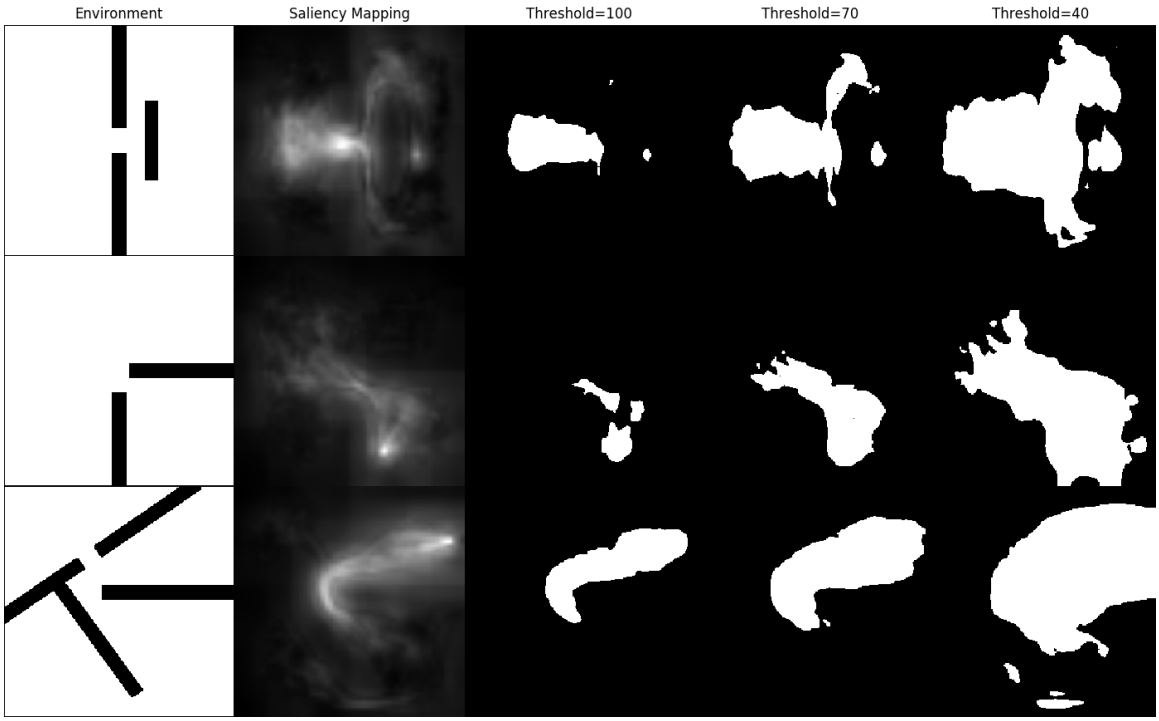

Figure A.1: Outputs of thresholding the saliency mapping output (second column) using 3 different threshold values (100, 70, 40)

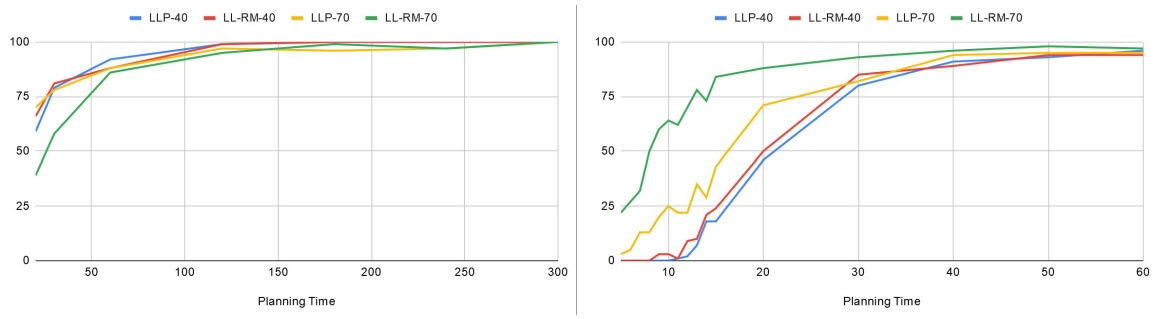

Figure A.2: Planner results when using data generated with threshold value = 40 and 70 (a) Planning Time(s) vs Success Rate(%) on Env-E (b) Planning Time(s) vs Success Rate(%) on Env-G

APPENDIX B

EXPERIMENTS WITH CVAE AND GANS

For the SE(2) - 4 DOF environment, we also experimented with another approach where the network was tasked with predicting the precise DOF value rather than classifying for a bin in the DOF's limits. We took inspiration from (Ichter *et al.*, 2018) and experimented with a CVAE as well as DCGANs (Radford *et al.*, 2015) however the networks were difficult to stabilize and required much more training time as compared to the classification approaches using encoder-decoder networks.

APPENDIX C

RAW DATA

The raw data for each of the domains can be found at the following urls:

- 3DOF:
  https://drive.google.com/file/d/1JBYjZ2alczjK6tmX3y5mlfekMU77dvv-
- 4DOF:
  https://drive.google.com/file/d/1zR4voKaFMncA5I9FaoFpJMK4lvvtmlrI
- 8DOF:
  https://drive.google.com/file/d/1-6fBHWtL_D8xYN2Hyw7Tx_D80y04P-m37