Universidade Federal da Paraíba

Centro de Informática

Programa de Pós-Graduação em Informática

SpaceYNet: a Regression Pose and Depth-scene Simultaneously

Dunfrey Pires Aragão

Tiago Pereira do Nascimento

(Orientador)

João Pessoa, Paraíba, Brasil

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Dunfrey Pires Aragão, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 31 de janeiro de 2020.

Aos trinta e um dias do mês de janeiro, do ano de dois mil e vinte, às dez horas, no Centro de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do Sr. Dunfrey Pires Aragão, vinculado a esta Universidade sob a matrícula nº 20181000617, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Sinais, Sistemas Digitais e Gráficos", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Tiago Pereira do Nascimento (PPGI-UFPB), Orientador e Presidente da Banca, Thais Gaudêncio do Rego (PPGI-UFPB), Examinadora Interna, Luiz Marcos Garcia Gonçalves (UFRN), Examinador Externo à Instituição. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado: "SpaceYNet: a Regression Pose and Depth-scene Simultaneously". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 31 de janeiro de 2020.

Prof. Dr. Ruy Alberto Pisani Altafim

Prof. Tiago Pereira do Nascimento
Orientador (PPGI-UFPB)

Profª. Thais Gaudêncio do Rego
Examinadora Interna (PPGI-UFPB)

Prof. Luiz Marcos Garcia Gonçalves
Examinador Externo à Instituição (UFRN)

Catalogação na publicação
Seção de Catalogação e Classificação

A659s Aragão, Dunfrey Pires.
SpaceYNet: a Regression Pose and Depth-scene simultaneously / Dunfrey Pires Aragão. - João Pessoa, 2020.
85 f. : il.

Orientação: Tiago Pereira do Nascimento.
Dissertação (Mestrado) - UFPB/PPGI.

1. Conjunto de dados. 2. depth-scene. 3. pose. 4. rede neural. 5. robô. I. Pereira do Nascimento, Tiago. II. Título.

UFPB/BC

Resumo

Um dos problemas fundamentais da robótica móvel é o uso de informações sensoriais para localizar um agente em um espaço geográfico e evitar que uma ação em ambientes desconhecidos possa levar a efeitos inesperados. Neste trabalho, desenvolvemos um sistema de relocalização global que denominamos SpaceYNet, com o objetivo de prever a posição do robô e evitar ações imprevistas a partir de uma imagem monocular RGB. Para isto, camadas *Inceptions* foram incorporadas em camadas simétricas de amostragem para regredir *depth-scene* e a estimativa de 6-DoF de forma simultaneamente. Além disso, implementamos o uso de redes recorrentes para aprender os informaç oes contextuais do ambiente. Também comparamos o SpaceYNet ao PoseNet e ContextualNet, sendo os estados da arte em regressão de pose de robô, usando CNN e LSTM para avaliar. A comparação compreendeu um conjunto de dados público e outro criado em um ambiente interno amplo. O SpaceYNet apresentou maior precisão nas percentuais globais quando comparado ao PoseNet na regressão de 6-DoF, sendo melhor em 58,42% dos casos com o conjunto de dados público em porcentagens globais quando comparado ao PoseNet, e 56,06% dos casos trabalhando com o novo conjunto de dados. Quando comparado ao SpaceYNet v.2 à rede ContextualNet, também apresentou vantagem de precisão, comparando o SpaceYNet v.1 e o SpaceYNet v.2, e a diferença é de 67,5% de maior precisão.

Palavras-chave: Conjunto de dados, *depth-scene*, pose, rede neural, regressão, robô.

Abstract

One of the fundamental dilemmas of mobile robotics is the use of sensory information to locate an agent in geographic space and avoid acting upon unknown environments that may lead to unexpected effects. In this work, we developed a global relocation system aiming to predict the robot position and avoid unforeseen actions from a monocular image, which we named SpaceYNet. Thus, Inception layers were incorporated into symmetric layers of down-sampling and up-sampling to solve depth-scene and 6-DoF estimation simultaneously. In addition, we implemented a recurrent solution to learn the contextual features of the environment. We also compared SpaceYNet to PoseNet and ContextualNet, being states of the art in robot pose regression, using CNN and LSTM in order to evaluate it. The comparison comprised one public dataset and one created in a large indoor environment. SpaceYNet showed higher accuracy in global percentages when compared to PoseNet in regressing 6-DoF, being better in 58.42% cases with the public dataset in global percentages when compared to the PoseNet, and 56.06% cases working with the newer dataset. When compared to SpaceYNet v.2 to the ContextualNet network, it also presented advantage of accuracy and comparing SpaceYNet v.1 and SpaceYNet v.2, and the difference is 67,5% of higher accuracy.

Keywords: Dataset, depth-scene, neural network, pose, regression, robot.

Contents

List of Symbols

3D *Three-dimensional*

6-DoF *Six Degrees of Freedom*

ANN *Artificial Neural Network*

CNN *Convolutional Neural Network*

EKF *Extended Kalman Filter*

IoU *Intersection-over-union*

LaSER *Laboratory of Systems Engineering and Robotics*

LSTM *Long Short Term Memory*

MAE *Mean Absolute Error*

MRF *Markov Random Field*

MSE *Mean Squared Error*

NN *Neural Network*

ReLU *Rectified Linear Unit*

RGB *Red-Green-Blue*

RGB-D *Red-Green-Blue-Depth*

RMSE *Root Mean Squared Error*

RNN *Recurrent Neural Network*

SGD *Stochastic Gradient Descent*

SIFT *Scale-invariant Feature Transform*

SLAM *Simultaneous Localization and Mapping*

YOLO *You Only Look Once*

List of Figures

List of Tables

Introduction

The automation of human activities has been critical in industrial environments and research because it allows recognizing patterns and performing specific actions. A mandatory characteristic to accomplish the activities is the previous knowledge of the operational environment, avoiding dysfunctions of the planned action. [Bourgault *et al.* 2002], in his work, claims that the accuracy of the mapping process depends directly on the accuracy of the localization process. To understand how the localization process occurs, it is first necessary to recognize the environments where the acts are necessary.

Location systems can apply to target two outcomes:

- to auto-locate in a given environment, and;
- avoid unexpected obstacles on the way.

The accomplishment of the purpose task can provide for the agent autonomy locomotion and allow the extraction of characteristics about the environment. The environment features are intrinsic and interconnected, locating consists on determining the position in a specific plane while orientation is to provide characteristics of space at a time t.

Many works assumed odometer approaches as a way to solve the problem of estimating pose in scope. That proposal demands mechanical artifact measures, such as the computation of a wheel operation, and the performance of the guided distance calculus. However, the approaches specified above may accumulate errors and uncertainties; wheels moving to uneven ground, sensors measuring unexpected obstructions, or even the ground not contacting the wheel are some of these problems.

[Deng and Zeng 2013] has shown a monocular vision solution to object detection and got a remark on which that approach has a general use perspective in the field of indoor navigation. Due to the advantages of its soft structure, such as economic power consumption

and a low amount of information. It is, therefore, a vital foundation technology of vision navigation. The problem is that the robot cannot detect the ward and self-localize.

In other circumstances, some researchers attempted to solve the localization problem by using other types of devices such as sensors (encoders, sonar, and the usage of lasers). Similar to the odometry approach, sensors also have limitations: natural deficiencies presented in electronic artifacts such as noise, signal loss occurrence (partial or total loss), and unexpected queries indirectly can be reported, surface imperfections as well. Furthermore, sonar and laser sensors may transmit a sign in front of a wall surface, receive it back, and the measurement may not match.

Attempting to resolve the self-localize tasks, attention to the Simultaneous Localization and Mapping (SLAM) method became a field investigated in the domain of robotics. The SLAM solution is a probabilistic mapping, addressing a statistical basis to represent associations between landmarks and handle geometric uncertainty. That can estimate the pose of the robot and reconstruct the visited plan from sensors, as discussed by [Durrant-Whyte and Bailey 2006] and [Dissanayake *et al.* 2001]. Part of the SLAM systems stand on the Extended Kalman Filter (EKF). Despite the advantages of simultaneously posing estimation models and map rebuilding, SLAM systems may be affected due to lack in matching scene features frame by frame, operating mechanism separated for location, and positional estimation based on landmarks.

Another field that has a growing scope is the practice of deep learning; this sort of application provides the machine to learn features about the view visited, by the training neuron's links. Discerning the impact and likelihood of a further extension of the use of deep learning in robotics, practicing that approach can answer the dilemma of self-localizing, displacement in space avoiding difficulties faced in predecessor resolutions such as odometry, sensors, and SLAM.

Studies of deep learning have influenced mobile robotics within the localization field tasks by the use of monocular [Kahn *et al.* 2018; Naseer and Burgard 2017], stereo [Xiao, Zhou and Gong 2018], or RGB-D cameras [Henry *et al.* 2013; Li *et al.* 2019]. Kendall and Cipolla [Kendall, Grimes and Cipolla 2015] have created PoseNet network, a Convolutional Neural Network (CNN) based on GoogLeNet 1V1 [Szegedy *et al.* 2015], that learns 2D features from camera images and provides the robot localization inside the environment.

However, the model fails to exploit the temporal relationship between current and past inputs.

In the robotic vision field, it is a common practice to detect minimal distinct features in images, mainly in indoor areas. Blank walls, constant compositions, and regular space plans from a single image may not provide enough information to produce an accurate prediction. Some studies even tried to detect natural landmarks to discriminate features, such as doors and stairs [Souto *et al.* 2017]. Accordingly, the use of a context provided by a sequence of previous images may introduce more distinguishable features and tackle challenging dilemmas.

Concerning that dilemmas locate and detect features from environments, there is significant evidence that the successful practice in deep networks may require several amounts of samples to the training step. Nevertheless, the work presented in [Ronneberger, P.Fischer and Brox 2015] presents a system called U-Net and training policy that relies on the data augmentation to use the handy evaluated representations also efficiently. The network consists of a route to take first the context from the data and then turn the features on an expanded symmetric path.

1.1 Motivation

Reviewing contemporary studies, promoting comparisons between them, those performs with singular duties. The propositions that are investigating further one simultaneous task are not correlated with pose location and space perception,which is the difficulty already.

The clearness architecture and outcomes obtained from U-Net to image segmentation, and the resolution to self-localize proposed by PoseNet: this work intends to join these ideas concepts and generates a new proposition called SpaceYNet to pose and depth-scene regression simultaneously through an RGB image.

SpaceYNet takes advantage of depth-scene prediction to enhance the mobile robot pose estimation simultaneously. Unlike other approaches that use traditional image features and other characteristic knowledge, SpaceYNet becomes useful in situations where the occlusion points are obstacles. Also, the proposed SpaceYNet needs to be able and easy to configure. It means to turn it into a framework to attach news functions as object detection, fitting as a human activity into the real scenario.

1.2 Contributions

The main objective of the work is to employ inception layers and recurrent neural networks within the symmetric architecture of down-sampling and up-sampling sequence layers to estimate the depth-scene image and regress the 6-DoF (six degrees of freedom of a rigid body in three-dimensional space) from the monocular RGB image of a mobile robot camera.

1.2.1 Subsequent Contributions

- Make 6-DoF ground-truth to the RGB-D Scenes Dataset v.2 [Souto *et al*. 2017] from as input to VisualSfM [Wu 2013];
- Create and publish a LaSER dataset [LaSER 2019] designated to terrestrial robot localization and compose it with RGB images, depth images, and a 6-DoF odometry file data;
- Analyze the selection of design architecture, insertion of optimizers and regularizers in SpaceYNet and that relation to outcome the result of the tasks proposed in depth-scene and pose regression;
- Compare the accuracy of the results obtained by SpaceYNet with the PoseNet using the RGB-D Scenes Dataset v.2 and using the LaSER dataset;
- Implement Recurrent Neural Networks and analyze its impacts on pose regression into SpaceYNet;
- Compare the results accuracy obtained by SpaceYNet with recurrent layers with the ContextualNet, PoseNet and first SpaceYNet version without recurrence layers using the RGB-D Scenes Dataset v.2 and using the LaSER dataset.

1.3 Methodology

Firstly, we earned an exploration and studies concerning neural networks and its technologies, supporting the proposal to localize and depth regression. There must be theoretical knowledge. More details are explored in 2.

Then, the study aimed to understand neural networks that have been used to construct and configure SpaceYNet and the experiments in this work, and also the use of recurrent networks and implements them into SpaceYNet. To support the RGB-D Scenes, Dataset v.2 is used with VisualSfM, which is a visual odometry tool, applying on the step where each image needs to have a position and depth label. The LaSER dataset was created inside the LaSER laboratory.

Finally, the dataset LaSER was submitted to SpaceYNet, performing experiments and measurements about losses and accuracy comparisons between the different versions of the SpaceYNet network, PoseNet, and ContextualNet to pose regression. The test procedures and validation will be discussed in 5.

1.4 Thesis Structure

Chapter 2 of this work will show the fundamental theories in order to understand deep learning and extraction of patterns from images using neural networks. Thus, a study will be presented in order to understand how the depth estimation of objects works on the scene and how to use deep learning in these cases.

Before we discuss the dataset and SpaceYNet evaluation, Chapter 3 discusses works on the literature correlated with robot localization and depth scene prediction, comparing them by showing studies and experiment results published.

Next, Chapter 4, we will be talking about the public and the generated datasets by the LaSER lab that we used on the work. Here, we get into neural network architecture proposed, detailing the SpaceYNet construction process.

Chapter 5 shows the SpaceYNet experiments being compared to other networks, their statically and graphical results about the network performance to pose regression and depth reconstruction based, taking some different image examples recorded in the same place and trying to regress it.

Chapter 6 will show the conclusions and the considerations about that work.

Chapter 2

Theoretical Framework of the Study

2.1 Mapping and Localization

The data science goes into collecting, storing, cleaning, and accessing the data, which correspond to database engineering and management. Part of the feedback about the questions "what is the problem that needs to be solved?" and "which dataset is disposed of to solve the problem?", the goal is to identify the right data to use to answer the question of the problem, the data-driven inquiry.

In robotic tasks, to act upon a specific environment, one must first know it. When it comes to robot-related tasks, a device is moving through the environment and may act upon it. First, it is required to consider the state, which means to know the robot's pose and the quantity of the state (the set of all mensurable magnitudes about the environment at that time), usually from sensors.

To know where the robot is and to map the world from distance measurements regarding landmarks of a referred environment can be understood in the localization process. The localization activity is investigated and ordered in several project areas, such as architecture, aviation, local construction-related activities, and civil security.

Primarily, it is essential to highlight that any environment contains characteristics. To map, it is performed at a stage known as characteristic extraction, achieving it by sensors interacting with the environment and producing structure observations. Sensor interactive activities can deliver info to the user as the six degrees of freedom (6-DoF) of the body, distance to a flat board, or the room temperature.

Furthermore, the environments can be generalized into indoors and outdoors scenes, and each background can be sub-denominated as a dynamic environment when there are moving objects in the scene, or a static environment, when all objects are fixed as well.

Figure 2.1, image extracted from [LaSER 2019], shows an indoor environment which, *a priori*, is static, and depending on interaction with outsiders, that can become dynamic with the behavior of more objects or human interaction. Figure 2.2, extracted from King's College dataset, created and presented by [Kendall, Grimes and Cipolla 2015], also used in [Kendall, Cipolla and Feb 2016; Kendall and Cipolla 2017], the activity has been executed upon the outdoor environment, an uncontrolled and dynamic area, with the presence of cars and people. In these cases, mapping becomes complicated and challenging, mainly due to the nature of being heterogeneous environments, uncontrolled illumination, and conceivable plan variations.

Figure 2.1: Static Indoor Environment. An environment is static if only the actions of an agent modify it. Image extracted from [LaSER 2019].

The ability to interpret information obtained through sensors (denominated by features extraction) and storage it can accumulate association errors and provide minimal data on the visited scene. Imprecision under the information collected by sensors and actuators, high model dimensions applied to describe the environment, the environment complexity, and correspondence between environment fields are sort of examples that make it challenging to map and to locate.

It addresses us to conclude that locating activity is focused on getting the observer pose

Figure 2.2: Dynamic Outdoor Environment. It is dynamic if other processes are operating on it. Image extracted from [Kendall, Grimes and Cipolla 2015].

acknowledgement in an addressed map by sensors data collection, submitting it to counts estimation based on previous information.

2.1.1 SLAM Method

The map task depends on three main questions, which are the environment nature examined, the robot's processing capability and the type of information that the sensor can provide, given by cameras RGB, sonar sensor, laser, GPS, and each one can work together or separately.

Looking for methods to construct a set of representations of the environment from information collected by sensors, SLAM (Simultaneous Localization and Mapping) proposes being a system which has as its primary purpose collecting data follow store it in control system memory [Thrun, Burgard and Fox 2005]. The SLAM advantage method is the prospect of navigating in an unknown environment, perform the mapping, and instantly determine the corresponding robot pose into the environment. SLAM excludes any *a priori* topological environment information or artificial infrastructure to estimate the location of the landmarks.

Each sensor that collects information from the environment can run apart and distinctly. Despite, the sensors are all subject to faults that may arise from environment interference

or by the sensor's internal system. Hence, the robot pose estimation can be affected and address errors. Further, the SLAM faces up the problem of associate observations stored at *t* moment with a later $t + n$ observation (being n future instant), to data association [Geiger, Lenz and Urtasun 2012]. Data association is essential for sensor measurements, and even if the capture of a target occurs at different angles, the association to recognition points must be established during the entire course.

In the data association task, establishing landmarks[1] is crucial to measure the dimensionality of objects to describe the scene (designate a corridor, a room, or a door). Whilst representing a two-dimensional environment, the landmarks number can reach a thousand, whereas representing a three-dimensional environment can reach millions. In Figure 2.3, the image extracted from[LaSER 2019], we extract and select sort landmarks (small circles in green color) selected.

Figure 2.3: The bi-dimensional image captured by the TurtleBot inside the LaSER lab shown landmarks points (green color) into a selected region. A data association is an assignment of observations to landmarks selected that can be points of interest, also called assignment problems. Image extracted from [LaSER 2019].

It is intrinsic to the dimension relationship of objects to map the detail level and format of stored pieces of information, which can lead to allocation problems in-memory systems

[1]Landmark is a sort of reference point used by the robot sensor for orientation and estimation data.

at the data processing step. The SLAM query is formulated in a sensor-based structure, and it modifies the underlying system structure variable in linear time.

Regarding those impasses, researchers have sought alternatives, turning their efforts into other methods and techniques to locate. One of these alternatives is the Artificial Neural Network (ANN), which has gained strength mainly due to the increase in the mass of available data and implementation uses of GPU.

2.2 Artificial Neural Networks (ANN)

ANNs are mathematical models inspired by human neural structures (or intelligent beings), acquiring learning through experience, that has been happening over decades. The main concept is to provide the ability to a computer program to adapt its execution based on the newly acquired information, defined to learn features, predicting results, or performing a particular action based on knowledge obtained from the data and performing measures to reach the goal.

The main building piece of a neural network is the neuron, which is an operating unit that we can look up as a node with input and outcome value. It means that node has an input signal, let it be called u, that goes into the node and is submitted to a mathematical procedure to give us an output y, which can be a signal multiplied by a constant, or addition formula, or some complex mathematical proposition, producing a outcome that is correlated with the target response.

The ANN model operation proposed by McCullock and Pitts [Mcculloch and Pitts 1943], can be summarized in four steps as follows and shown in Figure 2.4:

1. The first step is the data that will be used by the network as input to reach the outcomes proposition. The input data can be partitioned texts, image pixels, a vector of features or any data attached (In Figure 2.4, the inputs are **input 1, input 2, ..., input n**, being n the last data input);

2. Each input data is connected with a neuron or a group of neurons (all neurons on the layer or selected portion), called neuron layer, by synapses connections. Also, these connections called synapses will link any neuron layers with the next neuron layers

subsequent or last neuron layer before the prediction result. Each synapse also is called 'weights,' and it connects the data to a weighted sum function. The weights are represented in Figure 2.4 as **w1, w2, ..., wn**;

3. The information runs all network sequentially, neuron by neuron, connection by connection, coming from the first neuron until last one. The processing of information occurs into the neuron, that meant of a **weighted sum** $\sum$ to operate and to produce a level of activity by a function f;

4. The sum is suited to an f unity that produces an outcome **y**. The **y** value generated can be used as input to the next neuron layer or as the neural network output target.

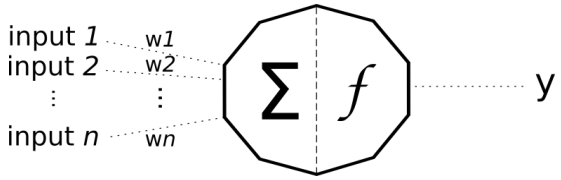


Figure 2.4: An ANN has units called neurons that can be a data input sequence (**input 1, input 2, ..., input n**) or the output of other neurons as well, and these neurons are connected by synapses (**w1, w2, ..., wn**), which are weighted values connections. Inside neurons, a function will perform a sort of sum $\sum$ calculation in *f*. For example, SpaceYNet uses the Rectified Linear Unit function (ReLU) function in full of the neural network; it means the function will return 0 value if input function receives any negative value and, for any positive value x, it returns that value. ReLU can be written mathematically as $f(x) = max(0, x)$.

The neuron outcome can be applied as input data to another neuron, stacking that. Stacking neurons layer by layer through sequential processing, creates structures of a neural network, and it happens a deep structure forming, named Deep Neural Network (DNN). Inspecting by a macrostructure, DNN is composed of three elemental groups (Figure 2.4):

- Input layer: is where the raw data connects to the first neuron layer as input. For example, getting each image pixel from a single input; all contents are organized as a vector and used as input data;

- Hidden layer: it does the processing of the raw data inserted on the previously input layer, performing computations and extracting features from that;

- Output layer: will produce the outcome response of all sequential process executed by the neural network, and it can be a specific class prediction, if the task is a classification problem definition, or an approximation value, a regression task.

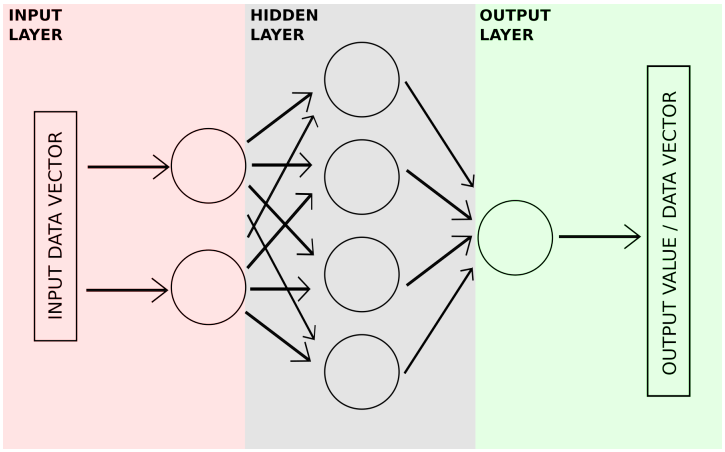


Figure 2.5: DNN is divided into three parts: input, hidden e output layers. The input layer receives the raw vector data that is representative and signifies specific information and will be used to train the network. The hidden layer commonly consists of a large number of neurons, neurons layers, and synaptic connections, processing the data through mathematical functions to produce output in each neuron. The output layer can be an unique categorical value or set of values that unifyly produce an output value.

2.2.1 The Dataset Importance

To achieve the human problem tasks using machine learning projects by neural networks, a crucial step is getting up with a good set of features to train the network, moving us to the question: "how can we use the data to predict the future?"; that involves the meaning of the data confidentiality in real scenarios to make decisions. Moreover, to collect the data one oughts to know if this data is acceptable, it is not redundant data, not adding value by just collecting more of it, being a smart data collected that have a targeted function to do, goes through the question: "how many examples is gave to the machine to learn (one or more than one million)?".

The process of getting the right data, extracting the patterns from these is described by a feature of engineering, that involves picking the most valuable features to train on among

existing features, linking them to assemble a further useful one, and if necessary create new features by collecting new data.

Nonetheless, being careful with the dataset's selection can avoid several issues inside the network processes such as an overgeneralizing or overfitting, avoiding the tuned model being very high accuracy on training data set, but do poorly on the unseen example, or the other extreme different result problem. However, it is important to mention that in some applications overfitting is not necessarily an issue or something that one needs to be concerned about, because the past data is very representative of the future. Complex models can detect secondary patterns in the data, but if the training set is noisy or small, then the model is likely to catch unexpected patterns in the noise signal itself.

Overfitting may occur when the model is overly complex relative to the amount. Consider an online-made test (shown in Figure 2.6), which captured two images (left and central images) corresponding to two specific robot poses (yellow and red color points on the map) in an indoor laboratory area, and the blue point was predicted by the network (the last right image). It should be noted that the prediction was affected, predicting wrong, and may it happens by the overfitting problem.

Figure 2.6: Network prediction illustration. The two first images are specific positions in an indoor laboratory, labeled to the position yellow and red on step 1-LAB. To the network prediction illustration, the pose predicted is the blue point on the map. Images extracted from [LaSER 2019].

Overfitting can occur for different reasons. Highlighting some of them, Figure 2.7 shows two graphs with two distinct curves. The axes on the graph are composed of a horizontal axis that defines the potential reason to cause overfitting and a vertical axis, expressing the behavior of error prediction value.

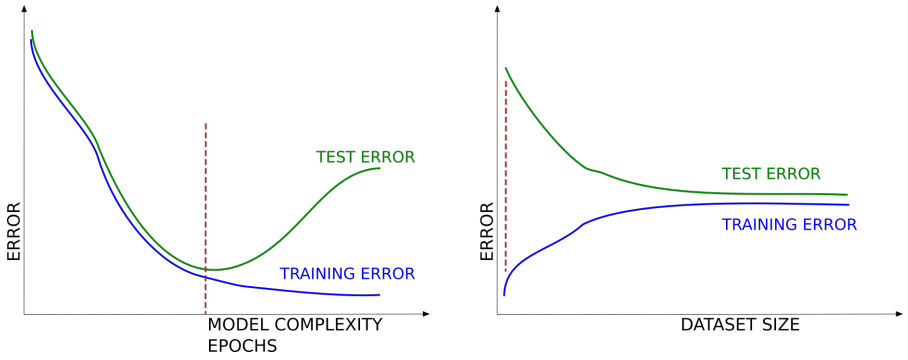


Figure 2.7: Overfitting loss curve example.

The left corner curve graph shows two potential candidates to produce the overfitting: model complexity and epochs (period size training). The first cause is under a trade-off dilemma among a high bias on the network, meaning the simplest model, with a complexity model quite settled to perform the data. The second cause depends on the time of the model to train the data, which can be resumed to the network starts to fit noises in training data and not the signal and features anymore. On the specific moment, the error to predict stops decreasing together in training and test steps, no more converging but diverging - the goal is to stop the train before that wreck, on the red line.

The right corner curve graph shows that the overfitting may happen because the size of the dataset is not sufficiently diverse, and is not enough for the network to learn to generalize the existent features on that. The solution is to expand the dataset numerically, growing by hand as getting extra data and appending to the original dataset, or implementing some of the methods to increase the dataset as, for example, data-augmentation, that takes some data from the original dataset and apply several manipulations, generating new data.

Other methods can be used to turn out the overfitting, such as insert dropout layers to decrease the upper-fit by the network avoiding complex adaptations on the dataset; use regularization that adds information avoiding extra weight adjustments.

2.2.2 Training Categories

As presented, a volume of data can be processed and managed by a neural network, and it is demanded in many varieties of application processing tasks. Machine learning has been used in several areas, including robotics (as the current work presented SpaceYNet, the

state of the art PoseNet and ContextualNet) and outside robotics domain, as, for instance, recommendation systems like Spotify and Netflix services, categorization of ad genre (sports, politics, economy), automatic selection spams, and in the manufacturing process to predict which mechanical tools to use in the step process as well.

Approaching using deep learning is coming to be larger and diverse and, in the same way, it is necessary to understand which machine learning technique framework is going used to apply models based on data. The essential key to this subject is to know what is necessary for the project work. By the answer to the previous question, there are several ways to the neural network to be trained based on the approach that will be taken. One way to ANNs learn the meaning of an object is to label that specific object and shows to the ANN several examples of the same object; this training category is known as supervised training.

Briefly describing it by a simple illustration, consider step **1 - LAB** into the white square in Figure 2.8 as a laboratory area map. Inside this lab, a terrestrial robot is performing a random path route (shown in step **2 - TRAIN**, which the start-point is the red arrow, and the end-point is the blue arrow), taking image shots about the environment and, for each image, it is getting the correspondent pose label. The reason for that it was provided to the ANN enough raw data, being used by the network to train and to predict the robot position using a unique pose image captured.

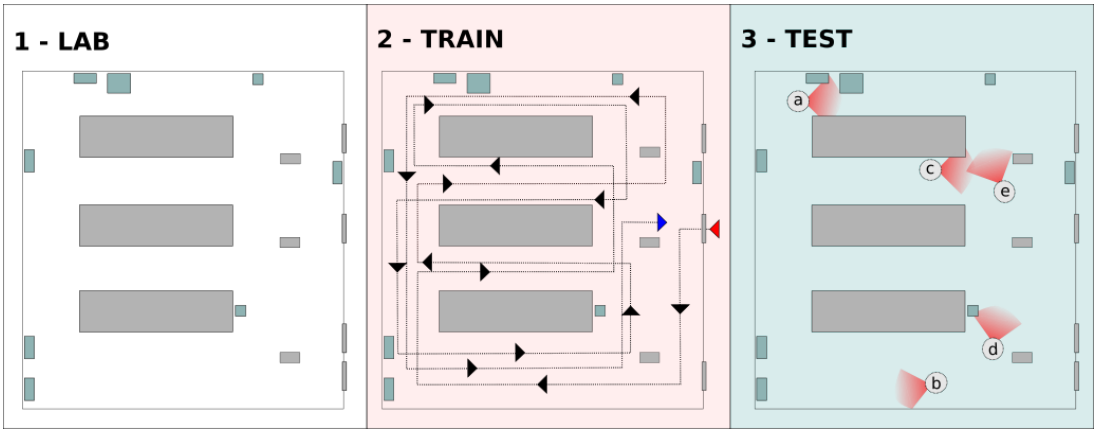


Figure 2.8: Artificial Neural Network macro structure.

The task on our example is to know the pose of the robot when requested, randomly or not inside the lab. The pose is the target outcome that happened. For example, to request the pose points [*a,b,c,d,e*] write down on step **3 - TEST** Figure 2.8).

In general terms, the ANN training process comprises the raw data set, process it, and extracts the features from it. The training process occurs in categories, through which we can extract two convenient categories: supervised (that best fits with the example given earlier) and unsupervised training[2]. Based on the type of data available and the target research task in question, it is chosen to train an algorithm using a specific learning model.

2.2.2.1 Supervised

The supervised learning algorithm is characterized by demands to provide features and labels from the input data. Supervised learning consists on adjusting the weights and thresholds (specific minimum value) of the connections units to get a desired expected categorical value.

Consider the last illustration that regresses the robot pose from an input image, it means the pixels are the network input. Take a specific pixel p on the coordinate (p_x^n, p_y^n), where n is the n-image on the time sequence path-route and x and y two steady coordinates. The pixel on the coordinate position (x, y) at the first image might not have the same pixel intensity on the position coordinate at the last image on the path route, turning the problem hard to solve by using just one pixel as data to get features.

However, by taking all pixels from each image, stacking them as a group g_n (n is the image into the time sequence path route), it is possible to compare image by image, group g by group g, pixel by pixel. The network makes correspondences within the extraction of features, from the group of pixels and their coordinates, and then labeling pose.

2.2.2.2 Unsupervised

On the other side, the unsupervised learning algorithm is applied to approaching tasks without earlier knowledge of the result that should be expected or infer patterns from a dataset without labeled reference. In this category of training, the extraction of features is made from the input data.

If the task manager believes that the data is discrete in some sense, the outcomes may be arranged based on associations linking the variables in the data. It implies that the unsupervised learning assignments can explore handling data clusters into combinations by similarity. For example, using new input images set (Figure 2.9) as sequence input to the

[2]In addition to these methodologies are also relevant the semi-supervised and reinforcement learning.

neural network. There are no labeling images toward the places. That cannot precisely classify which one is a specific place of the building, but resembling at the pictures the neural network can associate some of them and separate them by three groups (Figure 2.10)[3].

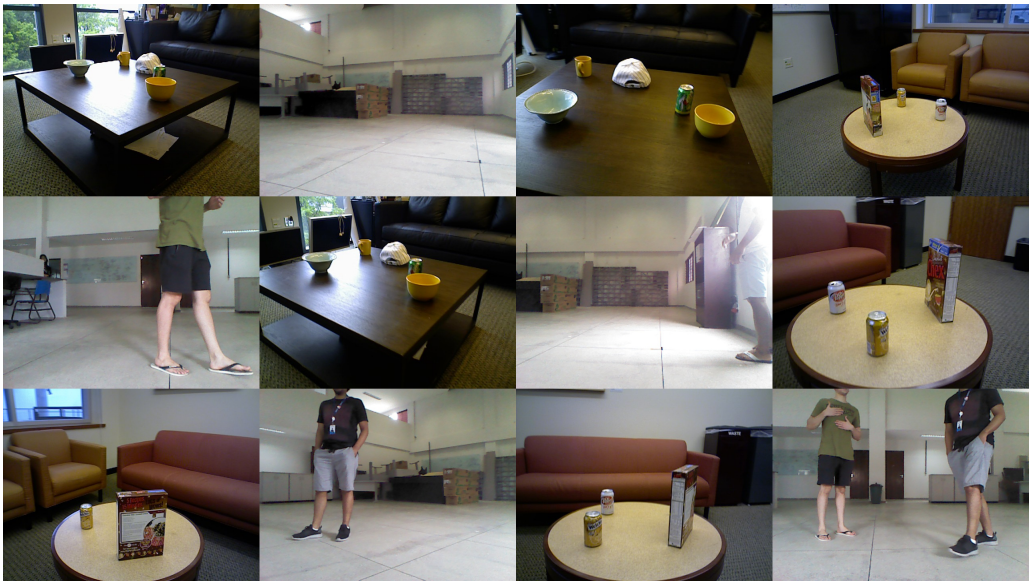


Figure 2.9: Each image will be a input to the unsupervised neural network. Images extracted from [LaSER 2019] and [Lai, Bo and Fox 2014].

2.2.3 Evaluation Metrics

Depending on some factors such as if the algorithm works well on small or large datasets, being fast on instances and performant, to measure those above that are qualitative and quantitative properties applying analytical metrics criteria.

To regression task as the robot pose on the map (unlike categorical values in classification tasks) may assume accuracy or recall, but that sort of metrics are not helpful, because they are not meant for continuous values.

Additionally, the standard evaluation metrics for regression are: Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). To make a selection concerning which of them to use, it is necessary to first understand how each one operates and their properties.

[3]That image was split by the hand into three possible groups. Considering that cluster job was done by the hand, there is the possibility of the cluster job be different in fact when done by the network.

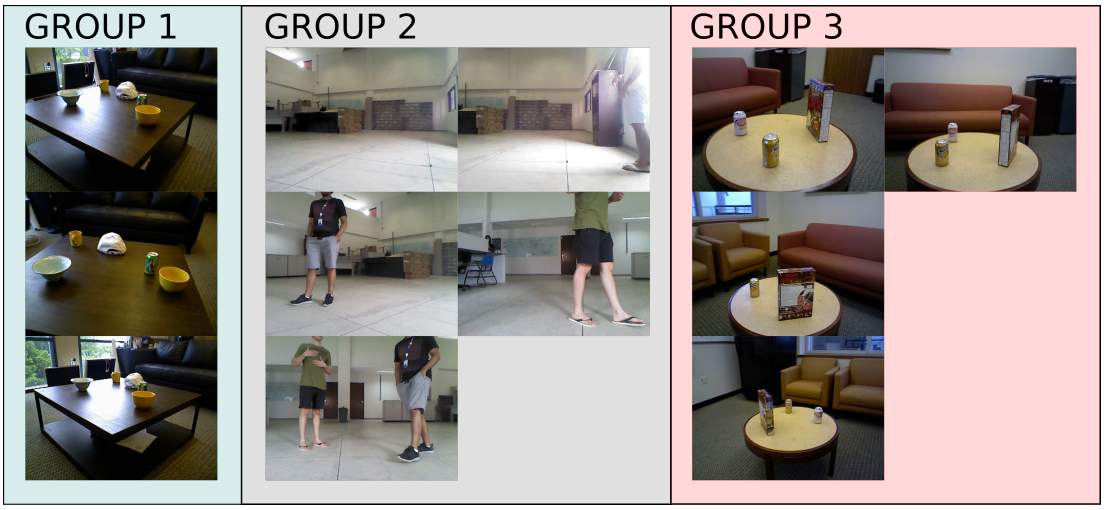


Figure 2.10: Image data set used as input for pseudo unsupervised neural network examples, clustering that by similarity. The point may a truly neural network not split that set as shown. Images extracted from [LaSER 2019] and [Lai, Bo and Fox 2014].

2.2.3.1 Mean Absolute Error (MAE)

The MAE evaluation metric is defined by:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.1}$$

The primary information about it is that it is necessary to take into it the average error between all estimated error. On 2.1, the left corner uses the sum of all n-estimation prediction and divide it by n, and meaning takes the average. The right corner produces the absolute value of the real value subtracted by the predictive value of each prediction. The purpose of producing the absolute value is that the regression can be fit and regress values below or above the correct value.

For example, to predict the sign power sent by the robot sensor does not make sense if it is a negative value. In this case, the lower value that is plausible is 0, meaning not detected sign.

Another case is working image pixels as input data to the neural network. Getting a pixel and trying to predict the level of one channel color, which couple spatial coordinates are (x,y) at the image (row and column of the pixel at corresponding image). Practicing only to

the blue channel (over RGB image), the MAE estimates:

$$\frac{1}{XY}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}|pixel_{x,y}^{B} - \hat{pixel}_{x,y}^{B}| \tag{2.2}$$

where $pixel_{x,y}^{B}$ is the level of the color channel blue on that pixel and (x,y) the coordinates position of the pixel (MAE values range from 0 to 255).

2.2.3.2 Mean Square Error (MSE)

The MSE evaluation metric is defined by:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.3}$$

The MSE evaluation remains the mean of squared errors. Being the computed difference between the predicted value and the real value over meaningful, it applies square on that difference, extra punishment. It means a quadratic value among the output value and real value. Applying again to the blue channel (over RGB image), the MSE estimates it by the function:

$$\frac{1}{XY}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}(pixel_{x,y}^{B} - \hat{pixel}_{x,y}^{B})^2 \tag{2.4}$$

where $pixel_{x,y}^{B}$ is the level of the color channel blue on that pixel and (x,y) the coordinates position of the pixel. The worst MSE value that can obtain is equal to 255^2.

For example, pick out a pixel with an intensity value 100 on the blue channel, and the predicted pixel value by the neural network is 110. From MAE deviation is $|100 - 110| = |-10| = 10$ instead of the MSE deviation $(100 - 110)^2 = -10^2 = 100$. For accumulated distances (not using only one instance, but full dataset), the MSE accumulation distance is much higher than in MAE.

2.2.3.3 Root Mean Square Error (RMSE)

Lastly, the RMSE evaluation metric is defined by:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.5}$$

The RMSE is quite similar to MSE. It needs to add the root to the value of the accumulated distance. The root applied to the square error modifies the value, being close to MAE.

The advantage is that the RMSE might be helpful when significant errors are particularly undesirable, that means penalizing more being more appropriate in particular cases.

The mean of the absolute error value (RMSE) is lower than the mean of the squared error value (MSE), close to MAE, but the penalty is a bit higher and inferior to MSE.

2.2.4 Convolutional Neural Network (CNN)

It was briefly presented, in the previous topic, how artificial neural networks operate, how to evaluate its efficiency and mathematical evaluation metrics to decrease the error in predicting an outcome, and the importance of obtaining a right dataset to be used to train the network.

Regarding the numerous tasks that can be explored as recognizing the human feel expression from phrases or topics as determining if a robot is in or out lab from images input, a standard model of neural network regularly used to tasks like that are the Convolutional Neural Networks (CNNs). CNN works extensively with translation-invariant as manipulation and extraction of patterns from an image, text classification, or extracting features as well.

CNN happens by a mathematical function mask sliding it across an input data involved, making local computations in local patches. For instance, adopt a random image as input data. Then, let us denote the image as a matrix tensor representation because, instead of the image, the computer recognizes the value of pixels. The mathematical function processes the formulate proposition with the data, and that process will generate newer three-dimensional data (may not significantly be same dimension on the input data).

Figure 2.11 displays the steps of the convolution method. The first step (IMAGE RGB - MATRIX 3D) is to choose up the RGB image as input, and separate it in depth dimensions (Images color usually have three channels (blue-green-red channels) and turn it into pixel values for each channel.

After, consider a filter (known by mask) that has a shape h x w x d (presented in the **CONVOLUTION PROCESS** step). The filter will slide through the image, extracting features and generating an output with the dimensions **h-f**$_h$**+1** x **w-f**$_w$**+1**. In this case, we are using the Sobel filter[4] as an example to extract vertical edges on the image. The result will

[4]Sobel filter is a detection edge algorithm used to find vertical and horizontal lines consisting of the process of finite differences, approximating the pixels intensity gradient from an image.

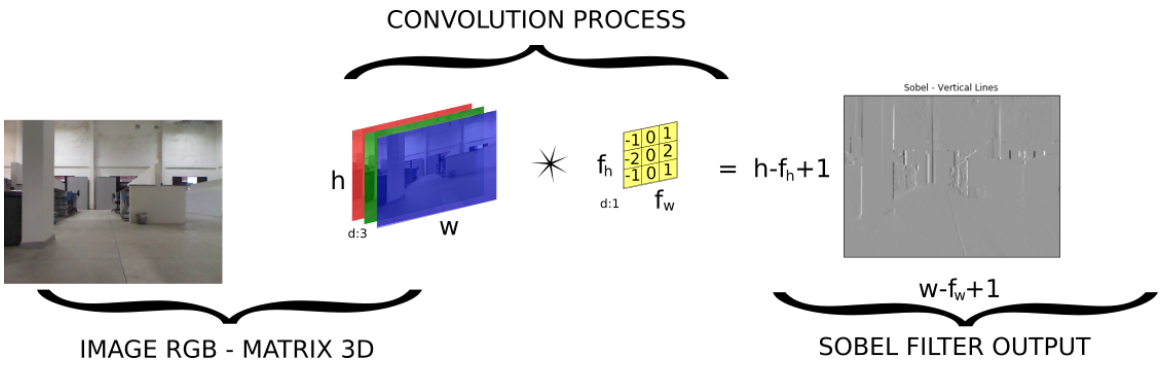


Figure 2.11: Convolution Method applied in an image RGB.

be another image in grayscale with vertical edges superimposed, presented in Figure 2.11 as **SOBEL FILTER OUTPUT**.

The process is called convolution because the filter will slide through each group of pixels and extract features, as presented in Figure 2.12. For each pixel group, the mask is applied and a new value is generated. The new values that are generated are considered to a new matrix of features.

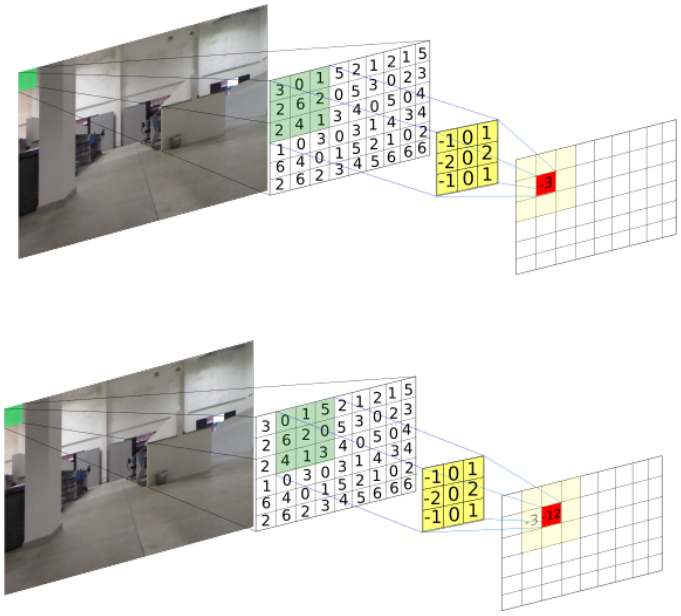


Figure 2.12: Example of convolution method applied to an image using the Sobel filter.

Adopting convolution method means to submit an *n*-dimensional set of tensors to a standard features called filter, which will extract feature representations from the image, maintaining relation between the pixels. At this stage, we order this step as the image 'feature extraction.' Each newer matrix returned will be submitted to a next select group of neurons.

Each neuron has a mathematical function design, as shown in Section 2.2, to learn patterns from images submitted.

2.2.4.1 GoogLeNet 1V1

The use of CNNs provide multiple benefits, such as the ability to generate and to learn representations of a two-dimensional image. That allows learning the state and scale in irregular structures in the data. Because of this advantage, in 2014, Google researchers explored it and proposed the GoogLeNet network [Szegedy *et al.* 2015], obtaining an error rate of 6.67% in the *ImageNet Large Scale Visual Recognition Challenge 2014* (ILSVRC2014), a hit of 93.33% accuracy in object classification. The challenge consists of predicting many refined objects categories and distinctively categories of objects. At that time of the competition, the result was a feat considered impressive.

Since then, GoogLeNet has become an essential performer in convolution-related research. The general structure of the network model can be divided into the trunk, inception modules, auxiliary classifiers, and output classifier.

Moreover, the idea about inception modules is that it is a design of good local network topology (a set of different operations that represents local network) composed by convolutions and pooling, grouped in varied scales, to operate in parallel and the result of these nodes are concatenated at the end, as shown in Figure 2.13. The red blocks act as a bottleneck that permits non-linearities and fewer parameters in inception models. The max-pooling layer summarizes the content of the previous layer.

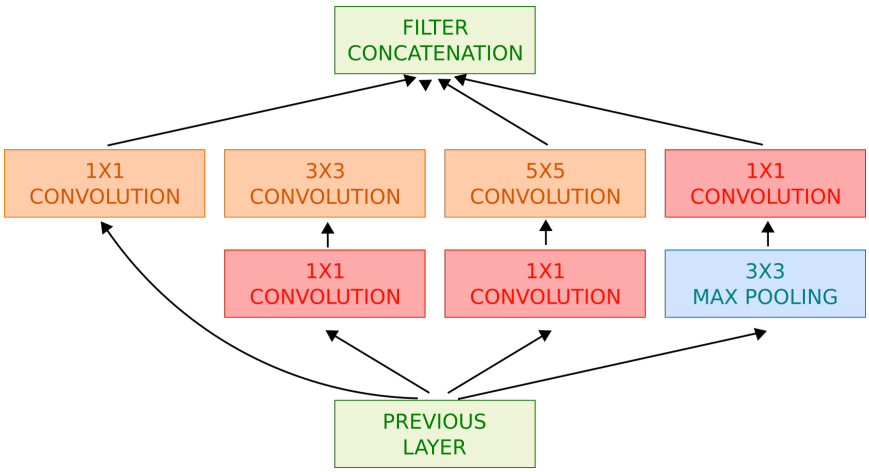


Figure 2.13: Inception Module in GoogLeNet 1V1.

In GoogLeNet, these local network topologies are stacked one on top of each other, nine times sequentially. This type of approach results in a high-performance model and the concatenation module allows the model to take advantage of multi-level feature extraction.

2.2.4.2 PoseNet

Based on state of the art GoogLeNet 1V1, Kendall, Grimes, and Cipolla proposed the network PoseNet [Kendall, Grimes and Cipolla 2015] (Figure 2.14) to handle inception convolutions making a deep learning framework for passive SLAM, a network can learn to regress the camera pose in an end-to-end supervised manner. PoseNet takes advantage of a trained model that needs a low memory footprint and constant runtime at inference. Also, it transfers learning-enabled practical training on commonly used medium-sized datasets.

Besides, PoseNet architecture is changing the end of the network, replacing by classifiers for Cartesian coordinate axes (X Y Z) and orientation of the quaternion vector (W P Q R), generating the pose $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ with camera pose 3D, $\mathbf{p} \in \Re^3$ and quaternion, $\mathbf{q} \in \Re^4$. The framework can relocalize the robot in medium-scale environments in order with a mono-camera operation.

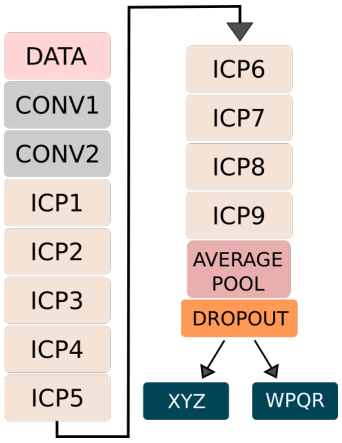


Figure 2.14: PoseNet based on GoogLeNet 1V1. DATA box is the input data to the network; the CONV box are independent convolutions applied to the initial data; the ICP box are the Inception Layers inserted at the network; AVERAGE POOL box is the Average Pool application upon outcome from the last Inception Layer; DROPOUT box represents the Bayesian method to turn the overfitting problem before predicting the robot pose 6-DoF; XYZ and WPQR box are the prediction layers make by the network.

The Stochastic Gradient Descent (SGD) is used by PoseNet to train, looking to minimize the pose loss function given by $\mathcal{L}_\beta(I_c) = \mathcal{L}_x(I_c) + \beta\mathcal{L}_q(I_c) = \|x - \hat{x}\|_2 + \beta \left\| q - \frac{\hat{q}}{\|\hat{q}\|_2} \right\|_2$ where $\mathcal{L}_x$ gives the position axes loss, $\mathcal{L}_q$ the quaternion loss, and β is a scaling term balancing between the two losses. The SGD algorithm's importance is regularly implemented to handle linear systems with missing data, addressing the issue of missing and large-scale data concurrently.

Also, in [Kendall, Grimes and Cipolla 2015] is applied an automated method of labeling data using Structure from Motion (SfM), related in Photogrammetry to generate the pose of a camera from an image of the scene, to large sets of camera regression data and applied the pre-trained learning transfer method as a classifier in large data recognition sets image, converging to a smaller error in less time, even with a very sparse training set, compared to zero training.

2.2.5 Recurrent Neural Network (RNN)

As occur in CNNs, researchers faced significant problems, concerned with what happens in a sequential space-time. For example, if a robot is moving out of a room and takes a shot image in front of an opened door, it is plausible to affirm the robot is going out, walk directly, and towards the opened door. However, without information about the robot's movement, we may also indicate the robot is walking back, going into the room, but looking at the door.

Another classic example is a natural language processing application, with words being the network entries. If words are considered isolated, it means to understand the contextual sentence by only one word, and will be not easy to determine. In contrast, when words are used together, knowing the order in which each of them occurred, it is possible to understand the text.

The central problem is that the inputs and outputs may have different patterns so that they can be of distinct dimensions. Each input is a sentence, so it is fair to presume that most of the training examples can be sentences of distinct dimensions. Also, the network does not share the knowledge acquired from different word positions of the text or image into the lab.

In some cases, process a single data is not sufficient to estimate the outcome problem needing previous data analysis. To answer it considering past events, a model that can be select is the Recurrent Neural Networks, which its idea is to estimate the current moment based

on past states, processing data stored as new entries are processed, as shown in Figure 2.15.

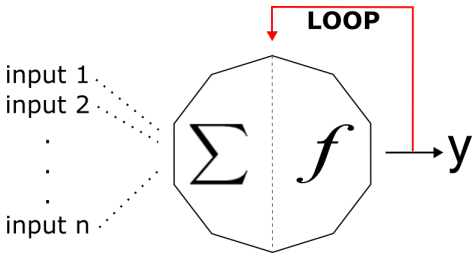


Figure 2.15: RNN cell. The RNN permits networks with loops in them, allowing persist the information.

RNN helps whenever we need the context from the previous input. RNN has become very popular since it helps solving many real-life problems that the industries are facing as it predicts the next word in a phrase, music composition tones selection, image captioning to predict if some act can happen, speech recognition using the context text, and many others.

However, the store states present in some challenges can get up a regular dilemma denominated vanishing. Vanishing happens when *sigmoid* or *tanh* functions are used in RNN neuron units. It implies that the subsequent output nodes of the network are less sensitive to the input at the beginning, when the last knowledge needs to be transferred to the first input if a sequence data is long enough. Facing this problem, in [Hochreiter and Schmidhuber 1997], the problem was solved using the LSTM neural network as an extension to RNN.

2.2.5.1 Long Short Term Memory (LSTM)

The Long Short Term Memory (LSTM) networks are a type of recurrent networks, introduced by Hochreiter and Schmidhuber in [Hochreiter and Schmidhuber 1997], which has been widely used in a variety of current problems, doing dilemmas that happened in previous networks associated with storing information for long periods, and the last information to adjust the network needed to be propagated back.

The solution was the insertion of gates to forget and remember information. To understand how the LSTM works, let us consider Figure 2.16.

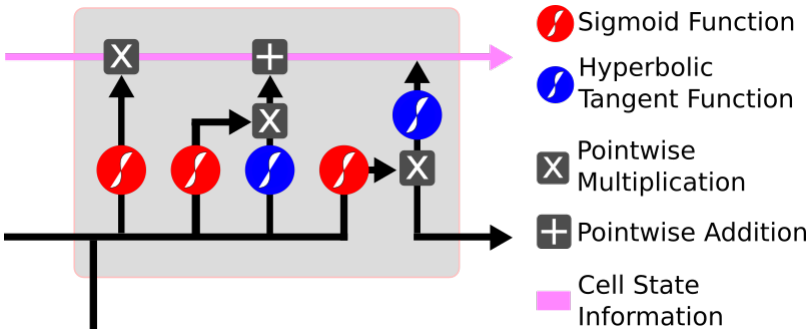


Figure 2.16: LSTM Cell

To store or not the information is decided in the cell state. The cell state is able to transfer relevant information throughout the processing of the sequence, cross the whole thread of interactions, add or remove information to this state cell, and set it according to the structured switches.

The input data comes from two black input lines, on the left side, that are concatenated. The decision to forget or remember data happens through sigmoid functions, represented by the red circles in Figure 2.16. To any value got, f multiplies it by 0, the result is 0, and it means to be 'forgotten.' On the other hand, any number multiplied by 1 results on the previous value inserted, and it means to not forget . At the same moment, a Hyperbolic tangent function (blue circles in Figure 2.16) also receives the input data, and produce outcome values between -1 and 1, helping to regulate the network. The network can learn which data is essential or not, then produce the obliquity output state for input to the state of the cell.

Subsequently, it needs to decide what new information is to be stored, being done in two parts that are combined for a link-state update (the pointwise multiplication). Then, a layer with the hyperbolic tangent function generates new values of information, submitted as storage candidates.

After that, it is necessary to update the value of the old cell state to a new value of cell state. This procedure is performed by multiplying (pointwise multiplication) the old state and adding new values of information (pointwise addition) to be stored.

Finally, we have the decision on which information will be produced based on the state of the cell, composed of a sigmoid function that decides which parts of the cell state to produce. This last step is essential for redundant cases. For example, once the network has already received an image, it will only generate information relevant to the scene until another image

is received.

Using the memory cell and gates to control knowledge progress is a benefit, preventing the gradient in the cell from vanishing too quickly. To dynamical observation problems, a single observation is represented by a tensor $X \in R^P$, where X is the observation, R is observation domain and P measurements. Getting the observation periodically, that gets us a sequence time t of $X_1, X_2, ..., X_t$, accumulating spatial and temporal structures [Shi *et al.* 2015].

2.2.5.2 ContextualNet

To improve the PoseNet outcomes, [Kendall and Cipolla 2017] applied changes on it, originating ContextualNet. This significant contribution was a change in the final classification layer by two LSTM layers added for the temporal analysis.

The first LSTM layer, $LSTM_1$, has 512 hidden cells and receives as input the vector of 2048 data from the connection with the last inception layer of the sequence and a Dense layer. $LSTM_2$ layer has 50 hidden cells and receives the data from $LSTM_1$ output. Figure 2.17 summarizes the architecture of the ContextualNet network.

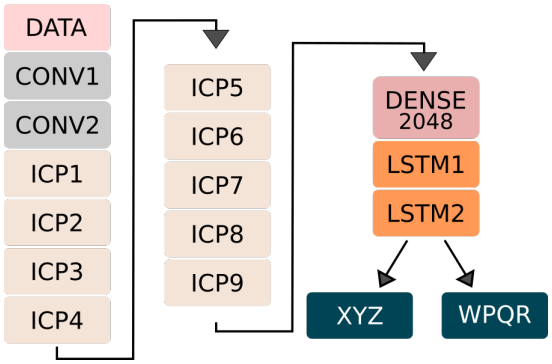


Figure 2.17: ContextualNet short network architecture.

The employment of LSTM layers in ContextualNet fits a nonlinear time-series model, where the nonlinearity is learned from the data, matching the nonlinearities of data, with predictions more reliable than PoseNet.

According to comparisons with PoseNet and ContextualNet, the pose (cumulative distribution for position and orientation) error difference was 14.31m between them, PoseNet presented 17.09m in the cumulative distribution for pose , whereas ContextualNet presented

2.78m. The significant error in PoseNet can mostly be accounted for by the point that there were minimum features accessible [Kendall and Cipolla 2017].

2.2.6 Autoencoders

Previously, we showed works that use supervised learning to flow their work as PoseNet and ContextualNet, based on convolutional and recursive networks to pose regression. Now we are analysing Autoencoders, which is an unsupervised technique that imposes a bottleneck based on minimizing the squared error between an observation, and a non-linear reconstruction in the network [Konda, Memisevic and Krueger 2014], acting as a compressor of the data representation of the input, and subsequent reconstruction of the data compressed.

Dimensionality reduction may act upon reducing scenarios with the problem of feature selection [Tomar *et al.* 2018] by the cost function Euclidean

$$\underset{W,W'}{\operatorname{argmin}} \|X - W'\phi(WX)\|^2 \tag{2.6}$$

where X consists of all training samples stacked, W the link weights from all input nodes and $\phi(WX)$ the output of the hidden nodes, to the reconstruction of error between the data and the decoded map of features, solving a non-convex problem by gradient descendent technique [Tariyal *et al.* 2016].

In Fig. 2.18, an Autoencoder is formed by two paths that are an encoder $h(\circ)$ and a decoder $g(\circ)$, working with the input x at $Re(x) = g(h(x))$, the reconstruction accuracy can be obtained by 2.6 with average reconstruction error $loss(x, Re(x))$ [Cai 2017], as supervised cost by the outcomes and the target.

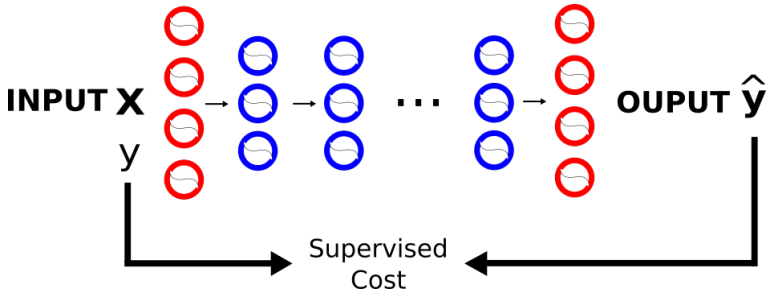


Figure 2.18: Autoencoder process performing gradient descent on a supervised cost, minimizing error in y with a supervised target value.

The Autoencoder can also be used to compact data for the input that the network provides, limiting the number of neurons in hidden layers, consequently reducing the number of features of the original input. Another use for Autoencoders is in dimensionality reduction [Paniagua and Lopez 2019], dealing with problems that bring down the data to a lower-dimensional scope. Currently, Autoencoders are a considerable section adopted in areas such as image classification [Xing, Ma and Yang 2016] or text classification [Wang, Shi and Yeung 2015] as well.

2.2.6.1 U-Net

A CNN network can aim to extract features from an image (or a set of them), but some tasks run in a different direction, getting an image from the features extracted (the features are as the neural network input), being similar to image reconstruction. That method is typically used for dimensionality reduction and to generate a new image, similar to the original target image, ignoring signal noise inside it. It is Autoencoder.

U-Net [Ronneberger, P.Fischer and Brox 2015] is one work that involves Autoencoder and, according to the authors, it is common sense that successful training of deep networks requires thousands of annotated training data samples. It is composed of a symmetric path, where the first symmetry path is destined to capture the context regression of the images, while the second path enables the location of the components. For a better comprehension, Figure 2.19 extracted from [Ronneberger, P.Fischer and Brox 2015], shows the macro architecture of their network.

In the U-Net architecture, the first symmetry has a Rectification with a Linear Unit (ReLU), followed by operations to down-sampling (a Max-Pooling operation 2x2 with strides 2). For each down-sampling step, the number of channel characteristics is doubled from the latter layer. The second symmetric part consists on performing a feature map up-sampling followed by 2x2 up-convolutions, with the corresponding map features concatenations in the first stage of the network, followed by 3x3 convolutions followed by ReLU rectification. Finally, the last layer is composed of convolution 1x1, generating a map of characteristics for final classification in the output.

The network and training approach that relied on the abundant use of data augmentation to use the handy annotated samples further efficient, proving itself valuable because it

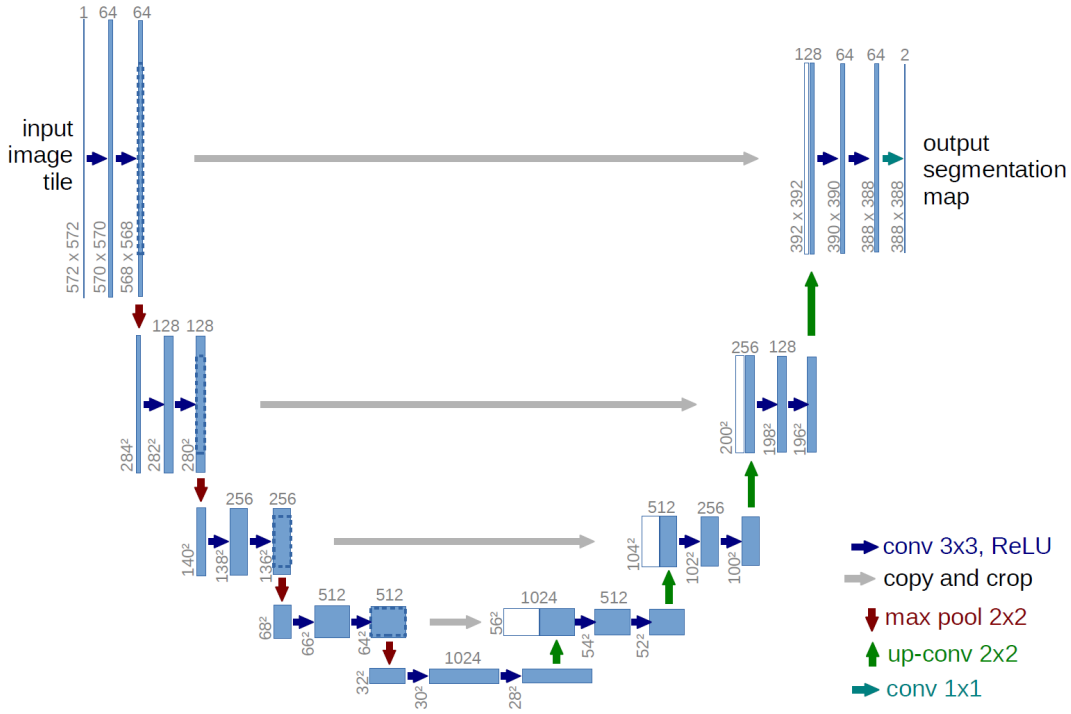


Figure 2.19: U-Net architecture, that is an Autoencoder network for image segmentation. Each blue box corresponds to a multi-channel resource map. The image dimension provided at the bottom-left corner of the box. White boxes represent maps of copied features. The arrows denote the result generated at the tail arrow is sending to concatenates with the operation result on the head arrow. Image extracted from [Ronneberger, P.Fischer and Brox 2015].

achieved a distortion error of 0.0003529 and a similarity error of 0.0382, defeating the completion of the sliding window convolutional network presented by [Ciresan *et al.* 2012], that result obtained is a distortion error of 0.000420 and an error of similarity of 0.0504.

2.3 Final Considerations

This chapter presented fundamental concepts to understand the rest of the work and the experiments performed - the reasons for using neural networks as an alternative to SLAM and odometry.

Convolutional, Recurrence, and Autoencoder networks are essential in this work because it works to extract features from the data obtained by a robot at that moment, verifying the

recurrence of each pixel in the course of time - understanding and predicting the robot pose. Also, combining the convolutional network and recurrent network to robot pose prediction is presented by PoseNet and ContextualNet networks, influencing this research and attaching new functionality by the use of Autoencoders.

Chapter 3

Correlated Works

In this chapter, we will review works relevant to the research, investigating the state of the art, examining what has already been published in the literature, and observing the presented adversities of the technologies.

In Section 3.1, a survey of the study and application of neural networks is conducted. The goal is to understand how to predict the observer pose outside and inside environments, and the depth of objects in an image. These papers are crucial due to the confluence of an effective localization system and the possibility of robot movement in the environment.

3.1 Works related to the use of neural networks for location

The application of Neural Networks has been part of the current study and applications in researches that focus on autonomous vehicles, aerial vehicles, and augmented reality. For a robot to perform its actions, it is vital to be aware of the location and working space.

In [Kendall, Grimes and Cipolla 2015] is proposed a real-time localization system called PoseNet. The system performs a pose regression in 6-DoF utilizing a monocular camera and modifying a part of the GoogLeNet [Szegedy *et al.* 2015]. The algorithm can operate inside or outside environments. Despite the results, the author leaves open the discussion about the lack of use of other methods of multicast geometry as a data source for training in the regression and exploration of probabilistic extensions.

Due to inherent issues, the author applies a Bayesian Convolutional NN using dropout to regularize the network and avoid overfitting in estimating the metric relocation er-

ror [Kendall, Cipolla and Feb 2016]. Besides, the question of uncertainties regarding rotational and translational meanings are correlated. Despite obtaining improvements in the results of the algorithm, in the NN training stage, RGB-D cameras were used to capture the depth of the objects in the image and, therefore, the system is not monocular RGB at completeness.

Considering the challenges presented in [Kendall, Grimes and Cipolla 2015; Kendall, Cipolla and Feb 2016], the work [Kendall and Cipolla 2017] proposes to learn robot pose as a fundamental practice to scene geometry, which is achieved through the structure of the movement. The main contribution is the investigation of a series of loss functions, applied to the NN, to learn to regress position and orientation simultaneously with the geometry of the scene.

After, [Patel, Emery and Chen 2018] proposes ContextualNet, which is the PoseNet with incremental network LSTM recurring in the CNN outputs. Using the context provided by the image sequence, more distinguishable features are learned and help challenging cases of observer location prediction by correlating the order of images presented to the neural network. The results obtained in this work have, as its main prerogatives, the reduction of the errors for the position and orientation for the third quartile of the cumulative distribution function.

Motivated by problems such as the fact that these lasers are affected by wrong readings and noise, in [Kuznietsov, Stückler and Leibe 2017] is adopted the supplemental use of 3D lasers to capture depth readings, and semi-supervised training. They used a Convolutional NN to calibrate the known camera, which was necessary to obtain the predicted depth map used to determine the inverse depth $p(x)$ in each pixel $x \in \Omega$ of the RGB I image.

For the extraction characteristics and representations from the images, a series of different approaches are presented, such as SIFT [Mortensen, Deng and Shapiro 2005], Wavelet [Strang and Nguyen 1996], Gabor filter [Nestares *et al.* 1998] and triangulation of the relation between a stereo camera pair and objects contained in the image [Salih and Malik 2012].

To object detection, [Choi, Lee and Zhang 2016] sought to solve the problem of simultaneous face recognition and orientation estimation of the human body in domestic media, using the YOLO algorithm [Redmon *et al.* 2015]. They proposed the lightweight convolu-

tional neural networks, an end to end system, for estimating human body orientation. The body orientation estimation model achieved 81.58% and 94% accuracy with a benchmark dataset and their dataset, respectively.

[Shah *et al.*] presented the FollowNet, an end-to-end differentiable neural architecture for learning multi-modal navigation policies. FollowNet maps natural language directions as well as visual and depth inputs to movement. FollowNet processes instructions using an attention device conditioned on its visual and depth input to focus on the relevant elements of the command while performing the navigation task. The authors evaluated the tool on a dataset of sophisticated natural language movements that led the agent through a rich and realistic dataset of simulated houses. The results obtained were 52% to the accuracy of being able to follow instructions via voice command, which was 30% better than the baseline used for robots of the genre [Shah *et al.*]. However, none of those above set out to predict the pose and depth scene simultaneously.

[Saxena, Chung and Ng 2008] presents a system based on learning the model of Markov Random Fields[1] (MRF) and extracting image characteristics by dividing it into packets and analyzing texture variation, gradients, and coloring. Thus, the MRF model operates in the modeling of the relationship between characteristics and depths, the relation between depths on the same scale and relation between depths at different scales. According to the author, algorithm errors can be attributed to failures or limitations of the training set. Also, laser readings are often incorrect for reflective/transparent objects.

Following the reasoning, what is sought is the possibility of constructing a map of an environment at the same time that it is located. Research with this purpose is also known as Simultaneous Localization and Mapping (SLAM) systems. For a given time, SLAM systems had limitations mainly for monocular applications for absolute scale estimation. Some proposals suggested a solution through the use of combined scene detection with a set of predefined 3D models to retrieve the initial scale based on the estimated object size [Gálvez-

[1] A Markovian process has the following property: the probability that a given event occurs in time n , given all previous and future events, depends only on the events at times $n - 1$ and $n + 1$. With probability modeling, the image becomes a set of variables and, in this way, it is possible to determine the joint distribution of these variables and calculate the probability of the realization that, due to the lack of information about the dependence among the random variables, conclude that only local interactions between the neighbors should be considered, obtaining approximate solutions.

López *et al*. 2016]. However, this alternative failed due to the proposition of the attempt of resolution and to the detriment of the absence of other forms and objects in the scene.

Regarding the problem showed in [Gálvez-López *et al*. 2016], [Tateno *et al*. 2017] proposed to estimate the location and reconstruction of the scene through the use of a set of distinct images, collected as key models, whose pose is subjected to global refinement based on the optimization of graphic representation. At the same time, the camera pose estimation is performed on each input image, estimating the transformation between the image and its closest key model. To maintain a high rate of images and to generate the depth map, CNN used only on key models. Also, an uncertainty map was developed to measure the reliance on pixels of each depth prediction.

About depth estimation, the work [Eigen and Fergus 2014] proposed the use of two convolutional networks in steps to predict the depth map of an image. It makes a global forecast first and then it performs local refinements. Thus, local networks were informed about their place in the global scene and could use this information in their refined predictions. Furthermore, one aspect in common with these works is that these are intended to solve a specific problem,some of them also propose to solve more than one activity.

Finally, there is the significant agreement that the successful practice in deep networks requires several amounts of samples to the training step. However, the work [Ronneberger, P.Fischer and Brox 2015] presented a network and training policy that relies on the data augmentation to use the available interpreted samples efficiently. Their architecture, called U-Net, consisted of a route to take the context and that path turns on an expanded symmetric path, allowing precise localization. The network is trained based on transmitted light microscopy images and was successful in reaching means of 92% and 77.5% IoU (Intersection-Over-Union[2]), respectively. Moreover, the image segmentation in 512x512 takes less than a second on a recent GPU.

[2]Given a set of images, the IoU measure gives the similarity between the predicted region and the ground-truth region for an object present in the set of images

3.2 Final Considerations

This chapter presents some works related to the object of SpaceYNet research. That collection of works were used to set and develop a knowledge basis and is intended to establish the relationship between NNs and pose prediction. We also presented the difficulties encountered for each study and the resolutions that they present.

Also, it is possible to observe that in no case it was possible to raise the resolution assignment of the pose and depth prediction tasks simultaneously as SpaceYNet does. A critical extra point is the fact that the proposal SpaceYNet can be implemented on terrestrial robots and drones, only requiring cameras in the testing steps properly.

Table 3.1 showed the difference between the networks previously introduced. The SpaceYNet fills the hold as the 6-DoF intersections with depth-scene prediction, which can help in occlusion cases, also treating the overfitting problem based on [Kendall, Cipolla and Feb 2016] work, and adopting the use of CNNs and RNNs, as ContextualNet. Because of the targets, to pose prediction can be applied through supervised learning; although to depth-scene prediction it requires to be semi-supervised by the use of Auto-encoder method.

The exploration continued trying to answer all the functionalities of the technologies preceded, aggregating them, producing a further robust and complete system, addressing more purposes.

Besides, it is not possible to mention all available literature on the subjects that are related. Therefore, a selection of contemporary and meaningful research for the development of the work was formed. Thus, one may consider the bibliography provided here as a starting point to understand the subject.

	Position	Quaternion	Depth	Overfitting Treatment	CNN	RNN	Supervised	Semi-supervised
PoseNet	X	X			X		X	
ContextualNet	X	X		X	X	X	X	
*	X	X		X	X		X	
**				X	X	X		X
YOLO		X		X	X			X
FollowNet		X	X		X			X
+			X	X	X		X	
++	X				X		X	
$			X		X		X	
U-Net			X	X	X			X
SpaceYNet	X	X	X	X	X	X	X	X

Table 3.1: Works that set out to solve problems and methods of approaches related to SpaceYNet. *[Kendall, Cipolla and Feb 2016]; **[Kuznietsov, Stückler and Leibe 2017]; +[Saxena, Chung and Ng 2008]; ++[Tateno *et al.* 2017]; $[Eigen and Fergus 2014].

Chapter 4

System Overview

In this chapter, the description of SpaceYNet's basic structure will be approached, considering the frames and extraction of the database related. Next, it will be conducted by an exposition of other investigated architectures. The last section shows how to deploy SpaceYNet's neural network.

4.1 Platform

The network requirement is coded in Python 3.7 style, employing Keras 2.0 framework with a virtual environment by Anaconda 3. Additionally, it will also be necessary to use the OpenCV 3.6 framework to the images preprocessing, before utilizing them as inputs on the network. We perform the training and test steps using a computer with an i9-8750 core, a GeForce RTX 2070, 16 GB RAM, and Ubuntu 19.04 64-bits.

4.1.1 VisualSFM

VisualSFM [Wu 2013] is a GUI application for 3D reconstruction using the Structure from Motion (SFM) method using 2D images. The reconstruction system combines different previous outlines: SIFT on GPU(SiftGPU), Multicore Bundle Adjustment, and Towards Linear-time Incremental Structure from Motion. VisualSFM exploits computational multicore parallelism for feature detection, feature matching, and bundle adjustment resolutions.

The tool was adopted to produce part of the database label, applied to each image, and

returning its position in the scene. By changes, there are selected only ordered labels concerning image axes and quaternions.

The process VisualSFM uses to address obstacles depends on several circumstances, such as the number and type of cameras used and whether the images. For cases where the single calibrated camera took images, the 3D structure and camera movement just need to be rescaled. To apply it in images wanting the real scale of the structure and movement in units of the world, it is necessary additional information such as the size of an object in the scenes and other information from another sensor, such as an odometer.

When selecting a part of the generated file solely, sort information is outcoming from VSfM, that the file filled has for each image the following format

<file name> <focal length> **<quaternion>** **<axis xyz>** <radial distortion> 0

The information presented assumes that each line of data represents a single image, wrote down by file name, and the labels referring to the quaternion and 3D position on the map - each line of information for each image as an input data. These are some examples of outputs produced by the VisualSfM:

```
1  scene_01/00040-color.jpg 568.005981445 0.641615655993 0.123734719404 0.7463792853
       0.126246944243 -72.6731840922 4.90854081422 -28.0129703466 -0.0746247847128 0
2  scene_01/00041-color.jpg 569.11706543 0.642308483609 0.124306054624 0.745719908558
       0.126057084572 -72.7367810836 4.71530520458 -28.1678671558 -0.0731074920603 0
3  scene_01/00038-color.jpg 568.590393066 0.639191304409 0.120231704203 0.748649853667
       0.128471499395 -72.5345137926 4.99573930099 -27.686606615 -0.0755798535132 0
4  scene_01/00037-color.jpg 567.937744141 0.637638529215 0.118858837401 0.749956260914
       0.129834380753 -72.5447135234 5.10166905217 -27.5263776775 -0.0740381809714 0
5  scene_01/00035-color.jpg 566.891662598 0.635404981426 0.115093671756 0.752062919226
       0.131974223109 -72.4613229605 5.2600038172 -27.2925235675 -0.0762192378329 0
```

4.2 Datasets

To train and benchmark the model on several datasets, image rescale is applied such that the shortest size is 256x256. The images are normalized, andinput pixel intensities are ranged by 0 and 1. The labels about the 6-DoF robot pose were standardized to the convergence of the neural network.

We perform the tests using two datasets:

1. A public dataset: RGB-D Scenes Dataset v.2 [Lai, Bo and Fox 2014];

2. LaSER dataset [LaSER 2019].

4.2.1 RGB-D Scenes Dataset v.2

The first dataset is the RGB-D Scenes Dataset v.2, also used in [Souto *et al.* 2017], which is a set of 14 scenes comprising a sum of 11422 image samples, provided with RGB-D image pairs (Fig. 4.1 gives an example extract from [Lai, Bo and Fox 2014]).

Figure 4.1: RGB-D Scenes Dataset v.2. On the left side, an RGB sample image and their correspondent depth image label is the righter figure. Images extracted from [Lai, Bo and Fox 2014].

This dataset is available to the research community and aids in the purpose of different methods for comparative analysis. The RGB-D Scenes Dataset v.2 is created to introduce HMP3D, which is a hierarchical sparse coding method to learn features from 3D point cloud data, trained handling a simulated dataset of virtual scenes generated using CAD models.

The dataset is formed with a selection of furniture images (chair, coffee table, sofa, table) and an object subset in the RGB-D Object Dataset (bowls, cups, cereal boxes, coffee mugs, and soda cans). Each scene is a point cloud generated by aligning a set of video frames using Patch Volumes Mapping [Henry *et al.* 2013]. All of the scene system combines features learned from raw RGB-D images and 3D point clouds directly, without any hand-on-design features.

Besides isolated views of the 300 objects, the RGB-D Object Dataset v.2 also includes 22 annotated video sequences of real scenes. The scenes comprise natural indoor environments including offices, rooms, and open areas. The objects inserted on the scene are visible from

different perspectives and distances as shown in. 4.2 and Fig. 4.3. Experiments on the RGB-D Scenes Dataset v.2 express that the suggested proposal can be used to label indoor scenes.

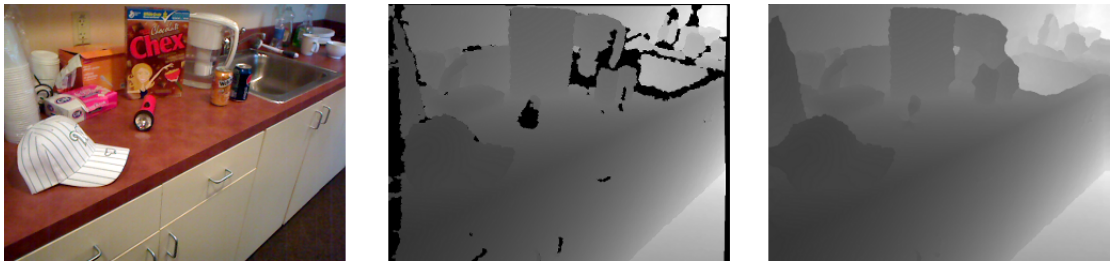


Figure 4.2: RGB-D Scenes Dataset v.2 - RGB image, label depth and labels depth.

Figure 4.3: RGB-D Scenes Dataset v.2 - Scene sparse points reconstruction.

The RGB-D Object Dataset v.2 does not have the pose label, so all dataset is submitted as input to VisualSfM [Wu 2013], as made in [Kendall, Grimes and Cipolla 2015], for camera relocation regression, generating their labels about camera axes and quaternion.

4.2.2 LaSER Dataset

Laboratory of Systems Engineering and Robotics (LaSER) made the second dataset using a Turtlebot 2. The LaSER lab has a dimension of 30x40 square meters, and the images were captured moving the robot inside the lab as shown in Fig. 4.4, using its RGB-D sensor (a Kinect camera) to acquire images and depths.

Figure 4.4: LaSER dataset images sample. Each RGB image sample of the interior of the environment has a respective label depth at the bottom. Both image files are in .png format. All RGB image data have associated the 6-DoF odometry, compressed in a text file.

The first path the Turtlebot's route performed inside the LaSER lab allowed it to acquire 17483 images RGB and 17483 pair images in grayscale, which draws the depth-scene label. The route performed is shown in Fig. 4.5.

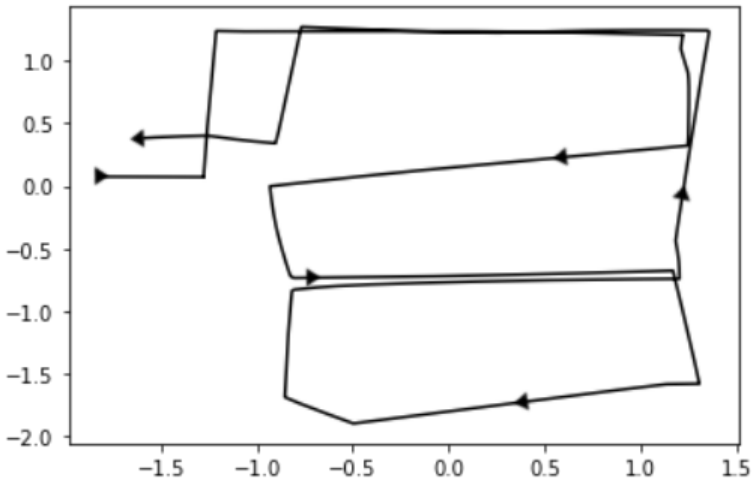


Figure 4.5: Turtlebot's route inside the LaSER laboratory applied to film the package consisting of RGB image samples of the environment with associated depth-scene and 6-DoF labels.

Each RGB has a depth image pair and the label position 6-DoF by well odometry of the

route, which includes the Cartesian pose and Quartenian axes. To terrestrial robots, it is to overcome the Z-axis test step because the robot is always on a flat surface. The odometry label is applied to the *z-score* to be able to train, predict the robot pose, and to converge the accuracy value. The variable amount is settled on the same scale, allowing to compare scores between different variables, so that can do by:

$$z = \frac{x - \mu}{\alpha} \tag{4.1}$$

where x is the singular 6-DoF position value, μ is the mean of the axis, and α is the standard deviation of the axes. Standardization of a dataset is a common requirement for many machine learning estimators: they might act up if the individual features do not look more or less like the normally distributed data (e.g., Gaussian with 0 mean and unit variance).

The dataset created by the LaSER lab provides an extensive robot tour, merging RGB images, depth scenes, and odometry labels, without an unspecified process to dress that.

4.3 SpaceYNet

Fig. 4.6 (SpaceYNet v.1) and Fig. 4.7 (SpaceYNet v.2) shows two proposed network designs. SpaceYNet adopts the network symmetry concept present in U-Net with less than half neurons number and inception layers as presented by GoogLeNet. The approach ended in three outputs: depth estimation and pose (position and orientation) estimation. In the case of SpaceYNet v.2, LSTM layers are added at the end, before prediction, to improve the robot pose regression.

The architecture is in two paths: one to depth-scene regression and other to pose regression. The whole network consists of 43 convolutions with the same paddings and ReLU activation. Also, it needs to be considered that the first ten layers are shared for both purposes (pose and depth-scene regression), eight exclusive to depth-scene prediction, and 25 unique to pose regression (that includes the inception layers), with two LSTM layers connected at the end.

For the depth-scene regression path, it is divided into two symmetry sides, the first one consists of convolutional layers followed by Max-Pooling operation 2x2 with strides 2 for down-sampling. For every down-sampling step, the number of channel characteristics is doubled. The second symmetry side performs a feature map by up-sampling layers, followed by

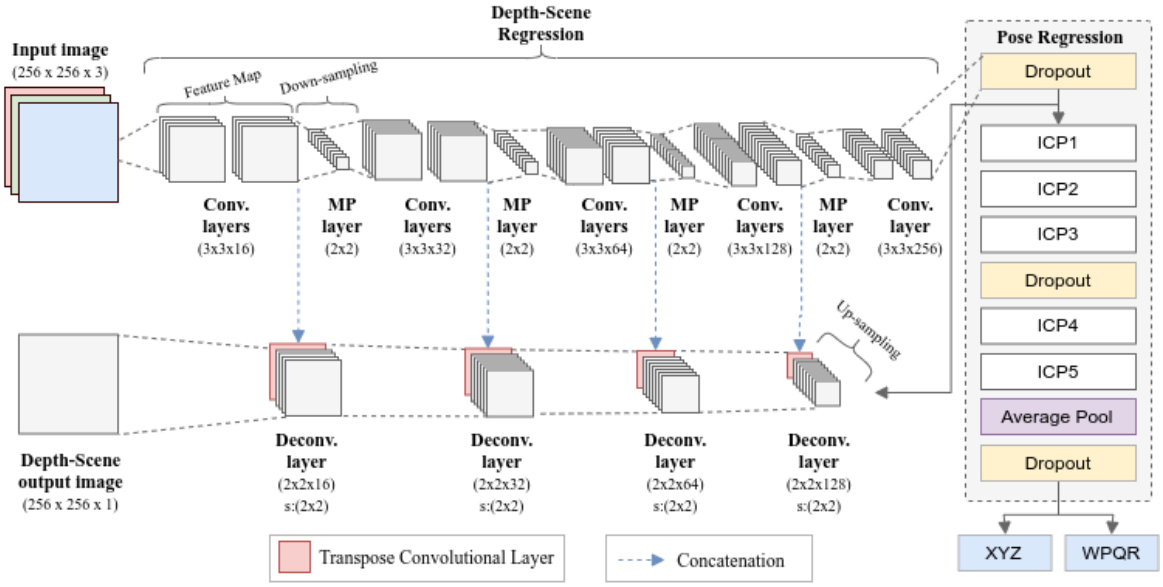


Figure 4.6: Architecture SpaceYNet v.1 proposed that it consists of two main pieces: one designated to the depth-scene regression, and another path to pose regression. To depth-scene regression path, any Inception layer is inserted, working downsampling images first and extracting features and, after, up-sampling to reconstruct based on features acquired. To pose regression, there are used Inception layers to extract more features.

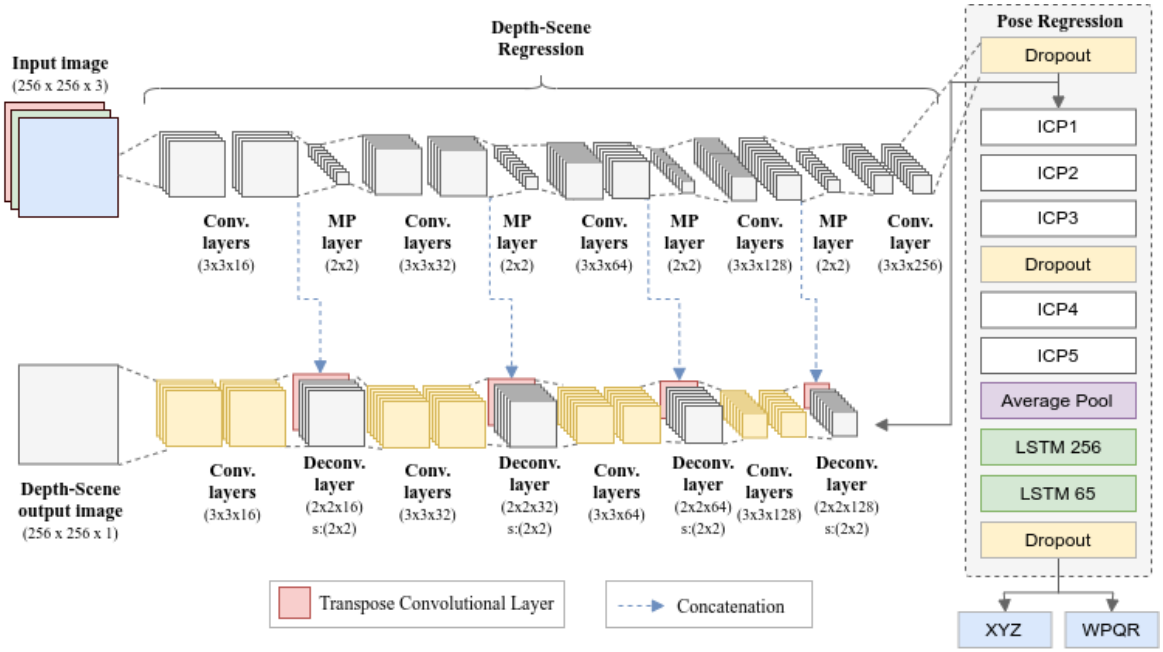


Figure 4.7: Architecture SpaceYNet v.2 proposed that two LSTM layers are inserted at the end, before pose regression.

2x2 up-convolutions concatenated with the corresponding map feature from the first symmetry output, then it is followed by 3x3 convolutions followed by ReLU rectification. Finally, the last layer is comprised of convolution 1x1, generating the depth-scene image output.

Furthermore, the Adam optimization algorithm with a learning rate of 10^{-4}, and the AMSGrad [Phuong and Phong 2019] (the AMSGrad uses a lower learning rate compared to standard Adam, and yet it incorporates the intuition to slowly decrease the effect of past gradients on the learning rate that use the peak of the latest squared gradient to update the parameters to apply in mini-batches that needs higher gradients) will be adopted.

SpaceYNet adopts dropouts that can be interpreted as a Bayesian approximation of a Gaussian probabilistic model, as shown in [Gal and Ghahramani 2016], observing the capture of general information rather than sample characteristics. The dropout regularization value adopted in the path is 30%, which helps to generalize features avoiding neurons to tune their weights considering specific features provided previously. Also, the MaxPooling is used for dimensionality reduction and makes the model more efficient.

4.3.1 Depth-scene Prediction

The first ten shared convolutions are subdivided into four subsequent partitions designed as:

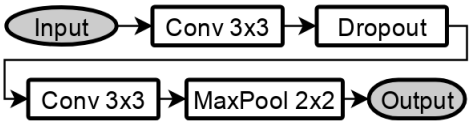


Figure 4.8: First four subsequent partitions.

Following, the next partition is the exception of the path symmetry, thus designed:

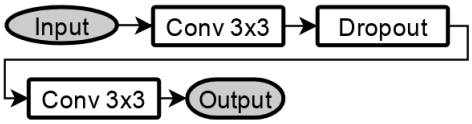


Figure 4.9: Last partition of the first path symmetry.

The last eight convolutions, designated for depth-scene output, can be subdivided into other four partitions, with the following format:

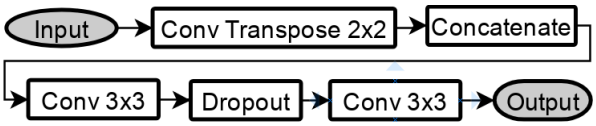


Figure 4.10: The last eight convolution layers, designated for depth-scene output. All presented in the other SpaceYNet path symmetry.

The concatenation occurs between the previous transpose convolution and the output with the same amount of convolution filter of the opposite symmetry. The last convolution, before the output, has the shape 1x1 with non-linear sigmoid in the layer activation. The Cost Function adopted is hereinafter:

$$F(x) = \frac{\alpha}{n} \sum_{i=1}^{n} |d_i - \hat{d}_i|^2 \tag{4.2}$$

where $\hat{d}_i$ is the estimated depth and d_i the fundamental depth of the truth of the image. The term α is used as a weighting factor in the depth loss function to have a similar magnitude with pose outputs. For this reason, that will make our model capable of capturing n-independents variables and mapping its relationship to n-dependents variables.

4.3.2 Pose Prediction

The twenty-five convolutions, unique to pose regression, are composed of five inception layers. At the end, we performed a similar change to PoseNet outputs for pose regression, presented below:

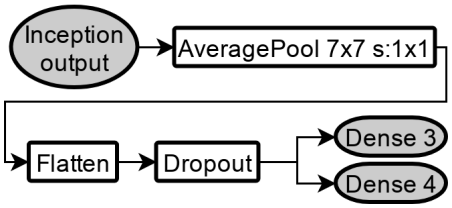


Figure 4.11: It is the last layer composition attached to the fifth Inception layer destined to pose regression - the output composed by two dense outputs that draw the 6-DoF, on end.

Considering that the pose $P = [p, q]$ with the position of the 3D camera $p \in \Re^3$ and quaternion, $q \in \Re^4$ - those [p, q] are the composition of 6-DoF, which refers to the freedom

of movement of a rigid body in a three-dimensional space. The pose was optimized using root on the Euclidean function about to be mentioned:

$$F(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \hat{p}_i)^2} + \sqrt{\frac{1}{\beta.n}\sum_{i=1}^{n}(q_i - \hat{q}_i)^2} \tag{4.3}$$

where $\hat{p}_i$ and $\hat{q}_i$ are the estimated and quaternion position and p_i and q_i are the fundamental position of truth and quaternion. The term β is used as a weighting factor to produce a comparable magnitude between the loss function so that the position and orientation error values.

Also, the quaternion is used to find the robot Z-axis rotation ψ (Yaw) (the ϕ (Row) and θ (Pitch) are components correspondent to the X- and Y-Axis rotation, respectively) , in the scenario performed by:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} arctan\frac{2(wx+yz)}{1-2(x^2+y^2)} \\ arcsin(2(wy-zx)) \\ arctan\frac{2(wz+xy)}{1-2(y^2+z^2)} \end{bmatrix} \tag{4.4}$$

4.4 Final Considerations

In this chapter, we presented the structure on which SpaceYNet can be implemented and perform pose regression and image depth analyses. The details conferred hereabout are required and distinguish the neural network so it can accurately perform the required analysis.

Creating a ground-truth for dataset RGB-D Scenes Dataset v.2 is known to be a time-consuming process by VisualSfM. Besides, the creation of the dataset created by the LaSER lab faced challenges in generating odometry labels due noises generated at the time and processed in the Extended Kalman Filter. However, the aforementioned dataset is an essential part available to the research community to experiment, considering its detail richness and the right amount of images to carry out the location training.

Furthermore, we discuss SpaceYNet, which has been presented in this chapter, which address is surely replicable by the community. The pose regression composition is similar to PoseNet when SpaceYNet v.1 is selected, and similar to ContextualNet when SpaceYNet v.2 is selected, adopting recurrent neural networks that can enhance the robot pose regres-

sion. Finally, we also presented how the data produced by the quaternion will be handled to produce rotation on the Z-axis.

Chapter 5

Experimental Evaluation

In this chapter, we show the workflow to train and test models,demonstrating information and precision predictions by SpaceYNet graphically, comparing it with PoseNet and ContextualNet. Also, there will be presented the analysis of optimizers, treatment setting, evaluation by increasing/decreasing network neurons process, and application of regularizers.

5.1 Experiments

5.1.0.1 Optimizer Analysis

Preliminary, we analyzed which optimizer fits better to our task problem. We tested SGD[1], Adam[2], Adadelta, and RMSProp. Besides, some loss functions have also been tested, which are Eq. 5.3, Eq. 5.2, and Eq. 5.1.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \hat{d}_i)^2} \tag{5.1}$$

$$\frac{1}{n}\sum_{i=1}^{n}(d_i - \hat{d}_i)^2 \tag{5.2}$$

$$\frac{1}{n}\sum_{i=1}^{n}|p_i - \hat{p}_i| \tag{5.3}$$

The test was to apply for faster convergence time in less amount of input data to train and look up validation results by ten epochs. The results presented when we used the loss

[1]Learning rate 10^{-5}, momentum 0.9 and *Nesterov*

[2]Learning rate 10^{-4} and *Amsgrad*

function Eq. 5.1 and the optimizer Adam obtained better convergence at the time, being 38.91%. Besides, it is possible to achieve using PoseNet, which also uses Adam optimizer.

	Eq. 5.1	**Eq. 5.2**	**Eq. 5.3**
SGD	32.45%	29.63%	27.78%
Adam	**38.91**%	31.79%	29.42%
Adadelta	26.24%	34.37%	28.56%
RMSProp	30.57%	28.04%	27.96%

Table 5.1: The Adam optimizer converges faster and higher than SGD, Adadelta, and RMSProp, performing with Eq. 5.1 to regress the robot pose.

5.1.1 Workflow

Fig. 5.1 shows the workflow adopted in the experiment steps, using the disposable dataset, RGB-D Scenes Dataset v.2 and LaSER.

Given as an input file, the text file containing the information as image file name and 6-DoF labels, from each read line, is getting the corresponding image files (RGB and depth-scene) from the memory. If the text file does not have the 6-DoF, the aggregated data is submitted to VisualSfM to generate the 6-DoF ground-truth using visual odometry.

The whole image data set (RGB and depth-scene images files) is resized and normalized, to apply it to the network. The whole label position data set is applied to be standardized. After it, the data is divided into two packages: 70% to the first pack, and 30% to the second pack. The first one will be used to train and validate the network to predict the robot pose and regress the depth-scene of the RGB image as input.

On the training/validation step, the network is generated and used to train, being evaluated by the use of optimizer and loss function the cost distance error. After that, it is possible to export the weighted network to the memory and generate visualizations such as the loss and accuracy of the network.

By use of the exported network, the test step is applied, adopting online and offline methods, getting some positions not shown to the network from the 30% split data to the offline method, and walking with the robot, randomly, to the online method. Then, the

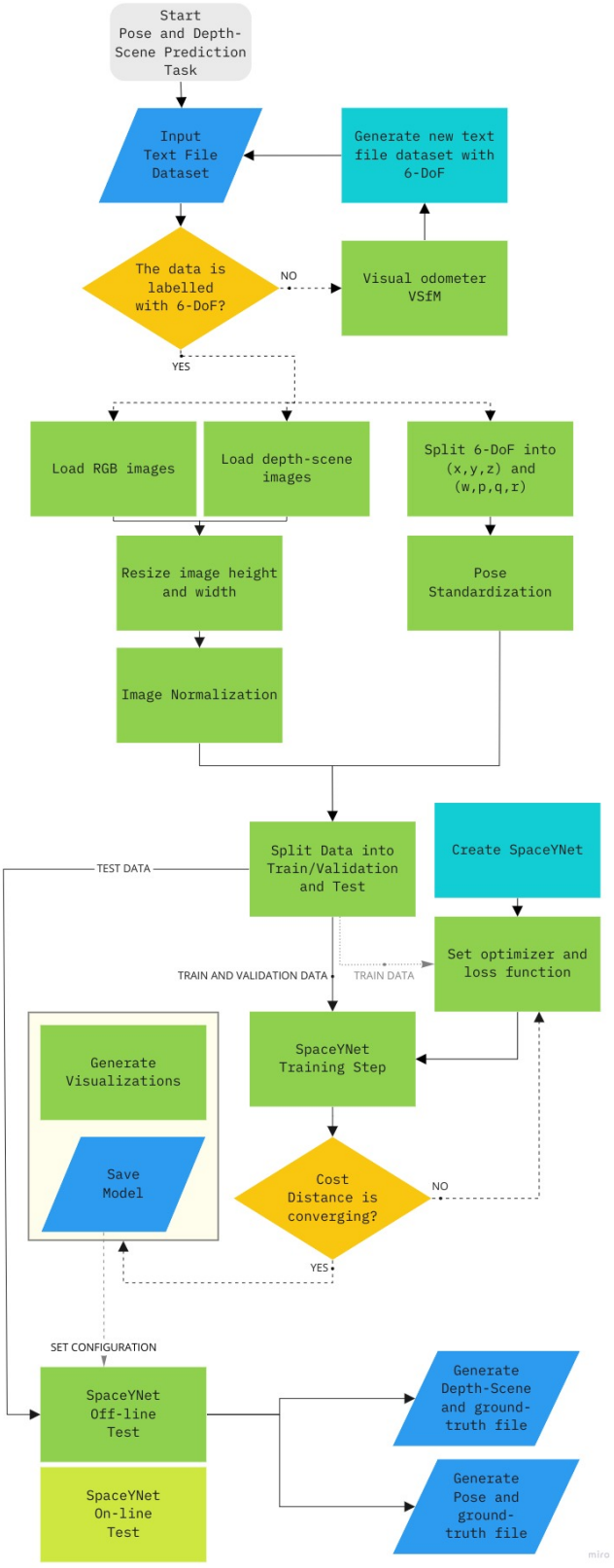


Figure 5.1: Distribution workflow process.

outputs of depth-scene and pose labels are generated, written in the file as output.

The train is set to 500 epochs (stops automatically if the validation cost not decrease after 30 epochs), with a learning rate equal to 0.00146 and $decay = \frac{learning\_rate}{epochs}$. The experiments to pose regression uses PoseNet's and ContextualNet's results to compare with SpaceYNet.

5.1.2 Tests using RGB-D Scenes Dataset v.2

5.1.2.1 SpaceYNet v.1 and PoseNet

Initially, both SpaceYNet v.1 and PoseNet were applied to pose tasks using the collection of images RGB-D Scenes Dataset v.2.

The completion of the algorithm that computes the expected real mean value adopts δ_i as the computed value by $|\kappa_i - \beta_i|$, where κ is the ground-truth value and β_i PoseNet regression output, adopting σ_i as the computed value by $|\kappa_i - \alpha_i|$, where κ is the ground-truth and the SpaceYNet regression output is displayed in Fig. 5.2.

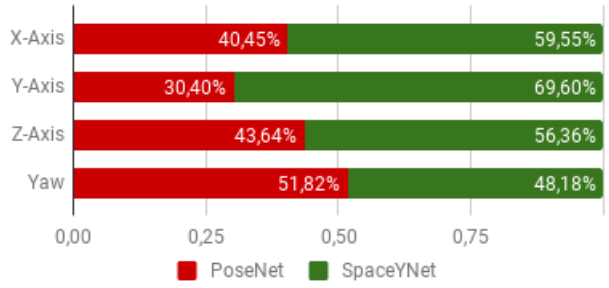


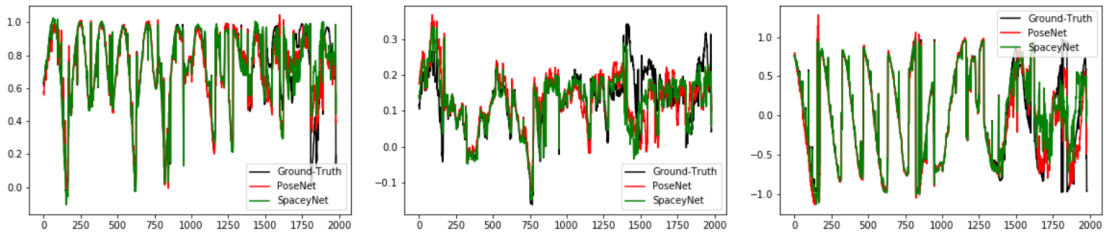
Figure 5.2: Comparative result of the approximate rate of PoseNet and SpaceYNet to the ground-truth on the RGB-D Scenes Dataset v.2. The values are measured by $\min(\delta, \sigma)$ for each pose regression, and the closer estimatives infer that the value for the network axis-in-question rate prediction increases. A similar procedure is used to measure the Yaw prediction approximation and incremental rate. It indicates that SpaceYNet is closer to axes regression in most of the cases and has a similar rate to predict the camera's rotation when matched to the ground-truth.

Returning to the first element from $\min(\delta_i, \sigma_i)$, is incremented the variable that counts PoseNet closer approximation; if it returns, the second element increments the variable that counts SpaceYNet's closer approximation. SpaceYNet is closer to the ground-truth 59.55%

in to predict X-axis, 69.60% in to predict Y-axis, and 56.36% in to predict Z-axis. Also, we extracted the matched information from Fig. 5.2 and it was noticed that:

- SpaceYNet was better to predict the 3D camera position $p \in \Re^3$ of $P = [p, q]$;
- PoseNet was better to predict the *Yaw* camera rotation but not too far;
- SpaceYNet is helpful to regress the camera location in 61.83% of the average cases.

Fig. 5.3 shows the spectrum prediction perspective of the ground-truth. The deviation between SpaceYNet, PoseNet, and the ground-truth in all axes signifies the estimated values predicted. Both neural networks present a noise signal to predict the Cartesian's axes $\in$ 6-DoF.



(a) X-axis Prediction error spectrum (b) Y-axis Prediction error spectrum (c) Z-axis Prediction error spectrum

Figure 5.3: Spectrum Pose Regression using the RGB-D Scenes Dataset v.2.

Furthermore, to calculate the accumulated error variables to predict the Axes (δ for PoseNet amount sum errors, and σ to discriminate the SpaceYNet amount sum errors), we adopted the following equations:

$$\delta = \sum_{0}^{i} |\kappa_i - \beta_i| \tag{5.4}$$

$$\sigma = \sum_{0}^{i} |\kappa_i - \sigma_i| \tag{5.5}$$

Using Equation 5.4 and Equation 5.5, respectively, to extract toward each Cartesian's axes $\in$ 6-DoF regression, brought out δ and σ, comparing between PoseNet and SpaceYNet values shown in Fig. 5.4.

Fig. 5.5 presents the depth prediction produced by the output of depth-scene from SpaceYNet. Lighter pixels (but not white) indicate mid proximity to the camera. Darker pixels indicate greater distance or higher proximity from the object to the camera, as well as

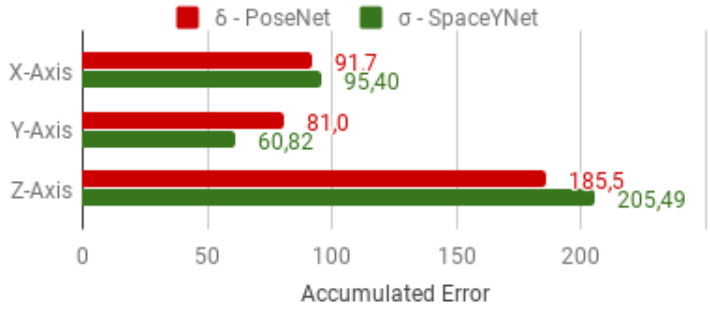


Figure 5.4: RGB-D Scenes Dataset v.2 - Accumulated Error produced by PoseNet and SpaceYNet.

the failure to capture distance information as in some noises captured in detail of the object. White pixels indicate blank walls. The SpaceYNet tends to be blurry or sharper than the original depth-scene image label, but it does not mean it does not come close to the expected label.

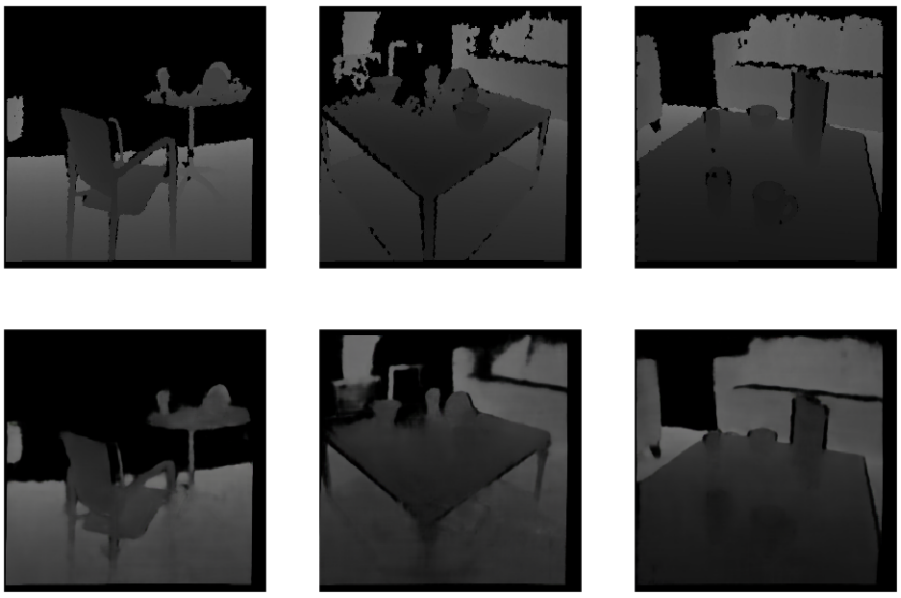


Figure 5.5: RGB-D Scenes Dataset v.2 Depth regression. The reference values are at the top and the respective forecasts of SpaceYNet are at the bottom.

Lastly, Fig. 5.6 presents the loss in pose predicting $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ with the position of the 3D camera $\mathbf{p} \in \Re^3$ and the orientation represented by the quaternion, $\mathbf{q} \in \Re^4$. Both lines apply to the training and validation steps. Also, the depth loss curve in the activity of predicting the depth scene was analysed, presented in Fig. 5.7.

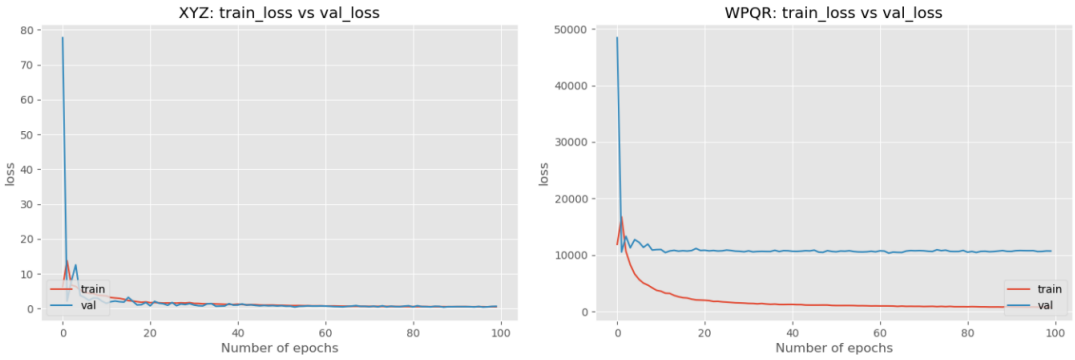


Figure 5.6: Pose loss $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ on the RGB-D Scenes Dataset v.2.

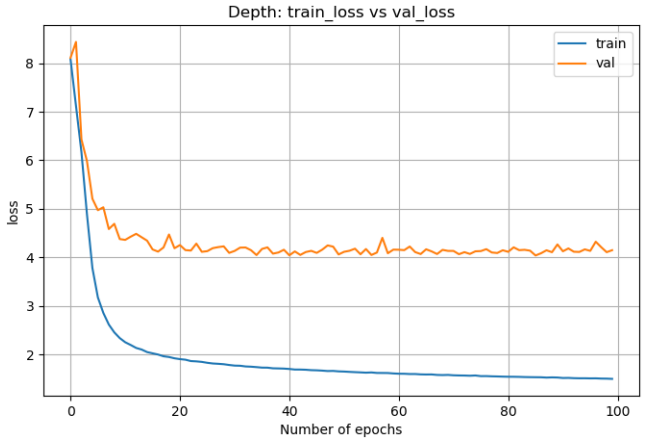


Figure 5.7: Depth loss produced by SpaceYNet.

5.1.3 Tests using LaSER Dataset

5.1.3.1 SpaceYNet v.1 and PoseNet

To the training/validation step using the LaSER Dataset, 70% of the amount of the data is reserved, 30% of the dataset reserved for the test step. As at RGB-D Scenes Dataset v.2 step, PoseNet and SpaceYNet are submitted to the algorithm to compute the mean approximation by the expected real value of their regression outputs. The variable δ_i represents the absolute difference value between the ground-truth and PoseNet, and σ_i, the absolute difference value formulation between SpaceYNet and the ground-truth. To the LaSER Dataset, SpaceYNet achieved a better-estimated mean than PoseNet in predicting the X-axis and the rotation of the robot in the plane, as shown in Fig. 5.8.

From Fig. 5.8 we can notice:

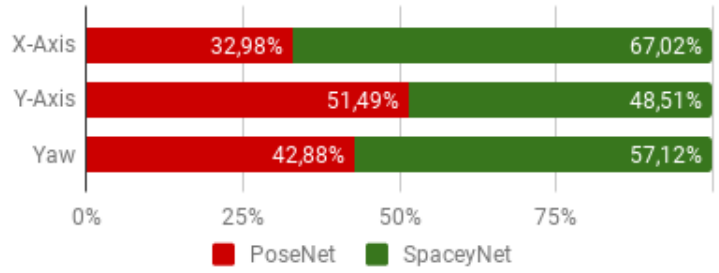


Figure 5.8: Comparative result approximate values of PoseNet and SpaceYNet to the ground-truth on the LaSER Dataset.

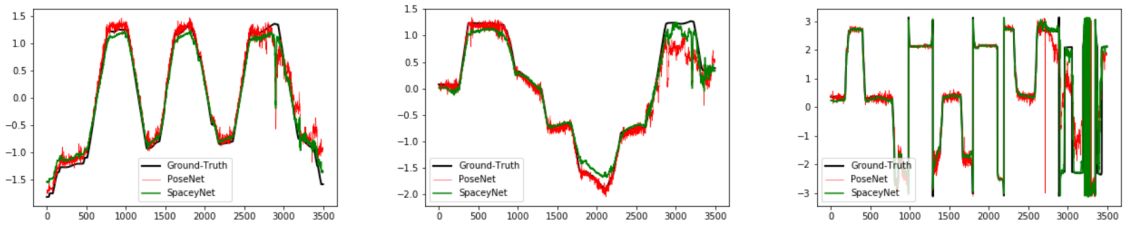
- SpaceYNet was better to predict two Axes of the 3D camera position $p \in \Re^3$ of $P = [p, q]$;
- PoseNet was better to predict the Y-Axis but not too far;
- SpaceYNet was better to predict the *Yaw* camera rotation but not too far;
- SpaceYNet is more helpful to regress the camera location in 57.55% of the average cases.

Comparisons with SpaceYNet's regression on each axis with PoseNet's regression and the ground-truth can be observed in Fig. 5.9. The cost among SpaceYNet, PoseNet, and the ground-truth in all axes express that the computed values predicted by SpaceYNet are more reliable than the measure values predicted by PoseNet. Besides, PoseNet has more noise than SpaceYNet to predict the position on the LaSER lab, which appears when the distance to the ground truth is computed (Fig. 5.11).

The spectrum graphs shown by Fig. 5.3 and Fig. 5.9 present to us cross-informations, which are:

- The Z-Axis value influences directly the X-Y-Axis error;
- Z-Axis error is higher than both networks, that can indicate a higher difficulty to estimate the up-down movement by PoseNet and SpaceYNet;
- In the case of LaSER Dataset, it can be noticed that the robot is providing better results when moving towards the wall;

- PoseNet presents itself noisier than SpaceYNet. Moving a little closer to the spectrum referent to the X-axis spectrum chart, in Fig. 5.10 it is possible to see visually how much PoseNet is noisier than SpaceYNet.



(a) X-axis Prediction error spectrum (b) Y-axis Prediction error spectrum (c) Yaw Prediction error spectrum

Figure 5.9: Prediction position for the LaSER Dataset

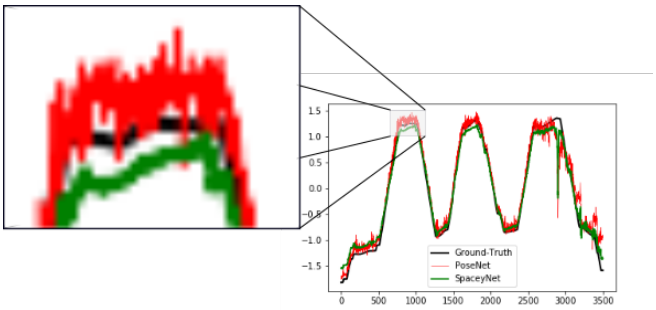
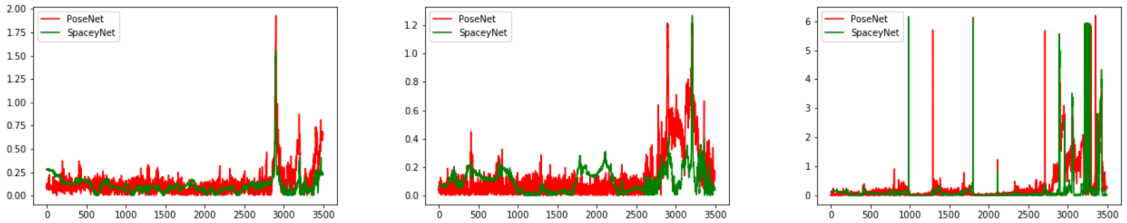


Figure 5.10: Noise visualization in pose regression between PoseNet and SpaceYNet. The black-line is the ground-truth, the red-line is PoseNet regression, and the green-line is SpaceYNet regression.



(a) X-axis Prediction distance spectrum (b) Y-axis Prediction distance spectrum (c) Yaw Prediction distance spectrum

Figure 5.11: Prediction distance for the ground truth in LaSER Dataset

The main difference in evaluating the RGB-D Scenes Dataset v.2 and LaSER datasets is because in the second case the Z-axis was not considered. That is because it is across a terrestrial robot task, and the Z-axis' value is constant. Fig. 5.12 shows the accumulated errors for each axis predicted by SpaceYNet.

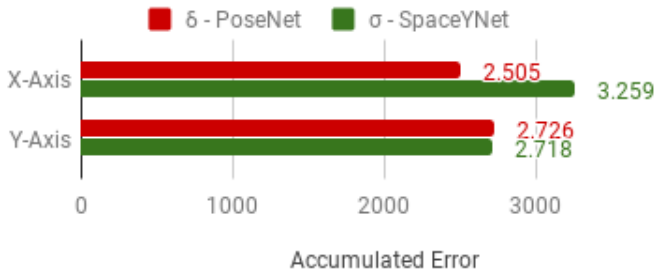


Figure 5.12: LaSER Dataset - Accumulated Errors produced by PoseNet and SpaceYNet.

We applied PoseNet and SpaceYNet to estimate the 6-DOF pose of the robot in each sent image. However, until now, we must analyze those images of the dataset to train and test our approach. This experiment was filmed[3] using the performed route at the LaSER Laboratory. The path performed in the experiment can be seen in Fig. 5.13.

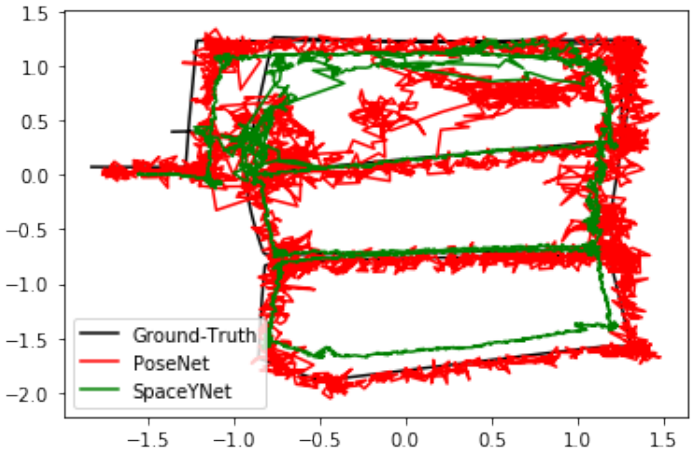


Figure 5.13: The trajectory performed by the Turtlebot as the ground-truth and used to evaluate SpaceYNet and PoseNet in the LaSER building. According to the graph, PoseNet shows to be noisier.

It can be noticed that SpaceYNet has less noise in predicting the robot's pose than PoseNet. For this experiment, we did not have sufficient relative localization to help tracking the robot's displacement. We only fused the gyroscope and wheel odometry of the robot. In future works, we will use visual odometry as a tool to improve the robot's pose, which will improve the robot's localization.

The result of the depth prediction produced by SpaceYNet is present in Fig. 5.14, show-

[3]See the video of the experiment at `https://youtu.be/oIV7D-RVRnM`

ing that the network can preserve information in its characteristic vectors concerning the depth prediction.

Figure 5.14: LaSER Dataset depth regression outputs. The reference values are at the top and the respective forecasts of SpaceYNet are at the bottom.

SpaceYNet has provided higher noise tolerance and, consequently, allows it to follow a route avoiding route deflection due to False-Positive to identify an obstacle where it does not exist.

The loss to predict the axis and quaternion and respective accuracy curves are shown below in Fig. 5.15.

Furthermore, Fig. 5.16 shows the loss curve in the activity of predicting the depth of the scene and the predictions made by the network for some positions. The result of the depth prediction produced by SpaceYNet is present in Fig. 5.14.

5.1.3.2 SpaceYNet v.2, ContextualNet and SpaceYNet v.1

Adopting SpaceYNet v.2, with the addition of recurrent networks attached at the end of the path designed to improve the robot pose from monocular image RGB, and comparing it to ContextualNet (that is the PoseNet improved by application of use of recurrent networks), the depth-scene regression stays identical, already not having adjustments on the depth-scene end network.

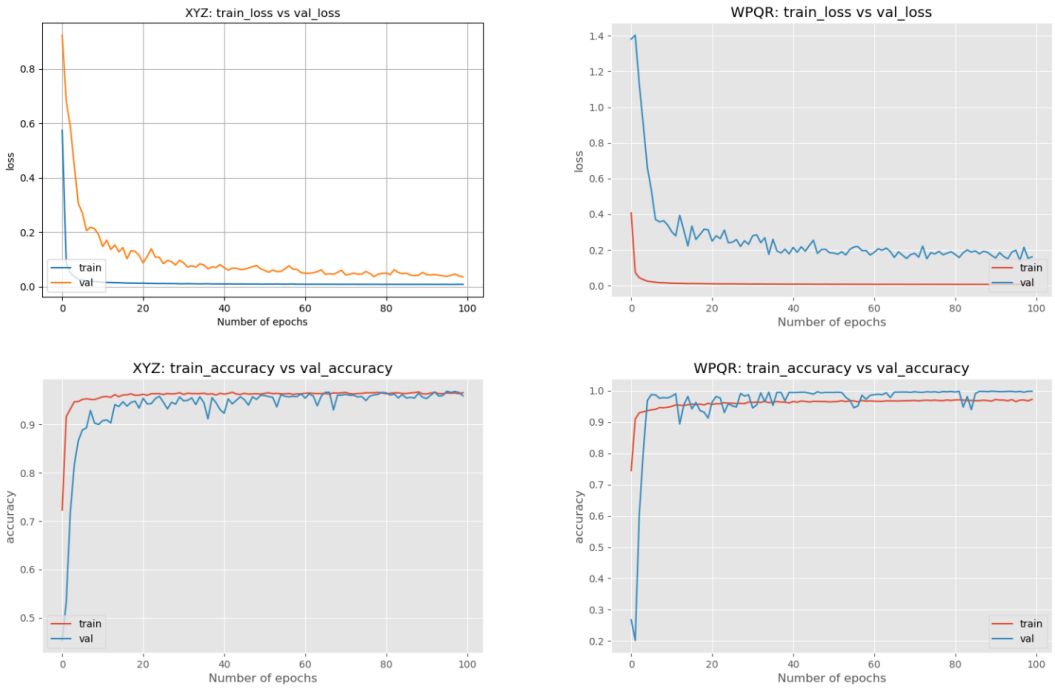


Figure 5.15: Pose loss and accuracy $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ on the LaSER Dataset

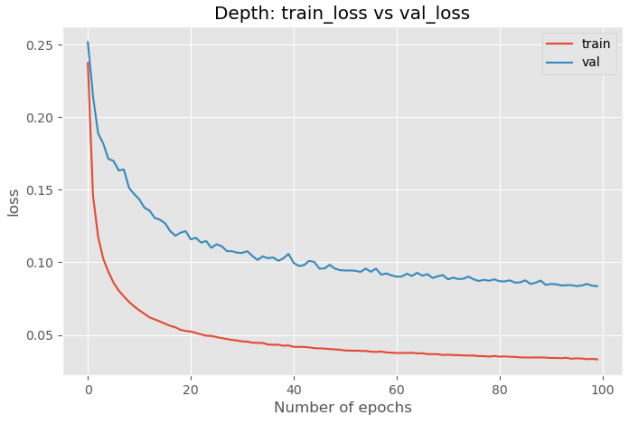
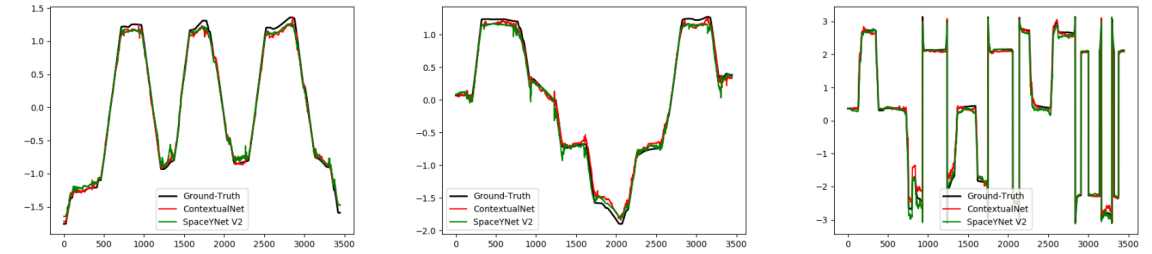


Figure 5.16: Depth loss produced by SpaceYNet.

Fig. 5.17 is possible to observe less noise when RNNs are employed to the robot pose regression task, enhancing both SpaceYNet's and the old PoseNet's (now called ContextualNet) performance.

Also, from Fig. 5.17, it is possible to extract some of cross-informations, which are:

- Robot pose that occurs at curves movements were noisier in SpaceYNet v.1, and more precisely in SpaceYNet, less noisy (Fig. 5.18 shows the noise reduction by RNNs



(a) X-axis Prediction error spectrum (b) Y-axis Prediction error spectrum (c) Yaw Prediction error spectrum

Figure 5.17: Spectrum pose regression between SpaceYNet v.2 and ContextualNet. The black-line is the ground-truth, the red-line is ContextualNet predictions, and the green-line is SpaceYNet v.2 predictions.

employment at the SpaceYNet);

- At the end of the route, all axes were affected by noise or did not predict closer in both SpaceYNet v.1 and PoseNet. RNNs, ContextualNet and SpaceYNet v.2 solve this challenging problem.

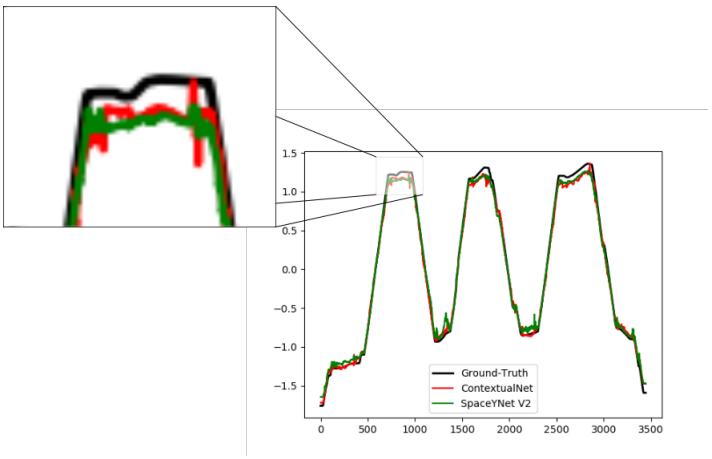


Figure 5.18: Noise visualization in pose regression between ContextualNet and SpaceYNet v.2. The black-line is the ground-truth, the red-line is ContextualNet predictions, and the green-line is SpaceYNet v.2 predictions.

The pose prediction with SpaceYNet v.1 is noisier than with SpaceYNet v.2; although the route predicts the pose, it is essential to get it accurately. Comparing these two networks, SpaceYNet v.1 and SpaceYNet v.2, it is possible to obtain the following results while comparing both of them (Fig. 5.19):

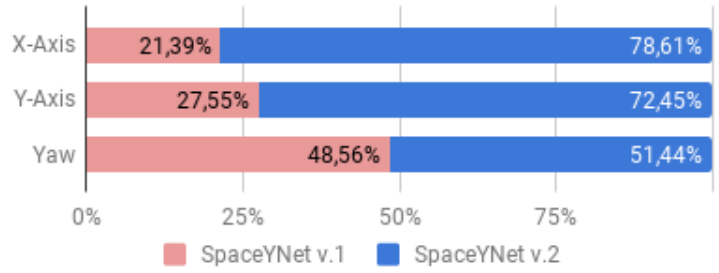


Figure 5.19: Comparative result approximate values of SpaceYNet v.1 and SpaceYNet v.2 to the ground-truth on the LaSER Dataset.

According to Fig. 5.19, SpaceYNet v.2 is 67.5% of times better than SpaceYNet v.1 if there are applied RNNs at the end of the network to improve the pose regression, as the comparison route regression is shown in Fig. 5.20 between them. When the route map is about SpaceYNet v.2 and ContextualNet, the comparison shows not being too far, as shown in Fig. 5.21.

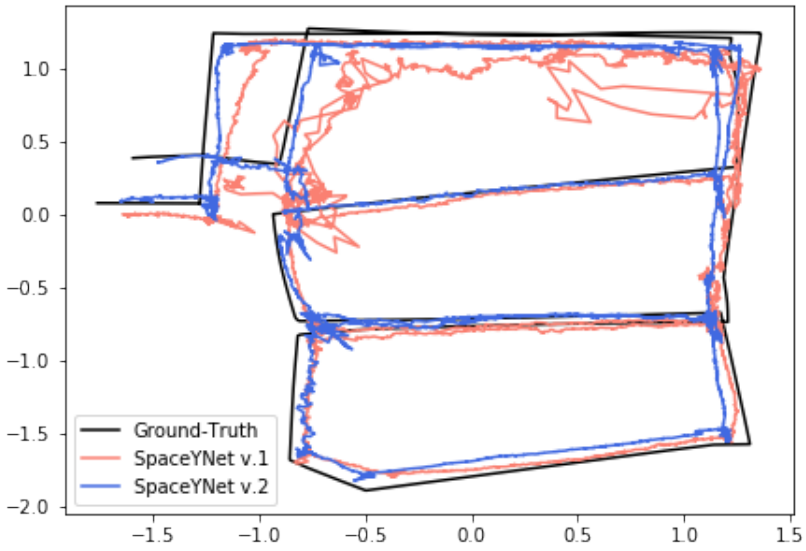


Figure 5.20: Route regression of SpaceYNet v.1 and SpaceYNet v.2 using the LaSER Dataset.

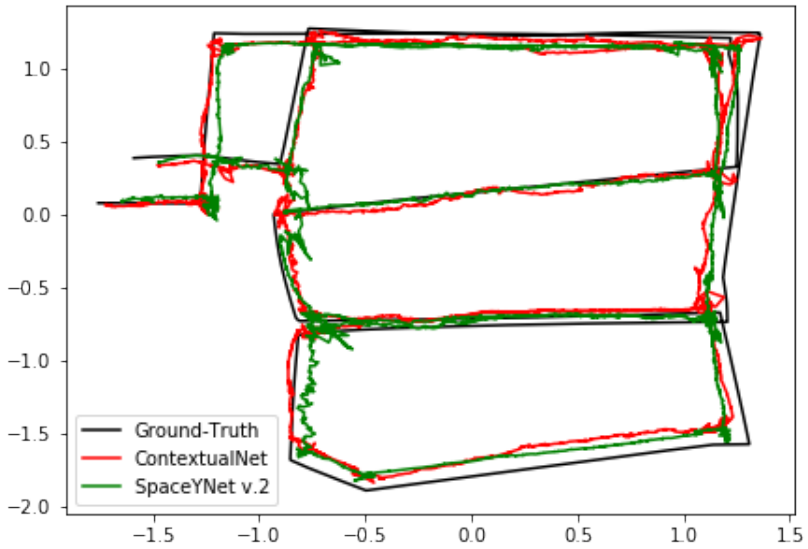


Figure 5.21: Route regression of SpaceYNet v.2 and ContextualNet using the LaSER Dataset.

5.2 Final Considerations

SpaceYNet can predict the monocular scene-depth and robot poses from an RGB image as an input data with higher runs to the state-of-the-art PoseNet and ContextualNet. Although there were challenges in training a network to perform two concurrent activities without loss of effectiveness in some, SpaceYNet preserved the information in its characteristic vectors concerning the depth prediction (as shown in Fig. 5.14).

The regression of pose axis in controlled environments and conditions with $\mathbf{p} \in \Re^3$ and $\mathbf{q} \in \Re^4$ has conferred remarkable outcomes, and when compared to the PoseNet and ContextualNet, it is noticed a better accuracy. The research experiments motivates us and leads to an encouraging path, and this will be done by expanding the collection of representations and potential variations in the neural network.

It demonstrated SpaceYNet's effectiveness on two large scale datasets. We perceived a significant improvement of results compared to the original loss proposed by PoseNet, narrowing the performance gap to conventional point-feature approaches.

Considering that these regressions are made in uncontrolled scenarios, to regress robot pose may be imprecise, and, for the robot to move without accidents, depth prediction deems

necessary, even without the robot being weak to locate itself, depth may cancel the challenge.

Chapter 6

Conclusion

SpaceYNet's results attest higher accuracy compared to the state-of-art PoseNet and ContextualNet. Besides, the network is able so far not only to pose regression, but also to regress the depth-scene of the environment, by the use of Autoencoder and CNN. Although there were difficulties in training a network to achieve two concurrent activities without loss of effectiveness in some of the tasks, SpaceYNet preserved the information in its characteristic vectors concerning depth prediction.

To improve the first proposition, (SpaceYNet v.1) there were selected study approaches such as the use of Bayesian Networks to resolve uncertainties states, that avoids overfitting problems as well. The outcomes are improved by the adoption of Recurrent Networks, achieving better results than the first SpaceYNet version.

The importance of depth-scene prediction evades the problem that, while robots are able to locate themselves in the environment, in real scenarios, they may suffer interferences from the real world, such as theft. Considering that those predictions are made in uncontrolled scenarios, predictions of positioning may be imprecise. Therefore, for the robot to move without accidents, depth prediction becomes imperative. It is an important task, especially in occlusion dilemmas.

The pose regression in controlled environments of the axes in $\mathbf{p} \in \Re^3$ and $\mathbf{q} \in \Re^4$ have shown good results. When compared to PoseNet and ContextualNet, it is noticed with better accuracy in general terms.

To complete, this work utilized a system of 3D reconstruction using structure from motion. Also, it assumed two datasets, the former being a public one, and the latter made by

LaSER laboratory and shared to be used in researches that need RGB image, depth-scene, and odometry, got by the implementation of Extended Kalman Filter attached to the robot odometry. Also, we provided a new dataset with a fully controlled environment. The dataset created by the LaSER laboratory provides an extensive robot tour, merging RGB images, depth-scenes, and pose labels, from the robot odometry, indeed without an unspecified process to dress that.

The input data introduces a bias which produces different patterns for the system, requiring the previous normalization of the input image and 6-DoF standardization.

Finally, as demonstrated, we worked with just requiring cameras in the testing steps correctly, and SpaceYNet can be implemented on terrestrial robots and drones, once 6-DoF ables it.

The research motivates us, as it leads to a promising path and will be done by increasing the set of samples and possible variations in the neural network.

Bibliography

[Bourgault *et al*. 2002]BOURGAULT, F. *et al*. Information based adaptive robotic exploration. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2002. v. 1, p. 540–545 vol.1.

[Cai 2017]CAI, Z. Feature Learning for RGB-D Data. n. August, 2017.

[Choi, Lee and Zhang 2016]CHOI, J.; LEE, B.; ZHANG, B. Human body orientation estimation using convolutional neural network. *CoRR*, abs/1609.01984, 2016.

[Ciresan *et al*. 2012]CIRESAN, D. C. *et al*. Deep neural networks segment neuronal membranes in electron microscopy images. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*. [S.l.: s.n.], 2012. p. 2843–2851.

[Deng and Zeng 2013]DENG, X.; ZENG, Q. Research on laser-assisted odometry of indoor uav with monocular vision. In: *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*. [S.l.: s.n.], 2013. p. 165–169.

[Dissanayake *et al*. 2001]DISSANAYAKE, M. W. M. G. *et al*. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, v. 17, n. 3, p. 229–241, June 2001. ISSN 1042-296X.

[Durrant-Whyte and Bailey 2006]Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, v. 13, n. 2, p. 99–110, June 2006. ISSN 1070-9932.

[Eigen and Fergus 2014]EIGEN, D.; FERGUS, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734, 2014.

[Gal and Ghahramani 2016]GAL, Y.; GHAHRAMANI, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*. [S.l.: s.n.], 2016.

[Gálvez-López *et al*. 2016]GáLVEZ-LóPEZ, D. *et al*. Real-time monocular object slam. *Robot. Auton. Syst.*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, v. 75, n. PB, p. 435–449, jan. 2016. ISSN 0921-8890.

[Geiger, Lenz and Urtasun 2012]GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2012. p. 3354–3361. ISSN 1063-6919.

[Henry *et al*. 2013]HENRY, P. *et al*. Patch volumes: Segmentation-based consistent mapping with RGB-D cameras. In: *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*. [s.n.], 2013. p. 398–405. ISBN 9780769550671. Disponível em: <http://reconstructme.net/>.

[Hochreiter and Schmidhuber 1997]HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

[Kahn *et al*. 2018]Kahn, G. *et al*. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2018. p. 1–8. ISSN 2577-087X.

[Kendall and Cipolla 2017]KENDALL, A.; CIPOLLA, R. Geometric loss functions for camera pose regression with deep learning. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. IEEE, 2017. v. 2017-Janua, p. 6555–6564. ISBN 9781538604571. ISSN 10504729. Disponível em: <http://ieeexplore.ieee.org/document/8100177/>.

[Kendall, Cipolla and Feb 2016]KENDALL, A.; CIPOLLA, R.; FEB, C. V. Modelling Uncertainty in Deep Learning for Camera Relocalization. p. 4762–4769, 2016.

[Kendall, Grimes and Cipolla 2015]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.]: IEEE, 2015. v. 2015 Inter, p. 2938–2946. ISBN 9781467383912. ISSN 15505499.

[Konda, Memisevic and Krueger 2014]KONDA, K.; MEMISEVIC, R.; KRUEGER, D. *Zero-bias autoencoders and the benefits of co-adapting features*. 2014.

[Kuznietsov, Stückler and Leibe 2017]KUZNIETSOV, Y.; STÜCKLER, J.; LEIBE, B. Semi-Supervised Deep Learning for Monocular Depth Map Prediction. feb 2017.

[Lai, Bo and Fox 2014]LAI, K.; BO, L.; FOX, D. Unsupervised feature learning for 3d scene labeling. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, p. 3050–3057, 2014.

[LaSER 2019]LASER. *LaSER Dataset*. Apr 2019. Disponível em: <https://www.kaggle.com/dunfrey/laser-dataset>.

[Li *et al*. 2019]LI, S. *et al*. *Sequential Adversarial Learning for Self-Supervised Deep Visual Odometry*. [S.l.], 2019. Disponível em: <http://arxiv.org/abs/1908.08704>.

[Mcculloch and Pitts 1943]MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943.

[Mortensen, Deng and Shapiro 2005]MORTENSEN, E. N.; DENG, H.; SHAPIRO, L. A sift descriptor with global context. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005. (CVPR '05), p. 184–190. ISBN 0-7695-2372-2.

[Naseer and Burgard 2017]Naseer, T.; Burgard, W. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2017. p. 1525–1530. ISSN 2153-0866.

[Nestares *et al*. 1998]NESTARES, O. *et al*. Efficient spatial-domain implementation of a multiscale image representation based on gabor functions. *J. Electronic Imaging*, v. 7, p. 166–173, 1998.

[Paniagua and Lopez 2019]PANIAGUA, J. L.; LOPEZ, J. A. Dimensionality Reduction Applied to Time Response of Linear Systems Using Autoencoders. *2019 IEEE Colombian Conference on Applications in Computational Intelligence, ColCACI 2019 - Proceedings*, IEEE, p. 1–6, 2019.

[Patel, Emery and Chen 2018]PATEL, M.; EMERY, B.; CHEN, Y.-y. ContextualNet : Exploiting Contextual Information using LSTMs to Improve Image-based Localization. p. 5890–5896, 2018.

[Phuong and Phong 2019]PHUONG, T. T.; PHONG, L. T. On the convergence proof of amsgrad and a new version. *CoRR*, abs/1904.03590, 2019.

[Redmon *et al*. 2015]REDMON, J. *et al*. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. Disponível em: <http://arxiv.org/abs/1506.02640>.

[Ronneberger, P.Fischer and Brox 2015]RONNEBERGER, O.; P.FISCHER; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. [S.l.: s.n.], 2015. p. 234–241.

[Salih and Malik 2012]SALIH, Y.; MALIK, A. S. Depth and geometry from a single 2d image using triangulation. In: *2012 IEEE International Conference on Multimedia and Expo Workshops*. [S.l.: s.n.], 2012. p. 511–515.

[Saxena, Chung and Ng 2008]SAXENA, A.; CHUNG, S. H.; NG, A. Y. 3-D depth reconstruction from a single still image. *International Journal of Computer Vision*, v. 76, n. 1, p. 53–69, 2008. ISSN 09205691.

[Shah *et al*.]SHAH, P. *et al*. Followne: Robot navigation by following natural language directions with deep reinforcement learning. *CoRR*.

[Shi *et al*. 2015]SHI, X. *et al*. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: *Advances in Neural Information Processing*

Systems. [s.n.], 2015. v. 2015-Janua, p. 802–810. ISSN 10495258. Disponível em: <https://arxiv.org/pdf/1506.04214.pdf>.

[Souto *et al*. 2017]SOUTO, L. *et al*. Stairs and doors recognition as natural landmarks based on clouds of 3d edge-points from rgb-d sensors for mobile robot localization. *Sensors*, v. 17, n. 1824, p. 1–16, 2017.

[Strang and Nguyen 1996]STRANG, G.; NGUYEN, T. *Wavelets and Filter Banks*. [S.l.]: Wellesley-Cambridge Press, 1996. ISBN 9780961408879.

[Szegedy *et al*. 2015]SZEGEDY, C. *et al*. Going deeper with convolutions. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 7.

[Tariyal *et al*. 2016]TARIYAL, S. *et al*. Deep Dictionary Learning. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 4, p. 10096–10109, 2016. ISSN 21693536.

[Tateno *et al*. 2017]TATENO, K. *et al*. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. apr 2017.

[Thrun, Burgard and Fox 2005]THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. [S.l.]: The MIT Press, 2005. ISBN 0262201623.

[Tomar *et al*. 2018]TOMAR, D. *et al*. Feature Selection Using Autoencoders. *Proceedings - 2017 International Conference on Machine Learning and Data Science, MLDS 2017*, v. 2018-January, p. 56–60, 2018.

[Wang, Shi and Yeung 2015]WANG, H.; SHI, X.; YEUNG, D.-Y. Relational stacked denoising autoencoder for tag recommendation. In: *AAAI*. [S.l.: s.n.], 2015.

[Wu 2013]Wu, C. Towards linear-time incremental structure from motion. In: *2013 International Conference on 3D Vision - 3DV 2013*. [S.l.: s.n.], 2013. p. 127–134. ISSN 1550-6185.

[Wu 2013]Wu, C. Towards linear-time incremental structure from motion. In: *2013 International Conference on 3D Vision - 3DV 2013*. [S.l.: s.n.], 2013. p. 127–134. ISSN 1550-6185.

[Xiao, Zhou and Gong 2018]XIAO, X.; ZHOU, Y.; GONG, Y.-J. RGB-âĂŸD' Saliency Detection With Pseudo Depth. *IEEE Transactions on Image Processing*, IEEE, PP, n. c, p. 1–1, 2018. ISSN 1057-7149.

[Xing, Ma and Yang 2016]XING, C.; MA, L.; YANG, X. Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *Journal of Sensors*, v. 2016, p. 1–10, 01 2016.