POLITECNICO
DI TORINO

Master of Science in Mechatronic Engineering

Dynamic Neural Networks to Generate Robotics Trajectories

Final Project Work

UPE/Poli Supervisor:
Prof. Dr. Ruben Carlo Benante

POLITO Supervisor:
Prof. Marcello Chiaberge

Author 1:
Muhammad Naeem Akhtar

Author 2:
D S A Asifur Reza

Universidade de Pernambuco
Poli - Escola Politécnica de Pernambuco
Graduação em Engenharia de Controle e Automação

Author 1:
Muhammad Naeem Akhtar

Author 2:
D S A Asifur Reza

Dynamic Neural Networks to Generate Robotics Trajectories

Work Presented on the Exchange Program in Universidade de Pernambuco in Control and Automation department to obtain the Master of Science Degree in Mechatronic Engineering, Department of Control and Computer Engineering (DAUIN) of Politecnico di Torino Italy.

UPE/Poli Supervisor: Professor Dr. Ruben Carlo Benante

POLITECNICO
DI TORINO

Dynamic Neural Networks to Generate Robotics Trajectories

Author 1:
Muhammad Naeem Akhtar

Author 2:
D S A Asifur Reza

Professor Dr. Ruben Carlo Benante – UPE/Poli Advisor

Professor Dr. João Fausto Lorenzato de Oliveira – Componente da Banca

Recife

19 December 2018

Acknowledgement

In the name of Allah, the Most Gracious and the Most Merciful.

I would like to express my appreciation:

To Politecnico di Torino, Italy and Universidade de Pernambuco, Brasil for providing the assistance required to successfully complete this work in a way of giving the opportunity to follow courses related to the thesis work (Robotics taught by Professor Bona Basilio in POLITO) and (Artificial Intelligence taught by Professor Ruben Carlo Benante in UPE/Poli).

To both the advisors Professor Ruben Carlo Benante (Poli/UPE) and Professor Marcello Chiaberge (POLITO), for accepting me to do research under their guidance and to understand the problems in different ways to find the right solutions.

To my friends and colleagues Carlos Alberto Nogueira de Souza Junior, Daidson Fonseca Alves and Gerônimo De Sá Barreto Neto, they were always helpful from the beginning and made it easy to complete the work on time.

Finally to my family, without whom none of this would have felt, for their continuous encouragement throughout learning, for unconditional support and for teaching that in life there are many obstacles, which we must face with maturity to achieve the much-desired goal. To Father and Mother, my role models.

MUHAMMAD NAEEM AKHTAR

Acknowledgement

At first, all the praises and thanks go to Almighty Allah for giving me the opportunity to attend the course and make me capable to prepare this thesis work. I could not overcome my whole challenges and could not make my successes without His willingness and help.

I owe my earnest gratitude to Politecnico di Torino (Turin, Italy) for providing me the opportunity to do the thesis work in Universidade de Pernambuco (Recife, Brazil) and giving me all the cooperation and guidance, which made me complete the research properly. The support I had got from both the Universities helped me to acquire a whole new experience which encouraged me to accomplish this paper satisfactorily.

I gratefully acknowledge the contributions of my supervisor Prof. Dr. Ruben Carlo Benante (UPE/POLI), who guided and instructed me throughout the completion of the research work. He instructed me to prepare this thesis paper and provided me his all-out efforts though being extremely busy. I would like to thank also Prof. Marcello Chiaberge (POLITO) for supporting me in different phases.

I am also thankful to my Group Members specially Carlos, Daidson and Geronimo as without their cooperation, coordination and help, it was really difficult to complete this research paper. This study was possible for me to finish because of the assistance of the concerned authorities who had given me their precious time and sincere efforts.

This work needed a lot of guidance and assistance from many people and I am extremely privileged to get all those throughout the completion of this research. The final outcome of this research could not be possible without their supervision and assistance. I owe my heartiest gratitude to them.

Last but not the least I would like to give my due respect and thanks to my mother who always encourages me in all aspects of my life. Also, my thanks go to my all family members who inspired me to do something different in life.

D S A Asifur Reza

December, 2018.

Resumo

O corrente trabalho apresenta Redes Neurais Artificiais capazes de gerar trajetórias de estado a partir do mapeamento de estados de espaço de uma dada estrutura. Os modelos focados são os de Redes Neurais Não-Supervisionadas, tais como a Growing Neural Gas (GNG), (Fritzke), Grow When Required (GWR), (Marsland); e uma nova rede com Topologia Dinâmica chamada de State Trajectory Generator (STRAGEN Off-line e STRAGEN On-line), (Benante). Este ultimo modelo, proposto por Benante, é uma importante contribuição acerca de mapeamento de espaço de dinâmicas de robôs, tornando possíveis as gerações de trajetórias. Os modelos permitem a utilização de vários critérios para sintetizar uma trajetória ideal de acordo com a área de interesse dos diferentes atributos que um mesmo domínio pode apresentar. Cada modelo trabalha em três fases: Treino, Geração de Trajetórias e Fase de Validação de Erros, com exceção do STRAGEN Off-line, que tem uma fase de Poda adicional, performada antes da fase de Geração de Trajetórias. Essas ferramentas são usadas para calcular a performance de robôs e para implementar e executar trajetórias otimizadas dos mesmos. Simulações e resultados para os domínios robóticos (com um braço robô bidimensional e um manipulador robótico PUMA 560) são considerados no fim como exemplos de diferentes domínios.

KEYWORDS: Two-Link (2D) and PUMA 560 (3D) manipulators, Forward Kinematics, Artificial Neural Networks with Dynamic Topology (GNG, GWR, STRAGEN), Trajectory Generation.

Abstract

This postulation work presents Artificial Neural Networks fit for generating state trajectories from the state space mapping of a framework. The models with the main focus are Unsupervised Neural Networks, which are Growing Neural Gas (GNG), (Fritzke), Grow When Required (GWR), (Marsland) and a new network with Dynamic Topology called State Trajectory Generator (STRAGEN Off-line and STRAGEN On-line), (Benante). The latter is a model proposed by Benante, is an important contribution towards mapping robot space dynamic to be able to generate trajectories. The models permit the utilization of various criteria for the synthesis of an ideal trajectory as per the area of interest of different attributes of the same domain. Each model works in three phases: Training, Trajectories Generation and Validation Error phase, apart from STRAGEN Off-line, which has an additional Pruning Phase, that is performed before the Trajectory Generation phase and after the training phase. These tools used for measuring robot performance and for implementing and executing optimized trajectories in the robots. Simulations and results for the robotic domains (with a two-dimensional and PUMA 560 robotic manipulators) are considered in the end as examples of different domains.

KEYWORDS: Two-Link (2D) and PUMA 560 (3D) manipulators, Forward Kinematics, Artificial Neural Networks with Dynamic Topology (GNG, GWR, STRAGEN), Trajectory Generation.

Summary

List of Figures

List of Tables

Table of Symbols

SYMBOLS	DESCRIPTIONS
$\boldsymbol{\alpha_i}$	DH parameter that defines rotation angle between two axes.
$\boldsymbol{\alpha}$	Parameter or auxiliary rate for several equations. In GWR is used $\alpha_n = \alpha_n = 1{,}05$ as parameters for the function non-trigger frequency.
$\boldsymbol{\alpha_f}$	Final learning rate of STRAGEN. The value is $\alpha_f = 0.1$.
$\boldsymbol{a_{max}}$	Maximum age allowed for a connection before it is eliminated.
$\boldsymbol{\overline{a}}$	Maximum activity limit that prevents STRAGEN to create nodes.
$\boldsymbol{a}$	Vector of activities of the winning node. a_k is the activity of subgroup $1 \leq k \leq l$ of the winning node.
$\boldsymbol{a_T}$	Maximum activity limit of the winning node in relation to an input pattern for not to insert a new node.
$\boldsymbol{a_i}$	Minimum signed distance in DH parameter.
$\boldsymbol{A}$	Set of network nodes, being $\lvert A \rvert$ the total number of nodes of the set (cardinality).
$\boldsymbol{A_i^{i-1}(q_i)}$	Homogeneous transformation matrix.
$\boldsymbol{\beta}$	Decay rate of local error in GNG.
$\boldsymbol{B^{(0)}}$	Non-normalized database, $L \times D$ dimensions, containing patterns for training.

SYMBOLS	DESCRIPTIONS
$\boldsymbol{B}$	Normalized database, $L \times D$ dimensions, containing input patterns for training.
$\boldsymbol{C}$	Set of connections, where $\mid C \mid$ is the total number of connections (cardinality).
$\boldsymbol{C_{i,j}}$	Connection function. 1 if nodes i and j are connected, and 0 otherwise.
$\boldsymbol{c_{n1,n2}}$	Connection c between nodes n_1 & n_2.
$\boldsymbol{\chi}$	The threshold for removing connections. If $\mid \mathrm{N}(s_1) \mid > 2$, remove all connections $c_{s_{1,n_i}}$ from set C for which we have $\mathrm{n} \in \mathrm{N}(s_1)$ and $Dist(s_1, \mathrm{n}) > \chi$.
$\boldsymbol{d_i}$	DH parameter that defines coordinate of $O_{i'}$.
$\boldsymbol{D_A}$	Dimension of a topological map A representative of input data M.
$\boldsymbol{D_m}$	Dimension of the input patterns $\boldsymbol{\xi}$ and vector weights $w \in M$.
$\boldsymbol{Dist(i,j)}$	Euclidean distance between i & j.
$\boldsymbol{Dist_1(s_i, s_j)}$	Smallest segment of a formed triangle by nodes s_1, s_2 and s_3, with s_i and s_j representing the two nodes belonging to the segment.
$\boldsymbol{Dist_2(s_i, s_k)}$	Smallest segment of a formed triangle by nodes s_1, s_2 and s_3, with s_i and s_k representing the two nodes belonging to the segment.
$\boldsymbol{\varepsilon}$	Learning rate. ε_b is the learning rate of the node winner. When not mentioned, use ε_b = 0.2. ε_n is the rate to the neighboring node of the winning node.

SYMBOLS	DESCRIPTIONS
$\boldsymbol{\varepsilon(t)}$	Exponential decay function.
$\boldsymbol{E_1}$	Measurement Error to evaluate the average size of the Connections.
$\boldsymbol{E_2}$	Measurement of Cost that evaluates the network's ability to minimize the distances between the input patterns and the nodes.
$\boldsymbol{E_v}$	Validation error.
$\boldsymbol{H}$	Neighborhood criterion of STRAGEN.
$\boldsymbol{E}$	Set of inactive connections, i.e., that never tripped by connecting a winning node s_1 and s_2, in the phase of STRAGEN.
$\boldsymbol{f(t,n)}$	Energy Diffusion Function.
$\boldsymbol{h_y(k)}$	Rank proximity function of NG.
$\boldsymbol{h_i}$	Size of firing variable for node i.
$\boldsymbol{\widetilde{h_\iota}}$	Relative indicator Frequency .
$\boldsymbol{h^{(0)}}$	Initial strength.
$\boldsymbol{I}$	Number of iterations for the validation phase.
$\boldsymbol{K}$	Constant $K < 1$.
$\boldsymbol{k}$	Number of connections emanating from the winning node s_1.

SYMBOLS	DESCRIPTIONS
$\boldsymbol{k-simplex}$	Dimension of the GCS network structure.
$\boldsymbol{\lambda}$	Number of iterations performed for GCS and GNG create a new node..
$\boldsymbol{\mu}$	Energy dispersion constant in the function f.
$\boldsymbol{o_0x_0y_0}$	System coordinate of origin .
$\boldsymbol{P}$	Percentage used to calculate the activity threshold $\overline{a_k}$.
$\overline{\omega}$	An empirically determined optimal constant, which the value is 1.5.
$\boldsymbol{p(\xi)}$	MB procedure that returns some random pattern $\boldsymbol{\xi}$ on first execution.
$\boldsymbol{P(\xi)}$	Distribution of probability of input signals.
$\boldsymbol{\phi_i}$	Candidates of input patterns in the MB procedure, with $i = 1,\ldots,Q$.
$\boldsymbol{Q}$	Number of candidates in the MB procedure. When not quoted, use Q = 10.
$\boldsymbol{\rho}$	Decent learning rate of STRAGEN.
$\boldsymbol{\sigma_f}$	Maximum number of times that a node is supposed to fire.
$\boldsymbol{\sigma_{s_1}}$	Number of firing of node s_1.

SYMBOLS	DESCRIPTIONS
s_1	Nearest node (winner) of the input pattern by some criterion of proximity.
s_2	Second node closest to the input pattern by some proximity criterion.
$S(t)$	Stimulus force, usually S (t) = 1.
θ_i	DH parameter of joint angles.
θ_k	Joint angles of robots.
t	Iteration or current time.
t_{max}	Maximum number of training iterations.
t_b & t_n	Decay parameters for the frequency function in GWR. Suggested value: t_b = 3.33 and t_n =14.3.
T	Transformation Matrix.
V_k	Group k of homogeneous information that makes up the Standard input.
V_η	Subvector of the vector of weights composed by the group (s) of homogeneous information used as a neighborhood criterion.
V_ζ	Subvector of the vector of weights composed of the group (s) of homogeneous information and used as an activity criterion.
ξ	Dimension of input pattern.
$\xi(t)$	Input pattern presented at time t.

SYMBOLS	DESCRIPTIONS
$\boldsymbol{\xi_h}$	Subvector of the input pattern ξ which represents the group h defined as the neighborhood criterion.
$\boldsymbol{\xi_\zeta}$	Subvector of the input pattern ξ which represents the group ζ defined as an activity criterion.
$\boldsymbol{\xi_{\eta,q_i}}$	Subvector η of an input vector ξ at point q_i.
$\boldsymbol{V(c)}$	Voronoi field.
$\boldsymbol{w_i}$	Weight vector.
$\boldsymbol{y(t)}$	Function with exponential decay in time.
$\boldsymbol{\zeta}$	Activity criteria of STRAGEN.

Chapter 1

Introduction

In ongoing ages, Robotics have spread to all segments in our day by day life. The use of mechanical technology, mainly Robotics is quickly expanding. Over the course of centuries, human beings have continuously endeavored to quest substitutes that would efficient enough to mimic their behavior in the numerous instances of interaction with the surrounding environment. Several enthusiasts have inspired this incessant exploration referring to philosophical, financial, social and scientific principles.

The term 'Robot' was first introduced by the Czech playwriter Karel Capek, who wrote the play 'Rossum's Universal Robots' in 1920. The image of the robot as a mechanical object begins in the 1940s when the Russian Isaac Asimov, the renowned science fiction writer, mentioned robot as an automation of human appearance but devoid of feelings. Afterward, the term has been introduced to a great diversity of mechanical devices such as autonomous land rovers, underwater vehicles, manipulators etc.

According to scientific interpolation of the science fiction scenario, the robot is understood as a machine that, independent of its exterior, is able to modify the environment in which it operates. Robotics are generally defined as the science studying the intelligent connection between perception and action. A virtual system that operates with some degree of autonomy, typically computer control, has at some point been called the robot.

> *The bridge between life and the machine is not trivial, and many questions that have always been open in philosophy are now, with the advent of computer science, receiving an answer with a new approach. This approach has in its vocabulary terms such as systems, sensitive to initial conditions, patterns, learning, prediction, artificial neural networks, emergent properties, states, representation of knowledge, planning, controls, etc*. (Benante, 2008)

Trajectory planning is a robotic task in which the robot must follow a prescribed route. It is formerly automated by an operator who controls the robots through an order of desired arm

positions, whereas these positions are kept in controller memory for further recall. The process described is time-consuming and inefficient, since during storage procedure the robot is out of operative action and the entire process ensures under the command of the robot operator. Additionally, in modern work, a robot is regularly required to perform several tasks and the robot controller needs to pursue different paths.

Trajectory planning can be carried out by neural networks for transient successions if the trajectory to be pursued is comprehended as an order of arm positions. The role of neural networks is to figure out how to relate successive states of a trajectory and to store that state changes for aggregate or fractional generation of scholarly directions. Throughout the reproduction, at the beginning of each sequence, the current positions of the robot are accessible to the network and it should respond with next one, until a target position has been satisfied.

> *Kelso suggests, for example, that the brain is a self-organized system, working according to synergistic laws, and for this reason it can present macroscopic phenomena such as vision, locomotion and muscular organization with the objective of picking up an object. Self-organized system can be adopted to find the robotics trajectory.* (Benante, 2008)

Barreto and Araujo (2002) proposed a self-organizing neural algorithm that learns the temporal order of input states in an unsupervised way. Discrete trajectory focuses are situated in feedforward weights while their pattern of occurrence in time is determined by lateral weights. The network can learn multiple trajectories with their points in common taking into account of fixed input units. In any case, this model can't handle trajectories with repeated points.

State trajectories include any valid sequences of state transitions of a learned or predetermined system, passing through possible intermediate states, starting from an initial point that can be given or determined by the system, and target point of the system. As it can be seen from the characterization, there is a wide range of domains in which this technique can be used, if for that the domain in question can be characterized by possible states and there are transitions between them.

Numerous models of dynamic topology have imperative attributes to be utilized to determine the robotics trajectories and which are the base to proceed with this work. Over the last centuries, many models had been introduced. Among these, the most important ones are: Self-Organizing Map (SOM) (Kohonen, 1987), Topology Representing Networks (TRN) (Martinetz, 1991), (Neural Gas and Competitive Hebbian Learning), Growing Cells Structures

(GCS) (Fritzke, 1994a), Growing Neural Gas (GNG) (Fritzke, 1997). The GWR model (Marsland, 2002), which is the result of improvement of previously mentioned models. Among several other important properties that will be studied in this algorithm, improvement should be done in the sense of adaptability and growth of the model's network .

More specifically, Benante (2008) proposed a new improved incremental self-organizing artificial neural network model called the State Trajectory Generator (STRAGEN), to generate state trajectories, in which the points provided are learned by the network in an unsupervised training guided by a procedure called Motor Babbling (MB). The optimality criteria to create connections between these points can be characterized in several types for each domain, being described in the own input pattern, and used by STRAGEN to create topological maps. The STRAGEN model has two approaches that differ in how they deal with the elimination of inappropriate states and connections between them, called STRAGEN Off-line and STRAGEN On-line.

This thesis work will mainly focus on three artificial neural network models: Growing Neural Gas (GNG), Grow When Required (GWR) and State Trajectory Generator (STRAGEN). The goal is to find trajectories of two-link planar manipulator and PUMA (Programmable Universal Manipulation Arm) 560 using these three algorithms. To implement these networks at first all possible points that the robot can move in 2-Dimensional and 3-Dimensional spaces have to be simulated respectively for two-link robot and Puma 560. A great idea is to use MATLAB software tool. Afterward, this data points can be trained by using these three networks according to their algorithm. As soon as the network is trained, it is possible to find state trajectories of the robot by energy diffusion algorithm.

This document is organized in 7 chapters. Chapter 2 reviews the general concepts of robotic systems. It includes forward kinematics and most importantly Denavit-Hartenberg (DH) convention from which data points of two-link robot and Puma 560 have been simulated. Chapter 3 describes some of the main models of artificial neural network (ANN) with fixed and dynamic topologies and raised some characteristics that were promising in these models. Chapter 4 reviews the GNG model, its algorithms and description of this method. Chapter 5 for GWR model. One can find a comparison between GWR and GNG at the end of this chapter. Chapter 6 presents a new dynamic topological model STRAGEN. This algorithm is subdivided into two approaches: STRAGEN Off-line and STRAGEN On-line. We will see their algorithm and difference among them at the end of this chapter. Chapter 7 emphasis on the results

generated by these three models, and a comparison among them. In addition, there is a more practical description of how the results were achieved. This chapter ends with the conclusion of this thesis work and improvements in future.

Chapter 2

Robotics System

In order to proceed with the thesis work, some general concepts should be discussed. This chapter describes the mechanical structures of robots, forward kinematics equation as well as Denavit-Hartenberg notation. Forward kinematics is concerned with the relationship between the joints of the robotic manipulator and the position of the end effector, for a given value of joint variables of the robot.

2.1 Robot Manipulators

A robot manipulator, in general consists of a sequence of rigid bodies (links) which are interconnected by means of articulations (joints). Normally a manipulator can be described in three sections which are:

- Arm, ensuring mobility.
- Wrist, conferring dexterity.
- End-effector, performing the task required of the robot.

The articulation between two consecutive links can be categorized into two section called prismatic joints and revolute joints. A prismatic joint (P) creates a linear relative motion between two links. On the other hand, a revolute joint (R) creates relative rotation between two links. The Degree of freedom (DOF) is related to the number of joints of a robot manipulator. It is distributed along the mechanical structure of a robot manipulator to have a sufficient number to execute a given task. In 3-Dimensional space, typically six DOFs are required where three DOFs are for positioning and other three are for orientation with respect to reference coordinate frame. If the number of DOFs are higher than the tasks variables, the manipulator is called Reduandant from a kinematics point of view, while in other case, if the DOFs are less than the task vaiables then the arm can not move properly.

The workspace represents the total volume swept out that the end-effector can access. Its shape and volumes are constrained by the geometric configuration of a manipulator as well as the presence of mechanical joints. The task required by the arm is to position the wrist. After that

is required to orient the end-effector. The entire set of points reachable by manipulator is called reachable workspace. On contrary, the workspace consists of points that manipulator can reach with an arbitrary orientation of end-effector is called dexterous workspace.

2.1.1 Forward Kinematics

Kinematic chain can be categorized into two sections: open kinematic and close kinematic. If there is only one sequence of links connecting the two ends of the chain is named as open. On the other hand, when a sequence of links create a loop is called closed kinematic. The aim of forward kinematic is to compute the position of end-effector.

Consider an open chain manipulator with n joints connected by n+1 links. Link 0 is conventionally settled to the ground. Each joint gives one DOF, correspond to a joint variable. To define a coordinate frame joined from Link 0 to Link n, it is reasonable to consider first the description of the kinematic relationship between consecutive links and then to obtain the general description of the manipulator. The position and orientation of the last frame with respect to the initial frame is given by a transformation matrix T_n^0 which is composed by homogeneous transformation matrix $A_i^{i-1}(q_i)$ (for $i = 1, \dots \dots n$) each of which is a function of each joint variables.

$$T_n^0 = A_1^0(q_1)A_2^1(q_2) \dots \dots \dots A_n^{n-1}(q_n)$$

Where q_i is the generalized coordinates for each joint. It can represent the motion of a multibody system, inherently taking into consideration the kinematic constraints acting on the system.

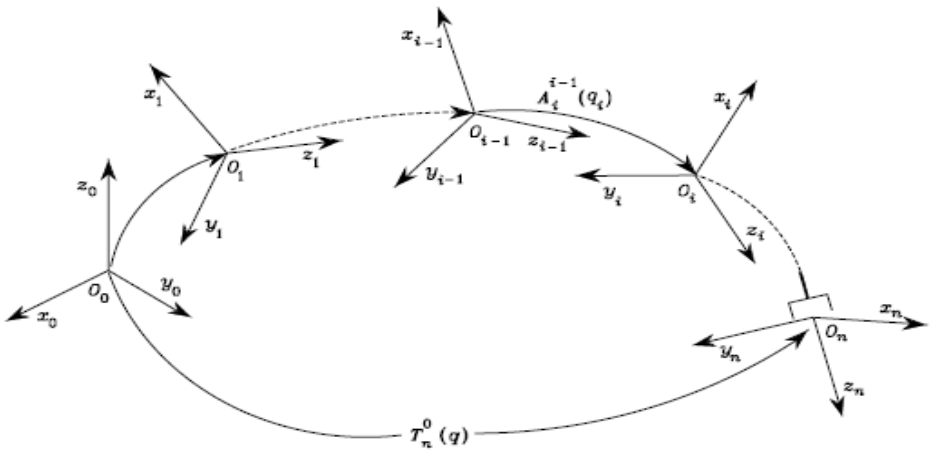


Figure 1: Coordinate transformation in open chain (Siciliano, 2008).

The actual coordinate transformation describing the position of orientation of end-effector with respect to the base frame is given by

$$T_e^b(q) = T_0^b T_n^0(q) \dots\dots\dots T_e^n$$

Where T_0^b represents the position and orientation of base with respect to Frame 0 and T_e^0 is w.r.t the end-effector.

2.1.2 Denavit-Hartenberg convention

Most commonly used convention for selecting reference frames in robotic systems is the Denavit-Hartenberg (1955) convention. Each homogeneous transformation matrix A_i is represented as a product of four basic transformation described below:

$$A_i = Rot_{z,\theta_i}, Trans_{z,d_i}, Trans_{x,\alpha_i}, Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the parameters $\theta_i, a_i, d_i, \alpha_i$ are linked to link i and joint i. The description of each parameters is:

- Parameter θ_i: defines the rotation angle between axes z_{i-1} and z_i about the axis x_i. The rotation is positive when counter-clockwise.
- Parameter a_i: defines the minimum signed distance between O_i and $O_{i\prime}$.
- Parameter d_i: defines coordinate of $O_{i'}$ along z_{i-1}.
- Parameter α_i: defines the rotation angles between axes x_{i-1} and x_i about the axis z_i. The rotation is positive when counter-clockwise.

Among the four parameters, two parameters are associated with translation and two parameters are associated with a rotation. Parameters, a_i, α_i, are always constant and depend only on the geometry of connection. While other two parameters θ_i, d_i, among them only one variable is depending on the type of joints.

- If joint i is revolute, then the variable parameter is θ_i
- If joint i is prismatic, then the variable parameter is d_i

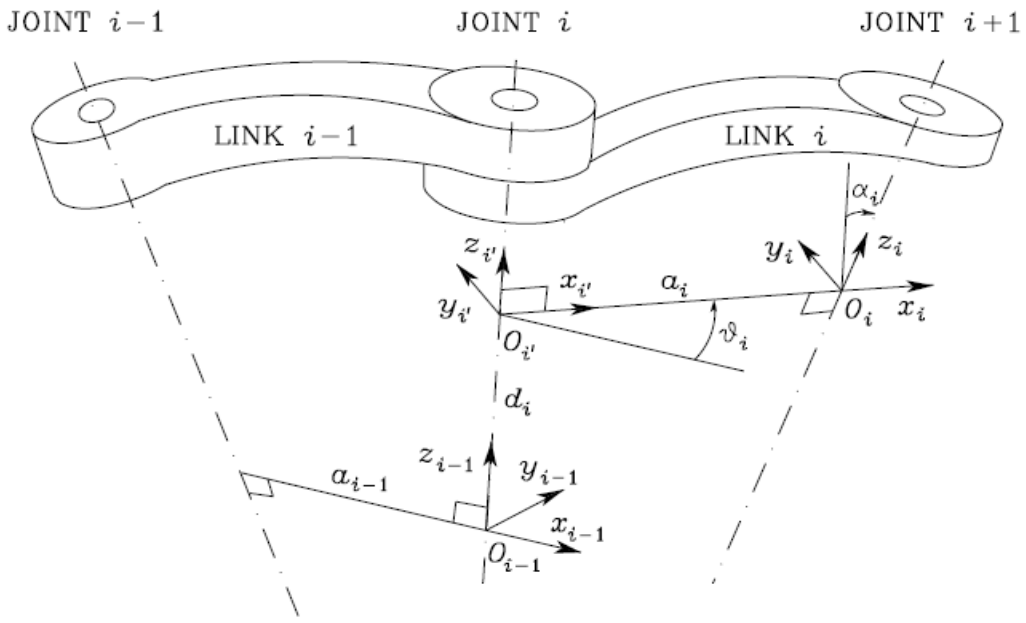


Figure 2: Denavit-Hartenberg kinematic parameters (Siciliano, 2008).

According to Figure 2, Axis i denote the axis of joint i connecting links i_{i-1} and i. To describe the Link frame i by DH convention is as:

- Choose axis z_i along the joint axis $i+1$.
- Locate the origin O_i at the intersection of axis z_i with common normal to axes z_{i-1} and z_i. Also, locate $O_{i'}$ at the intersection of the common normal with axis z_{i-1}.
- Choose axis x_i along the common normal to axes z_{i-1} and z_i with the direction from joint i to $i+1$.
- Choose the axis y_i to complete a right-handed frame.

DH convention is applied both for prismatic and revolute joints. Among these parameters, two are associated with translation and two are associated with rotations. Indeed, DH doesn't handle parallel z-axis very well. Nonetheless, most of the kinematics libraries accept DH parameter.

2.2 Two-Link Planar Manipulator

Robots can be mathematically defined as a chain of serially connected links, assuming that each joint has a degree of freedom, either translational or rotational. A robot manipulator with n joints will have $n + 1$ links, since each joint is connecting two links. For a robot with n links, numbered from $1\ to\ n$, the links are numbered from $0\ to\ n$, starting from the base. By this convention, joint i connects link $i - 1\ to\ i$ and vice versa. When joint i is actuated, link i and link i-1 moves.

Consider the two-link planar robot. By means of the convention of robotics, the system is a Revolute-Revolute (RR) robot. The joint axis $z_o\ and\ z_1$, pointing out of the page are not shown in the Figure 3.

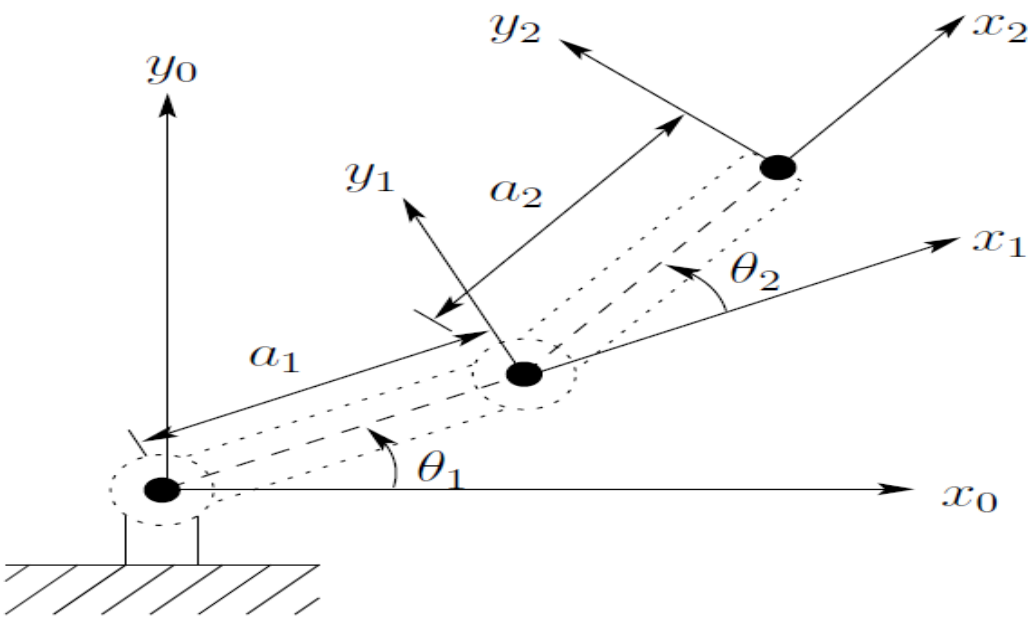


Figure 3: Two-link planar robot (Spong, 2004).

2.2.1 DH Parameter

Reference frame $o_0x_0y_0$ has been chosen. When the base frame has been chosen, the $o_1x_1y_1$ frame of link1 is fixed according to DH convention, where the origin o_1 is positioned at the intersection of z_1 and the page. The link2 frame $o_2x_2y_2$, where the end effector of 2R robot is located, is fixed by choosing the origin o_2. The table of D-H parameters of 2R is shown as follows:

Link	α_i	a_i	d_i	θ_i
1	0	a_1	0	θ_1
2	0	a_2	0	θ_2

Table 1: DH Parameter for two-link manipulator.

2.2.2 Transformation Matrix

Homogeneous transformation matrix of joint 1 and joint 2 is given by:

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To simplify the notation in the transformation matrix, the trigonometric function can be re-written as follows:

$s_1 = \sin(\theta_1)$ $\qquad$ $c_1 = \cos(\theta_1)$

$s_{12} = \sin(\theta_1 + \theta_2)$ $\qquad$ $c_{12} = \cos(\theta_1 + \theta_2)$

The transformation matrix T is given as follow:

$$T_1^0 = A_1$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the first three rows and columns represent the orientation of the $o_2x_2y_2$ w.r.t base frame and last columin of T_2^0 shows the position (x and y components) of the origin o_2 w.r.t the base.

2.2.3 Data Point Simulation

Once the kinematic equations have been solved, it is possible to create database consisting of positions in XY plane as a result of combinations of angles of orientation by using the software MATLAB. For the purpose of generating trajectory between two random points, training of the

working environment of the manipulator is needed. Training is done by using Neural Networks (NN), which are described in the later chapters. For both Trajectory Generation and Training, first thing that is needed is the uniformly distributed database within the working range of our machine.

Points	**X**	**Y**	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$
1	3.5	0	0	0
2	3.4631	0.3306	0	25.8593
...	...	...	...	...
729	1.1836	2.9214	90	307.9
...	...	...	...	...
1058	-1.3962	2.9141	135	314.3120
...	...	...	...	...
2423	1.4337	-2.9140	315	315.7472
...	...	...	...	...
2650	2.9446	-1.6240	345	327.4713
...	...	...	...	...
2736	2.9937	-1.2258	357	314.6301

Table 2: Database of positions and corresponding angles for two-link robot.

Table 2 demonstrates database of two-link robot where the random positions of the end effector is represented and their corresponding angles of rotation in 2-Dimensional space. For creating homogenous database of two link robot, following criteria have been used:

1- Link 1, a_1 = 2 cm

2- Link 2, a_2 = 1.5 cm

3- $\theta_{1,i} = \theta_{1,(i-1)} + 3$ s.t $\theta_{1i} \leq 360 - 3$

4- $\theta_{2,i} = \theta_{2,(i-1)} +$ rand·30 s.t $\theta_{2,i} \leq 360\text{-}30$ $\forall\theta_{1i}$ $i \in \{1,2,3, \dots\}$

The simulation has been carried out in MATLAB using forward kinematics concept, where possible points of the two-link robot can be seen in 2D space.

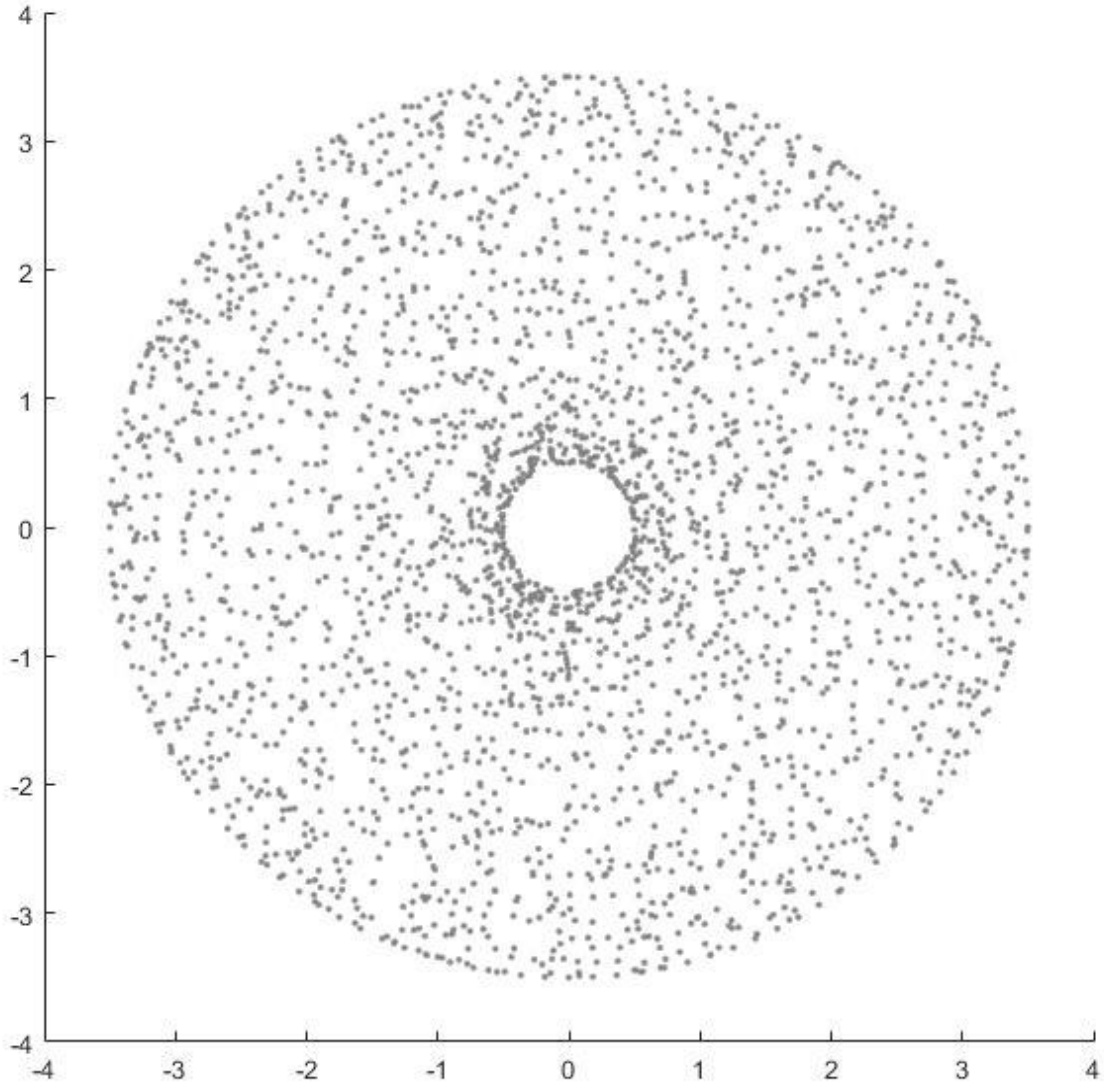


Figure 4: Cloud of 2736 random positions of a two-link robot according to the table.

These training points shown above are useful as a starting phase for artificial neural network algorithms discussed in the upcoming chapters.

2.3 Puma 560

Different types of robots are used in industries. PUMA (Programmable Universal Manipulation Arm) 560 is one of the industrial robots that has been used in many industries. This robot was initially developed for General Motors. It is RRRRRR with 6 links and 6 degrees of freedom. The figure can be shown below:

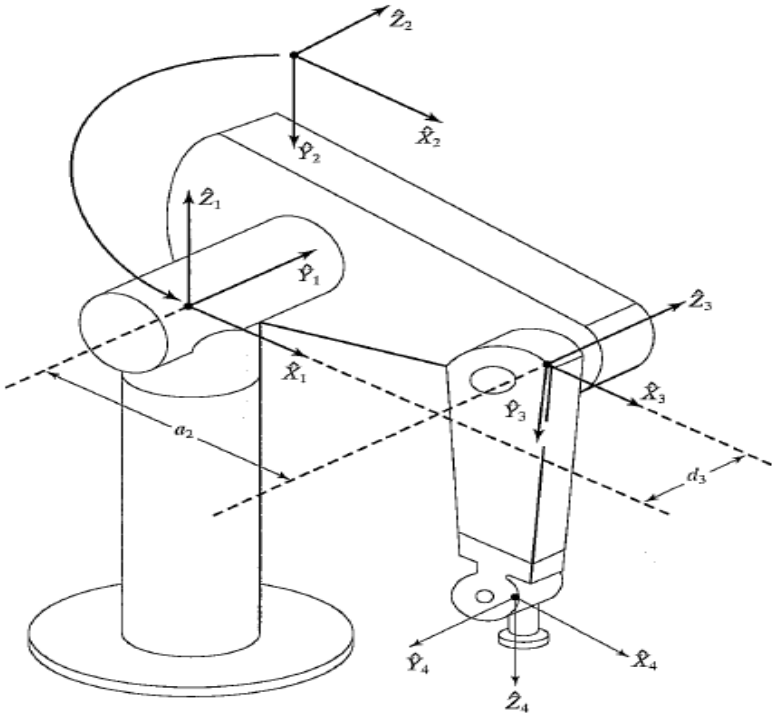


Figure 5: Frame assignment for PUMA 560 (Craig, 2005).

From this Figure 5, it can be seen that link-frame parameters corresponding to all joint angles are equal to zero. The joint axes 4 5 6 intersect at a common point and mutually orthogonal.

2.3.1 DH Parameter

One of the main concerns for the robotic system is to find the position of the end effector of a robotic manipulator most accurately. For this case, the Denavit-Hartenberg (DH) parameter should be adopted to find end effector position and angles. DH convention is the most common approach of forward kinematics. After the frame assignment, it is possible to compute the DH parameters of PUMA 560.

Link i	α_i	θ_i	a_i	d_i
1	90°	θ_1	0	d_1
2	0°	θ_2	a_2	0
3	90°	$\theta_3 + 90$	0	0
4	-90°	θ_4	0	d_4
5	90°	θ_5	0	0
6	0°	θ_6	0	0

Table 3: DH parameter for PUMA 560.

To better understand the forearm position of the robot:

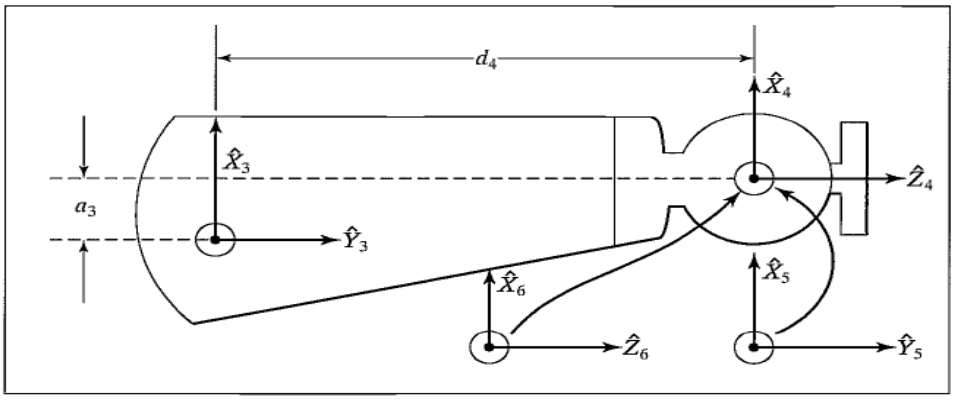


Figure 6: Frame assignments for the forearm of PUMA560 (Craig, 2005).

2.3.2 Transformation Matrix

After the DH convention of PUMA560, it is possible to derive a 4×4 transformation matrix where the first 3 rows and columns show the rotation and last column shows the translation.

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_6^5 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By the convention of Matrix multiplication property, it is possible to obtain T_0^6 as:

$$T_0^6 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5$$

2.3.3 Data Point Simulation

After solving DH parameter and forward kinematics, one can simulate it on MATLAB to derive the position and corresponding angles of rotation of end effector. By using MATLAB, a database of points can be generated that the robot can move in 3D space. Each point represents a position in X Y Z and corresponding angles of rotation.

It is possible to create a table with data points of Puma robot as done in section 2.2.3. Uniformly distributed data points of Puma 560 can be achieved by using following criteria:

1- Parameter a_2 = 2 cm
2- Parameter d_1 = 2 cm
3- Parameter d_4 = 1.5 cm
4- $\theta_{1,i} = \theta_{1,(i-1)} + 3$ s.t $\theta_{1i} \leq 320$
5- $\theta_{2,i} = \theta_{2,(i-1)} + 3$ s.t $\theta_{2,i} \leq 250$
6- $\theta_{3,i} = \theta_{3,(i-1)} + \text{rand} \cdot 90$ s.t $\theta_{3,i} \leq 270$ $\forall \theta_{1i} \quad i \in \{1,2,3,\dots\}$
7- XY positions generated by above-mentioned criteria of θ values must be apart from each other by a distance of more than 0.15cm.

Table 4 shows an excerpt from the database of Puma 560 where positions of end effector are labeled as X Y Z and angles of rotation $\theta_1, \dots, \theta_6$. It can be noted that the values of θ_4, θ_5 *and* θ_6 are zero because the angles belong to the wrist of robot and by changing these angles, it does not have any effect of the positions of end effector, only in case $d_6 \neq 0$.

Points	X	Y	Z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	3.34	-5.06	2.67	0	0	0	0	0	0
2	2.10	-1.83	0.50	0	0	73.43	0	0	0
...	...	...	...	...	...	...	...	...	...
650	-2.96	0.86	3.63	6	9	225.58	0	0	0
...	...	...	...	...	...	...	...	...	...
999	-0.46	0.21	-0.55	9	42	264.91	0	0	0
…	...	...	...	...	...	...	...	...	...
1484	-0.22	0.14	-0.47	12	249	70.23	0	0	0
...	...	...	...	...	...	...	...	...	...
1754	2.11	-1.81	1.04	15	249	249.70	0	0	0
...	...	...	...	...	...	...	...	...	...
2909	0.53	-3.44	1.83	30	0	62.25	0	0	0
...	...	...	...	...	...	...	...	...	...
5879	-1.17	-0.37	-0.14	60	36	202.19	0	0	0
...	...	...	...	...	...	...	...	...	...
8550	-1.86	-1.33	1.41	120	180	261.75	0	0	0
...	...	...	...	...	...	...	...	...	...
11041	-1.52	-2.04	0.71	180	132	73.68	0	0	0
15257	1.73	1.45	2.95	318	222	233.61	0	0	0

Table 4: Database of positions and corresponding angles for Puma 560.

According to the table, it is optimal to plot the database of Puma 560 in 3-dimensional space where all possible points are shown that robot can reach. The simulation is done in MATLAB by solving forward kinematic equations. Figure 7 shows the uniformly distributed points of Puma robot.

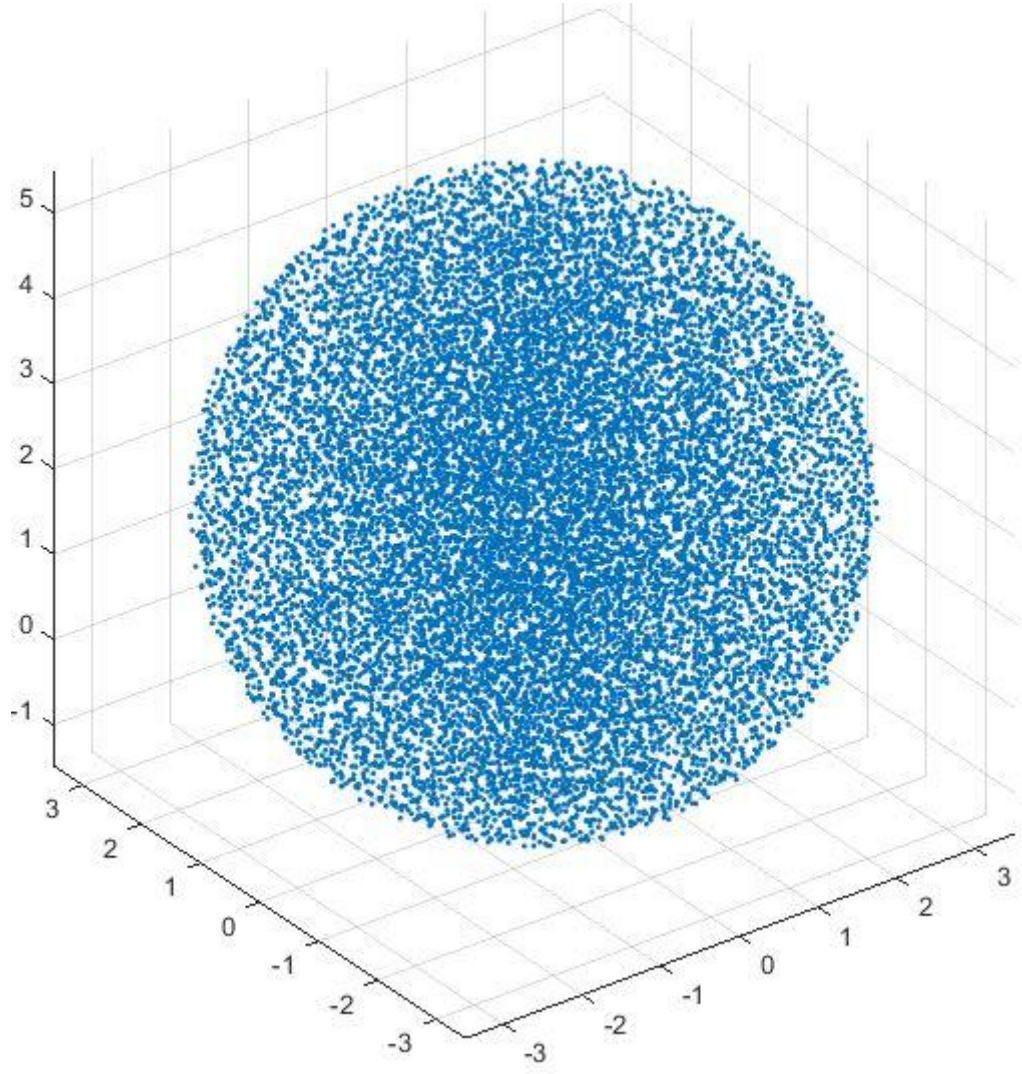


Figure 7: Data Points in 3D of PUMA 560.

In total, the plot shown above has 15257 random points. It can be called as training points, which will be used as starting point of artificial neural networks such as GNG, GWR and STRAGEN in next chapters, to train them. After training, it can be possible to find the trajectory of the robot described in Chapter 7.

Chapter 3

Neural Network Models

In order to proceed with the purpose of work to find continuous trajectory between two end points, first database needs to be trained, for which some artificial neural network models should be discussed. The operation and properties of these models can be used as a foundation and inspiration for this thesis work. Supervised learning can be understood by analyzing the training data to produce an inferred funtion, which can be used for mapping of new patterns.

3.1 Self-Organizing Map (SOM)

The Self Organizing Map (SOM) algorithm, defined by Finnish professor T. Kohonen, is a type of unsupervised Artifical Neural Network (ANN). It uses a competitive learning procedure for discretized representation of high dimension information into low dimensional, while simultaneously preserving similarity relations between the presented data items.

Kohonen (1987) points out one of the common properties of the brain, which was ignored by the learning machine, is the order that appears in the processing units, called neurons. This order is necessary for the correct topological representation of the input data and is obtained by a relatively simple algorithm.

> *''Although a part of such ordering in the brain were determined genetically, it will be intriguing to learn that an almost optimal spatial order, in relation to signal statistics, can completely be determined in simple self-organizing process under the control of received information*" stated by KOHONEN in 1987.

The SOM consists of a regular, usually two-dimensional grid, onto which a distribution of input items is projected nonlinearly. The mapping tends to preserve the topological-metric relations between the input items. Matching procedure is used for the projection. Every grid unit, a generalized model is thought to be associated. For every input pattern, the closet model in a metric is identified. The collection of models is enhanced to approximate all inputs.

The focal property of SOM is that it frames a nonlinear projection of a high-dimensional data manifold on a regular, low-dimensional matrix. In the display, the clustering of the data spaces

as well as the metric-topology relations of data items are plainly visible. If the data items are vectors, the components of which are variables with a definite meaning such as descriptors of statistical data, or measurements that describe a process, the SOM matrix can be utilized as a preparation on which every one of the factors can be shown independently using color coding. The sort of consolidated presentation has been discovered exceptionally helpful for the comprehension of the common conditions between the factors, and in addition to the structure of the database.

Most of the Artificial Neural Network basically follow functions in two modes:

- Training: to build up the map from existing input samples.
- Mapping: classify a new input vector.

The self-organized map defines a mapping from a higher-dimensional input space to a lower-dimensional map space. At the point when it's being trained, the map can arrange a vector from input space via looking through the nodes with nearest (short separation metric) to the input space vector.

3.1.1 Algorithm

The Algorithm below as described by Benante (2008, Page 53):

1. Initialize the network with random weight vector, such that w_i is the vector of weights of connections between the input and node i.
2. Initialize the neighborhood function, $N(i)$, for the largest neighborhood.
3. Display the input pattern $\xi = [\xi_1 \ \xi_2 \ ... \ \xi_D]^T \in R^D$, where ξ_j is the input j in node i.
4. Calculate the Euclidean Distance $Dist(\xi, w_{s1})$ between the input ξ and the weight vector at each node, given by the equation:

$$Dist(\xi, w_i) = \|w_i - \xi\|$$

5. Let s_1 is the node that has the shortest distance $Dist(\xi, w_{s1})$, i.e., winning node.
6. Update the weight vector w_i, for each node i, belonging to the neighborhood $N(s_1)$, according to the equation:

$$w_i = w_i + \varepsilon \cdot (\xi - w_i)$$

7. Decrease the neighborhood of $N(s_1)$.
8. Repeating the Neighborhood function until reaching an iteration limit.

It ought to be recalled that the neighborhood function $N(s_1)$ contains the neighbor nodes of the winning node, which will likewise have their weights updated (Gaussian) in the direction of the current pattern of input. The neighborhood function is responsible for the ordering of the nodes that tend to respond for the similar input pattern or for comparable input patterns. It is necessary that this function tends to zero to ensure the convergence of the method.

3.1.2 Example

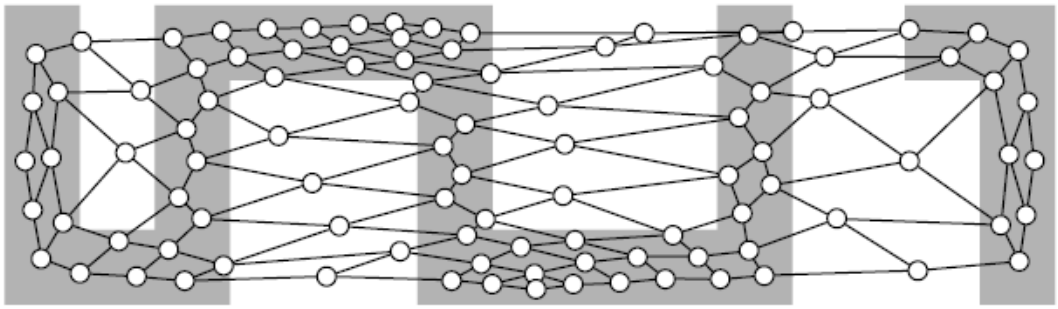


Figure 8: Kohonen map after 10 000 signals for clustered data distribution (Fritzke, 1997).

Figure 8 demonstrates a case of a self-organizing map of Kohonen with an initial two-dimensional, rectangular structure finds a configuration that distorts its initial grid to a new arrangement that minimizes the error of response at every unit. The rectangular map consists of 100 nodes. It very well may be noted that, because of characteristics of the data mass (discontinuous and convex), the Kohonen map can not disperse its units legitimately.

3.2 Topology-Representing Networks (TRN)

The previous model used for the low dimensional dataset. It emerges conflicts dealing with high-dimensional informational collection. In practice, high-dimensional data set is used to be analyzed. This model proposed by Martinetz (1991) is based on a combination of two techniques:

- Neural gas vector quantization.
- Competitive Hebbian learning rule.

The point of the topology display is to give a strong representation of data set, where the hidden information structure is protected. For a given data structure, where a number of points (weights) are the predefined parameters. The idea is to distribute an explicit number of nodes in a region of the input data, according to some likelihood functions, following Neural Gas algorithm and Competitive Hebbian Learning. This topology may have different dimensions for different areas of the input. At the beginning, it chooses randomly a pattern from the given data set and condenses all weights nearer to this pattern. After this, two weights nearest to the selected input patterns, which is random, will be connected. The edges which surpass a predefined threshold will be expelled toward the end. The process is continued until a limiting criterion is introduced.

Dynamic Topology Representing Networks (DTRN), introduced by J. Si, S. Lin, the topology graph incrementally develops by adding and eliminating edges and vertices. At the initial step, begins with only one node. The algorithm performs a vigilance test in each iteration. In the event that the nearest (winner) node to the randomly chosen input information pattern falls flat the test, then a new node will be generated and it will connect to the winner. In another case, if the first winner and the second winner are not connected, the algorithm creates an edge between them. This algorithm is same as TRN when the connections those exceeds a predefined limit, will be removed.

3.2.1 Neural Gas (NG)

Neural gas is an artificial neural network, followed by Self-organizing map. The algorithm essentially based on finding the optimum data representation on featured vectors. NG refers only to the isolated nodes, which will be distributed in a R^D space, according to the probability function $P(\xi)$. The principal of NG is that, for each input signal ξ, all the nodes must adapt according to classified order, i.e., the nearest first, second etc, with relation to the input signal ξ. Over time the number of nodes will be adapted until the first winner is matched.

3.2.1.1 Algorithm

Algorithm underneath as described by Benante (2008, Page 58):

1. Initialize the set A with n units of i, and at random positions $w_i \in R^D, i = 1,2, \dots, n$, random such that $A = \{i_1, i_2, i_3, \dots, i_n\}$. Initialize the time parameter $t = 0$.
2. Generate at random an input signal ξ, according to $P(\xi)$.
3. Order the elements of A, as well their indices $(i_0, i_1, i_2, \dots, i_{n-1})$, according to the distance between w_k and ξ, such that:

$$\|w_k - \xi\| < \|w_{k+1} - \xi\|$$

4. Set the featured vector (weight) according to the equation:

$$\Delta w_i = \varepsilon(t) \cdot h_y(k) \cdot (\xi - \mathrm{w_i})$$

With the dependence of time: $y(t) = y_i(y_f/y_i)^{t/t_{max}}$, $\varepsilon(t) = \varepsilon_i(\varepsilon_f/\varepsilon_i)^{t/t_{max}}$, and $h_y(k) = \exp(-k/y)$, where $y(t)$ and $\varepsilon(t)$ are decay functions, for which the values of initial (y_i, ε_i) and final (y_f, ε_f) must be chosen, t_{max} is the limit of iterations and k is the index vector of positions of i_k in the proximity ranking of the nodes and $h_y(k)$ is a ranking proximity function.

5. Increment the time parameter $t = t + 1$.
6. If $t < t_{max}$, continue from the step 2.

3.2.1.2 Example

The simulation results of NG algorithm are shown below. Following Martinetz (1993), this simulation was done by choosing the following parameters:

$y_i = 10$, $y_f = 0.01$, $\varepsilon_i = 0.5$, $\varepsilon_f = 0.005$, $t_{max} = 40\,000$

The simulation results are obtained for this algorithm by applying 40 000 input signals. It is noted that no topological information involved in neural gas, i.e., there are no neighborhood connections. The distribution of nodes reflects only the density function probability of input space.

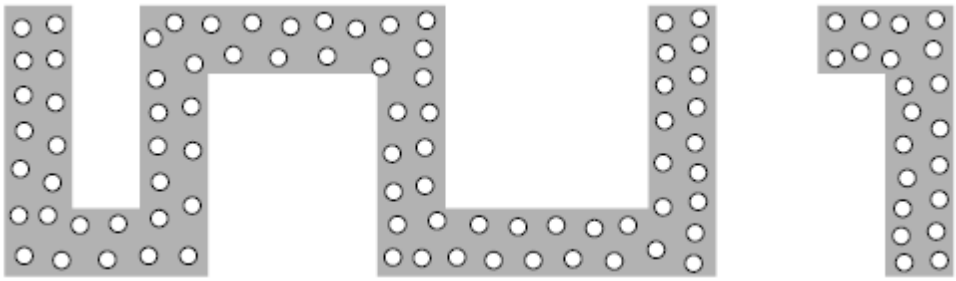


Figure 9: Neural gas after 40 000 input signals for the clustered data distribution (Fritzke, 1997).

The Neural Gas model does not create a connection between the signals. Therefore, we can't implement this topology in many cases. To overcome this case, the complete TRN algorithm combines NG with CHL, which creates the connections.

3.2.2 Competitive Hebbian Learning (CHL)

CHL is proposed by Martinetz (1993) topology considers a number of nodes in the R^D space and progressively insert topological connections among them by evaluating input signals drawn from a data distribution $P(\xi)$. The quotation of the method is given below:

''For each input signal ξ connect the two closest centers (measured by Euclidean distance) by an edge.'' (CHL)

The resulting graph generates a subgraph of the Delaunay Triangulation. This corresponds to the arrangement of centers. The induced Delaunay triangulation, a subgraph of Delaunay triangulation, the algorithm is limited to connect nodes only if at least part of their Voronoi regions are in regions with probability P(ξ)> 0.

The NG method does not use a specific topology. A 'network' simply consists of a number of disconnected centers in R^D. Two centers are only connected if the common border of Voronoi polygons lies any event partially in a region where P(ξ)> 0.

Figure 10 presents (a) Delaunay Triangulation connects points having neighboring Voronoi polygons (thin lines). Sub-figure (b) shows the induced Delaunay triangulation (thick lines) is obtained by masking the original Delaunay triangulation with a data distribution P(ξ) (shaded).

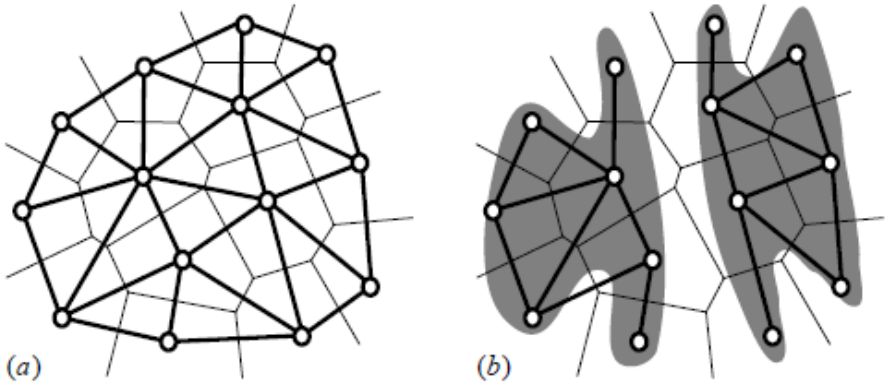


Figure 10: (a) The Delaunay Triangulation (b) The induced Delaunay triangulation (Martinetz and Schulten, 1994).

3.2.2.1 Algorithm

Below the algorithm is described by Benante (2008, Page 60):

1. Initialize the set A with n units of i at random positions $w_i \in R^D, i = 1,2, \dots, n$, such that $A = \{i_1, i_2, i_3, \dots, i_n\}$. Initialize the connection set C, $C \subset A \times A$, with the empty set $C = \{\ \}$, i.e., no connections.
2. Generate randomly the input signal ξ, according to $P(\xi)$.
3. Determine the nodes s_1, s_2 such that:

$$\|w_{s1} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A$$

$$\|w_{s2} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A - \{s_1\}$$

4. If it does not exist already, insert a connection between s_1 and s_2 to C:

$$C = C \cup \{(s_1, s_2)\}$$

5. Continue from Step 2 until a maximum number of input signals is reached.

3.2.2.2 Example

The edges are developed by the input data submanifold lying on the centers or in its vicinity. The others are ineffective for the purpose of this algorithm, is called dead units. In Figure 11, we can see that stating the algorithm with a number of units at random positions, edges are inserted between best-matching and second-best matching unit for each input signal. The positions of the units for circular data distribution remain unchanged, which possibly leads to dead units not affecting the network. While for clustered data distribution, a considerable

number of dead units with any connection can be visualized. the random initialization of center positions leads to unevenly well-represented regions of the input data submanifold.

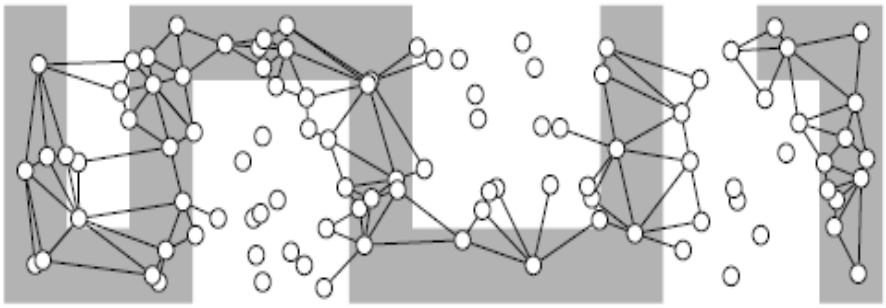


Figure 11: Competitive Hebbian learning for clustered data distribution (Fritzke, 1997).

3.2.3 Combination of NG and CHL

An viable method for learning topology can be achieved if, for a given mass of input data, first execute NG algorithm to position the nodes within the areas of interest in the probability distribution and then to apply the CHL algorithm to create the topology as a combination of these two models, which includes the connections between the nodes created by NG algorithm.

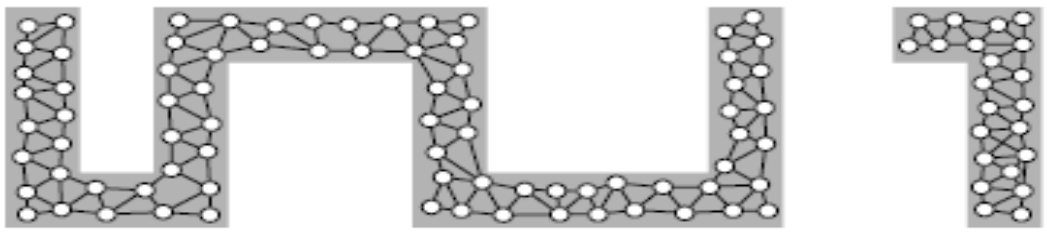


Figure 12: Example of TRN after 40 000 iterations. Parameters of the simulation is the same as for NG and CHL (Fritzke, 1997).

In relation to the previous techniques, the quality of the representation, which collectively covers all the points in the shaded area maintains the representation of the irregularity of the input function.

The last structure appeared in figure imitates exceptionally well the representation of hidden distribution. Since a parameter decay is required for NG algorithm, the number of adaption steps have to be defined in advance.

3.3 Growing Cell Structure (GCS)

The model of GCS has been proposed by Fritzke (1991) to overcome some difficulties arising in the Self-organizing map. The aim of this model is to enable the artificial neural network itself to find a suitable neural structure for a huge amount of given data. The GCS model advances in the sense that there is no longer pre-defined structure, but only a few nodes, in addition to a mechanism to create or remove nodes. With this strategy, this network can grow changing both its shape, position and the number of elements that compose it.

The goal of growing cell structure is the creation of topology-preserving mapping from the input space R^D into a topological structure of A of equal or lower dimensionality k. The meaning of Topology-preserving can be understood as follows:

- Input vectors which are close in R^D should be mapped onto neighboring nodes in A.
- Neighboring nodes in A should have similar input vectors mapped onto them.

Most cases, the above property is not obtained because in general, a reversible mapping from high dimensional space onto lower dimensional does not exist. A growing cell structure generates k dimensional topological structure can be visible as a projection onto a nonlinear, discretely sampled submanifold.

The objective of GCS is to generate a topological map A representative of input data M, with smaller dimension $D_A < D_m$ which meets the following requirements:

- Signals of similar entities in M are mapped onto topologically close elements of A.
- Topologically close elements in A have similar input signals.
- Regions of M with high probability density are represented in A by a high density of elements.

The initial topology of this model is a set A of $k + 1$ nodes (vertices) of a graph simplex, where k is the dimension of the graph. During network execution, nodes will be added and those considered unnecessary will be removed, always maintaining the characteristic k-simplex network. The model preserves the topology of the input data if the dimension of the graph k is the same as the input space (Marsland, 2002).

Another important notation in this model is the Voronoi Region. A given set of nodes with their reference vector defines a distinct barrier of the input space is called as Voronoi Region. Voronoi filed $V(c)$ can defined as the receptive field of each node c and its reference vector w_c.

$$V(c) = \{p \in R^D | \ (\|p - w_c\| < \|p - w_d\|) \ \forall d \in A, d \neq c\}$$

The Voronoi field of c consists of those points in R^D for which w_c is the nearest of all currently existing reference vectors.

3.3.1 Algorithm

The model adapts the reference vectors and inserts the node that connects in between. Two different types of scenario in the algorithm described by Benante (2008, Page 65) that are:

- Learning (adaption) rates are constant over time. Let ε_b for the winning node s_1 and ε_n for the direct neighbors of s_1.
- Only the winning nodes and their direct neighbors are adapted.

To perform a number of adaption steps for an unused unit is hypothesis proportionate to embed another node and interpolate its reference vector from neighbors. Executing stepwise-adaption, insertion can be quicker than situating. The parameter can be chosen small and constant if adaption isn't required for moving units in the extensive separation.

1. Choose an input signal ξ according to probability distribution function $P(\xi)$.
2. Locate the winning unit s_1 , the shortest Euclidean distance according to:

$$\|w_{s1} - \xi\| = \min_{i \in A} \|w_i - \xi\|$$

3. Update the weights of s_1 and its direct neighborhood:

$$\Delta w_{s1} = \varepsilon_b(\xi - w_{s1})$$

$$\Delta w_i \ = \varepsilon_n(\xi - w_i) \ \forall i \ \in N(s_1)$$

4. Add a unit to the s_1 signal counter such that:

$$h_{s1} = h_{s1} + 1$$

5. Decrease all the signal counter by a fraction such that:

$$\Delta h_i = -\alpha h_i$$

- Inserting node and its connection at every λ step:

  The capacity of GCS is to embed a lot of nodes $i \in A$ to such an extent that the vector w_i is the agent of probability function $P(\boldsymbol{\xi})$ with $\xi \in M$. This generally happen when every node i has the similar probability to be the champ of s_1 for a given information $\boldsymbol{\xi}$.
  The indicator of Relative frequency is given by:

$$\breve{h}_\iota = \frac{h_i}{\sum_{\forall j \in A} h_j}$$

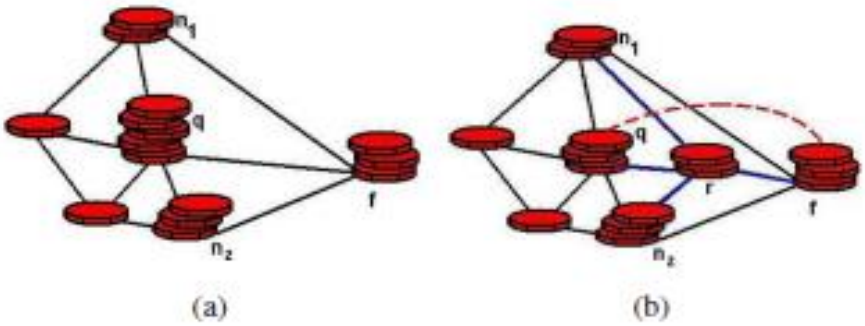


Figure 13: GCS Network. (a) Before insertion (b) After insertion of node r (Benante, 2008).

The GCS algorithm consists of indicator frequency and inserts a new node r in the center of Voronoi Region which is ineffectively represented. This should be possible by picking a halfway point between the node with most relative frequency q and its most distant neighbor f.

$$w_r = (w_q + w_f)/2$$

The new formation of vector connection should maintain the k-simple graphical characteristics, maintaining the hyper tetrahedrons. This can be done according to the following steps:

- The old connections between q and r are undone.
- The new node r is connected to q, f and neighbors that are common to both (in this case n_1 *and* n_2).

After the insertion of new nodes, the h_i input signal counter of the neighboring nodes i should be redistributed as follows:

$$\Delta h_i = \frac{\left|F_i^{(new)}\right| - \left|F_i^{(old)}\right|}{\left|F_i^{(old)}\right|} \cdot h_i$$

Where $|F_i|$ is the D_m-dimesional volume of the Voronoi Region F_i. And the new node r has its counter initially set to

$$h_r = -\sum_{i \in N(r)} \Delta h_i$$

- Removal of nodes:

The GCS algorithm includes a criterion for the removal of so-called "superfluous" nodes. A node will be regarded as superfluous if it is located in a region of low probability density. Since, probability density is generally very difficult to obtain, the GCS estimates the $\tilde{p}$ value by the relative frequency of signals from a node in relation to the size of its Voronoi map. The cut-off threshold k to be chosen depends on the problem, since the densities vary for each case, making it very difficult to define it as an absolute value. An approximation based on a normalized probability density in relation to the total volume of all hypercubes minimizes this difficulty, making $k = 0.09$ (Fritzke, 1994) an appropriate standard value for most cases.

3.3.2 Example

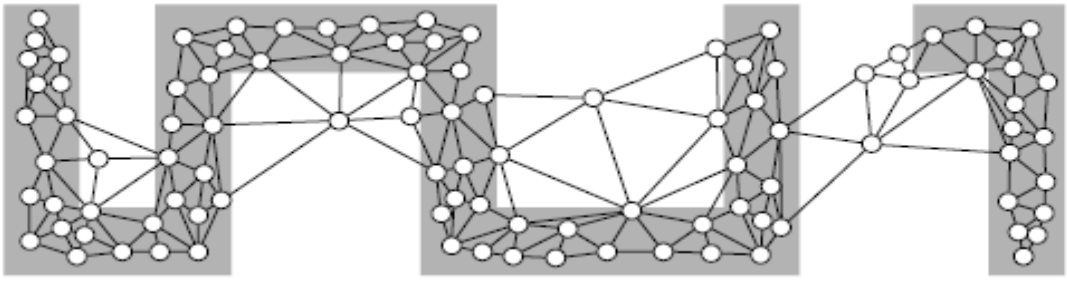


Figure 14: Growing neural cell structure after 20 000 signals. The parameters of the simulation: $\alpha = 0.2, \beta = 0.005, \lambda = 200, \varepsilon_b = 0.02, \varepsilon_n = 0.006$. (Fritzke, 1997).

The above figure shows the result of applying a variation of the proposed GCS algorithm for a convex and discontinuous mass of data with areas that have zero probability.

Chapter 4

Growing Neural Gas (GNG)

The model GNG, introduced by Fritzke (1997), it tends to be viewed as an incremental variation of the TRN model, is derived from GCS. The motivation behind this model is to generate a graph that reflects the topological input data manifold. This graph has a dimensionality which differs with the dimensionality of the input pattern.

Growing Neural Gas model is an unsupervised incremental clustering algorithm. The connections between the neighbors can be used as information about an interpolation scheme where function values for arbitrary positions in R^D. This model shares some properties with Topology Representing Network (TRN). The node addition procedure in GNG, has some advantages in relation to the fixed TRN. Furthermore, GNG does not have the topological constraints of k-simplex graph as of GCS model. At first, arbitrary edges are allowed. Its topology based on how many different dimensions are required to represent the input data. There are still some differences that can be seen by the algorithm presented below.

4.1 Algorithm

Consider networks consisting of (Benante, 2008, Page 72):

- a set A of nodes (or units). Each node $i \in A$ has a reference vector $w_i \in R^D$, which can be understood as its position in input space.
- a set of C connections (or edges) between the pair of nodes. These connections have not weights. Their sole perform is to define the topological structure.

The principle thought is to include new nodes successively, starting from a network through the assessment of local statistical measures during the presentation of the data and it's in the previous adaption step (Fritzke, 1997).

1. Initialize the set A with two units $n_1 \; and \; n_2$ in positions $w_{n1} \; and \; w_{n2}$ belonging to R^D from randomly selected input P (ξ).

$$A = \{n_1, n_2\}$$

Initialize the set of C connection with a connection between n_1 & n_2 and set the age of connection to zero.

$$C = \{c_{n1,n2}\}$$

$$age_{(n_1,n_2)} = 0$$

2. Generate an input signal ξ, according to P (ξ).

3. For every node i in the network, determine nodes $s_1, s_2 \in A$ such that:

$$\|w_{s1} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A$$

$$\|w_{s2} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A - \{s_1\}$$

   Where w_i is the weight vector of node i.

4. If a connection does not exist already, then insert the connection between s_1, s_2 to C:

$$C = C \cup \{C_{s1,s2}\}$$

   In any case, set the age connection to be zero (refresh the age):

$$age_{(s1,s2)} = 0$$

5. Add the squared distance between the input signal and the nearest unit in input space to a local error variable.

$$\Delta E_{s1} = \|w_{s1} - \xi\|^2$$

6. Move s_1 and its direct topological neighbors towards ξ by fractions ε_b and ε_n, respectively of the total distance.

$$\Delta w_{s1} = \varepsilon_b(\xi - w_{s1})$$

$$\Delta w_i = \varepsilon_n(\xi - w_i) \ \forall i \in N(s_1)$$

   Where $N(s_1)$ is the set of direct topological neighborhood of s_1.

7. Increase the age of all connections emanating from s_1.

$$age_{(s_1,i)} = age_{(s_1,i)} + 1 \quad \forall i \in N(s_1)$$

8. Remove the edges with an age larger than a_{max}. If this results in units having no edges emanating, remove them as well.
9. If the number of input signals is an integer multiple of parameter λ, then insert new node as follow:
    - Determine the unit q as having the largest accumulated error and determine f, a direct neighbor farthest from q.
    - Interpolate a new unit r between q and f: $w_r = (w_q + w_f)/2$.
    - Insert connections between r and q and between r and f. Also, remove the connections between q and f.
    - Decrease the error variable of q and f.
    - Interpolate the error variable of r between q and f.
10. Decrease the error of all units:

$$\Delta E_i = -\beta E_i \quad \forall i \in A$$

11. Continue from step 2 until a stop criterion (maximum iterations) has been reached (for example, the size of the network, or some measurement of performance).

4.2 Description of this method

The method described above has an adaption phase (Step 6) that moves all the nodes towards those areas $P(\xi) > 0$. In Step 6, adds a new connection between the nearest and the second-nearest unit, in relation to an input signal ξ. This generates a single connection in the induced Delaunay triangulation. The removal of nodes described in Step 8, is needed to eliminate connections that are no longer part of Delaunay Triangulation. This can be performed in conjunction with Step 7 that ages all the connections and with Step 4 that resetting age of those which already exist between nearest and second-nearest units. Removing and inserting connections in this model endeavors to develop and keep up the actuated Delaunay triangulation which is a gradually moving target because of the adaption of reference vectors. Step 5 accumulates the square of the distance between the signal ξ and the winning node, in order to identify nodes that are arranged in areas of the input space where the mapping is poorly represented.

According to the algorithm described in (Section 4.1), data points of two-link planar robot has been trained (Figure 15). The left figure initializes the algorithm with two random nodes and edge connection between them. At every λ iterations, this algorithm continues to add nodes until stop criterion (t_{max}= 40000) has been reached.

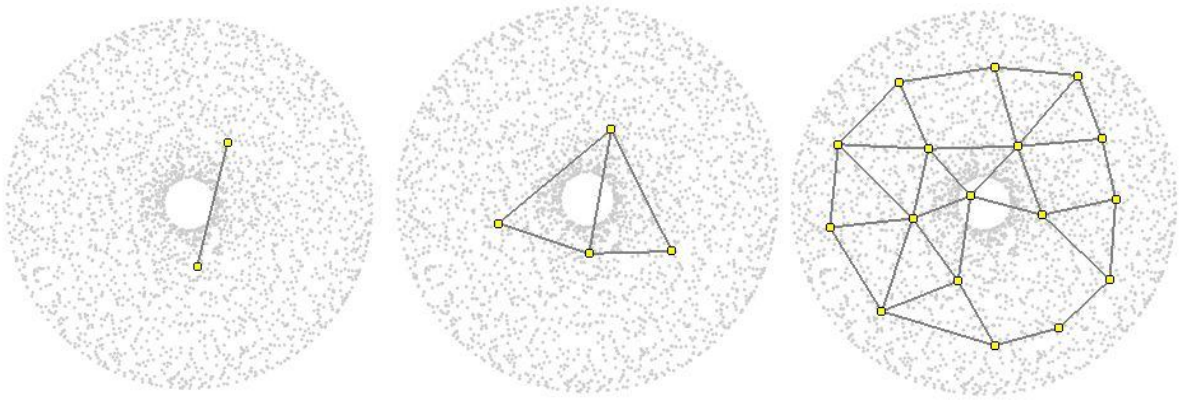


Figure 15: GNG algorithm execution. (Left) 1-199 iteration, (Middle) 400 Iterations, (Right) 1500 Iterations.

4.3 Discussion

For a given distribution $P(\xi)$ of input signals, GNG model is capable of making explicit the important topological relations. This model differs from GCS in a sense that it adopts less rigid topological definition. Comparing the GNG model with NG or CHL, one could have an important advantage as it does not require a predefine size of the network. The GNG model is a type of dynamic topology network similar to GCS. All parameters are constant in time and the number of adaption steps need not to be defined as priori. The variation of the network performance regarding the variations of its parameter was evaluated and the GNG network presents clear advantages in comparison with the Multi-layer Perceptron (MLP) and GCS. Due to its fractal-like characteristics, the learning process can be interrupted by any time, which has a good representation of input signals, given the computation time.

Fritzke (1997) comments that this model is not suitable for the data visualization as the model is not intended to decrease the dimensionality of input data signals unless a low-dimensional input data signal is presented.

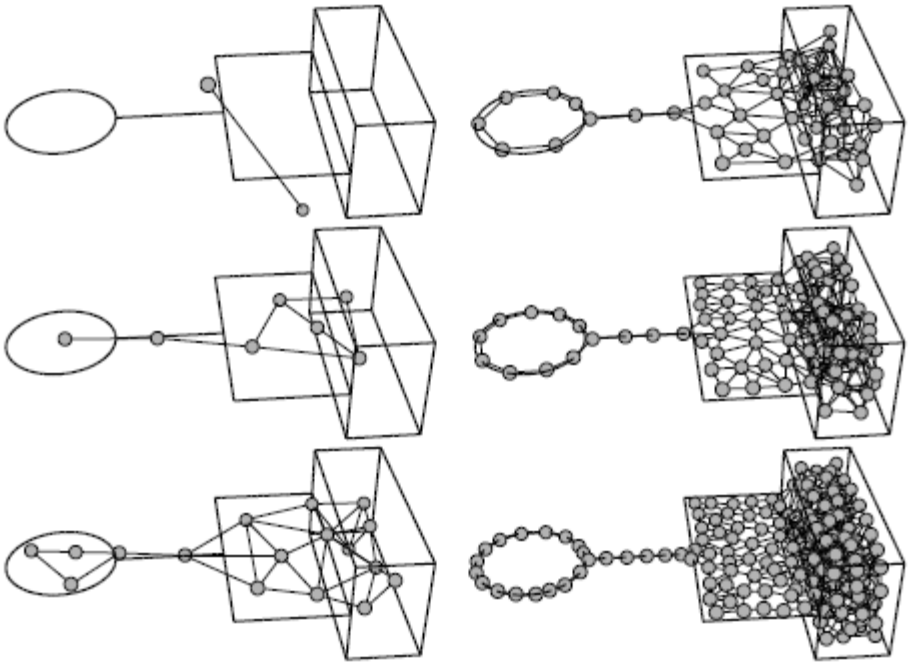


Figure 16: A geometric figure composed of shapes of different dimensions, being mapped by GNG. Simulation parameters are: $\alpha = 0.5$, $a_{max} = 88$, $\beta = 0.005$, $\lambda = 200$, $\varepsilon_b = 0.05$, $\varepsilon_n = 0.0006$ (Fritzke, 1997).

Figure 16 illustrated growing neural gas adapts to signal distribution which has different dimensionalities in a different area of input space. The initial network consists of two randomly place units and the network size of 7, 17, 50, 100 and 200 after 1000, 3000, 9600, 19 000 and 39 600 input signals respectively. The last one is not the least one since the process to be continued inconclusively.

This model applies very well for grouping and vector quantization. It is possible to combine with radial basis functions to achieve a supervised model incremental (Fritzke, 1997). The GNG model, however also depends on a parameter that controls the number of steps that controls the insertion and removal of nodes. The insertion of nodes in this model has been done as GCS, in order to minimize accumulated error at every step.

Chapter 5

Grow When Required (GWR)

The dynamic topologies described until now usually add a node (or add a layer of the node) in the position where accumulated error is high, or where there is need of topological adjustment. For each input signal an edge connection is created between the two nodes which best matched the unit and the second-best matching unit. These edge connections have an associated age. Initially, the ages are set to zero and increment at each time step. Nodes are added based on a predefined parameter λ iteration step. By this convention, the network always grows at the same rate according to the input data presented and continue to grow until the stopping criteria are fulfilled. The exception is that the edge that creates a connection between the best-matching unit and second-best unit, whose age is set to zero. Edges which exceeds the maximum age a_{max} $(constant\ parameter)$ are being removed. Node that has no edge connections is called dead node.

The Grow When Required (GWR), introduced by Marsland (2002), overcomes the limitation of using a criterion to insert nodes at a regular time interval. Rather than adding a node at every λ step, this algorithm can add a node at any time, depending on node insertion criterion. New nodes are added depending on input and current winning node, instead of adding them where the accumulated error is high or to divide the region that has the highest utilization rate.

A new node is added when the activity of the best-matching node (which is a function of the distance between the weights of the node and input) is not sufficiently high. The activity of the node calculated by taking the Euclidean distance between the weights for the node and the input.

5.1 Algorithm

Let A be the set of nodes of a self-organizing map and $C \subset A \times A$ the set of connections between the nodes in the map field. Let P(ξ) is the probability distribution function of the input data ξ. Finally, consider w_i be the weight vector of node i with dimension D (Benante, 2008, Page 77):

1. Initialize the set A with two units $n_1\ and\ n_2$ in positions $w_{n1}\ and\ w_{n2}$ belonging to R^D from randomly selected input P (ξ).

$$A = \{n_1, n_2\}$$

Define C, the connection set, to be the empty set

$$C = 0$$

2. Generate an input signal ξ, according to P (ξ).
3. For every node i in the network, determine nodes $s_1, s_2 \in A$ such that:

$$\|w_{s1} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A$$

$$\|w_{s2} - \xi\| \leq \|w_i - \xi\| \quad \forall i \in A - \{s_1\}$$

Where w_i is the weight vector of node i.

4. If there exists no connection between s_1, s_2 then create it.

$$C = C \cup \{C_{s1,s2}\}$$

Otherwise, set the age of the connection to zero.

$$age_{(s_1,s_2)} = 0$$

5. Calculate the activity of the best matching unit s_1

$$a(s_1) = \exp(-\|\xi - w_{s1}\|)$$

6. If $(a(s_1) < a_T)$ and $(\overline{h_{s1}} < h_T)$, then a new node must be added between the two best matching unit $s_1\ and\ s_2$, according to the step below where $a(s_1)$ is the activity threshold and $\overline{h_{s1}}$ is the firing counter:

(a) Add the new node r,

$$A = A \cup \{r\}$$

(b) Create the new weight vector for node r, setting the weights to be the average of the weights for the best matching node and the input vector:

$$w_r = (w_s + \xi)/2$$

(c) Insert the connections between (r, s_1) and between (r, s_2) and remove the connection between (s_1, s_2):

$$C = C \cup \{C_{r,s1}, C_{r,s2}\}$$

$$C = C - \{C_{s1,s2}\}$$

If a new node is not added, adapt the position of the winning node and its neighborhood i that is the node to which it is connected

$$\Delta w_{s1} = \varepsilon_b \times \overline{h_{s1}} \times (\boldsymbol{\xi} - w_{s1})$$

$$\Delta w_i = \varepsilon_n \times \bar{h}_\iota \times (\boldsymbol{\xi} - w_i) \qquad \forall i \in N(s_1)$$

Where $0 < \varepsilon_b < \varepsilon_n < 1$ and $\bar{h}_\iota$ is the value of firing counter for node i.

7. Increase the age of all connections emanating from s_1:

$$age_{(s_{1,i})} = age_{(s_{1,i})} + 1 \quad \forall i \in N(s_1)$$

Where $N(s_1)$ is the set of the direct topological neighborhood of s_1.

8. Reduce the firing counter $\overline{h_{s1}}$ from the winner node s_1 and $\bar{h}_\iota$ from its direct neighborhood.

$$\overline{h_{s1}}(t) = h^{(0)} - \frac{S(t)}{\alpha_b}(1 - \exp(-\alpha_b t/t_b))$$

$$\bar{h}_\iota(t) = h^{(0)} - \frac{S(t)}{\alpha_n}(1 - \exp(-\alpha_n t/t_n))$$

Where h_i is the size of firing variable for node i. h_0 is the initial strength, usually h_0 =1 S(t) is the stimulus force normally set to $S(t) = 1$. And the values of other constants controlling the behavior of the curve are $\alpha_b = 1.05,\ \alpha_n = 1.05,\ t_b = 3.33,\ t_n = 14.3$.

9. Check if there is any node to remove i.e. if there exist any nodes that no longer have any neighbors or edge that are older than the greatest allowed age a_{max}, in that case, remove them.

10. Continue from step 2 until some stop criterion has been reached (for network size, or some measure of performance).

The algorithm described above has been executed on the database of two-link robot (Figure 17). GWR can insert node when the activity threshold and firing counter both are less than a_T and h_T respectively. GWR is much faster than GNG in terms of inserting nodes. Figure 17 shows an example of GWR execution. After 50 iterations, it already added several nodes and edge connection between them. While in GNG, it add nodes at every 200 iteration. GWR does not need any time interval to add nodes as the network grows on demand.

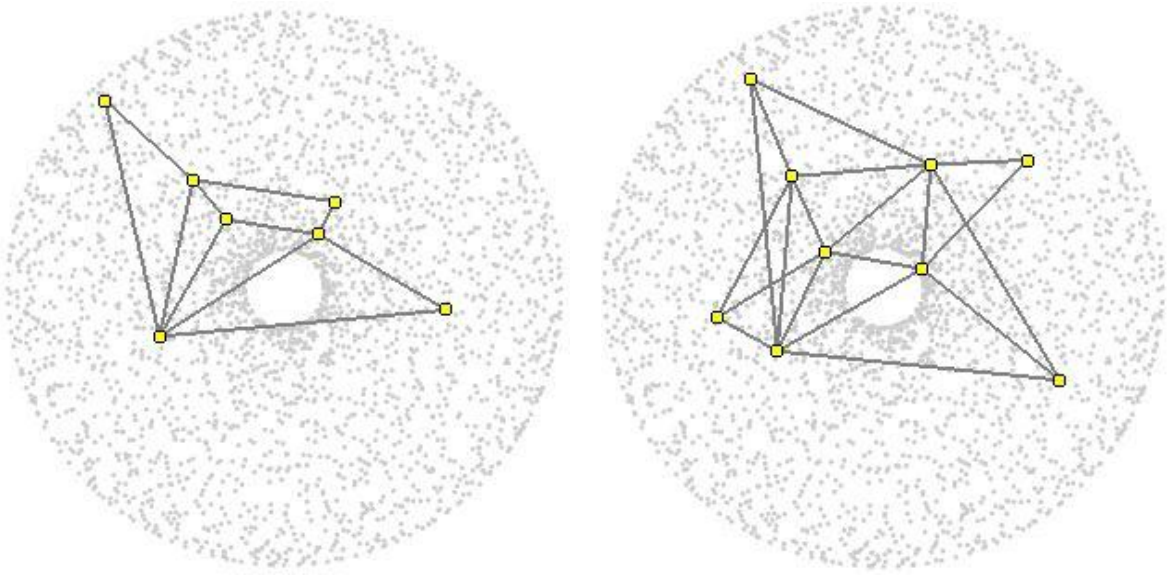


Figure 17: GWR algorithm execution. (Left) 50 iterations, (Right) 100 Iterations.

5.2 Example

This example shows the behavior of GWR network for a distributed database (Marsland, 2002). The samples in this dataset are drawn from the random unit square. The top left diagram of above figure shows areas of positive probability occurrence of an input pattern. The next diagram shows the structure of GWR network after 80 patterns presented to the network. Afterward, it can be seen that the patterns presented in the network are increased but nodes and edges remain almost constant. This figure also shows that the GWR network is a type of topology preserving. Where data is represented in 2D, the GWR forms a 2D representation of this data set.

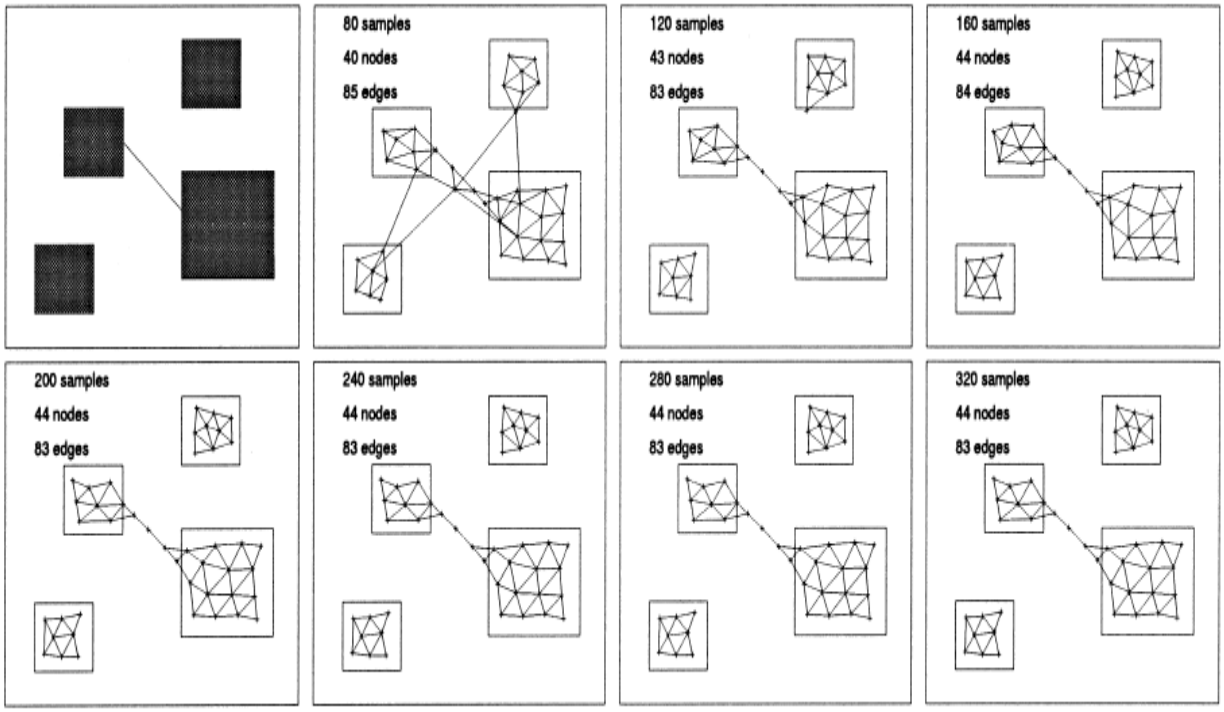


Figure 18: GWR learns a dataset consists of four square and one line. Parameters for this simulation are: $a_T = 0.99, \varepsilon_b = 0.05, \varepsilon_n = 0.006$ (Marsland, 2002).

5.3 Discussion

To measure the connections between nodes, Marsland (2002) used two cost functions, so that the combinations of these constraints could indicate a network with performance. The first measure of cost, E_1 penalizes the network that has neighbors located very distant from each other.

$$E_1 = \sum_i \sum_{j<i} C_{i,j} \cdot \|w_i - w_j\|^2$$

Where

$$C_{i,j} = \begin{cases} 1 & for\ (i,j)\ connected \\ 0, & otherwise \end{cases}$$

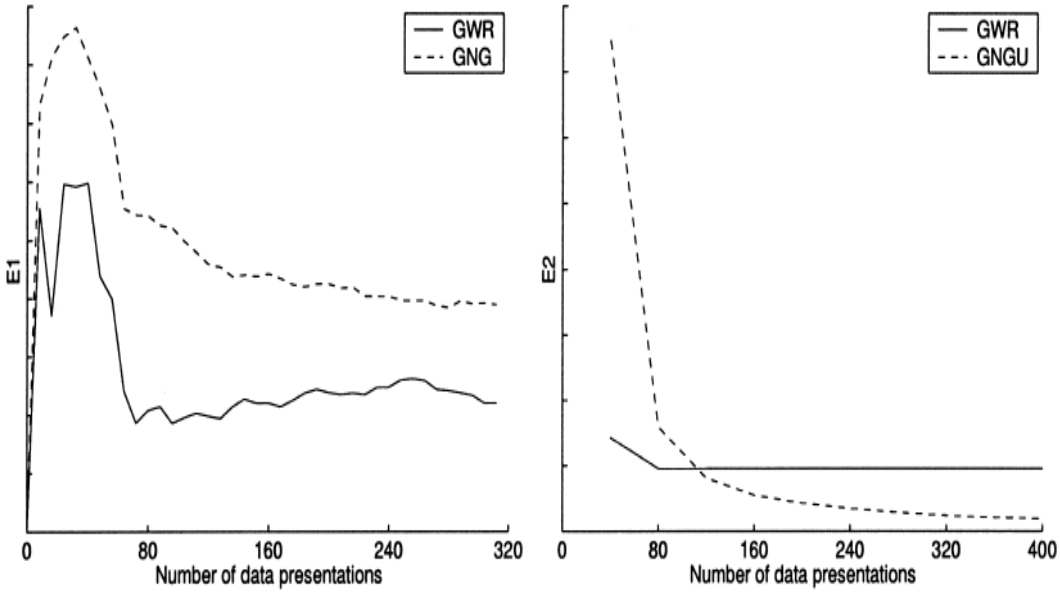


Figure 19: The left figure shows measurement error for E_1 GWR and GNG. The right figure shows measurement error E_2 for GWR and GNGU (Marsland, 2002).

It can be noted that GWR quickly decrease the E_1 error after 80 iterations. It means that this network keeps the neighboring nodes at a small distant already during the start of the execution process. In this example, topology function of GWR is $\phi(0) = 0.0023$.

The second cost function returns an evaluation of the ability of the network to minimize the distance between the input patterns the nodes. In this case $v \to \infty$, the cost function considers only the winning node.

$$E_2 = \sum_k \sum_i \|\xi - w\|^2 \cdot \frac{e^{-v \cdot \|\xi_k - w_i\|^2}}{\sum_j e^{-v \cdot \|\xi - w_j\|^2}}$$

In the right figure, the mapping shows the performance between GWR and GNGU (Growing Neural Gas with Utility). The mapping rapidly decreases for both networks. It reaches a constant level and this level is controlled by a threshold a_T, that allows insertion of new nodes. GWR network is fixed after 80 iterations as no new nodes are added to this network and nodes do not move. The GNGU network continues to decrease as it adds nodes at a regular time interval. The error value E_2 for GWR is lower than GNGU at a point where the both networks have the same number of nodes.

A more sharper response of measurement error E_1 and E_2 can be obtained when there is a sudden change in the probability distribution (or input map) that the network must learn.

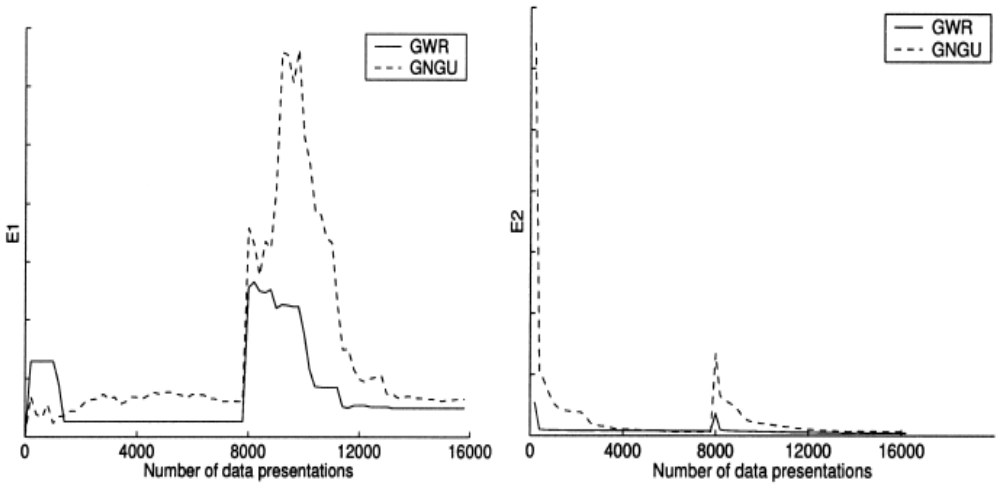


Figure 20: (a) measurement error E1 and (b) E2 for GWR and GNGU (Marsland, 2002).

In Figure 20, there is a change in the input map after 8000 data presentation. It can be noted that the GNGU model maintains, in case of E1, more distinct neighbors and much longer than the GWR model. For E2, the cost measure between the input pattern and the winning node that responded it. However, both networks recover rapidly and return to something like the same level as for original data distribution.

5.4 Limitation of GWR

The GWR network has more parameters to be adjusted compare to GCS and GNG model. There's no need to set such a parameter that indicates the growth rate found in GCS and GNG model, as GWR grows on demand. However, it is still necessary to adjust an age limit for the aging (death connection) a_{max}, but this information difficult to define. It is also necessary to adjust the activation rate limit a_T and the firing limit of a node h_T that together define the growth rate of the network and subsequently replace the parameter λ.

When inserting new nodes, the GWR algorithm is based on the conditions shown in the following table:

Condition	Activity	Firing	Insert
1	High	Low	No
2	High	High	No
3	**Low**	**Low**	**Yes**
4	Low	High	No

Table 5: Condition for inserting a new node in GWR.

Only acceptable criterion to insert a new node is condition 4, configures the input pattern in such a way that is far from the winning node and the winning node has a high number of firing. As like in GWR model, it does not distinguish heterogeneous information that can compose the description of states represented by its nodes, distortion occurs in the information when the nodes are modified according to the equation described in step 6(b) which considers the arithmetic mean between vectors, assuming a linearity in the represented data. Another problem of deformation of the data arises in condition 3, when the algorithm decides that a winning node that is far from the represented pattern and has firing at a low level, should also train and move toward the input pattern to represent it. Successive adaption steps indicated in the last equation in step 6(c), minimize the Euclidean distance between the input pattern and the information contained in the node. Therefore, also reduce the error when considering this computed distance. However, the heterogeneous information contained in the node is modified linearly and the fact of having a minor error no longer corresponds to the quality of the stored information.

Since the activity of the winning node is high in condition 1 and 2, the node is well represented the input pattern and the adaption that is made to fit the new input data is small and probably is related to minimize noise effects thus not to lead to large distortion.

The SOM model has a predetermined geometry in its connections. The TRN model creates connections, with the CHL algorithm, based on similarities of the nodes after their distribution by the input space made by NG. The GCS model maintains a hyper tetrahedron mesh formed by its connections, while in the GNG and GWR model the connections are simply indicators of

neighborhoods, which may disappear or reappear as the nodes connected by them. Since the information represented in the nodes can be heterogeneous and grouping criteria for the generation of trajectories, it is necessary a more complete interpretation of the role of connections in the space of states.

The GWR does not reduce the size of the state space and is able to represent it faithfully in different dimensions. It does not present topological defects for disconnected or concave areas.

Chapter 6

State Trajectory Generator (STRAGEN)

The State Trajectory Generator (STRAGEN), introduced by Benante (2008), is a self-organized incremental neural network model that can manage complex heterogeneous information, such as angles, torques and joint positions. Self-organized because it doesn't need a supervisor to teach how to create a topological map. The map is created using a given criterion that optimizes its structure. It is incremental because the model can grow and shrink as indicated by the measure of data and the quality of representation map. STRAGEN shields the heterogeneous information from being mixed with one another, by using n-dimensional pre-configured groups of similar data represented in each node of the network. This data contains kinematics and dynamical information, they are stored in the nodes during the training phase and can be used at any time, by pattern completion. Once the network had been trained with the problem space of the robot, it is enough and necessary to give the starting and target points, and STRAGEN trained network will generate a trajectory from one to another using a diffusion energy algorithm. The trajectory can be generated to optimize different criteria, such as minimum distance, minimum torque variation or minimum joint angles variation.

Benante (2007a) mentioned different phases of the algorithm. The algorithm of STRAGEN is composed of three phases:

(a) Training Phase: represents the topology of the solution space and adapts itself while reading samples from a database.

(b) Pruning Phase: eliminates unsuitable or unnecessary nodes and connections.

(c) Trajectory Generator Phase: the algorithm tries to find the best trajectory between two points, according to a given criterion.

STRAGEN can be subdivided into two approaches. The two approaches stand for an artificial neural network with the dynamic topology on non-linear systems. One is On-line approach and the other one is Off-line approach. The Off-line approach consists of all the three phases described above while the On-line approach consists of Training and Trajectory phases.

6.1 Description of the method

This section will describe two artificial neural work model with dynamic topology for a solution of generating the state trajectory for nonlinear system. STRAGEN off-line has a pruning phase of connections that must be performed after training, while STRAGEN on-line performs pruning during training. Both models are in common for definitions of variables, Motor Babbling procedure and calculation of validation procedure (Benante, 2008, Page 94).

In the beginning of training phase, a pre-processing phase is required to normalize the database $B^{(0)}$ of dimension $L \times D$, where L is the number samples (Lines) and D is the dimension of each samples i, such that $1 \leq i \leq L$, should be normalized before starting the training phase. Each sample $w_j \in \mathbb{R}^D$ is normalized using its maximum and minimum values, assuring that the normalized B such that $w_{ij} \in B \Rightarrow 0 \leq w_j \leq 1$

$$w_{ij} = \frac{b_{ij} - min_k(b_{kj})}{max_k(b_{kj}) - min_k(b_{kj})}$$

Where $1 \leq k \leq L$ and $b_{ij} \in B^{(0)}$. The weight vector is defined as, $w_i = [w_1 \dots w_D]^T$. The weight vector w may contain heterogeneous information from various domains. To deal with this different information, the weight vector is divided into groups having similar information, to form possible neighborhood criteria:

$$w_i = [V_1\ V_2 \dots V_m]^T$$

where the vector $V_i, i = 1, \dots, m$ represents a set of variables of a domain that can be grouped together and operated as a whole. The dimension of each group is $V_i \in \mathbb{R}^{D_i}$ and $\sum_{i=1}^{m} D_i = D$. To clarify this concept in a robotic domain, for example: groups can be together with angles and torques of the joints. A group, for example the spatial positions of the joints can be divided into subgroups that are vectors representing the spatial position of each meeting individually.

It is necessary to select $1 \leq \zeta \leq m$ to be the activity group used to calculate the activity of the network. Also to select $1 \leq \eta \leq m$ to be the neighborhood criterion, used to evaluate the proximity of one stimulus and its representation in the topological map among other proximity evaluations.

The activity threshold $\overline{a_k}$ for the group V_ζ is defined by a chosen percentage, 0<P<1, of the maximum Euclidian distance in dimension $D_{\zeta K}$, for each subgroup inside V_ζ that represents an

independent information: $\overline{a_k} = \exp(P.\sqrt{D_{\zeta K}})$, $1 \leq k \leq l$, where l is the number of homogeneous subgroups that compose V_ζ.

6.1.1 Motor Babbling (MB) Procedure

Motor babbling (MB) (Benante, 2008, Page 96) was used to create a cartesian position map to join motion coordinates. MB works to generate endogenous movements through state trajectories and learning isolated points on the presentation map from one coordinate system to another. MB helps the model to create connections between the nodes that have a significant effect in the generation of trajectories. STRAGEN adopts MB to form trajectories while the isolated points as stimuli. It is noted that SOM based networks suffer from structural hill climbing. Taking advantage of this property, i.e., the SOM are known to be sensitive to the order which data is presented. Therefore, presenting the data in a convenient order that helps the algorithm to create nodes and connections in a more feasible way.

For each iteration t, when a new sample is required for the Training phase, MB procedure p ($\boldsymbol{\xi}$) returns some random pattern $\boldsymbol{\xi}$ on first execution. From second execution onwards, for each new sample required by the network, MB chooses Q random candidates not yet presented. For these Q candidates, MB returns the nearest to the last presented input $\boldsymbol{\xi}$, according to some criterion of minimum Euclidean distance. In this way, the training is randomly directed within a neighborhood. The MB procedure is given below:

1. If t =1, choose a random pattern $\boldsymbol{\xi}(\boldsymbol{t})$ from database B of dimension $L \times D$, and remove this sample to avoid repetition.
2. If t >1, choose with uniform probability Q candidates to be next pattern $\phi_i, i = 1, \dots, Q$ and choose

$$\boldsymbol{\xi}(t) = \arg\min_{\forall i}\{Dist(\boldsymbol{\xi}(t-1), \phi_i\}$$

Remove $\boldsymbol{\xi}(t)$ from the database to avoid repetition.

This procedure is the part of Training phase. Once the network is trained, it is possible to calculate the validation error of the topology.

6.1.2 Validation Phase

In STRAGEN off-line, the validation phase occurs after the pruning phase. In STRAGEN on-line, since there is no pruning phase, the validation phase can be performed during the training phase. The validation phase allows to check the error of representation of the problem space, generated by the node. The procedure is given as follows:

1. Repeat for $i = 1, \ldots, I$ iterations:
    - Generate an input signal ξ from the database according to MB procedure
    - Calculate the iteration error i, the distance between the sample and nearest node according to the criterion adopted:

$$e_i = \|\xi - w_{s1}\|$$

2. Return the validation error as:

$$E_v = \frac{\sum_i e_i}{I}$$

6.1.3 Initialization of the Algorithm

Let A be a set of nodes and C be a set of connections between these nodes. Let the input distribution be P (ξ), for inputs ξ as defined in MB procedure.

Let the learning rate be ε_b. Define the neighborhood creation η and the activity creation ζ such that $1 \leq \eta, \xi \leq m$, for some group V_η and V_ζ and the final learning rate decay be $\alpha_f \approx 0$.

Before starting the algorithm, initialize the set A with 2 nodes n_1 and n_2 and placed at w_{n1} and w_{n2}. In $\mathbb{R}^D$, representing 2 random patterns from the dataset:

$$\text{A} = \{n_1, n_2\}$$

in which:

$$w_{ni} = [w_{i1} \ldots w_{iD}]^T = [V_1 \ldots V_\zeta \ldots V_\eta \ldots V_m]^T$$

Initialize the connection set C with one connection between the first two nodes $C = \{c_{n1,}c_{n2}\}$. These parameters settings and initialization are common both in STRAGEN on-line and STRAGEN off-line.

6.2 STRAGEN Off-Line

Various phases of STRAGEN off-line in a flow diagram (Figure 21). Motor babbling is set to data samples to be used during the training phase after pre-processing step.

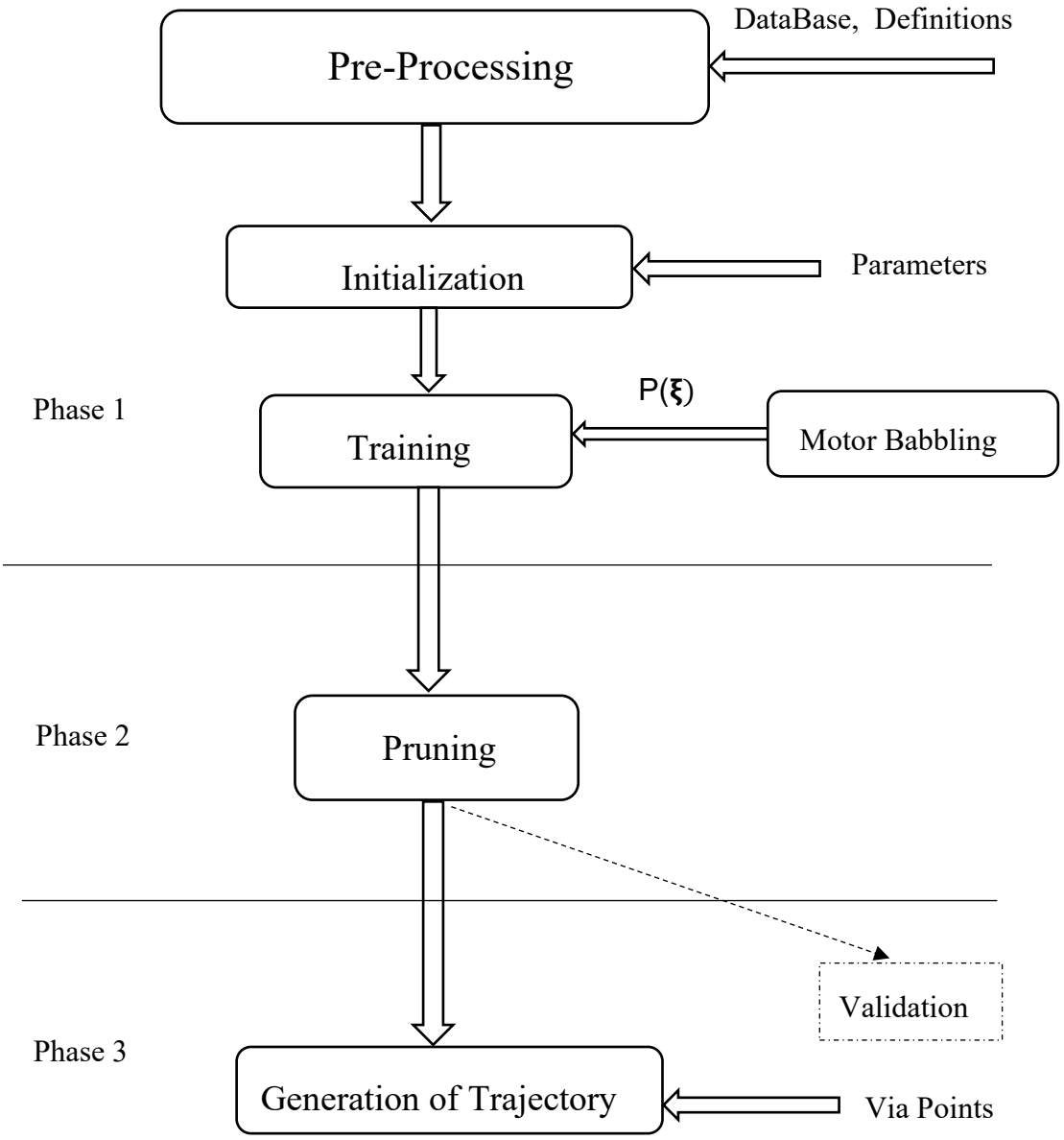


Figure 21: Flow-chart of STRAGEN Off-Line (Benante, 2008).

In the Training Phase, the model adapts and creates a representation of the state space topology while the MB procedure reads the database. In the Pruning Phase, the model reassesses the connections created during training and eliminates those that are considered unnecessary or unwanted. In the third phase, the Generation of Trajectories, the algorithm finds the best path

between two points according to the energy diffusion algorithm (Zeller, 1997). After the training phase and before the trajectory generation phase, one can perform the Validation Phase to measure the topological map error created by the model.

6.2.1 Training Phase

The training algorithm described by Benante (2008, Page 98) as follows:

1. Generate a data sample from P (ξ) as input to the network, according to Motor bubbling procedure,

$$\xi = [\xi_1 \dots \xi_\zeta \dots \xi_\eta \dots \xi_m]^T$$

2. For each node i in the network, calculate the distance from the input $\|\xi_\eta - V_{\eta,i}\|$, and determine the best matching unit and the second best s_1 , $s_2 \in A$, using the chosen neighborhood criterion η, such that:

$$\|V_{\eta,s1} - \xi_\eta\| \leq \|V_{\eta,i} - \xi_\eta\|, \forall i \in A$$

$$\|V_{\eta,s2} - \xi_\eta\| \leq \|V_{\eta,i} - \xi_\eta\|, \forall i \in A - \{s_1\}$$

3. Add one to the number of wins of s_1: $\sigma_{s1} = \sigma_{s1} + 1$.

4. Insert a new connection between s_1 and s_2 in C, if there is not one yet

$$C = C \cup \{C_{s1,s2}\}$$

5. To calculate the activity of the stimulus ξ with respect to winner node s_1, it is necessary to use only part of the information in both vectors. The information used is the group number ζ, that defines the activity criteria. Then, V_ζ is subdivided in l homogenous subgroups per joint, because the activity is calculated independently for each joint:

$$a_k = \exp(-\|\xi_{\zeta_k} - V_{\zeta_{k,s1}}\|) , 1 \leq k \leq l$$

6. If any activity is less than the established threshold for that group, ($a_{1,s1} < \overline{a_1}$ OR $a_{k,s1} < \overline{a_k}$), then a new node s_3 must be added in the exact location of the input sample:

a) Add the new node s_3 to the A set: $A = A \cup \{s_3\}$

b) Create a new weight vector associated with the node s_3, i.e., $w_{s_3} = \boldsymbol{\xi}$

c) Remove the connection (s_1, s_2) from C

d) Calculate the distance $Dist = \{Dist(s_3, s_1), Dist(s_3, s_2), Dist(s_1, s_2)\}$ where:

$$Dist(s_3, s_1) = \left\| V_{\eta,s_3} - V_{\eta,s1} \right\|$$
$$Dist(s_3, s_2) = \left\| V_{\eta,s_3} - V_{\eta,s2} \right\|$$
$$Dist(s_1, s_2) = \left\| V_{\eta,s1} - V_{\eta,s2} \right\|$$

e) Select the 2 shortest distances $Dist_1 and\ Dist_2$:

$$Dist_1(s_i, s_j) = \arg\min(Dist)$$

$$Dist_2(s_i, s_k) = \arg\min(Dist - \{Dist_1(s_i, s_j)\})$$

f) Insert new connections between the nodes considered to determine $Dist_1(s_i, s_j)\ and\ Dist_2(s_i, s_k)$.

$$C = C \cup \left\{c_{s_i,s_j}, c_{s_i,s_k}\right\}$$

7. If a new node was not inserted in step (6), update the positions of the winning node s_1:

$$\forall w_{s1} = \rho \times (\xi - w_{s1})$$

Where:

$$\rho = \begin{cases} \epsilon_b \times \alpha_f^{\left(\frac{\sigma}{\sigma_f}\right)}, & \sigma \leq \sigma_f \\ \epsilon_b \times \alpha_f, & \sigma > \sigma_f \end{cases}$$

And $0<\varepsilon_b<1$ is the learning rate; $\alpha_f \approx 0$ is the final learning rate; σ is the counter of the number of times that a winner node has fired, and α_f is the maximum number of times a node is supposed to fire.

8. Repeat from step 1 up to the maximum number of iterations t_{max}.

6.2.2 Pruning Phase

After the training phase (Benante, 2008, Page 100), run the pruning phase for I iterations to remove unsuitable nodes and links. The pruning phase will eliminate all unused links that do not disconnect the graph, all unused nodes and all isolated nodes.

1) Create a set $N = A$ of all nodes and a set $E = C$ for all links.
2) Repeat for I iterations:
    - Generate a data sample $P(\xi)$, according to MB procedure.
    - Exclude the winner node s_1 from the $N = N - \{s_1\}$ and exclude the connection c_b between the best and the second-best unit, from set $E = E - \{c_b\}$.
3) Repeat for all connections remaining in E:
    - Check if a possible deletion of connection c_b between two nodes creates a disconnected graph.
    - If that is the case, do not delete the connection c_b. Otherwise, delete connection c_b permanently.
4) Eliminate all nodes remaining in set N (nodes that never won a competition)
5) Eliminate isolated nodes, i.e., disconnected nodes.

An example of removal of the connection is shown below in Figure 22. In (a) before removal, the network presents two candidate connections for removal. The top link in (a) that connects two nodes, should be removed for any input pattern presented. After deletion, in (b) STRAGEN holds the central connection between two nodes.

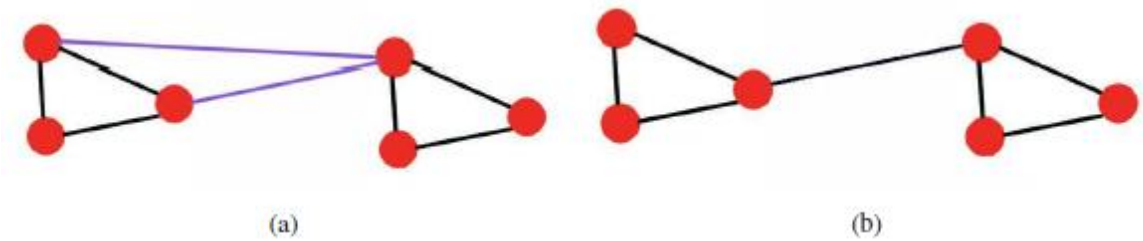


Figure 22: Connection removal in STRAGEN Off-line. (a) before removal (b) after removal. (Benante, 2008).

6.2.3 Trajectory Generation Phase

Generation of trajectory is explained by Benante (2008, Page 102). The use of diffusion energy (Zeller, 1997) allows to find the trajectory of the trained network. It is possible to provide two or more possible points in the state space after training the network. Trajectory is obtained as output from the network that passes through these points which have high energy distribution rather than others starting from the initial point, which has the lowest.

Let any initial point ξ_{int} and target point ξ_{targ} find the best matching node n_{int} for the initial point and the best matching node n_{targ} for the target point among trained neurons.

Set an energy diffusion function that defines a flux of energy diffusing through the links of the network such as $f(t,n)$, for all nodes n, aiming to find a chain of nodes $n_{S,S-1\ldots,1,0}$, starting from the target point n_{targ}= n_S to the initial point n_{int}= n_0, where S is the (unknown) size of the trajectory.

The algorithm of Zeller (1997) is given below:

1) Initialize the diffusion function $f(0,n) = 0\ \forall n \neq n_{targ}$ and $f(0,n_{targ}) = 1$
2) Repeat for all $n \in A$ until $f(t,n_{int}) \neq 0$

$$f(t+1,n) = \begin{cases} 1, & \forall t, if\ n = n_{targ} \\ \mu \sum_{j \in N_n} f(t,j), & if\ n \neq n_{tag} \end{cases}$$

Where N_n is the set of all nodes that are neighbors of n and $|N_n|$ is its cardinality, and $\mu = K/N_n$, K<1 such as $K = |N_n| /(|N_n| + 1)$.

This procedure starts diffusing energy from the target, towards all the nodes, until perhaps reaches the initial node which is guaranteed if there is a possible route (Zeller, 1997). The trajectory is formed when the initial node receives any amount of energy and to define it simply, start from the initial node always choosing to next node the neighbor of higher energy, until the target node (single node with energy equal to one) is found. The final trajectory is formed by the nodes $T = \{n_0, n_1, \ldots, n_{S-1}, n_S\}$. The diffusion algorithm of energy does not take in account the distance between nodes, but only their neighborhood.

6.2.4 Difference Between STRAGEN Off-Line and GWR

Benante (2008) explained the difference between STRAGEN off-line and GWR in Page 103. STRAGEN off-line demonstrates a series of ideas that make it possible to generate trajectories when compare to other models described in above Chapters. In comparison to GWR model, from which STRAGEN was initially inspired as the basis for the network.

The topological map training has the MB phase, which permits the STRAGEN model to make the state space representation by considering comparable states as per the vicinity measure given by the expressed separation (neighborhood) rule. MB strategy allows the creation of maps with coherent state transitions from the beginning of the training, by giving, regardless of whether randomly guided, an order in the state transition.

The pre-processing phase of STRAGEN presents principal changes for the treatment of heterogeneous data from nonlinear framework in order to produce state trajectories. The vital point is to normalize database that made to remove possible impacts between distinct groups of information for the calculation of neighborhood proximity. The grouping of this information in the node are additionally fundamental with the that they are dealt with independently, staying away from distortions in the adaption operations and permitting their use as an improved standard of the recognized framework.

The GWR has several parameters to be adjusted. Nonetheless, among those parameters, two of them are essentially identified to the insertion of nodes: the activity limit and the firing limit. STRAGEN does not use firing limit to insert nodes. With respect to the activity limit, it is constant and modifies by a percentage P which is based on the maximum size of Euclidean distance of normalized state space. The value of percentage P is 1% and for more accurate map, it is 3% in vast majority of cases. In Step 4 of the training phase, the inserted connection represents the possible route between the nodes s_1 and s_2 , since they are the two nodes nearest to the provided point ξ , as indicated by the criteria. The process is equivalent for both in STRAGEN and GWR.

The method for the insertion of new nodes is done in STRAGEN is a vital component of the model. According to the table 5, in the GWR algorithm a new node is inserted if and only if the new winner is distant and active. A table of inserting of a node in STRAGEN is given below:

	Activity				
Condition	$V_{\zeta 1}$	$V_{\zeta 2}$	…	$V_{\zeta n}$	**Insert**
1	High	High	…	High	No
2	Low	High	…	High	Yes
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
3	Low	Low	…	Low	Yes

Table 6: Condition for inserting a new node in STRAGEN.

By looking at Table 6, it is noted that the procedure of inserting node in STRAGEN is more combative, adding a node whenever the activity of the winning node is not sufficiently considered for all groups that make up the activation criterion. STRAGEN makes separation between two ambiguous positions where the end effector is at the same point, but the other joints are in various configurations not being treated similarly. To illuminate table 6, consider a two-dimensional robot, the activation criterion for the spatial position of the joint is V_{ζ}, the position of the joint J_2 is $V_{\zeta 1}$and the position of the end effector J_3 is $V_{\zeta 2}$ considered seperately. In step 6, STRAGEN off-line algorithm shows the process of inserting new nodes. In GWR, as observed, in addition to node activity, the frequency of firing is also evaluated, so a new node will only be added if the winner is far from the standard and has been used frequently. When one of these two characteristics fails (activity and frequency), rather than inserting a new node the training (displacement) of the winning node is made towards the input pattern. STRAGEN off-line learned nodes must stay to the points that they represent and ought not move to keep away from the distortion of information about the learned state. The way of adapting STRAGEN, together with the conditions of insertion of a new node, gurantees that a node will only adapt if it is activated by an input pattern close enough to be considered a noise of the same pattern.

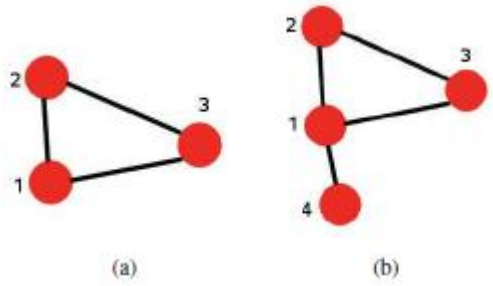


Figure 23: Example of insertion of nodes and connection in STRAGEN (Benante, 2008).

When the network inserts nodes, STRAGEN recalculates the connections that this new node will have with the winning nodes and vice. The process can be understood by Figure 23. Sub-figure (a) shows the network before insertion of node 4. In subfigure (b), node 4 has as topologically close to the nodes 1 (winner) and 3, vice. STRAGEN calculates the distances between the pairs of nodes (4,1), (4,3) and (1,3), and holds only the two smallest connections, measured according to the established neighborhood criterion, in this case the connections (4,1) and (1,3), so that the inserted node 4 has only one connection.

6.3 STRAGEN On-Line

The algorithm is composed of two phases, intermediate stage of Pruning phase does not exist here. Firstly, the training phase creates a representation of the state space topology, adapting and pruning unnecessary connections and useless nodes during the process. Secondly, trajectory generation phase, the model is requested to find the best trajectory between two points directly or passing through intermediate points (called via-points). During the execution of second phase, the validation error is calculated.

STRAGEN on-line also performs the pre-processing phase described in section 6.1 before starting the training phase, to normalize the database $B^{(0)}$, characteristic of the weight vector $w_i = [w_{1\ldots} w_D]^T = [V_1\ V_2 \ldots V_m]^T$, neighborhood creation η and the activity creation ζ. Calculate the activity limit $\overline{a_k}$ based on percentage P of the space state that a node must respond, usually P=1%.

Various phases of STRAGEN on-line shown in the following flow diagram (Figure 24). Motor babbling is set to data samples to be used during the training phase according to section 6.1.1. Initialize the algorithm as described in section 6.1.3.

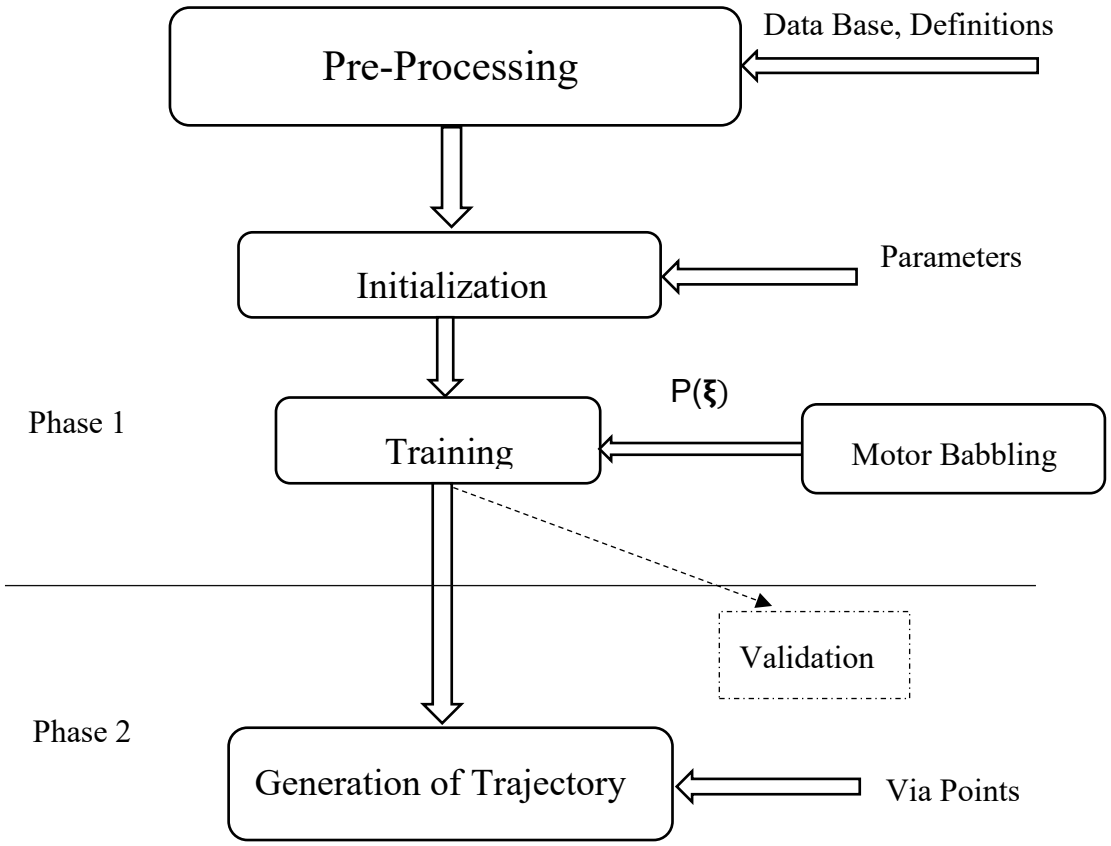


Figure 24: Flow-chart of STRAGEN On-line (Benante, 2008).

6.3.1 Training Phase

STRAGEN On-line training algorithm follows the STRAGEN Off-line, with the difference of few modifications. This algorithm can eliminate nodes and connections during the training phase without the need of the pruning phase. The procedure of STRAGEN On-line is identical from Step 1 to Step 7 of STRAGEN Off-line. The algorithm described by Benante (2008, Page 108) given below from Step 8.

8. Calculate the mean size and standard deviation of all connections $k = |N(s_1)|$ emanating from the winning node s_1, and the threshold for removing connections χ:

$$\bar{m} = \frac{\sum_{i=1}^{k} Dist(s_1, n_i)}{k}$$

$$\bar{d} = \sqrt{\frac{\sum_{i=1}^{k} |Dist(s_1, n_i) - \bar{m}|^2}{k-1}}$$

$$\chi = \bar{m} + \bar{\omega} \cdot \bar{d}$$

Where $|\,\mathrm{N}(s_1)\,|$ is the number of neighbors of s_1, $n_i \in \mathrm{N}(s_1)$ and $\bar{\omega} = 1.5$, is an empirically determined optimal constant.

9. If $|\,\mathrm{N}(s_1)| > 2$, remove all connections $c_{s_{1,n_i}}$ from set C for which we have $\mathrm{n} \in \mathrm{N}(s_1)$ and $Dist(s_1, \mathrm{n}) > \chi$.
10. Remove all isolated nodes $\mathrm{n} \in \mathrm{A}$, i.e., nodes without at least one neighbor.
11. Repeat from step 1 mentioned in 6.2.1 to the maximum number of iterations t_{max} or some stop criterion is satisfied.

6.3.2 Trajectory Generation phase for smallest path

Given any initial point ξ_{q0}, target point ξ_{qf} and any intermediate point ξ_{qi}, $0 < l < f$, called via-points, find the smallest path according to algorithm (Dijkstra, 1959) from q_o and q_f, passing through all via-points q_i.

1. Let be the chain of target points $q_o \ldots q_f$
2. Let $i = 0$ and $j = 1$
3. Repeat until $j = f$
    - Find the winning nodes s_i and s_j which represents q_i and q_j respectively

$$\|V_{\eta,s_i} - \xi_{\eta,qi}\| \leq \|V_{\eta,k} - \xi_{\eta,qi}\|$$

$$\left\|V_{\eta,s_j} - \xi_{\eta,qj}\right\| \leq \|V_{\eta,k} - \xi_{\eta,qj}\| \;\; \forall k \in A$$

- Find the shortest path from s_i to s_j according to Dijkstra algorithm.
- Move to next stretch, $i = i + 1$ and $j = j + 1$.

Since all nodes are required to follow some path, each node provides additional information for the complete execution of the trajectory, avoiding the need of calculation.

6.4 Difference between STRAGEN Off-Line and STRAGEN On-Line

STRAGEN off-line has the pruning phase that allows the creation of a topological map that represents the state space very well. During the pruning phase, one has full access to configure the final map and to choose the connections and nodes that will be removed according to well-characterized criteria. STRAGEN on-line eliminates nodes and connections that do not win once for all input points presented in the pruning phase that performs during training as long as the map is not disconnected. This method affirms that only futile connections are eliminated. It helps to keep the map connected allowing all states to be accessible. Conversely, STRAGEN off-line does not appropriately represent disconnected areas.

STRAGEN on-line exhibits a smooth technique for the disposal of connections in a dynamic and programmed way which takes into account the density of nodes that is currently representing the map to be learned. The actualized methodology is straightforward, innovative and comprises of wiping out connections for which the distance of neighbors of the winner are larger than the threshold χ only for nodes that have more than two neighbors, as described in Step 9.

STRAGEN on-line map can also represent disconnected areas, created in such a way that no isolated nodes emerge in the process and without the computational cost of evaluating each connection removed. STRAGEN on-line does not remove connections from nodes that have just a single or two connections. There are two steps in the algorithm that remove connections. Step 9, which deals with the removal of connections, and Step 6 in section 6.2.1, which carries out the node insertion process. Step 6c, although removing one of the connections between the

winning nodes near the inserted node, does so only to replace it with others that best represent the similarity between the three nodes (winner first, second and new inserted node), and the third method is in which it removes nodes, which does not have at least one neighbor (no connections/isolated).

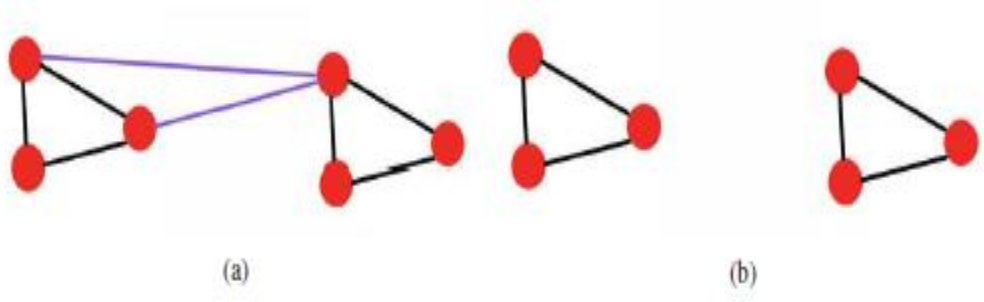


Figure 25: Example of STRAGEN On-line. (a) before removing connection (b) after removing connection (Benante, 2008)

Figure 25 shows an example of the removal of the connections larger than χ.

Chapter 7

Results And Conclusions

The algorithms of GNG, GWR, STRAGEN explained in chapters 4, 5 and 6 respectively, were implemented successfully using MATLAB software related to MATHWORKS.

7.1 Simulation

The algorithm was divided in two distinct parts for their implementation. Phase 1 comprises the training of the network. Phase 2 comprises the generation of the state trajectory and open loop control. The parameters used will always be: learning rates of the winning node was ε_b= 0.1 and for its neighbors ε_n = 0.007, maximum age a_{max} =50, λ = 200, β = 0.995 and α = 0.5 for GNG, learning rate of the winning node ε_b = 0.2 and its neighbors ε_n = 0.006, maximum age a_{max} = 50, activity threshold a_T = 0.8, firing threshold h_T=0.1, initial strength h_0 = 1, stimulus strength S(t) = 1 of firing node, constants controlling the behavior of the curve α_b = α_n = 1.05, t_b = 3.33, t_n = 14.3 for GWR, final learning rate α_f =0.1, activity threshold a_T = 0.8, maximum of estimated firing per node σ_f = 2* t_{max} /L, and number of candidates in the MB procedure is defined as Q = 100 for STRAGEN. For STRAGEN, experiments will be performed, one to each criterion defined above, in which the database is used as input data with points representing the position of the end-effector of the robot (2D and PUMA-560). The maximum number of iterations t_{max} = 40000, used to train the network.

The motivation behind these simulations is to test the models, to collect data with respect to the trajectories generated and to compare the neighborhood criteria and how they influence the trajectory and mapping of the input space.

At the end of the section, the results will be compared with the simulations done with the models GNG, GWR and STRAGEN.

.

7.1.1 Network Training (Phase 1)

This segment exhibits the execution of the models GNG, GWR, STRAGEN on various systems. The first is an extremely straightforward, which implemented on a two-link robot's dataset in 2D Figure 4, and second actualized on PUMA-560 database in 3D Figure 7, intended to demonstrate how the algorithms generate mappings from information spaces of (generally) high dimension to bring down dimensional map fields. The first simulations above in subsections 2.2.3 was done with the sequential presentation of the points, so that the joint 1 (shoulder) rotated the whole cycle, of 0^o to 360^o (at the interval of 3), then for joint 2 (elbow) rotate to the next range (angles defined of 0^o to 360^o, at the interval of $30^o \cdot$rand) for two link robot, until all the points were presented once.

Presenting the points in an order that allows the learning of neurons and their connections makes it possible to simulate. GNG and GWR follow random input pattern, while STRAGEN follows the MB characteristics, in which the first input pattern is randomly followed by the input patterns nearby to the previous one, and removing already used in order to avoid repetitions. However, this order must not be sequential, but rather contain some randomness that allows the model to create diverse neighborhoods and with greater robustness and stability.

Figure 26 and Figure 27, demonstrate the maps produced by utilizing previously mentioned models for training the 2-dimensional and 3-dimensional database when the neighborhood criterion: the Euclidean distance between the positions of the end-effector (DE). It very well may be seen the assessment of the network when the neighborhood criterion DE was used. This criterion of special distance between nodes is generally used in artificial neural networks. The adopted neighborhood rule is the key to the following stages of the algorithm.

The 2D network trained by models GNG, GWR and STRAGEN, with this criterion (DE), generated a map of 202, 397 and 412 nodes. Which were able to represent the training of 2736 points with cumulative errors of 0.31704, 0.12954 and 0.10389 respectively. These results are obtained for 40000 iterations and it can be seen that for the same number of iterations different nodes have been created. The cumulative error measures the quadratic Euclidean distance between the input patterns and the winning nodes representing them. In the case of error equal to zero it indicates that the pattern is exactly represented by a node. The accumulated error of all nodes informs if the mapping is representing the input data appropriately. As it will be seen later, the errors between the input space mappings using the adopted criteria vary little and are

therefore not decisive in the choice of the best criteria. The best criterion should be chosen based on information other than error, for example characteristics about the types of trajectories it generates, such as minimal trajectory, smooth movement and no jerks, beyond the coverage of state space. In general overview, trajectory generated in the upcoming section is also linked to the number of nodes generated in training. As the diffusion energy starts with two random nodes: the initial and final one and distributes the energy from the final until it reaches the initial one. If there exist a higher number of nodes in the trained data, there will be more certainties to find the most correct trajectory. This can be one of the reasons of robustness of STRAGEN.

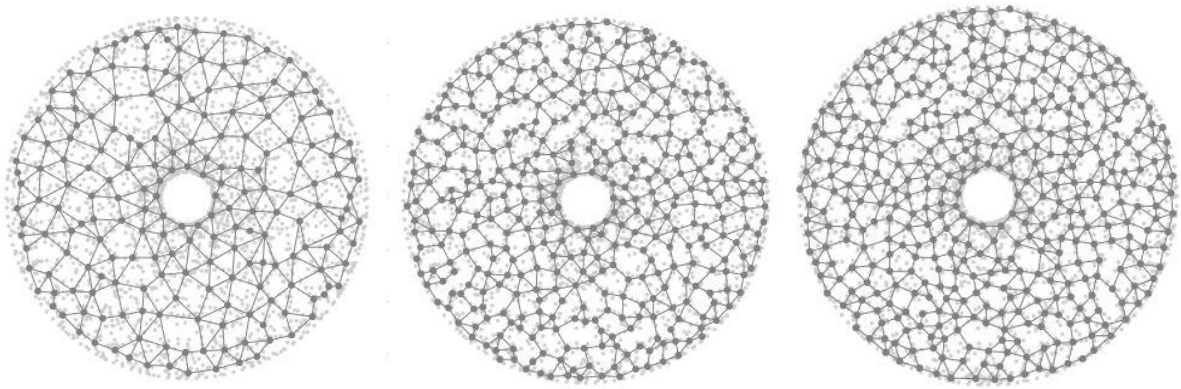
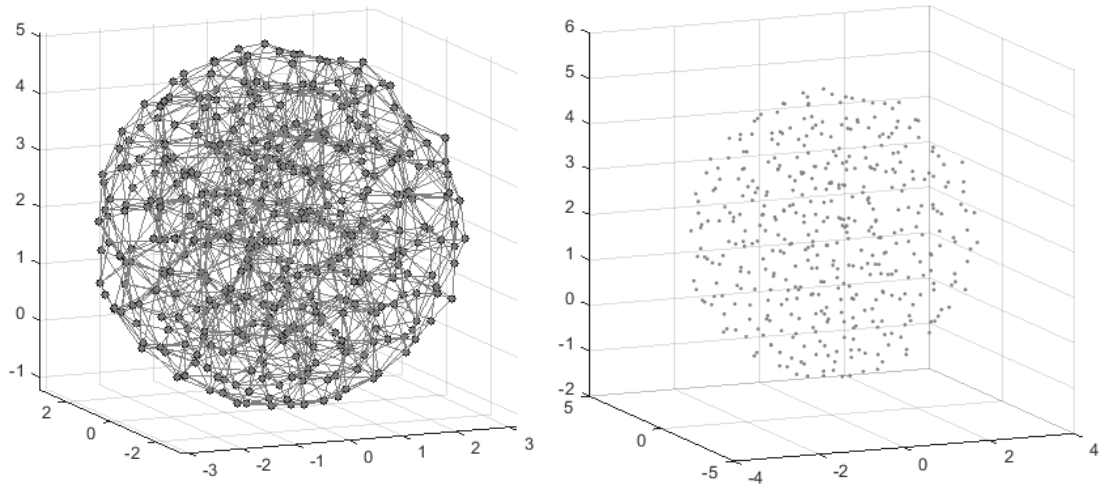


Figure 26: Trained Network of GNG, GWR and STRAGEN. Figure shows 2D networks, (Left) GNG, (Middle) GWR, (Right) STRAGEN after training.

Figure 27 demonstrates 3D network trained by models GNG, GWR and STRAGEN, by using criterion (DE) generated a map of 427, 752 and 778 nodes. Which were able to represent 15257 high dimensional base points to lower dimensional training with cumulative errors of 0.5670, 0.5025 and 0.4868 with iterations 85000, 10000 and 10000 respectively.



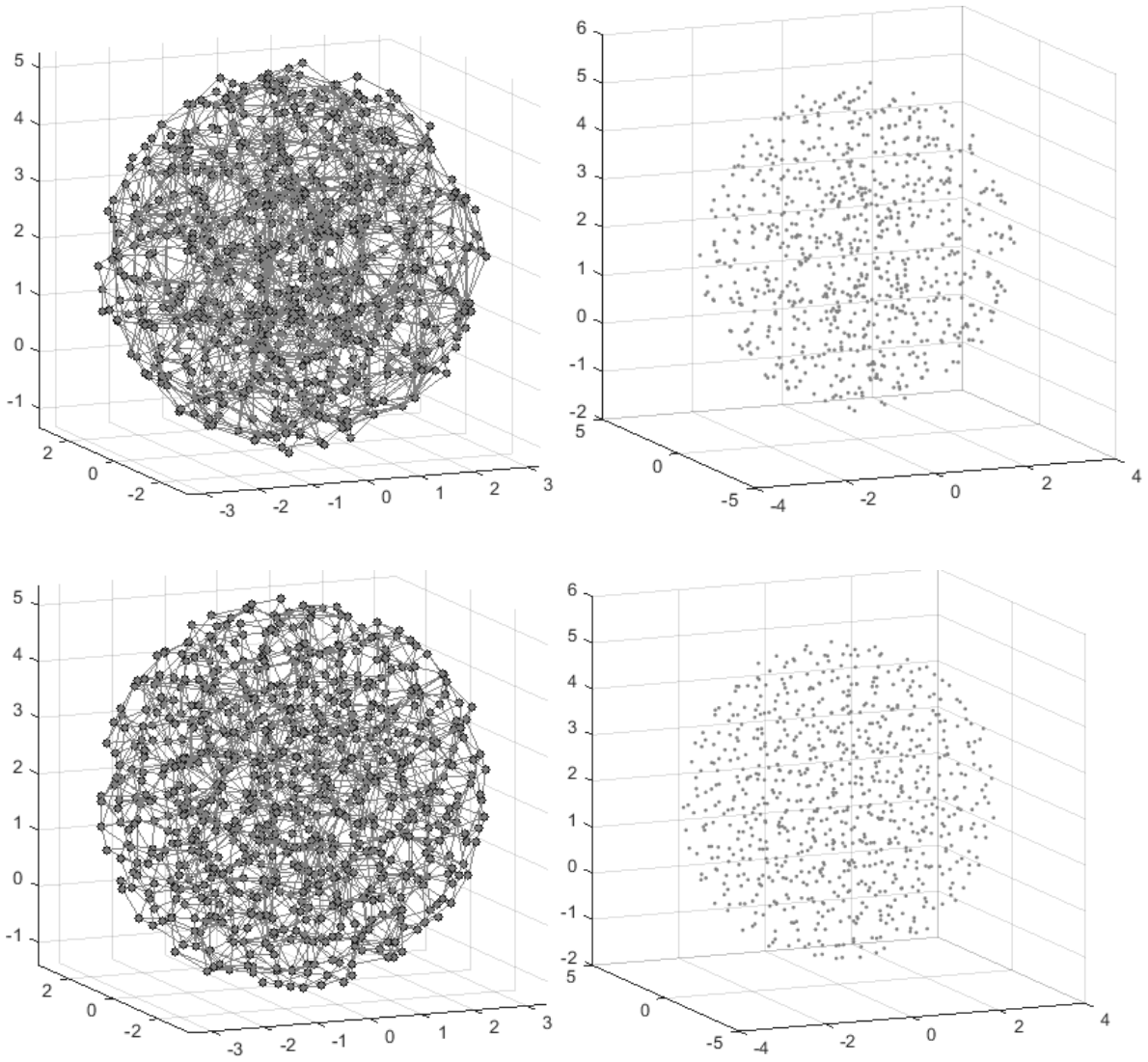


Figure 27: Trained 3D Network by GNG, GWR and STRAGEN with (all left) and without (all right) connections. Following in vertical direction (first) GNG, (second) GWR, and (third) STRAGEN after training.

7.1.2 Trajectory Generation (Phase 2)

Network training is followed by the phase of its trajectory generation. Diffusion energy algorithm was tested for this phase.

Energy diffusion algorithm permits the passage of any two in the input space and automatically chooses as starting node the triumphant node (particularly close) from the first entry point, and the final nodes the winning node of the second entry point.

Once you have chosen the desired points, the process of dispersion of energy begins. In the beginning, all the trained points have dissemination energy equivalent to zero. This procedure

dependably starts from the final node, which will have the energy equivalent to 1, from which fractions will go to their neighbors and neighbors of neighbors, successively until the energy of an underlying node isn't exactly same as zero.

The generation of trajectory can be understood by looking the Figure 28. The white node in this figure shown below represents the beginning of the trajectory and the black node (marked with energy equal to 1) represents the last node. The dispersion thinks a unit of vitality in the last hub and disseminates this vitality towards the underlying hub until the point that its esteem is not quite the same as 0. The generation of trajectory follows from the initial node towards the final node, choosing as the next node the one with the highest energy among the neighboring nodes connected to the previous node. On the off chance that there exist a different selection of nodes with the same amount of energy, the algorithm will choose the first one.

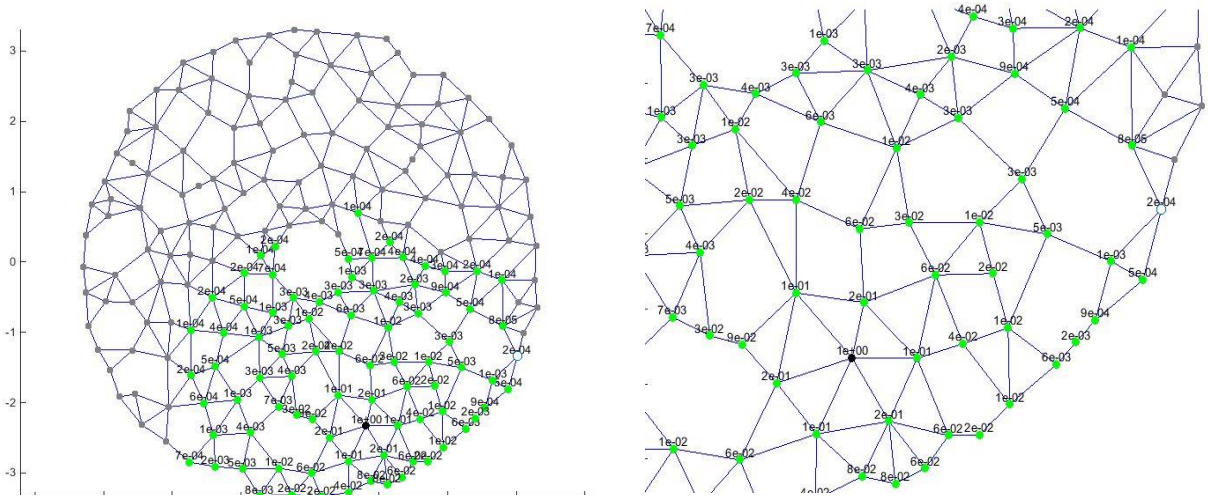


Figure 28: Diffusion of Energy: Trajectory generation by the nodes, from the initial node to the target node, with the energies. The left figure shows Diffusion of energy from the target point (black) to the starting point (white). The right one shows a better view of the energy values .

The simulation in Figure 28 has been carried out with criterion DE, it can be observed that for this simulation the algorithm easily finds the initial node, starting from the final node and diffusing the energy from neighbor to neighbor. Most of the network does not even receive some energy. Once the initial node is reached, the diffusion algorithm is terminated, because it has a trajectory linking both nodes. In case of the criterion DE, the neighborhoods denote the nearest nodes specially in relation to the position of the end-effector and the algorithm finds the smallest possible route in number of nodes (even though there were ambiguities in the choice of trajectory stretches).

Trajectory formed by the nodes with the energy values are: $2e^{-04}, 1e^{-03}, 5e^{-03}, 1e^{-02}, 4e^{-02}, 1e^{-01}$,1. Starting from the initial node, with energy of $2e^{-04}$, the options of selecting the next node are the lower node with value $5e^{-04}$, the upper node with energy of $8e^{-05}$ and the middle one that is chosen has $1e^{-03}$. In this second node of the trajectory, the options from which to select the one with energy greater than the previously selected node, are the lower node $9e^{-04}$ and the upper node that is chosen $5e^{-03}$. From this third node, there is an ambiguity. This node has four links with energies $1e^{-03}$ (returning to the previous node), the upper node with energy $3e^{-03}$, and two higher-energy nodes with values equal to $1e^{-02}$. The algorithm decided to walk underneath because it was the first node (the lower one) with the energy higher it acquired. From the fourth node it has four links and among them choose the highest energy having value $4e^{-02}$. From this fifth node, it walks for the sixth value $1e^{-01}$. From the sixth node, finally reaches the target node of the trajectory (seventh), with energy equal to 1. By calling the lower node (or the chosen one), the upper ambiguous option and so on, we can generate all the trajectories by following this method, and with this we realize that choice does not lead to in bad choices, because trajectories will always have the same total cost. The final trajectory can be shown in the following figure:

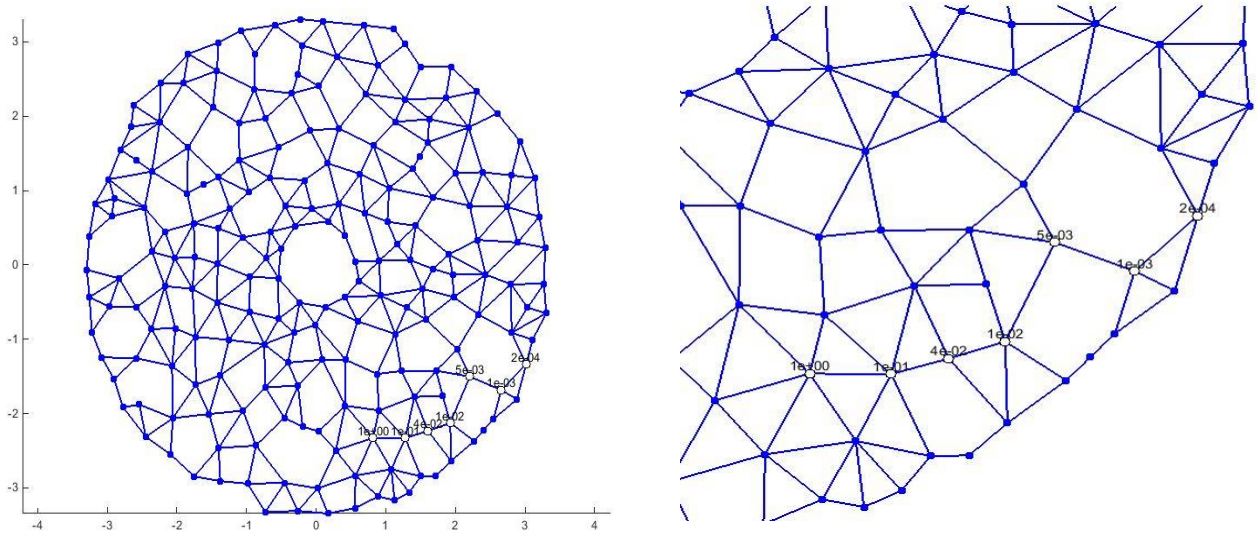


Figure 29: Trajectory formed by diffusion Energy. The left figure shows trajectory of the trained network and right one stands for a better view and nodes are indicated with corresponding energy values.

The diffusion algorithm, as implemented, is faster than other methods like Dijkstra, because it calculates the diffusion of energy from final node, until it reaches the initial node with energy

different from zero, unlike in Dijkstra it calculates the distances between all the nodes of the network, which creates time-consuming problem in complex and large networks. Diffusion energy consists of opening up in amplitude, several layers of energy, from the final node, until a layer finally reaches the target creating the trajectory.

7.1.2.1 Two-Link Robot

Final trajectory for a two-link robot according to the trained network of GNG, GWR & STRAGEN can be seen in Figure 30. The procedure of finding trajectory is the same for all of the three networks. Initial node labeled in white and final node labeled in red color in this figure. Trajectory is formed by diffusion algorithm according to the above-stated procedure.

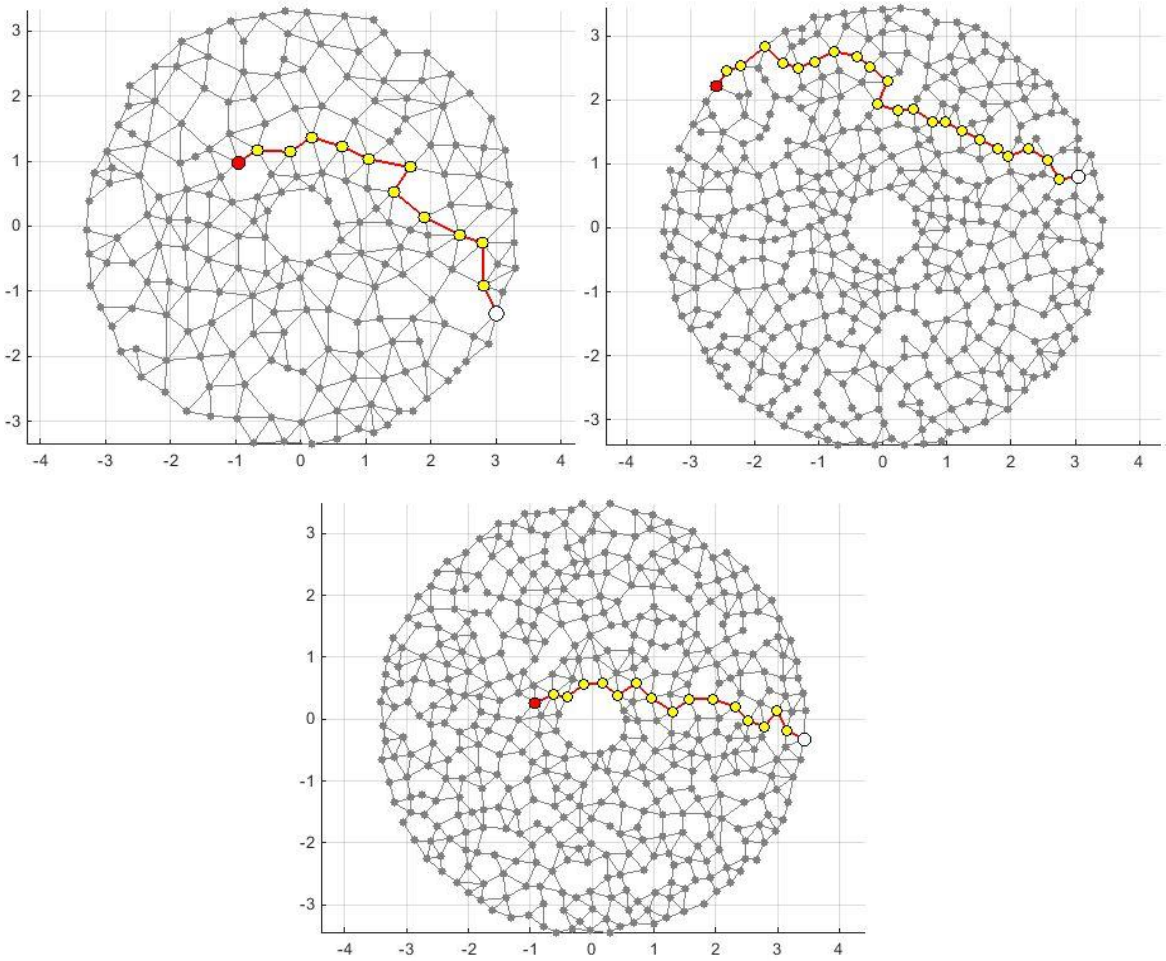


Figure 30: Final trajectory of two-link Robot. Top left and top right figure show the trajectory of the trained network of GNG and GWR. The bottom one shows for STRAGEN.

7.1.2.2 PUMA 560

Trajectory of Puma 560 has been achieved according to same procedure described in section 7.1.2. Puma robot has a 6 DOF structure and the data points of this robot has been done in section 2.3.3. As the Puma robot moves in 3D space, it has more data points compare to two-link robot. These data points have been trained by GNG, GWR and STRAGEN. After training, it is optimal also to find the trajectory of this robot. As it has presented 15257 data points in 3D space, there exists more connection between the nodes after training.

Figure 31 presents trajectory of Puma 560 trained by GNG network. The right figure shows the trajectory only taking into account the nodes for a better view.

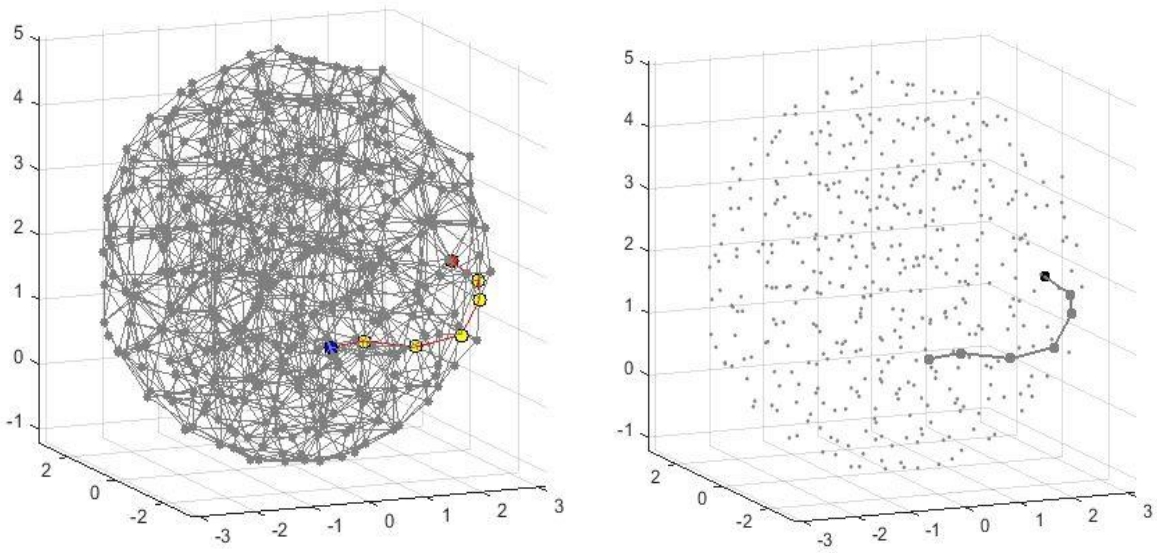


Figure 31: Trajectory of Puma 560. Network trained by GNG.

The same approach goes for GWR network and can be seen below:

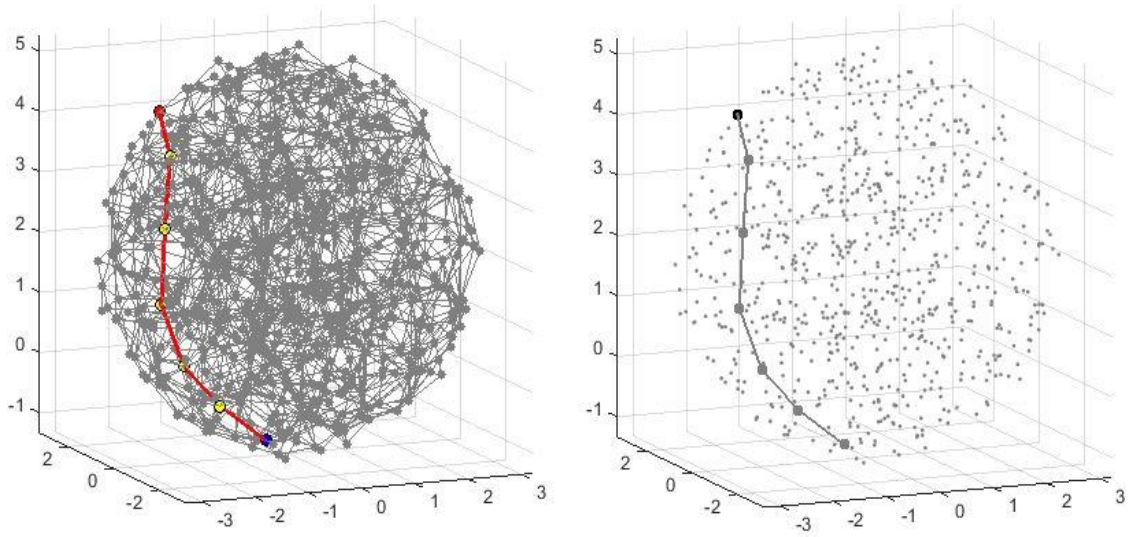


Figure 32: Trajectory of Puma 560. Network trained by GWR.

In the following, we conclude with STRAGEN. The initial node is labeled as blue and final node is labeled as red (left Figure, same for all networks).

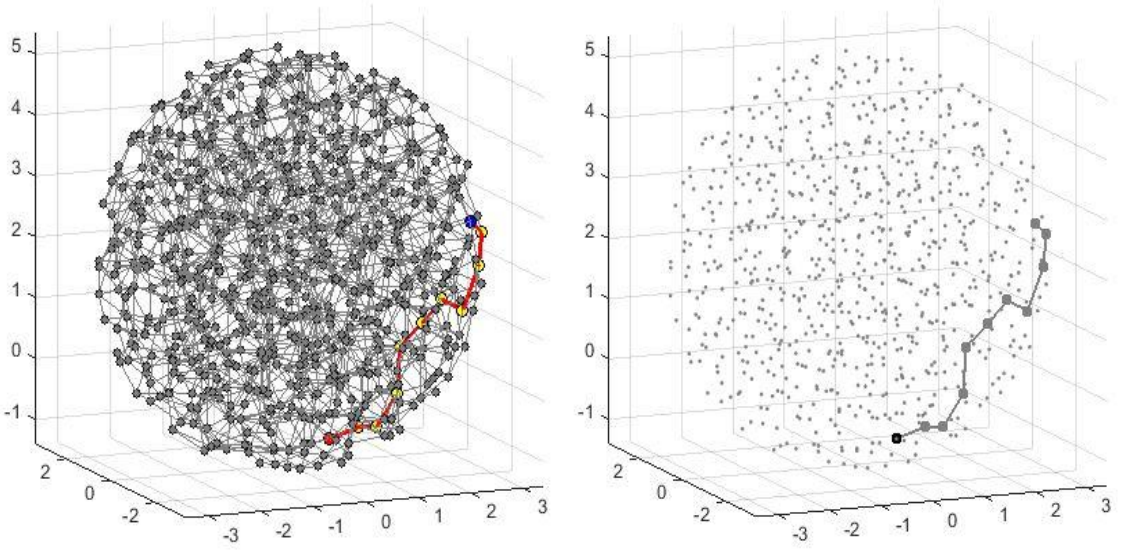


Figure 33: Trajectory of Puma 560. Network trained by STRAGEN.

7.1.3 Validation Error (Phase 3)

The models GNG, GWR and STRAGEN were trained with the same database to be compared. The database was presented to GNG and GWR using the original procedure, i.e., the samples presented randomly. STRAGEN uses Motor Babbling procedure as a strategy to present the samples of the base in an order that facilitates the creation of connections appropriate to the neighborhood criterion used. The models GNG and GWR models create their neighbors based

on information given by the Euclidean distance of the complete weight vectors. Therefore, these models do not distinguish what would be a suitable neighborhood for different criteria based on homogeneous information groups of the input vector.

The database was presented (t_{max} = 1... = 40000) for the three models (GNG, GWR and STRAGEN. The number of iterations for the validation phase was also $I = 40000$. The topology created by these models to represent the state space can be compared by Table 7.

	GNG	**GWR**	**STRAGEN On-Line**
Validation Error	0.31704	0.12954	0.10389
Number of Nodes	202	414	418
Number of connections	449	758	879

Table 7: Comparison Between GNG, GWR and STRAGEN

In every one of the reenactments done till now, STRAGEN demonstrated better validation error and distribution of nodes in state space.

STRAGEN creates new nodes in the exact position of the input pattern, while GNG and GWR create nodes based on the average of the first and second best matching node. As the weight vector is formed by heterogeneous information (positions) vital for the estimation of direct and inverse kinematics and dynamics, the calculation of the mean made by GNG and GWR for the whole vector causes the information stored that there to be distorted during training.

Figure 34 shows the evolution of the validation error for all three models according to the algorithm described in section 6.1.2. The error graph has been achieved after 40 000 iterations. At very beginning, GWR and GNG start with low error value compare to STRAGEN. Analyzing this figure, one can see that the advantage of STRAGEN is the rapid identification of the system at the beginning of its training, which makes it appropriate to be used almost

immediately at the start of the simulation. Over time, the GWR algorithm approaches STRAGEN precision, both being stabilized and with better result than the GNG algorithm.

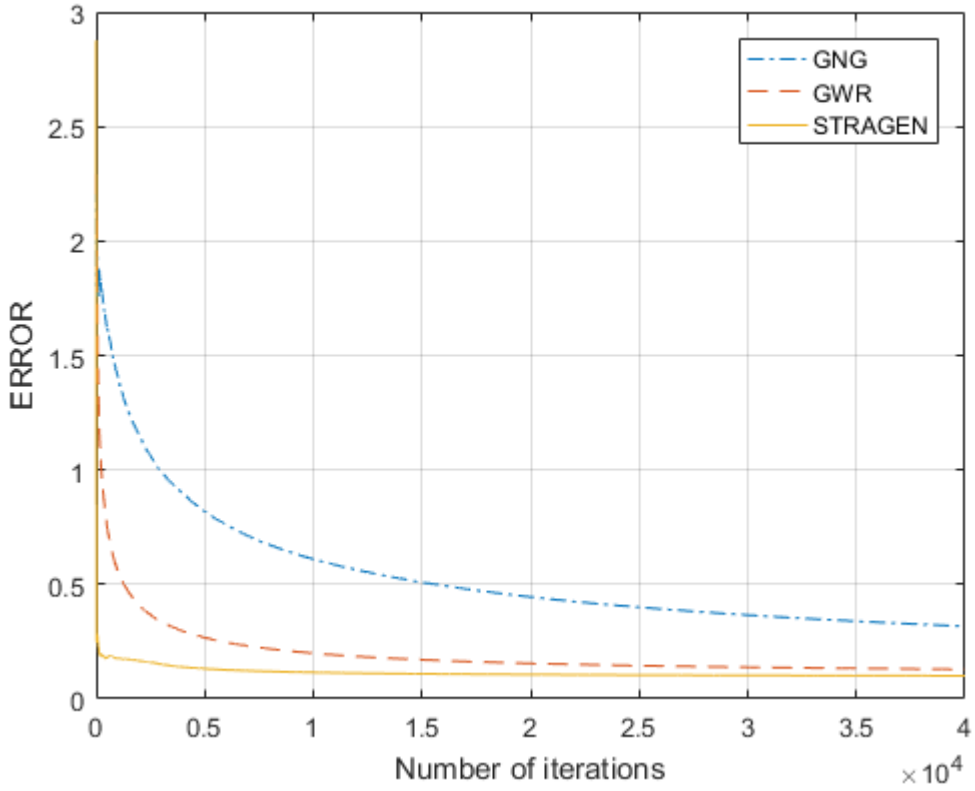


Figure 34: Validation Error of GNG, GWR and STRAGEN.

A table can be made by taking into account the number of iterations at random interval and corresponding validation error.

Iterations	GNG	GWR	STRAGEN
1	1.40157	0.74274	2.66229
2000	1.15011	0.41052	0.17573
10000	0.61188	0.19818	0.11913
40000	0.31704	0.12954	0.10389

Table 8: Validation Error for GNG, GWR and STRAGEN.

It ought to be recalled that in spite of the fact that the error displayed by GWR is close to STRAGEN. The algorithm accomplishes this convergence at the expense of the twisting of the information contained in the topological map by its learning process and insertion of nodes.

The growth of the number of connections in terms of number of iterations is given in Figure 35. It is imperative to note that STRAGEN, even after stabilizing the number of nodes needed to cover state space with good precision, continues to draw similarities from the exhibited patterns and to create (and also remove), at a lower rate, connections between nodes close by according to the established neighborhood criterion.

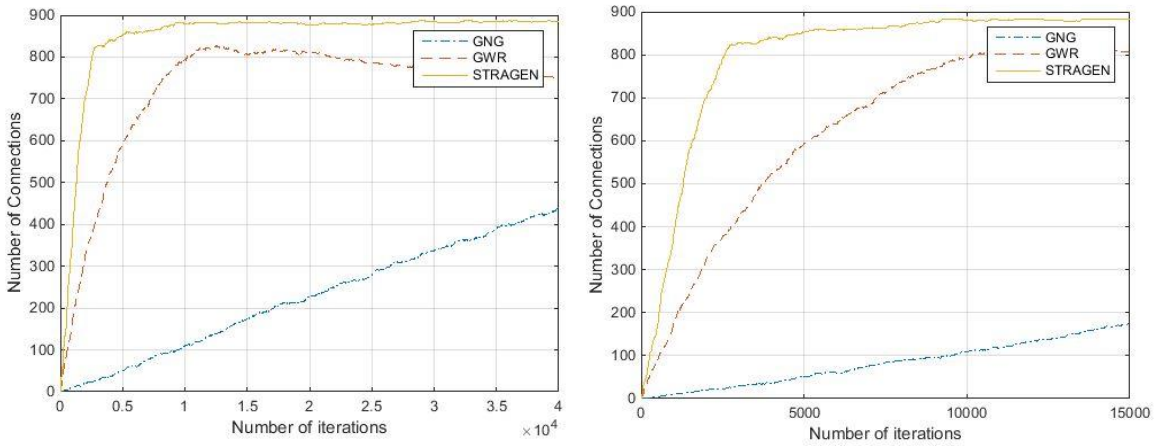


Figure 35: Network growth in terms of number of connections per iterations

The growth of the topological map for GNG, GWR and STRAGEN networks can be seen in Figure 36. GNG has direct development dependent on λ. In principle, STRAGEN also presents a straight growth. In any case, this growth is reliant on the emergence of new patterns not yet trained and is augmented by the Motor Babbling procedure. Once the nodes precisely presented the patterns, the growth of STRAGEN stabilizes immediately. The growth system of GWR topological map makes it a trade-off between creating a node or moving an existing node, and with this your growth curve is smooth. One of the challenges of GWR is to configure its parameter with the goal that the bend has greatest use for the area and database in question.

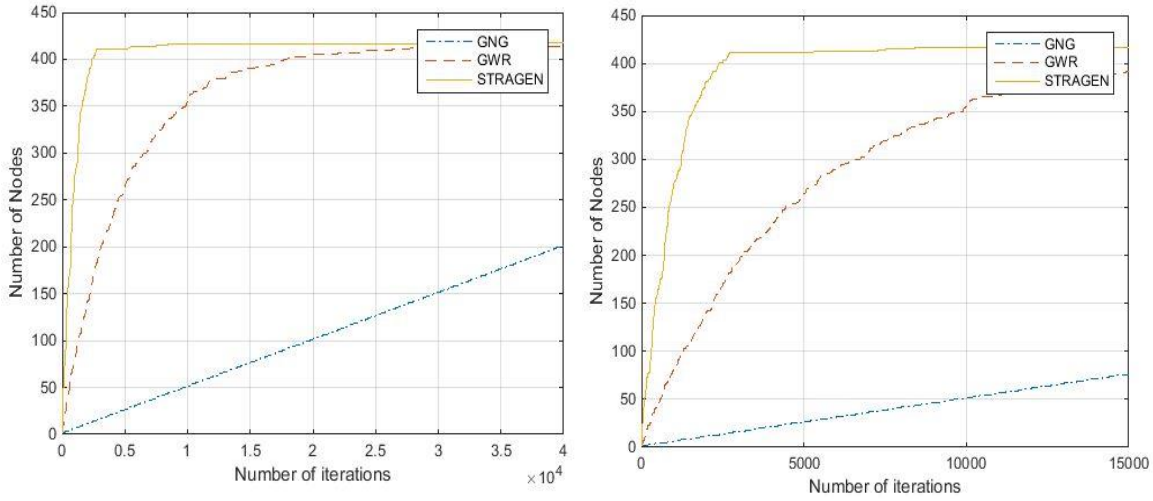


Figure 36: Network growth in terms of number of nodes per iterations.

7.1.3.1 Comparison Between the Graphs

By comparing the graphs of validation error and network growth, it can be seen that the error plot descrease as the network grows. Mathematically it can be said that error is inversely proportional to the number of nodes and connections between them with respect to to the number of iterations. It is also noted that GNG is slow as it increments the number of nodes after a predefined number of iterations, whilst GWR and STRAGEN grows faster than GNG as different node insertion criteria has been used in these algorithms. GWR needs to fulfill two conditions (activity threshold and firing threshold) for inserting node, while STRAGEN has just one condition (activity limit) in common to GWR. Activity limit is the exponential of the distance between winner and the input node. It means if the input is more near to the winning node, there are more chances to increment the network. Since random input is generated for GWR until the stopping criterion has been reached, there can be a chance of higher or lower distance between the winner and input. On the contrary, for STRAGEN input are generated by MB procedure and it inserts a new node exactly at the same position of input make the network agile. Following the formula of validation error (which is the distance between input and winner) is decresing. After 5000 iterations, STRAGEN grows to its maximum possible size and validation error reaches to its minimum value faster. For the remaining iterations it covers the remaining input states till the maximum number of iterations has been reached and graphs reached to a constant value.

7.2 Conclusions And Further Works

This work presented three artificial neural networks with dynamic topology to train the database of two-link robot and Puma 560 for the generation of state trajectories and make comparison among these networks. Various requirements were raised during this work to gain the outcome in different phases.

Data points simulation of the robots described in Chapter 2, is the very first task to proceed our work. After doing the simulation of robots according to DH convention, homogeneous data base had been achieved. GNG, GWR and STRAGEN use this data base to train the network by their algorithms. The development of the code was not an easy task. After several trails, we succeeded to implement the network for training the database.

During the training of the network, STRAGEN has higher growth in terms of adding number of nodes and number of connections described in section 7.1.3. At every comparison, STRAGEN shows a better response compare to other models. STRAGEN represents a strategy that can deal with heterogeneous (positions, angles and torques) information while other models have some difficulties in training.

STRAGEN is able to perform the generation of trajectories. It takes a process of generation of trajectories that follows up on the topological map learned. STRAGEN actualizes this procedure using established classical global search algorithm: energy diffusion (Zeller, 1997). Once a topological map has been created that represents the state space and contains information for navigation according to some established criteria, one can use any global search algorithms or local search techniques to generate the trajectory.

Taking advantage of a process inspired by Motor Babbling, STRAGEN utilizes this strategy to learn the connections of the input space. It is worth mentioning that MB was designed for multi-purpose learning between motor and visual systems that STRAGEN sums up its utilization for figuring out how to guide the mapping of nonlinear systems. The use of MB allows the STRAGEN exploit in creating more appropriate connections to represent the trajectories of the identified system. The MB also allows STRAGEN to learn systems, thus avoiding the mapping of unreachable or obstructed areas in the state space.

In addition, STRAGEN off-line eliminate unnecessary nodes and connections during pruning phase. But this is more time consuming to perform this phase as it has to be done for the same iteration number that has been used in training phase. While STRAGEN on-line removes unused node and connections during training phase.

Unlike the GCS, which attempts to preserve a k-simplex structure at the cost of creating "artificial" neighborhood relationships, and the SOM whose neighborhood relations are predefined, the STRAGEN creates these connections on demand and considers the topological density of the state space, thus resembling the GWR. However, taking care that the GWR is not to maintain the connections that represent the state transitions, not eliminating them due to a usage system or age. There is no need to eliminate dead or little-used connections by representing an unknown domain, whose reference to the transition between states is the similarity between certain groups of characteristics, not the time of use of the system.

Further work will be done in hardware platform. The objective is to program the robot according to the algorithms that have been simulated in Chapter 7. It would be more economical and time redundant system to find the trajectories of the robot by artificial neural networks instead of providing command to the robot.

Bibliographic References

Araujo, Aluzio F. R. and Barreto, Guilherne de A., **"Context in Temporal Sequence Processing: A Self -organizing Approach and its Application to Robotics''**, IEEE, 2002.

Araujo, Aluzio F. R. and Barreto, Guilherne de A., **"Competitive and Temporal Hebbian Learning for Production of Robot Trajectories''**, University de Sao Paolo, 1998.

Araujo, Aluzio F. R. and Barreto, Guilherne de A. and Ritter, Helge J., **"Self-Organizing Feature Maps for Modelling and Control of Robotic Manipulator''**, Journal of Intelligent and Robotic System, Kluwer Academic Publisher, Netherlands, 2003.

Araujo, Aluzio F. R. and Barreto, Guilherne de A. and Ritter, Helge J., **"Identification and Control of Dynamic System Using the Self-Organizing Map''**, IEEE, 2004.

Benante, Ruben C. and Araujo, Aluizio F. R. and Ludermir, Teresa B., **"Automatização na Escolha de Parâmetros para o Modelo Incremental GNG Usando Algoritmos Genéticos''**, University Federal de Pernambuco, 2004.

Benante, Ruben C. and Araujo, Aluizio F. R., **"Self-organizing Maps to Generate State Trajectories of Manipulators''**, 2007a.

Benante, Ruben C. and Pedro, Leonardo M. and Massaro, Leandro C. and Belini, Valdinei L. and Araujo, Aluizio F. R. and Caurin, Glauco A. P. Member IEEE, **''A Self-Organizing State Trajectory Planner applied to an Anthropomorphic Robot Hand''**, 2007b.

Benante, Ruben Carlo**, ''Geração de Trajetórias de Estados por Mapas Auto-organizáveis com Topologia Dinâmica''**, TESE DE DOUTORADO, Universidade Federal de Pernambuco.

Craig, John J., **"Introduction to Robotics – Mechanics and Control''**, Third Edition, Pearson Educational, Inc., 2005.

Dijkstra, E. W., **"A Note on Two Problems in Connexion with Graphs''**, Numerische Mathematik 1, 269-271, 1959.

Fritzke, Bernd, **"Growing cell structures - a self-organizing network for unsupervised and supervised networks''**, Neural Networks 7(9), 1441–1460, 1994.

Fritzke, Bernd, **"A growing Neural Gas Network Learns Topologies''**, Ruhr-University Bochum, Germany, 1996.

Fritzke, Bernd, **''Unsupervised ontogenic networks''**, 1997.

Holmstrom, Jim Uppsala University, **"Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG''**.

J. Si, S. Lin, **"Dynamic Topology Representing Networks''**, Neural Networks 13(6): 617-27, 2000.

Lee, Byung-Joo and Kyoichi, Sugimoto, **"Construction of a Position Maintained Trajectory''**.

Martinetz, T. M., Berkovich, S. G. & Schulten, K. J., **"A 'neural-gas' network learns topologies, *in* T. Kohonen, K. Mäkisara, O. Simula & J. Kangas, eds, 'Artificial Neural Networks''**, North-Holland, Amsterdan, pp. 397–402, 1991

Martinetz, T. M. and Berkovich, S. G. and Schulten, K. J., **"Neural-gas network for vector quantization and its application to time-series prediction''**, *IEEE Transactions on Neural Networks* 4(4), 558–569, 1993.

Marsland, Stephen and Shapiro, Jonathan and Nehmzow, Ulrich, **"A self-organising network that grows when required'',** Neural Networks 15, 1041-1058, 2002.

Siciliano, Bruno and Sciavicco, Lorenzo and Villani, Luigi and Oriolo, Giuseppe, **"Robotics Modelling, Planning and Control''**, Springer-Verlag London Limited, 2009

Spong, Mark W. and Hutchinson, Seth and Vidyasagar, M., **"Robot Dynamics and Control''**, 2004.

Villmann, Thomas and Der, Ralf and Herrmann, Michael and Martinetz, Thomas M., **''Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement''**, IEEE Transactions On Neural Networks, Vol. 8, No. 2, 1997.

World Economic Forum, **"Technology and Innovation for the Future of Production**", Switzerland, 2017.

Zeller, M. and Schulten, K., **"Vision-Based Robot Motion Planning Using A Topology Representing Neural Network''**, 1997.

Cite this thesis as:

Akhtar M.N.; Reza, D.S.A.A., Dynamic Neural Networks to Generate Robotics Trajectories. Recife, 2018. 96Pg. Thesis (Projecto de final de Curso): Master of Science Degree in Mechatronic Engineering. Universidade de Pernambuco & Politecnico di Torino.

BibTeX:

```
@MASTERTHESIS {
            author = ‘‘ Akhtar, M.N.; Reza, D.S.A.A.’’
            title = ‘‘Dynamic Neural Networks to Generate Robotics Trajectories’’
            school = ‘‘Universidade de Pernambuco & Politecnico di Torino’’
            year = ‘‘2018’’
            address = ‘‘Recife-PE, Brazil’’
            month =‘‘December’’
}
```