Avelino Forechi Silva

Appearance-based Global Localization with a Hybrid Weightless-Weighted Neural Network Approach

Vitória, ES

February, 2018

Avelino Forechi Silva

Appearance-based Global Localization with a Hybrid Weightless-Weighted Neural Network Approach

Doctoral thesis submitted to the Postgraduate Program in Informatics of the Federal University of Espírito Santo as partial requirement for obtaining the degree of Doctor of Computer Science, according to the Internal Rules of the aforementioned postgraduate program.

Federal University of Espírito Santo

Technological Centre

Postgraduate Program in Informatics

Supervisor: Ph.D. Alberto Ferreira De Souza

Co-supervisor: Ph.D. Thiago Oliveira dos Santos

Vitória, ES

February, 2018

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)
Sandra Mara Borges Campos – CRB-6 ES-000593/O

Silva, Avelino Forechi, 1978-
S586a Appearance-based global localization with a hybrid weightless-weighted neural network approach / Avelino Forechi Silva. – 2018.
104 f. : il.

Orientador: Alberto Ferreira De Souza.
Coorientador: Thiago Oliveira dos Santos.
Tese (Doutorado em Ciência da Computação) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Redes neurais (Computação). 2. Aprendizado de máquinas. 3. Veículos autônomos. 4. Robótica. I. Souza, Alberto Ferreira De. II. Santos, Thiago Oliveira dos. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Appearance-based Global Localization with a Hybrid Weightless-Weighted Neural Network Approach

Avelino Forechi Silva

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 02 de fevereiro de 2018 por:

Prof. Dr. Alberto Ferreira De Souza (Orientador)
UFES/ES

Prof. Dr. Thiago Oliveira dos Santos (Coorientador)
UFES/ES

Profª. Drª. Claudine Santos Badue Gonçalves (Examinador Interno)
UFES/ES

Prof. Dr. Elias de Oliveira (Examinador Interno)
UFES/ES

Prof. Dr. Edilson de Aguiar (Examinador Externo)
MPI Informatik/Alemanha

Pro. Dr. Patrick Marques Ciarelli (Examinador Externo)
UFES/ES

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória-ES, 02 de fevereiro de 2018.

To my lovely wife Karolini and my beloved son Henrique.

Acknowledgements

Firstly, I must thank God for keeping me faithful. Next, I thank all the support, love and patience dedicated by my wife Karolini Herzog, especially in the last five years. We have passed similar situations during my master and undergraduate and you have always been by my side, thank you. I also cannot forget to thank my parents for understanding my absence and for supporting me in my quest for knowledge. I thank my in-laws who have always been around to help in times of need and also share happy moments. I have much to thank my supervisor, Professor Dr. Alberto Ferreira De Souza, for the opportunity to learn and collaborate in such relevant research projects for the world. I admire your patience to teach and seek the best for all. Your insightful ideas and guidance help me out through this journey, thank you. To my co-supervisor, Professor Dr. Thiago Oliveira dos Santos, I am very grateful for his willingness to discuss in detail solutions to many ideas proposed. I should also thank all of my lab mates, who, after all these years together, are like a family to me, the LCAD family. In particular, I would like to thank Lucas Veronese, Filipe Mutz, Vinicius Cardoso, Thómas Teixeira, Josias, Oliveira, Raphael Carneiro, Helio Perroni, Rodrigo Berriel, Matheus Nogueira and Leonardo Muniz. All contributed positively for pleasant moments together and healthy discussions. I also have to thank FAPES, FINEP and CAPES for the financial support during my doctorate.

"We move in order to see and we see in order to move."
(J.J. Gibson)

Abstract

Currently, self-driving cars rely greatly on the Global Positioning System (GPS) infrastructure, albeit there is an increasing demand for global localization alternative methods in GPS-denied environments. One of them is known as appearance-based global localization, which associates images of places with their corresponding position. This is very appealing regarding the great number of geotagged photos publicly available and the ubiquitous devices fitted with ultra-high-resolution cameras, motion sensors and multicore processors nowadays. The appearance-based global localization can be devised in topological or metric solution regarding whether it is modelled as a classification or regression problem, respectively. The topological common approaches to solve the global localization problem often involve solutions in the spatial dimension and less frequent in the temporal dimension, but not both simultaneously.

It was proposed an integrated spatio-temporal solution based on an ensemble of kNN classifiers, where each classifier uses the Dynamic Time Warping (DTW) and the Hamming distance to compare binary features extracted from sequences of images. Each base learner is fed with its own binary set of features extracted from images. The solution was designed to solve the global localization problem in two phases: mapping and localization. During mapping, it is trained with a sequence of images and associated locations that represents episodes experienced by a robot. During localization, it receives subsequences of images of the same environment and compares them to its previous experienced episodes, trying to recollect the most similar "experience" in time and space at once. Then, the system outputs the positions where it "believes" these images were captured.

Although the method is fast to train, it scales linearly with the number of training samples in order to compute the Hamming distance and compare it against the test samples. Often, while building a map, one collects high correlated and redundant data around the environment of interest. Some reasons are due to the use of high frequency sensors or to the case of repeating trajectories. This extra data would carry an undesired burden on memory and runtime performance during test if not treated appropriately during the mapping phase. To tackle this problem, it is employed a clustering algorithm to compress the network's memory after mapping. For large scale environments, it is combined the clustering algorithms with a multi hashing data structure seeking the best compromise between classification accuracy, runtime performance and memory usage.

So far, this encompasses solely the topological solution part for the global localization problem, which is not precise enough for autonomous cars operation. Instead of just recognizing places and outputting an associated pose, it is desired that a global localization

system regresses a pose given a current image of a place. But, inferring poses for city-scale scenes is unfeasible at least for decimetric precision. The proposed approach to tackle this problem is as follows: first take a live image from the camera and use the localization system aforementioned to return the image-pose pair most similar to a topological database built as before in the mapping phase. And then, given the live and mapped images, a visual localization system outputs the relative pose between those images. To solve the relative camera pose estimation problem, it is trained a Convolutional Neural Network (CNN) to take as input two separated images in time and space in order to output a 6 Degree of Freedom (DoF) pose vector, representing the relative position and orientation between the input images. In conjunction, both systems solve the global localization problem using topological and metric information to approximate the actual robot pose.

The proposed hybrid weightless-weighted neural network approach is naturally combined in a way that the output of one system is the input to the other producing competitive results for the Global Localization task. The full approach is compared against a Real Time Kinematic GPS system and a Visual Simultaneous Localization and Mapping (SLAM) system. Experimental results show that the proposed combined approach is able to correctly global localize an autonomous vehicle 90% of the time with a mean error of 1.20m compared to 1.12m of the Visual SLAM system and 0.37m of the GPS, 89% of the time.

Keywords: deep learning. convolutional neural networks. weightless neural networks. autonomous vehicle navigation.

Resumo

Atualmente, veículos autônomos dependem muito da infra-estrutura do Sistema de Posicionamento Global (GPS, da sigla em inglês), embora haja uma demanda crescente de métodos alternativos de localização global em ambientes com ausência de sinal de GPS. Um deles é conhecido como localização global baseada em aparência, que associa imagens de lugares com sua posição correspondente. Isso é muito atraente com relação à grande quantidade de fotos disponíveis publicamente com metadados geográficos e também se considerados os dispositivos móveis equipados com câmeras de altíssima resolução, sensores de movimento e processadores multi-núcleos disponíveis atualmente. A localização global baseada em aparência pode ser concebida como sendo uma solução topológica ou métrica quanto ao fato de ser modelada como um problema de classificação ou regressão, respectivamente. As abordagens topológicas comumente utilizadas para resolver o problema de localização global envolvem soluções na dimensão espacial e menos freqüentemente na dimensão temporal, mas não simultaneamente.

Foi proposta uma solução espaço-temporal integrada baseada em um conjunto de classificadores kNN, onde cada classificador usa Dynamic Time Warping (DTW) e a distância de Hamming para comparar *Features* binárias extraídas de seqüências de imagens. Cada classificador é treinado com seu próprio conjunto binário de *Features* extraídas das imagens. A solução foi projetada para resolver o problema de localização global em duas fases: mapeamento e localização. Durante o mapeamento, o classificador é treinado com uma seqüência de imagens e locais associados que representam episódios experimentados por um robô. Durante a localização, ele recebe subseqüências de imagens do mesmo ambiente e as compara com os episódios experimentados anteriormente, tentando relembrar qual foi a "experiência" mais semelhante considerando tempo e espaço simultaneamente. Então, o sistema exibe as posições onde "acredita" que essas imagens foram capturadas.

Embora o método seja rápido para treinar, ele escala linearmente com o número de amostras de treinamento, ao calcular a distância de Hamming e compará-la com as amostras de teste. Muitas vezes, ao construir um mapa, ocorre dos dados coletados serem altamente correlacionados e redundantes em torno do ambiente de interesse. Algumas razões se devem ao uso de sensores com alta freqüência de amostragem ou o caso de trajetórias repetidas. Esses dados extras podem ocasionar uma sobrecarga indesejada sobre a memória e o desempenho em tempo de execução durante o teste, se não for tratado adequadamente durante a fase de mapeamento. Para enfrentar este problema, foi empregado um algoritmo de agrupamento (*Clustering*) para comprimir a memória da rede após o mapeamento. Para ambientes de maior escala, combinamos os algoritmos de agrupamento com uma estrutura de dados com múltiplas tabelas de espalhamento (*Hash Tables*) buscando o

melhor equilíbrio entre a precisão da classificação, o desempenho em tempo de execução e o uso de memória.

Até aqui, o que foi discutido abrange apenas a parte de solução topológica para o problema de localização global, que não é suficientemente precisa para a operação de carros autônomos. Em vez de apenas reconhecer locais e produzir uma pose associada, é desejado que um sistema de localização global calcule uma pose dada uma imagem atual de um lugar. Mas inferir poses para cenas numa escala de cidade é uma tarefa muitas vezes inviável, pelo menos, para precisão decimétrica. A abordagem proposta para tentar resolver este problema é a seguinte: primeiro capture uma imagem ao vivo da câmera e use o sistema de localização acima mencionado para retornar o par de pose e imagem mais semelhante a um banco de dados topológico construído como antes na fase de mapeamento. E então, dadas as imagens ao vivo e mapeadas, um sistema de localização visual calcula a pose relativa entre essas imagens. Para resolver o problema de estimativa de pose relativa entre câmeras, é treinada uma Rede Neural Convolucional (CNN, da sigla em inglês) seguindo o projeto de uma arquitetura Siamesa para tomar como entrada duas imagens separadas no tempo e espaço e então produzir um vetor de pose com 6 graus de liberdade (6-DoF, da sigla em inglês), representando a posição relativa e orientação entre as imagens de entrada. Em conjunto, ambos os sistemas solucionam o problema de localização global usando informações topológicas e métricas para aproximar a pose real do robô.

A abordagem proposta de se combinar rede neurais híbridas, com e sem peso, é uma forma natural de unificar as duas abordagens. De forma que a saída de um sistema seja a entrada para o outro e produza resultados competitivos para a tarefa de localização global. A abordagem completa é comparada então com um GPS cinemático de tempo real (RTK, da sigla em inglês) e um sistema visual de localização e mapeamento simultânemos (SLAM, da sigla em inglês). Os resultados experimentais mostram que a abordagem proposta completa é capaz de localizar globalmente um veículo autônomo em 90% do tempo com um erro médio de 1,20m em comparação com 1,12m alcançado pelo sistema de SLAM visual e 0,37m do GPS-RTK em 89% do tempo.

Palavras-chaves: deep learning, redes neurais convolucionais, redes neurais sem peso, carros autônomos.

List of Figures

List of Tables

List of abbreviations and acronyms

ADAS	Advanced Driver Assistance Systems
BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Network
DARPA	Defense Advanced Research Projects Agency
DLT	Direct Linear Transformation
DNN	Deep Neural Networks
DTW	Dynamic Time Warping
FAST	Features from Accelerated Segment Test
FCN	Fully Convolutional Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
IARA	Intelligent and Autonomous Robotic Automobile
kNN	k Nearest Neighbor
LCAD	Laboratório de Computação de Alto Desempenho
LGN	Lateral Geniculate Nucleus
MAE	Maximum Allowed Error
OGM-MCL	Occupancy Grid Mapping Monte Carlo Localization
ORB	Oriented FAST and Rotated BRIEF
RANSAC	RANdom SAmple Consensus
RTK	Real Time Kinematic
SAD	Sum of Absolute Differences
SIFT	Scale-Invariant Feature Transform
SFA	Slow Feature Analysis

SLAM	Simultaneous Localization And Mapping
SPS	Slanted Plane Smoothing
SURF	Speeded Up Robust Features
UFES	Universidade Federal do Espírito Santo
VGG	Visual Geometry Group
VG-RAM	Virtual Generalizing Random Access Memory
VLAD	Vector of Localized Descriptors in Aggregates
WiSARD	Wilkes, Stonham and Aleksander Recognition Device
WNN	Weightless Neural Network

Contents

1 Introduction

This chapter presents an overview of the thesis, as well as defines the basis for the following chapters. Therefore, this chapter comprises the context in which the thesis is embedded, the motivation for conducting this thesis, the hypotheses, the research objectives, and the methodological aspects used for guiding this research. Finally, the structure of this document is presented.

1.1 Context

From the beginnings of remote-controlled cars showcased in 1939 New York World's Fair (Futurama) to sensor-fitted self-driving cars showcased in DARPA Challenges (BUEHLER; IAGNEMMA; SINGH, 2007; BUEHLER; IAGNEMMA; SINGH, 2009) was a big step forward promoted mainly by software (e.g. probabilistic models) moving gradually to the adoption of Advanced Driver Assistance Systems (ADAS) for the public in general. In 2005, Stanford's robot Stanley won the DARPA Grand Challenge. Two years later, Carnegie Mellon University's robot Boss won the DARPA Urban Challenge. This challenge required driverless car to traverse 60 miles of urban streets under controlled conditions. Since then, there has been great advances both in hardware and software. Though software is still the key to autonomous driving and its main functions: perception, planning and control (THRUN, 2010). For the competitions none of following tasks were required: pedestrian and cyclist detection, traffic signs and traffic lights detection and identification. These are major perception tasks, in which Deep Learning techniques have been achieving superhuman performance. Though, Simultaneous Localization And Mapping systems are still dominated by probabilistic models.

This thesis dissertation, therefore, is inserted in this context, which is characterized by: (i) the growing adoption of machine learning in robotics; (ii) high performance computing devices; (iii) an increasing data volume and data-driven algorithms.

1.2 Motivation

Cognition can be defined as our ability to understand the world and ideas through our senses and our memory of past experiences. We can roughly categorize our various cognitive abilities according to our senses, leading to categories such as visual cognition, auditory cognition or tactile cognition. We can also categorize our cognitive skills according

to our ability to understand - via our senses and long/short-term memory systems - abstract concepts of space (spatial cognition) and movement of our body (motor cognition). The human long-term memory systems, particularly the Episodic Memory, are essential for answering daily questions which involves "what", "when" and "where" (TULVING, 2002). The Episodic Memory system (named accordingly to the Tulving's taxonomy (TULVING, 1972)) stores sequences of events (episodes encoded in time and in space) experienced or imagined by an individual, allowing them to access their episodes in whole or in part later. Such ability of our Episodic Memory system is important, among other functions, for our localization in space throughout time (BURGESS; MAGUIRE; O'KEEFE, 2002).

The discoveries about spatial cognition on rats (WIDLOSKI; FIETE, 2014) suggest that a special configuration of neuronal cells in the *Hippocampus* and in the *Enthorinal Cortex* is able to encode internal representations of the environment and the rat's position in relation to it (allocentric). These cells include the so called *Place Cell*, which spikes only in certain places at the environment (O'KEEFE; DOSTROVSKY, 1971). They would provide to the animal a continuously updated allocentric representation of the space and the animal's position in this space (O'KEEFE; NADEL, 1978). Further studies in adjacent regions of the *Hippocampus*, which have direct interconnection with it, led to the discovery of another type of position cell in the *Entorhinal Cortex* of rats called *Grid Cell* (HAFTING et al., 2005). *Grid Cells* generate action potentials (spikes) when the animal passes by specific and regularly spaced locations in the environment, suggesting path integration. *Grid Cells* interact with other types of specialized cells such as *Head Direction Cells* and *Border Cells* (MOSER; KROPFF; MOSER, 2008). *Head Direction Cells* are active when the animal's head points to a specific direction of the environment, while *Border Cells* are active when the animal is close to an edge of a specific part of the environment. These cells of the *Entorhinal Cortex*, along with the *Place Cells* of the *Hippocampus*, establish coherent *maps* of the local space that are kept in the animal's memory for each environment, regardless of the speed and the direction of motion, and even regardless of the identity of specific landmarks found in the environment known by the animal (i.e. small changes in the environment do not significantly affect the pattern of activation of the cells) (LEUTGEB et al., 2005). The reactivation of *Place Cells* and other cells whose activation represents prior experiences can lead to recollections of environments (maps) previously visited by means of the *Episodic Memory* (BURGESS, 2008).

Analogously, autonomous robots should possess cognitive capabilities like humans in order to understand the space around them and to interact with it. Ideally, it would be necessary for robots to possess at least visual, auditory, tactile, motor and spatial cognition to be fully autonomous. Despite being far from that, the state of the art in robotics has made great strides in that direction, mainly due to probabilistic approaches employed to solve, for example, the Simultaneous Localization And Mapping (SLAM) (THRUN; BURGARD; FOX, 2005) and the Navigation (BUEHLER; IAGNEMMA; SINGH, 2009) problems.

SLAM is perhaps the most fundamental problem in Robotics because autonomous mobile robots need to know where they are (i.e. localize themselves) in the environment, while simultaneously building a map of the environment, in order to navigate through it and perform activities of interest. On the other hand, when a map is available, the robot just needs to localize itself by examining the map and sensing the environment. That is the Localization problem and it can be further divided in the Global Localization and the Position Tracking (THRUN; BURGARD; FOX, 2005) problems. The latter assumes that the initial robot position is known whilst the former does not. In addition, the latter is bounded to the local space around the true position of the robot whereas the former is not. In fact, the Global Localization problem encompasses the Position Tracking problem and it is in general more difficult to solve.

Nonetheless, SLAM and Localization algorithms rely on a good initial position that is usually obtained with Global Position System (GPS) sensors. However, due to signal coverage restrictions and other related problems, this type of sensor may have an increased localization error that compromises its use in urban streets. An alternative approach to solve the Global Localization problem is by modeling it as a Place Recognition problem, in which the question to answer is whether or not a similar image of a place has already been seen. The problem is commonly approached in three broader steps: (i) extract features from images captured in places visited by the robot, (ii) build and update an internal representation of these places, and (iii) predict their locations based on upcoming images and, optionally, on temporal information (LOWRY et al., 2015).

Places might change appearance, either because they represent different locations or different perspectives of the same place. However, their relative locations remain unchanged within the same time span when performing a similar route. We explore this spatio-temporal relation by modeling the Place Recognition problem as sequences of ordered image-position pairs and finding the most similar subsequence according to the k-Nearest Neighbor (kNN) classification algorithm and the Dynamic Time Warping (DTW) dissimilarity measure.

The Place Recognition problem modeled as a classification problem solves just the first part of the Global Localization problem, given that it learns past place images and returns the most similar place location where the robot was and not its current position. An complementary approach to solve the Global Localization problem would be to compute a transformation from the past place to the current given their corresponding images. Which involves estimating a relative camera position given the past and current place images. This problem differs from Visual Odometry (VO) (NISTER; NARODITSKY; BERGEN, 2004) and Structure from Motion (SfM) (HUANG; NETRAVALI, 1994) problems. Although all of them can be characterized as Visual Localization methods, estimating a relative camera pose is more general than the other two. Visual Odometry, for instance, computes motion from subsequent image frames while the relative camera position method computes

motion from non contiguous image frames. Structure from Motion computes motion and structure of rigid objects based on feature matching at different times or from different viewpoints or cameras, while estimating the relative camera pose along the road does not benefit from structure of rigid objects, most of the time, given that the camera motion is roughly orthogonal to the image plane. The Relative Camera Pose Estimation problem is addressed here based on Convolutional Neural Networks (CNN) in order to regress the relative pose between current and past place views given by the images taken from cameras in the past and current places, respectively.

The proposed hybrid weightless-weighted neural network approach is naturally combined in a way that the output of one system is the input to the other producing competitive results for the Global Localization task. The full approach is compared against a Real-Time Kinematic GPS (RTK-GPS) and a Visual Simultaneous Localization and Mapping (V-SLAM) system (JÚNIOR et al., 2015). Experimental results show that the proposed combined approach is able to correctly global localize an autonomous vehicle 90% of the time with a mean error of 1.2m compared to 1.12m of the V-SLAM system and 0.37m of the RTK-GPS (89% of the time).

1.3 Research Hypothesis

Inspired by the brain Episodic Memory notion, its Memory Consolidation process and its Depth Perception mechanism it is possible to build Ensemble and Deep Learning models for image-based global localization problem in order to improve the localization accuracy. Whether considering the spatiotemporal dimension as episodes, or clustering in the feature space places revisited over time. For latter recollecting the past viewpoint of a place and comparing it to the current view in order to find one self's position.

In this thesis, a hybrid Weightless-Weighted Neural Network approach is proposed to address the Global Localization problem, subdivided in two parts: one based on Weightless Neural Networks (WNN) for solving the Place Recognition Problem and its counterpart based on Convolutional Neural Networks (CNN) for solving the Visual Localization Problem. The first part of the approach takes inspiration from the human long-term Episodic Memory system and is based on an ensemble of kNN classifiers. Each classifier uses a DTW-Hamming distance metric to compare binary features extracted from sequences of images. The first approach is designed to solve the Place Recognition problem in two phases: mapping and localization. During mapping, it is trained with a sequence of images and associated locations that, here, represents episodes experienced by an autonomous robot. During localization, it receives subsequences of images of the environment and compares them to its previous experienced episodes, trying to recollect the most similar "experience" in time and space at once. Then, the system outputs the

positions where it "believes" these images were captured. The second part of the approach is based on CNN's, which are directly inspired by the models of simple and complex cells and the LGN-V1-V2-V4-IT hierarchy in the visual cortex ventral pathway (LECUN; BENGIO; HINTON, 2015). And more related to depth perception is the middle temporal region, which is part of visual cortex pathway LGN-V1-V2-V3-V4-MT. The MT region, specifically, responds for stereoscopic depth perception around the fixation point defined by the eyes' vergence movement (KOMATI; DE SOUZA, 2003).

1.4 The Research Track

Some results of the following publications are also part of this thesis:

- **FORECHI, Avelino** ; OLIVEIRA-SANTOS, Thiago ; BADUE, Claudine ; DE SOUZA, Alberto F. . Sequential appearance-based Global Localization using an ensemble of kNN-DTW classifiers. In: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, Rio de Janeiro. (In Press)
- **FORECHI, Avelino** ; DE SOUZA, Alberto F. ; BADUE, Claudine ; OLIVEIRA-SANTOS, Thiago . Sequential appearance-based Global Localization using an ensemble of kNN-DTW classifiers. In: 2016 International Joint Conference on Neural Networks (IJCNN), 2016, Vancouver.
- **FORECHI, Avelino** ; DE SOUZA, Alberto F. ; NETO, Jorcy O. ; DE AGUIAR, Edilson ; BADUE, Claudine ; GARCEZ, Artur d'Avilla ; OLIVEIRA-SANTOS, Thiago . Fat-Fast VG-RAM WNN: A High Performance Approach. Neurocomputing (Amsterdam), 2015.
- DE AGUIAR, Edilson ; **FORECHI, Avelino**; VERONESE, Lucas ; BERGER, Mariella ; DE SOUZA, Alberto F. ; BADUE, Claudine ; OLIVEIRA-SANTOS, Thiago . Compressing VG-RAM WNN memory for lightweight applications. In: 2014 International Joint Conference on Neural Networks (IJCNN), 2014, Beijing.

1.5 Outline of this Thesis

This chapter presented the first part of this thesis (Chapter 1), in which the general aspects are described, namely: the research context, the motivation for performing this thesis, the research hypothesis, the research objectives, and the methodological aspects. The remaining content of this thesis is organized as follows:

- **Chapter 2**: This chapter describes the state of the art of Place Recognition and Visual Localization both categorized as feature-based or learning-base approaches.

- **Chapter 3**: This chapter describes the state of the art regarding Machine Learning techniques necessary for grounding this work, which includes, in general: weightless and weighted neural networks.

- **Chapter 4**: This chapter presents the proposed approach to solve the global localization problem in the space of appearance in contrast to traditional Global Positioning System (GPS) sensors. The proposed approach is twofold: a Weightless Neural Network (WNN) to solve place recognition as a classification problem, and a Convolutional Neural Network (CNN) to solve visual localization as a metric regression problem.

- **Chapter 5**: This chapter presents the experimental setup used to evaluate the proposed system and also shows and discusses the outcomes of the experiments. It starts describing the autonomous vehicle platform used to acquire the datasets, follows presenting the datasets themselves, some parameter tuning, describes the methodology used in the experiments, and finally the results.

- **Chapter 6**: This chapter summarizes the general ideas discussed in this work and the overall results achieved.

2 Related Works

This chapter describes the state-of-the-art methods for solving Place Recognition and Visual Localization problems, in which those methods can be further categorized in feature-based or learning-based models. The first is a more traditional approach and employs well established handcrafted feature detectors algorithms to directly represent images in feature space. The latter, employs *Deep Learning* techniques to train Deep Neural Networks (DNN) to find more abstract representations for input images as deeper the network layer is.

2.1 Place Recognition

Place recognition systems serve a great variety of applications such as robot localization (ENGELSON, 1994; MUR-ARTAL; MONTIEL; TARDOS, 2015; JÚNIOR et al., 2015), navigation (MATSUMOTO; INABA; INOUE, 1996; CUMMINS; NEWMAN, 2007; MILFORD; WYETH, 2009), loop closure detection (LYNEN et al., 2015; LABBE; MICHAUD, 2013; GALVEZ-LOPEZ; TARDOS, 2012; HOU; ZHANG; ZHOU, 2015), through geo-tagged services (WEYAND; KOSTRIKOV; PHILBIN, 2016; LIN et al., 2015; HAYS; EFROS, 2008). They also come in a great variety of models, algorithms and innovative approaches, going from models based on handcrafted features such as SURF (BAY et al., 2008), SIFT (LOWE, 2004), ORB (RUBLEE et al., 2011), trough data-driven models to learn their own CNN features (LECUN et al., 1998).

Since the first forms of image representation (ARGAMON-ENGELSON, 1998) have been applied to place recognition problem, a myriad of approaches have emerged, largely thanks to recent advances in *Deep Learning*, attempting to solve this still open problem. The Argamon-Engelson's method for localizing a mobile robot indoors is based on matching image signatures, which represents an image as a set of arrays of coarse-scale measurements over subimages. Their results show that the method enables accurate place recognition, comparable to the results for localization using sonar-based occupancy grid maps (ELFES, 1989). Being the first topological while the latter characterized as metric localization methods in the field of Robotics.

Following the topological approach, the authors describe in (ULRICH; NOURBAKHSH, 2000) an appearance-based location recognition system that uses a panoramic vision system to sense the environment. Color images are classified in real time based on the nearest neighbor algorithm, image histogram matching and a simple voting scheme.

The system was evaluated with eight cross-sequence tests in four distinct environments, three indoors and one outdoors. In all cases, the system successfully tracked the robot position, i.e., the system correctly classified between 87% and 98% of the input images.

Wolf, Burgard and Burkhardt, presented in (WOLF; BURGARD; BURKHARDT, 2002) an image-based approach to mobile robot localization, that integrates an image retrieval system with Monte-Carlo localization. The image retrieval process is based on features that are invariant to image transformations such as translations, rotations, and limited scale. The sample-based Monte-Carlo localization technique allows their robot to efficiently integrate multiple measurements over time. Both techniques are combined by extracting a set of possible view-points for each image using a two-dimensional map of the environment.

Cummins and Newman, in their seminal paper (CUMMINS; NEWMAN, 2008), employed a probabilistic approach to the topological SLAM problem in the space of appearance, named FAB-MAP. Besides localization, FAB-MAP can be applied to the loop closure detection problem, given that the algorithm complexity is linear in the number of places in the map (CUMMINS; NEWMAN, 2007). Nonetheless, it requires an offline step to build a visual vocabulary based on bag-of-words (SIVIC; ZISSERMAN, 2003) and SURF features (BAY et al., 2008). FAB-MAP 2.0 (CUMMINS; NEWMAN, 2009) extended the previous version to support large scale datasets up to 1000km (CUMMINS; NEWMAN, 2010).

FAB-MAP was tested in conjunction with RatSLAM (GLOVER et al., 2010), which is another appearance-based approach to the SLAM problem. RatSLAM (MILFORD; WYETH; PRASSER, 2004) is a biologically inspired SLAM approach that uses a simplified visual odometry in addition to appearance-based image template-matching for building topological maps consisting of simulated Place Cells (O'KEEFE; DOSTROVSKY, 1971) activations by a Continuous Attractor Network (CAN) (TRAPPENBERG, 2003). The joint test of FAB-MAP and RatSLAM allows for life-long mapping by fusing the probabilistic local feature of FAB-MAP with the pose cell filtering and experience mapping of RatSLAM.

RatSLAM performance alone was evaluated on a 66km long urban street, with many loops (MILFORD; WYETH, 2008). Results showed that RatSLAM is capable of building maps online, closing loops and performing global localization. However, to do global localization, it requires several image frames. RatSLAM was also capable of building a topological correct map across day-night cycles (MILFORD et al., 2014) with improved patch-based visual odometry, whole image matching and some post validation steps. For the first time, a single-image matching approach achieved superior performance than the state-of-the-art sequence-based method SeqSLAM (MILFORD; WYETH, 2012).

In (FOX; PRESCOTT, 2010), authors proposed a navigation system inspired on hippocampal capabilities of associative memory and spatial navigation. They proposed a

mapping of the hippocampal formation onto a Temporal Restricted Boltzmann Machine (TAYLOR; HINTON; ROWEIS, 2007). In (SAUL; PRESCOTT; FOX, 2011), they present an attempt to scale the model up towards robotic navigation using real world images, though still limited by finite set of locations and dependent of SURF features (BAY et al., 2008).

In (CHEN et al., 2014a; CHEN et al., 2015a) authors present a biologically-inspired multi-scale mapping system mimicking the rodent multi-scale map. Unlike hybrid metric-topological multi-scale robot mapping systems, their system is homogeneous, distinguishable only by scale, like rodent neural maps. They train each network to learn and recognize places at a specific spatial scale, and then combining the output from each of these parallel networks. The results demonstrate that a multi-scale approach universally improves place recognition performance and is capable of producing better results than state-of-the-art FAB-MAP.

SeqSLAM is yet another appearance-based SLAM that calculates the best candidate matching of an image within a segment of a sequence of previously seen images. Although this approach can handle normal and extreme conditions in the environment appearance, even for long running distances, it is not capable of performing continuous global localization as the previous authors' approach VibGL (LYRIO JÚNIOR et al., 2014) is. In (MILFORD et al., 2015), the authors improved the viewpoint invariance of the SeqSLAM algorithm by using a Convolutional Neural Network (CNN) to estimate depth from mono images in order to generate warped images representing lateral camera shifts.

In (CHEN et al., 2014b) authors presented a place recognition technique based on CNN, by combining the features learnt by CNNs with a spatial and sequential filter. They use a pretrained network called Overfeat (SERMANET et al., 2014) to extract the output vectors from all layers (21 in total). For each output vector they build a confusion matrix from all images of training and test datasets using Euclidean distance to compare the output feature vectors. Following they apply a spatial and sequential filter before comparing the precision-recall curve of all layers against FAB-MAP and SeqSLAM. Although, middle layers 9 and 10 perform way better (85.7% recall at 100% precision) than the 51% recall rate of SeqSLAM, the authors do not find any automatic mechanism for selecting the best layer. In (CHEN et al., 2015b), authors use Large Margin Nearest Neighbor (LMNN) algorithm (WEINBERGER; SAUL, 2009) to learn a Mahalanobis distance metric which clusters images from the same place and moves apart images from different locations. Their results demonstrate overall improvements in place recognition performance across two distinct datasets and three different feature types, including CNN features. Improvements were found as well for single and sequence-based image matching cases.

In (ARANDJELOVIC et al., 2017), the authors addressed the problem of large-scale

place recognition, where the four most relevant contributions to solve the problem are as follows. Firstly, a convolutional neural network (CNN) architecture that is trainable in an end-to-end manner directly to the place recognition task. The main component of this architecture, NetVLAD, is a new VLAD layer inspired by the "Vector of Localized Descriptors in Aggregates" commonly used in image recovery. The layer is easily connectable in any CNN architecture and able to learn via backpropagation. Secondly, they proposed a new loss ranking for weakly supervised classification, which allows end-to-end learning of network's parameters from images that depict the same place over time, downloaded from Google Street View Time Machine. Thirdly, they develop an efficient training procedure that can be applied to large-scale place recognition with noisy labels. Finally, they show that the proposed architecture and training procedure significantly outperforms both non-learned image features and CNN features in challenging place recognition datasets.

In (LYRIO JÚNIOR et al., 2014), authors employ a Weightless Neural Network (WNN) with holistic and feature-based architecture (DE SOUZA et al., 2008) to global localize a self-driving car. Their method, called VibGL, has linear complexity like FAB-MAP and does not require any a priori pre-processing (e.g. to build a visual vocabulary). It achieved average pose precision of about 3m in a few kilometers-long route without using any temporal information. VibGL was later integrated into a vision-only metric SLAM system named VIBML (JÚNIOR et al., 2015). To this end, VibGL stores landmarks along with GPS coordinates for each image during map construction. VIBML performs position tracking by using stored landmarks to search for corresponding ones in current observed images. Additionally, VIBML employs an Extended Kalman Filter (EKF)(THRUN; BURGARD; FOX, 2005) for predicting the robot state based on a car-like motion model and corrects it using landmark measurement model (THRUN; BURGARD; FOX, 2005). VIBML was able to localize an autonomous car with average positioning error of 1.12m and with 75% of the poses with error below 1.5m in a 3.75km path (JÚNIOR et al., 2015).

Milford has also investigated the visual place recognition problem under the DTW framework in (MILFORD, 2012) and (MILFORD, 2013), achieving robust results under small and poor-quality images. His simple approach, that uses Sum of Absolute Differences (SAD) to represent each frame by a scalar real value, allows for direct use of the classical DTW (designed for univariate time series). That confirms the importance of comparing sequences of images and not just single images; it also sets the minimal standards and opens up the field for more sophisticated approaches. However, the problem complexity increases with the size of dimensional space, what could make univariate classifiers to fail when extended to higher dimensions (ALONSO et al., 2008).

To better handle the multivariate case, it was proposed in (FORECHI et al., 2016) and will be further discussed in this thesis a combination of a feature subset ensemble (ALY; ATIYA, 2006) of kNN-DTW classifiers defined in Hamming space. More specifically,

the Place Recognition problem is addressed as a multivariate binary time series extracted from image sequences and classified by an ensemble of kNN classifiers using the DTW-Hamming distance as a dissimilarity measure. Differently from the stacking ensemble method presented in (ALONSO et al., 2008), the proposed method employs a feature subset ensemble of kNN-DTW directly on the multidimensional sequence space instead of dividing the sequence in several uni-dimensional sequences to perform kNN-DTW on them individually for later combining them in the multidimensional space. To the best of our knowledge such approach combination of a feature subset ensemble using kNN classifiers and DTW-Hamming distance has never been proposed before.

2.2 Visual Localization

Visual Localization refers to the problem of inferring the 6-DOF (Degrees Of Freedom) camera pose associated to where images were taken. This problem is commonly formulated as 2D-to-3D registration problem (MIDDELBERG et al., 2014; DONOSER; SCHMALSTIEG, 2014; LI et al., 2012; LIM et al., 2012; SATTLER; LEIBE; KOBBELT, 2011; IRSCHARA et al., 2009). Consisting, basically, in finding matches between 2D image points and 3D structure points for latter applying the Direct Linear Transformation (DLT) (HARTLEY; ZISSERMAN, 2004) to get the pose vector. Nevertheless, these approaches might fail to correctly localize images when the 3D structure points are not available.

In (SONG et al., 2015), authors proposed a system to localize an input image according to the position and orientation information of multiple similar images retrieved from a large reference dataset using nearest neighbor and SURF features (BAY et al., 2008). The retrieved images and their associated pose are used to estimate their relative pose with respect to the input image and find a set of candidate poses for the input image. Since each candidate pose actually encodes the relative rotation and direction of the input image with respect to a specific reference image, it is possible to fuse all candidate poses in a way that the 6-DOF location of the input image can be derived through least-square optimization. Experimental results show that their approach performs comparable with civilian GPS devices in image localization. Perhaps applied to front-facing camera movements, their approach might not work properly as most of images would lie along a line causing the least-square optimization to fail.

In (AGRAWAL; CARREIRA; MALIK, 2015), the task of predicting the transformation between pair of images was reduced to 2D and posed as a classification problem, so that the components of the 3-DOF camera pose were individually binned into 20 uniformly spaced bins. For training they followed Slow Feature Analysis (SFA) method (CHOPRA; HADSELL; Y., 2005) by imposing the constraint that temporally close frames should have similar feature representations disregarding either the camera motion and

the motion of objects in the scene. This may explain the authors' decision to treat visual odometry as classification problem, since the adoption of SFA should discard "motion features" and retain the scale-invariant features, which are more relevant to the problem as classification problem than to the original regression problem. Nevertheless, the purpose was to investigate whether the awareness of egomotion could be used as a supervisory signal for feature learning. To this end, their results showed that using the same number of training images, features learned using egomotion as supervision compare favorably to features learned using class-label as supervision on the tasks of scene recognition, object recognition, visual odometry and keypoint matching.

Kendall et. al. proposed a monocular 6-DOF relocalization system named PoseNet (KENDALL; GRIMES; CIPOLLA, 2015). PoseNet employs a convolutional neural network to regress the 6-DOF camera pose from a single RGB image in an end-to-end manner with no additional engineering or graph optimization. The system can operate indoors and outdoors in real time. It obtained approximately 3m and 6 deg accuracy for large scale (500 x 100m) outdoor scenes and 0.6m and 10 deg accuracy indoors. Its most salient features relate greatly to the environment structure, since camera moves around buildings and mostly of the time it faces them.

The task of locating the camera from single input images can be modelled as a Structure of Motion (SfM) problem in a known 3D scene. The state-of-the-art approaches do this in two steps: firstly, projects every pixel in the image to its 3D scene coordinate and subsequently, use these coordinates to estimate the final 6D camera pose via RANSAC. In (BRACHMANN et al., 2017), the authors proposed a differentiable RANSAC method called DSAC in an end-to-end learning approach applied to the problem of camera localization. So far deep networks have not been able to overcome traditional approaches based on RANSAC. The authors show that by directly minimizing the expected loss of the output camera pose, robustly estimated by the DSAC, they have achieved an increase in accuracy.

In (OLIVEIRA et al., 2017), the authors proposed a metric-topological localization system based on images that consists of two deep networks trained for visual odometry and place recognition, respectively, whose predictions are combined by a successive optimization. Those networks are trained using the output data of a high accuracy LiDAR-based localization system similar to ours. The VO network is a Siamese-like CNN, trained to regress the translational and rotational relative motion between two consecutive camera images using a loss function similar to (KENDALL; GRIMES; CIPOLLA, 2015), which requires an extra balancing parameter to normalize the different scale of rotation and translation values. For the topological system part, they discretized the trajectory into a finite set of locations and trained a deep network based on DenseNet (HUANG et al., 2017) to learn the probability distribution over the discrete set of locations.

Summing up, a place recognition system capable of correctly locating a robot

through discrete locations serves as an anchor system because it limits the error of the odometry system that tends to drift. The natural question that arises is how to combine different sources of information without carrying an extra burden. Visual odometry lies in a continuous configuration space while topological locations lies in a discrete configuration space. By recalling that place recognition system stores pairs of image-pose about locations, one approach would be to find a relative estimate of the camera pose, considering as input the recollected image-pose from the place recognition system and a live camera image in order to output a 6-DOF relative camera pose. And that applied to the recollected camera pose takes the live camera pose in a global frame of reference.

The Place Recognition system presented here is based on Weightless Neural Networks such as the ones first proposed in (FORECHI et al., 2015; FORECHI et al., 2016), whilst the Visual Localization system is based on a Convolutional Neural Network similar in spirit with (MELEKHOV et al., 2017). In the end, our hybrid weightless-weighted approach does not require any additional system to merge the results of the topological and metric systems. Our relative camera pose system was developed concurrently to (MELEKHOV et al., 2017) and differs in network architecture, loss function, training regime and application purpose. While they regress translation and quaternion vectors, we regress translation and rotation vectors; while they use the ground-truth pose as supervisory signal in L^2-norm with an extra balancing parameter (KENDALL; GRIMES; CIPOLLA, 2015); we employ the L^2-norm on 3D point clouds transformed by the relative pose; while they use transfer learning on a pre-trained Hybrid-CNN (ZHOU et al., 2014) topped with two fully-connected layers for regression, we train a Fully Convolutional Network (LONG; SHELHAMER; DARRELL, 2015) from scratch. Finally, their system application is more related to Structure from Motion, where images are object-centric and ours is route-centric. We validated our approach on real world images collected using a self-driving car while theirs were validated solely using indoor images from a camera mounted on a high-precision robotic arm.

3 Background

This chapter describes the state of the art of weightless and weighted neural networks necessary for grounding this work.

3.1 Weightless Neural Networks

Weightless Neural Networks (WNN), also called n-tuple classifiers (ROHWER, 1995), are powerful machine learning techniques (MORCINIEC; ROHWER, 1995) that offer fast training and test (ALEKSANDER, 1998). In contrast with standard feed-forward Neural Networks (MISRA; SAHA, 2010), which store knowledge in the form of synaptic weights, WNN store knowledge mainly in Random Access Memory (RAM) inside the network's neurons, which makes training and test fast. Their first version was proposed by Bledsoe and Browning in 1959 (BLEDSOE; BROWNING, 1959) and, after that, many other implementations were proposed (see (LUDERMIR et al., 1999) for a review). All implementations share the following two properties: (i) the synapses of WNN neurons do not carry weights (hence the denomination of these neural networks), and each of them only collect a single bit (zero or one) from the network's inputs; and (ii) the network's knowledge is stored in binary look-up tables inside the network's neurons. It is important to note that the interconnection pattern between a layer of WNN neurons and its inputs stores knowledge as well, but typically a static knowledge, i.e. once created, the interconnection pattern does not change.

In WNNs, the synapses of each neuron of a layer of neurons collect a vector of bits from the network's input (or other neural layers of the network) and use this vector of bits as a key to access the neuron's look-up table. During training, this access key is used to store an expected output, associated with the input vector, into the look-up table (supervised learning). During test, the input vector is used as an access key for reading one of the previously learned outputs from the look-up table. In its most simple implementation, the look-up table is a random access memory (RAM); therefore, training and test can be made in one shot. Training basically consists of storing the desired output value into the look-up table's address that is associated with the neuron input vector, while test consists of retrieving the information stored in the look-up table's address that is associated with the neuron input vector (ALEKSANDER, 1966). However, in this simple implementation, the neuron memory size may become prohibitive if the neuron input vector is too large (the neuron memory size for this simple WNN is equal 2^p, where p is the neuron input vector size).

To overcome this limitation, many solutions have been investigated. A well-known solution is the Wilkes, Stonham and Aleksander Recognition Device (WiSARD) proposed in (ALEKSANDER; THOMAS; BOWDEN, 1984). WiSARD overcomes this limitation by splitting the p-sized neuron input vector in q segments, each one used to address a specific RAM memory module of size $2^{p/q}$. During training, a p-sized training input pattern is split in q parts and each p/q-sized input sub-pattern is used to address the corresponding RAM module; the addressed position in each RAM module receives (stores) the label value (RAM module's output value) associated with the input pattern. During test, the p-sized test input pattern is also split in q parts and each p/q-sized input sub-pattern is used to address the corresponding RAM module; the neuron's output value is the most frequent label value observed among the outputs of the q RAM modules. Thanks to this organization, the total amount of memory required per neuron is reduced from 2^p to $q \times 2^{p/q}$. This framework, and many variants, have been used for a large variety of applications, including indoor positioning systems (CARDOSO et al., 2013), robot localization system (YAO et al., 2003; CORAGGIO; De Gregorio, 2007; NURMAINI; HASHIM; JAWAWI, 2009), recognition of DNA chains (SOUZA et al., 2012), feature tracking in images (FRANCA et al., 2010), data clustering (CARDOSO et al., 2011), and audio recognition (PRADO et al., 2007). However, given the constraints imposed by dividing the neuron input vector in q segments, the representation capacity of the network is reduced (ALEKSANDER; GREGORIO; FRANÇA, 2009), which hinders its generalization ability (ALEKSANDER; MORTON, 1995). As a result, the performance of applications using WiSARD may be lower than that obtained with related techniques, such as Virtual Generalizing Random Access Memory WNN (VG-RAM WNN, or VG-RAM for short). This can be seen in (BERGER et al., 2013), where the accuracy of a traffic sign recognition system with WiSARD and VG-RAM were compared.

The VG-RAM WNN is a type of WNN that only requires storage capacity proportional to the training set (ALEKSANDER, 1998). They have shown high classification performance for a variety of multi-class classification applications, including text categorization (DE SOUZA et al., 2008; DE SOUZA et al., 2009), face recognition (DE SOUZA et al., 2008; DE SOUZA; BADUE; PEDRONI, 2010; DE MORAES; DE SOUZA; BADUE, 2011), and traffic sign detection and recognition (BERGER et al., 2013; BERGER et al., 2012; DE SOUZA et al., 2013). During training, VG-RAM neurons store pairs of associated input-output patterns, i.e. input binary vectors and associated outputs, instead of only the output, as mentioned above for the WNNs. Thanks to their training method, VG-RAM's memory footprint is optimized, and its training performance in terms of time is maintained. One disadvantage of VG-RAMs, however, is their test time, which depends on the size of their neurons' memory, i.e. the number of trained input-output pairs. During test, VG-RAM's neurons compute their outputs by searching for the learned input-output pair whose input is the closest to the current neuron's input – the output of this input-output

pair is used as the neuron's output.

3.1.0.1 VG-RAM WNN Architecture

The architecture of VG-RAM WNNs comprises neural layers with many neurons connected to input layers (e.g. images or other neural layers) through a set $\mathbf{S} = \{s_1, \ldots, s_p\}$ of synapses. Since VG-RAM neurons generate outputs in the form of label values, t, the output of a neural layer can also function as an image, where each pixel is the output of a neuron.

The synapses of a VG-RAM neuron are responsible for sampling a binary vector $I = \{i_1, \ldots, i_p\}$ (one bit per synapse) from the input layer according to the neuron's receptive field. Each bit of this vector is computed using a synapse mapping function that transforms non-binary values in binary values. Individual neurons have private memories, i.e. look-up tables, that store sets $\mathbf{L} = \{L_1, \ldots, L_j, \ldots, L_m\}$ of learned binary input vectors I and corresponding output labels t, i.e. $Lj = (I_j, t_j)$ (input-output pairs). An illustration of a single layer of such a network is presented in Figure 1.

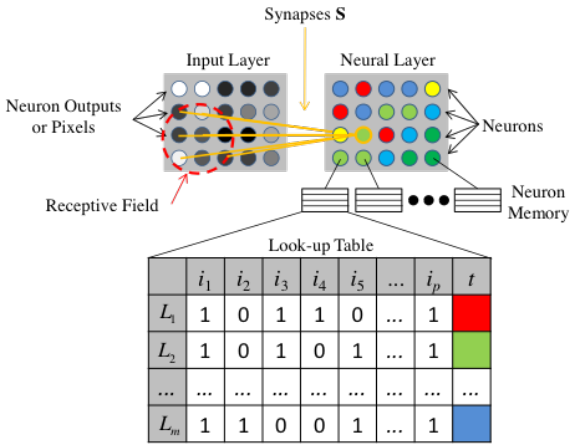


Figure 1 – Illustration of a single layer VG-RAM WNN. The VG-RAM WNN Neural Layer comprises many neurons that are connected to the Input Layer (e.g. an image) by a set of Synapses $\mathbf{S} = \{s_1, \ldots, s_p\}$. These synapses sample binary vectors $I = \{i_1, \ldots, i_p\}$ from the Input Layer according to the Receptive Field of each neuron. The bits of these vectors are computed using a synapse mapping function that transforms non-binary values in binary values. Each neuron of the Neural Layer has a private memory (Look-up Table) of size m that stores a set $\mathbf{L} = \{L_1, \ldots, L_j, \ldots, L_m\}$ of $L_j = (I_j, t_j)$ input-output pairs learned during training. Each neuron shows an output activation value, t, read from its memory (Look-up Table column t and Neural Layer colored circles). This value corresponds to the output of the input-output pair, j, whose input I_j is the closest to the current binary input vector I extracted by the neuron's synapses during testing phase. Redrawn after (AGUIAR et al., 2014).

VG-RAMs operate in two phases: a training phase, in which neurons learn new pairs of binary input vectors and corresponding output labels (input-output pairs); and

a test phase, in which neurons receive binary input vectors and respond with the label values associated with the closest binary input vectors in the input-output pairs previously learned.

More specifically, in the training phase, each neuron includes a new input-output pair, or line L, in its local memory as follows. Firstly, an input image is set in the Input Layer of the VG-RAM (Figure 1). Secondly, the corresponding Neuron Outputs are set in the Neural Layer of the VG-RAM (expected output value t of each neuron for the input image set). Finally, one input-output pair (binary input vector I and corresponding output label t) is extracted for each neuron and is added into its memory as a new line $L = (I, t)$. In the test phase, each neuron computes an output label t as follows. Firstly, an input image is set in the Input Layer of the VG-RAM. Secondly, a binary input vector I is extracted for each neuron. Finally, each neuron searches its memory to find the input-output pair $L_j = (I_j, t_j)$ whose input I_j is the closest to the extracted input I, and sets the corresponding Neuron Output of the Neural Layer with the output value t_j of this pair. In case of more than one pair with an input at the same minimum distance of the extracted input, the output value is randomly chosen among them. The memory search is sequential and the distance function is the Hamming distance.

The Hamming distance between two binary patterns can be efficiently computed at machine code level in current 64-bit CPUs and GPUs of personal computers using two instructions: one instruction to identify the bits that differ in the two binary patterns, i.e. bit-wise exclusive-or; and another instruction to count these bits, i.e., population count instruction.

VG-RAM WNNs scores high machine learning performance within an acceptable response time for many daily-life applications with (i) not a very large number of training examples or (ii) not requiring real-time performance. However, lately, the amount of training data has grown together with the need for real-time machine learning applications. Given the architecture of this type of network, two major limitations arise when dealing with applications with large training datasets: test time and memory usage. The first is associated with the search for the closest pattern, which is performed sequentially in the memory of each neuron. The second is associated with the pairs of examples that have to be kept in memory. These hinder not only the wide use of VG-RAM with real time applications, but also its use with devices with limited amount of memory, such as low-power embedded systems (AGUIAR et al., 2014). Both problems are related to the size of $\mathbf{L}$, which may slow-down the sequential search and increase the memory usage. The size of $\mathbf{L}$ is proportional to the number of synapses p and the number of samples m, where the latter is more likely to grow in future relevant applications.

3.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) (LECUN et al., 1998) have their roots in Fukushima's Neocognitron model (FUKUSHIMA, 1980), which in turn takes inspiration in Simple and Complex Cells receptive fields and hierarchical structure discovered by Hubel and Wiesel in monkey striate cortex (HUBEL; WIESEL, 1968). Neocognitron network is self-organized during an unsupervised learning training and is able to recognize stimulus patterns based on shape similarity regardless of their position. The network consists of an input layer followed by three interleaved layers of Simple and Complex cells. Where, besides the inhibitory cells role, Simple cells act as filters and Complex cells perform pooling over Simple cells output.

A filter, in Image Processing (GONZALEZ; WOODS, 2007), refers to accepting (passing) or rejecting certain frequency components. The equivalent in the spatial domain is known as kernel. A kernel consists of a matrix W of dimensions W_x and W_y, i.e. receptive field size. W is filled with predefined values that applied on the pixels of input image f transforms it into another image, now filtered. The filtered image g is then generated while the kernel center traverses each pixel of the input image f in an operation known as convolution. Each convolution step generates a new value for the pixel (x, y) of the filtered image g, which is computed by summing the products of each element w_{ij} of the kernel W by the pixel $(x + i, y + j)$ within the region delimited by the kernel in each step. Given a kernel W of dimensions $W_x \times W_y$, take $W_x = 2a + 1$ and $W_y = 2b + 1$, where a and b are positive integer numbers. This simplified formulation allows odd-sized filters only but even-sized kernels are also possible. It is assumed odd-sized kernels and $w(0, 0)$ is considered to be the middle element in W just to easy indexing in the equation below:

$$g(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) f(x - i, y - j), \tag{3.1}$$

where x and y vary so that each element in W traverses all pixels in f, considering default stride of one pixel in both x and y directions, respectively, $s_x = 1$ and $s_y = 1$. When stride is greater than one in at least one of the dimensions, the resulting filtered image g is smaller than input image f in that corresponding dimension. Summing up, stride controls how the filter convolves around the input image by setting the amount by which the filter is shifted at each convolution step. Another kernel parameter that controls the resulting size g_x and g_y of filtered image g is through padding the input image borders with zeros. Notice that the g_x and g_y dimensions decrease by $2a$ and $2b$, respectively, after convolutions. Let's say, for example, one applies 5×5 filters to a 32×32 input image without padding $p_x = p_y = 0$. The resulting filtered image size would be 28×28. Now if one wants the filtered image size remains the same, one can apply a zero padding of size $p_x = 2$ and $p_y = 2$ to that input image.

Often, when convolving images the input is a volume instead of a plane. To do that an extra dimension should be added to a kernel to map the input image channels dimension $c \in \mathbb{N}_{\geq 1}$. The modified convolution equation becomes

$$g(x, y, c) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \sum_{m=1}^{c} w(i, j, m) f(x - i, y - j, m) \tag{3.2}$$

In Image Processing the kernel matrix coefficients w_{ij} are set a priori according to the problem. For instance, whether the desired operation is smoothing, one can set all $w_{ij} = 1/(W_x \times W_y)$ for a kernel of dimensions W_x and W_y. Whether the desired operation is finding edges, one can set the kernel matrix W to a Laplacian Edge Detector as in the Figure 2.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 2 – Laplacian Edge Detector Kernel with dimensions 3×3.

Filters such as the Laplacian, have the disadvantage of being susceptible to noise and not being robust to illumination changes. This would require further normalization in the input image and might not work properly in different scenarios without parameter engineering. Much of these drawbacks are related to the fact that the kernel matrix coefficients are set fixed. To overcome such limitations one can resort to a particular neural network architecture that learns from the data what should be the kernel matrix coefficients. Such neural networks are called Convolutional Neural Networks (CNN) (LECUN et al., 1998).

This kind of Artificial Neural Network implements convolution as in the Equation 3.2 (without the optional bias term) while process the input image f in the forward step. The output is again a filtered image g_k for each filter bank k. As one can stack many filters the set $G = g_1, ..., g_k$ is called feature maps. One can have many layers similar to that, one on top of other, as the output depth dimension k of feature maps of the previous convolutional layer is mapped to the input depth dimension c. The resulting output volume size will be $k \times W_x \times W_y$, where

$$\begin{aligned} g_x &= floor((f_x + 2p_x - W_x)/s_x + 1) \\ g_y &= floor((f_y + 2p_y - W_y)/s_y + 1) \end{aligned} \tag{3.3}$$

The backward step involves computing the derivatives of the error with respect to each weight coefficient w_{ij}^k of each feature map g_k of each filter bank W^l of each layer l through backpropagation (RUMELHART; HINTON; WILLIAMS, 1986).

So far, a convolutional layer just compute affine transformations and, to solve highly non-linearity problems, is necessary to add non-linearity to the output layer. There is a variety of activation functions one can add to the output layer, while the most popular for learning from image data are the Rectified Linear Units (ReLU) (NAIR; HINTON, 2010). ReLU's use the activation function $g(x) = max(0, x)$.

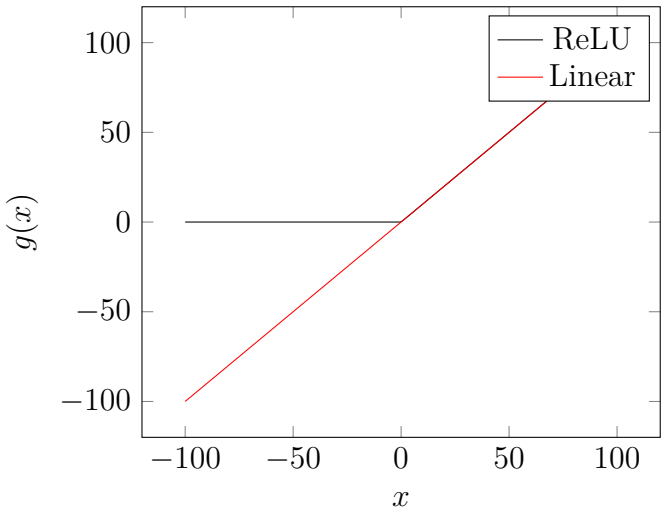


Figure 3 – Rectified Linear Units *vs.* Linear Units.

ReLU's are easy to optimize because they are very similar to linear units, as shown in Figure 3. The only difference between a linear unit and ReLU is that ReLU is zero for half of its domain. This causes the derivatives of ReLU's to remain large whenever the unit is active (GOODFELLOW; BENGIO; COURVILLE, 2016).

A typical convolutional network layer consists of three main steps. In the first step, the layer executes several convolutions in parallel to produce a feature map of linear activations. In the second step, each linear activation is transformed through a non-linear activation function, such as ReLU. In the last step, a pooling function is used to further modify the output of the layer (GOODFELLOW; BENGIO; COURVILLE, 2016). A pooling function replaces the output of the convolutional network layer in a given location with a value representing the nearby outputs and, consequently, reduces the output dimension as occurs for kernel stride. For example, the max pooling operation returns the maximum input value within a rectangular neighborhood. Pooling helps make representation increasingly invariant to small translations of the input.

Dropout (SRIVASTAVA et al., 2014) is a computationally inexpensive but powerful method of regularizing a large family of models. During training, Dropout masks parts of the input using binary samples from a bernoulli distribution. Each input element has a probability of p of being dropped or zeroed. For comparison, dropout can be thought of as a method for making bagging practical for many large neural networks. Remember that to learn with bagging, we define k different models, construct k different data sets by sampling

the training set with replacement, and then train model i in data set i. Dropout aims to approximate this process by training an ensemble consisting of all sub-networks that can be formed by removing non-output units from an underlying base network (GOODFELLOW; BENGIO; COURVILLE, 2016). Considering an implementation where the outputs are scaled by a factor of $1/(1-p)$ during training simplifies the forward step during evaluation. This allows all elements of the input to be considered in evaluation.

Figure 4 shows the filters learned by the first convolutional layer of the AlexNet Network (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) that won the Imagenet Large Scale Visual Recognition Challenge 2012 (DENG et al., 2009). ImageNet dataset has over than 15 million labeled high-resolution images belonging to roughly 22,000 categories. In the Figure 4, each of the 96 filters has size $11 \times 11 \times 3$ and the input image size is $224 \times 224 \times 3$. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs. The top three filter rows were learned on a GPU while the bottom three filter rows were learned on another. Curiously, splitting the convolutional layers into two GPU's leads one GPU to learn largely color-agnostic kernels while the other GPU learns color-specific kernels.

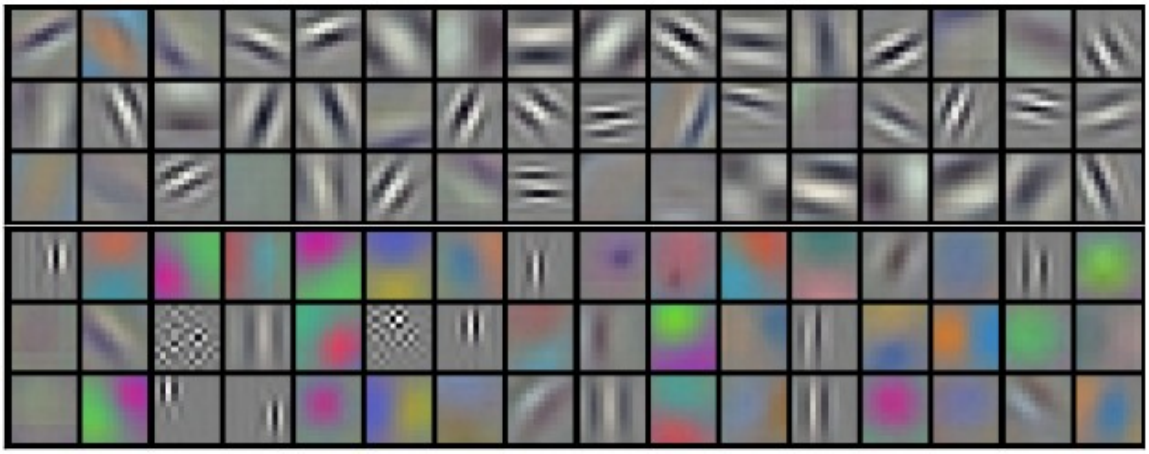


Figure 4 – Filters learned by first convolutional layer of AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Each of the 96 filters has size $11 \times 11 \times 3$ and the input image size is $224 \times 224 \times 3$. The top three filter rows were learned on a GPU while the bottom three filter rows were learned on another. Figure courtesy of (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

4 Appearance-based Global Localization with a Hybrid WNN-CNN Approach

This chapter presents the proposed approach to solve the global localization problem in the space of appearance in contrast to traditional Global Positioning System (GPS) sensors. The proposed approach is twofold: (i) a Weightless Neural Network (WNN) to solve place recognition as a classification problem, and (ii) a Convolutional Neural Network (CNN) to solve visual localization as a metric regression problem. Given a live camera image the first system recollects the most similar image and its associated pose within a predefined range, and the latter compares the recollected image to the live camera image in order to output a 6D relative pose. The final outcome is an estimation of current robot pose, which is composed by the relative pose given by system (ii) applied to the recollected image pose given by system (i).

Section 3.1.0.1 of the previous chapter presented VG-RAM's architecture designed to solve image-related classification problems by employing a committee of VG-RAM units, called neurons, which are individually responsible to represent the binary patterns extracted from the input and their corresponding label. Those neurons are organized in layers and can also serve as input to further layers in the architecture. As a machine learning method, its use consists of a training phase to store pairs of inputs and labels, which will be compared to new inputs in the testing phase. Considering that its testing-time increases linearly with the number of training samples, in Section 4.1 is presented a faster neuron memory search leveraged by an indexed data structure with sub-linear runtime for uniformly distributed patterns. Unfortunately, this data structure requires a significant amount of additional memory to index the memory of each VG-RAM neuron. Thus, to compensate this memory overhead, in Section 4.1.1 is presented a clustering method over the VG-RAM neuron's memory to compress the number of representative input patterns associated to each label in each neuron memory.

4.1 Faster Neuron Memory Search with Fat-Fast VG-RAM

The search for the closest pair of VG-RAM neuron is sequential, which is costly in terms of time if there are many training samples. To cope with this problem, in this thesis we present a Fat-Fast version of VG-RAM WNNs. Our Fat-Fast VG-RAMs employ multi-index chained hashing for fast searching in the neuron's memory. Even though our chained hashing technique increases the VG-RAM memory's consumption (fat), it reduces

test time substantially (fast), while keeping most of the machine learning performance of VG-RAM. To address the increase in memory consumption of the Fat-Fast VG-RAMs, we have employed data clustering techniques for reducing the overall size of their neurons' memory (a preliminary study of memory clustering techniques for VG-RAM is presented in (AGUIAR et al., 2014)).

The search for the nearest input-output pair $L_j = (I_j, t_j)$ whose input I_j is the closest to the neuron input vector I is a k-nearest neighbor search in a Hamming space, where k is equal to 1. To efficiently address this problem at bit level and return the exact k nearest neighbors, Norouzi et al. (NOROUZI; PUNJANI; FLEET, 2012; NOROUZI; PUNJANI; FLEET, 2014) employed a method where the searched pattern I is divided in sub-patterns that, in turn, are used to index multiple hash tables (one table per sub-pattern). They have shown that a good compromise between access speed and memory usage can be found for the hash tables. Following in the same direction, this thesis brings the multiple hash tables idea into the VG-RAM WNN in order to speed up the test phase. Differently from the approach mentioned above, we are not restricted to an exact match of the nearest neighbor. Instead, we are interested on a good trade-off between classification accuracy and test time, when considering real world applications.

In the current approach, multiple hash tables are created for the memory of each VG-RAM neuron considering the binary space defined by its input vector I; we named these modified VG-RAM neurons Fat-Fast VG-RAM neurons. Figure 5 gives an overview of our multi-index hash approach to the neuron memory architecture.

Since we are working with bits, a very effective hash function $\Psi(I)$ can be created by simply dividing the binary input vector $I = i_1, \ldots, i_p$ into h binary segments, i.e., binary sub-vectors $\mathbf{V} = \{V_1, \ldots, V_k \ldots, V_h\}$. Each binary sub-vector comprises p/h bits regularly sampled from I, i.e., $V_k = 0, 1, \ldots, v, \ldots, 2^{p/h}$. $\mathbf{V}$ is used as index to the multi-index hash table $\mathbf{H} = H_1, \ldots H_k \ldots, H_h$, where $1 \leq h \leq p$. Each V_k is directly used as an address of the respective hash table, H_k, of $\mathbf{H}$. Each hash table H_k has $2^{p/h}$ unique buckets, $B_{k,v}$, containing $|B_{k,v}|$ indexes to memory lines. Buckets are necessary because several learned memory lines may share the same sub-vector code value, i.e., may have conflicting V_k addresses (e.g., see conflicting lines L_1 and L_3 of bucket $B_{1,0}$ in Figure 5). Buckets can store references to up to m memory lines that have conflicting addresses, where m is the number of learned input-output pairs, i.e., $|\mathbf{L}|$. An efficient implementation of buckets can be achieved with arrays of variable size.

To train a Fat-Fast neuron, it is necessary to fill its memory with input-output pairs $L_j = (I_j, t_j)$ following the same training procedure of a standard VG-RAM. In addition, it is necessary to populate the multi-index hash table, $\mathbf{H}$, with references to each one of the learned pairs. For that, the binary codes $\mathbf{V}$ are extracted from I_j using $\Psi(I_j)$, and one reference to L_j is added to the respective bucket of each hash table of $\mathbf{H}$.

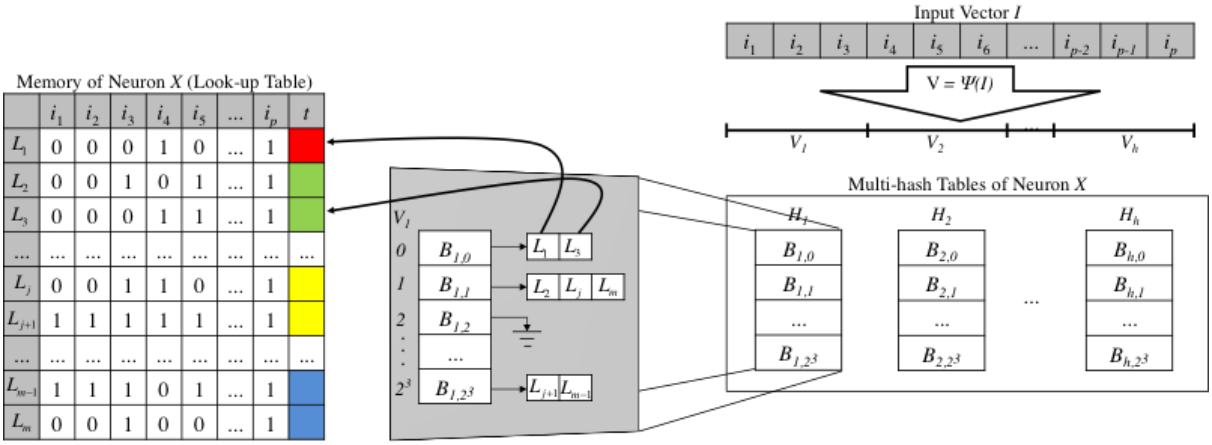


Figure 5 – Fat-Fast VG-RAM neuron memory. A Fat-Fast Neuron X reads, via its synapses, the binary input vector I. The figure illustrates the process of mapping an input vector, I, of size equal to p bits, into buckets, $B_{k,v}$, that stores references to the set of memory lines that have to be inspected during memory search (see arrows above). The input vector I is firstly divided into sub-vectors, $\mathbf{V} = \{V_1, \ldots, V_k \ldots, V_h\}$, where $V_k = \{0, 1, \ldots, v, \ldots, 2^{p/h}\}$. These sub-vectors are used to address the multi-index hash table, $\mathbf{H} = \{H_1, \ldots H_k \ldots, H_h\}$. Each address, V_k, points to a bucket $B_{k,v}$ of lines, L, that should be considered in the search. Note that this illustration uses binary codes, V_k, of 3 bits only for clarity. Redrawn after (FORECHI et al., 2015).

During test, fast search for the nearest neighbor of a binary input vector I can be performed for each Fat-Fast VG-RAM neuron as follows. Firstly, the function $\Psi(I)$ is used to retrieve $\mathbf{V}$. Subsequently, each V_k is used as an address to the bucket $B_{k,v}$. Finally, the input vector I_j of each memory line $L_j = (I_j, t_j)$ of each selected bucket is examined as a candidate for the nearest input to the neuron input vector I. Note that, all candidates are compared with I using the Hamming distance between the full bit vectors, i.e., the p bits of I and I_j. The Fat-Fast VG-RAM output is the label t_j of the nearest line, L_j. If more than one candidate line is at the same closest distance to I, the neuron output is randomly selected among them.

It may occur that the number of lines in a selected bucket is zero. In these cases, no memory line is considered for that bucket. If none of the h selected buckets has candidate lines, the neuron output is taken from a random line of memory. On the other hand, it may also occur that there are too many lines in one or more selected buckets. In such cases, the searching time could increase substantially. Hence, in these cases, a maximum of r candidates in each large bucket (bucket size bigger than r) is randomly selected for comparison with the input vector I, where r is a Fat-Fast parameter.

Indeed, the parameter r is required to be tuned manually and accordingly to each particular application, i.e., one needs to balance the trade-off between accuracy and runtime, conditioned to the amount of available RAM. As discussed, a higher parameter

r increases the accuracy close to VG-RAM levels, but at the same time slows down the system, and vice versa. The multi-index hash tables of Fat-Fast VG-RAM WNN neurons are only used as a mean to reduce the number of memory lines to be examined. In practice, converting the vector I into $\mathbf{V}$ can be costly if performed inappropriately. In order to avoid unnecessary processing, in our Fat-Fast implementations, we kept p/h equal to the number of bits used in the basic types of common programming languages, which facilitates retrieving the address V_k of each bucket. Yet, it is possible to determine an optimal p/h ratio that minimizes the total number of collisions inside the multi-index hash (NOROUZI; PUNJANI; FLEET, 2012; NOROUZI; PUNJANI; FLEET, 2014). Thus, given a fixed number of synapses p, the optimal value for h is $\log_2(m)$, where m is the total number of training examples. But here we consider p/h fixed because, otherwise, it could affect the runtime measurements given that h is computed based on different values of m.

The search time of each neuron in standard VG-RAM WNN is $O(m)$ because the Hamming distance has to be calculated for every example in the memory. The search time with the multi-index hash tables approach is at the most $O(h \times r)$ because only up to r lines per hash table will have the Hamming distance computed. For applications with large number of training examples, the gain might be huge for small r because $h \ll m$. Since we are interested in the overall machine learning power of the network, it is important to know the effect of changes in r. The larger the value of r is, the more the search for the nearest neighbor approximates the exact search. In the other hand, the smaller the value of r is, the more is the impact on machine learning performance, since some good neuron memory lines could be left out of the Hamming distance comparison.

4.1.1 Memory Size Reduction

Fat-Fast VG-RAM WNN takes sub-linear time for testing new inputs thanks to the way it re-organizes the neurons' memory using a special multi-index hash system but, on the other hand, it increases the network's memory footprint in exchange for faster test time. To tackle the memory size increase issue, we proposed the use of data clustering techniques to reduce the overall size of the neurons' memory (AGUIAR et al., 2014). We aim at reducing the overall size m of the neurons' memory, thus improving runtime performance and reducing memory footprint, while keeping an acceptable classification accuracy. Assuming that the memory of each neuron will have a large amount of redundant or irrelevant information, we should be able to accomplish relevant compression by carefully eliminating input-output pairs of the neurons' memory $\mathbf{L}$. This can be achieved by using a clustering algorithm to find the most relevant input-output pairs in the memory of each neuron as described next.

From a number of clustering techniques (XU; WUNSCH, 2005; JAIN; MURTY;

FLYNN, 1999; ROKACH, 2009; MACQUEEN, 1967), and considering the fact that the clustering procedure needs to be done many times for each neuron (see below), we decided to use k-Means (MACQUEEN, 1967) due to its simplicity and efficiency. The k-Means algorithm partitions a set $\mathbf{I}$ of $|\mathbf{I}|$ vectors into k clusters, $k < |\mathbf{I}|$, where the parameter k is set a priori. This iterative partitioning scheme minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. For our experiments, we used the Hamming distance to define the centroids of each cluster.

Figure 6 illustrates Fat-Fast VG-RAM neuron's memory reduction framework that works as follows. Initially, the original memory of each neuron, containing m lines, is sorted according to its output label t (i.e. similar images of same place). The result is a set of lines ordered according to each output type, e.g. the set of lines $\mathbf{L^1} = \{L^1_1, ..., L^1_{m_1}\}$ for outputs of type equal to label $t = 1$, $\mathbf{L^2} = \{L^2_1, ..., L^2_{m_2}\}$ for outputs of type equal to label $t = 2$, until $\mathbf{L^l} = \{L^l_1, ..., L^l_{m_l}\}$ for outputs of type equal to label $t = l$. Subsequently, we apply k-Means separately to each set of lines $\mathbf{L^1}, \mathbf{L^2}, \ldots, \mathbf{L^l}$, partitioning each one of them into k clusters. Each cluster has a centroid C that is equivalent to a memory line L. k-Means computes the set $\mathbf{C^1} = \{C^1_1, ..., C^1_{k_1}\}$ of centroids from the lines of set $\mathbf{L^1}$, where $k_1 < m_1$ and $m_1 = |L^1|$; the set $\mathbf{C^2} = \{C^2_1, ..., C^2_{k_2}\}$ of centroids from the lines of set $\mathbf{L^2}$, where $k_2 < m_2$; and so on until the set $\mathbf{C^l} = \{C^l_1, ..., C^l_{k_l}\}$, where $k_l < m_l$. The centroid sets $\mathbf{C^1}, \mathbf{C^2}, \ldots, \mathbf{C^l}$ become the new memory entries for the respective neuron (see Figure 6). This process is repeated for all neurons, i.e., the clustering process is performed $n \times l$ times. As a result, the number of entries in the neuron's memory is reduced according to the specified number of clusters k_j of each label type j. To maintain the original memory balance between entries for the different output types, we used different values of k for each label type, i.e., $k_j = m_j \times rl$, where $0 < rl < 1$ is the memory retention level - the lower the rl, the higher the memory reduction (rl expresses the amount of memory that remains after memory reduction). As a general rule, the parameter rl can be chosen as closest to one as possible, regarding the amount of available memory. We believe that this formulation, which determines the number of clusters k, is an important property of our approach since it allows changing the memory compression level depending on the application requirements. For instance, this approach could be used for resource-restricted applications by appropriately selecting the parameter rl that best fit the hardware requirements. For further discussion about tuning the parameter rl the interested reader can refer to (AGUIAR et al., 2014), where the parameter rl is studied under a more resource-restricted settings.

It is important to note that the proposed approach modifies the contents of the neurons' memory. It can even create a new entry that was not part of the learning process. We see this as an advantage of the framework since it is able to summarize the memory information according to the data. Please note that the input patterns $I_j = \{i_1, \ldots, i_p\}$ of all centroids $C_j = (I_j, t_j)$ must be binary, therefore, the centroids' elements need to be rounded to values either 0 or 1.

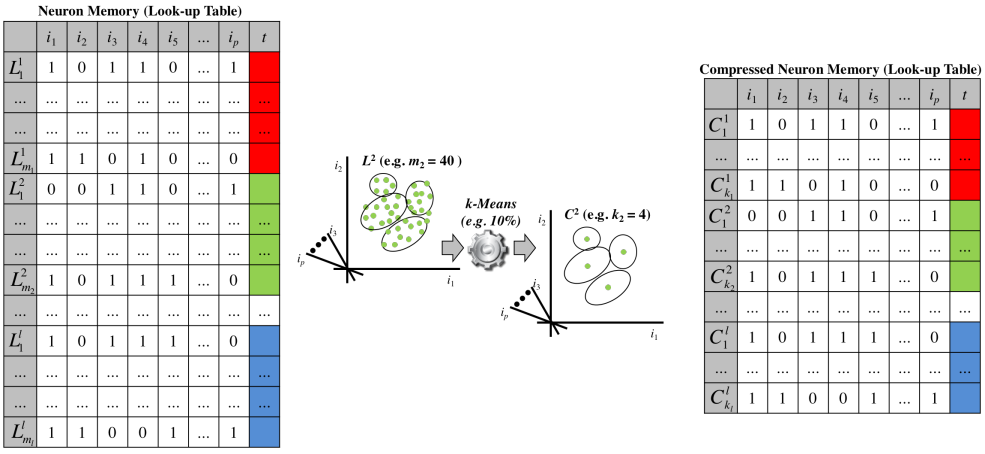
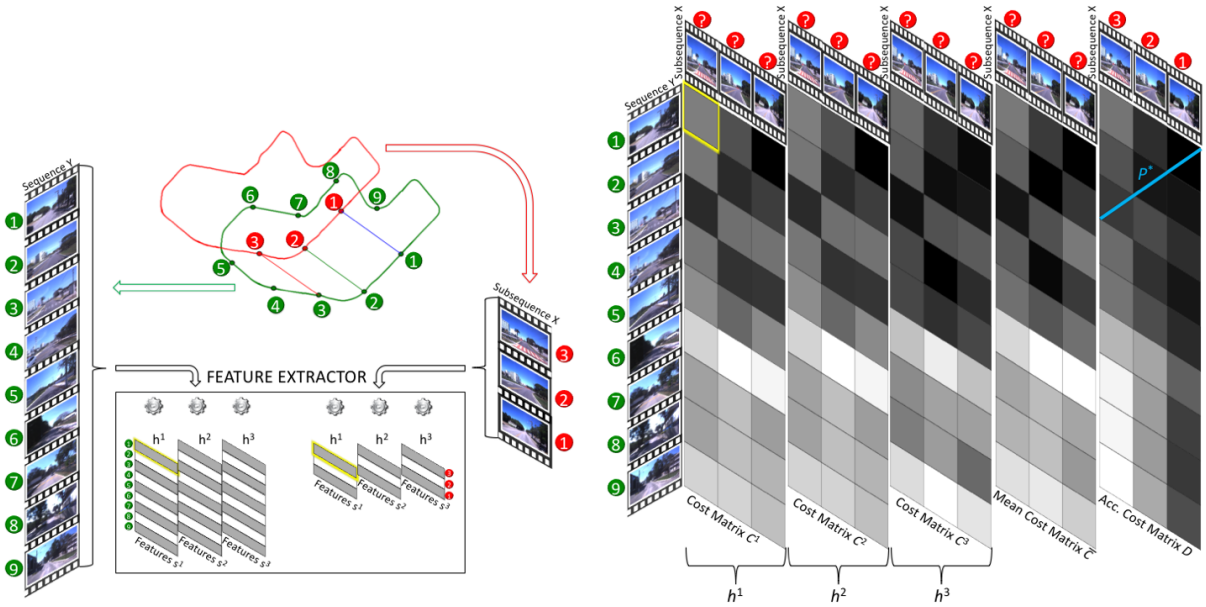


Figure 6 – Memory reduction framework. The original memory of each neuron, containing m lines, is treated as a set of points in a multi-dimensional space. First, the neuron's memory is grouped according to its output type t. Thereafter, each group is clustered separately using k-Means, where k is defined based on the trade-off between accuracy vs. runtime. At the end, the resulting centroids for each output type group become the new entries for the reduced memory, substituting the original memory information. Redrawn after (AGUIAR et al., 2014).

4.2 Sequential Image Classification for Place Recognition

This section presents a description of the proposed system designed to solve the Place Recognition problem in the space of appearance. The proposed system approach employs an Ensemble Learning method for visual place recognition, where sequences of place images are modelled as a binary time series problem. Firstly, the system is trained with a sequence of image-position pairs Y, collected using a robotic car driven along a given trajectory and illustrated by the green curve in Figure 7a. The image sequence Y is then transformed into a sequence of binary feature vectors for each classifier and stored along with their associated position. Secondly, image sequences are transformed into sequences of binary feature vectors by extracting binary features from each image as described in (LYRIO JÚNIOR et al., 2014) and illustrated in Figure 7a (see for example that there are distinct feature subsets s^1, s^2, s^3 for each of the classifiers h^1, h^2, h^3, respectively). Finally, to test the system, the robotic car is driven a second time along the same trajectory, illustrated by a red curve in Figure 7a, collecting subsequences of images X (see for example the subsequence of three frames in Figure 7a). As before, images are transformed to binary feature vectors and stored. Following, each classifier computes the Hamming distance between each feature vector of the sequence Y and the subsequence X to store them in a cost matrix C (see for instance the binary feature vectors $y1$ and $x3$ for the classifier h^1, as highlighted in yellow in Figure 7a, and the corresponding distance value stored in a cell of the cost matrix C^1, as highlighted in yellow in Figure 7b). The cost matrices values are averaged and then used in the Dynamic Time Warping (DTW) algorithm to

find an optimal path P^* between X and Y. The system's output is then a subsequence of the associated positions (the three red label circles located above the Accumulated Cost Matrix D in Figure 7b).



(a) (b)

Figure 7 – Illustration of subsequence of image-position pairs with length $N = 3$ globally localized using an ensemble of three kNN-DTW classifiers h^1, h^2, h^3. (a) Illustration of the pre-processing steps for sampling images and extracting features. On the left is the image sequence used for training. On the right is the image subsequence used for testing. In both cases, image sequences are transformed to binary time series. In the middle are two robot trajectories around the same region, but that occurred in two different moments in time: the training depicted in green and the test in red. The image-position pairs of the first trajectory, in green, are sampled at intervals of 300m creating a sequence of 9 samples that are used for training. The red image-position data points are sampled the same way but in a subsequence of 3 samples for each test. (b) Illustration of the corresponding image-position samples used for training and test as indicated by the green and red numbered circles. The question marks in some red circles indicate that the corresponding image-position pair is not globally localized, i.e. the correspondence between them and what was previously learned is unknown. This is only known after computing the local cost matrix $\mathbf{C}^1, \mathbf{C}^2, \mathbf{C}^3$ of the classifiers h^1, h^2, h^3 and the resulting mean cost matrix $\bar{\mathbf{C}}$ and the accumulated cost matrix $\mathbf{D}$. After that, the optimal warping path P^* for the subsequence is determined by the minimum cost path crossing the matrix $\mathbf{D}$ from right to left as indicated by the blue line in the figure. The cost values were normalized to the range $[0 : 255]$ for visualization purposes only. Redrawn after (FORECHI et al., 2016).

4.2.1 Dynamic Time Warping

The DTW (MÜLLER, 2007) metric is used for comparing two time-dependent sequences, $X := (x_1, x_2, ..., x_n, ..., x_N)$ and $Y := (y_1, y_2, ..., y_m, ..., x_M)$, of lengths N and $M \in \mathbb{N}$, respectively. The sequence items x_n and y_m are defined as vectors and belong to a binary feature space denoted by $\mathcal{F} = \{0, 1\}^S$, i.e. $x_n, y_m \in \mathcal{F}^S$, where $S \in \mathbb{N}$ is the size of the feature vectors. The binary features of vectors x_n and y_m are extracted from images of test and training sequences, respectively. The Hamming distance is used as the cost function $c(x_n, y_m)$ to compare two feature vectors $x_n, y_m \in \mathcal{F}^S$.

$$c : \mathcal{F}^S \times \mathcal{F}^S \to \mathbb{N}_+ \tag{4.1}$$

Hence, the more similar x_n and y_m are, the smaller $c(x_n, y_m)$ is. By computing the Hamming distance c for each pair of feature vectors x_n, y_m and arranging it in matrix form results in the local cost matrix $\mathbf{C} \in \mathbb{N}^{N \times M}$ defined by $C(n, m) := c(x_n, y_m)$.

4.2.2 Binary Feature Vectors

The binary feature vectors x_n and y_m extracted from the images of test and training sequences are computed in the same way as the bit vectors extracted by VG-RAM neurons (i.e. kNN classifiers) in (LYRIO JÚNIOR et al., 2014). Following their approach, VG-RAM neurons form a single Neural Layer that is connected to two input images used for place recognition: the first is simply a cropped version of the image captured by the robot's frontal camera and the second is a Gaussian filtered version of the first one. Each neuron is connected with these two inputs according to two different synaptic interconnection patterns: one holistic and the other feature-based (DE SOUZA et al., 2008), respectively. That enforces a distinct image view for each classifier as in a feature subset ensemble (ALY; ATIYA, 2006). These two synaptic patterns read integer-valued pixels and convert them to binary values forming the binary feature vectors x_n and y_m depending on whether it is a test or a training image sequence, respectively.

4.2.3 Ensemble of kNN-DTW

The proposed approach can be formally defined as a feature subset ensemble $\mathbf{H} = (h^1, h^2, ..., h^T)$ of size $T \in \mathbb{N}$ based on kNN classifiers using the DTW-Hamming distance $c(x_n, y_m)$ as the dissimilarity measure to compare binary feature vectors x_n and y_m. As it was said, each classifier gives a different view of all images in a dataset by selecting different feature subsets, instead of being trained with independent bootstrap samples as in bagging. The proposed Ensemble of kNN-DTW comprises a set $\mathbf{H}$ of classifiers h^t, in which every classifier holds a local cost matrix $\mathbf{C}^t$ that is averaged to form the average

cost matrix $\bar{\mathbf{C}}$.

$$\bar{\mathbf{C}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{C}^t. \tag{4.2}$$

From the average cost matrix $\bar{\mathbf{C}}$, the accumulated cost matrix $\mathbf{D}$ is computed by using dynamic programming, as it is usual in DTW (MÜLLER, 2007). Finally, a warping path that aligns sequences X and Y can be found using matrix $\mathbf{D}$. An illustration of these matrices and path can be found in Figure 7b.

A warping path is also a sequence $P = \{p_1, ..., p_L\}$ with the following matching pairs $p_l = (n_l, m_l) \in [1:N] \times [1:M]$ for $l \in [1:L]$ that satisfies three conditions: boundary, monotonicity and step size (MÜLLER, 2007). According to the definition, an optimal warping path P^* between sequences X and Y is a warping path P^* having minimal total cost among all possible warping paths. The step size condition adopted through all the experiments is $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$ for $l \in [1:L-1]$.

As described in (MÜLLER, 2007), the DTW requires some modifications to deal with subsequences. Since the goal here is to find an optimal warping path P^* for a test subsequence X given a training sequence Y, some modifications (e.g. relaxation of boundary condition for matching the start and end of the training sequence Y) were implemented accordingly to (MÜLLER, 2007). For an example, see the blue path P^* that connects subsequence X with length $N = 3$ and sequence Y with length $M = 9$ in Figure 7b. Following the notation, $P^* = \{(1,1),(2,2),(3,3)\}$.

Finally, the ensemble predicts a sequence with the corresponding positions of the indices m_l in the optimal warping path P^*, and the corresponding position of the index m_1 is used as output. Since the method is used with a sequence of streamed images (e.g. a robotic car along a trajectory), the previous predictions are used to stabilize the output by returning the most frequent index considering the N previous P_k^* predictions, where $k = 1$ is the current, i.e., the most recent sequence prediction for $k \in [1:N]$. The most frequent index is calculated considering the values predicted for m_k in the P_k^* previous sequences.

It is important to note that kNN-DTW neurons do not benefit from the proposed neuron's memory indexing and resulting runtime performance presented in Section 4.1. The reason is that DTW cost matrix is computed over all neuron's memory as just described in this section, thus it does not benefit from the indexing data structure designed to minimize comparisons along the entire neuron's memory.

4.3 CNN-based Relative Camera Pose Estimation for Visual Localization

In this section it is described the system proposed to solve the Relative Camera Pose Estimation problem by training a Siamese-like Convolutional Neural Network (S-CNN) to regress a 6D pose vector. Once trained, the S-CNN should infer the relative camera pose vector between two input images.

The traditional approach in Computer Vision (HARTLEY; ZISSERMAN, 2004) to compute the relative camera pose transformation consists of: i) extracting features from any two images using SURF (BAY et al., 2008), SIFT (LOWE, 2004), ORB (RUBLEE et al., 2011), DAISY (TOLA; LEPETIT; FUA, 2010) or corner features of checkerboard images when the objective is to calibrate a camera or a stereo rig. ii) estimating the Fundamental Matrix based on the coordinates of the extracted feature for non calibrated cameras, or in case of calibrated cameras estimating the Essential Matrix instead. iii) decomposing the Fundamental or Essential Matrix into tree matrices, Intrinsic Matrix K, Rotation Matrix R and Translation Matrix T. The rotation and translation matrices represent the transformation between the two images and can be easily converted into a 6D vector form to store the relative pose between cameras.

4.3.1 Architecture

The network architecture adopted here is similar to the siamese network proposed in (HANDA et al., 2016). Their siamese network takes inspiration from the VGG-16 network (SIMONYAN; ZISSERMAN, 2015) and uses 3×3 convolutions in all but the last two layers where 2×1 and 2×2 convolutions are used to compensate for the 320×240 resolution used as input as opposed to the 224×224 used in original VGG-16. Also, top fully connected layers were replaced by convolutional layers turning the siamese network into a Fully Convolutional Network (FCN) (LONG; SHELHAMER; DARRELL, 2015).

The siamese network proposed in (HANDA et al., 2016) takes in a pair of consecutive frames, I_t and I_{t+1}, captured at time instances t and $t+1$, respectively, in order to learn a 6-DoF visual odometry pose. The adopted siamese network, here, takes in a pair of image frames distant up to d meters from each other. More specifically, the siamese network branches are fed with pairs of key frames I_K and live frames I_L, in which the relative distance of live frames to key frames are up to d meters. The network's output is a 6-DoF pose vector, δ_{pred}, that transforms one image coordinate frame to the other. The first three components of δ_{pred} correspond to rotation and the last three components correspond to translation.

Similarly to the siamese network proposed in (HANDA et al., 2016), the architecture

adopted here fuses the two siamese network branches early to ensure that the relevant spatial information is not lost by the depth of the network. It is also avoided any pooling operations throughout the network, again to ensure that the spatial information is preserved. Despite the vast majority of convolutional architectures alternates convolution and max-pooling layers, max-pooling layers can be replaced for a larger-stride convolution kernel, without loss in accuracy on several image recognition benchmarks according to (SPRINGENBERG et al., 2014). Based on this finding and seeking to preserve spatial information through the network, there is no pooling layers in the network adopted here.

Figure 8 shows the siamese network adopted in this work. All convolutional layers, with the exception of the last three, are followed by a non-linearity, PReLUs (HE et al., 2015). The major differences to the siamese network proposed in (HANDA et al., 2016) are the dropout layers (SRIVASTAVA et al., 2014) added before the last two layers and a slight larger receptive field in earlier layers. Dropout was chosen for regularization since the original siamese network (HANDA et al., 2016) was trained on synthetic data only and does not generalize. The receptive field of early layers were made larger because it is desired to filter out high frequency components of real world data.

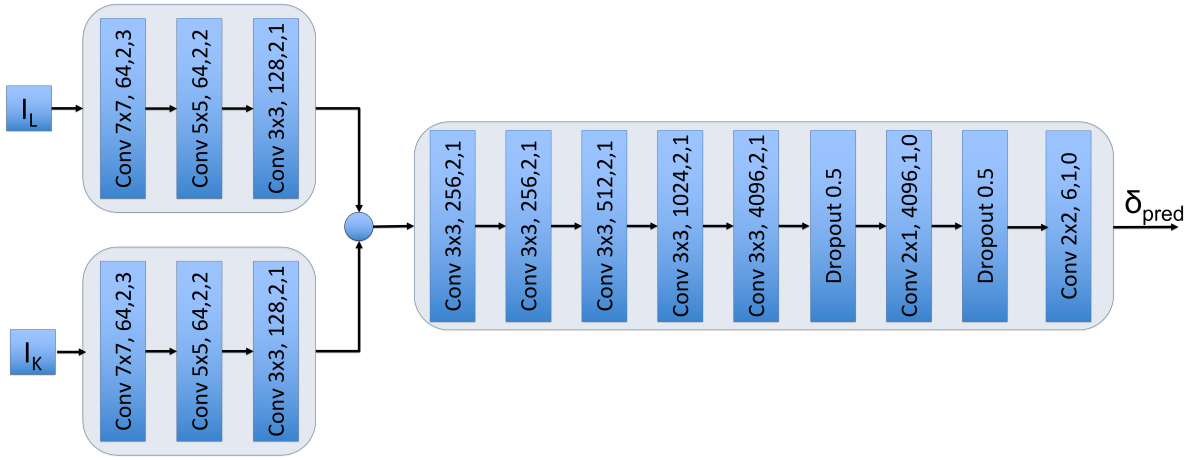


Figure 8 – Siamese CNN architecture for relative pose regression. The siamese network branches takes in pairs of a key frame I_K and a live frame I_L and outpus a 6-DoF relative pose δ_{pred} between those two frames. The Siamese network is a Fully Convolutional Network (FCN) built solely with convolutional layers followed by PReLU non-linearity. Each convolutional layer is composed by the following parameters ($kernel - size \times kernel - size$, output channels, stride, padding). Moreover, there are one dropout layer before each of the last two layers for regularization purposes.

4.3.2 Loss Function

Perhaps the most demanding aspect of learning camera poses is defining a loss function which is capable of learning both position and orientation. Kendall et al. noted that a model which is trained together to regress the position and orientation of the

camera is better than to separate models supervised on each task individually. They first proposed a method to combine position and orientation into a single loss function with a linear weighted sum (KENDALL; GRIMES; CIPOLLA, 2015). However, since position and orientation are represented in different units, a scaling factor was needed to balance the losses.

In (KENDALL; CIPOLLA, 2017), they recommend the reprojection error as a more robust loss function. The reprojection error is given by the difference between the 3D points in the world projected onto the 2D image plane using the ground truth and predicted camera pose. Instead, we chose the 3D projection error of the scene geometry since it is a representation that combines rotation and translation naturally in a single scalar loss, similar as to the reprojection error. However, the 3D projection error is expressed in the same metric space as the camera pose (HARTLEY; ZISSERMAN, 2004) and, thus provides more interpretable results than comparing, for instance, a loss function in pixels and an error in meters.

Basically, a 3D projection loss converts rotation and translation measurements into 3D point coordinates, as defined in Equation 4.3,

$$\mathcal{L} = \frac{1}{w \cdot h} \sum_{u=1}^{w} \sum_{v=1}^{h} \left\| T_{pred} \cdot p(x)^T - T_{gt} \cdot p(x)^T \right\|_2 , \tag{4.3}$$

where x is a homogenized 2D pixel coordinate in the live image, $p(x)$ is the 4×1 corresponding homogenized 3D point obtained by projecting the ray from that given pixel location (u, v) into the 3D world via inverse camera projection and live depth information, $d(u, v)$, at that pixel location. The norm $\|\cdot\|_2$ is the Euclidean norm applied to $w \cdot h$ points, where w and h are the depth map width and height, respectively. Moreover, it is worth to mention that the intrinsic camera parameters are not required to compute the 3D geometry in the loss function described by Equation 4.3. The reason is that the same projection $p(u, v)$ is applied to both prediction and ground truth measurements.

Therefore, this loss function naturally balances translation and rotation quantities depending on the scene and camera geometry. The key advantage of this loss is that it allows the model to vary the weighting between position and orientation, depending on the specific geometry in the training image. For example, training images with geometry which is far away would balance rotational and translational loss differently to images with geometry very close to the camera. If the scene is very far away, then rotation is more significant than translation and vice versa (KENDALL; CIPOLLA, 2017).

Projecting geometry through a neural network model is a differentiable operation that involves matrix multiplication. In (HANDA et al., 2016) is provided a 3D Spatial Transformer module which explicitly defines these transformations as layers with no learning parameters but allow computing backpropagation from the loss function to the

input layers. Figure 9 illustrates how the geometry layers fits the siamese network. There is a relative camera pose, either given by the siamese network or by the ground truth. There is also geometry layers to compute 3D world point from both relative camera poses.

On top left, it is shown the base and top branches of the siamese network that receives as input a pair of live frame I_L and key frame I_K and outputs a relative camera pose vector δ_{pred}. This predicted vector is then passed to the SE3 Layer that outputs a transformation matrix T_{pred}. Following, the 3D Grid Layer receives as input a live depth map D_L, the camera intrinsics matrix K and the T_{pred}. First, it projects the ray at every pixel location (u, v) into the 3D world by multiplying the inverse camera matrix K^{-1} by the homogenized 2D pixel coordinate (u, v) and then multiplying the result by the corresponding live depth $d(u, v)$. The resulting homogenized 3D point is finally transformed by the relative camera transformation encoded in the predicted matrix T_{pred}. The ground truth relative pose is also passed through the SE3 Layer and the resulting transformation matrix T_{gt} is applied to the output of the 3D Grid Layer produced before to get a view of the 3D point cloud according to the ground truth.

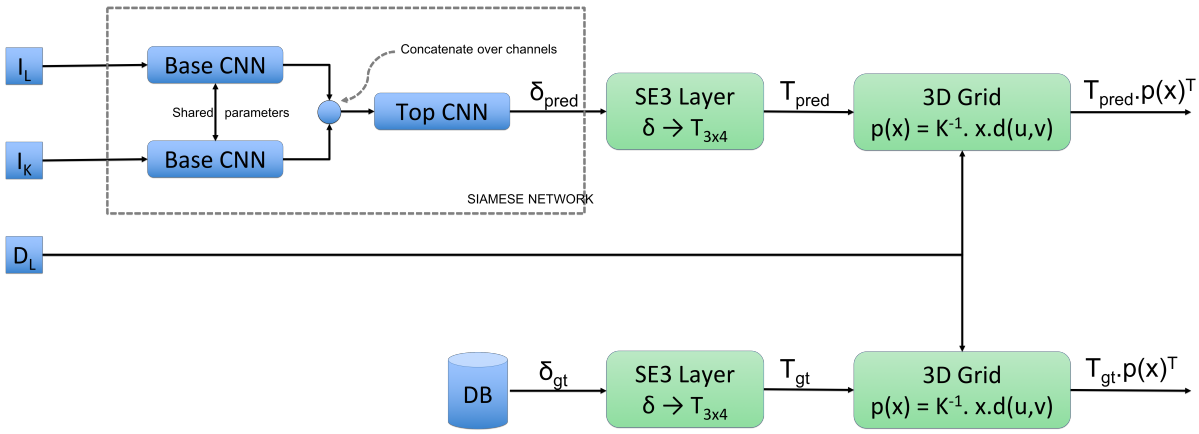


Figure 9 – Siamese and Geometry networks jointly applied for learning relative camera poses. On top left, it is shown the base and top branches of the siamese network. Its predicted vector δ_{pred} is passed to the SE3 Layer that outputs a transformation matrix T_{pred}. Then, the 3D Grid Layer receives as input a live depth map D_L, the camera intrinsics matrix K and the T_{pred}. First it projects the ray at every pixel location (u, v) into the 3D world and then multiplies the result by the corresponding live depth $d(u, v)$. The resulting homogenized 3D point is finally transformed by the predicted matrix T_{pred}. The ground truth relative pose is also passed through the SE3 Layer and the resulting transformation matrix T_{gt} is applied to the output of the 3D Grid Layer produced before to get a view of the 3D point cloud according to the ground truth.

4.4 The Hybrid Weightless-Weighted Neural Network Combined Approach for Global Localization

Lastly, it is presented the integration of the system described in Section 4.2 for solving the Place Recognition problem with the system described in Section 4.3 for solving the Visual Localization problem. Both systems together solve the problem of global localization, which consists of inferring the current live camera pose G_L given a live camera image I_L only. Note that the only input to the whole system is the live image, as depicted by the smallest square in Figure 10.

Figure 10 shows the workflow between the systems of place recognition (WNN approach) and visual localization (CNN approach) working together to provide the live global pose G_L given the live camera image I_L. Live image I_L is input for both WNN and CNN systems, while the WNN recollected image is passed to the CNN system as the key frame image I_K. Given the image pair, the CNN system outputs the relative camera pose δ_{pred} that is applied to the key global pose G_K to give the live global pose G_L.

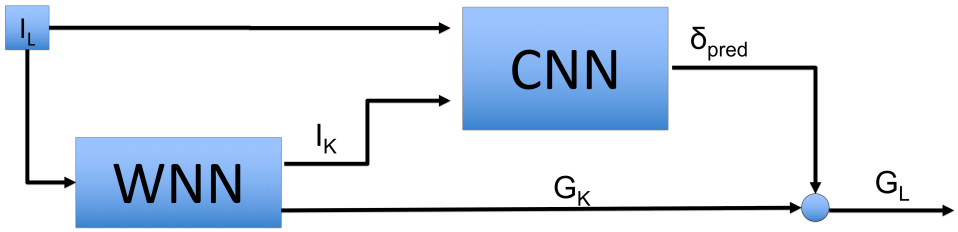


Figure 10 – The combined WNN-CNN systems workflow. The live image I_L is the only input to the whole system WNN-CNN that outputs the corresponding live global pose G_L. The WNN system provides the key frame image I_K to the CNN system, which together with the live image I_L outputs the relative camera pose δ_{pred}. That last applied to the key global pose G_K gives the live global pose G_L.

5 Experimental Methodology

This chapter presents the experimental setup used to evaluate the proposed system and also shows and discusses the outcomes of the experiments. It starts describing the autonomous vehicle platform used to acquire the datasets, follows presenting the datasets themselves, some parameter tuning, and describes the methodology used in the experiments. Then, it presents a comparison between the performance of standard VG-RAM, Fat-Fast VG-RAM and kNN-DTW neurons for the Place Recognition problem in terms of three metrics: classification accuracy, true positives, and false positives. Following, it presents a comparison between the performance of the global localization system and the GPS measurements collected with the autonomous vehicle platform.

5.1 Experimental Setup

The data used to evaluate the performance of the proposed system were collected using the Intelligent and Autonomous Robotic Automobile – IARA (Figure 11). IARA is an experimental robotic platform with several high-end sensors based on a Ford Escape Hybrid that is currently being developed at the Laboratório de Computação de Alto Desempenho – LCAD (acronym in Portuguese for High Performance Computing Laboratory) of the Universidade Federal do Espírito Santo (UFES), in Brazil. For more details about IARA specifications please refer to (LYRIO JÚNIOR et al., 2014; JÚNIOR et al., 2015).

Figure 11 – IARA: The autonomous vehicle of LCAD-UFES

5.1.1 Autonomous Vehicle Platform

The datasets used in this work were built using IARA's frontal Bumblebee XB3 stereo camera to capture VGA-sized images at 16fps, and IARA's localization module (VERONESE et al., 2016) to capture associated poses (6 Degrees of Freedom – 6-DoF).

IARA's localization module is based on a Monte Carlo particle filter (THRUN; BURGARD; FOX, 2005) and on a Grid Map (ELFES, 1989) built with cell grid resolution of 0.2m, as detailed in (MUTZ et al., 2016). Poses computed by the localization module have the precision of about the grid map resolution, as verified in (VERONESE et al., 2016).

So far, the term *position* has been employed with the meaning of a pose vector without the orientation component. Although those terms are used interchangeably here, the term *pose* is more appropriate to express the actual relation between the pair of image and pose given that the camera is fixed, oriented ahead and has a narrow 66° Horizontal Field Of View (HFOV).

5.2 Datasets

For the experiments, it was collected several laps data in different dates. For each lap, IARA was driven at speeds up to 60 km/h around UFES campus. An entire lap around the university campus has an extension of about 3.57 km. During laps, both image and pose data of IARA were acquired synchronously, amounting for more than 85 thousand pairs of image and pose. Table 1 summarizes all laps data in ten different sequences. Sequence 10 accounts for two laps, laps 06 and 08 are partial laps and all the others are full laps. The difference in days between sequence 1 and 12 covers more than six years. Such time difference resulted in a challenging testing scenario since it captured substantial changes in the campus environment. Such changes includes differences in traffic conditions, number of pedestrians, and alternative routes taken due to obstructions on the road. Also, there were substantial infrastructure modifications of buildings alongside the roads in between dataset recording. The complete set of sequences selected for the experiments is called UFES-LAPS and will be made public available in the following link upon publication.

5.2.1 Data Set 1 - Place Recognition with one lap data

The first experiment is set for comparison between the performance of the standard VG-RAM, the Fat-Fast VG-RAM and the kNN-DTW neurons in the context of Place Recognition and with regards different spacing between places represented by images and poses.

So, for the first experiment, two full laps were chosen from Table 1. The first lap data was recorded on 3rd of October 2012 (UFES-LAP-01) and comprises a sequence of 5,821 image-pose pairs, while the second lap data was recorded on 18th of April 2014 (UFES-LAP-02) and comprises a sequence of 4,414 image-pose pairs. The difference in size between same spacing datasets is mainly due to frame rate difference between recordings. The difference in days between the recording of the first and the second lap data is about

Table 1 – Ufes Dataset Sequences

Lap Sequence	Lap Date (mm-dd-yyyy)	Lap Sampling Spacing None	1m	5m
UFES-LAP-01	10-03-2012 02:16 PM	5,821	2,536	646
UFES-LAP-02	04-18-2014 12:35 PM	4,414	2,283	635
UFES-LAP-03	08-25-2016 03:31 PM	7,165	2,868	682
UFES-LAP-04	08-25-2016 03:47 PM	6,939	2,726	679
UFES-LAP-05	08-25-2016 04:17 PM	6,404	2,663	680
UFES-LAP-06	08-30-2016 05:40 PM	1,808	725	170
UFES-LAP-07	10-21-2016 04:15 PM	9,405	2,855	669
UFES-LAP-08	01-19-2017 07:23 PM	1,869	704	171
UFES-LAP-09	11-22-2017 05:20 PM	7,965	2,832	665
UFES-LAP-10	12-05-2017 09:35 AM	17,935	6,012	1,398
UFES-LAP-11	01-12-2018 04:30 PM	7,996	2,868	669
UFES-LAP-12	01-12-2018 04:40 PM	7,605	2,899	662
TOTAL		**85,326**	**31,971**	**1,672**

one year and a half. Such time difference resulted in a challenging test scenario since it captured substantial changes in the campus environment. Such changes include differences in traffic infrastructure and dynamic objects, such as cars and pedestrians. Also, there were substantial building infrastructure modifications alongside the roads between datasets' recording.

In average, the distances between UFES-LAP-01 and UFES-LAP-02 corresponding image-pose pairs is 1.20m ($\sigma = \pm 0.96$). To evaluate the effect of learning an increasing number of images, the UFES-LAP-01 dataset was sampled at five different intervals: 30m, 15m, 10m, 5m and 1m. After sampling the UFES-LAP-01, five datasets were created: a 1-meter spacing dataset with a total of 2,536 image-pose pairs, a 5-meter dataset with a total of 646 image-pose pairs, a 10-meter dataset with 333 image-pose pairs, a 15-meter dataset with 225 image-pose pairs, and a 30-meter dataset with 114 image-pose pairs. The same was done with the UFES-LAP-02 resulting in five datasets with a total of 2,283, 635, 331, 224 and 113 image-pose pairs.

To validate the proposed system for the place recognition problem, a set of experiments was run with the two datasets mentioned above. The UFES-LAP-01 datasets were used to train the system about the trajectory and the UFES-LAP-02 datasets were used to test the accuracy of the system by estimating poses along the previous experienced trajectory. Conversely, the UFES-LAP-02 datasets were used for training and the UFES-LAP-01 datasets for test. Five experiments were run using, for training, the 1-meter, 5-meter, 15-meter, and 30-meter spacing datasets sampled from UFES-LAP-01, and, for test, the 1-meter, 5-meter, 15-meter, and 30-meter spacing datasets sampled from UFES-LAP-02, respectively. In other words, the first experiment used the 1-meter spacing UFES-LAP-01

dataset for training and the 1-meter spacing UFES-LAP-02 dataset for test; the second experiment used the 5-meter spacing UFES-LAP-01 dataset for training and the 5-meter spacing UFES-LAP-02 dataset for test; and so on.

To define the ground-truth, the correspondences between the two lap data were established using the Euclidean distance between pairs of image-pose from each lap. The UFES-LAP-02 dataset was firstly sampled at five different intervals (1m, 5m, 10m, 15m and 30m), which created five test datasets (1-meter, 5-meter, 10-meter and 30-meter spacing UFES-LAP-02 datasets). Following, the UFES-LAP-01 dataset was matched with the five test datasets using the Euclidean distance as dissimilarity measure. Finally, the UFES-LAP-01 dataset was sampled according to the five different spacing, which created five training datasets (1-meter, 5-meter, 10-meter and 30-meter spacing UFES-LAP-01 datasets); the eventual non-matched data were taken off from the test datasets. The final sizes of paired training and test datasets are 2,057, 552, 272, 194 and 99 for the 1-meter, 5-meter, 10-meter, 15-meter and 30-meter spacing, respectively.

5.2.2 Data Set 2 - Place Recognition with multiple lap data

The second experiment is also set for comparison between the performance of the standard VG-RAM, the Fat-Fast VG-RAM and the kNN-DTW neurons in the context of Place Recognition but this time with regards an increasing number of images and poses representing same places along time.

For this experiment, all full laps were chosen from Table 1 but the first two datasets from the first experiment. Images from UFES-LAP-01 and UFES-LAP-02 laps are not considered for this experiment given those images were collected with a different camera from all others sequences listed in Table 1. The two partial laps UFES-LAP-06 and UFES-LAP-08 were disregards given their small size and small coverage of the lap. The resulting selected datasets picked for the second experiment are named UFES-WNN-LAPS and its complete list includes the following datasets: UFES-LAP-03, UFES-LAP-04, UFES-LAP-05, UFES-LAP-07, UFES-LAP-09, UFES-LAP-10, UFES-LAP-11 and UFES-LAP-12.

To validate the proposed system for the place recognition problem over time, a set of experiments was run with the UFES-WNN-LAPS dataset mentioned above. The UFES-WNN-LAPS was further split in training and test datasets. The training dataset is named UFES-WNN-LAPS-TRAIN and comprises all datasets from UFES-WNN-LAPS but the following two: UFES-LAP-07 and UFES-LAP-09. The UFES-2017112 dataset is renamed to UFES-WNN-LAPS-TEST to be used for test. This way, UFES-WNN-LAPS-TRAIN dataset were used to train the system about the trajectory along time and the UFES-WNN-LAPS-TEST dataset were used to test the accuracy of the system by estimating poses along the previously experienced trajectories. An experiment was run using, for training, the 5-meter spacing datasets sampled from UFES-WNN-LAPS-TRAIN, and, for

test, the 1-meter spacing datasets sampled from UFES-WNN-LAPS-TEST, respectively. In other words, the first experiment used the 5-meter spacing UFES-WNN-LAPS-TRAIN dataset for training, than it will named UFES-WNN-LAPS-TRAIN-5M. While the 1-meter spacing UFES-WNN-LAPS-TEST dataset were used for test and named UFES-WNN-LAPS-TEST-1M.

To define the ground-truth, the correspondences between every two lap data were established using the Euclidean distance between pairs of image-pose from each lap of training and test datasets with a third dataset, for pose registration purposes only. So, the UFES-LAP-07 dataset was reserved for pose registration only and none of its image were considered in this experiment. Firstly, it was sampled at a fixed 1m spacing interval to create UFES-WNN-LAPS-REG-1M. Following, the UFES-WNN-LAPS-TRAIN-1M dataset was matched with the registration dataset UFES-WNN-LAPS-REG using the Euclidean distance as proximity measure. Finally, the same procedures were applied to the UFES-WNN-LAPS-TEST-1M dataset. The final sizes of registered training and test datasets are 4,415 and 2,784, respectively.

5.2.3 Data Set 3 - Visual Localization

The last experiment is set to evaluate the precision of the system proposed in Section 4.3 for relative camera pose estimation when a ground truth key frame is available and when it is not. That is the general case for the global localization in which the key frame is given by one of the weightless-based systems presented in Chapter 4.

For this experiment, all full laps were chosen from Table 1 but the first two datasets from the first experiment. The resulting selected datasets picked for the third experiment are named UFES-CNN-LAPS and its complete list includes the following datasets: UFES-LAP-03, UFES-LAP-04, UFES-LAP-05, UFES-LAP-06, UFES-LAP-07, UFES-LAP-08, UFES-LAP-09, UFES-LAP-10, UFES-LAP-11 and UFES-LAP-12.

To validate the proposed system for the visual localization problem, a set of experiments was run with the UFES-CNN-LAPS dataset mentioned above. The UFES-CNN-LAPS was further split into training, validation and test datasets. The training dataset is named UFES-CNN-LAPS-TRAIN and comprises all datasets from UFES-CNN-LAPS, but the following three: UFES-LAP-06, UFES-LAP-08 and UFES-LAP-09. The UFES-LAP-09 dataset is renamed to UFES-CNN-LAPS-TEST to be used for test. The remaining two datasets, UFES-LAP-06 and UFES-LAP-08, compose the validation dataset, named UFES-CNN-LAPS-VALID. This way, UFES-CNN-LAPS-TRAIN dataset was used to train the system about relative camera poses along the trajectory, the UFES-CNN-LAPS-VALID is used during training to select the best parameters of the network. Lastly, the UFES-CNN-LAPS-TEST dataset was used to test the accuracy of the system by estimating relative camera poses with respect to previously experienced trajectories. An

experiment was run using, for training, a crossing of 5- and 1-meter spacing datasets sampled from UFES-CNN-LAPS-TRAIN and, for test, using a crossing of 5- and 1-meter spacing datasets sampled from UFES-CNN-LAPS-TEST, respectively. In other words, the first experiment used the 5-meter spacing UFES-CNN-LAPS-TRAIN dataset for the key frames, named UFES-CNN-LAPS-TRAIN-5M. While the 1-meter spacing UFES-CNN-LAPS-TRAIN dataset was used for selecting live frames and it was named UFES-CNN-LAPS-TRAIN-1M. The same procedure applies to validation and test dataset, resulting in the following datasets, respectively: UFES-CNN-LAPS-VALID-5M/1M and UFES-CNN-LAPS-TEST-5M/1M.

To define the ground-truth, the relative distances between every two lap data were established using the Euclidean distance between pairs of key frame and live frame along with their corresponding poses. The UFES-CNN-LAPS-TRAIN-1M dataset was matched with the UFES-CNN-LAPS-TRAIN-5M using the Euclidean distance as proximity measure to select the closest key frame from the 5m spacing dataset. The same procedure applies to the UFES-CNN-LAPS-TEST-1M/5M and UFES-CNN-LAPS-VALID-1M/5M datasets. The crossing data combinations for each dataset is as follows. For training data, it is crossed the data of every live frame in UFES-CNN-LAPS-TRAIN-1M with the key frames in UFES-CNN-LAPS-TRAIN-5M. For the validation data, live frames come from lap data in UFES-CNN-LAPS-VALID-1M dataset while the key frames can be in any lap data from UFES-CNN-LAPS-VALID-5M or UFES-CNN-LAPS-TRAIN-5M dataset. The same procedure applies to test dataset. Select the live frames from lap data in UFES-CNN-LAPS-TEST-1M and the key frames from lap data in UFES-CNN-LAPS-TEST-5M or UFES-CNN-LAPS-TRAIN-5M dataset. The complete combinations is found in Appendix A, listed in Table 5 for training, in Table 6 for validation and in Table 7 for test, respectively. The final sizes after crossing the lap data of training, validation and test datasets are, respectively: 98,404, 3,471 and 14,249.

All images were resized and cropped to 640×380 pixels in order to remove the car's hood. Depth data, needed for 3D point projection, were generate with the Slanted Plane Smoothing (SPS) stereo algorithm (YAMAGUCHI; MCALLESTER; URTASUN, 2014). SPS stereo algorithm is a dense stereo method employing a slanted plane model. It jointly estimates a super-pixel segmentation, boundary labels, and a dense depth estimate from a pair of stereo images.

5.3 Metrics

The performances of the standard VG-RAM is compared against that of Fat-Fast VG-RAM and kNN-DTW VG-RAM in terms of memory usage, classification precision, and classification test time. The memory usage is measured in bytes with the following

equations:

$$M_{vg-ram} = (p/b + sizeof(t)) \times m \times n \tag{5.1}$$

$$M_{fat-fast} = M_{vg-ram} + h \times (2^{p/h} + m) \times sizeof(m) \times n \tag{5.2}$$

$$M_{knn-dtw} = M_{vg-ram} + sizeof(m) \times m \times n \times N \tag{5.3}$$

In Equations 5.1 and 5.2, p is the number of synapses, $sizeof(.)$ computes the size in bytes of its argument, m is the number of training samples, n is the number of neurons, and h is the number of hash tables of H. In Equation 5.3, N is the length of test subsequence X. As Equation 5.1 shows, the memory consumption of the standard VG-RAM, M_{vg-ram}, is equal to the number of bytes necessary to store one input-output pair, $L = (I, t)$, where I requires $\frac{p}{b}$ bytes and t requires $sizeof(t)$ bytes (in our experiments, t is represented by an integer, i.e. $sizeof(t) = 4$ bytes; and b is the number of synapses represented by a byte), times the number of learned pairs, m, times the number of neurons, n. The memory consumption of the Fat-Fast VG-RAM, Equation 5.2, is equal to M_{vg-ram} plus the number of bytes necessary to store the multi-indexed hash table H, which can be inferred by inspecting Figure 5. The memory consumption of the kNN-DTW VG-RAM, Equation 5.3, is equal to M_{vg-ram} plus the number of bytes necessary to store the local cost matrices C, which can be inferred by inspecting Figure 7b.

The classification precision was measured by counting the number of hits in the test dataset and dividing it by the size of this dataset. It is considered a hit when an image-pose pair is within an established Maximum Allowed Error (MAE) in frames from the expected test image-pose pair. The time spent in classification task – i.e., finding the nearest pattern in memory – was estimated by dividing the time it took to classify all samples of test dataset by the number of samples of this dataset. For each classification system implemented, we computed classification time while using standard and Fat-Fast VG-RAM neurons. To compare the performances in terms of time of these two cases, we computed the speedup obtained with the use of Fat-Fast neurons dividing the classification time with standard by the classification time with Fat-Fast VG-RAM neurons.

5.4 Tuning Parameter Selection

Before comparing the performance of kNN-DTW and VG-RAM neurons in terms of classification accuracy, true positives, and false positives. In this section, it is first analyzed the impact of varying the length N of the subsequence X on the classification accuracy of kNN-DTW. The ensemble parameters were chosen accordingly to (LYRIO JÚNIOR et

al., 2014) as follows: an ensemble of classifiers with size $T = 96 \times 54 = 5,184$, and binary feature vectors with size $S = 128$.

Figure 12 shows the classification accuracy results using the 1-meter spacing UFES-LAP-01 dataset for training and the 1-meter spacing UFES-LAP-02 dataset for test. The vertical axis represents the classification accuracy (i.e., the percentage of estimated image-pose pairs that were within an established Maximum Allowed Error (MAE) in frames from the first image-pose pair of a given query subsequence). The horizontal axis represents the MAE in frames. Finally, the graph curves represent the results for different subsequence lengths.

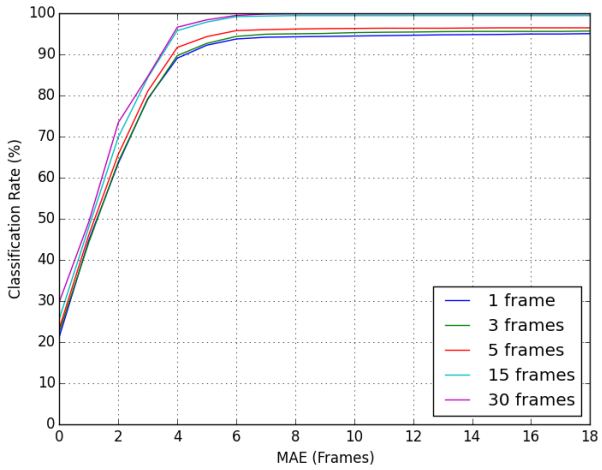


Figure 12 – kNN-DTW neuron classification accuracy for different subsequence lengths using UFES-LAP-01 dataset for training and the UFES-LAP-02 dataset for test.

As the graph of Figure 12 shows, the kNN-DTW neuron classification accuracy increases with MAE for all the subsequence lengths. Nevertheless, it is not a straightforward decision as selecting the highest length given that there is a trade-off between the subsequence length, latency and system runtime. But considering this trade-off and the previous accuracy results(LYRIO JÚNIOR et al., 2014) reaching a plateau around MAE equals 5, the choice for the subsequence length is $N = 5$.

5.5 Experimental Results

We evaluate in this section individually and in conjunction the performances of systems designed to solve the global localization problem. The global localization problem can be subdivided in two parts: place recognition and visual localization. The first part is addressed in this work with the help of Weightless Neural Networks (WNN) (FORECHI et al., 2016; FORECHI et al., 2015; AGUIAR et al., 2014) and the second part is based on Convolutional Neural Networks (CNN) (LECUN et al., 1998). The first approach was presented in Sections 4.1, 4.1.1, 4.2 and the second one in Section 4.3. We assess the WNN

Table 2 – Memory consumption of Fat-Fast, kNN-DTW and standard VG-RAM neurons in the Place Recognition task

Neuron Type	Synapses	int (B)	Samples	Neurons	Hash Tables	Size (MB)
VG-RAM	128	4	4,415	5,184	-	437
Fat-Fast	128	4	4,415	5,184	8	11,503
kNN-DTW	128	4	4,415	5,184	-	873

approaches capabilities with regards spatio-temporal similarity measure as presented in Section 4.2, memory retention levels presented in Section 4.1.1, and also with regards the multi-index hashing techniques as it was presented in Section 4.1. We assess the CNN approach as a Visual Localization regressor for relative camera pose estimation.

5.5.1 All Neurons Memory Consumption

The additional memory consumption imposed by the Fat-Fast and kNN-DTW neurons can be visualized in Table 2 (see Equations 5.1, 5.2 and 5.3). In Table 2, the first column describes the neuron type under analysis, the intermediate columns are the parameters of equations, and the last column is the memory consumption of neuron types. To take advantage of the ability of current machines to handle 8-bit and 16-bit addresses, in our implementations, the sub-vector V_k size (p/h) was restricted to these two sizes. As Table 2 shows, the use of Fat-Fast neurons increases memory consumption by more than 26 times for the place recognition task (11,503/437 $\approx$ 26x); while the use of kNN-DTW neurons doubles memory requirement (873/437 $\approx$ 2x).

5.5.2 kNN-DTW Neuron Classification Accuracy

This subsection compares the performance of the kNN-DTW neuron with that of the VG-RAM neuron by means of the relationship between the amount of frames learned by the classifiers and their classification accuracy. The system classification accuracy is measured in terms of how close the system's estimated image-pose pair is to the first image of a given query subsequence. The kNN-DTW uses subsequences of length $N = 5$, but recall that it outputs the corresponding position to the index m_1 only.

Figure 13 shows the classification accuracy results obtained using UFES-LAP-01 datasets for training and UFES-LAP-02 datasets for test. The vertical axis represents the percentage of estimated image-pose pairs that were within an established Maximum Allowed Error (MAE) in frames from the first image-pose pair of a given query subsequence. The MAE is equal to the amount of image-pose pairs that one has to go forward or backwards in the test dataset to find the corresponding query image. The horizontal axis represents the MAE in frames. Finally, the curves represent the results for different spacing datasets.

As the graph of Figure 13a shows, the kNN-DTW neuron classification accuracy

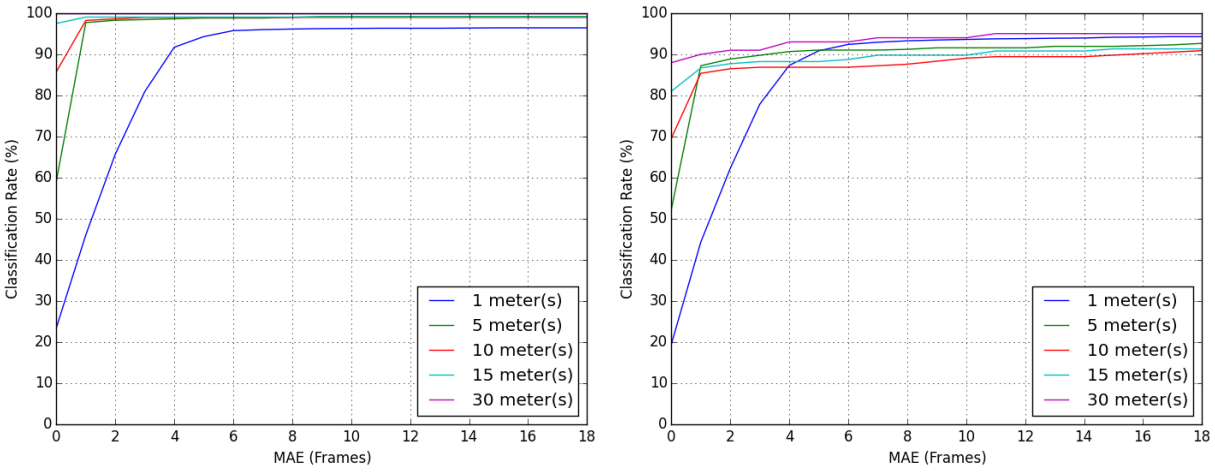


Figure 13 – Classification accuracy of kNN-DTW (a) and VG-RAM (b) neurons for different Maximum Allowed Error (MAE) in frames using UFES-LAP-01 dataset for training and the UFES-LAP-02 dataset for test.

increases with MAE for all the spacing datasets and reaches a plateau at about 5 frames for all training datasets. However, for the 1-meter spacing training dataset, the kNN-DTW neuron classification uncertainty is large in the beginning of the curve due to the similarity between images in the near by image-pose pairs. If one does not accept any system error (MAE equals zero), the accuracy is only about 23%. But, if one accepts an error of up to 5 frames (MAE equals 5), the accuracy increases to about 94%. On the other hand, when using a dataset with a larger spacing between image-pose pairs for training, the system accuracy increases more sharply. For example, when the system is trained with the 5-meter spacing dataset, with MAE equals 1m, the classification rate is about 98%. Although the system might show better accuracy when trained with large-spaced datasets, the positioning error of the system increases. This happens because one frame of error for the 1-meter training dataset represents a much smaller error in meters than one frame of error with large-spaced training dataset.

When comparing the graphs of Figures 13a and 13b, it can be observed that, for all training datasets, the kNN-DTW neuron outperforms VG-RAM neuron in terms of classification accuracy. Please note that the kNN-DTW neuron curves (Figure 13a) present accuracy consistently above the VG-RAM neuron curves (Figure 13b), from start to end. For example, kNN-DTW neuron accuracy reaches 100% for the 30-meter spacing training dataset whilst the previous one stops at about 94%. Even when considering lower tolerance in the number-of-frames, the kNN-DTW neuron is better. Taking apart the 1-meter curve, all the others reach a plateau at about 98% with MAE equals 1m, whereas the VG-RAM neuron achieves up to 90%.

Figure 14 shows the classification accuracy results obtained using UFES-LAP-02

datasets for training and UFES-LAP-01 datasets for test. The vertical axis represents the percentage of estimated image-pose pairs that were within an established Maximum Allowed Error (MAE) in frames from the first image-pose pair of a given query subsequence. The MAE is equal to the amount of image-pose pairs that one has to go forward or backwards in the test dataset to find the corresponding query image. The horizontal axis represents the MAE in frames. Finally, the curves represent the results for different spacing datasets.

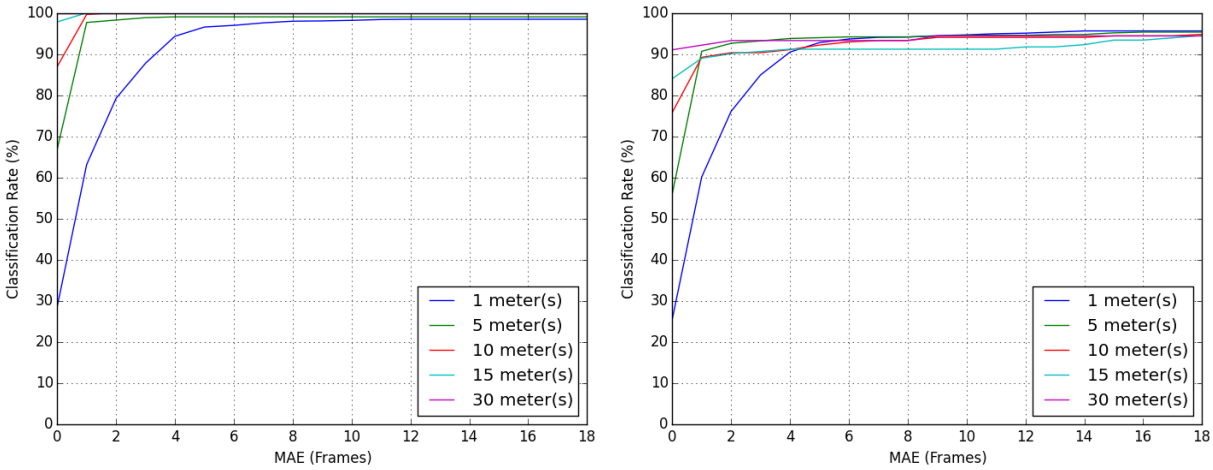


Figure 14 – Classification accuracy of kNN-DTW (a) and VG-RAM (b) neurons for different Maximum Allowed Error (MAE) in frames using UFES-LAP-02 dataset for training and the UFES-LAP-01 dataset for test.

As the graph of Figure 14a shows, the kNN-DTW neuron classification accuracy increases with MAE for all the spacing datasets and reaches a plateau at about 5 frames for all training datasets. However, for the 1-meter spacing training dataset, the kNN-DTW neuron classification uncertainty is large in the beginning of the curve due to the similarity between images in the near-by image-pose pairs. If one does not accept any system error (MAE equals zero), the accuracy is only about 28%. But, if one accepts an error of up to 5 frames (MAE equals 5), the accuracy increases to about 96%. On the other hand, when using a dataset with a larger spacing between image-pose pairs for training, the system accuracy increases more sharply. For example, when the system is trained with the 5-meter spacing dataset, with MAE equals 1m, the classification rate is about 98%. Although the system might show better accuracy when trained with large-spaced datasets, the positioning error of the system increases. This happens because one frame of error for the 1-meter training dataset represents a much smaller error in meters than one frame of error with large-spaced training dataset.

When comparing the graphs of Figures 14a and 14b, it can be observed that, for all training datasets, the kNN-DTW neuron outperforms the VG-RAM neuron in terms of classification accuracy. Please note that the kNN-DTW neuron curves (Figure 14a) present

accuracy consistently above the VG-RAM neuron curves (Figure 14b), from start to end. For example, the kNN-DTW neuron accuracy reaches 100% for the 30-meter spacing training dataset whilst the previous one stops at about 94%. Even considering lower MAE, the kNN-DTW neuron is better. Taking apart the 1-meter curve, all the others reach a plateau at least at about 98% with MAE equals 1m, whereas the VG-RAM neuron achieves up to 93%.

5.5.3 kNN-DTW Neuron True and False Positives

After analyzing classification accuracy, in this subsection, the performance of kNN-DTW neuron is compared with that of VG-RAM neuron by means of true and false positives for the 1-meter training dataset and 1-meter test dataset, with MAE equals 5. This allows us to identify patterns of positioning errors in the datasets. As before, the proposed system is given subsequences of length $N = 5$.

Figure 15 shows the true and false positives results using the 1-meter spacing UFES-LAP-01 dataset for training and the 1-meter spacing UFES-LAP-02 dataset for test. The vertical and horizontal axis represent global coordinates in meters. Blue dots represent estimated positions that were inside the tolerance of 10m from the correct image-pose pair, i.e., true positives. Red circles represent the estimated image-pose pairs that were outside the tolerance of 10m, i.e., false positives. Red circles are connected to their corresponding correct image-pose pairs (green crosses) through a line.

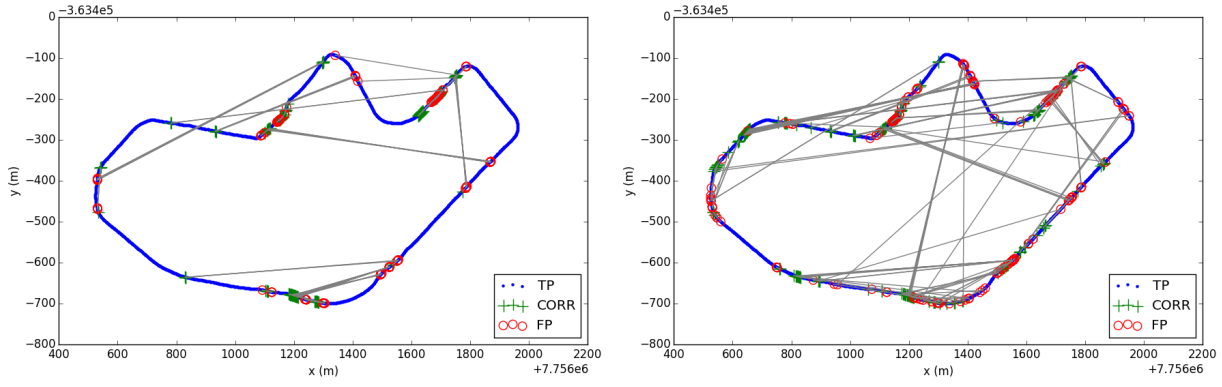


Figure 15 – True and false positives for kNN-DTW (a) and VG-RAM (b) neurons using 1-meter spacing UFES-LAP-01 dataset for training and 1-meter spacing UFES-LAP-02 dataset for test. Blue dots represent true positives (TP); lines connect false positives (FP) represented by red circles and their respective ground-truth correspondences represented by green crosses (CORR)

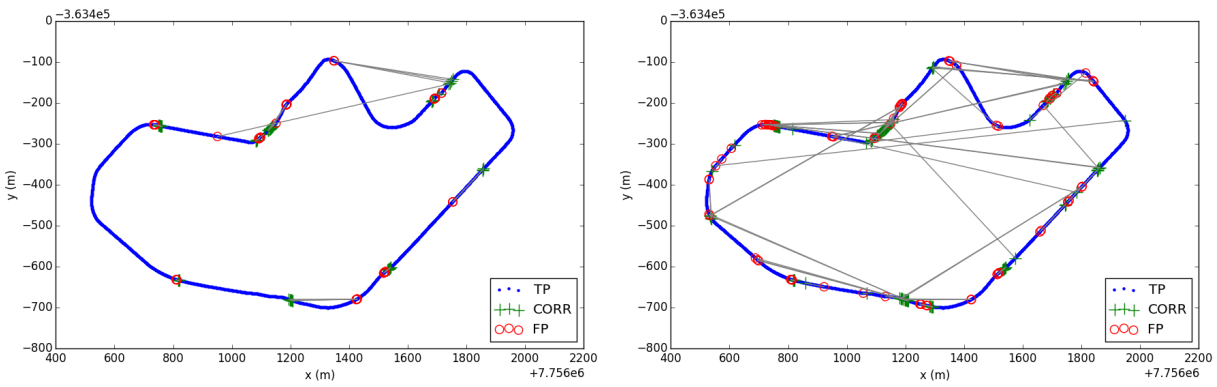
When comparing the graphs of Figures 15a and 15b, it can be observed that kNN-DTW neuron outperforms the VG-RAM neuron in terms of false positives. In the graph of Figure 15b, which shows the results for the VG-RAM neuron, it can be seen

large red zones spread all over the graph. These results are much different from those for the kNN-DTW neuron, shown in the graph of Figure 15a, where it can be observed a much smaller number of red spots and lines crossing the trajectory. There are two major red spots in Figure 15a: one in the top middle part of the trajectory (where there is a tree-covered lane just after the bend) and the other in the top right part (where the trajectory starts).

Figure 16 shows the true and false positives results using the 1-meter spacing UFES-LAP-02 dataset for training and the 1-meter spacing UFES-LAP-01 dataset for test. The vertical and horizontal axis represent global coordinates in meters. Blue dots represent estimated positions that were inside the tolerance of 10m from the correct image-pose pair, i.e., true positives. Red circles represent the estimated image-pose pairs that were outside the tolerance of 10m, i.e., false positives. Red circles are connected to their corresponding correct image-pose pairs (green crosses) through a line.

When comparing the graphs of Figures 16a and 16b, it can be observed that kNN-DTW neuron outperforms the VG-RAM neuron in terms of false positives. In the graph of Figure 16b, which shows the results for the VG-RAM neuron, it can be seen large red zones spread all over the graph. These results are much different from those for the kNN-DTW neuron, shown in the graph of Figure 16a, where it can be observed a much smaller number of red spots and lines crossing the trajectory.

Overall, to the extent of these experiments, the kNN-DTW neuron - that combines the DTW-Hamming distance metric to compare binary features from sequences of images - proved be a better approach than the VG-RAM neuron - that uses the Hamming distance to compare binary features from single images only.



(a) (b)

Figure 16 – True and false positives for kNN-DTW (a) and VG-RAM (b) neurons using 1-meter spacing UFES-LAP-02 dataset for training and 1-meter spacing UFES-LAP-01 dataset for test. Blue dots represent true positives (TP); lines connect false positives (FP) represented by red circles and their respective ground-truth correspondences represented by green crosses (CORR)

A demo video, recorded at 30 frames per second and fast-forwarded at 20x, shows the kNN-DTW neuron performance on a complete lap around the university campus (see video at <https://github.com/LCAD-UFES/SABGL>). In this video, the 1-meter spacing UFES-LAP-01 dataset was used for training whereas the 1-meter spacing UFES-LAP-02 dataset was used for test.

5.5.4 Performance of kNN-DTW Neuron without Memory Reduction

This subsection further compares kNN-DTW and VG-RAM neurons for solving the place recognition problem (FORECHI et al., 2016; LYRIO JÚNIOR et al., 2014). So far, both approaches were analyzed given one lap for training and another one for test. But looking forward in the context of global localization it is expected that a self-driving car drives itself the same routes more than once, and thus being subject to environmental changes along time. That poses the life-long global localization problem, where such changes appear as different views of the same scenes along time. Which raises concerns such as precision and runtime performance, but also the amount of memory retained as the data increases. This is an interesting scenario to validate the memory compression approach described in Section 4.1.1 and the Fat-Fast VG-RAM neuron presented in Section 4.1. In the next paragraphs, we detail the experiments and the results obtained with the proposed compression framework for different datasets other than previously analyzed in (AGUIAR et al., 2014; FORECHI et al., 2015).

The following modifications were necessary for kNN-DTW neuron in order to compensate for the increase in the number of samples per label. One is to add the first two extra tuples to the original step size condition as shown in $p_{l+1} - p_l \in \{(1,3),(1,2),(1,0),(0,1),(1,1)\}$ for $l \in [1 : L-1]$. Recall that the first and second elements of tuples accounts for comparison steps along test sequences of size N and neuron's memory sequence of size M, respectively. Another improvement is the adoption of local weights (MÜLLER, 2007) in the form of a weight vector $(w_d, w_h, w_v) \in \mathbb{R}^3$ applied to the computation of the accumulated cost matrix $\mathbf{D}$ through the recurrence formula 5.4 of dynamic programming.

$$D(n,m) = min \begin{cases} D(n-1,m-1) + w_d.c(x_n, y_m) \\ D(n-1,m) + w_h.c(x_n, y_m) \\ D(n,m-1) + w_v.c(x_n, y_m) \end{cases} \tag{5.4}$$

The first modification favors comparisons in a wider range of neuron's memory and the second is applied such that it decreases the local cost in the horizontal direction and increases in the diagonal and vertical direction according to the following weight vector $(2.0, 0.5, 1.0)$. The directions are related to the accumulated cost matrix $\mathbf{D}$ formation. The need for these modifications indicates that kNN-DTW neuron parameters are sensitive to the number of samples that represents a class.

Figure 17 shows the classification accuracy results obtained using UFES-WNN-LAPS-TRAIN-5M datasets for training and UFES-WNN-LAPS-TEST-1M dataset for test. The vertical axis represents the percentage of estimated image-pose pairs that were within an established Maximum Allowed Error (MAE) in frames from the first image-pose pair of a given query subsequence. The MAE is equal to the amount of image-pose pairs that one has to go forward or backwards in the test dataset to find the corresponding query image. The horizontal axis represents the MAE in frames. Finally, the curves represent the results for different spacing datasets.

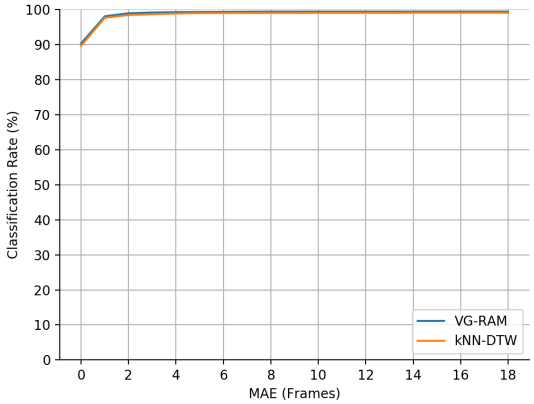


Figure 17 – Classification accuracy of VG-RAM and kNN-DTW neurons for different Maximum Allowed Error (MAE) in frames using UFES-WNN-LAPS-TRAIN-5M dataset for training and the UFES-WNN-LAPS-TEST-1M dataset for test and no memory compression.

As the graph of Figure 17 shows, kNN-DTW and VG-RAM neurons classification accuracy increases with MAE when tested with the UFES-WNN-LAPS-TEST-1M dataset and trained with the UFES-WNN-LAPS-TRAIN-5M dataset, reaching a plateau at about 2 frames. If one does not accept any system error (MAE equals zero), the accuracy of both neurons is about 90%. But, if one accepts an error of up to 1 frame (MAE equals 1m), the accuracy of both neurons increases to about 98%. Although the system shows a high accuracy when trained with a large-spaced dataset, the positioning error of the system increases. This happens because one frame of error for the 5-meter training dataset represents a bigger error in meters than one frame of error if it were trained with smaller-spaced training dataset at the expend of a lower accuracy, as shown in (FORECHI et al., 2016). Conversely, this happens less than 10% of the time for MAE equals zero and less then 2% of the time for MAE equals 1m.

Moreover, using the memory reduction technique, standard VG-RAM and kNN-DTW neurons' memory consumption reduces linearly with rl (see Equation 5.1 and Equation 5.3). This is illustrated by Figure 18, where it is shown the memory footprint of the standard VG-RAM and kNN-DTW neurons for the considered retention levels.

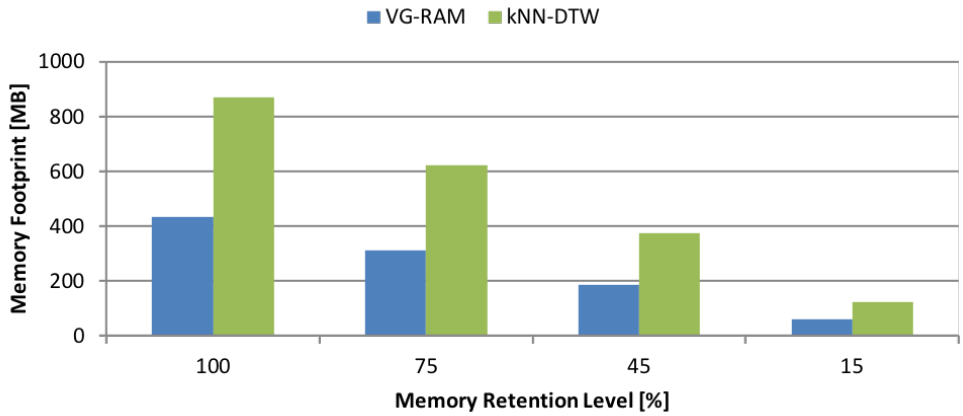


Figure 18 – Memory footprint of the Place Recognition System implemented with standard VG-RAM neuron, and kNN-DTW neuron with subsequence $N = 5$.

The first experiment has set the benchmark for kNN-DTW and VG-RAM neurons with regards classification accuracy, given that all laps from UFES-WNN-LAPS-TRAIN-5M were used for training without memory size reduction.

5.5.5 Performance of Fat-Fast Neuron without Memory Reduction

In this section, it is evaluated the classification precision and speed up of system implemented with Fat-Fast neurons, against those of system implemented with standard VG-RAM neurons. The impact on speed up and classification precision performance is analyzed as the memory size reduces. For that, we are particularly interested in comparing the performances of VG-RAM neurons and Fat-Fast VG-RAM neurons because the latter imposes a high demand on memory allocation.

The bars in the graph of Figure 19 show the precision of standard VG-RAM and Fat-Fast neurons on the classification of the UFES-WNN-LAPS-TEST-1M test dataset using UFES-WNN-LAPS-TRAIN-5M training dataset and p/h = 16 bits. There is one bar for standard VG-RAM neuron, and several bars for Fat-Fast neuron, each for a different value of the parameter r (see Section 4.1). The precision value is denoted in the left vertical axis. As the bars in the graph of Figure 19 show, Fat-Fast neurons causes only a small reduction in the classification precision - less than 1% (in about 90%) - for r varying from 30 to 1. With $r = 1$, the loss in precision is equal to 0.82%.

The curve in the graph of Figure 19 shows the speedup obtained with Fat-Fast neurons; the speedup value is denoted in the right vertical axis. As the curve in Figure 19 shows an speedup of at least 7x for large values of r. Moreover, for $r = 5$ the speed is about 9.5 and for $r = 1$, is about 10.5. Those speedups are modest, one order of magnitude lower compared to tasks with bigger datasets such as for Traffic Sign Recognition (FORECHI et al., 2015).

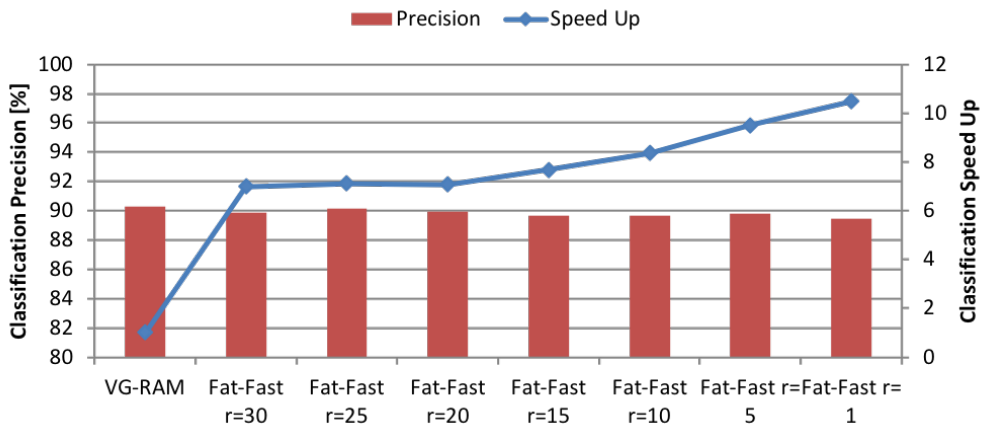


Figure 19 – Bars: Classification precision for implementations with standard VG-RAM neuron, and Fat-Fast neuron with different values of r for p/h = 16 bits. Curve: Speedup of the Fat-Fast implementation for different values of r.

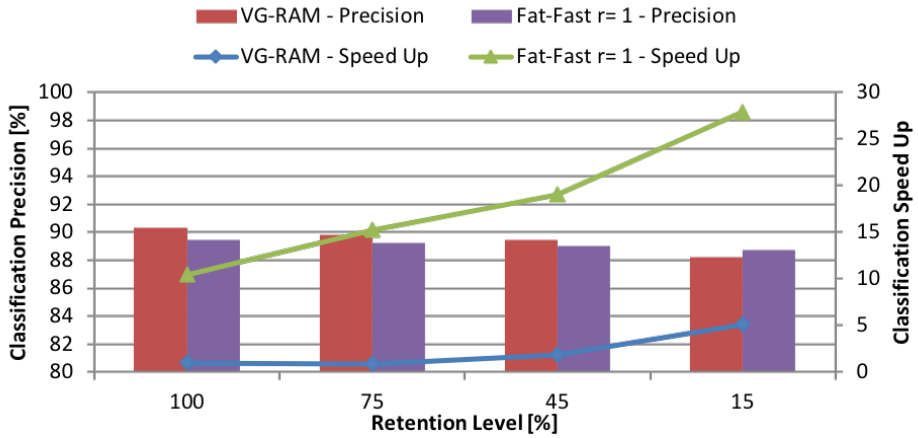
5.5.6 Performance of Fat-Fast Neuron with Memory Reduction

In the next experiments, it is evaluated the combined impact on Fat-Fast neurons performance using the memory reduction technique presented in Section 4.1.1. It is measured the Fat-Fast precision and speedup for different memory retention levels, for sub-vector V_k size (p/h) equal to 16 bits, and for r equal to 30 and 1. It is also examined the impact of memory reduction on standard VG-RAM neuron.

The bars of the graph of Figure 20 show the precision of the standard and Fat-Fast neuron implementations of the place recognition system for different memory retention levels, rl. It is examined four memory retention levels: $rl = 15\%$, $rl = 45\%$, $rl = 75\%$ and, when no memory reduction has been made, $rl = 100\%$. The Fat-Fast bars are for a system with $p/h = 16$ bits, and $r = 30$. As can be seen in Figure 20, the impact on precision of a retention of 75% of the original memory size (or training set size, m; see Section 4.1.1) is very small for both, standard and Fat-Fast neurons; a retention of 15% of the original memory has still a small impact on precision - a decrease of about 1%. Smaller retention levels produce higher impact on precision.

The curves of the graph of Figure 20 show the speedups of the place recognition system for different memory retention levels, rl. As these curves show, Fat-Fast presents better performance for all rl. In addition, using the memory reduction technique, standard VG-RAM memory consumption reduces linearly with rl (see Equation 5.1), while the same does not occur with Fat-Fast neuron, since part of the size of the hash tables, H, does not depend on the size of the training set, m (see Equation 5.2, term $2^{p/h}$). This is illustrated by Figure 21, where it is shown the memory footprint of the standard VG-RAM and Fat-Fast neurons for the considered retention levels.

(a)



(b)

Figure 20 – Impact of different memory retention levels rl on the performance of the Place Recognition System implemented with standard VG-RAM neuron, and Fat-Fast neuron with $p/h = 16$ bits and for r equal to 1 and 30.

5.5.7 Performance of the combined approach for Global Localization

In this section, it is first described the training procedure for the Convolutional Neural Network (CNN) and later compared the performance of CNN alone and CNN+WNN hybrid approach with GPS, where standard VG-RAM and GPS are more accurate. For the standard VG-RAM, it occurs 90.3% of the time if disregarding all false positive and assuming a Maximum Allowed Error (MAE) equals to zero. For the GPS, it is where the signal quality is stable, which occurs 89.65% of the time. For this experiment, a signal is considered stable when GPS quality indicator is greater than 0 [1].

The Convolutional Neural Network (CNN) was trained on UFES-CNN-LAPS-TRAIN-5M/1M dataset using images from left camera of Bumblebee XB3 and the depth image computed with SPS stereo (YAMAGUCHI; MCALLESTER; URTASUN, 2014), being both image and depth cropped to 320 × 240. The training data amounts for 98,404

[1] $http://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_GGA.html$

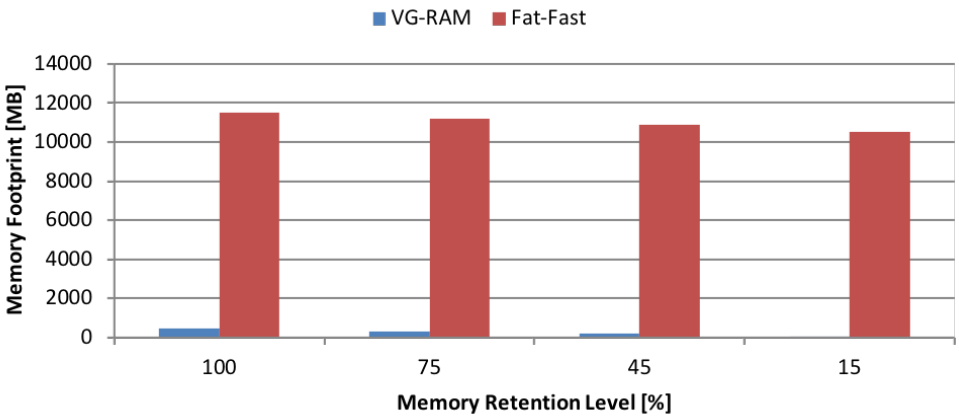


Figure 21 – Memory footprint of the Place Recognition System implemented with standard VG-RAM neuron, and Fat-Fast neuron with $p/h = 16$ bits and $r = 30$.

samples of key and live frames, both from left camera. No data augmentation was used.

The network was trained with Adam optimizer (KINGMA; BA, 2014) using mini-batches of size 24. Adam hyper-parameters β_1 and β_2 were set to 0.9 and 0.999, respectively. The learning rate is initially set to 0.0001 and decreased by a factor of 2 at each epoch, as shown in Figure 22.

The graph of Figure 22 shows learning rate decaying schedule for 25,000 training iterations. The vertical axis represents learning rates values and the horizontal axis represents training iterations with the aforementioned batch size. At all, the network was trained for 7 epochs, with 4,101 iterations per epoch.

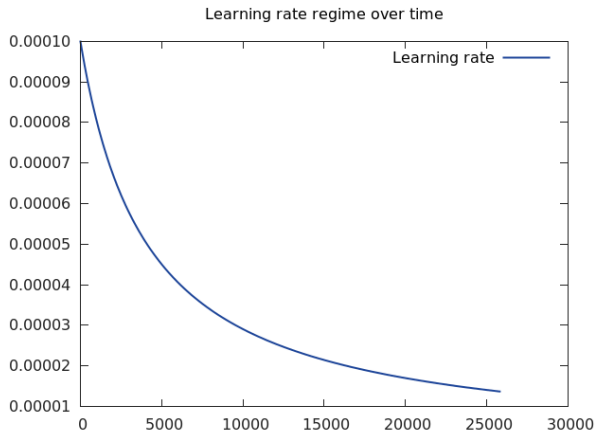


Figure 22 – Learning rate decay schedule adopted for training the convolutional neural network.

To prevent network from overfitting, it is employed Dropout layers (SRIVASTAVA et al., 2014) and Early Stopping (GOODFELLOW; BENGIO; COURVILLE, 2016). There are two Dropout layers in the Convolutional Network Architecture presented in Section 4.3, one after the second-to-the-last CNN layer and other after the third-to-the-last CNN layer.

Both have 50% units randomly dropped at each training iteration.

The curves of the graph of Figure 23 show the network training evolution using UFES-CNN-LAPS-TRAIN-5M/1M dataset for training and UFES-CNN-LAPS-VALID-5M/1M dataset for validation. The vertical axis represents the error in meters and the horizontal axis represents the number of iterations. The curve in indigo presents the loss function error as in Equation 4.3, while the curve in green presents the positioning error measured with the Euclidean distance on the training data. The curve in red is also measured with the Euclidean distance, but represents the positioning error of validation data.

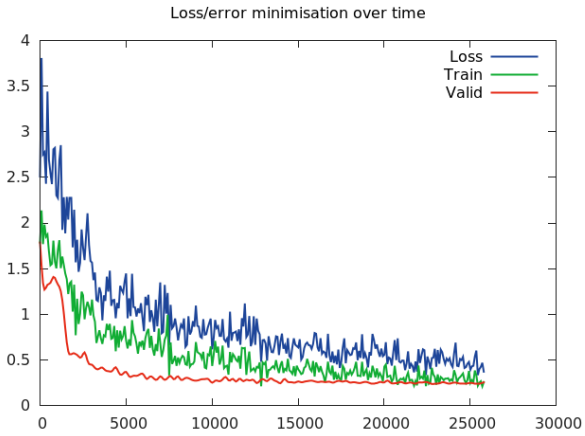


Figure 23 – Network training evolution. The vertical axis represents the error in meters and the horizontal axis represents the training iterations. The curves in indigo, green and red represents the error measured in meters of the loss function, training and validation data.

As the graph of the Figure 23 shows, the loss function curve stays consistently above all others, while the validation error curve crosses the training error curve after 25,000 iterations. What indicates that the network may started overfitting. Following early stopping criteria, the training was interrupted and the best model, which achieves smaller positioning error on validation data, was saved.

Figure 24 shows some GPS coordinates of UFES-LAP-09-LAP-09-5m-1m test dataset for qualitative analysis. There are four plots in Figure 24, each one representing a 50-long sequence selected from the test dataset above. The vertical and horizontal axis represent global coordinates in meters. Green dots represents the ground truth while the red dots represent GPS coordinates. The correspondences between each ground truth and GPS coordinates are represented by a varied-colored line connecting the green and red dots, respectively.

As by visually inspecting the graphs of sequences a-d of Figure 24 one can easily spot some positioning errors of GPS, but it becomes clearer when compared to the mean error and standard deviation of each of the sequences. From better to worse performance:

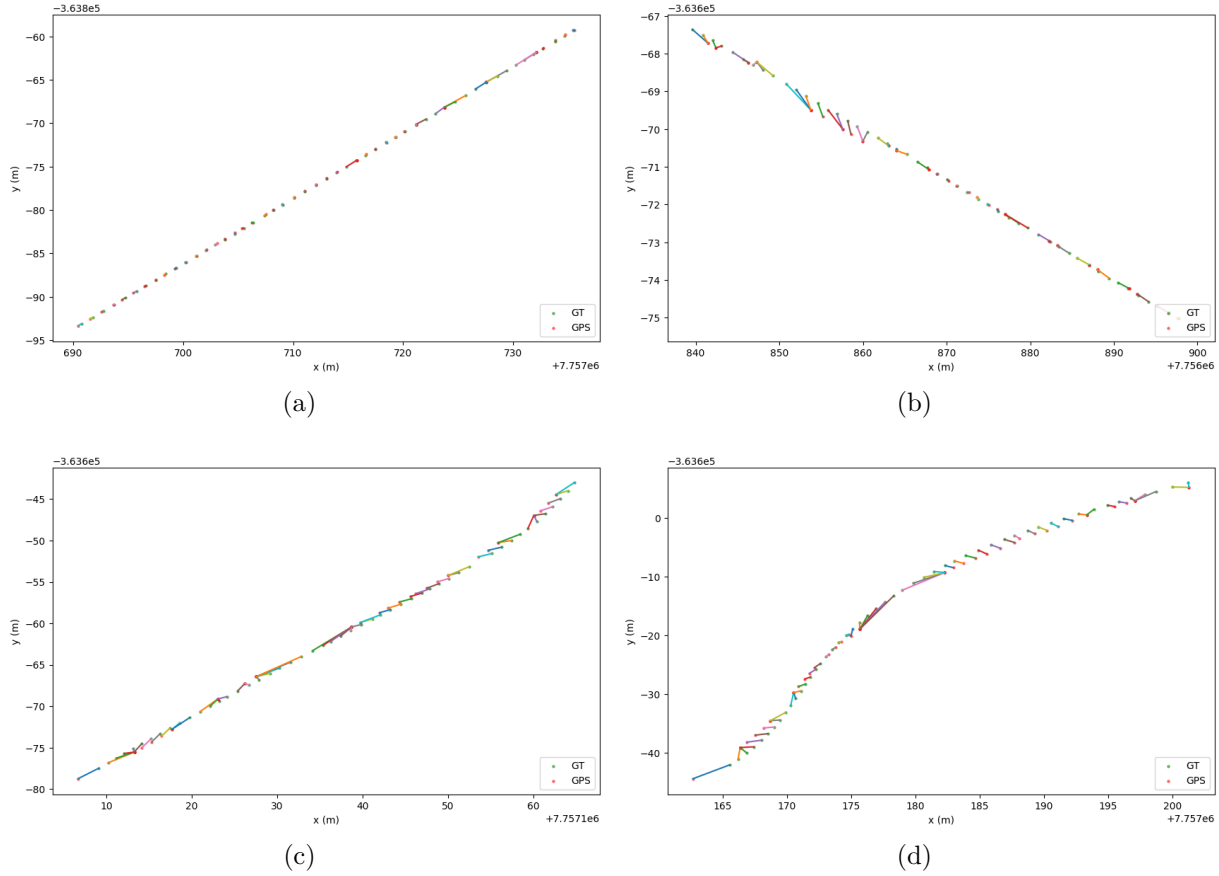


Figure 24 – Performance of GPS-RTK Trimble on UFES-LAP-09-LAP-09-5m-1m test dataset. Each sub-figure represents a sequence of 50 points randomly selected from UFES-LAP-09-LAP-09-5m-1m test dataset. Green dots represents the ground truth while the red dots represent GPS coordinates. Connecting the corresponding ground truth and GPS dots are varied-colored lines.

sequence in Figure 24a finds a mean error of 0.41m and standard deviation of 0.61m; sequence in Figure 24b finds 0.81m and 0.74m, respectively; sequence in Figure 24c finds 1.39m and 1.25m, respectively; sequence in Figure 24d finds 1.88m and 1.14m, respectively. For errors in heading direction it finds the following pairs of mean and standard deviation expressed in radians for: sequence in Figure 24a (0.4099,0.9020); sequence in Figure 24b (0.1186,0.0315); sequence in Figure 24c (0.5370,0.4231); and sequence in Figure 24d (0.5720,0.1059).

Figure 25 shows some coordinates of UFES-LAP-09-LAP-09-5m-1m test dataset inferred by convolutional network. There are four plots in Figure 25, each one representing a 50-long sequence randomly selected from the test dataset above. The vertical and horizontal axis represent global coordinates in meters. Green dots represents the ground truth while the red dots represent the CNN output coordinates. The correspondences between each ground truth and CNN output coordinates are represented by a varied-colored line connecting the green and red dots, respectively.

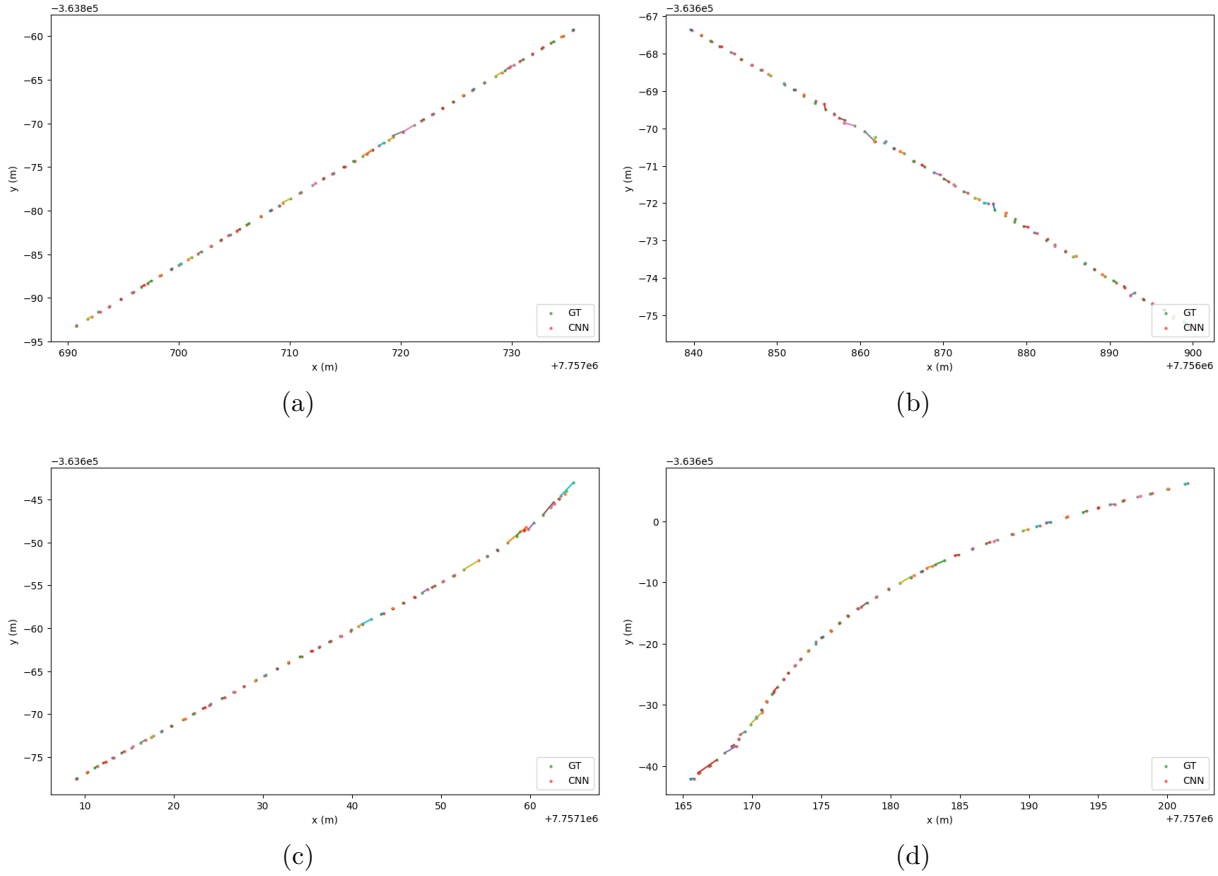


Figure 25 – Performance of CNN approach on UFES-LAP-09-LAP-09-5m-1m test dataset. Each sub-figure represents a sequence of 50 points randomly selected from UFES-LAP-09-LAP-09-5m-1m test dataset. Green dots represents the ground truth while the red dots represent CNN output coordinates. Connecting the corresponding ground truth and CNN dots are varied-colored lines.

Again by visually inspecting the graphs of sequences a-d of Figure 25 one can verify fewer positioning errors of CNN and only becomes clear when compared to the mean error and standard deviation of each of the sequences: sequence a in Figure 25a finds a mean error of 0.29m and standard deviation of 0.28m; sequence b in Figure 25b finds 0.27m and 0.27m, respectively; sequence c in Figure 25c finds 0.37m and 0.50m, respectively; sequence d in Figure 25d finds 0.41m and 0.58m, respectively. For errors in heading direction it finds the following pairs of mean and standard deviation expressed in radians for: sequence a in Figure 25a (0.0022,0.0021); sequence b in Figure 25b (0.0019,0.0017); sequence c in Figure 25c (0.0032,0.0032); and sequence d in Figure 25d (0.0037,0.0049).

Figure 26 shows some coordinates of UFES-LAP-03-LAP-09-5m-1m test dataset jointly inferred by the weightless and convolutional networks. There are four plots in Figure 26, each one representing a 50-long sequence randomly selected from the test dataset above. The vertical and horizontal axis represent global coordinates in meters. Green dots represents the ground truth while the red dots represent the WNN+CNN combined output

coordinates. The correspondences between each ground truth and WNN+CNN combined output coordinates are represented by a varied-colored line connecting the green and red dots, respectively.

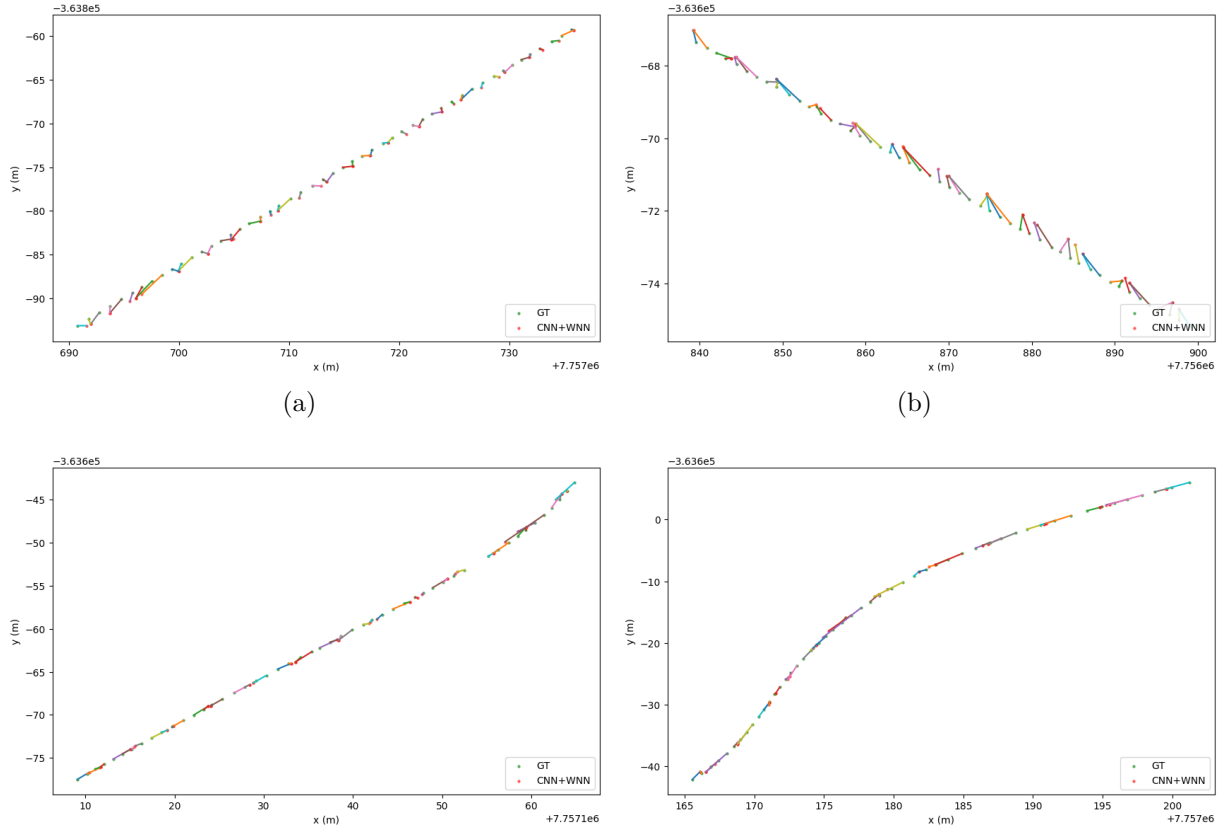


Figure 26 – Performance of WNN+CNN approaches on UFES-LAP-03-LAP-09-5m-1m test dataset. Each sub-figure represent a sequence of 50 points randomly selected from UFES-LAP-03-LAP-09-5m-1m test dataset. Green dots represents the ground truth while the red dots represents WNN+CNN combined output coordinates. Connecting the corresponding ground truth and WNN+CNN dots are varied-colored lines.

Finally by visually inspecting the graphs of sequences a-d of Figure 26 one can verify fewer positioning errors of CNN and only becomes clear when compared to the mean error and standard deviation of each of the sequences: sequence a in Figure 26a finds a mean error of 0.89m and standard deviation of 0.55m; sequence b in Figure 26b finds 1.27m and 0.81m, respectively; sequence c in Figure 26c finds 1.41m and 1.09m, respectively; sequence d in Figure 26d finds 1.18m and 0.94m, respectively. For errors in heading direction it finds the following pairs of mean and standard deviation expressed in radians for: sequence a in Figure 26a (0.0042,0.0025); sequence b in Figure 26b (0.0017,0.0015); sequence c in Figure 26c (0.0215,0.0236); and sequence d in Figure 26d (0.0109,0.0184).

The CNN approach alone, although unfeasible, serves as theoretical benchmark. It

Table 3 – Systems positioning error in meters. Each table row represents the mean and standard deviation for each system indicated in first column of sequences a-d of Figures 25, 24 and 26.

	Figure (a)		Figure (b)		Figure (c)		Figure (d)	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
GPS	0.41m	0.61m	0.81m	0.74m	1.39m	1.25m	1.88m	1.14m
CNN+GT	0.29m	0.28m	0.27m	0.27m	0.37m	0.50m	0.41m	0.58m
CNN+WNN	0.89m	0.55m	1.27m	0.81m	1.41m	1.09m	1.18m	0.94m

Table 4 – Systems heading error in radians. Each table row represents the mean and standard deviation for each system indicated in first column of sequences a-d of Figures 25, 24 and 26.

	Figure (a)		Figure (b)		Figure (c)		Figure (d)	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
GPS	0.4099	0.9020	0.1186	0.0315	0.5370	0.4231	0.5720	0.1059
CNN+GT	0.0022	0.0021	0.0019	0.0017	0.0032	0.0032	0.0037	0.0049
CNN+WNN	0.0042	0.0025	0.0017	0.0015	0.0215	0.0236	0.0109	0.0184

presents the best statistics for all sequences and also for the full sequence with the following mean and standard deviations of distance error (0.25m, 0.28m) and heading direction error (0.0024rad, 0.0026rad). When analyzing results GPS with that of WNN+CNN approach the comparison of the four short sequence ends in a tie but looking the entire sequence statistics GPS wins. GPS statistics for the entire dataset is as follows: mean and standard deviation of positioning error is (0.37m, 0.54m) and heading direction error mean and standard deviation is (0.1067rad, 0.3712rad). The corresponding statistics for WNN+CNN approach are the following: (1.20m, 0.93m) and (0.0121rad, 0.1665rad). Table 3 summarizes the statistics for positioning error in meters of sequences a-d of Figures 25, 24 and 26. Table 4 summarizes the statistics for heading error in radians of sequences a-d of Figures 25, 24 and 26.

Figure 27 shows the results as box plots with median, inter-quartile range and whiskers of the error distribution for each system. It was measured the positioning error of the proposed system and GPS by means of how close their estimated trajectories are to the trajectory estimated by the OGM-MCL system (our ground truth) on the UFES-LAPS-TEST-1M test dataset.

As shown in Figure 27, positioning errors of GPS and GT+CNN systems are equivalent. For the WNN+CNN system, the positioning error of 50% of the poses are under 1m and of 75% of the poses are under 2.3m. Extending the comparison to the Visual SLAM system (JÚNIOR et al., 2015) in a similar context, the combined approach has mean positioning error of 1.20m, slightly higher than the 1.12m performed by the

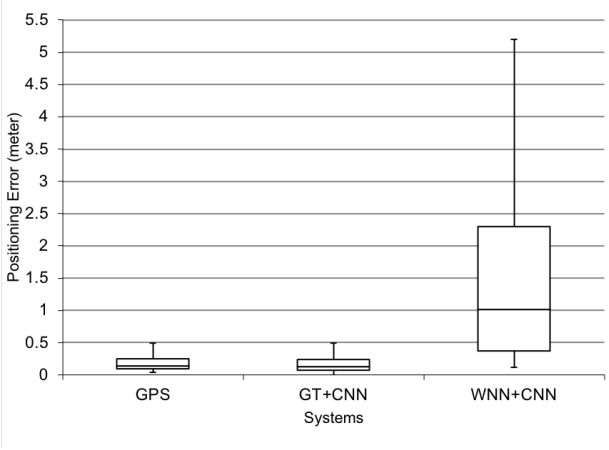


Figure 27 – Comparison between hybrid WNN-CNN system and GPS positioning error.

Visual SLAM system in the same trajectory. Considering they serve to different purposes, combined results of the hybrid WNN-CNN approach looks promising.

6 Conclusion

This chapter summarizes the general ideas discussed in this work and the overall results achieved.

6.1 Discussion

It has been shown that the VG-RAM neuron has high classification performance for a variety of multi-class classification applications, such as text categorization (DE SOUZA et al., 2008; DE SOUZA et al., 2009), face recognition (DE SOUZA et al., 2008; DE SOUZA; BADUE; PEDRONI, 2010; DE MORAES; DE SOUZA; BADUE, 2011), and traffic sign detection and recognition (BERGER et al., 2012; DE SOUZA et al., 2013). In this work, it was studied the place recognition problem as a classification task using the standard VG-RAM neuron, the Fat-fast neuron and the kNN-DTW neuron. The place recognition problem, as designed here, is solved in two phases: mapping and localization. During mapping, the kNN-DTW neuron is trained with a sequence of images and associated poses representing episodes experienced by an autonomous car. During localization, it receives subsequences of images from the same environment and compares them to its previous experienced episodes trying to recollect the most similar "experience" in time and space at once. Then, the system outputs the positions where it "believes" these images were captured. For the Fat-Fast and standard VG-RAM neurons, it is used single images instead of a sequence of images. Moreover, to solve the global localization problem, which consists in measuring the actual robot position and orientation, it is trained a convolutional neural network to learn the relative pose (position and orientation) of images taken from a live camera compared to the recollected images taken from a camera in the past. This way, the recollected pose is composed with the relative pose in a transformation that approximates the robot actual pose better than just outputting where it was in the past when traversing a specific route.

As mentioned before, the bottleneck of VG-RAM based methods is the testing phase. Standard VG-RAM neurons perform a linear search on their training set to find the output value with the nearest input vector. Fat-Fast neurons, in the other hand, exploits the regularity and constancy among bit substrings within its input vectors to perform a faster indexed search. Such an improvement in the technique enables achieving high speedups during the test phase of applications with very large training datasets. A drawback of the Fat-Fast neuron is the increase of the used memory because of the multi-indexed hash tables. Applications running on desktop, with good memory capacity,

impose no additional problem. However sometimes, it is desirable to keep the use of the memory to a certain level following the constraints of a specific machine. To cope with such cases and to reduce even more the testing time of the applications, we also propose to use a memory reduction technique (AGUIAR et al., 2014) together with the fat-fast approach.

To address these problems and propose a solution for place recognition based with large training datasets, this work investigated the use of kNN-DTW and Fat-Fast neurons along with data compression techniques to reduce the overall size of the neurons' memory, while keeping a high and acceptable classification performance. Under the hypothesis that not all data stored in the memory of a neuron are relevant for good classification performance of the network, it is applied data clustering techniques to eliminate redundant or irrelevant examples from the memory (i.e. to eliminate lines of the look-up table illustrated in Figure 1. Given a challenging multi-class classification task, such as the place recognition task, it was compared the classification performance of the standard VG-RAM neuron using full memory with that of Fat-Fast and standard VG-RAM using the memory compressed at different levels. Additionally, it was compared the performance of the kNN-DTW with that of standard VG-RAM neuron, which showed that although sensitive to redundant data kNN-DTW performed equivalent to standard VG-RAM neuron for multiple lap data. The results showed that it is possible to run such heavy multi-class classification task implemented with Fat-Fast VG-RAM neuron at fast response times on standard computers. Moreover, the results showed that all three neurons studied were able recognize a place visited many times before at nearly 90% of the time.

Nevertheless, to solve the global localization problem it is required more than just outputting the position where the robot was during mapping phase. It is desired a means to approximate the actual robot position and orientation with respect to the recollected pose. Such task involves computing a relative camera pose given two images. This problem was tackled here by training a siamese-like convolutional neural network that takes as input two images and regress a 6-DoF relative pose. It was advocated that using a geometry loss to project the 3D points transformed by network's output pose is better approach than using the ground truth pose as the backpropagation signal. It naturally balances the differences in scaling of the position and rotation units, for instance. It was verified also that the loss function error is consistently above training and validation errors, until convergence. The geometry loss function apparatus demonstrated being a robust loss for the task of regressing the relative pose. For the final experiment, the best trained model was applied in conjunction with the VG-RAM neuron to solve the global localization problem. It was shown that the combined results of the hybrid weightless-weighted neural network approach were on pair with a Visual SLAM system, although needs improvements compared to RTK-GPS precision.

6.2 Future Work

A direction for future work, involves extending this work for larger datasets and longer distances such as from Ufes to Guarapari. These larger data sets would not fit entirely into the memory of the WNN neuron but could alternatively be represented using Information Retrieval techniques such as inverted files (BAEZA-YATES; RIBEIRO-NETO, 2011). In addition, most training algorithms of Deep Neural Networks (DNN) are data-hungry, and, as more data are provided, it is expected an increase in accuracy and regularization as shown in (CONTRERAS; MAYOL-CUEVAS, 2017). They improved the PoseNet's (KENDALL; GRIMES; CIPOLLA, 2015) localization accuracy by increasing the number of training trajectories while maintaining a constant-size CNN. Deep learning models have demonstrated superhuman performance in some tasks. However, training DNN using standard supervised techniques requires large amounts of correctly labeled data, which is costly in robotics context. Being able to use noisy labeled data or even unlabeled data involves applying weakly supervised techniques (ZHOU, 2017) and would open doors to many new applications in robotics, such as solving the Visual SLAM problem using end-to-end deep learning techniques. For the relative pose estimation task studied here, one possibility to overcome noisy labeled data is to incorporate its uncertainty in the loss function as an extra parameter to learn the uncertainty related to the pose as discussed in (KENDALL; CIPOLLA, 2017)

Also, it is interesting experimenting with recent new models such as Capsule Networks (SABOUR; FROSST; HINTON, 2017) and Gated Feature Learning models (MEMISEVIC, 2013). The latter uses multiplicative interactions to learn content-independent features to encode only the transformation between image pairs. As these models learn a representation for image pairs related by some transformation it is interesting to check whether they can learn to relate pairs of key frames and live frames, like those used to train the convolutional network in Section 4.3, by some transformation between them instead of their content. In (KONDA; MEMISEVIC, 2015) they found promising results at training a CNN for Visual Odometry, i.e., predicting velocity and direction changes. Their architecture uses a single type of computational module and learning rule to extract visual motion, depth, and finally odometry information from the raw data. Representations of depth and motion are extracted by detecting synchrony across time and stereo channels using network layers with multiplicative interactions. The extracted representations are turned into information about changes in velocity and direction using a convolutional neural network. Which points again to weakly supervised techniques as a direction for future work.

Bibliography

AGRAWAL, P.; CARREIRA, J.; MALIK, J. Learning to see by moving. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. v. 2015 Inter, p. 37–45. Cited on page 31.

AGUIAR, E. de; FORECHI, A.; VERONESE, L.; BERGER, M.; DE SOUZA, A. F.; BADUE, C.; OLIVEIRA-SANTOS, T. Compressing VG-RAM WNN memory for lightweight applications. In: *2014 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2014. p. 1063–1070. Cited 9 times on pages 37, 38, 44, 46, 47, 48, 64, 70, and 84.

ALEKSANDER, I. Self-adaptive universal logic circuits. *Electronics Letters*, v. 2, n. 8, p. 321–322, 1966. Cited on page 35.

ALEKSANDER, I. FROM WISARD TO MAGNUS: A FAMILY OF WEIGHTLESS VIRTUAL NEURAL MACHINES. In: *Ram-Based Neural Networks*. [S.l.]: World Scientific, 1998. p. 18–30. Cited 2 times on pages 35 and 36.

ALEKSANDER, I.; GREGORIO, M. D.; FRANÇA, F. A brief introduction to Weightless Neural Systems. In: *Proceedings of ESANN 2009*. [S.l.: s.n.], 2009. p. 22–24. Cited on page 36.

ALEKSANDER, I.; MORTON, H. *An Introduction to Neural Computing*. [S.l.]: International Thomson Computer Press, 1995. 322 p. Cited on page 36.

ALEKSANDER, I.; THOMAS, W. V.; BOWDEN, P. A. WISARD: a radical step forward in image recognition. *Sensor Review*, v. 4, n. 3, p. 120–124, 1984. Cited on page 36.

ALONSO, C.; PRIETO, Ó.; RODRÍGUEZ, J. J.; BREGÓN, A. Multivariate Time Series Classification via Stacking of Univariate Classifiers. In: OKUN, O.; VALENTINI, G. (Ed.). *Supervised and Unsupervised Ensemble Methods and their Applications*. [S.l.]: Springer Berlin Heidelberg, 2008, (Studies in Computational Intelligence, v. 126). p. 135–151. Cited 2 times on pages 30 and 31.

ALY, M.; ATIYA, A. Novel Methods for the Feature Subset Ensembles Approach. *International Journal of Artificial Intelligence and Machine Learning*, v. 6, n. 4, 2006. Cited 2 times on pages 30 and 50.

ARANDJELOVIC, R.; GRONAT, P.; TORII, A.; PAJDLA, T.; SIVIC, J. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, XX, n. X, p. 1–14, 2017. Cited on page 29.

ARGAMON-ENGELSON, S. Using image signatures for place recognition. *Pattern Recognition Letters*, v. 19, n. 10, p. 941–951, 1998. Cited on page 27.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval: The Concepts and Technology behind Search*. [S.l.]: Addison–Wesley Professional, 2011. 944 p. Cited on page 85.

BAY, H.; ESS, A.; TUYTELAARS, T.; Van Gool, L. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, v. 110, n. 3, p. 346–359, 2008. Cited 5 times on pages 27, 28, 29, 31, and 52.

BERGER, M.; FORECHI, A.; DE SOUZA, A. F.; DE, J.; NETO, O.; VERONESE, L.; NEVES, V.; De Aguiar, E.; BADUE, C. Traffic Sign Recognition with WiSARD and VG-RAM Weightless Neural Networks. *Journal of Network and Innovative Computing ISSN*, v. 1, p. 2160–2174, 2013. Cited on page 36.

BERGER, M.; FORECHI, A.; SOUZA, A. F. D.; De Oliveira Neto, J.; VERONESE, L.; BADUE, C. Traffic sign recognition with VG-RAM Weightless Neural Networks. In: *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*. [S.l.]: IEEE, 2012. p. 315–319. Cited 2 times on pages 36 and 83.

BLEDSOE, W. W.; BROWNING, I. Pattern recognition and reading by machine. In: *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference on - IRE-AIEE-ACM '59 (Eastern)*. [S.l.]: ACM Press, 1959. (IRE-AIEE-ACM '59 (Eastern)), p. 225–232. Cited on page 35.

BRACHMANN, E.; KRULL, A.; NOWOZIN, S.; SHOTTON, J.; MICHEL, F.; GUMHOLD, S.; ROTHER, C. DSAC — Differentiable RANSAC for Camera Localization. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2017. p. 2492–2500. Cited on page 32.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. *The 2005 Darpa grand challenge : the great robot race*. [S.l.]: Springer, 2007. 520 p. Cited on page 21.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. *The DARPA Urban Challenge : autonomous vehicles in city traffic*. [S.l.]: Springer, 2009. 625 p. Cited 2 times on pages 21 and 22.

BURGESS, N. Spatial cognition and the brain. *Annals of the New York Academy of Sciences*, v. 1124, p. 77–97, 2008. Cited on page 22.

BURGESS, N.; MAGUIRE, E. A.; O'KEEFE, J. The Human Hippocampus and Spatial and Episodic Memory. *Neuron*, v. 35, n. 4, p. 625–641, 2002. Cited on page 22.

CARDOSO, D. O.; GAMA, J.; De Gregorio, M.; FRANÇA, F. M. G.; GIORDANO, M.; LIMA, P. M. V. WIPS: the WiSARD Indoor Positioning System. In: *Proceedings of ESANN 2013*. [S.l.: s.n.], 2013. p. 521–526. Cited on page 36.

CARDOSO, D. O.; LIMA, P. M. V.; De Gregorio, M.; GAMA, J.; FRANÇA, F. M. G.; GREGORIO, M. D.; GAMA, J. Clustering data streams with weightless neural networks. In: *Proceedings of ESANN 2011*. [S.l.: s.n.], 2011. p. 201–206. Cited on page 36.

CHEN, Z.; JACOBSON, A.; ERDEM, U.; HASSELMO, M.; MILFORD, M. Multi-scale bio-inspired place recognition. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2014. p. 1895–1901. Cited on page 29.

CHEN, Z.; LAM, O.; JACOBSON, A.; MILFORD, M. Convolutional Neural Network-based Place Recognition. In: *2014 Australasian Conference on Robotics and Automation (ACRA 2014)*. [S.l.: s.n.], 2014. p. 8. Cited on page 29.

CHEN, Z.; LOWRY, S.; JACOBSON, A.; HASSELMO, M. E.; MILFORD, M.

Bio-inspired homogeneous multi-scale place recognition. *Neural Networks*, Elsevier Ltd, v. 72, p. 48–61, 2015. Cited on page 29.

CHEN, Z.; LOWRY, S.; JACOBSON, A.; GE, Z.; MILFORD, M. Distance metric learning for feature-agnostic place recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.]: IEEE, 2015. p. 2556–2563. Cited on page 29.

CHOPRA, S.; HADSELL, R.; Y., L. Learning a similiarty metric discriminatively, with application to face verification. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2005. p. 349–356. Cited on page 31.

CONTRERAS, L.; MAYOL-CUEVAS, W. Towards CNN Map Compression for camera relocalisation. 2017. Cited on page 85.

CORAGGIO, P.; De Gregorio, M. WiSARD and NSP for Robot Global Localization. In: *Proceedings of the 2nd International Work-conference on Nature Inspired Problem-Solving Methods in Knowledge Engineering: Interplay Between Natural and Artificial Computation, Part II*. [S.l.]: Springer-Verlag, 2007. (IWINAC '07), p. 449–458. Cited on page 36.

CUMMINS, M.; NEWMAN, P. Probabilistic Appearance Based Navigation and Loop Closing. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. [S.l.]: IEEE, 2007. p. 2042–2048. Cited 2 times on pages 27 and 28.

CUMMINS, M.; NEWMAN, P. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, v. 27, n. 6, p. 647–665, 2008. Cited on page 28.

CUMMINS, M.; NEWMAN, P. Highly scalable appearance-only SLAM-FAB-MAP 2.0. *Robotics: Science and Systems*, p. 1–8, 2009. Cited on page 28.

CUMMINS, M.; NEWMAN, P. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, v. 30, n. 9, p. 1100–1123, 2010. Cited on page 28.

DE MORAES, J. L.; DE SOUZA, A. F.; BADUE, C. Facial access control based on VG-RAM weightless neural networks. In: *Proceedings of the International Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. p. 444–450. Cited 2 times on pages 36 and 83.

DE SOUZA, A. F.; BADUE, C.; MELOTTI, B. Z.; PEDRONI, F. T.; ALMEIDA, F. L. L. Improving VG-RAM WNN Multi-label Text Categorization via Label Correlation. In: *2008 Eighth International Conference on Intelligent Systems Design and Applications*. [S.l.]: IEEE, 2008. v. 1, p. 437–442. Cited 2 times on pages 36 and 83.

DE SOUZA, A. F.; BADUE, C.; PEDRONI, F. VG-RAM Weightless Neural Networks for Face Recognition. *Face Recognition*, p. 171–187, 2010. Cited 2 times on pages 36 and 83.

DE SOUZA, A. F.; BADUE, C.; PEDRONI, F.; OLIVEIRA, E.; DIAS, S. S.; OLIVEIRA, H.; SOUZA, S. F. de. Face Recognition with VG-RAM Weightless Neural Networks. In: *Artificial Neural Networks*. [S.l.]: Springer, 2008. p. 951–960. Cited 4 times on pages 30, 36, 50, and 83.

DE SOUZA, A. F.; FONTANA, C.; MUTZ, F. W.; De Oliveira, T. A.; BERGER, M.;

FORECHI, A.; De Oliveira Neto, J.; AGUIAR, E. de; BADUE, C. Traffic sign detection with VG-RAM weightless neural networks. In: *Proceedings of the International Joint Conference on Neural Networks*. [S.l.]: IEEE, 2013. p. 1–9. Cited 2 times on pages 36 and 83.

DE SOUZA, A. F.; PEDRONI, F.; OLIVEIRA, E.; CIARELLI, P. M.; HENRIQUE, W. F.; VERONESE, L.; BADUE, C. Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing*, 2009. Cited 2 times on pages 36 and 83.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255. Cited on page 42.

DONOSER, M.; SCHMALSTIEG, D. Discriminative feature-to-point matching in image-based localization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 516–523, 2014. Cited on page 31.

ELFES, A. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, v. 22, n. 6, p. 46–57, 1989. Cited 2 times on pages 27 and 58.

ENGELSON, S. *Passive Map Learning and Visual Place Recognition*. PhD Thesis (PhD Thesis), 1994. Cited on page 27.

FORECHI, A.; DE SOUZA, A. F.; BADUE, C.; OLIVEIRA-SANTOS, T. Sequential appearance-based Global Localization using an ensemble of kNN-DTW classifiers. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2016. v. 2016-Octob, p. 2782–2789. Cited 6 times on pages 30, 33, 49, 64, 70, and 71.

FORECHI, A.; DE SOUZA, A. F.; NETO, J. O.; AGUIAR, E. de; BADUE, C.; GARCEZ, A.; OLIVEIRA-SANTOS, T. Fat-Fast VG-RAM WNN: A High Performance Approach. *Neurocomputing*, v. 183, n. Weightless Neural Systems, p. 56–69, 2015. Cited 5 times on pages 33, 45, 64, 70, and 72.

FOX, C.; PRESCOTT, T. Hippocampus as unitary coherent particle filter. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Ieee, p. 1–8, 2010. Cited on page 28.

FRANCA, H. L.; SILVA, J. C. P. da; LENGERKE, O.; DUTRA, M. S.; De Gregorio, M.; FRANCA, F. M. G. Movement persuit control of an offshore automated platform via a RAM-based neural network. In: *2010 11th International Conference on Control Automation Robotics and Vision*. [S.l.]: IEEE, 2010. p. 2437–2441. Cited on page 36.

FUKUSHIMA, K. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980. Cited on page 39.

GALVEZ-LOPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, v. 28, n. 5, p. 1188–1197, 2012. Cited on page 27.

GLOVER, A.; MADDERN, W.; MILFORD, M.; WYETH, G. FAB-MAP + RatSLAM: Appearance-based SLAM for multiple times of day. *2010 IEEE International Conference on Robotics and Automation*, Ieee, p. 3507–3512, 2010. Cited on page 28.

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. *Development*, p. 607, 2007. Cited on page 39.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: The MIT Press, 2016. Cited 3 times on pages 41, 42, and 75.

HAFTING, T.; FYHN, M.; MOLDEN, S.; MOSER, M.-B.; MOSER, E. I. Microstructure of a spatial map in the entorhinal cortex. *Nature*, v. 436, n. 7052, p. 801–6, 2005. Cited on page 22.

HANDA, A.; BLOESCH, M.; PĂTRĂUCEAN, V.; STENT, S.; MCCORMAC, J.; DAVISON, A. Gvnn: Neural network library for geometric computer vision. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2016. v. 9915 LNCS, p. 67–82. Cited 3 times on pages 52, 53, and 54.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. [S.l.: s.n.], 2004. 672 p. Cited 3 times on pages 31, 52, and 54.

HAYS, J.; EFROS, A. A. IM2GPS: estimating geographic information from a single image. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE, 2008. v. 05, p. 1–8. Cited on page 27.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. v. 2015 Inter, p. 1026–1034. Cited on page 53.

HOU, Y.; ZHANG, H.; ZHOU, S. Convolutional neural network-based image representation for visual loop closure detection. In: *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*. [S.l.: s.n.], 2015. v. 39, n. 3, p. 2238–2245. Cited on page 27.

HUANG, G.; LIU, Z.; MAATEN, L. van der; WEINBERGER, K. Q. Densely Connected Convolutional Networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2017. p. 2261–2269. Cited on page 32.

HUANG, T. S.; NETRAVALI, A. N. Motion and Structure from Feature Correspondences: A Review. *Proceedings of the IEEE*, v. 82, n. 2, p. 252–268, 1994. Cited on page 23.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, v. 195, n. 1, p. 215–243, 1968. Cited on page 39.

IRSCHARA, A.; ZACH, C.; FRAHM, J. M.; BISCHOF, H. From structure-from-motion point clouds to fast location recognition. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*. [S.l.: s.n.], 2009. p. 2599–2606. Cited on page 31.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM Computing Surveys*, v. 31, n. 3, p. 264–323, 1999. Cited 2 times on pages 46 and 47.

JÚNIOR, L. J. L.; OLIVEIRA-SANTOS, T.; BADUE, C.; DE SOUZA, A. F. Image-based mapping, global localization and position tracking using VG-RAM weightless neural

networks. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.]: IEEE, 2015. p. 3603–3610. Cited 5 times on pages 24, 27, 30, 57, and 80.

KENDALL, A.; CIPOLLA, R. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2017. p. 6555–6564. Cited 2 times on pages 54 and 85.

KENDALL, A.; GRIMES, M.; CIPOLLA, R. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. v. 2015 Inter, p. 2938–2946. Cited 4 times on pages 32, 33, 54, and 85.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. [S.l.], 2014. Cited on page 75.

KOMATI, K. S.; DE SOUZA, A. F. Using weightless neural networks for vergence control in an artificial vision system. *Applied Bionics and Biomechanics*, v. 1, n. 1, p. 21–31, 2003. Cited on page 25.

KONDA, K.; MEMISEVIC, R. Learning Visual Odometry with a Convolutional Network. *International Conference on Computer Vision Theory and Applications*, p. 486–490, 2015. Cited on page 85.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2012. p. 1097–1105. Cited on page 42.

LABBE, M.; MICHAUD, F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, v. 29, n. 3, p. 734–745, 2013. Cited on page 27.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Cited on page 25.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Cited 4 times on pages 27, 39, 40, and 64.

LEUTGEB, S.; LEUTGEB, J. K.; BARNES, C. A.; MOSER, E. I.; MCNAUGHTON, B. L.; MOSER, M.-B. Independent codes for spatial and episodic memory in hippocampal neuronal ensembles. *Science (New York, N.Y.)*, American Association for the Advancement of Science, v. 309, n. 5734, p. 619–23, 2005. Cited on page 22.

LI, Y.; SNAVELY, N.; HUTTENLOCHER, D. P.; FUA, P. Worldwide Pose Estimation Using 3D Point Clouds. In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2012. p. 15–29. Cited on page 31.

LIM, H.; SINHA, S. N.; COHEN, M. F.; UYTTENDAELE, M. Real-time image-based 6-DOF localization in large-scale environments. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2012. p. 1043–1050. Cited on page 31.

LIN, T. Y.; CUI, Y.; BELONGIE, S.; HAYS, J. Learning deep representations for

ground-to-aerial geolocalization. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. v. 07-12-June, p. 5007–5015. Cited on page 27.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. v. 07-12-June, p. 3431–3440. Cited 2 times on pages 33 and 52.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, v. 60, n. 2, p. 91–110, 2004. Cited 2 times on pages 27 and 52.

LOWRY, S.; SUNDERHAUF, N.; NEWMAN, P.; LEONARD, J. J.; COX, D.; CORKE, P.; MILFORD, M. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, PP, n. 99, p. 1–19, 2015. Cited on page 23.

LUDERMIR, T.; CARVALHO, A.; BRAGA, A.; SOUTO, M. M. Weightless neural models: a review of current and past works. *Neural Computing Surveys*, p. 41–61, 1999. Cited on page 35.

LYNEN, S.; BOSSE, M.; FURGALE, P.; SIEGWART, R. Placeless place-recognition. In: *Proceedings - 2014 International Conference on 3D Vision, 3DV 2014*. [S.l.]: IEEE, 2015. p. 303–310. Cited on page 27.

LYRIO JÚNIOR, L. J.; OLIVEIRA-SANTOS, T.; FORECHI, A.; VERONESE, L.; BADUE, C.; De Souza, A. F. A. Image-based global localization using VG-RAM Weightless Neural Networks. In: *2014 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2014. p. 3363–3370. Cited 7 times on pages 29, 30, 48, 50, 57, 64, and 70.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.]: The Regents of the University of California, 1967. p. 281–297. Cited 2 times on pages 46 and 47.

MATSUMOTO, Y.; INABA, M.; INOUE, H. Visual navigation using view-sequenced route representation. In: *International Conference on Robotics and Automation*. [S.l.]: IEEE, 1996. v. 1, n. April, p. 83–88. Cited on page 27.

MELEKHOV, I.; YLIOINAS, J.; KANNALA, J.; RAHTU, E. Relative Camera Pose Estimation Using Convolutional Neural Networks. Springer, Cham, p. 675–687, 2017. Cited on page 33.

MEMISEVIC, R. Learning to relate images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1829–1846, 2013. Cited on page 85.

MIDDELBERG, S.; SATTLER, T.; UNTZELMANN, O.; KOBBELT, L. Scalable 6-DOF localization on mobile devices. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2014. v. 8690 LNCS, n. PART 2, p. 268–283. Cited on page 31.

MILFORD, M. Visual Route Recognition with a Handful of Bits. *Proceedings of Robotics*

Science and Systems Conference,University of Sydney, v. 2012, 2012. Cited on page 30.

MILFORD, M. Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, v. 32, n. 7, p. 766–789, 2013. Cited on page 30.

MILFORD, M.; SHEN, C.; LOWRY, S.; SUENDERHAUF, N.; SHIRAZI, S.; LIN, G.; LIU, F.; PEPPERELL, E.; LERMA, C.; UPCROFT, B.; REID, I. Sequence Searching With Deep-Learnt Depth for Condition- and Viewpoint-Invariant Route-Based Place Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2015. p. 18–25. Cited on page 29.

MILFORD, M.; VIG, E.; SCHEIRER, W.; COX, D. Vision-based Simultaneous Localization and Mapping in Changing Outdoor Environments. *Journal of Field Robotics*, v. 31, n. 5, p. 814–836, 2014. Cited on page 28.

MILFORD, M.; WYETH, G. Mapping a suburb with a single camera using a biologically inspired SLAM system. *Robotics, IEEE Transactions on*, v. 24, n. 5, p. 1038–1053, 2008. Cited on page 28.

MILFORD, M.; WYETH, G. Persistent Navigation and Mapping using a Biologically Inspired SLAM System. *The International Journal of Robotics Research*, SAGE Publications, v. 29, n. 9, p. 1131–1153, 2009. Cited on page 27.

MILFORD, M.; WYETH, G. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. *2012 IEEE International Conference on Robotics and Automation*, Ieee, p. 1643–1649, 2012. Cited on page 28.

MILFORD, M.; WYETH, G.; PRASSER, D. RatSLAM: a hippocampal model for simultaneous localization and mapping. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* [S.l.]: IEEE, 2004. v. 1, p. 403–408. Cited on page 28.

MISRA, J.; SAHA, I. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, v. 74, n. 1-3, p. 239–255, 2010. Cited on page 35.

MORCINIEC, M.; ROHWER, R. 1 Introduction 2 The n-tuple recognition method. p. 1–11, 1995. Cited on page 35.

MOSER, E. I.; KROPFF, E.; MOSER, M.-B. Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*, Annual Reviews, v. 31, p. 69–89, 2008. Cited on page 22.

MÜLLER, M. *Information retrieval for music and motion*. [S.l.]: Springer-Verlag New York, Inc., 2007. Cited 3 times on pages 50, 51, and 70.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on*, v. 31, n. 5, p. 1147–1163, 2015. Cited on page 27.

MUTZ, F.; VERONESE, L. P.; OLIVEIRA-SANTOS, T.; AGUIAR, E. de; CHEEIN, F. A. A.; DE SOUZA, A. F. Large-scale mapping in complex field scenarios using an autonomous car. *Expert Systems with Applications*, Pergamon, v. 46, p. 439–462, 2016. Cited on page 58.

NAIR, V.; HINTON, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In: *Proceedings of the 27th International Conference on Machine Learning*. [S.l.: s.n.], 2010. p. 807–814. Cited on page 41.

NISTER, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* [S.l.]: IEEE, 2004. v. 1, p. 652–659. Cited on page 23.

NOROUZI, M.; PUNJANI, A.; FLEET, D. Fast search in hamming space with multi-index hashing. *Computer Vision and Pattern ...*, 2012. Cited 2 times on pages 44 and 46.

NOROUZI, M.; PUNJANI, A.; FLEET, D. J. Fast Exact Search in Hamming Space With Multi-Index Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 36, n. 6, p. 1107–1119, 2014. Cited 2 times on pages 44 and 46.

NURMAINI, S.; HASHIM, S. Z. M.; JAWAWI, D. N. A. Modular weightless neural network architecture for intelligent navigation. *International Journal of Advances in Soft Computing and its Applications*, v. 1, n. 1, p. 1–18, 2009. Cited on page 36.

O'KEEFE, J.; DOSTROVSKY, J. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, v. 34, n. 1, p. 171–175, 1971. Cited 2 times on pages 22 and 28.

O'KEEFE, J.; NADEL, L. *The hippocampus as a cognitive map*. [S.l.: s.n.], 1978. Cited on page 22.

OLIVEIRA, G. L.; RADWAN, N.; BURGARD, W.; BROX, T. *Topometric Localization with Deep Learning*. [S.l.], 2017. Cited on page 32.

PRADO, C. B. do; FRANCA, F. M. G.; COSTA, E.; VASCONCELOS, L. A new intelligent systems approach to 3D animation in television. In: *Proceedings of the 6th ACM international conference on Image and video retrieval - CIVR '07*. [S.l.]: ACM Press, 2007. (CIVR '07), p. 117–119. Cited on page 36.

ROHWER, R. Two Bayesian treatments of the n-tuple recognition method. In: . [S.l.: s.n.], 1995. p. 1–6. Cited on page 35.

ROKACH, L. A survey of Clustering Algorithms. In: MAIMON, O. (Ed.). *Data Mining and Knowledge Discovery Handbook*. [S.l.]: Springer US, 2009. p. 269–298. Cited 2 times on pages 46 and 47.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*, Ieee, p. 2564–2571, 2011. Cited 2 times on pages 27 and 52.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Cited on page 40.

SABOUR, S.; FROSST, N.; HINTON, G. E. Dynamic Routing Between Capsules. In: *Advances in Neural Information Processing Systems 30*. [S.l.: s.n.], 2017. p. 3859–3869. Cited on page 85.

SATTLER, T.; LEIBE, B.; KOBBELT, L. Fast image-based localization using direct

2D-to-3D matching. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2011. p. 667–674. Cited on page 31.

SAUL, A.; PRESCOTT, T.; FOX, C. Scaling up a Boltzmann machine model of hippocampus with visual features for mobile robots. In: *2011 IEEE International Conference on Robotics and Biomimetics*. [S.l.]: IEEE, 2011. p. 835–840. Cited on page 29.

SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In: *International Conference on Learning Representations (ICLR2014)*. [S.l.: s.n.], 2014. p. 1312.6229. Cited on page 29.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *International Conference on Learning Representations (ICLR2015)*. [S.l.: s.n.], 2015. Cited on page 52.

SIVIC, J.; ZISSERMAN, A. Video Google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, n. ICCV, p. 2–9, 2003. Cited on page 28.

SONG, Y.; CHEN, X.; WANG, X.; ZHANG, Y.; LI, J. Fast Estimation of Relative Poses for 6-DOF Image Localization. In: *Proceedings - 2015 IEEE International Conference on Multimedia Big Data, BigMM 2015*. [S.l.]: IEEE, 2015. p. 156–163. Cited on page 31.

SOUZA, C. R.; NOBRE, F. F.; LIMA, P. V. M.; SILVA, R. M.; BRINDEIRO, R. M.; FRANÇA, F. M. G. Recognition of HIV-1 subtypes and antiretroviral drug resistance using weightless neural networks. In: *Proceedings of ESANN 2012*. [S.l.: s.n.], 2012. p. 429–434. Cited on page 36.

SPRINGENBERG, J. T.; DOSOVITSKIY, A.; BROX, T.; RIEDMILLER, M. Striving for Simplicity: The All Convolutional Net. In: *International Conference on Learning Representations (ICLR2015)*. [S.l.: s.n.], 2014. Cited on page 53.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014. Cited 3 times on pages 41, 53, and 75.

TAYLOR, G. W.; HINTON, G. E.; ROWEIS, S. Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems*, 2007. Cited on page 29.

THRUN, S. Toward robotic cars. *Communications of the ACM*, ACM, v. 53, n. 4, p. 99, 2010. Cited on page 21.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005. 672 p. Cited 4 times on pages 22, 23, 30, and 58.

TOLA, E.; LEPETIT, V.; FUA, P. DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n. 5, p. 815–830, 2010. Cited on page 52.

TRAPPENBERG, T. Continuous attractor neural networks. *Recent developments in biologically inspired . . .*, p. 1–30, 2003. Cited on page 28.

TULVING, E. *Episodic and semantic memory*. 1972. 381–403 p. Cited on page 22.

TULVING, E. Episodic memory: from mind to brain. *Annual review of psychology*, v. 53, p. 1–25, 2002. Cited on page 22.

ULRICH, I.; NOURBAKHSH, I. Appearance-based place recognition for topological localization. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, v. 2, n. April, p. 1023–1029, 2000. Cited on page 27.

VERONESE, L. d. P.; GUIVANT, J.; CHEEIN, F. A. A.; OLIVEIRA-SANTOS, T.; MUTZ, F.; AGUIAR, E. de; BADUE, C.; De Souza, A. F. A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. [S.l.]: IEEE, 2016. p. 520–525. Cited 2 times on pages 57 and 58.

WEINBERGER, K. Q.; SAUL, L. K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *The Journal of Machine Learning Research*, v. 10, p. 207–244, 2009. Cited on page 29.

WEYAND, T.; KOSTRIKOV, I.; PHILBIN, J. Planet - photo geolocation with convolutional neural networks. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2016. v. 9912 LNCS, p. 37–55. Cited on page 27.

WIDLOSKI, J.; FIETE, I. How Does the Brain Solve the Computational Problems of Spatial Navigation? In: *Space, Time and Memory in the Hippocampal Formation*. [S.l.]: Springer Vienna, 2014. p. 373–407. Cited on page 22.

WOLF, J.; BURGARD, W.; BURKHARDT, H. Using an image retrieval system for vision-based mobile robot localization. *Image and Video Retrieval*, p. 108–119, 2002. Cited on page 28.

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, v. 16, n. 3, p. 645–678, 2005. Cited 2 times on pages 46 and 47.

YAMAGUCHI, K.; MCALLESTER, D.; URTASUN, R. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2014. v. 8693 LNCS, n. PART 5, p. 756–771. Cited 2 times on pages 62 and 74.

YAO, Q.; BEETNER, D.; OSTERLOH, B.; BRAUNSCHWEIG, T. U.; BECTNER, D.; WUNSCH, D. A RAM-based neural network for collision avoidance in a mobile robot. In: *Proceedings of the International Joint Conference on Neural Networks, 2003.* [S.l.]: IEEE, 2003. v. 4, n. figure 2, p. 3157–3160. Cited on page 36.

ZHOU, B.; LAPEDRIZA, A.; XIAO, J.; TORRALBA, A.; OLIVA, A. Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems 27*, p. 487–495, 2014. Cited on page 33.

ZHOU, Z.-H. A brief introduction to weakly supervised learning. *National Science Review*, 2017. Cited on page 85.

Appendix

APPENDIX A – Tables

This chapter contains some complementary information.

Table 5 – Ufes Training Dataset List for Visual Localization

UFES-20160825-20160825-5m-1m 7	UFES-20160825-01-20160825-5m-1m
UFES-20160825-20160825-01-5m-1m	UFES-20160825-01-20160825-01-5m-1m
UFES-20160825-20160825-02-5m-1m	UFES-20160825-01-20160825-02-5m-1m
UFES-20160825-20161021-5m-1m	UFES-20160825-01-20161021-5m-1m
UFES-20160825-20171205-5m-1m	UFES-20160825-01-20171205-5m-1m
UFES-20160825-20180112-5m-1m	UFES-20160825-01-20180112-5m-1m
UFES-20160825-20180112-02-5m-1m	UFES-20160825-01-20180112-02-5m-1m
UFES-20160825-02-20160825-5m-1m	UFES-20161021-20160825-5m-1m
UFES-20160825-02-20160825-01-5m-1m	UFES-20161021-20160825-01-5m-1m
UFES-20160825-02-20160825-02-5m-1m	UFES-20161021-20160825-02-5m-1m
UFES-20160825-02-20161021-5m-1m	UFES-20161021-20161021-5m-1m
UFES-20160825-02-20171205-5m-1m	UFES-20161021-20171205-5m-1m
UFES-20160825-02-20180112-5m-1m	UFES-20161021-20180112-5m-1m
UFES-20160825-02-20180112-02-5m-1m	UFES-20161021-20180112-02-5m-1m
UFES-20171205-20160825-5m-1m	UFES-20180112-20160825-5m-1m
UFES-20171205-20160825-01-5m-1m	UFES-20180112-20160825-01-5m-1m
UFES-20171205-20160825-02-5m-1m	UFES-20180112-20160825-02-5m-1m
UFES-20171205-20161021-5m-1m	UFES-20180112-20161021-5m-1m
UFES-20171205-20171205-5m-1m	UFES-20180112-20171205-5m-1m
UFES-20171205-20180112-5m-1m	UFES-20180112-20180112-5m-1m
UFES-20171205-20180112-02-5m-1m	UFES-20180112-20180112-02-5m-1m
UFES-20180112-02-20160825-5m-1m	
UFES-20180112-02-20160825-01-5m-1m	
UFES-20180112-02-20160825-02-5m-1m	
UFES-20180112-02-20161021-5m-1m	
UFES-20180112-02-20171205-5m-1m	
UFES-20180112-02-20180112-5m-1m	
UFES-20180112-02-20180112-02-5m-1m	

Table 6 – Ufes Validation Dataset List for Visual Localization

UFES-20160825-01-20160830-5m-1m	UFES-20160825-01-20170119-5m-1m
UFES-20160825-02-20160830-5m-1m	UFES-20160825-02-20170119-5m-1m
UFES-20161021-20160830-5m-1m	UFES-20161021-20170119-5m-1m
UFES-20171205-20160830-5m-1m	UFES-20171205-20170119-5m-1m
UFES-20180112-20160830-5m-1m	UFES-20180112-20170119-5m-1m
UFES-20180112-02-20160830-5m-1m	UFES-20180112-02-20170119-5m-1m
UFES-20160830-20160830-5m-1m	UFES-20170119-20170119-5m-1m

Table 7 – Ufes Test Dataset List for Visual Localization

UFES-20160825-20171122-5m-1m
UFES-20160825-01-20171122-5m-1m
UFES-20160825-02-20171122-5m-1m
UFES-20161021-20171122-5m-1m
UFES-20171205-20171122-5m-1m
UFES-20180112-20171122-5m-1m
UFES-20180112-02-20171122-5m-1m
UFES-20171122-20171122-5m-1m