الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Département : Génie électrique Et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme d'
Ingénieur d'Etat

Spécialité
Automatique et Informatique Industrielle

- Thème -

Visual Place Recognition Using Hierarchical Temporal Memories Theory

Réalisé par
LAHOUAL Yassine

Les membres de Jury:

M. Abbadi belkacem	Président
Mme. Rebai karima	Promotrice
Mme. Bouchama samira	Examinatrice
Mme. Chouaf seloua	Examinatrice

Algier, le 15/09/2022
19 صفر من 1444 هـ

Année universitaire 2021 –2022

Dedications

I dedicate this work of mine to whoever wished me good luck, truly from his heart.

Acknowledgement

الحمد الله رب العالمين الذي اعانني على اتمام هذا البحث وصلاة وسلام على محمد خير خلق الله.

And of course, my beloved friends who gave me love & passion, the ones I had so beautiful time with, all of them gets a special thanks.

Then, I would like to express my sincere gratitude to my supervisor Dr. REBAI for her patience and support. She consistently checked me through the learning process of this dissertation. Next, I would like to thank most of the people in my university "National Higher School of Technology" the ones who helped me and wished me good luck.

ياســين

Table of Contents

Chapitre I : Visual Place Recognition....6
I.1 Introduction....7
I.2 What Is a Place?....7
I.3 Simultaneous Localization and Mapping (SLAM)....8
I.3.1 Mapping....8
I.3.2 Localization....9
I.4 Visual Place Recognition (VPR)....9
I.5 What Are the Major Challenges?....11
I.5.1 Appearance Change and Perceptual Aliasing....12
I.5.2 Viewpoint Changing....12
I.5.3 Scalability....13
I.6 How are Places Handled?....13
I.6.1 Place Modeling: Describing Place....13
I.6.2 Place Mapping: Storing Data....18
I.6.3 Place Recognition: Belief Generation....20
I.7 Conclusion....22
Chapitre II : Hierarchical Temporal Memory....23
II.1 Introduction....24
II.2 Hierarchical Temporal Memory Overview....24
II.2.1 Region....25
II.2.2 Hierarchy....26
II.2.3 Sparse Distributed Representations....27
II.2.4 The role of time....28
II.3 Sparse Distributed Representation Modeling....29
II.3.1 Capacity....30
II.3.2 Tolerance to Noise....30
II.3.3 Encoding Data....31
II.4 Hierarchical temporal memory modeling....31
II.4.1 HTM Neuron Model....31

II.4.2 Learn High-Order Sequence 32
II.4.3 Spatial and temporal concepts 34
II.5 Conclusion: 39
Chapitre III : An HTM-based approach for VPR 40
III.1 Introduction 41
III.2 General overview of the proposed approach 42
III.3 Data Handling 43
III.4 HTM Algorithmic Methods 45
III.4.1 Spatial Pooler Algorithm 45
III.4.2 Temporal Pooler Algorithm 49
III.5 Scoring Recognitions 53
III.5.1 Computing the recognition score 54
III.5.2 Computing the recognition likelihood 54
III.6 Conclusion 56
Chapitre IV : HTM-VPR Implementation and Evaluation 57
IV.1 Introduction 58
IV.2 Data choosing and treating 59
IV.2.1 Extracting frames from the videos 60
IV.2.2 Combining frames 60
IV.2.3 Extracting features from frames 60
IV.2.4 Binarization 61
IV.3 Running the ANN of HTM 62
IV.4 Scoring Recognitions 64
IV.5 Evaluation of the proposed approach 67
IV.6 Conclusion 69

List of Figures

Figure I.1 : An occupancy grid map; 9
Figure I.2 : VPR system's input is query, returns comparable database images. 10
Figure I.3 : Comparing the most similar between query images and the input image. 11
Figure I.4 : Illustration of the seasonal environment changes. 12
Figure I.5 : Matching key-points of the same place seeing from different orientation. 14
Figure I.6 : Extraction of the HOG feature descriptor. 15
Figure I.7 : Subdivided image into grids, each is separately treated. 16
Figure I.8 : Segmentation of the view. 17
Figure I.9 : Example depth-map obtained using DTAM method. 18
Figure I.10 : Camera trajectory with LSD-SLAM, showing keyframe depth-maps. 18
Figure II.1 : Picture of the part of the human brain that makes up the neocortex. 25
Figure II.2 : A section of an HTM region. 26
Figure II.3 : Neocortex spread flat made up of columns, image from [50]. 26
Figure II.4 : Architecture of the three layers-HTM, image from [51]. 27
Figure II.5 : Illustration of coupled HTM networks. 27
Figure II.6 : An HTM region that shows sparse distributed active cells. 28
Figure II.7 : Region's input as vector. 29
Figure II.8 : SDR forms before and after learning. 29
Figure II.9 : Common active cells at the same time means there is some similarity. 30
Figure II.10 : Modeled neuron has three input, the out put is its state. 32
Figure II.11 : Presentation of the cortex's six cellular layers model. 32
Figure II.12 : Learning shared sub-sequences (BC) in high-order sequences. 34
Figure II.13 Proximal dendrite model [51]. 35
Figure II.14 : Example of using spatial pooling. 36
Figure II.15 : Distal dendrite model [51]. 37
Figure II.16 : Example of using temporal memory- Phase 1. 38
Figure II.17 : Example of using temporal memory- Phase 2. 39
Figure III.1 : shows multiple phases of the general architecture 42
Figure III.2 : Same position different appearance, from the treated Data. 43
Figure III.3 : Matching features of two images using SURF descriptor. 44
Figure III.4 : Spatial Pooler (phase 1). 46
Figure III.5 : Spatial Pooler (phase 2). 47
Figure III.6 : Spatial Pooler (phase 3). 48
Figure III.7 : Temporal Pooler (phase 1). 50
Figure III.8 : Temporal Pooler (phase 2). 51
Figure III.9 : General architecture of VPR using HTM. 53
Figure IV.1 : Organization of the whole developed functions. 58
Figure IV.2 : The path followed by the car with multiple loops. 59
Figure IV.3 : The above picture shows the frame between the combined. 60

Figure IV.4 : The above picture shows what the **MATLAB Code IV—4** plots. 61
Figure IV.5 : Illustration of what the temporal pooler plots. 63
Figure IV.6 : Recognition Scores of each frame from **MATLAB Code IV—7**. 64
Figure IV.7 : Recognition Scores of each frame from **MATLAB Code IV—7**. 65
Figure IV.8 : Zoom in **Figure IV.7** from **MATLAB Code IV—7**. 65
Figure IV.9 : Recognition Likelihood of sequence from **MATLAB Code IV—7**. 65
Figure IV.10 : A screenshot of the video from **MATLAB Code IV—9**. 66
Figure IV.11 : Precision-recall curve of the HTM-based VPR. 68

List of Tables and Codes

List of tables:

Table I.1 : Mapping Frameworks for VPR. 19
Table IV.1 : the abbreviations used for system evaluation. 67

List of codes:

MATLAB Code III—1 : Matching two frames using SURF from the main code. 71
MATLAB Code IV—2 : Extracting frames from videos from the main code. 71
MATLAB Code IV—3 : Combining both frames from the main code. 71
MATLAB Code IV—4 : Extracting local features (SURF) from the main. 72
MATLAB Code IV—5 : Between the combined from the main code. 72
MATLAB Code IV—6 : Features Binarization. 72
MATLAB Code IV—7 : Including HTM modules in the main script. 73
MATLAB Code IV—8 : Detect Recognition likelihood from the main. 74
MATLAB Code IV—9 : Sequential Recognition function. 74
MATLAB Code IV—10 : Drawing a moving car from the main. 75
MATLAB Code IV—11 : Performance Evaluation code. 75

Nomenclature

AI	Artificial Intelligence
AV	Autonomous Vehicles
BoW	Bag of Word
CNN	Convolutional Neural Network
CV	Computer Vision
DTAM	Dense Tracking and Mapping
FAST	Features from Accelerated Segment Test
FoV	Field-of-View
GPS	Global Positioning System
HCT	Hull Census Transform
HMMs	Hidden Markov Models
HOG	Histogram of Oriented Gradient
HOSM	High-Order Sequence Memory
HTM	Hierarchical Temporal Memory
LBP	Local Binary Pattern
LCD	Loop Closure Detection
PA	Perceptual Aliasing
RGB-D	Red Green Blue-Depth
SDR	Sparse Distributed Representation
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SM	Sequence Memory
SP	Spatial Pooler
SURF	Speeded Up Robust Features
TP	Temporal Pooler
VPR	Visual Place Recognition

General introduction

Context

The improvement of more reliable long-term localization and perception systems has made self-driving cars more adept at navigating in unexpected and chaotic environments. Finding loops in areas that have been revisited helps long-term localization guarantee continuous mapping and bounded error over time. In long-term localization systems, revisited locations are found using techniques like place recognition and loop closure. Place recognition, specifically, is a perceptual method that identifies previously visited locations based on visual, structural, or semantic information.

Problematic

Visual Place Recognition for mobile robot global re-localization is a well-studied problem, where contrastive learning-based representation training methods have led to state-of-the-art performance. However, these methods are mostly designed for treating single image-based in Visual Place Recognition, where sequential information, which is common in robotics, is only used as a post-processing step for filtering single image match scores. Despite treating sequence of images, place recognition still faces some challenges, namely places with similar appearances; places that change in appearance over time; places that are perceived from different viewpoints; and scalability of the proposed approaches in large environments.

Objective

The ability to recognize and predict temporal sequences of sensory inputs is vital for survival in natural environments. Based on many known properties of cortical neurons, hierarchical temporal memory recently has been proposed as a theoretical framework for sequence learning in the cortex. This dissertation analyzes the properties of hierarchical temporal memory of treating sequence memory and applying it to sequence learning and prediction problems for Visual Place Recognition streaming data, due to hierarchical temporal memory capacity of continuously learning a large number of variable order temporal sequences.

Organization of this dissertation

This dissertation is organized as follows:

Chapter I: Role of Visual Place Recognition for vehicle and some related works.

Chapter II: The theory of Hierarchical Temporal Memory.

Chapter III: HTM-based approach for VPR.

Chapter IV: Coding, dataset choosing, testing, performance evaluation.

Chapitre I : Visual Place Recognition

I.1 Introduction

In the last decade, a lot of studies and improvements have been made in Autonomous Vehicles (AV) and robotic vision, because of that, intelligent vehicles are becoming more capable of navigating in complex and wide environment. For a better autonomous operation, a strong navigation system is needed. The key to achieve a higher level of autonomy is related to the adopted solution for Simultaneous Localization and Mapping (SLAM). The purpose of SLAM is that the vehicle needs to build the environment's map and localize itself within it. of its surroundings then correct mis-localization within unknown environment. Visual Place Recognition (VPR) is the heart component of SLAM and its often named as closed loop detection (LCD); a system which can decide whether the place has previously been visited or not. This chapter discusses the main topic of this dissertation; visual place recognition in general and its role for Autonomous Vehicles (AV). It starts with the definition of place in robotics context in Section (I.2), followed up with the understanding of SLAM in Section (I.3). Section (I.4) gives a brief look into Visual Place Recognition. Section (I.5) shows the major challenges of VPR. Then Section (I.6) highlights into the depth of VPR and its divisions which are how a place is modeled (described)? how a place is mapped (stored)? and how the place can be recognized? Including some related work that surpassed the main problems that are mentioned in Section (I.5). In the end the conclusion is given in Section (I.7).

I.2 What Is a Place?

Actually, there is no unique and universally accepted definition of a 'Place', therefore, Multiple definitions are cited bellow:

Firstly, the definition of a place depends on the navigation context, and may either be considered as:

- A precise/specific position (As a zero-dimensional point, a location describes a component of the environment).
- A larger area (The abstraction of a region can alternatively be characterized as a place), where a region could be:
  - A two-dimensional subset of the environment. For example, in some cases, a room in a building may qualify as a single place, while in others, it may contain multiple places.
  - A three-dimensional area based on the environment or robot's requirements.

A place, unlike a robot pose, does not have an orientation, therefore pose invariance, which ensures recognition independent of the robot's orientation within the place, is a continuing difficulty in place recognition.

Secondly, in terms of appearance, a place can be defined as a location that is distinct from others nearby based on sensory information known as a place signature or place description.

A topological place is described as having a specific appearance configuration, and the physical borders of a place are defined as the point where the appearance dramatically changes, referred to as a "gateway". This qualitative concept of topological places as regions that are visually homogeneous needs to be quantified.

Places are segments of the physical world that can have any given scale at the limit, a place may represent a single location to an entire region of discrete locations [1]. The segments physical boundaries can be defined resorting to different segmentation criteria: time step, traveled distance, or appearance. In particular, the appearance criterion is widely used in place recognition. In such a case, a new place is created whenever the appearance of the current location (determined from the sensor measurements) differs significantly from locations that were previously observed. Similarly, Korrapati et al. [2] used image sequencing partitioning techniques to group visually similar images together as topological graph nodes. The addition of semantic data to maps can improve planning and navigation tasks [3] and requires place recognition to be linked with other recognition and classification tasks, especially scene classification and object recognition.

I.3 Simultaneous Localization and Mapping (SLAM)

To achieve robotic localization, the system requires a map to recognize a previously visited location. Simultaneous Localization and Mapping is the process of creating such maps of the surrounding environment for the purpose of localization (SLAM). SLAM has been a hot topic in the robotics and computer vision (CV) sectors for the past couple decades. It is made up of the following elements:

I.3.1 Mapping

It's the process of internally producing and storing a representation of the outside world. The sort of environment and sensory information used to create world maps might differ. As seen in **Figure I.1**, occupancy grid maps divide the world into uniformly spaced discrete sections, each section is occupied or unoccupied. For occupancy grid maps, few assumptions are made; each cell is either free or occupied, and each cell is independent of the others. Each region has a probability to be occupied by an obstacle based on a cumulative knowledge and statuses. A 0.5 probability to be an obstacle have been assigned to those regions that are neither occupied nor explored. Such occupancy-based grid maps are ideal for crowded indoor situations since they can be easily created using laser or sonar-based data but demand a lot of memory. These maps are commonly used in obstacle avoidance algorithms, such as potential field method. Feature maps, on the other hand, are compact, sparse, and consist of discrete landmarks with their coordinates. With information about landmarks collected from the vision sensors, they are ideal for use in outside contexts. Pose graphs are a type of map that depicts the robot's trajectory graphically. The nodes orientation describes the robot’s pose (position and orientation). Edges describe the spatial connection between the poses, can be used for loop closure[1].

[1] Loop closure detection: is an important problem for any SLAM system and it is the process of detecting whether a robot has returned to a previously visited location.

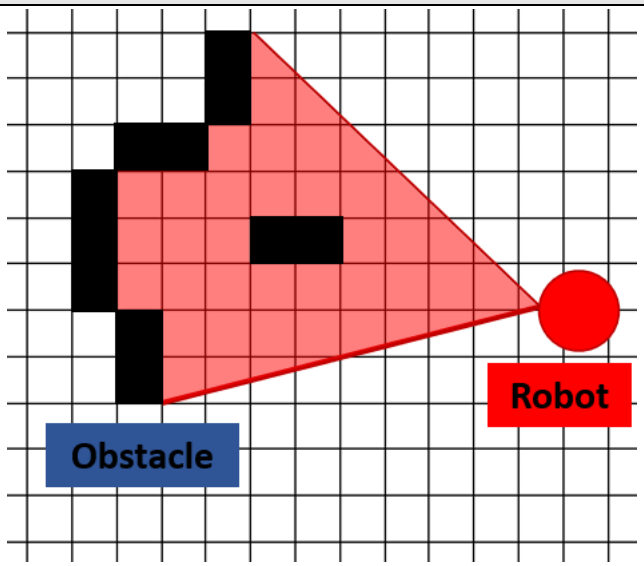


Figure I.1 : An occupancy grid map;
The world is subdivided into discrete regions. The light red represents the observed regions by the robot, inside that the black grids are obstacles. The white grids are not observed to the robot.

I.3.2 Localization

The method of tracking the position of a robot inside a map using different types of sensors is known as localization. This module takes the acquired data by sensors data and generates a belief about where robot is on the map. Generally, visual sensors are the most used sensors known by the rich provided information. The belief is used by the algorithm to determine if the current location has been viewed before or it is new. When two places look alike, it's typically because they're in the same area. This assumption, however, does not hold true when there are substantial weather and lighting changes that affect the visual measurements, which could cause the robot to incorrectly localize within the map.

Localization can take place on a local or global scale. The robot in global localization assumes no prior information and is free to move anywhere on the map. The kidnapped robot problem arises from such re-localization of the robot utilizing visual information and a map. Global localization can handle the kidnapped robot problem but still difficult to achieve in environments when the places are fairly similar. However, the robot usually has some past knowledge of its current position and map, which is referred to as local localization. Prior knowledge is derived from the robot's previous states. Local localization aims at compensating odometric errors during robot navigation. They necessitate that the robot's original location is known, and they are often unable to recover if the robot's position is lost (kidnapped).

I.4 Visual Place Recognition (VPR)

An important question posed that describes the main purpose of VPR which is “given an image of a place, can a human, animal, or robot decide whether or not this image is of a place it has already seen?” [1].

Visual place recognition is one of the most important parts of visual Autonomous Vehicles, and can be defined as the task of identifying the same locations visited despite significant

changes in appearance and viewpoint at different periods of time. Visual place identification is an important part of Spatial Artificial Intelligence (AI), as it allows robotic platforms and intelligent augmentation platforms like augmented reality devices to sense, recognize and interpret the physical environment. It has gained lot of attraction in recent times due to its many advantages over other sensor modalities such as low cost, low power consumption, easy setup and ability to work even in areas with no Global Positioning System signal (In an outdoor GPS offers a cheap solution for localization). Also, the potential applications of place recognition in the field of loop closures, localization, etc., making it a critical challenge to tackle under the umbrella of Simultaneous Localization and Mapping (SLAM). Place recognition, in particular, is a reliable way for removing accumulated mistakes in SLAM because revisiting sites allow loop-closure in built maps, or geo-referenced pose estimates when performed relative to previously collected image sets in a fixed map. Therefore, the task of VPR is to find associations between one or multiple query images with a database of images of known places, known as image retrieval task. So, it can detect loop-closure during robot's motion and contribute in the improvement of SLAM quality. Due to its importance for long-term operation, a particularly active research field is recognizing places despite severe changes of the environmental conditions induced by day-night cycles, changing weather, or seasons. At its core, VPR is an image retrieval problem and many developments and results from the computer vision community significantly contributed to the state of the art. However, the application in the context of mobile robotics frames the image retrieval task and typically provides additional structure and information.

It is an active field of research and many different approaches have been proposed and evaluated in many different experiments.

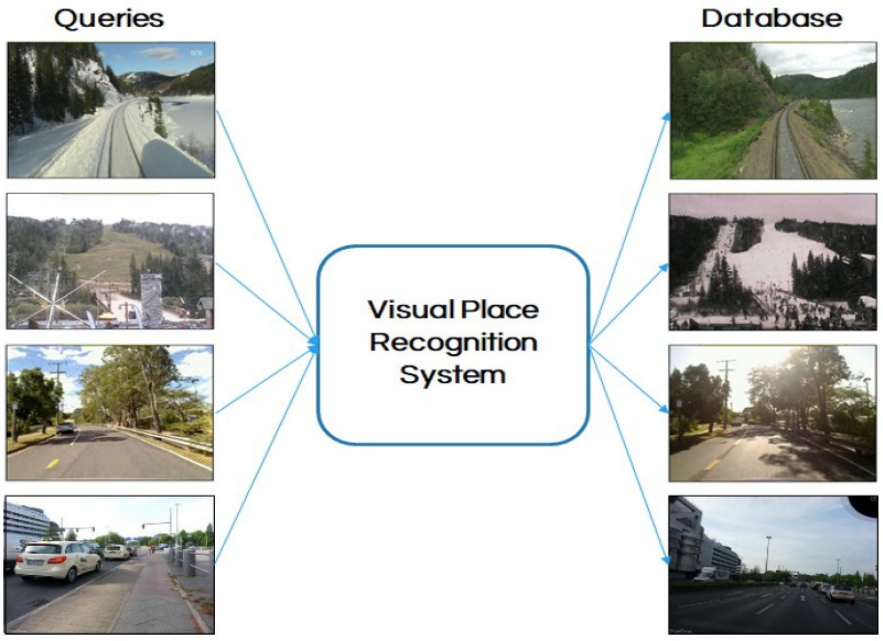


Figure I.2 : VPR system's input is query, returns comparable database images.

Captured pictures of places within an environment are saved in a database as part of a visual place recognition system. As a result, the query is compared to the stored database at runtime using the closest database picture, which is regarded a currently visible place (location), as

illustrated in **Figure I.2**. Recent advances in computer vision have improved the performance of localization systems that employ a variety of robust feature detectors to efficiently interpret visual information.

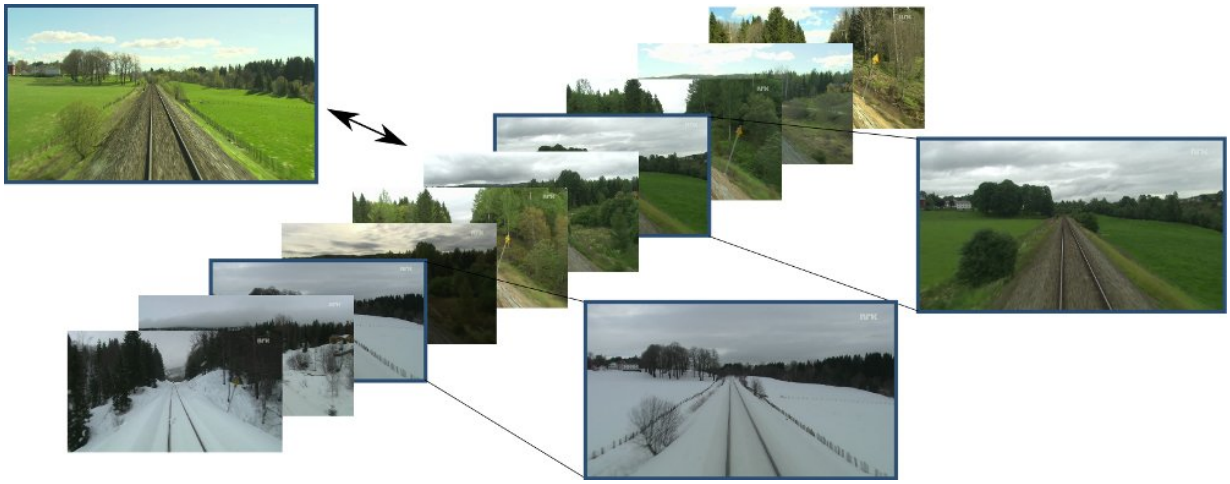


Figure I.3 : Comparing the most similar between query images and the input image.

The image in the top-left corner of the **Figure I.3** is compared to the images from the map. The task is to find the two corresponding images of the same location (on the right side), despite the fact that they are subject to severe appearance changes due to the different seasons.

I.5 What Are the Major Challenges?

The main goal of visual place recognition is to match two images so that the system can determine if it has seen that location previously. And it is notably that place recognition approaches are becoming more and more sophisticated as the environment and operation conditions become more similar to real-world situations. The present state-of-the-art in place recognition techniques, for example, can operate over large regions in real-world conditions situations with unique performances. It has been highlighted as important issue to be able to locate in and create maps of dynamic environments [4]. Despite these achievements, place recognition task poses a number of challenges due to three major reasons. This section discusses over these challenges which they occur when:

- Two distinct places have similar appearances (also known as perceptual aliasing), and the same place with appearance changes over time due to light conditions like day-night variation, weather, seasonal or structural changes as shown in **Figure I.4** Images taken from the Oxford Robotcar, and Norland dataset.



Figure I.4 : Illustration of the seasonal environment changes.

- The same places are observed from different positions and/or with different viewpoints.
- Scalability of the proposed approaches in large environments.

These problems must be overcome, to allow effective robot's localization and robust place recognition and as a result a perfect long-term localization.

I.5.1 Appearance Change and Perceptual Aliasing

Many studies have focused on appearance-changing and perceptual aliasing in particular. Because autonomous robots move for long periods of time, their perception systems must deal with changing that may happen over time, such as changing weather or seasonal conditions, or structural modifications. While the appearance change issue occurs when the appearance of the same place varies over time, the perceptual aliasing occurs when distinct places appear to be the same. These factors have an important impact on place recognition performance and consequently on loop closure detection.

Various research works have addressed these issues. The belief generation perspective and Sequence-based matching techniques [5] [6] are emphasized as extremely successful in these conditions. The task of aligning a pair of template and query sequences is known as sequence matching, and it may be done via minimal cost flow [7], or probabilistically using Hidden Markov models [8] or Conditional Random Fields [9]. Place modeling approaches can also handle this problem: extracting condition-invariant features [10], extracting for example features using convolutional neural network and other architectures of Deep learning [11]. Descriptor normalization [12], or unsupervised approaches such as dimensionality reduction [13], change removal [14] and other methods are also proposed.

I.5.2 Viewpoint Changing

Place recognition is especially difficult when revisiting a location from another position [15]. This is especially true for methods that rely on sensors with a limited field-of-view (FoV) or lack geometrical sensing capabilities. These sensors only catch a fraction of the

environment while visiting a location, and upon revisiting from a different angle or position, the scene may seem differently or even more features may be felt, resulting in an entirely different place model.

In some studies, the use of panoramic cameras [16] or 3DLiDAR [17] have been proposed and tested, this reduces the effect of how future visitors interpret orienting places. As a result, using sensors and methods that aren't affected by orientation (also known as viewpoint-invariant) makes location recognition more reliable and robust.

In other works, visual-based techniques have turned to semantic-based features to solve these issues [18].

I.5.3 Scalability

Scalability is another important aspect of place recognition [19], [20]. As self-driving cars operate in wider regions, more locations are visited and maps grow in size, increasing processing load and lowering inference efficiency. Therefore, some approaches like: efficient indexing [21], hierarchical searching [22], hashing and scalar quantization [20], Hidden Markov Models (HMMs) [19], or learning regularly repeating visual patterns [23] are all methods for increasing inference efficiency.

I.6 How are Places Handled?

Place recognition is the process of recognizing places within a global map, using clues from the surrounding environments. In general, it is divided into three modules: place modeling, place mapping, and place recognition also known as belief generation.

I.6.1 Place Modeling: Describing Place

Place modeling is the way of mapping data from a sensor space to a descriptor space, for modeling the surrounding environment, sensors such as 3D LiDARs, RADARs and cameras are highly needed for data acquisition. A number of place recognition approaches have been developed, each with its own method of collecting information. In VPR methods, some ones select parts of the image that are highly notable (Local Feature Descriptors), while others process the entire scene description (Global Feature Descriptors) or a combination of the types of descriptors. Similarly, there are techniques that make use of three-dimensional data and state-of-the-art neural network-based techniques. In this method, an image is transferred into the input layer(s), and the responses are used as feature descriptors.

I.6.1.1 Local Feature Descriptors

Local feature techniques evaluate visual data (images) first, then identify meaningful key-points at different spatial positions. The detection is based on different pixel patterns in combination with a description of the spatial position. For retrieving the final multi-dimensional feature vector, concatenation of the neighbor pixels is performed within each position.



Figure I.5 : Matching key-points of the same place seeing from different orientation.

Local features are usually unaffected by affine transformations like image scaling and rotation. Consequently, conventional local feature descriptors can be used in regions with similar environments and situations.

Local descriptors are usually robust to viewpoint changes because local feature extraction is a two-stage process (detection then description), it is typical to combine different techniques for each. For example, using the detection technique FAST to find key-points in the image, then use Scale Invariant Feature Transform SIFT to describe the key-points.

Local Binary Pattern (LBP) is an effective texture pattern descriptor to describe the key-points texture patterns of a visual place. It is commonly utilized in image processing applications. The research community can now improve place description and recognition thanks to the recent development of local binary feature detectors. These detectors are highly efficient and insensitive to monotonic gray scale changes.

Each image may have hundreds of key-points, making it inefficient to compare image features directly. By quantifying local features into a vocabulary that can be compared using text retrieval techniques, the bag-of-words (BoW) approach improves efficiency. A feature space, such as SIFT or SURF descriptors, is partitioned into a finite number of visual words using the bag-of-words approach.

The geometric structure of the place gets ignored by the initial bag-of-words model. Consequently, the resulting place description is pose-invariant, which means the place can be recognized regardless of the robot's location within it. However, adding geometric information to the process has been found to improve place matching robustness, resulting in greater identification performance, especially under changing settings.

I.6.1.2 Global feature Descriptors

The methods discussed in the preceding section focused on sub-regions/parts of the image. Such feature extraction approaches are highly efficient under viewpoint variance and partial occlusions, but they do not generally consider the full picture or the structure of the place appearance. In comparison, this section examines the global feature detectors which process and characterize the full image for place recognition.

Global features describe the image as a whole to generalize the entire object features include contour representations, shape descriptors. Global descriptors are widely used for place recognition for example Histogram of Oriented Gradient (HOG) and color histograms.

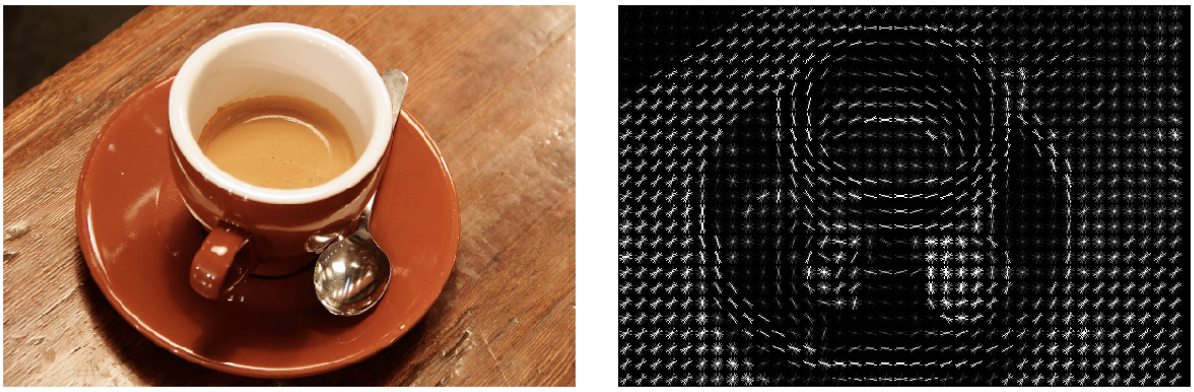


Figure I.6 : Extraction of the HOG feature descriptor.

By predefining the key-points in the image, global descriptors can be generated from local feature descriptors. For example, Badino et al in [24] to perform localization, they used whole-image descriptors based on SURF features known as WI-SURF.

A well-known global descriptor GIST which stands for “a low dimensional representation of the scene, which does not require any form of segmentation”. And it has been used for place recognition in [13].

Furthermore approaches, Grid-based picture partitioning illustrated by **Figure I.7** is followed by the concatenation of each grid feature to obtain the final descriptor. For large-scale place recognition, global features are generally easy to establish and scalable. These approaches are unaffected by conditional changes, although they are less effective when the viewpoint changes.



Figure I.7 : Subdivided image into grids, each is separately treated.

I.6.1.3 Combining Local and Global Techniques

Local and global descriptors both have various advantages and disadvantages. For early visual place recognition techniques, both descriptor techniques have shown an increasing amount of integration.

Local feature descriptors can be recombined to introduce additional locations that the robot hasn't seen before, rather than only defining a place in terms of a past robot position. For example, Lynen et al. in [25] created a 2-D space of descriptor votes, with high vote density locations representing loop closure candidates. Local features can be mixed with metric data to provide metric localization corrections.

Global descriptors are less flexible than local descriptor approaches, because whole-image descriptor comparison methods imply that the camera viewpoint remains consistent. Combining a bag-of-words technique with a Gist descriptor on image segments can help minimize this problem to some extent.

The research community has developed numerous frameworks for location recognition based on a mix of feature descriptors, prioritizing local and global feature individual advantages. For example, Hull Census Transform (HCT) is a combination of local and global descriptor-based techniques proposed in [26], it gets filtered by the SURF features, which are made up of convex hulls, and then estimated their relative magnitude, resulting in a set of binary vectors efficient so it can detect scenes. A similar approach in [27] combined local features edges and color histograms, Harris Corner detector is used, followed up with SIFT description.

Global descriptors depend on pose more than local descriptors, though still local descriptors get outperformed by global descriptors when it comes to place recognition in changing conditions like lighting changes. Because large segments display some of the condition invariance of whole images and small segments display the pose invariance of local features. Therefore, it is better to use global descriptors on image segments rather than whole images.

Sűnderhauf et al. in [28], using edge boxes as segments coupled with mid-layers convolutional neural network (CNN) feature [29], to detect and extract landmarks as shown in **Figure I.8**.

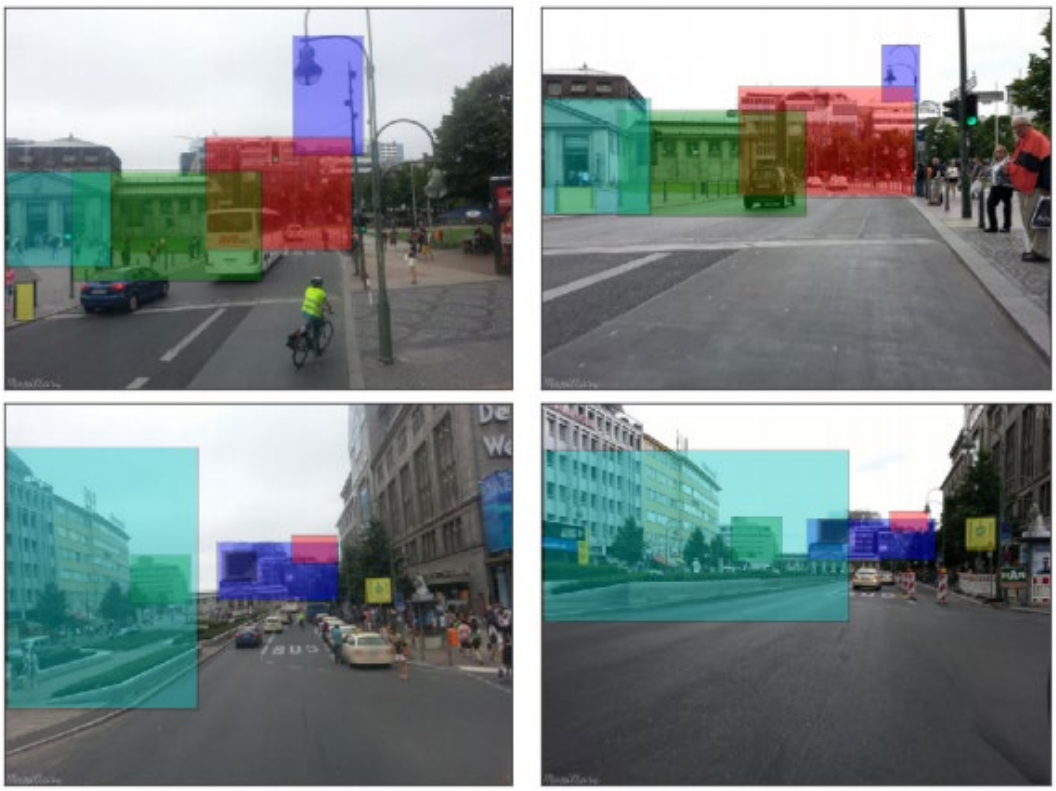


Figure I.8 : Segmentation of the view.

The colored boxes present landmarks which matched correctly between deferent viewpoints of a place [29].

I.6.1.4 Including Three-Dimensional Information

The image processing techniques mentioned above are appearance-based as 2D visual data, mean that the data was modeled directly in the visual domain rather than using a geometric model. The development of dense mapping techniques has been spurred by the use of data from additional sensors such as lasers or RGB-D cameras that give dense depth information as well as image data in many systems. Place recognition can be improved by these sensors by using of 3-D object data, geometric data can also be extracted from conventional cameras to allow metric calculation of the robot pose.

Dense Tracking and Mapping DTAM [30] stores dense metric information about every pixel, and Large-Scale Direct LSD-SLAM keeps semi-dense depth data on the parts of the image's structure and content. Obstacle avoidance, metric planning, localization and mapping are all possible with dense metric data. Therefore, fully autonomous vision-only navigation can be performed[30].

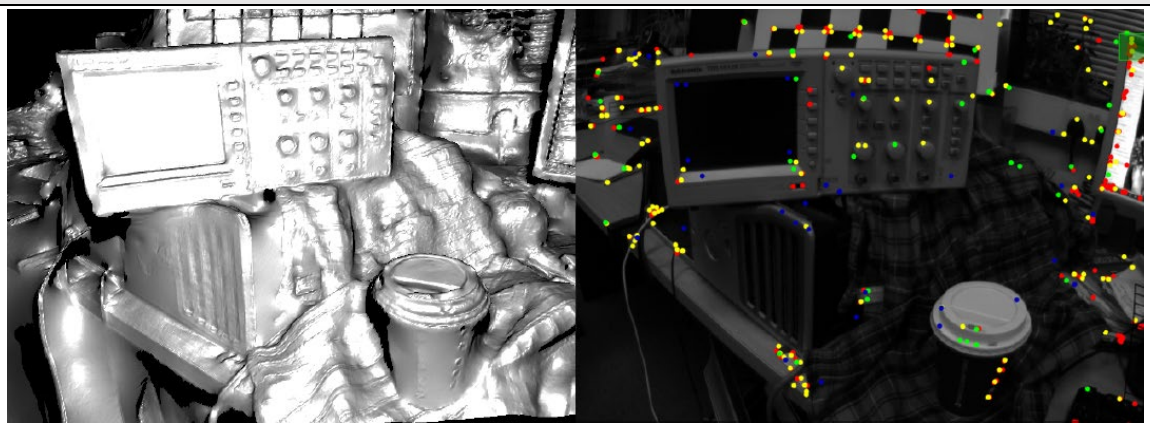


Figure I.9 : Example depth-map obtained using DTAM method.

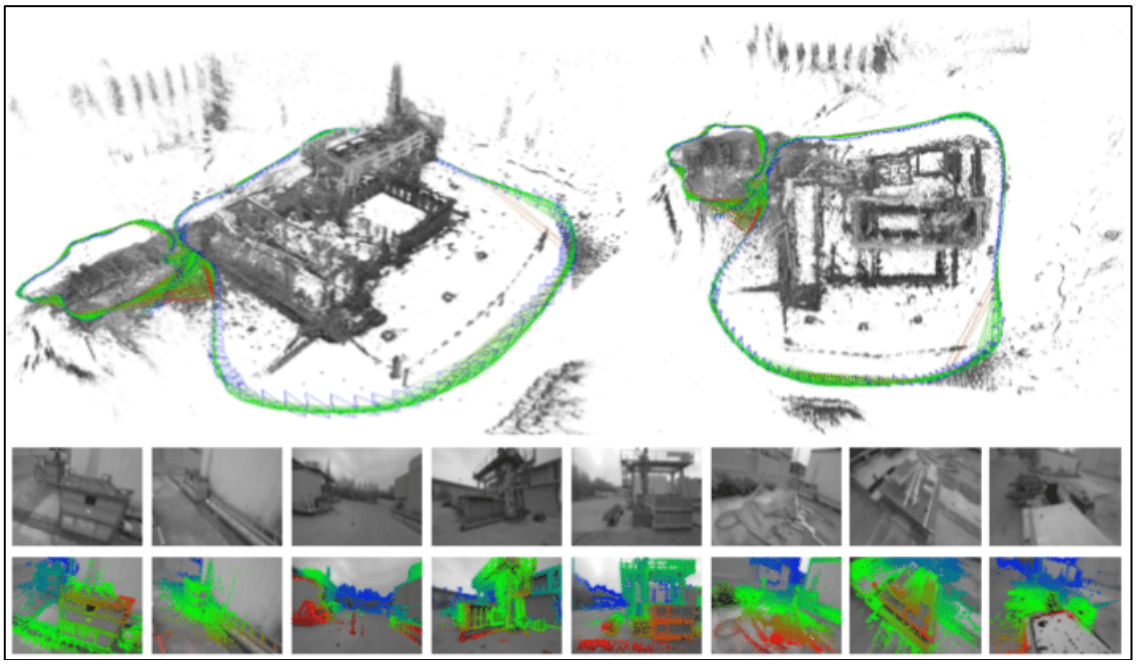


Figure I.10 : Camera trajectory with LSD-SLAM, showing keyframe depth-maps.

Database of 3-D object models is stored by SLAM++ [31], and it uses this 3-D database to recognize objects during navigation, and these objects are used as high-level place features. This representation has advantages over low-level place features. It offers rich semantic information that save memory by storing object labels rather than the entire object models in the map [31].

I.6.2 Place Mapping: Storing Data

Place mapping refers to the process of maintaining a faithful stored representation of the robot's knowledge of its environment, to which the current observation is compared. To this end, place recognition approaches rely on various mapping frameworks and map update mechanisms. Considering the mapping frameworks, two main approaches are spotted: database [32] and graph-base like topological [33].

When it comes to map updating, database frameworks are rarely update during operation, however topological frameworks are capable of that. Update strategies include simple

methods, which update nodes as loops occur, or more sophisticated ones, where long- short-term memory-based methods are used [34].

The Table I.1 displays a taxonomy of mapping approaches, which depends on the level of physical abstraction in the map and whether or not metric information is included in the place description

Level of map abstraction	Place description type	Comments
Pure image retrieval	Appearance-based	No position information
Topological	Appearance-based	Includes transition information
Topological-metric	Appearance-based	Includes metric information between but not within places
Topological-metric	Sparse metric information (landmark maps)	SLAM system – includes metric information between and within places
Topological-metric	Dense metric information (occupancy grid maps)	SLAM system – includes metric information between and within places

Table I.1 : Mapping Frameworks for VPR.

I.6.2.1 Database: Pure Image Retrieval

Pure image retrieval stores arbitrary amounts of images about each place in the map without any relation between them, it assumes that matching is based only on appearance similarity with no associated position information and applies image retrieval techniques that are not specific to place-based information. Although, a worthy information is lost when relative position information is not included, therefore a computationally efficient indexing techniques can be exploited. Those kinds of frameworks are mostly utilized for retrieval efficiency in pure image retrieval.

As the robot travels for long trajectory and visits a large number of places, the need of storage will increase but search efficiency will decrease. Thus, maps must be designed while considering large-scale efficiency and to ensure it. Image retrieval can be expedited using inverted indices if a bag-of-words model is employed to quantize the descriptor space, instead of requiring a linear search of all images in the database.

I.6.2.2 Graph-Base: Topological Maps

Topological maps are graph-based frameworks, which use nodes and edges to represent the map. The edges reflect the relationships between the nodes, whereas the nodes themselves represent places in the actual world. The topological-metric map varies from a pure topological map in that it uses metric information in the edges such as (relative position, orientation, or metric distance), whereas pure topological maps do not.

1) Pure Topological Maps

Pure topological maps retain information about the relative locations of places but not metric information about the dimension of nodes and the distance between them. Topological information can be used to both increase the number of correct place matches and filter out incorrect matches.

As it is mentioned above, image retrieval techniques can be improved by using BoW and inverted index, similarly topological maps can benefit from a location prior to speed up matching by requiring the place recognition system to only seek places that are known to be close to the robot's present location. To sample probable locations, a sampling-based approach such as a particle filter can be utilized. [35].

Place recognition with low-resolution data is possible thanks to the addition of topological information to the recognition process, which reduces memory needs. Successful place recognition was accomplished by the problem formulation as convex minimization with pictures as tiny as 48 pixels [36]. Even in difficult cases where pictures were blurred or viewed under diverse conditions such as different times of day, topological information enabled visual location recognition with as few as 32 pixels per image [37].

2) Topological-Metric Maps

Similar to how topological information may improve image retrieval, metric information can improve topological maps (distance, direction, or both on the map edges). The popular VPR approaches SeqSLAM is originally topological systems.

Metric information can represent only the relative poses between each node (places), where in this case topological-metric maps is counted as appearance-based. However, other approach for navigation can consist a metric-topological graph overlaid with local occupancy grids, metric information on the position of landmarks or objects in a location may be stored within each node.

I.6.3 Place Recognition: Belief Generation

A place recognition system's ultimate goal is to define whether a location has been visited previously or not. As a result, the basic purpose of any place recognition system is to construct a belief distribution by reconciling visual input with stored map data. The belief generating module is another name for this process. This distribution represents the likelihood or confidence that the current visual input matches to a visited place on the robot's map. As a result, this module is in charge of generating loop candidates based on belief scores, which may be calculated using frame-to-frame [38], sequence of frames [39], hierarchical [40], graphs [41] approaches.

It is common knowledge that if two place descriptions appear to be similar, it is possible they have been taking from the same physical spot means a greater likelihood, although the degree to which this is true depends on the environment. For example, Perceptual aliasing occurs when various areas are indistinguishable in repeating poses. Changing conditions, on the other hand, might make the same place look significantly different at various times.

I.6.3.1 Frame to Frame Method

In location recognition, the frame-to-frame matching method is the most popular. This method computes the belief distribution by matching just one frame at a time, and it employs different distance measurements, for similarity score computation, between the description of frames (such as cosine [42], Euclidean distance [43], and Hamming distance [44]).

I.6.3.2 Sequence of Frames Method

Sequence-based techniques use cost flow minimization [45] to discover matches in the similarity matrix and compute scores based on sequences of successive frames. Hidden Markov Models can be used to accomplish sequence matching in a probabilistic framework.

I.6.3.3 Hierarchical Method

Multiple matching algorithms are combined in a single location recognition framework using hierarchical methodologies. The coarse-to-fine architecture [46], for example, chooses the best candidates and then finds the best from those candidates.

I.6.3.4 Probabilistic Method

It can be used for detecting likelihood place matching. It is generally based on Gaussians. Different hypothesis and approaches have been developed in this context and especially for appearance-based probabilistic localization.

I.7 Conclusion

Visual place recognition has made great advances in the last few years, but still a long way from a universal place recognition system for robots that is robust and widely applicable across a range of robotic platforms and varying environments.

Visual place recognition is benefitting from research in other fields, particularly the great strides being achieved in computer vision in the fields of deep learning, image classification, object recognition, and video description. While techniques such as CNNs depend on Big Data and Big Compute, techniques such as cloud robotics and online/offline processing paradigms could be exploited even by small cheap mobile platforms. Developments in GPU hardware and novel camera sensors will inspire new concepts in place recognition as well as improving the efficiency and robustness of existing approaches.

Finally, what can visual place recognition offer to other research tasks? By necessity and opportunity, visual place recognition has taken up the challenge to solve condition invariant recognition to a degree that many fields have not, although under a more tightly constrained task specification than other tasks such as scene interpretation. The experience gained may have valuable applications, both in other robotic tasks such as object recognition and object classification in the wild, and in a diverse range of other areas including remote sensing, environmental monitoring, and other tasks that require recognition and identification in uncontrolled environments.

Chapitre II : Hierarchical Temporal Memory

II.1 Introduction

There are many tasks that humans find easy to do where computers and robots are currently unable to fulfil the quest. Visual pattern recognition, interpreting spoken language, identifying and controlling items by touch, and navigating in a complicated environment are all simple tasks for humans, and the neocortex is responsible for most of them.

Hierarchical Temporal Memory (HTM) is a technology that aims to simulate how the brain and its neocortex execute these tasks. In this model, both spatial and temporal properties of input data are considered. Therefore, for many cognitive activities, HTM holds the possibility of developing systems that perform as well as human. Jeff Hawkins, author of (Thousand Brains: A New Theory of Intelligence) [47], believes that Hierarchical Temporal Memory (HTM) will be applicable for many systems in the future. He launched a company, called Numenta, committed to modeling the human brain's function and applying it to artificial intelligence (AI). Numenta has developed and continuous to improve the Hierarchical Temporal Memory (HTM) framework described that simulates the Neocortex, the part of the brain that is responsible for human intelligence.

The problem of visual place recognition for mobile robots has been presented in the previous chapter. It has many challenging questions related to the environment and appearance changes. Therefore, to address this problem, the consideration of spatio-temporal information can considerably improve its performances. The HTM has this potential and seems to be an interesting solution for VPR problem. Hence, it is important to start by presenting HTM and its theoretical background before the development of an approach that exploit them for VPR application which will be presented in the next chapter.

This chapter presents the HTM and all the necessary concepts needed for its comprehension and exploitation. It starts with an overview of the HTM in the next section (II.2), which includes HTM principal’s and some HTM characteristic’s such as why it has a hierarchical structure? and why the time plays a critical role? Then, Section (II.3) describes the used mechanism for data presentation (Sparse distributed representation) and its major properties like capacity and tolerance to noise. Followed up by Section (II.4) that presents the HTM model and how it can process a sequence of inputs, even it includes examples for good illustration of HTM. Finally, in the section (II.5) present the conclusion.

II.2 Hierarchical Temporal Memory Overview

Hierarchical Temporal Memory (HTM) is a type of machine learning aimed at gaining the neocortex's structural and algorithmic features.

The neocortex shown in the **Figure II.1** is the mammalian brain's center of intelligent thinking. It is responsible for high-level vision, listening, touching, moving, talking, and planning [48]. The neuronal circuitry of the neocortex is incredibly uniform. HTM presents a theoretical framework for comprehending the neocortex's various functions. Its implementation differs from typical computer programming. Programmers design unique programs to tackle specific difficulties using today's computers. HTMs, on the other hand, are

taught by being given to a continuous stream of sensory input. It is a generic system that can realize many functionalities according to what it has been exposed to.

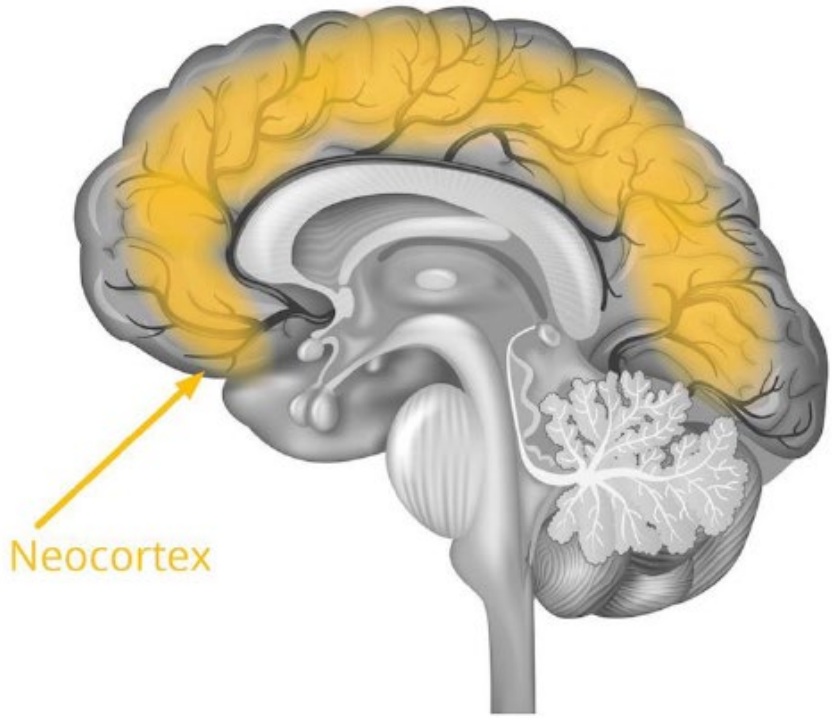


Figure II.1 : Picture of the part of the human brain that makes up the neocortex.

HTM is a memory-based technology at its heart [49]. HTM networks rely on storing a huge collection of patterns and sequences after being trained on a vast amount of time-varying data. The used method for storing data and using it differs fundamentally from the traditional programming approaches. The structure of traditional computer memory is flat, and thus lacks an inherent concept of time. On top of the flat computer memory, a programmer may create any type of data organization and structure. They have complete control over how and where data is kept. HTM memory, on the other hand, is more constrained. It is based on time and has a hierarchical architecture. Data is always kept in a dispersed manner. The HTM's user determines the hierarchy's size but the HTM decides where and how data is kept [48].

Some of the main concepts of HTM are discussed in this sub-section, including what is HTM regions and the process of its construction, its hierarchical structure and the sparse distributed representations (SDR) that characterize its information coding. This sub-section explains also the role of time (sequence) in HTM.

II.2.1 Region

The neocortex is a vast 2mm thick sheet of neural cells [48]. The biologists and neuroscientists split the neocortex into separate sections or "regions". Some regions **Figure II.2** take information directly from the senses, whereas others receive it after passing through

several regions. It has also observed that these regions interact with each other and are organized in a hierarchical structure.

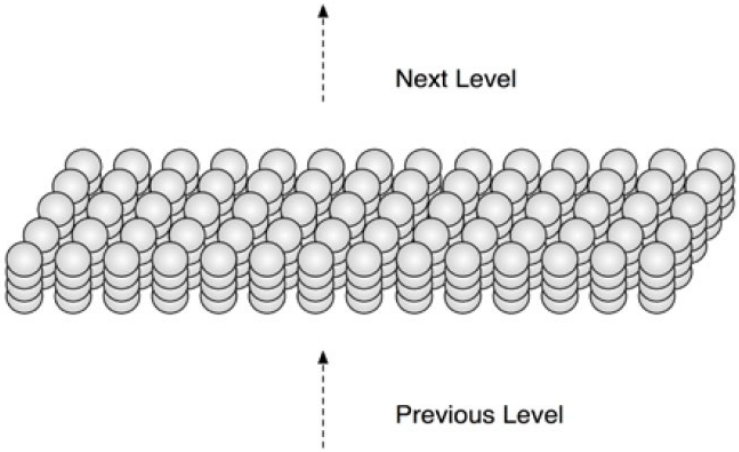


Figure II.2 : A section of an HTM region.

In terms of details, all neocortical regions appearance are same as shown the **Figure II.3**. They differ in size and position in the hierarchy, but they are generally identical. A neocortical region with a depth of 2mm usually includes six layers: five layers of cells and one non-cellular layer. Many linked cells are grouped in columns in each layer of the neocortical region. A sheet of densely linked cells placed in columns also makes up HTM regions.

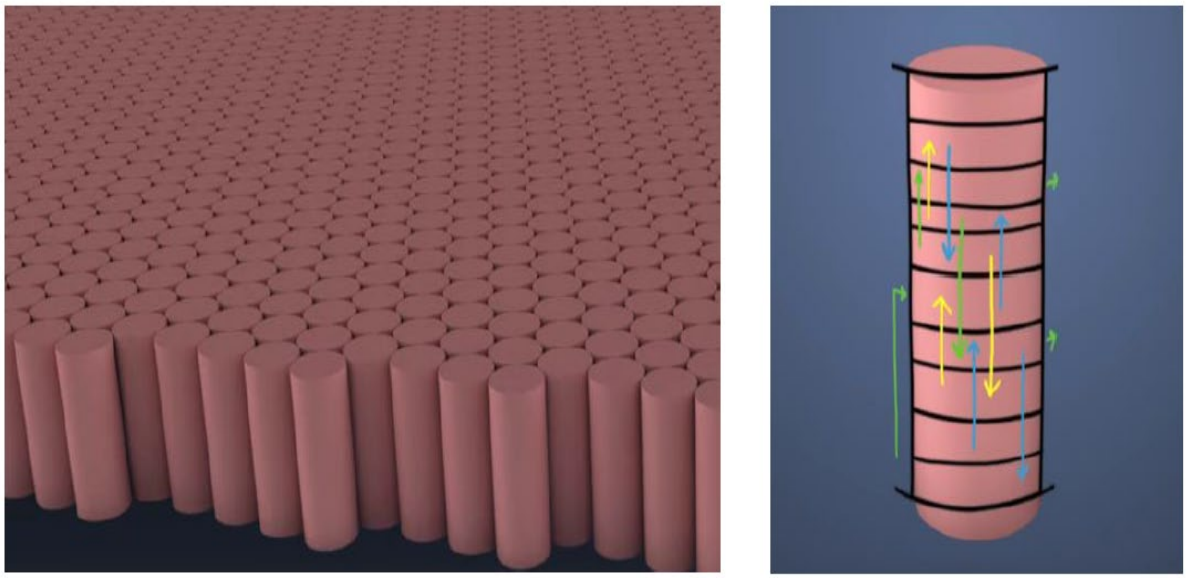


Figure II.3 : Neocortex spread flat made up of columns, image from [50].

II.2.2 Hierarchy

Each HTM region indicates a level of hierarchy, and an HTM network is made up of regions grouped in a hierarchical order [48]. **Figure II.4** presents the architecture of the three layers-HTM and shows the cells organization. To avoid confusion, it is important to precise that effectively “Region” and “Level in the hierarchy” represent the same entity. The appellation

"Region" is used to refer to its internal functions while the term "Level" is reserved to indicate its role in the hierarchy.

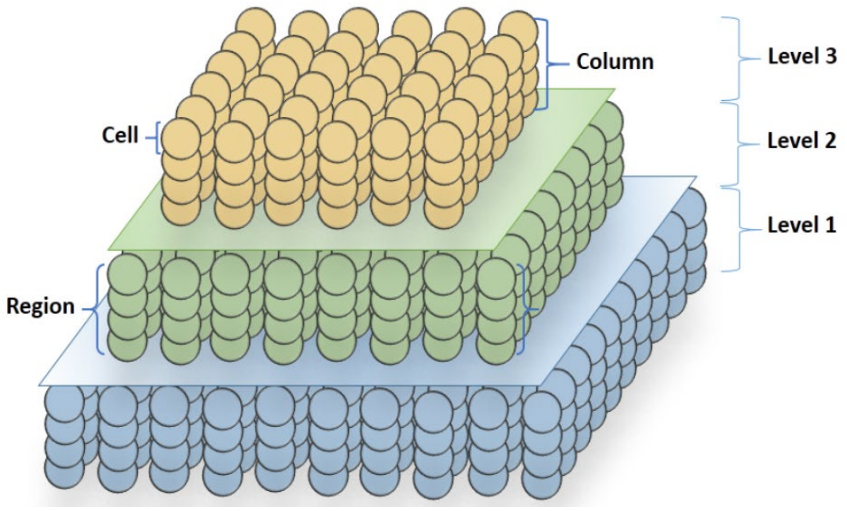


Figure II.4 : Architecture of the three layers-HTM, image from [51].

Multiple HTM networks can be coupled. This kind of structure makes sense if the input data came from more than one source or sensor. For example, one network might be processing auditory information and another network might be processing visual information as shown in **Figure II.5**.

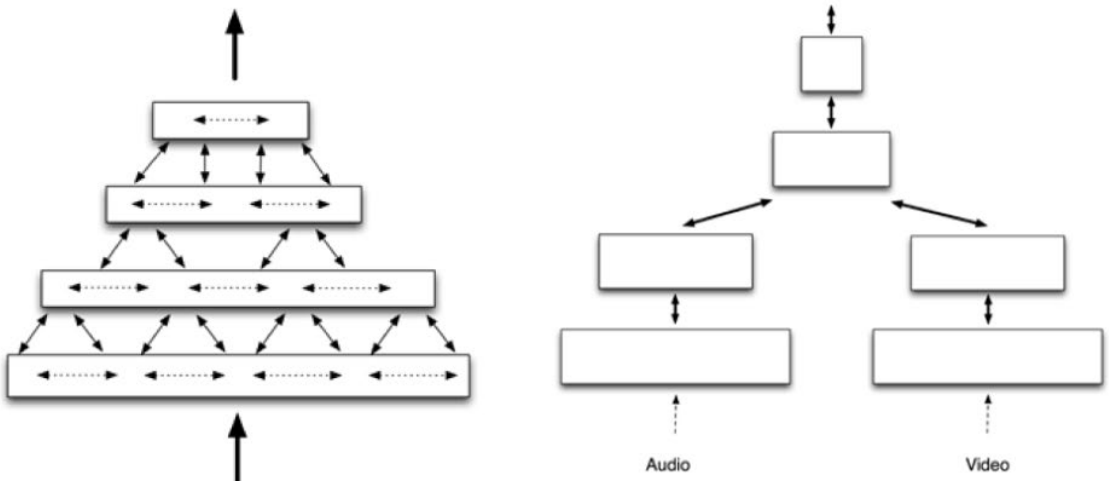


Figure II.5 : Illustration of coupled HTM networks.

Efficiency is an advantage of hierarchical structure. Since patterns learnt through each level of the hierarchy are reused when coupled in innovative ways at upper levels, it greatly decreases training time and memory use. However, a single HTM region may handle several smaller prediction problems.

II.2.3 Sparse Distributed Representations

Although the neocortex's cells are extensively linked, inhibitory cells ensure that only a small fraction of them are active at any given moment [49]. Thus, within a huge population of cells, data in the brain is always encoded by a tiny fraction of activated neurons. A "Sparse Distributed Representation, SDR" is the name for this type of encoding. The term (sparse) in

SDR highlights the fact that just tiny fractions of neurons are activated at the same moment, (Distributed) states that in order to represent information, multiple neurons must be activated **Figure II.6**.

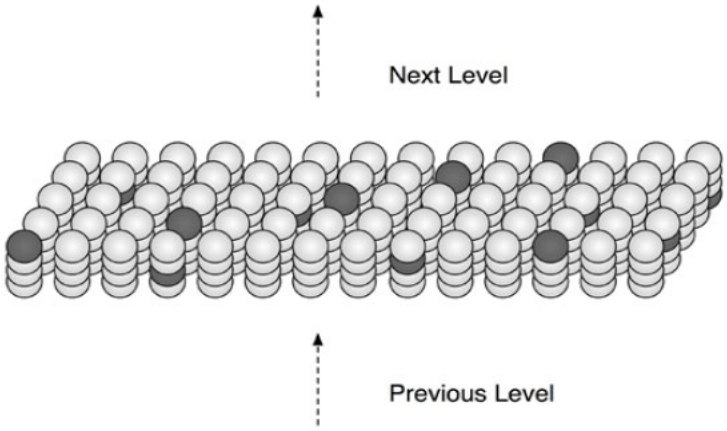


Figure II.6 : An HTM region that shows sparse distributed active cells.

Indeed, the memory processes inside an HTM region rely on SDRs and would not function otherwise. An HTM region's input is always in form of distributed representation, but it may or may not be sparse, therefore the first thing it does is encoding the real data into SDRs.

II.2.4 The role of time

The sequential nature of data (temporal dimension) has an important role in HTM and that can be discussed from different viewpoints according to the following phases: Learning, Inference and prediction.

•**Learning:** In order to learn, all HTM systems must be exposed to time-changing inputs during training [48]. For example, imagine a cat is running. At each instance in time the cat causes a pattern of activity in the eye. The human brain perceives these patterns as different views of the same cat, but mathematically the patterns are entirely dissimilar. The brain learns that these different patterns mean the same thing by observing them in sequence. It can learn to identify cats by seeing many instances of many breeds of cats, not just one single view of one single cat. Therefore, time is considered the "supervisor" that teaching the brain which spatial patterns go together.

Typically, an HTM network needs to be trained with lots of data. The HTM algorithms are responsible for learning temporal sequences from a stream of input information.

Finding patterns and then sequences of patterns in data input is how an HTM region learns about its environment. The region does not "know" what its inputs represent; it works in a purely statistical realm. It searches for input bits forms that occur frequently together, which is termed as spatial patterns. It then looks for how these spatial patterns appear in sequence over time, which is called temporal patterns or sequences.

•**Inference:** without using time, the human brain can infer almost nothing from tactile and auditory senses. The ability to infer based on quick visual exposure is a special case made possible by the statistical properties of vision and years of training [48]. The general case for vision, hearing, and touch is that inference requires time-changing inputs.

After an HTM has learnt the patterns in its environment, it may infer new inputs. An HTM will match input to already learnt spatial and temporal patterns when it gets it. The essence of inference and pattern matching is properly matching fresh inputs to already stored sequences.

•**Prediction:** learning and recognizing sequences are the basis of forming predictions. Once an HTM learns what patterns are likely to follow other patterns, it can predict the likely next pattern(s) given the current input and immediately past inputs [48].

Every region of an HTM stores sequences of patterns. By matching stored sequences with current input, a region forms a prediction about what inputs will likely arrive next.

In summary, time-series-based in Hierarchical Temporal Memory plays a crucial role in learning, inference and prediction.

II.3 Sparse Distributed Representation Modeling

Data should be transformed into sparse distributed representation (SDR) before being sent into the HTM learning process. The neocortex encodes information through sparse distributed rhythms of brain activity, according to neuroscientists [48]. Only a tiny percentage of neurons are active at any given moment, while the others are silent. Cortical activity is limited. Each activated neuron has a symbolic meaning. As a result, information is spread and no single neuron is important.

The bits in an SDR reflect neurons in the brain, with 1 meaning an active neuron and 0 meaning an inactive neuron. A SDR can take the form of a vector as shown by **Figure II.7**.

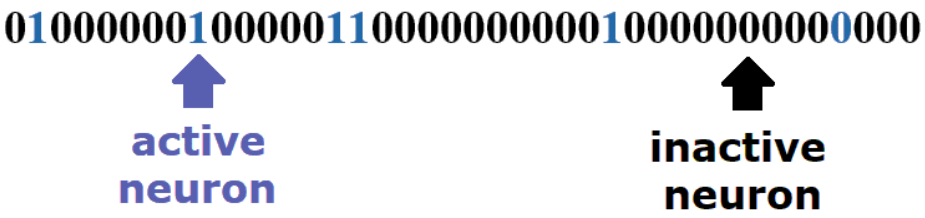


Figure II.7 : Region's input as vector.

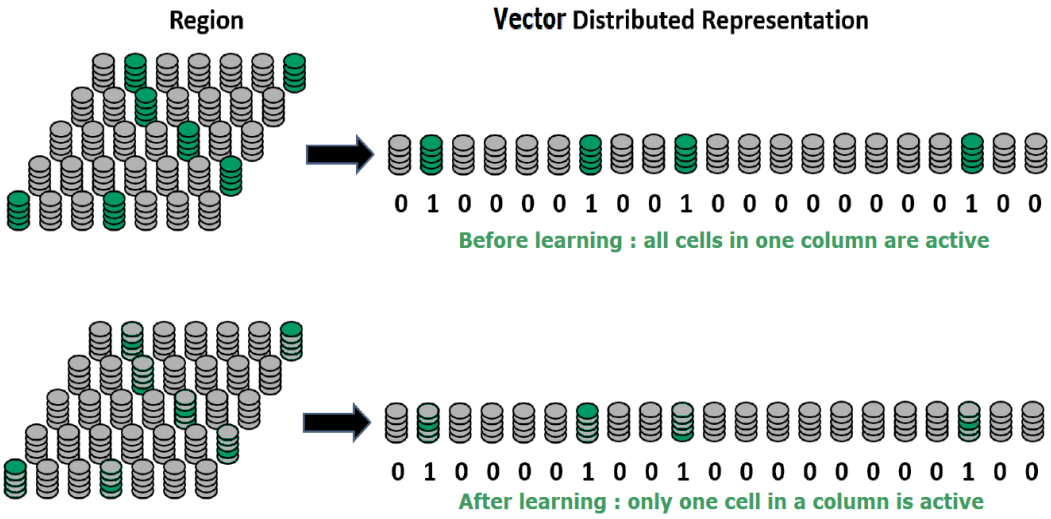


Figure II.8 : SDR forms before and after learning.

The above picture shows a representation of an SDR before and after learning. Initially, all cells in one column are active. Then, after the learning, only one cell of a previously active column is continuous to be active.

In recent years, the idea of sparse representations has attracted a lot of attention in theoretical neuroscience [52], [53], [54]. SDRs offer several benefits over neocortical information representations, such as enormous capacity and noise tolerance, subsampling, and so on.

Some of the main properties of SDR are discussed in the following sub-sections, including capacity of a SDR and how tolerance to noise it is, and how the data should be encoded so the SDR accept it.

II.3.1 Capacity

Also known as uniqueness, and it is the number of unique encoding possible in **n**-dimensional vector with **w** of active neurons [55]. The number of SDRs that can be represented by a vector of (n, w) properties is:

$$\binom{n}{w} = \frac{n!}{w!(n-w)!} \qquad \text{II.1}$$

Example:

For $n = 100$ and $w = 5$, the number of encodings is 75 287 520.

With more typical values $n = 2048$ and $w = 40$, the SDR representation space is extremely large at 2.37×10^{84} encodings.

With this much of space, the chances of two random SDRs being similar are quite low. As a result, HTM can easily recognize information.

The percentage that two random SDRs with the same **n** and **w** is equal is $1/\binom{n}{w}$, which is very low.

II.3.2 Tolerance to Noise

Sensing noise, transmitting data failures, and natural device boundaries are all common causes of incorrect or missing data in real-world sequence learning. A successful sequence learning algorithm must be robust to input noise [55].

The results in [54] show that neurons storing data via SDRs have a lot of representational capacity, and their robustness to noise is good enough that they can reliably classify with up to 50% noise. For example, two SDRs **x** and **y** have **n = 40** and $w_x = w_x = 4$

$$x = [0100000010000011000000000010000000000000]$$
$$y = [0001000010000010100000000010000000000000]$$

Figure II.9 : Common active cells at the same time means there is some similarity.

The overlap between **x** and **y** is **3** and if threshold $\theta \leq 3$, the two vectors match.

The idea of match comes in handy for determining if two SDRs are semantically similar. However, perfect match, which needs $\theta = w_x = w_y$ is unattainable in SDR since noise always occurs. As a result, by reducing the value of θ an inexact match is required. Lowering it, on the other hand, increases the chance of a false match (false positives) with another random SDR. This problem may be solved by selecting proper parameter values.

II.3.3 Encoding Data

Basically, the HTM sequence memory uses sparse distributed representations (SDRs) to handle actual sequence learning cases. In such cases where data is not binary, so it is a must to use an encoder to transform the original data to a binary representation in order to be used for input to the Spatial Pooler [56], which convert it to SDRs. As a result, the encoder has a strong influence on what aspects of the data contribute to the similarity.

All encoders used in HTM applications must meet the following requirements:

1. SDRs with overlapping active bits should come from semantically identical data.
2. The output SDR should always be the same, if the same input data enters again.
3. For all inputs, the output should have the same dimensionality (total amount of bits).
4. For all inputs, the output should have identical sparsity and enough one-bits to tolerate noise and subsampling.

II.4 Hierarchical temporal memory modeling

In this section, details of the neuron used in HTM are presented. Then, how HTM deals with high-order sequence. Finally, a very detailed of spatial and temporal concepts.

II.4.1 HTM Neuron Model

A neocortical neuron is known to have hundreds of synapses on its dendrites [48]. **Proximal synapses**, which are located near to the cell body, have a significant impact on the cell. **Distal synapses** these are the major synapses and they are located distant from the cell body. The activation of a single distal synapse has relatively little influence on the cell body, but the mechanism by which distant synapses affect the cell body has long been an unknown

The neurons in a region receive information from three different sources [48]; feedforward, contextual, and feedback information as shown in **Figure II.10**. Where, distal synapses indicate lateral connections within a layer and feedback connections from a higher area, whereas proximal synapses indicate feedforward inputs into the layer.

Each cell has two zones: a **proximal zone** with a single dendritic segment and a **distant zone** with a collection of independent dendritic segments.

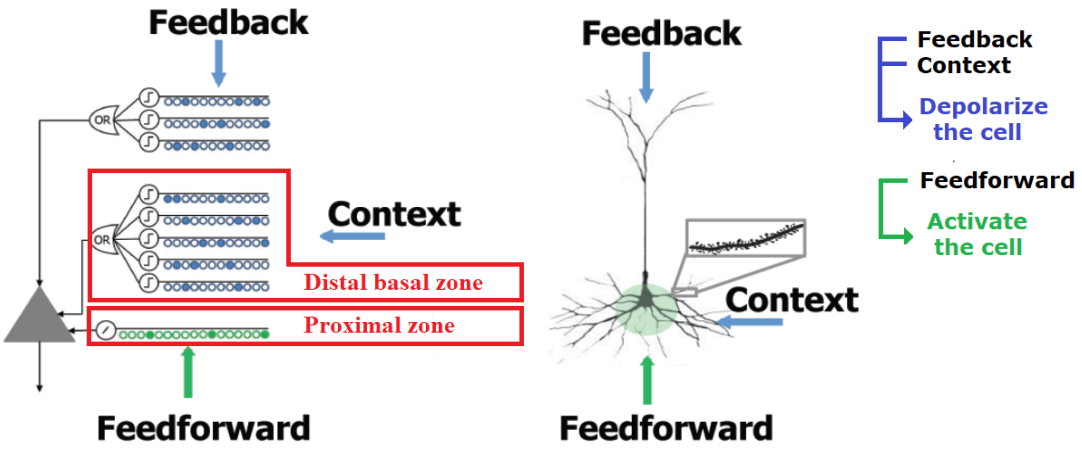


Figure II.10 : Modeled neuron has three input, the out put is its state.

The output of the neuron is always binary: it is either active or not, but the cell has three states

- **Active state:** coming from the activation of the proximal zone.
- **Predictive state:** also known as depolarized state. Each distal dendritic segment has lateral synaptic connections to other neurons in the layer. When the number of active connections surpasses a certain threshold, a segment becomes active. Instead of causing the cell to fire, an active segment leads the cell to enter a predictive state.
- **Non-active state:** indicates that a cell is not activated.

II.4.2 Learn High-Order Sequence

The term high-order refers to "high-order Markov chains" which have use context, for example, a first-order Markov chain predicts the next state depending on the current state only. Similarly, an nth-order Markov chain will look at the current and previous **n** states to predict the next state [49].

The HTM network is made up of a layer of HTM neurons arranged in columns and each of which contains several cells, as shown in **Figure II.11**.

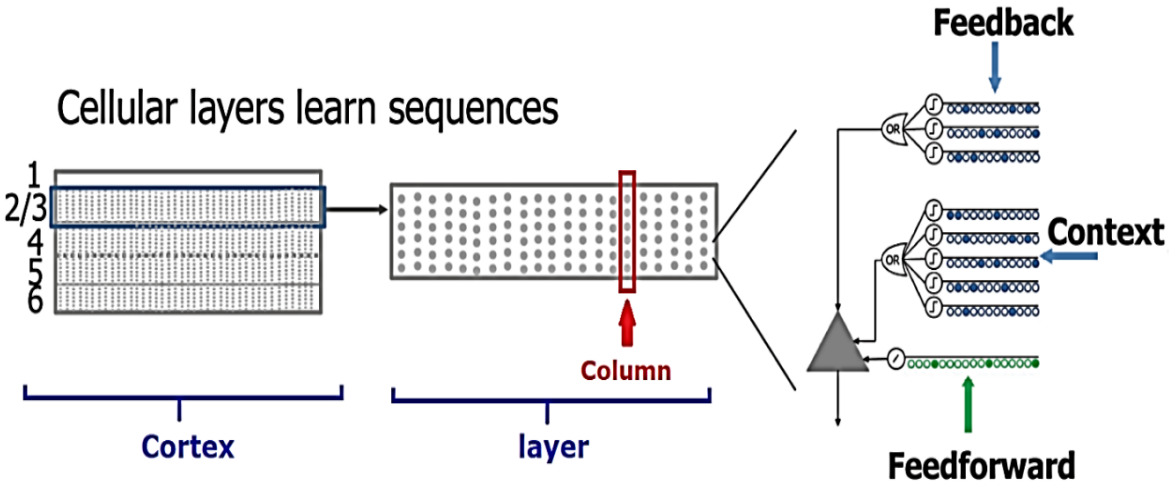


Figure II.11 : Presentation of the cortex's six cellular layers model.

At any time, cells in the same column get the same feed-forward input. Each cell has a unique distal basal segment which connects the network's cells.

The learning process of temporal patterns is divided into two steps [49]:

- **Step 1:** HTM learns input bit spatial patterns by looking for input bits that frequently appear together. Because each column is related to a subset of input, the column becomes active if certain bits occur together. As a result, a set of active columns represents a spatial pattern.
- **Step 2:** transition between spatial patterns is learned by HTM. When a cell becomes active, it establishes connections with other active cells. Because distal synapses detect patterns and operate as pattern recognizers, cells can foretell when they will become active. When the distal basal segment encounters a learned (memorized) pattern for the second time, it depolarizes the cell and puts it into a predictive state. When an expected input comes, the cell makes the correct prediction and activates. If an unexpected input enters, all of the cells in the specified columns will perceive it as a new pattern and become active. This is how all of the cells function together. As a result, HTM can remember and retain patterns, as well as foretell which pattern will happen.

Example:

This example is taken from [49], consider the abstract sequences **A-B-C-D** and **X-B-C-Y** to demonstrate the intuition underlying these representations (**Figure II.12** before learning). In this case, knowing whether the sequence began with **A** or **X** is necessary to make the proper prediction after (**C**). The subset of columns that include active cells represents the current inputs (black). This collection of active columns is independent of time and only depends on the current input. Then after learning, distinct cells in this subset of columns will be active based on historical context predictions (**B'** vs. **B"**, **C'** vs. **C"**, **Figure II.12** after learning). These cells lead to prediction of the element following (**C**) based on the collection of cells with lateral connections to columns representing **C** (**D** or **Y**).

A lot of intriguing aspects emerge from this dual representation paradigm. First, sparse representations enable the model to make several predictions at the same time. If the get network input (**C**) without any context, all cells in the columns reflecting the (**C**) input will fire, resulting in (**D**) and (**Y**) predictions. Second, because information is stored through the distributed coactivation of numerous cells, the model is naturally robust to both input noise and system failures such as the loss of neurons and synapses [49].

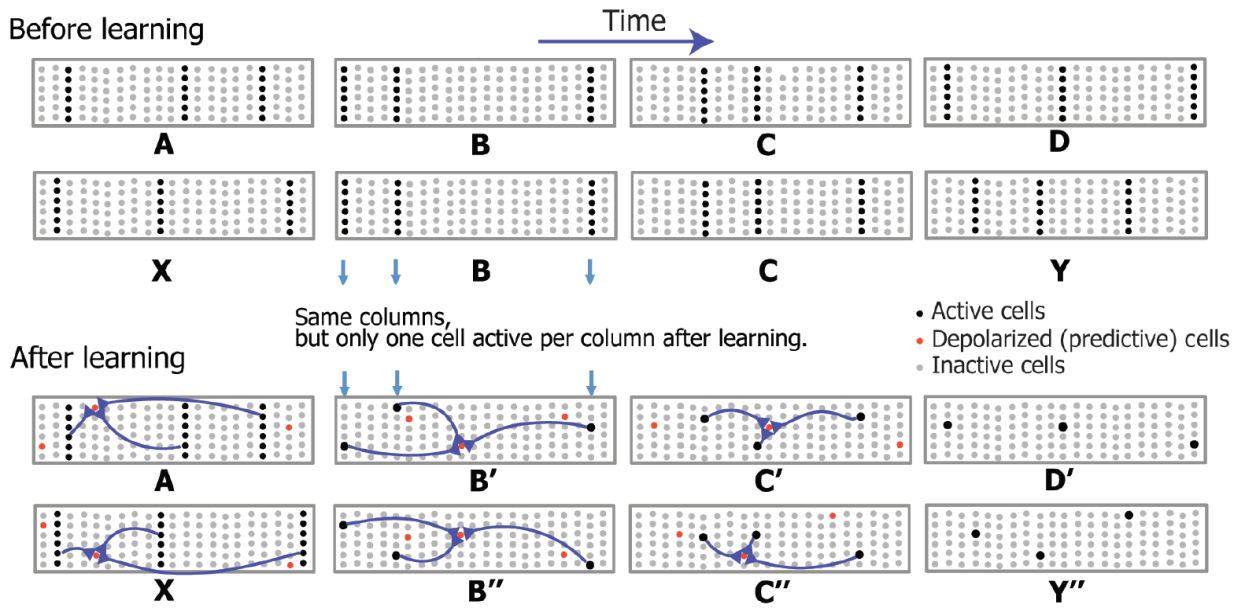


Figure II.12 : Learning shared sub-sequences (BC) in high-order sequences.

This above **Figure** shows that:

- **Before learning:** all the cells in a column become active.
- **After learning:** Through intra-column inhibition, cells that are depolarized through lateral connections become active quicker and block other cells in the same column from firing.

II.4.3 Spatial and temporal concepts

Practically, the mentioned above steps of High-Order Sequence Memory (HOSM) are named Spatial Pooler (SP) and Temporal Pooler (TP). The processing of feedforward input is done by Spatial Pooler, and the processing of feedforward and contextual input is done by Temporal Pooler (temporal Memory). The feedback connections are not addressed because current HTM theory suggests that feedback is an optional component.

II.4.3.1 Spatial Pooler

1) Spatial Pooler Role:

The spatial pooling (SP) is the first and most critical stage in HTM and it is responsible for converting the binary input data into SDR. The SDR representation of the input is created by selecting a few columns to reflect the input patterns. Typically, every column in the HTM region is directly connected to the input space or other regions in the hierarchy. The columns of the first region are connected to the input space while other columns of each region are connected to the lower regions using proximal segments as shown in **Figure II.13**.

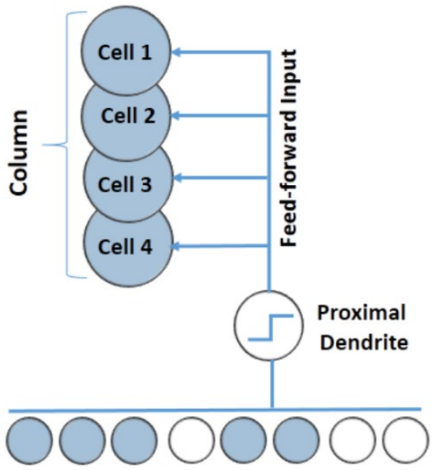


Figure II.13 Proximal dendrite model [51].

When the synapses are active and connected to a reasonable number of active bits in the input space, the proximal dendrite segment will be active. This will select that column to contest with its neighbors for the right to represent the region's input. The column with more active synapses and active inputs inhibits and turns OFF its neighbors (winner). The spatial pooler generates a vector as its output. The vector length corresponds to the number of the columns and the ON bits (1) of the vector represent the winning columns [48].

2) Spatial Pooler Details:

The SP has a number of important characteristics, including quick adaptability to changing input data, enhanced noise robustness through learning, efficient cell utilization, and cell death resistance [48].

- Start with an input consisting of a fixed number of bits. These bits might represent sensory data or they might come from another region lower in the hierarchy.
- Assign a fixed number of columns to the region receiving this input. Each column has an associated dendrite segment. Each dendrite segment has a set of potential synapses representing a subset of the input bits. Each potential synapse has a permanence value. Based on their permanence values, some of the potential synapses will be valid.
- For any given input, determine how many valid synapses on each column are connected to active input bits.
- The number of active synapses is multiplied by a “boosting” factor which is dynamically determined by how often a column is active relative to its neighbors.
- The columns with the highest activations after boosting disable all but a fixed percentage of the columns within an inhibition radius. The inhibition radius is itself dynamically determined by the spread (or “fan-out”) of input bits. There is now a sparse set of active columns.
- For each of the active columns, user adjust the permanence values of all the potential synapses. The permanence values of synapses aligned with active input bits are increased. The permanence values of synapses aligned with inactive input bits are decreased. The

changes made to permanence values may change some synapses from being valid to not valid, and vice-versa.

3) Example:

Here is a mathematical example of SP processing taking from [55]:

The SP process takes as input a binary vector of length N (in practice mostly this vector is usually sparse). Each column in the SP represents proximal segments in a cell. A binary matrix with dimensions $N \cdot C$ can be used to represent the set of connected synapses in the SP. This matrix is not restricted to be sparse, but in practice is almost always sparse. The result of the vector-matrix multiplication is a vector of overlap counts, the third data frame in **Figure II.14**. An inhibition step chooses the column winners to form the output.

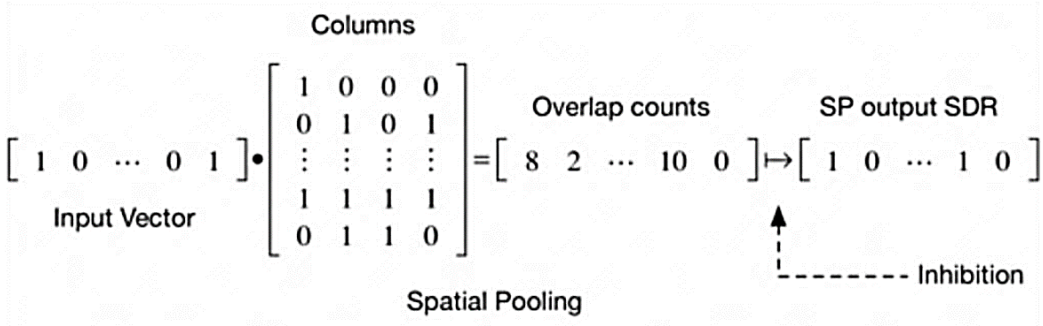


Figure II.14 : Example of using spatial pooling.

II.4.3.2 Temporal Pooler

1) Temporal pooler role:

The temporal pooler (TP) is responsible for learning sequences and making predictions [48]. This stage picks which neurons to fire after computing the activated columns via the SP. It extends the HTM ability from identifying features and classes of inputs to predict future inputs from past patterns, based on the current temporal context of the current input.

This stage picks which neurons to fire after computing the activated columns via the Spatial Pooler algorithm. By default, the distal dendrites of neurons potentially connect to the neighboring neurons of the same layer **Figure II.15**. These dendrites allow neurons to predict their activation by growing synapses to the previously active neurons. Therefore, a neuron is depolarized and put in a predictive state if its distal dendrites detect the activation that occurs before itself. The depolarized neurons are primed to fire in the following time step if the feedforward columnar activation includes those neurons. If a depolarized neuron becomes active, it inhibits the rest of the neurons in the same column [48].

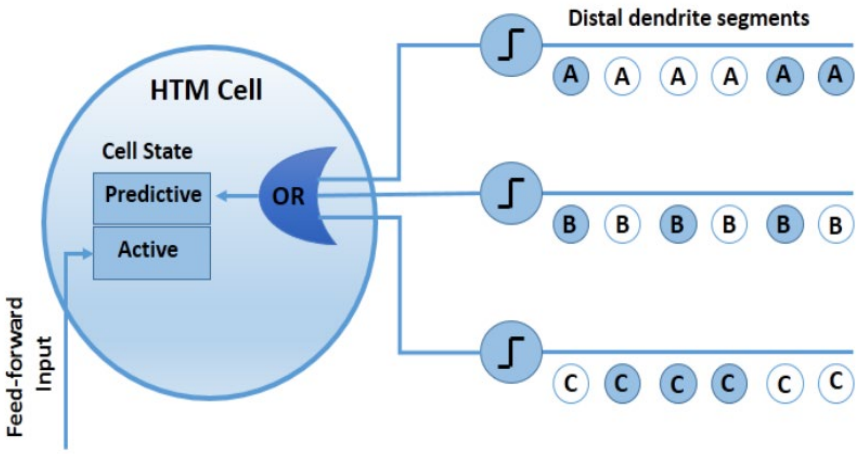


Figure II.15 : Distal dendrite model [51].

All the neurons become active if there is no depolarization. The resulting neural activation computed on this stage represents the contextual information of the columnar activation. As a result, the same columnar activations may have different active neurons based on the previous activations.

2) Temporal Pooler Details:

Temporal pooler starts where the spatial pooler left off, with a set of active columns representing the feed-forward [48].

- For each active column, check for cells in the column that are in a predictive state, and activate them. If no cells are in a predictive state, activate all the cells in the column. The resulting set of active cells is the representation of the input in the context of prior input.
- For every dendrite segment on every cell in the region, count how many established synapses are connected to active cells. If the number exceeds a threshold, that dendrite segment is marked as active. Cells with active dendrite segments are put in the predictive state unless they are already active due to feed-forward input. Cells with no active dendrites and not active due to bottom-up input become or remain inactive. The collection of cells now in the predictive state is the prediction of the region.
- When a dendrite segment becomes active, modify the permanence values of all the synapses associated with the segment. For every potential synapse on the active dendrite segment, increase the permanence of those synapses that are connected to active cells and decrement the permanence of those synapses connected to inactive cells. These changes to synapse permanence are marked as temporary.
- This modifies the synapses on segments that are already trained sufficiently to make the segment active, and thus lead to a prediction. However, the HTM algorithm look to extend predictions further back in time if possible. Thus, it picks a second dendrite segment on the same cell to train. For the second segment HTM algorithm choose the one that best matches the state of the system in the previous time step. For this segment, using the state of the system in the previous time step, increase the permanence of those synapses that are connected to active cells and decrement the permanence of those synapses connected to inactive cells. These changes to synapse permanence are marked as temporary.

- Whenever a cell switches from being inactive to active due to feed-forward input, the HTM traverse each potential synapse associated with the cell and remove any temporary marks. Thus, HTM update the permanence of synapses only if they correctly predicted the feed-forward activation of the cell.
- When a cell switches from either active state to inactive, undo any permanence changes marked as temporary for each potential synapse on this cell. HTM does not want to strengthen the permanence of synapses that incorrectly predicted the feed-forward activation of a cell.
- Note that only cells that are active due to feed-forward input propagate activity within the region, otherwise predictions would lead to further predictions. But all the active cells (feed-forward and predictive) form the output of a region and propagate to the next region in the hierarchy.

3) Example:

Here is a mathematical example of TP processing taking from [55]:

- Following the SP, the first phase of TM establishes the current active temporal state. The cells in every column anticipated in the previous time step t-1, as well as the actual winning columns in the SP output SDR, are examined in this phase. A fully anticipated cells in winning columns remain ON to show the system's active temporal state. **Figure II.16** shows a schematic of this operation. The first matrix is the $1\times C$ SP output SDR, followed by the $L\times C$ matrix predicted state matrix. Row-by-row element-wise multiplication (Represented by the symbol Θ) of these two matrices is used to determine the active state, which is also $L\times C$. All three structures are sparse.

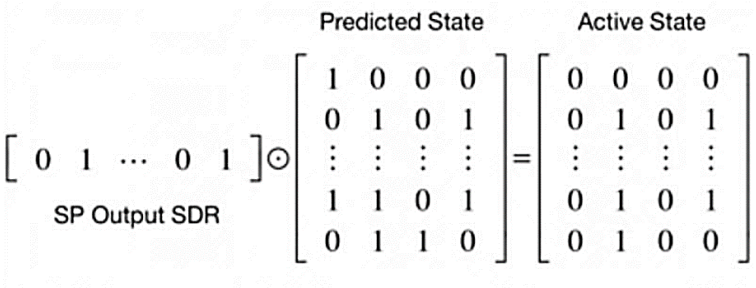


Figure II.16 : Example of using temporal memory- Phase 1.

- The second phase of TM matches individual cell segments to the actual active state to derive predictions for the future time step. **Figure II.17** shows this operation schematically. Each cell contains a list of segments, where each segment is represented by an SDR of size $L\times C$. For each segment the TM computes a match between this representation and the current active state. If the match of any segment is above the threshold, the predicted state for this cell will be ON.

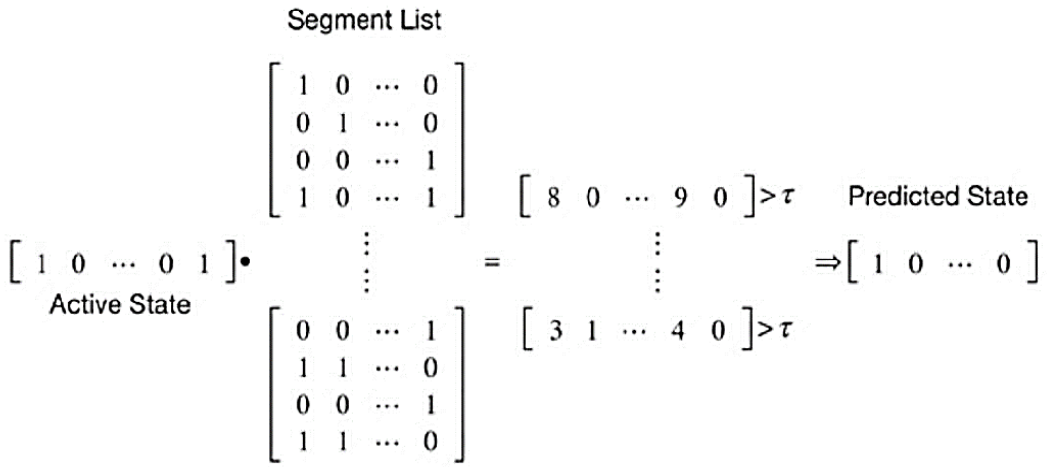


Figure II.17 : Example of using temporal memory- Phase 2.

II.5 Conclusion:

In this chapter, a comprehensive survey of the Hierarchical Temporal Memory architecture has been presented, its properties have been also reviewed. In addition, multiple basic examples of HTM process were included for a better understanding.

The HTM cortical learning algorithm embodies what the scientists believe is a basic building block of neural organization in the neocortex. It shows how a layer of horizontally-connected neurons learns sequences of specific patterns. Variations of the HTM cortical learning algorithm for the weight update process in the HTM SP and TM are used in different layers of the neocortex

Chapitre III : An HTM-based approach for VPR

III.1 Introduction

Visual place recognition is an essential capability for autonomous mobile robots which use cameras as their primary sensors. Although there has been a considerable amount of research in the topic, it has been introduced in the first chapter. The high degree of image variability poses extra research challenges. Following advances in neuroscience, new biologically inspired models have been developed. Inspired by the human neocortex, hierarchical temporal memory model has potential to identify temporal sequences of spatial patterns using sparse distributed representations, which are known to have high representational capacity and high tolerance to noise (more details are given in the second chapter). These features are interesting for place recognition applications. This chapter proposes the usage of Hierarchical Temporal Memory theory with the aim of solving the Visual Place Recognition problem during a long-term of an unsupervised operation.

This chapter presents our approach that uses HTM for VPR. Therefore, it will discuss all its steps. It starts with a general overview of the proposed approach in the next section (III.2), Then, Section (III.3) shows how the visual data will be treated with image features extractors and descriptors (the local feature descriptor SURF will be employed). Followed up by Section (III.4) that presents the HTM algorithms, a detailed description and pseudocode are included. The section (III.5) gives a theoretical way of handling the output of the HTM based on Probabilistic Method. Finally, in the section (III.6) the conclusion.

III.2 General overview of the proposed approach

The proposed approach for solving VPR problem have a lot of phases that the sequential input should pass through them one by one, as illustrated in **Figure III.1.**

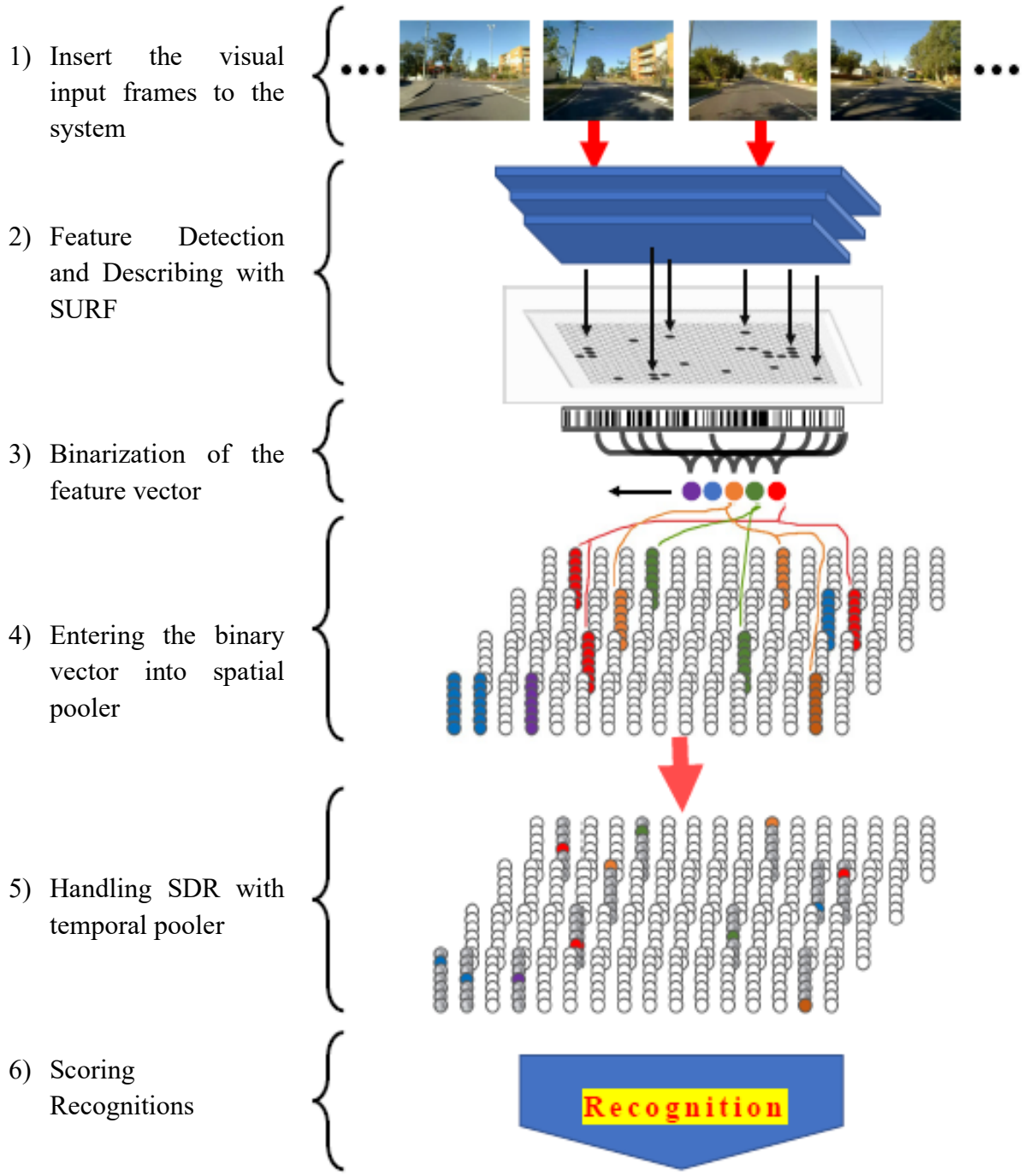


Figure III.1 : shows multiple phases of the general architecture

III.3 Data Handling

Autonomous mobile robots can be used in many challenging applications such as environment reconstruction, search and rescue, autonomous driving and planetary exploration. In order to be able to succeed in such complex scenarios, it should be able to recognize a previously visited place. Therefore, the proposed systems must take into account the context of image variations caused by different times of the day, weather, lighting and seasons conditions (as seen in **Figure III.2** that shows two frames from the dataset that is going to be considered in this project).



Figure III.2 : Same position different appearance, from the treated Data.

Therefore, the proposed architecture must be able to describe places. The most common approaches are based on local image features using extractors and descriptors such as SURF. so, because of that, Speeded Up Robust Features (SURF) was employed, which is a widely-used **scale and rotation invariant** descriptor presented by Bay et al [57] and according to its authors SURF features outperformed a lot of methods respect to repeatability, robustness and distinctiveness of the descriptors.

Interest Point Detectors:

The detection method is based on the Hessian matrix to detect both the location and the scale of interest points. The Hessian matrix is approximated, using a set of box-type filters. and it relies on integral images to reduce the computation time.

Local Descriptors:

The descriptor process represents a distribution of Haar-wavelet responses within the interest point neighborhood and makes an efficient use of integral images. Three different versions of the descriptor have been studied: the standard SURF descriptor, which has a dimension of 64, the extended version (E-SURF) with 128 elements and the upright version (U-SURF). The U-SURF version is not invariant to rotation and has a length of 64 elements [57].

Example:

A MATLAB code that links feature points using SURF to extract and describe features of the two frames of the **Figure III.2**:

The **MATLAB Code** for this function 'Matching two frames using SURF' is giving in APPENDIX A.

Figure III.3 shows the matching result between the extracted features from the previous images. The two frames are superposed.

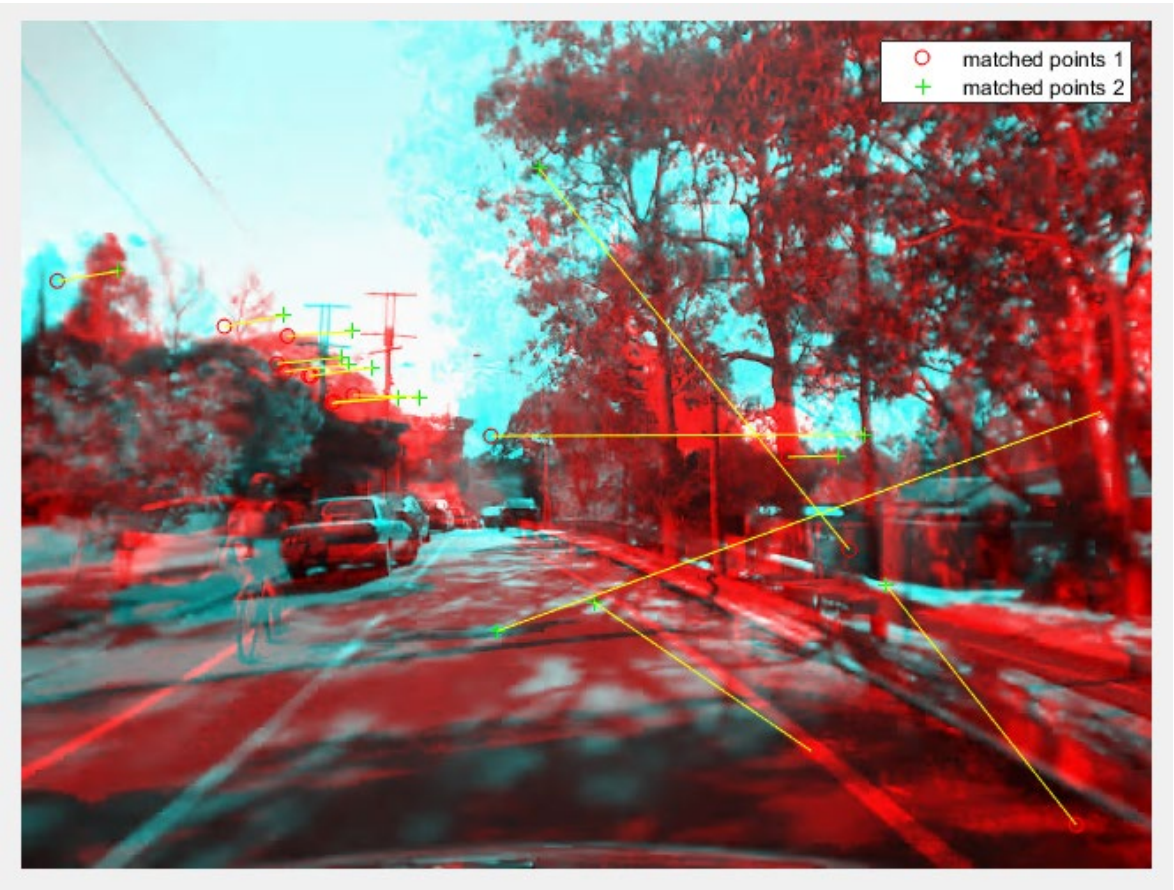


Figure III.3 : Matching features of two images using SURF descriptor.

III.4 HTM Algorithmic Methods

Hierarchical Temporal Memory is a continuous, online and unsupervised neural computation method. It achieves learning via adapting the SDR by time which describes the state.

III.4.1 Spatial Pooler Algorithm

The SP uses both of competitive Hebbian learning and homeostatic excitability control to transform random binary input patterns towards SDR. The SP has a number of important characteristics, including quick adaptability to changing input data, enhanced noise robustness through learning, efficient cell utilization, and cell death resistance [48]. This spatial pooling operation is usually split into three distinct phases that occur in sequence:

- **Phase 1:** compute the overlap with the current input for each column (overlap)
- **Phase 2:** compute the winning columns after inhibition (inhibition)
- **Phase 3:** update synapse permanence and internal variables (learning)

The initialization of the spatial pooler involves connecting each column to a random subset of input pattern bits through proximal synapses. Then, the permanence values of the proximal synapses for the columns in the region are initialized random and are closer to the synapse threshold value to ensure that the synapse is connected in the first few training iterations.

The permanence values are initialized such that half of the synapses in a column are connected and the other half are almost closer to being connected. This allows all the columns to participate in computing the input patterns.

III.4.1.1 Phase 1: Overlap

The overlap score of a column is the number of connected bits from the input to this column multiplied by its boost factor (a connected bit to a column means it has a synapses permanence above a connection threshold). If the overlap is greater than or equal to the stimulation-threshold value, it is as the overlap value for that column, and the column will be activated in the next phase (winner). Otherwise, the column's overlap value is set to zero and hence the column is inactive in the next phase [58].

The Boosting operation (multiplication by the scalar boost) increases the column's chances of participation in the next phase and penalizes the columns that are so active frequently.

Stimulation-threshold denotes the threshold for treating the input overlap as noise.

Figure III.4: shows the calculated overlaps of columns and how they are connected to the input.

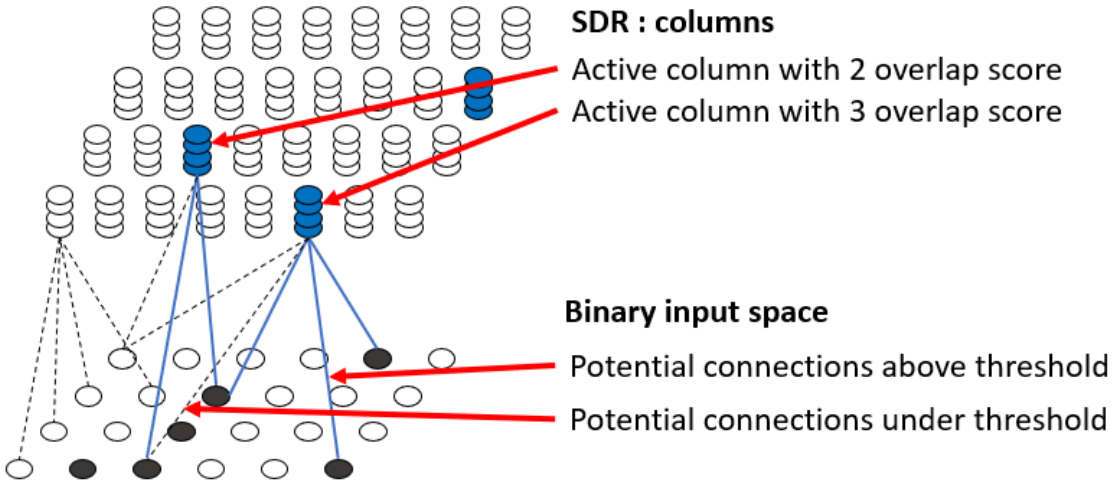


Figure III.4 : Spatial Pooler (phase 1).

- Algorithm

Algorithm 1 presents the Phase 1 pseudocode of spatial pooler.

Algorithm 1: SP - Phase 1

```
1 foreach col ∈ layer.columns do
2     col.overlap ← 0;
3     foreach syn ∈ col.proxDend.connectedSynapses do
4         col.overlap ← col.overlap + syn.active;
5     if col.overlap < stimulusThreshold then
6         col.overlap ← 0;
7     else
8         col.overlap ← col.overlap · col.boost;
```

III.4.1.2 Phase 2: Inhibition

There is competition among the columns to become active because the number of active columns is fixed at any time, and these active columns inhibit the rest [58]. The inhibition can be carried out in a global fashion meaning all the columns of a layer is compared with each other, or it can be local within a columnar neighborhood.

Figure III.5 shows the resulting active columns after applying local inhibition on the activation at Phase 1

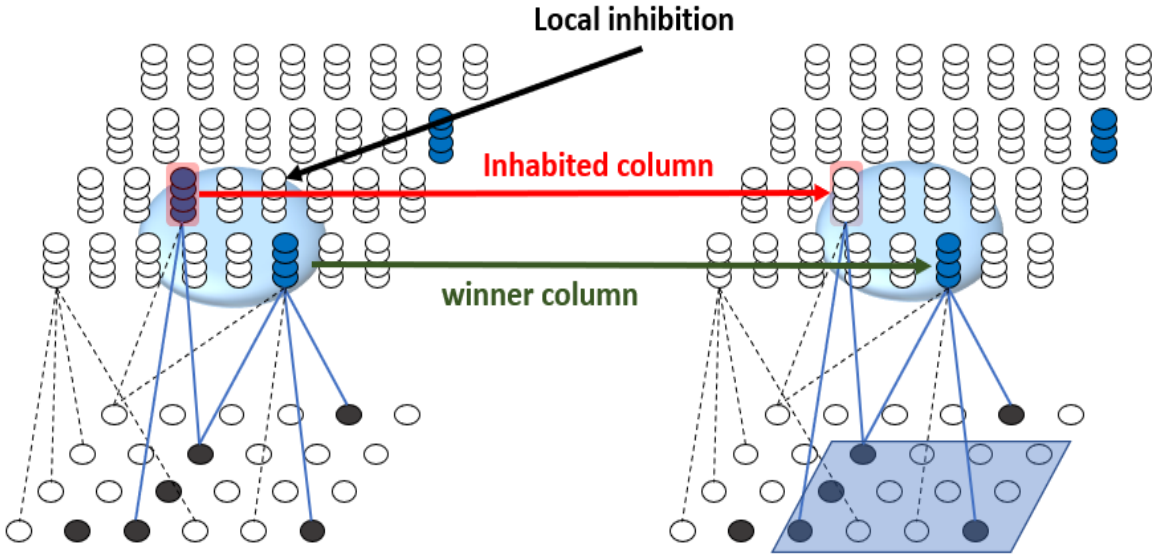


Figure III.5 : Spatial Pooler (phase 2).

- Algorithm

Algorithm 2 presents the Phase 2 pseudocode of spatial pooler. ***kmaxOverlap*** function takes all the columns as input and calculates the minimum overlap among the highest overlapping **k** columns. **k** denotes the number of active columns at a time and is calculated by multiplying the column count with sparsity (SDR of 2048 columns and sparsity of 2% means **k** should equal to 40 columns active at each time).

Algorithm 2: SP - Phase 2

```
1 foreach col ∈ layer.columns do
2     if col.overlap > k_maxOverlap(layer.columns) then
3         col.active ← true;
4     else
5         col.overlap ← 0;
```

III.4.1.3 Phase 3: Learning (adapt permanence)

In this phase, the proximal synapses of the active columns resulting after the inhibition are adapted [58]. It is important to note that the algorithm only adjusts the synapses of active columns and the rest remains the same. For all the potential synapses of a column, the synaptic permanence is incremented if the source input bit is active. Permanence is decremented otherwise. It ensures that columns specialize to a group of input patterns over time. As seen in **Figure III.6** that shows the synaptic adaptation of a proximal dendrite.

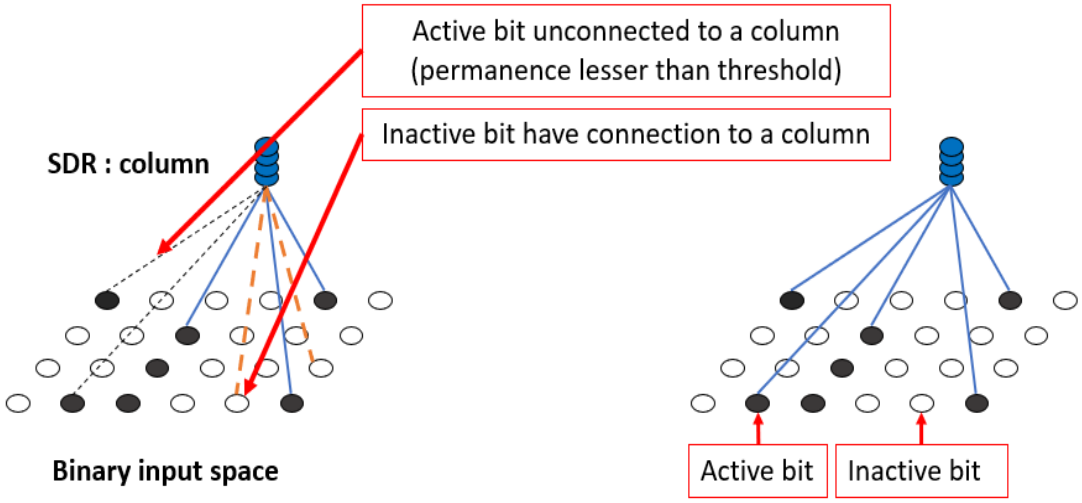


Figure III.6 : Spatial Pooler (phase 3).

Learning only involve the active columns at the moment, learning will be by incrementing and decrementing their permanence value based on the bits of current input space.

There are bumping and boosting mechanisms to imitate neural homeostasis. These provide a more uniform activation distribution among the layer.

- **Bumping:** All the proximal synapse permanences are incremented for the columns with lesser input overlap (***overlapdutycycle***). Thus, these columns are reinforced to have more connected synapses that can contribute to their overlap.
- **Boosting:** This mechanism scales the overlap score of every column with the column's boost value. This value depends on the activation frequency of the column (***activedutycycle***). Therefore, frequently activated columns have a lower boost value while lesser used columns have a higher boost value.

The activation duty cycle (***adc***) and overlap duty cycles (***odc***) of a column are updated during this phase to bump the proximal synapses of the columns and compute the boost values for the next timestep's SP - Phase 1.

- Algorithm

Algorithm 3 presents the Phase 3 pseudocode of spatial pooler.

- The function ***minOdc*** calculates the minimum overlap duty cycle among all the columns.
- ***boostStrength*** controls the intensity of leveling this distribution.
- ***period*** represents the moving average window of duty cycles with respect to time step count

Algorithm 3: SP - Phase 3

```
foreach col ∈ layer.columns do
    foreach syn ∈ col.proxDend.potentialSynapses do
        if syn.active then
            syn.permanence ←
              min(maxPermanence, syn.permanence + increment);
        else
            syn.permanence ← max(0, syn.permanence − increment);
    col.odc ← (col.odc · (period − 1) + col.overlap)/ period;
    if col.odc < minOdc(layer.columns) then
        foreach syn ∈ col.proxDend.potentialSynapses do
            syn.permanence ← syn.permanence + increment · 0.1;
    col.adc ← (col.adc · (period − 1) + col.active)/ period;
    col.boost ← exp(−boostStrength · (col.adc − localAreaDensity));
```

III.4.2 Temporal Pooler Algorithm

The output of Spatial pooler is the active columns of the SDR, these columns are used as an input to Temporal pooler stage. The temporal pooler picks which neurons to put in (active/predictive) state in the current time. By default, the distal dendrites of neurons potentially connect to the neighboring neurons of the same layer. These dendrites allow neurons to predict their activation by growing synapses to the previously active neurons. Therefore, a neuron is depolarized and put in a predictive state if its distal dendrites detect the activation that occurs before itself [59].

The depolarized neurons are primed to fire in the following time step. If a depolarized neuron becomes active, it inhibits the rest of the neurons in the same column. All the neurons become active if there is no depolarization at the same column (bursting). The resulting neural activation computed on this stage represents the contextual information of the columnar activation. As a result, the same columnar activations may have different active neurons based on the previous activations.

The pseudocode is split into four Temporal Memory phases that are explained above to run sequentially learning in an HTM algorithm.

- **Phase 1:** Compute the active state of each cell.
- **Phase 2:** Learning Segments Assignment.
- **Phase 3:** Segment Adaptation.
- **Phase 4:** compute the predicted state of each cell.

Phase 3 is only required for learning. However, unlike spatial pooling, Phases 1 and 2 contain some learning-specific operations when learning is turned on. Since temporal pooling is significantly more complicated than spatial pooling, it starts with listing the inference-only version of the temporal pooler, followed by a version that combines inference and learning. A description of the phases and the pseudocodes are provided in the following subsections:

III.4.2.1 Phase 1: Active State Cells

The first phase would activate all the cells of a column if that column did not have any depolarized cells from the previous time step (bursting). If there are any depolarized cells of an active column, only those cells get activated, and the rest of the column is inhibited.

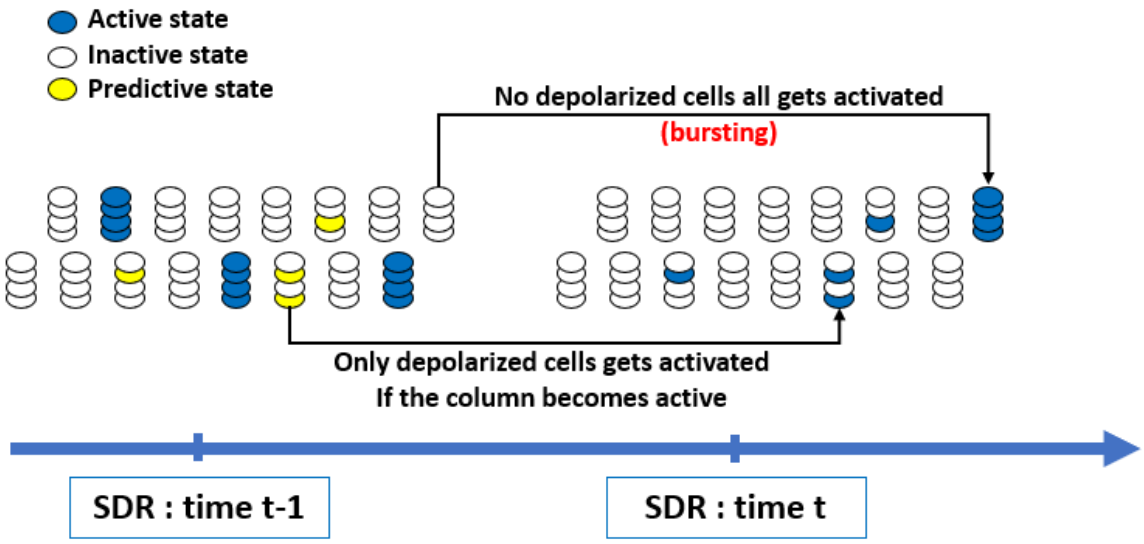


Figure III.7 : Temporal Pooler (phase 1).

- Algorithm

Algorithm 4 presents the Temporal pooler - Phase 1 pseudocode.

Algorithm 4: TM - Phase 1

```
1 foreach cell ∈ layer.predictiveCells do
2     if cell.parentCol.active then
3         cell.active ← true;
4         cell.parentCol.predicted ← true;
5 foreach col ∈ layer.activeColumns do
6     if !col.predicted then
7         foreach cell ∈ col.neurons do
8             cell.active ← true;
```

III.4.2.2 Phase 2: Learning Segments Assignment

Temporal pooler should pick a cell in a column to express the temporal context. Therefore, if a column burst means all cells within the column will be active. For solving this problem, the idea of a winner cell came up: the spatial pooler algorithm selected a column to be active leads to tow different cases:

- If there is a correctly predicted cell in the column that cell is automatically the winner cell.

- If there is no predicted cell in the column (bursting case): investigate through potential synapses of a distal segment.
    - if any of the cells was almost about to be predicted, means that the cell has distal segments that match the previously active cells but their permanent values are not high enough to form a connection, set this as a winner cell.
    - if there were no matching segments, means none of the cells in the current column have any segments that match any previous winner cells so the SDR have no clue what's going on, in such a case this will be considered as a new sequence. Therefore, the algorithm should find a cell with the fewest number of segments and make that as winner cell this ensures that the SDR is utilizing as many cells as possible.

As seen in **Figure III.8** shows two time-steps of the SDR, and the three mentioned cases first busting with no clue, busting with almost predicted cell and successfully predicted cells get activated.

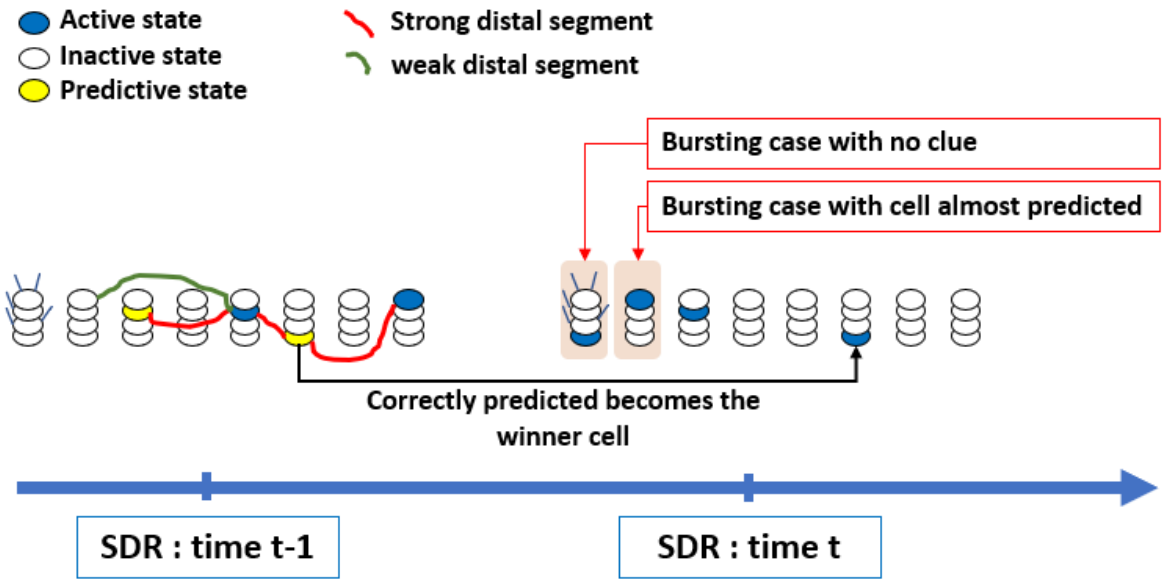


Figure III.8 : Temporal Pooler (phase 2).

- Algorithm:

Algorithm 5 presents the Temporal pooler - Phase 2 pseudocode.

The main objective of this phase is to assign learning segments to active cells. A learning segment is a flagged segment which is allowed for synaptic adaptation on Temporal pooler - Phase 3.

For a given cell, the function ***bestMatchingSegment*** finds the matching segment that has the highest overlap with the previous activation. The resulting segment is chosen as the learning segment for that bursting cell. If there are no matching segments, a new empty segment is created via ***createSegment*** function and set as the learning segment.

Algorithm 5: TM - Phase 2

```
1  foreach cell ∈ layer.predictedCells do
2      foreach seg ∈ cell.segments do
3          if seg.prevActive
4              seg.learning ← true;
5  foreach cell ∈ layer.burstingCells do
6      bestSeg ← bestMatchingSegment(cell);
7      if bestSeg
8          bestSeg.learning ← true;
9      else
10         newSeg ← cell.createSegment();
11         newSeg.learning ← true;
```

III.4.2.3 Phase 3: Segment Adaptation

The synapses of the learning segments are adapted using the same logic in SP – Phase 3 with a single difference; the synapses sample the previous input activity, not the current activity. Therefore, a synapse is strengthened if the source was active in the previous time step and weakened otherwise. Temporal Memory has its own permanence increment and decrement parameters that are different from Spatial Pooler.

- Algorithm:

Algorithm 6 presents the Temporal pooler - Phase 3 pseudocode.

Algorithm 6: TM - Phase 3

```
1  foreach seg ∈ layer.learningSegments do
2      foreach syn ∈ seg.potentialSynapses do
3          if syn.prevActive then
4              syn.permanence ←
                 min(maxPermanence, syn.permanence + increment);
5          else
6              syn.permanence ← max(0, syn.permanence − increment);
```

III.4.2.4 Phase 4: Depolarization

This is the phase where the HTM layer outputs its prediction. The active neurons resulting from Temporal pooler – Phase 1 stimulate all their postsynaptic targets. Every neuron of the HTM layer is iterated to check whether it can predict its own activity on the following time step using the current neural activation. A neuron is depolarized if it has a distal segment with an input overlap that exceeds the activation threshold. The depolarized cells represent the union of all neural activations that may happen at the next time step based on the temporal context.

- Algorithm

Algorithm 7 presents the Temporal pooler – Phase 4 pseudocode.

Algorithm 7: TM - Phase 4

```
1 foreach col ∈ layer.columns do
2     foreach cell ∈ col.neurons do
3         foreach seg ∈ cell.segments do
4             foreach syn ∈ seg.connectedSynapses do
5                 seg.overlap ← seg.overlap + syn.active;
6             if seg.overlap >= activationThreshold
7                 seg.active ← true;
8                 seg.parentCell.predictive ← true;
```

III.5 Scoring Recognitions

The definition of recognized place is a point or multiple ones that occurs in time where the system notices them from its memory.

Recognition can be in spatial pattern, where an individual data instance is considered known before, independent of where it occurs in the data stream, the recognition can also be in temporal pattern, if the temporal sequence of data is relevant. Detecting temporal known patterns in practical applications is valuable as they can serve in many tasks.

Based on known properties of cortical neurons, Hierarchical Temporal Memory (HTM) is a theoretical framework for sequence learning in the cortex [60]. HTM implementations operate in Real time and have been shown to work well for prediction tasks [49]. HTM networks continuously learn and model the spatiotemporal characteristics of their inputs, but they do not directly model recognitions and do not output a usable recognition score. This section describes the proposed technique for applying HTM to detect recognized data.

Figure III.9 shows an overview of the process. At each point in time, the input data is fed to a HTM network. Therefore, an additional calculation on the output of the HTM is needed to specify what data is recognized by the HTM.

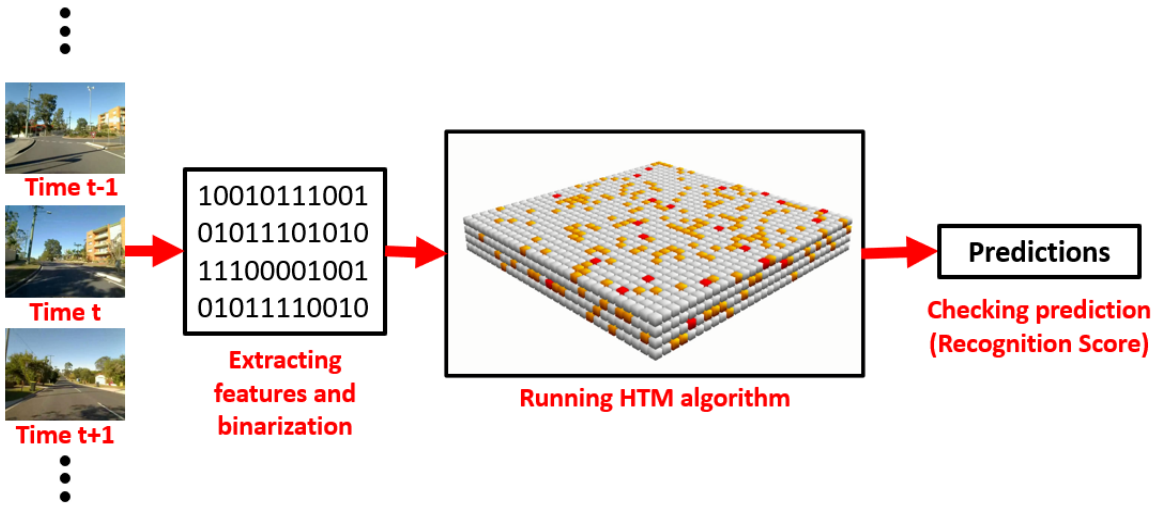


Figure III.9 : General architecture of VPR using HTM.

Firstly, compute a measure of prediction error ***RecognitionScor***e of each single input. Then, using a probabilistic model of it, to compute ***RecognitionLikelihood*** of each sequence, a likelihood means that the system is in a sequence of a known input. A threshold on this likelihood determines whether an anomaly or a known sequence is detected [61]. In the following subsections provides a description of the techniques for the additional steps of computing the prediction error and recognition likelihood:

III.5.1 Computing the recognition score

Given the current input x_t, $SP(x_t)$ is a sparse encoding of the current input, and $P(x_{t-1})$ is the sparse vector representing the HTM network's internal prediction of a $SP(x_t)$. The dimensionality of both vectors is equal to the number of columns in the HTM network. Let the prediction error is the recognition score Rs(t), be a scalar value inversely proportional to the number of bits common between the actual and predicted binary vectors:

$$Rs(t) = \frac{P(x_{t-1}).SP(x_t)}{|SP(x_t)|} \qquad \text{III.1}$$

Where | $SP(x_t)$ | is the scalar norm, i.e., the total number of one-bits in $SP(x_t)$. In Eq. III.1 the error $Rs(t)$ will be 1 if the current $SP(x_t)$ perfectly matches the prediction, and 0 if the two binary vectors are orthogonal (they share no common 1 bits). $Rs(t)$ gives an instantaneous measure of how well the underlying HTM model predicts the current input x_t.

Changes to the underlying statistics are handled automatically due to the continuous learning nature of HTMs [49]. If there is a shift in the behavior of the system, the recognition score will be low at the point of the shift, but will automatically start recognize that due to the model fast adaption.

In HTM network, multiple predictions are represented in $P(xt)$ as a binary union of each individual prediction. If two completely different inputs are both possible and predicted, receiving either input will lead to a 100% recognized. Any other input will generate a positive error.

III.5.2 Computing the recognition likelihood

The prediction error described above represents an instantaneous measure of the predictability of only the current input stream [61]. To handle a sequence of inputs rather than thresholding the recognition score $Rs(t)$ directly, the second step is introduced to model the distribution of error values as an indirect metric, and use this distribution to check for the likelihood that the current sequence is recognizable. The recognition likelihood is thus a probabilistic metric defining how much is known the current state is based on the prediction history of the HTM model. To compute the recognition likelihood, a window of the last W inputs predictions error values is used. The distribution is modeled as a rolling normal distribution[2] where μ_t is the sample mean, and σ_t^2 is the variance, are continuously updated from previous error values as follows:

[2] The distribution of prediction errors is not technically a normal distribution, but in this research is considered as it is normal distribution. A normal distribution (also known as Gaussian distribution) is a type of continuous probability distribution for a real-valued random variable.

$$\mu_t = \frac{\sum_{i=0}^{i=W-1}(1-Rs_{t-i})}{W} \quad \text{III.2}$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1}(1-Rs_{t-i}-\mu_t)^2}{W-1} \quad \text{III.3}$$

Then, a threshold is applied to the Gaussian tail probability[3] (Q-function [55]) to decide whether or not to declare a recognition. Finally, the recognition likelihood as the complement of the tail probability:

$$\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1}(1-Rs_{t-i})}{W'} \quad \text{III.4}$$

$$Rl(t) = Q\left(\frac{\tilde{\mu}_t-\mu_t}{\sigma_t}\right) \quad \text{III.5}$$

W' is a window for a short-term moving average, where $W' < W$, the duration for computing the distribution of prediction errors. Now a threshold can be used to report what is known, as recognized sequence $Rl(t) \geq 1-\varepsilon$.

It is important to note that $Rl(t)$ is based on the distribution of prediction errors, not on the distribution of underlying metric values xt. As such, it is a measure of how well the model is able to predict, relative to the recent history.

[3] The tail probability is the probability that a variable will be larger than x standard deviations above the mean.

III.6 Conclusion

To conclude, time-series data largely driven by the rise of connected real-time data sources, this data presents technical challenges and opportunities. One fundamental capability for streaming analytics is to model each stream in an unsupervised fashion. Application constraints require systems to process data in real- time, not batches. This chapter presents the proposed approach for recognized streams detection algorithm for the sake of solving the problem of visual place recognition. The technique is based on an online sequence memory algorithm called Hierarchical Temporal Memory that is capable of detecting spatial and temporal pattern in predictable way. The algorithm meets the requirements of real-time, continuous, online detection without any supervision.

Therefore, the approach proposed was presented in this chapter, starting by proposing a method of describing and treating the visual data, then inserting it into the HTM program which was presented precisely, finally a proposition of a probabilistic method for treating the output of the last step to decide whether the currant sequence is known or not, basically this means recognizing a visual place from the past.

Chapitre IV : HTM-VPR

Implementation and Evaluation

IV.1 Introduction

Hierarchical Temporal Memory (HTM) is a special type of artificial neural network (ANN), that differs from the widely used approaches. It is suited to efficiently model sequential data as mentioned early. The network receives raw information from the dataset; thus, a signal processing method is needed to convert the raw input into useful information. The whole architecture consists of four separable units, which are: extracting and describing features, binarization, running artificial network of HTM and scoring recognitions (see chapter III for more details). All the activations inside the HTM network (big array) are done automatically, the user can not choose which cell will be active. In this chapter, provides MATLAB codes of each step **Figure IV.1**. Furthermore, the HTM's performance will be evaluated in terms precision and recall curve a widely used method to evaluate VPR systems.

This chapter presents the codes that uses HTM for VPR in MATLAB. Therefore, it will discuss all its steps. The next section (IV.2) describes the dataset that is going to be treated by the proposed system. This section includes also features extraction, description using SURF descriptor and binarization. Section (IV.3) shows how the binarized inputs flow through the artificial neural network of HTM and its modules. Section (IV.4) presents the Scoring Recognitions MATLAB code, following the theory given in chapter III. The section (IV.5) discusses the evaluation of the proposed system. Finally, in the section (IV.6) a conclusion.

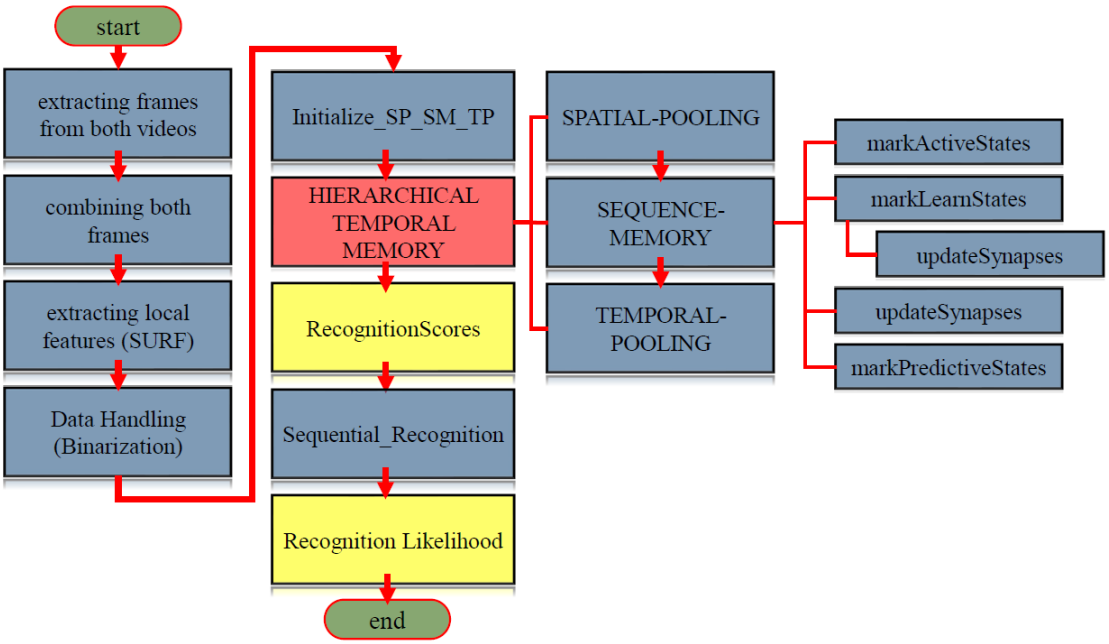


Figure IV.1 : Organization of the whole developed functions.

IV.2 Data choosing and treating

The data considered in this project is composed of two videos, were filmed by a real car through streets in Saint Lucia at different time of a day and month. This dataset is hard to handle because the different speed of the car may change the sequence of frames (Same place but because of the change in speed it may jump some of the frames lead into a different sequence), the data is so long, therefore, the car may see the same sequence from a different place.

The data has the following details:

- Multiple loops (24 crossroads).
- The two videos were filmed in different time of day and month.
  - **First:** taking in 19/08/2009 at 08:45 with 21815 frames.
  - **Second:** taking in 11/09/2009 at 15:45 with 20127 frames.
- The length of the videos:
  - **First:** 24 minutes and 14 seconds with 21815 frames.
  - **Second:** 22 minutes and 22 seconds with 20127 frames.

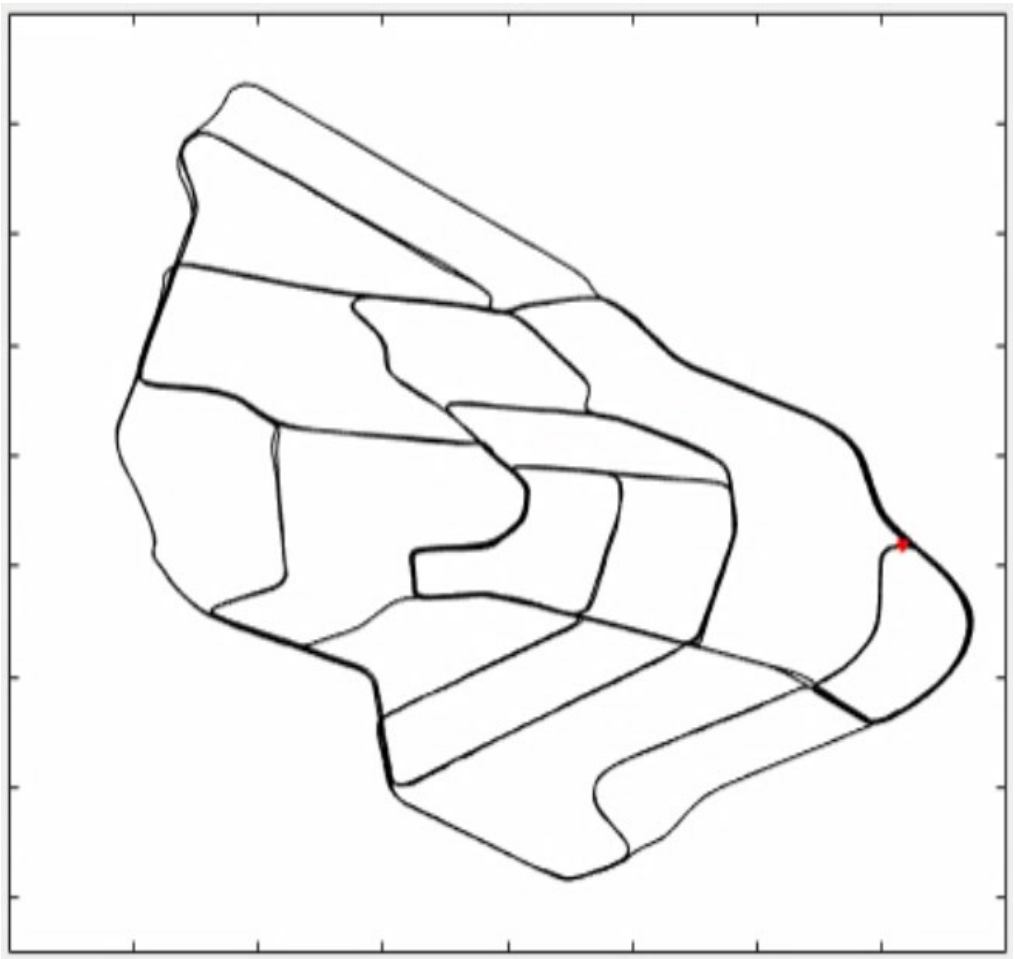


Figure IV.2 : The path followed by the car with multiple loops.

IV.2.1 Extracting frames from the videos

The MATLAB code extract frames from each video and store frames of the same video in the same folder with the name of its order in the video.

The **MATLAB Code** for this function 'Extracting frames' is giving in APPENDIX A.

IV.2.2 Combining frames

The MATLAB code combine frames from each video and rename the frames so when inserting it into the HTM network will read it as a long video.

- ***First:*** keep frames $1 \rightarrow 20038$, and remove the rest.
- ***Second:*** rename frames $6073 \rightarrow 21815$, to be $20039 \rightarrow 35871$,and remove the rest.

The **MATLAB Code** for this function 'Combining both frames' is giving in APPENDIX A.

- Frame between the Combined sequences:

The frame in between the combined frames was chosen because it is in the same place with some challenging changes to be recognized by HTM network.



Figure IV.3 : The above picture shows the frame between the combined.

IV.2.3 Extracting features from frames

The MATLAB code extracts and describes features by using **SURF** from each frame and store them in a new variable named **final features**. More details about SURF are in chapter III.

The **MATLAB Code** for this function 'Extracting local features (SURF)' is giving in APPENDIX A.

- Between the combined sequences:

The following MATLAB code plots the frame in between the combined sequences with its local features highlighted.

The **MATLAB Code** for this function 'Between the combined' is giving in APPENDIX A.



Figure IV.4 : The above picture shows what the **MATLAB Code IV—5** plots.

MATLAB Code IV— (4, 5) has the function ***detectSURFFeatures*** which is included in MATLAB that returns a **SURF Points** object, points, containing information about SURF features detected in the 2-D grayscale input image. The ***detectSURFFeatures*** function implements the Speeded-Up Robust Features (SURF) algorithm to find blob features.

IV.2.4 Binarization

To perform SLAM, the ability to recognize a previously visited place is a fundamental task [1]. Therefore, place recognition systems must build and keep a consistent map of the environment by comparing incoming data with representations that are already included in the map. Thus, the GPS information is added to the final binary code, the **final feature** vector coming from the last step has 640 bits to binarize it, each bit will be coded into 4 binary bits means the final vector that represent the image has 2560 bit, in addition, GPS pose is coded into 9 binary bits.

The final representation of the frame has 2575 binary bits, 6 zeroes are added to only split the information. The help that GPS provides is equal to 0,3% which is so low. $\frac{9}{2575} = 0{,}003$

The **MATLAB Code** binarize the **final features** of the last step and adds GPS information is 'Features Binarization' giving in APPENDIX A.

IV.3 Running the ANN of HTM

HTM network has three main modules [62]:

- **Spatial pooler:** to create sparse representations of the input
- **Sequence memory:** to record the temporal order
- **Temporal pooler:** to create a stable representation of each sequence.

Currently, implementations of the HTM network exist in Python [63], C++ [64], and Java [65], but they are dependent on many packages. The above-mentioned modules are taking from [62], which presents a standalone MATLAB implementation under the title MatlabHTM: A sequence memory model of neocortical layers for anomaly detection, for testing on the Numenta Anomaly Dataset (NAB) [61] converted from python NUPIC [63] into MATLAB relying on these references [59] [60] [66]. In this work, only the three modules SP SM TP are taken from [62] to be used for building HTM network to treat the problem of visual place recognition, in this dissertation the SP and TP involve learning. **Figure IV.1** describes the organization of the whole developed functions in the network from treating the input and binarizing it, then followed up by the HTM modules, in the end scoring recognitions.

Before inserting the input into the HTM, the HTM should be initialized in the main script.

0) Initialization: This function has two main purposes

- Sets up the parameters of the tree modules (thresholds, rewards, permaneces, etc.)
- Sets up the arrays of the three modules (array size, its content)

The **MATLAB Code** for this function is giving in [62].

1) The spatial pooler (SP): This function takes the binary semantic representation of each input and turns it into an SDR using Hebbian learning, as outlined by Hawkins et al. [66].

The **MATLAB Code** for this function is giving in [62].

2) The sequence memory (SM): builds sequential correlations between consecutive measurements. This is the core of the HTM's functionality. It can identify sequences such as A-B-C-D as well as X-B-C-Y from the example **Figure II.12** page (34). Each cell can be in three states: not active (OFF), active (ON), or predictive. SM uses four operations to build high order representations.

- **Mark-Active-States:** identifies a cell as active when the corresponding bit input from the spatial pooler is ''ON''.
- **Mark-Predictive-States:** sets a cell to predictive if its dendrites meet a threshold of ''ON'' synapses from the currently active cells.
- **Mark-Learn-States:** selects the cells that will be learning.
- **Update-Synapses:** simulate the strengthening and weakening of connections by engaging the active synapses, as it happens in the brain. Through Hebbian learning, the permanence increases for each synapse within all the cells marked to learn and decreases for all erroneously predicted cells.

The **MATLAB Code** for this function is giving in [62].

3) **The temporal pooler (TP)** builds a steady SDR for each sequence learned. To create a module that merges the different spatial features of a sequence into a uniform pattern. Therefore, the function receives both overlaps of active and predictive cells then sum their weighted averages to reject similar spatial semantics within different sequences.

This function developed by [62] relying on the information in Bolliger's overview [67] and tested in Python.

The **MATLAB Code** for this function is giving in [62].

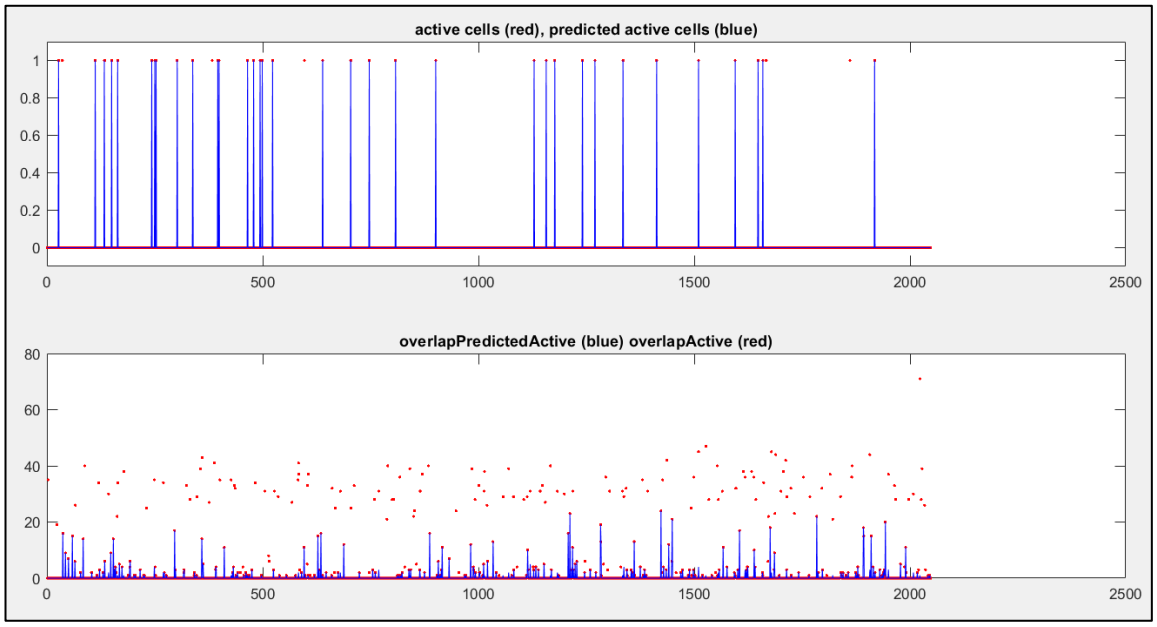


Figure IV.5 : Illustration of what the temporal pooler plots.

The above plot is divided into two separate diagrams:

- The first: shows SDR with active cells (red), predicted active cells (blue)
- The second: shows overlap-Active (red), overlap-Predicted-Active (blue)
  (Overlap-Active number of active synapses for each dendrite)

All the modules and the necessary functions explained above are called in the main script as shown in **MATLAB Code IV—7** which outputs Recognition-Score of each frame $R(t)$ that was highlighted in equation III.1 page 54, to be used in the next step.

The **MATLAB Code** for the main script that calls the three modules is giving in APPENDIX B.

IV.4 Scoring Recognitions

The computed **Recognition-Scores** by comparing the prediction with the SP output, that left out the HTM network system is needed for calculating the **Recognition-Likelihood** as cited in equation III.5 from the page 55. A **Sequential-Recognition** function is needed to detect changes in between predictable and unpredictable scenarios through a shifting normal distribution [61], more details are in chapter III.

The MATLAB code plot Recognition Scores of each frame and Recognition Likelihood of each sequence.

The **MATLAB Code** for this function 'Recognition likelihood' is giving in APPENDIX B.

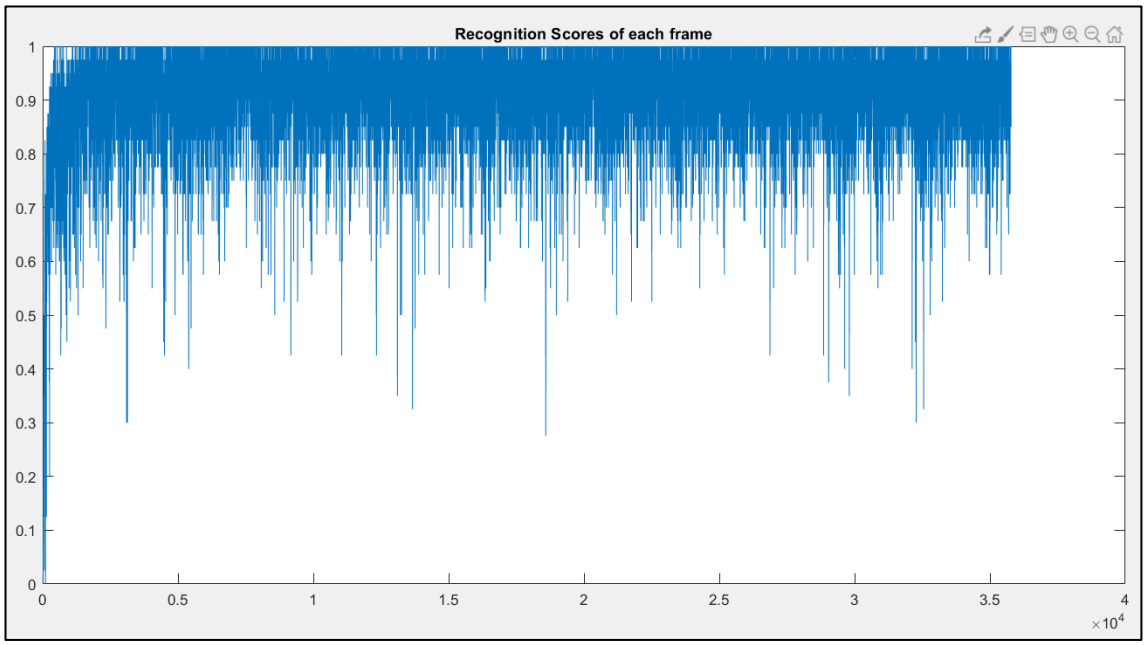


Figure IV.6 : Recognition Scores of each frame from **MATLAB Code IV—8**.

The above picture shows recognition scores of each frame while **TP.historyLength** is set to 100, and it is between 30% to 100%. In **Figure IV.7** shows how the plot is between 80% and 100% while **TP.historyLength** is set to 500.

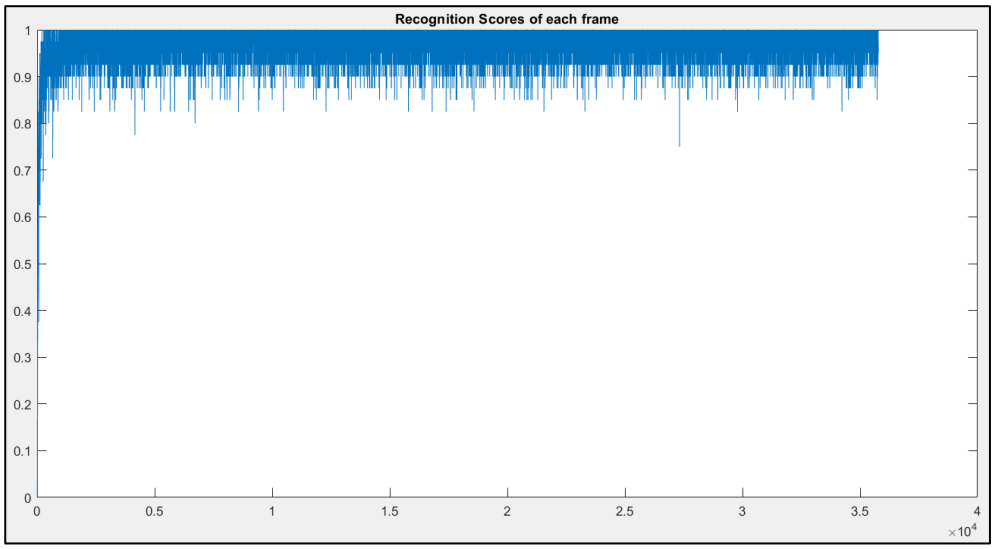


Figure IV.7 : Recognition Scores of each frame from **MATLAB Code IV—8**.

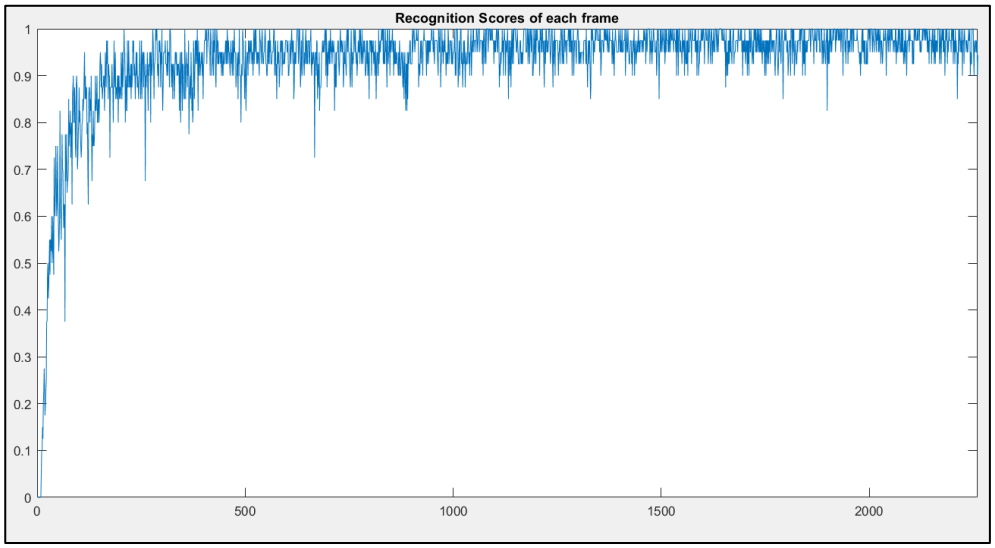


Figure IV.8 : Zoom in **Figure IV.7** from **MATLAB Code IV—8**.

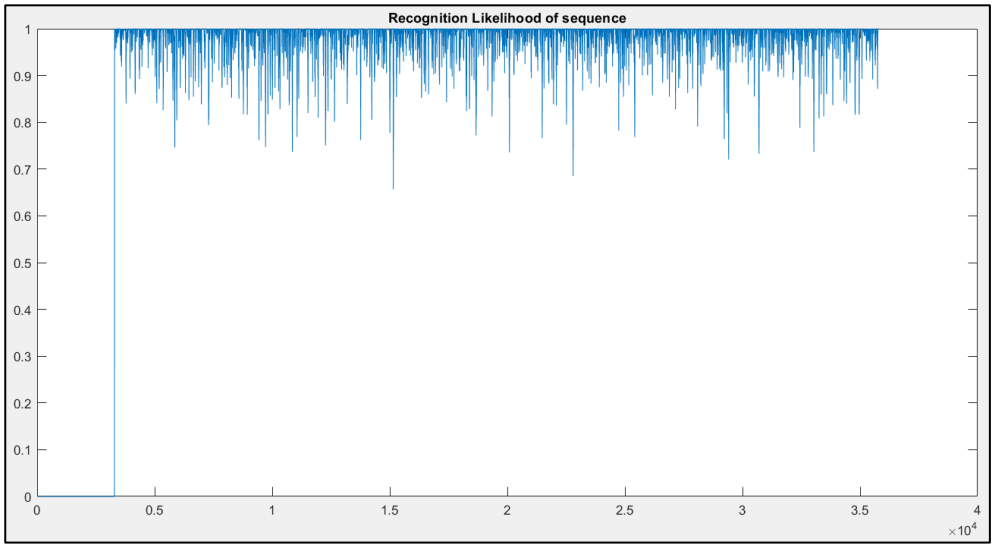


Figure IV.9 : Recognition Likelihood of sequence from **MATLAB Code IV—8**.

The above picture shows Recognition Likelihood which represent recognition of sequences while **TP.historyLength** is set to 500. Clearly there is a jump in begging of the plot that is because of code ignores that period until recognition scores settle as shown in **MATLAB Code IV—9**.

MATLAB Code IV—8 is in need of Sequential Recognition function the input of it, is a vector of raw scores from HTM, and a window (the size of the sequence).

The **MATLAB Code** for this function 'Sequential Recognition' is giving in APPENDIX B.

A MATLAB code plots a simulation of a moving car where if it appears as green means the car is in a known sequence, if it is red then it is in an unknown sequence. Visited is a vector with the length of number of frames.

- Visited place before, $visited(i) = 1$
- Unvisited place before, $visited(i) < 1$

The **MATLAB Code** for this function 'Moving car' is giving in APPENDIX B.

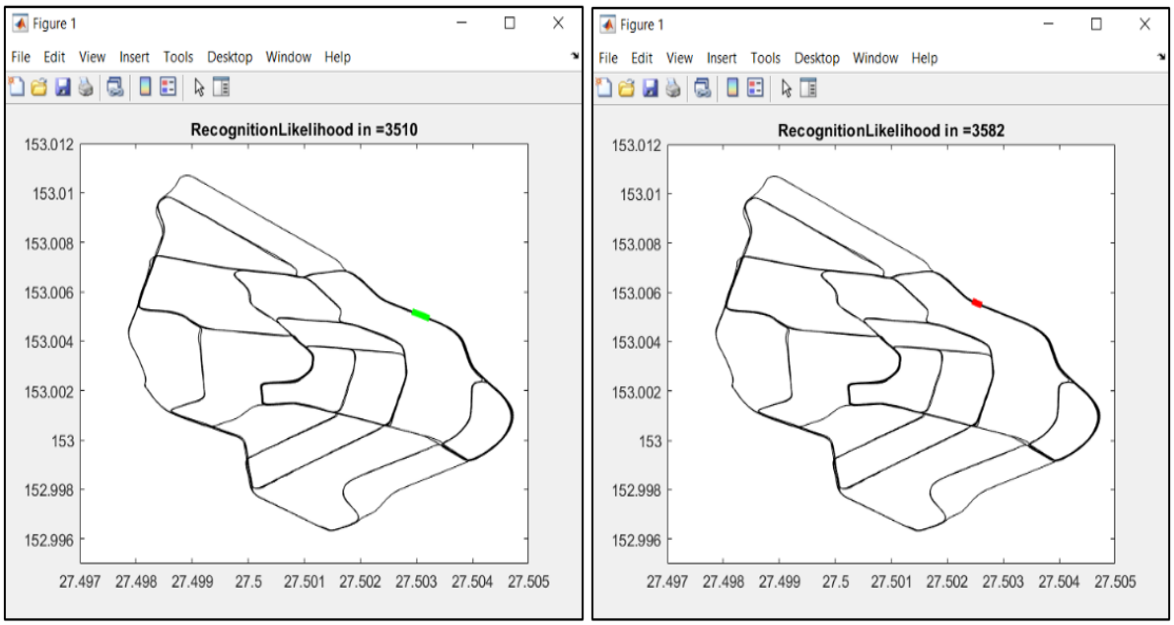


Figure IV.10 : A screenshot of the video from **MATLAB Code IV—10**.

IV.5 Evaluation of the proposed approach

Place recognition systems are typically evaluated using precision and recall metrics and their relationship via a precision-recall curve [1]. A system selects matches based on a particular confidence measure (in this case is Recognition-Scores). Precision is defined as the proportion of selected matches that are true positive matches, and recall is the proportion of true positives to the total number of actual matches, that is

$$Precision = \frac{True\ Positives}{Predicted\ True} = \frac{TP}{TP+FP} \quad \text{IV.1}$$

$$Recall = \frac{True\ Positives}{Actual\ True} = \frac{TP}{TP+FN} \quad \text{IV.2}$$

A perfect system would be one that achieves precision of 100% and recall of 100%. Precision and recall are often related to each other via a precision–recall curve, which plots recall against precision for a range of confidence values [1]. This curve is based on some measures (Table IV):

TP (True Positive): number of correctly recognized places,

TN (True Negative): number of correctly detected new places (without confusion with other places),

FP (False Positive): number of revisited places that are incorrectly recognized.

FN (False Negative): number of revisited places but not recognized.

	REALITY	
HTM	TP True positives	FP False negatives
	FN False negatives	TN True negatives

Table IV.1 : The necessary measurement for Precision-recall evaluation

The MATLAB code calculate and draws the curve of the relation between precision and recall.

The **MATLAB Code** for this function 'System Evaluation' is giving in APPENDIX B.

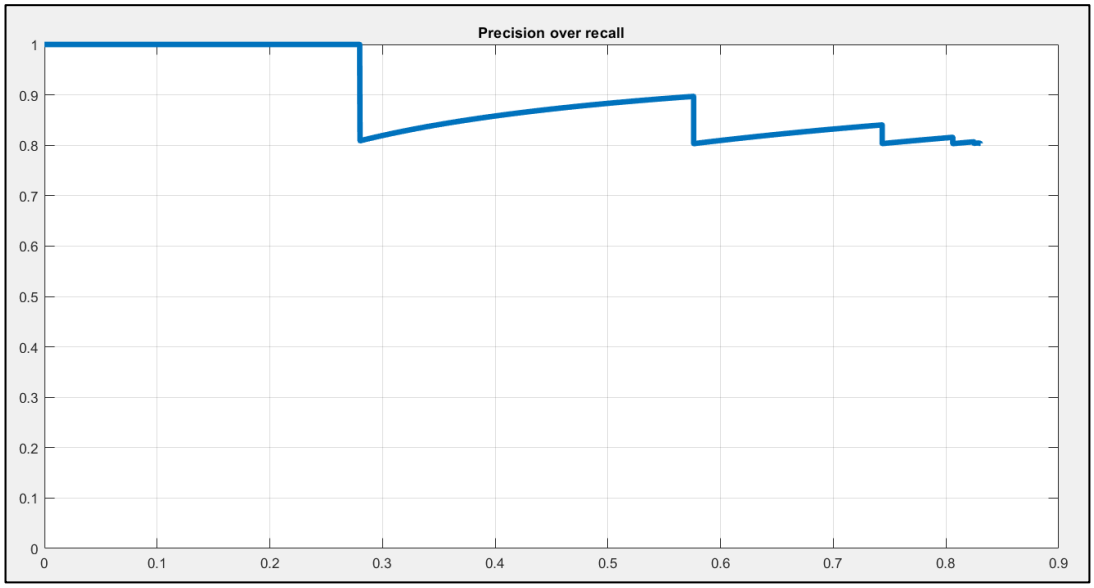


Figure IV.11 : Precision-recall curve of the HTM-based VPR.

The above picture shows precision-recall curve which represent the tradeoff between precision and recall for **TP.historyLength=500**. The system has 100% precision up to 29% recall, the precision is always above 80%.

IV.6 Conclusion

This chapter went through the whole MATLAB codes for implementing a visual place recognition system using a Hierarchical Temporal Memory (HTM) model. HTM is originally an artificial neural network based on neuroscience research focused on the structure of the neocortex using mini-column networks of neurons and its main modules are spatial pooler, sequence memory temporal pooler. All this has been explained including the MATLAB codes, plots, examples, etc.

Also, the proposed system has been evaluated by Precision-Recall method which is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

The general conclusion

In this work, Hierarchical Temporal Memory theory was applied, a recently developed neural network model inspired from human brain, to real-time sequence learning problems with time-varying input streams of Visual Place Recognition, which is the main heart of Simultaneous Localization and Mapping robots. The model satisfies a set of properties that are important for online sequence learning from noisy data streams with continuously changing statistics, a problem the cortex has to solve is dynamic maps of where the robot is navigating. This work showed that HTM algorithm has a strong flexibility and its ability to be used in an automated fashion. Although HTM is still at a very early stage compared to other traditional neural network models, it satisfies these properties and shows promising results on real-time sequence learning problems.

Also, this dissertation presented the development of software using MATLAB from the very beginning to the end, passing through treating data, extracting features, describing and binarizing them into vector. Then running the Hierarchical Temporal Memory Artificial Neural Network to handle the visual input data of the Visual Place Recognition, using recognizing spatial pattern and temporal pattern, finally using a probabilistic theory for categorizing the classes of known place or not.

The objective that was set before this study of solving Visual Place Recognition problems using Hierarchical Temporal Memory has been reached. Therefore, there is some recommendations, a future work of the same topic can be improved by adding two cameras like one on the left and the other on the right so if the three of them indicates a recognized place means its highly correct, even maybe choosing Convolutional Neural Networks for describing the input data like alexnet-conv3 or NetVLAD can improve the performance of the system, since they are showing great success in perception tasks.

Appendices

APPENDIX A:

```
filename1 = strcat('DATA/01/frames/287.jpg');
filename2 = strcat('DATA/02/frames/341.jpg');
img1 = rgb2gray(imread(filename1));
img2 = rgb2gray(imread(filename2));
points1 = detectSURFFeatures(img1);
points2 = detectSURFFeatures(img2);
[features1, valid_points1] = extractFeatures(img1, points1);
[features2, valid_points2] = extractFeatures(img2, points2);
indexPairs = matchFeatures(features1,features2) ;
matchedPoints1 = valid_points1(indexPairs(:,1));
matchedPoints2 = valid_points2(indexPairs(:,2));figure;
showMatchedFeatures(img1,img2,matchedPoints1,matchedPoints2);
legend('matched points 1','matched points 2');
```

MATLAB Code III—1 : Matching two frames using SURF from the main code.

```
VR1=VideoReader('DATA/01/video.avi');
for img = 1:VR1.NumberOfFrames;
    Iv1 = read(VR1, img);
    filename=strcat(num2str(img),'.jpg');
    In1 = ['DATA/01/frames/' filename];
    imwrite(Iv1,In1);
end
VR2=VideoReader('DATA/02/video.avi');
for img = 1:VR2.NumberOfFrames;
    Iv2 = read(VR2, img);
    filename=strcat(num2str(img),'.jpg');
    In2 = ['DATA/02/frames/' filename];
    imwrite(Iv2,In2);
end
```

MATLAB Code IV—2 : Extracting frames from videos from the main code.

```
srcFiles = dir('*.jpg');
length(srcFiles)
zero = 20038;
for id = 1:length(srcFiles)
    newname = num2str(zero+id);
    [~, old_name,ext] = fileparts(srcFiles(id).name);
    rename = strcat(newname,ext);
    movefile(srcFiles(id).name, rename);
end
```

MATLAB Code IV—3 : Combining both frames from the main code.

```
srcFiles = dir('DATA/combined/frames/*.jpg');
for i = 1:length(srcFiles)
    filename = strcat('DATA/combined/frames/',srcFiles(i).name);
    gray_img = rgb2gray(imread(filename));
    points = detectSURFFeatures(gray_img);############# SURF
    [features, valid_points] = extractFeatures(gray_img,...
        points.selectStrongest(10));
    [rf cf]=size(features);
    for n=1:rf*cf
         finalfeatures(n,i)=features(n);
    end
    if (i==15000)
        save('finalfeatures10.mat','finalfeatures')
    elseif(i==length(srcFiles))
        save('finalfeatures10.mat','finalfeatures')
    end
end
```

MATLAB Code IV—4 : Extracting local features (SURF) from the main.

```
filename1 = strcat('DATA/01/frames/20039.jpg');
filename2 = strcat('DATA/02/frames/6073.jpg');
gray_img1 = rgb2gray(imread(filename1));
gray_img2 = rgb2gray(imread(filename2));
points1 = detectSURFFeatures(gray_img1);############# SURF
points2 = detectSURFFeatures(gray_img2);############# SURF
[~, valid_points1] = extractFeatures(gray_img1, points1);
[~, valid_points2] = extractFeatures(gray_img2, points2);
subplot(1,2,1),imshow(imread(filename1)); hold on;
plot(valid_points1.selectStrongest(50),'showOrientation',true);
title('img 20039,from the first video');
subplot(1,2,2),imshow(imread(filename2)); hold on;
plot(valid_points2.selectStrongest(50),'showOrientation',true);
title('img  6073,from the second video');
```

MATLAB Code IV—5 : Between the combined from the main code.

```
V = finalfeatures;
H = V';
A = H-min(min(H));
A = 12*A;
[rr cc]=size(A);
for j=1:cc
    for i=1:rr
        Binary_input(i,4*j-3:4*j)=num2str(dec2bin(A(i,j),4))-'0';
    end
end
into_SP_10f = Binary_input;
center = [27.5005 153];
for i=1:35781
    distance(i,1) = 47000*norm(center - D4(i,:));
end
for i=1:35781
    gps_coded(i,:) = num2str(dec2bin(max(distance(i)),9))-'0';
end
into_SP_10f_GPS = [gps_coded zeros(35781,6) into_SP_10f];
```

MATLAB Code IV—6 : Features Binarization.

APPENDIX B:

```
%% HIERARCHICAL TEMPORAL MEMORY
%%
clc;clear all;close all;

global  SP SM TP data frame

data.All_inputs = importdata ('into_SP_10f_GPS.csv');
[data.N_inputs, data.N_input_bits]=size(data.All_inputs);

initialize_SP_SM_TP;

SM.inputPrevious = zeros(SM.N, 1);
data.inputSDR = [];
predictions = [];
SP.boost = ones (SM.N, 1);

fprintf('\n frame = %d --> %d', 1, 499);
for frame = 1:data.N_inputs
        if mod(frame,500)==0
            fprintf('\n frame = %d --> %d', frame, frame + 499);
        end
    %% SPATIAL-POOLING BEGINS HERE:
        %%
        x = data.All_inputs(frame,:);
        SM.input = spatialPooler (x, false);%2048 col of only 40
active
        data.inputSDR = [data.inputSDR; SM.input];

        %%Compute anomaly score
        pi = logical(sum(SM.cellPredicted));
        predictions = [predictions; pi];
        RecognitionScores(frame) = nnz(pi & SM.input)/nnz(SM.input);
                %%% nnz : returns the number of nonzero elements in
matrix.

    %% SEQUENCE-MEMORY BEGINS HERE:
        %%
        %%to compute the active cells in SM and also the predictions
for the next time instant.
        sequenceMemory (1);%learnFlag=1

    %% TEMPORAL-POOLING BEGINS HERE:
        %%
        if (frame > 4000)
            %%perform only after some iterations -- pooling makes
sense over a period of time.
            temporalPooler (true, 1);%displayFlag=1
            TP.unionSDRhistory (mod(frame-1,
size(TP.unionSDRhistory, 1))+1, :) =  TP.unionSDR;
        end

    SM.inputPrevious = SM.input;
    SM.cellActivePrevious = SM.cellActive;
    SM.cellLearnPrevious = SM.cellLearn;
end
```

MATLAB Code IV—7 : Including HTM modules in the main script.

```
%% detect Recognition likelihood
%%


Recognition_Likelihood = Sequential_Recognition(RecognitionScores,
60);
t=1:data.N_inputs;
figure()
plot(t,RecognitionScores),title('Recognition Scores of each frame')
figure()
plot(t,Recognition_Likelihood),title('Recognition Likelihood of
sequence')
```

MATLAB Code IV—8 : Detect Recognition likelihood from the main.

```
function Recognition_Likelihood = Sequential_Recognition
(RecognitionScores, Width)

anomalyScores = 1 - RecognitionScores;
N = length(RecognitionScores);
%probabationary period:frame<trN,Ignore scores up to this point
(first loop)

trN = round(0.092*N);
%for 2 <= t <= N
width = round(Width/3);
Ut = zeros (N, 1);
U_t = zeros (N, 1);
sigma_2 = zeros(N, 1);
%---------the sample mean big window
for t = Width+1 : N
    Ut (t) = sum(anomalyScores(t-Width : t))/(Width-1);
end
%---------the sample mean small window
for t = width+1 : N
    U_t (t) = sum(anomalyScores(t-width : t))/width;
end
%---------the variance
for t = Width+1 : N
    sigma_2 (t) = sum((anomalyScores(t-Width : t)-Ut(t)).^2)/(Width-
1);
end
sigma = sqrt(sigma_2);

%% Compute tail Recognition_Likelihood

%---------z is needed inside the Q-function
z = (U_t - Ut)./(sigma);
z = (z > 0).*z; % keep only positive z
%probabationary period:frame<trN,Ignore scores upto this point
z (1:trN) = 100;% big number so the recognition is very small

%---------use the Q-function to get the recognition likelihood
Recognition_Likelihood = exp(-(z.^2)/2);

%plot(Recognition_Likelihood,'b'); title ('Recognition Likelihood');
axis([0 N 0 1]);
end
```

MATLAB Code IV—9 : Sequential Recognition function.

```
%% drawing a moving car
%%
D4 = load('DATA/combined/fGPS_rest.txt');
G4_x = D4(:,1);G4_y = D4(:,2);
s4 = 'RecognitionLikelihood in =';
L = length(D4)-30;
for i=1:L+30
    I = num2str( i );
    plot(G4_x,G4_y,'k');title(append(s4,I))
        xlim([27.497 27.505])
        ylim([152.995 153.012])
    hold on
    TEST(i) = (Recognition_Likelihood(i)*Visited(i));
    if(TEST(i)<0.975)%0.975 means 39 correctly predicted cells out of 40
        c='r';
    else
        c='g';
    end
    k4_x=G4_x(i:i+30);k4_y=G4_y(i:i+30);
    plot(k4_x,k4_y,c,'LineWidth',4);
    xlim([27.497 27.505])
    ylim([152.995 153.012])
        hold off
    pause(0.001)
end
```

MATLAB Code IV—10 : Drawing a moving car from the main.

```
%% System Evaluation
%%
Actual_True = zeros(35781,1);
TP = zeros(35781,1);
FP = zeros(35781,1);
confidences = RecognitionScores;
for i=1:35781
    if (Visited(i)==1)
        Actual_True(i)=1;
    end
    if (Recognition_Likelihood(i)>0.975)
        if (Visited(i)==1)
            TP(i)=1;
        else
            FP(i)=1;
        end
    end
end
AT = sum(Actual_True);
M = [RecognitionScores TP FP];
M = sortrows(M,[1 2],'descend');
acc_TP = zeros(35781,1);
acc_FP = zeros(35781,1);
for i=1:35781
   acc_TP(i) = sum(M(1:i,2));
   acc_FP(i) = sum(M(1:i,3));
end
Pr = acc_TP./(acc_TP+acc_FP);
Rc = acc_TP/AT;
plot(Rc,Pr),title('Precision over recall')
```

MATLAB Code IV—11 : Performance Evaluation code.

References

[1] S. Lowry *et al.*, "Visual Place Recognition: A Survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, 2016, doi: 10.1109/TRO.2015.2496823.

[2] H. Korrapati, J. Courbon, Y. Mezouar, and P. Martinet, "Image Sequence Partitioning for outdoor mapping," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1650–1655, 2012, doi: 10.1109/ICRA.2012.6224892.

[3] M. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller, "Learning Semantic Maps from Natural Language Descriptions," 2016, doi: 10.15607/rss.2013.ix.004.

[4] F. Maurelli, M. Carreras, K. Rajan, and D. Lane, "Guest editorial: special issue on long-term autonomy in marine robotics," *Auton. Robots*, vol. 40, no. 7, pp. 1111–1112, 2016, doi: 10.1007/s10514-016-9606-9.

[5] O. Vysotska and C. Stachniss, "Effective visual place recognition using multi-sequence maps," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1730–1736, 2019, doi: 10.1109/LRA.2019.2897160.

[6] L. Bampis, A. Amanatiadis, and A. Gasteratos, "Encoding the description of image sequences: A two-layered pipeline for loop closure detection," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016-Novem, pp. 4530–4536, 2016, doi: 10.1109/IROS.2016.7759667.

[7] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows," *Proc. Natl. Conf. Artif. Intell.*, vol. 4, pp. 2564–2570, 2014, doi: 10.1609/aaai.v28i1.9057.

[8] P. Hansen and B. Browning, "Visual place recognition using HMM sequence matching," *IEEE Int. Conf. Intell. Robot. Syst.*, no. Iros, pp. 4549–4555, 2014, doi: 10.1109/IROS.2014.6943207.

[9] C. Cadena, D. Gálvez-López, J. D. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 871–885, 2012, doi: 10.1109/TRO.2012.2189497.

[10] P. Yin *et al.*, "A multi-domain feature learning method for visual place recognition," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 319–324, 2019, doi: 10.1109/ICRA.2019.8793752.

[11] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "\href{https://arxiv.org/pdf/1501.04158.pdf}{On the performance of ConvNet features for place recognition}," *2015 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 4297–4304, 2015.

[12] T. Naseer, G. L. Oliveira, T. Brox, and W. Burgard, "Semantics-aware visual localization under challenging perceptual conditions," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2614–2620, 2017, doi: 10.1109/ICRA.2017.7989305.

[13] Y. Liu and H. Zhang, "Visual loop closure detection with a compact image descriptor," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1051–1056, 2012, doi: 10.1109/IROS.2012.6386145.

[14] S. Lowry and M. J. Milford, "Supervised and Unsupervised Linear Learning Techniques for Visual Place Recognition in Changing Environments," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 600–613, 2016, doi: 10.1109/TRO.2016.2545711.

[15] S. Garg, N. Suenderhauf, and M. Milford, "Don't look back: Robustifying place categorization for viewpoint- and condition-invariant place recognition," *Proc. - IEEE*

Int. Conf. Robot. Autom., pp. 3645–3652, 2018, doi: 10.1109/ICRA.2018.8461051.
[16] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, J. J. Yebes, and S. Gamez, "Bidirectional loop closure detection on panoramas for visual navigation," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 1378–1383, 2014, doi: 10.1109/IVS.2014.6856457.
[17] M. Angelina, U. Gim, and H. Lee, "PointNetVLAD : Deep Point Cloud Based Retrieval for Large-Scale Place Recognition Supplementary Material," *Cvpr*, no. c, pp. 1–2, 2021.
[18] S. Garg, N. Suenderhauf, and M. Milford, "Semantic–geometric visual place recognition: a new perspective for reconciling opposing views," *Int. J. Rob. Res.*, 2019, doi: 10.1177/0278364919839761.
[19] Y. Latif, A. D. Doan, T. J. Chin, and I. Reid, "SPRINT: Subgraph Place Recognition for INtelligent Transportation," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 5408–5414, 2020, doi: 10.1109/ICRA40945.2020.9196522.
[20] S. Garg and M. Milford, "Fast, Compact and Highly Scalable Visual Place Recognition through Sequence-based Matching of Overloaded Representations," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3341–3348, 2020, doi: 10.1109/ICRA40945.2020.9196827.
[21] T. Cieslewski and D. Scaramuzza, "Efficient decentralized visual place recognition using a distributed inverted index," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 640–647, 2017, doi: 10.1109/LRA.2017.2650153.
[22] M. Mohan, D. Galvez-Lopez, C. Monteleoni, and G. Sibley, "Environment selection and hierarchical place recognition," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2015-June, no. June, pp. 5487–5494, 2015, doi: 10.1109/ICRA.2015.7139966.
[23] L. Yu, A. Jacobson, and M. Milford, "Rhythmic representations: Learning periodic patterns for scalable place recognition at a sublinear storage cost," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 811–818, 2018, doi: 10.1109/LRA.2018.2792144.
[24] H. Badino, D. Huber, and T. Kanade, "Real-time topometric localization," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1635–1642, 2012, doi: 10.1109/ICRA.2012.6224716.
[25] S. Lynen, M. Bosse, P. Furgale, and R. Siegwart, "Placeless place-recognition," *Proc. - 2014 Int. Conf. 3D Vision, 3DV 2014*, pp. 303–310, 2015, doi: 10.1109/3DV.2014.36.
[26] M. L. Wang and H. Y. Lin, "A hull census transform for scene change detection and recognition towards topological map building," *IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc.*, pp. 548–553, 2010, doi: 10.1109/IROS.2010.5652817.
[27] J. Wang and Y. Yagi, "Robust location recognition based on efficient feature integration," *2012 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2012 - Conf. Dig.*, pp. 97–101, 2012, doi: 10.1109/ROBIO.2012.6490950.
[28] N. Sünderhauf *et al.*, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," *Robot. Sci. Syst.*, vol. 11, 2015, doi: 10.15607/RSS.2015.XI.022.
[29] N. Sünderhauf *et al.*, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," *Robot. Sci. Syst.*, vol. 11, 2015, doi: 10.15607/RSS.2015.XI.022.
[30] F. Dayoub, T. Morris, B. Upcroft, and P. Corke, "Vision-only autonomous navigation using topometric maps," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1923–1929, 2013, doi: 10.1109/IROS.2013.6696611.
[31] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1352–1359, 2013, doi: 10.1109/CVPR.2013.178.
[32] S. Garg *et al.*, "Look no deeper: Recognizing places from opposing viewpoints under varying scene appearance using single-view depth estimation," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 4916–4923, 2019, doi: 10.1109/ICRA.2019.8794178.

[33] Z. Hong, Y. Petillot, D. Lane, Y. Miao, and S. Wang, "TextPlace: Visual place recognition and topological localization through reading scene texts," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 2861–2870, 2019, doi: 10.1109/ICCV.2019.00295.
[34] T. Morris, F. Dayoub, P. Corke, G. Wyeth, and B. Upcroft, "Multiple map hypotheses for planning and navigating in non-stationary environments," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2765–2770, 2014, doi: 10.1109/ICRA.2014.6907255.
[35] W. Maddern, M. Milford, and G. Wyeth, "Capping computation time and storage requirements for appearance-based localization with CAT-SLAM," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 822–827, 2012, doi: 10.1109/ICRA.2012.6224622.
[36] Y. Latif, G. Huang, J. Leonard, and J. Neira, "An Online Sparsity-Cognizant Loop-Closure Algorithm for Visual Navigation," *Robot. Sci. Syst.*, 2014, doi: 10.15607/RSS.2014.X.036.
[37] M. Milford, "Vision-based place recognition: How low can you go?," *Int. J. Rob. Res.*, vol. 32, no. 7, pp. 766–789, 2013, doi: 10.1177/0278364913490323.
[38] Z. Chen, F. Maffra, I. Sa, and M. Chli, "Only look once, mining distinctive landmarks from ConvNet for visual place recognition," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017-Septe, pp. 9–16, 2017, doi: 10.1109/IROS.2017.8202131.
[39] M. Gadd, D. De Martini, and P. Newman, "Look Around You: Sequence-based Radar Place Recognition with Learned Rotational Invariance," *2020 IEEE/ION Position, Locat. Navig. Symp. PLANS 2020*, pp. 270–276, 2020, doi: 10.1109/PLANS46316.2020.9109951.
[40] Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020, doi: 10.1109/TPAMI.2018.2889473.
[41] P. Gao and H. Zhang, "Long-term Place Recognition through Worst-case Graph Matching to Integrate Landmark Appearances and Spatial Relationships," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1070–1076, 2020, doi: 10.1109/ICRA40945.2020.9196906.
[42] S. Garg, N. Suenderhauf, and M. Milford, "LoST? Appearance-Invariant Place Recognition for Opposite Viewpoints using Visual Semantics," *Robot. Sci. Syst.*, 2018, doi: 10.15607/RSS.2018.XIV.022.
[43] Z. Liu *et al.*, "LPD-Net: 3D point cloud learning for large-scale place recognition and environment analysis," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, no. Iccv, pp. 2831–2840, 2019, doi: 10.1109/ICCV.2019.00292.
[44] L. Wu and Y. Wu, "Deep Supervised Hashing with Similar Hierarchy for Place Recognition," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 3781–3786, 2019, doi: 10.1109/IROS40897.2019.8968599.
[45] S. M. Siam and H. Zhang, "Fast-SeqSLAM: A fast appearance based place recognition algorithm," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 5702–5708, 2017, doi: 10.1109/ICRA.2017.7989671.
[46] Z. Liu *et al.*, "SeqLPD: Sequence Matching Enhanced Loop-Closure Detection Based on Large-Scale Point Cloud Description for Self-Driving Vehicles," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1218–1223, 2019, doi: 10.1109/IROS40897.2019.8967875.
[47] J. Hawkins and R. Dawkins, *A thousand brains : a new theory of intelligence*.
[48] J. Hawkins, S. Ahmad, and D. Dubinsky, "HTM Cortical Learning Algorithms," pp. 1–68, 2011.
[49] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, 2016, doi: 10.1162/NECO_a_00893.

[50] J. Hawkins, M. Lewis, M. Klukas, S. Purdy, and S. Ahmad, "A framework for intelligence and cortical function based on grid cells in the neocortex," *Front. Neural Circuits*, vol. 12, no. January, 2019, doi: 10.3389/fncir.2018.00121.
[51] A. M. Zyarah, "Design and Analysis of a Reconfigurable Hierarchical Temporal Memory Architecture Design and Analysis of a Reconfigurable Hierarchical Temporal Memory Architecture," no. June 2015, 2015.
[52] M. Chalk, O. Marre, and G. Tkacik, "Towards a unified theory of efficient, predictive and sparse coding: Supplementary information," pp. 1–17.
[53] Y. Liu, G. Zhai, K. Gu, X. Liu, D. Zhao, and W. Gao, "Reduced-reference image quality assessment in free-energy principle and sparse representation," *IEEE Trans. Multimed.*, vol. 20, no. 2, pp. 379–391, 2018, doi: 10.1109/TMM.2017.2729020.
[54] S. Ahmad and J. Hawkins, "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory," pp. 1–18, 2015, [Online]. Available: http://arxiv.org/abs/1503.07469
[55] S. Ahmad and J. Hawkins, "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory," pp. 1–18, 2015, [Online]. Available: http://arxiv.org/abs/1503.07469
[56] S. Purdy, "Encoding Data for HTM Systems," 2016, [Online]. Available: http://arxiv.org/abs/1602.05925
[57] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008, doi: 10.1016/j.cviu.2007.09.014.
[58] J. Heyder, "Hierarchical Temporal Memory Software Agent: In the light of general artificial intelligence criteria," 2018.
[59] S. Ahmad, "Temporal Memory Algorithm," 2021.
[60] J. Hawkins and S. Ahmad, "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex," *Front. Neural Circuits*, vol. 10, no. March, pp. 1–13, 2016, doi: 10.3389/fncir.2016.00023.
[61] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017, doi: 10.1016/j.neucom.2017.04.070.
[62] I. Bautista, S. Sarkar, and S. Bhanja, "MatlabHTM: A sequence memory model of neocortical layers for anomaly detection," *SoftwareX*, vol. 11, p. 100491, 2020, doi: 10.1016/j.softx.2020.100491.
[63] "numenta/nupic: Numenta Platform for Intelligent Computing is an implementation of Hierarchical Temporal Memory (HTM), a theory of intelligence based strictly on the neuroscience of the neocortex." https://github.com/numenta/nupic (accessed Sep. 07, 2022).
[64] "numenta/nupic.core: Implementation of core NuPIC algorithms in C++ (under construction)." https://github.com/numenta/nupic.core (accessed Sep. 07, 2022).
[65] "numenta/htm.java: Hierarchical Temporal Memory implementation in Java - an official Community-Driven Java port of the Numenta Platform for Intelligent Computing (NuPIC)." https://github.com/numenta/htm.java (accessed Sep. 07, 2022).
[66] "Biological and Machine Intelligence (BAMI)." https://numenta.com/resources/biological-and-machine-intelligence/ (accessed Sep. 07, 2022).
[67] "Overview of the Temporal Pooler · numenta/htmresearch Wiki." https://github.com/numenta/htmresearch/wiki/Overview-of-the-Temporal-Pooler (accessed Sep. 07, 2022).

The summary

Abstract

Visual place recognition is an essential capability for autonomous mobile robots which use cameras as their primary sensors. Following advances in neuroscience, new biologically inspired models have been developed. Inspired by the human neocortex named hierarchical temporal memory. This dissertation proposes a system that recognize visual place from the past by using the hierarchical temporal memory model.

Keywords: Visual place recognition, autonomous mobile robots, hierarchical temporal memory.

Résumé

La reconnaissance visuelle des lieux est une capacité essentielle pour les robots mobiles autonomes qui utilisent des caméras comme capteurs primaires. Suite aux avancées en neurosciences, de nouveaux modèles d'inspiration biologique ont été développés. Inspiré par le néocortex humain nommé mémoire temporelle hiérarchique. Ce mémoire propose un système qui reconnaît la place visuelle du passé en utilisant le modèle de mémoire temporelle hiérarchique.

Mots-clés: Reconnaissance visuelle des lieux, robots mobiles autonomes, mémoire temporelle hiérarchique.

ملخـــص

التعرف على المكان المرئي هو قدرة أساسية للروبوتات المتنقلة المستقلة التي تستخدم الكاميرات كأجهزة استشعار أساسية. بعد التقدم في علم الأعصاب، تم تطوير نماذج جديدة مستوحاة بيولوجيًا. مستوحاة من القشرة الدماغية الجديدة البشرية المسماة الذاكرة الزمنية الهرمية. تقترح هذه الأطروحة نظامًا يتعرف على المكان المرئي من الماضي باستخدام نموذج الذاكرة الزمنية الهرمي.

كلمات مفتاحية: التعرف على المكان المرئي، للروبوتات المتنقلة المستقلة، الذاكرة الزمنية الهرمية.