POLITECNICO DI TORINO

MASTER DEGREE THESIS

Department or Control and Computer Engineering

Mechatronic Engineering

a.y. 2023/2024

Predictive algorithm and guidance for human motion in robotics teleoperation

Supervisors:

Alessandro RIZZO
(*Politecnico di Torino*)

Domenico PRATTICHIZZO
(*Università di Siena*)

Enrico TURCO
(*Istituto Italinao di Tecnologia*)

Valerio BO
(*Istituto Italinao di Tecnologia*)

Candidate:

Francesco STOLCIS

March 2024

POLITECNICO DI TORINO

Abstract

Department or Control and Computer Engineering

Mechatronic Engineering

Predictive algorithm and guidance for human motion in robotics teleoperation

by Francesco STOLCIS

This thesis proposes a novel approach for enhancing the teleoperation of robotic arms through shared control mechanisms, integrating advanced machine learning techniques and classical control methods. The primary objective is to develop a robust system capable of autonomously guiding the robotic arm towards predicted objects while maintaining operator oversight. The framework employs a Long Short-Term Memory (LSTM) classification model to predict the location and nature of objects within the robot's workspace. This predictive capability enables the system to anticipate the operator's intentions and adaptively plan trajectories. Furthermore, artificial potential fields are utilized to generate guidance commands that assist the operator in maneuvering the robotic arm towards the predicted objects. By combining the predictive power of LSTM with the reactive nature of potential fields, the system achieves a seamless fusion of human expertise and autonomous control, ensuring both efficiency and safety in teleoperation tasks. The proposed approach is implemented and validated through comprehensive simulations and real-world experiments using a teleoperated robotic arm platform. Performance evaluations demonstrate the effectiveness and reliability of the shared control system, showcasing improved object manipulation capabilities and reduced cognitive workload for the operator. Additionally, the system's adaptability to varying environmental conditions and object dynamics is examined, highlighting its potential for deployment in diverse teleoperation scenarios. Overall, this thesis contributes to the advancement of teleoperated robotic systems by introducing a novel framework that seamlessly integrates machine learning and classical control techniques. The proposed shared control approach offers enhanced capabilities for object manipulation tasks, paving the way for more efficient and intuitive human-robot collaboration in various domains, including manufacturing, healthcare, and disaster response.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
CAD	**C**omputer-**A**ided **D**esign
CNN	**C**onvolutional **N**eural **N**etwork
CPD	**C**oherent **P**oint **D**rift
CV	**C**omputer **V**ision
D	**D**imension(s)
DOF	**D**egrees **O**f **F**reedom
EM	**E**xpectation-**Maximization**
EROSS	**E**uropean **R**obotic **O**rbital **S**upport **S**ervices
EU	**E**uropean **U**nion
FAIR	**F**acebook's **AI R**esearch lab
FOV	**F**ield **O**f **V**iew
GEO	**G**eostationary **E**quatorial **O**rbit
GMM	**G**aussian **M**ixture **Model**
GPU	**G**raphics **P**rocessing **U**nit
HRNet	**H**igh-**Resolution Net**work
ICP	**I**terative **C**losest **P**oint
IOD	**I**n **O**rbit **D**emonstration
LEO	**L**ow **E**arth **O**rbit
ML	**M**achine **L**earning
MSE	**M**ean **S**quared **E**rror
NN	**N**eural **N**etwork
RANSAC	**Ran**dom **Sa**mple **C**onsensus
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
RPY	**R**oll, **P**itch, **Y**aw
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SURF	**S**peeded **U**p **R**obust **F**eature
MSER	**M**aximally **S**table **E**xtremal **R**egions
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
PnP	**P**erspective-**n**-**P**oint
SfM	**S**tructure **f**rom **M**otion
SPN	**S**pacecraft **P**ose **N**etwork
SWaP	**S**ize **W**eight **a**nd **P**ower

List of Symbols

Symbol	Name	Unit
E_{CNN}	Landmark Regression error	px
E_{NN}	Landmark Mapping error	cm
E_T	Translation error	cm
E_R	Rotation error	°
S_T	Translation score	-
S_R	Rotation error	rad

Chapter 1

Introduction

1.1 Thesis Objective

In the expanse of space, satellite missions and on-orbit services have become critical assets, serving a myriad of applications including Earth observation, global communication, and scientific research.

The progressive introduction of AI algorithms into various environments, including space applications, represents a significant leap forward in technological advancement. In the context of pose estimation in space, the incorporation of AI brings a multitude of benefits that enhance the autonomy of satellite operations.

In recent years, we've witnessed a rapid proliferation of on-orbit satellites, driven by advancements in technology and the need for enhanced space services. As the number of these satellites continues to rise, the complexities associated with their safe and effective navigation, rendezvous, and scientific missions have grown in tandem. This is where AI shines, as it steps in to revolutionize the field of satellite pose estimation.

AI algorithms, equipped with their machine learning capabilities, enable satellites to process vast amounts of data from onboard sensors with remarkable precision and efficiency. This means an elevated level of accuracy in determining a satellite's position, orientation, and trajectory. But the benefits go beyond mere precision.

AI algorithms, equipped with their machine learning capabilities, enable satellites to process vast amounts of data from onboard sensors with remarkable precision and efficiency. One remarkable development is the ability to estimate a satellite's position and orientation using just a single camera, eliminating the need for a stereo-camera setup. This innovation not only enhances accuracy but also reduces hardware complexity, making satellite design more cost-effective. AI-driven monocular camera-based pose estimation empowers satellites to autonomously process visual data, adjust to dynamic orbital environments, and make informed decisions, even in the midst of complex maneuvers, ensuring the mission's success and safety.

Moreover, the increased autonomy provided by AI minimizes the need for constant human intervention and ground control. This not only reduces operational costs but also allows human operators to focus on more strategic aspects of the mission, enhancing productivity and mission efficiency. As we look to the future, AI algorithms promise to usher in a new era of space exploration and satellite operations.

In summary, the progressive introduction of AI algorithms in space applications, particularly in pose estimation, opens the door to enhanced accuracy, real-time adaptability, autonomy, and overall mission efficiency. This transformative technology propels us closer to unlocking the full potential of space exploration and satellite services.

The objective of this thesis is to implement the rendezvous of a collaborative satellite using AI algorithms, with a particular emphasis on their applications in mono camera-based visual pose estimation. The focus is specifically directed towards a detailed analysis of rendezvous operations within the 200-20cm distance range from a non-cooperative satellite. This project delves into the critical aspects of pose estimation throughout the entire trajectory of the rendezvous process, extending from the initial approach to the final berthing phase.

1.2 Thesis structure

The thesis is structured in further five chapters:

Chapter 2 - *Background*:
This chapter provides a comprehensive overview of key concepts necessary for the correct understanding of this work, with a focus on monocular camera models, perspective projection, pose estimation, and a general introduction to deep learning models.

Chapter 3 - *State-of-art*:
This chapter delves into monocular pose estimation methods, covering classic approaches like RANSAC and SfM, and exploring modern techniques such as end-to-end learning with networks like PoseNet and Mask R-CNN. The chapter also introduces feature learning, emphasizing CNN-based methods like HRNet for predicting 2D landmark locations. Moreover, some studies about spacecraft pose estimation and their use of deep learning architectures are presented. The chapter also delves into point set alignments, highlighting the widely used and advanced algorithms like Coherent Point Drift (CPD) technique employed in the method for final pose estimation.

Chapter 4 - *Algorithms and Methods*:
This chapter delves into the methodology's core algorithms and techniques. It outlines the offline architecture, detailing the 2D-3D correspondence process, landmark regression, and the neural network-based landmark mapping. The chapter then presents the online architecture, covering real-time processing and the Coherent Point Drift technique for pose estimation. Implementation challenges and dataset considerations are also discussed, providing a comprehensive overview of the applied methods.

Chapter 5 - *Implementation and Experiments*:
This chapter presents the tools and technologies employed for the project implementation and the evaluation metrics for pose estimation, Landmark Regression, Landmark Mapping are described. The chapter culminates in the assessment of both training and test datasets, showcasing the method's robustness and generalization across diverse scenarios. Overall, it provides comprehensive exploration of the research's implementation and experimentation phases.

Chapter 6 - *Discussions and Conclusions*:

The Chapter delves into challenges faced by on-board AI systems in space missions, focusing on verifiability and computational load. It emphasizes the significance of minimizing translation errors for accurate maneuvering in the proposed multi-model configuration. The section explores potential improvements, including enhanced landmark selection and strategies to fortify system robustness.

Chapter 2

State of the Art

This thesis project focuses on the concept of shared control during the teleoperation of a robotic arm. In this context it is important to define firstly the concept of teleoperation and the architecture of a telerobotic system, subsequently the shared control case.

2.1 Teleoperation

2.1.1 Teleoperation system

Teleoperation, a field at the intersection of robotics, control systems, and telecommunications, represents a advacement of the human action in scenarios where direct human involvement is impractical, hazardous, or impossible. At its core, teleoperation involves the remote control of machines, robots, or other systems, where the operator is at a significant distance from the equipment being controlled. This separation can range from a few meters, as in bomb disposal robots, to thousands of kilometers, as seen in space rovers operated from Earth.

The operational framework of teleoperation systems is rappresented by several key components:

- **the operator control unit (OCU):** The OCU serves as the interface through which the human operator interacts with the teleoperation system, typically featuring control devices like joysticks, gloves, or even more sophisticated haptic devices that provide tactile feedback, simulating the sensation of touch. This feedback is crucial, as it enhances the operator's ability to perform complex manipulations remotely by offering a semblance of physical presence at the teleoperator's location.
- **the teleoperator :** The teleoperator, equipped with various sensors, actuators, and sometimes autonomous control capabilities, executes the tasks directed by the operator. These tasks can range from simple pick-and-place operations to intricate surgical procedures, depending on the system's design and intended application. The sophistication of the teleoperator is a critical factor in the system's overall performance, particularly its ability to execute precise movements and provide feedback that accurately reflects the remote environment.
- **communication link:** Bridging the OCU and the teleoperator is the communication link, which can be wired or wireless, depending on the application's requirements and constraints. This link transmits control signals from the operator to the teleoperator and sends sensory feedback (visual, haptic, etc.) back

to the operator. The quality of this link, in terms of latency, bandwidth, and reliability, significantly impacts the system's effectiveness and the operator's ability to control the teleoperator accurately and responsively.

An overview of a telerobotic system can be expressed as follow:

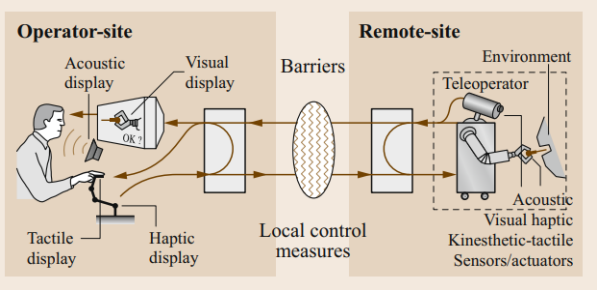


FIGURE 2.1: telerobotic system

2.1.2 Telerobotic Architecture

The architecture of a telerobotic system is a critical factor in determining the effectiveness, safety, and usability of the system. The architecture defines the interaction between the human operator and the teleoperator, the distribution of control tasks, and the level of autonomy in the system. The choice of architecture depends on the specific task requirements, the complexity of the environment, the operator's expertise, and the capabilities of the teleoperator. There are three architectures structures for telerobotic systems, each with distinct characteristics and applications:

- **Direct Control:** Direct Control is the simplest form of teleoperation, where every action taken by the human operator is directly translated into movements or actions of the teleoperator. This one-to-one correspondence means that the operator has immediate and complete control over the teleoperator's actions. Direct control systems are highly dependent on the operator's skill and attention, as there is minimal to no automation involved. This architecture is most effective in environments where precise, real-time control is necessary, and the task complexity is manageable by the human operator without additional automated support.
- **Shared Control:** Shared Control represents a more advanced teleoperation approach, where control is divided between the human operator and autonomous system functions. In this architecture, the operator and the teleoperation system share the task of controlling the teleoperator. The distribution of control can be dynamic, with the level of automation adjusting based on the task complexity, operator workload, or other contextual factors. Shared control systems are designed to combine human cognitive abilities with the precision and reliability of automated systems, optimizing task performance and potentially reducing operator fatigue. This approach is particularly beneficial in complex or unpredictable environments, where human judgment is essential, but so is the efficiency and consistency of automated processes.
- **Supervisory Control:** Supervisory Control takes a step further in integrating automation into teleoperation. In this architecture, the human operator takes on a supervisory role, monitoring the operation and intervening only when necessary. The teleoperator, equipped with a higher degree of autonomy, can perform

tasks independently based on pre-programmed instructions or algorithms. The operator's primary responsibilities include setting goals, specifying constraints, monitoring progress, and intervening in case of unexpected events or system failures. Supervisory control is well-suited for operations in highly dangerous or inaccessible environments, or for tasks that require extended periods, where continuous direct human control is impractical.

2.2 Shared Control

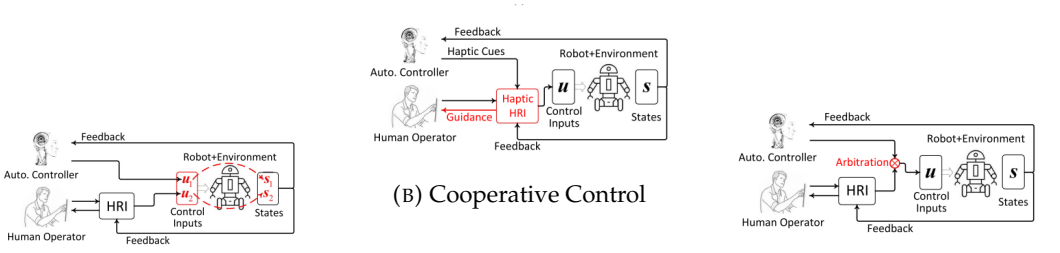
The adopted architectur to infer the teleoperation of the robotic arm is the shared control one, aiming to combine human cognitive capabilities with robotic efficiency to reduce the operator's workload and enhance task performance.
This architecture is pivotal in scenarios where fully autonomous robotic control is unfeasible due to unpredictable and complex environments.
More specifically, there are three main ways to implement the shared control architectur: Semi-Autonomous Control (SAC), State-Guidance Shared Control (SGSC), and State-Fusion Shared Control (SFSC).

- **Semi-Autonomous Control (SAC):**In SAC the division of control tasks between the human operator and the autonomous system is distinctly demarcated. In this approach, the autonomous system and the human operator control separate variables, allowing each to leverage their strengths for improved task execution. This delineation enables the human operator to focus on high-level decision-making and strategic task elements, while the autonomous system manages the execution of routine or precise actions.
  The SAC model is particularly beneficial in complex and dynamic environments where human intuition and strategic oversight are crucial for success but are complemented by the precision and reliability of autonomous robotic actions. It facilitates a collaborative interaction where the operator's cognitive load is reduced, and the efficiency and safety of operations are enhanced by offloading specific control tasks to the robotic system

- **State-Guidance Shared Control (SGSC):** In SGSC the autonomous system provides direct guidance to the human operator, which can take various forms such as visual, auditory, or haptic feedback. This guidance is designed to assist the operator in making more informed decisions or in executing tasks with greater precision.
  The key principle behind SGSC is to enhance the operator's situational awareness and to facilitate the decision-making process without overtaking the human's control. This is achieved by allowing the autonomous system to suggest or warn the operator based on the system's perception and analysis of the environment or the task at hand. SGSC strategies are particularly useful in scenarios where human intuition and judgment are critical, but where the complexity or danger of the task necessitates additional inputs that can augment human capabilities. By providing guidance, SGSC aims to reduce the cognitive load on the operator, improve task performance, and enhance safety without diminishing the human's role in the control loop

- **State-Fusion Shared Control (SFSC):** SFSC strategy goes beyond merely dividing tasks or providing guidance; it integrates the intentions and control signals of the human operator and the robot through a fusion mechanism. The fusion

process often involves sophisticated algorithms that can weigh the contributions of the human and the robot based on factors like task complexity, operator expertise, and system capabilities.
The goal of SFSC is to harness the strengths of both human and machine: the adaptability, judgment, and intuition of the human, alongside the precision, reliability, and speed of the machine. This integrated approach allows for a more seamless and efficient execution of tasks, particularly in environments that are too complex for fully autonomous robotics or require a level of decision-making beyond current autonomous capabilities. By dynamically adjusting the level of influence between human and machine based on real-time feedback and task requirements, SFSC strategies ensure that the control system remains adaptable and responsive to the nuances of each specific operation. This not only improves the performance and safety of telerobotic systems but also enhances the user experience by creating a more intuitive interaction between the human operators and the robotic system.



(A) Semi-Autonomous Control

(B) Cooperative Control

(C) Supervised Autonomous Control

FIGURE 2.2: Types of Shared Control

In the context of this project the share controll architecture is implemented through the SFSC strategy. Since the aim is to help the teleoperator during teleoperation, this strategy ensures that the human operator and the robotic system can work together seamlessly, leaving the operator in control of the task while providing the necessary support and assistance to enhance task performance and safety.
To properly implement the strategy, the problem has to suibdivided into three main compartements: intention recognition, guidance and autonomous control.

2.3 Intention Recognition

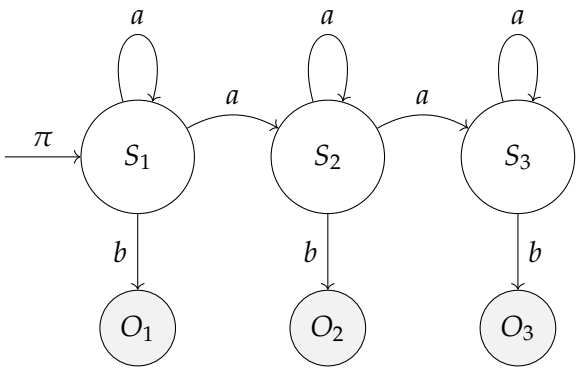
The core challenge of intention recognition lies in accurately inferring the user's goals from a set of possible actions, often in real-time and under uncertainty. This process involves the collection and analysis of data generated by human actions, which could be in the form of physical movements, verbal commands, or control signals. The ultimate aim is to enable robots or autonomous systems to anticipate the needs or next actions of their human partners, allowing for proactive assistance.

2.3.1 Metodologies

Intention recognition methodologies span a range of approaches, reflecting the diversity of applications and the complexity of interpreting human intent.
At this day, there isn't a single, universally accepted method for intention recognition that applies across all scenarios in the teleoperation of robotic arms. Instead, the field

comprises a variety of techniques, each with its strengths and limitations, tailored to specific contexts, tasks, and environments

- **Probabilistic Models:** Techniques such as Hidden Markov Models (HMMs), as extensively explained in [3] and [1],is a statistical model that represents systems with probabilistic properties. It is characterized by its ability to model systems where the states are hidden and not directly observable, but the outcomes or observations associated with these states are observable.
  An HMM is defined by three primary elements: the state transition probability matrix **A**, which specifies the likelihood of transitioning from one state to another; the observation probability matrix **B**, detailing the probability of observing a certain symbol when in a specific state; and the initial state probability vector π, indicating the likelihood of the system starting in each state.



Legend:
S: State
O: Observation
a: Transition Probability
b: Observation Probability
π: Initial State Probability

FIGURE 2.3: Representation of a Hidden Markov Model (HMM)

As outlined in , while the application of HMMs in intention recognition for teleoperated robotic arm manipulation offers notable advantages in handling temporal dynamics and model flexibility, it concurrently presents substantial challenges, particularly in the necessity for precise and well-documented a priori knowledge. The complex nature of human decision-making processes complicates the task of accurately predicting operator behavior at each timestep of the teleoperation task; this is evident in the methodology, where the transition matrix **A** is constructed based on empirical observations. This approach results in a relatively simplistic policy for state transitions, potentially oversimplifying the nuanced and variable nature of human intent during teleoperation tasks.

- **Inverse Reinforcement Learning (IRL):** Inverse Reinforcement Learning (IRL) is a sophisticated method for deciphering and forecasting the intentions behind human operators' actions in teleoperation, where understanding complex decision-making is key to enhancing human-robot interactions.
  Brian D. Ziebart et al. [14] extend IRL's capabilities through a probabilistic framework grounded in the maximum entropy principle, adept at navigating the uncertainties in teleoperation decision-making. This approach, by favoring

the least biased action distribution based on observed behaviors, provides a refined representation of operators' reward functions, thus accurately modeling their intent.
The incorporation of maximum entropy IRL into intent recognition significantly improves how robotic systems can preemptively adjust to and fulfill human operator goals, shifting from merely reactive to proactive adaptations. Despite these advancements, implementing IRL in real-time applications is hindered by the extensive computational resources needed to process complex data and infer human intent in a fast and accurate manner, presenting a challenge in the practical deployment of IRL-enhanced robotic systems.

- **Partially Observable Markov Decision Processes (POMDPs):** Partially Observable Markov Decision Processes (POMDPs) are a class of decision-making models that consider situations where the state of the system is partially observable or uncertain to the decision-maker. In a POMDP framework, the decision-maker must rely on observations that may provide indirect or incomplete information about the system's state to make decisions. This model extends the Markov Decision Process (MDP) framework by incorporating a layer of uncertainty regarding the system's actual state, making it particularly suited for complex environments where full information is not available.
  In the context of shared control systems, POMDPs play a crucial role in modeling the interaction between a human user and an autonomous system, especially when the system's understanding of the user's intent is uncertain. the authors in [8] delve into this by formalizing shared autonomy as a POMDP, which assists in minimizing the expected cost-to-go with an unknown goal. This approach acknowledges that while the autonomous system may not confidently predict the user's specific goal until it is nearly achieved, it can still offer valuable assistance by optimizing actions that are generally helpful across multiple potential goals. When the system has high confidence in a single user goal, the framework focuses assistance more narrowly to support that specific objective. This balance allows for more effective and adaptive assistance, even in the face of uncertainty about the user's exact intentions.
  Despite its great ability to handle uncertainty, give focused help based on goals, and use hindsight optimization to make real-time processing better, this method still encounters problems with how complex its computations are, requiring substantial computational resources to function effectively. Additionally, the performance of this approach is deeply interconnected with the precision of its observational models for accurately perceiving and interpreting actions. Inaccurate models significantly impair the system's capability to discern the user's intentions, leading to challenges in providing timely and relevant assistance. Therefore, it is crucial to enhance the accuracy of these models to ensure they accurately reflect real-world scenarios, thereby optimizing the method's efficiency

- **Recurrent neural networks:** Recurrent Neural Networks (RNNs) are a specialized type of artificial neural networks crafted to recognize and interpret patterns in sequential data, including but not limited to text, genomic sequences, handwriting, or numerical time series from various sensors. What sets RNNs apart from traditional feedforward neural networks is their unique architecture featuring directed cycles in their connections. This design allows them to retain information over time, enabling the network to maintain a form of 'memory'. Such a capability is invaluable for tasks where understanding the context or the

sequence in which data points appear is crucial. By iterating through elements in a sequence and maintaining a state that accumulates information seen thus far, RNNs effectively process sequences by leveraging their internal state (memory) to manage a range of inputs sequentially. This characteristic renders them incredibly useful for a variety of applications, including language modeling, speech recognition, and forecasting in time series data. Despite their advantages, RNNs are not without their challenges. Notably, they can be difficult to train effectively due to issues like vanishing and exploding gradients, which hinder their ability to learn long-range dependencies within the data.
As elaborated in [13] to overcome these obstacles, Long Short-Term Memory (LSTM) networks emerge as a powerful solution within the domain of shared control of robotic arms, especially for tasks like intention prediction. The LSTM, with its unique ability to learn and remember over long periods, has significantly improved the predictability and accuracy of user intention in shared control systems. This advanced form of RNN addresses the core limitations of its predecessors by efficiently handling long-term dependencies. The robustness of Long Short-Term Memory (LSTM) networks against time gaps in data input, coupled with their capacity to adapt to individual user preferences, significantly boosts the efficiency and personalization of shared control systems. However, the sophisticated nature of LSTMs brings about challenges, including increased computational complexity and a greater need for extensive training data. Despite these challenges, the integration of LSTMs into Recurrent Neural Network (RNN) technology marks a significant leap forward. It expands the potential for more intuitive and adaptable human-robot collaboration within shared control environments, navigating through the practical difficulties associated with their implementation.

2.4 Guidance

In the domaine of shared control systems, guidance embodies a sophisticated integration of feedback and control strategies, facilitating seamless and intuitive interactions between humans and robots. At the heart of guidance lies the objective to harmonize the decision-making capabilities and adaptability of human operators with the accuracy, reliability, and operational efficiency of robotic systems. This harmony is achieved through a continuous exchange of information, where the system furnishes the human operator with immediate, pertinent data, feedback, and actionable advice. This enables the operator to make well-informed decisions that guide the robot's actions. Such a reciprocal flow of information significantly enhances the effectiveness of task performance, elevates safety standards, and guarantees an advanced level of control and adaptability. This acts as a critical link between human operators and automated systems, effectively reducing the workload.

2.4.1 Metodologies

Guidance methodologies in shared control systems showcase a broad spectrum of approaches, highlighting the complex interplay between human operators and robotic mechanisms, especially in teleoperation scenarios;among these, Artificial Potential Fields (APF), the Dynamic Window Approach (DWA), the Vector Field Histogram

(VFH), and Rapidly-exploring Random Trees (RRT) have been proven to be particularly effective in guiding robotic systems in shared control environments.

- **Artificial Potential Fields (APF):** The implementation of Artificial Potential Fields (APF) [11] is integral to enhancing real-time shared control in teleoperation frameworks, focusing on obstacle avoidance and efficient path planning. APF employs virtual forces that guide the robotic system by creating a potential field where obstacles generate repulsive forces, and the goal exerts an attractive force. This dynamic allows the robot to navigate by following the gradient of the potential function, akin to traversing a landscape of peaks and valleys designed to steer clear of obstacles while being drawn toward the target. Upon setting a goal, APF aligns the robot's movement with the operator's intent by modulating these virtual forces, thereby facilitating an intuitive interaction between the human operator and the robotic system.
  The primary challenges associated with APF include the robot's susceptibility to getting trapped in local minima—regions where the potential gradient is zero, preventing further movement towards the goal—and the method's purely reactive nature to collision avoidance. In standard APF applications, the robot alters its course only when in close proximity to obstacles, which may not be efficient for advanced navigation and obstacle avoidance in cluttered environments.

  However, as presented in [6], the problem of obstacle avoidance in cluttered environments can be efficiently overcome with the dynamic generation of escape points around obstacles. These escape points are designed to help the robotic system bypass obstacles smoothly while avoiding the pitfalls of local minima.
  This approach not only addresses the issue of the robot getting stuck but also allows for more proactive obstacle avoidance by modifying the robot's trajectory in advance, rather than merely reacting when close to obstacles.

- **Dynamic Window Approach (DWA):** he Dynamic Window Approach (DWA) is a fundamental technique in mobile robotics, utilized for motion planning and obstacle avoidance in dynamic environments. It operates by constraining the robot's velocity to a "dynamic window," considering its current state and capabilities. Within this window, potential velocities are evaluated using a cost function that incorporates factors such as obstacle distance, goal direction, and robot velocity. By selecting the velocity that maximizes this function while ensuring collision-free navigation, the DWA facilitates rapid decision-making in real-time motion control.
  To fit this method to a shared control contex, in [2] a novelty method has been developed. Building upon DWA principles, the Biomimetical Dynamic Window Approach (BDWA) integrates human navigation behaviors into robotic motion planning. This extension aims to generate robot movement resembling human path preferences, enhancing user interaction with assistive devices.
  BDWA achieves this by selecting velocities aligning with human-like paths, guided by a reward function assessing trajectory resemblance. Through iterative refinement, BDWA balances safe navigation with human-like movement, optimizing the shared control experience in assistive technologies.

Chapter 3

Methods

In this chapter we will discuss the methods used to achieve the objective of the research and why they are the best suited for the preveiously explained data

3.1 Human Intent Recognition

As mentioned in the previous chapter, the most prominent features of human intent recognition were the two main categories of data, the first being the distance from the object and the second one beeing the direction of the end effector. Another importsnt feature is the time, as the data is collected over time, it is important to take into account the time series nature of the data. Given the nature of the data a valid approach could've been the one used in the paper , using HMM.
Although valid, having as trainable parameter only the rate parameter λ of the exponential distribution, the introduction of hand tunable coefficients as the weight parameters for the observation probability, the model will result biased and will simplify the complex nuances of the human intention. Since the need to comprehend the intrsinc connection between distance and direction and how this relation varies in time, a more sophidticated model was needed.
A novel approach in intent recognition is to use Recurrent Neural Networks, more specifically Long Short Term Memory Networks, a type of RNN that is capable of learning long-term dependencies.

3.1.1 Recurrent Neural Networks

A recurrent neural network (RNN) is a specialized form of artificial neural network designed to handle time series data or sequential data. Unlike standard feedforward neural networks, which assume that data points are independent of each other, RNNs are tailored for scenarios where the relationship between sequential data points is crucial. In such cases, where one data point is dependent on the previous ones, the neural network architecture must be adapted to capture these dependencies. RNNs achieve this through the concept of "memory," enabling them to retain information from previous inputs to inform the generation of subsequent outputs in the sequence.

- $x_t \in \mathbb{R}$ denotes the input at time step t. For simplicity, x_t is assumed to be a scalar with a single feature. This concept can be generalized to a d-dimensional feature vector.
- $y_{t+1} \in \mathbb{R}$ represents the output of the network at time step $t+1$.
- $h_t \in \mathbb{R}^m$ stores the values of the hidden units or states at time t, also known as the current context.

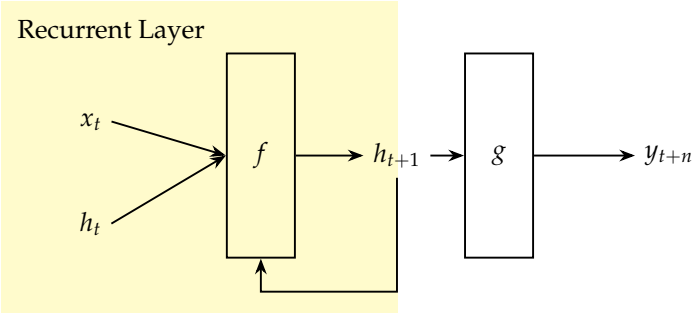


FIGURE 3.1: Recurirsive Neural Network (RNN)

- $h_{t+1} \in \mathbb{R}^m$ denotes the hidden state at time $t+1$

At every time step, the network can be unfolded for k time steps to obtain the output at time step $k+1$. This unfolded network is akin to a feedforward neural network. The rectangle in the unfolded network indicates an operational sequence. An example of single layer RNN is shown in the figure below:

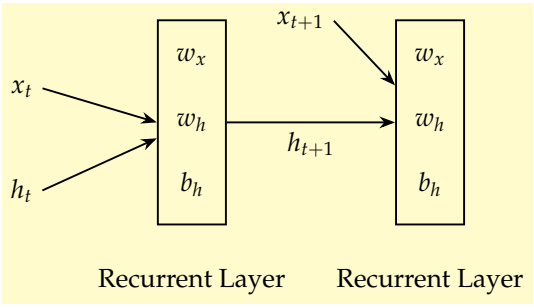


FIGURE 3.2: Recurirsive Neural Network single layer

- $w_x \in \mathbb{R}^m$ are weights associated with inputs in the recurrent layer
- $w_h \in \mathbb{R}^{m \times m}$ are weights associated with hidden units in the recurrent layer
- $w_y \in \mathbb{R}^m$ are weights associated with hidden units to output units
- $b_h \in \mathbb{R}^m$ is the bias associated with the recurrent layer
- $b_y \in \mathbb{R}$ is the bias associated with the feedforward layer

For instance, with an activation function f, the next hidden state is computed as:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x x_t + w_h h_t + b_h)$$

The output y_t at time t is determined by:

$$y_t = f(h_t, w_y) = f(w_y \cdot h_t + b_y)$$

Here, $\cdot$ signifies the dot product.
Consequently, during the feedforward pass of an RNN, the network computes the values of the hidden units and the output after k time steps. The network's weights are temporally shared. Each recurrent layer possesses two sets of weights: one for the input and another for the hidden units. The last feedforward layer, which calculates

the final output for the kth time step, resembles a conventional layer in a traditional feedforward network.

The Activation Function Any activation function may be employed within the recurrent neural network. Commonly used functions include:

- Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Tanh function: $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- ReLU function: $\text{ReLU}(x) = \max(0, x)$

Training and Vanishing Gradient Training neural networks involves adjusting their weights to minimize the difference between the predicted output and the actual target values. For networks that process sequential data, like time series or natural language, this training must account for the temporal dependencies within the data. Recurrent Neural Networks (RNNs) are specifically designed for this purpose, capable of maintaining state across sequential inputs through their looping architecture. However, training RNNs presents unique challenges due to their recurrent nature, necessitating specialized techniques to effectively learn from sequences.

- **Unrolling the RNN Through Time :**

  To apply BPTT, we first unroll the RNN across time, transforming it into an equivalent deep feedforward network. This unrolled structure maps each time step to a distinct layer in the network, facilitating the application of backpropagation as in traditional neural networks. The unrolled RNN can be mathematically represented as:

  $$\begin{aligned} &x_t : \text{Input at time step } t, \\ &h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) : \text{Hidden state at time step } t, \\ &o_t = g(W_{ho}h_t + b_o) : \text{Output at time step } t, \end{aligned}$$

  where f and g denote non-linear activation functions, W_{hh}, W_{xh}, and W_{ho} are weight matrices, and b_h, b_o are bias vectors.

- **Forward Pass :**

  During the forward pass, we sequentially compute the activations and outputs for each time step, from $t = 1$ to T, where T is the sequence length. This process mirrors the execution of a feedforward neural network but incorporates temporal dependencies through the hidden states.

- **Computing the Cost:**

  Following the forward pass, the network's performance is evaluated using a cost function C, which measures the discrepancy between the predicted output sequence $\{o_1, o_2, ..., o_T\}$ and the target sequence. The choice of C is dependent on the task, with Mean Squared Error (MSE) and Cross-Entropy being common choices for regression and classification tasks, respectively.

- **Backward Pass (BPTT) :**

BPTT involves the backward propagation of the cost function's gradients through the unrolled network, extending traditional backpropagation to account for the network's temporal structure. The gradients of C with respect to the weights, such as $\frac{\partial C}{\partial W_{hh}}$, $\frac{\partial C}{\partial W_{xh}}$, and $\frac{\partial C}{\partial W_{ho}}$, are computed using the chain rule and aggregated over all time steps, facilitating the update of weights based on their influence on future outputs.

- **Weight Update :**

  The final step in BPTT is updating the network's weights using the computed gradients. This is typically done using optimization algorithms like Stochastic Gradient Descent (SGD) or Adam, with update equations of the form:

$$W = W - \eta \frac{\partial C}{\partial W},$$

  where W represents the weight matrices (W_{hh}, W_{xh}, W_{ho}) and η is the learning rate.

A significant challenge in the Backpropagation Through Time (BPTT) process is the phenomenon referred to as the *vanishing gradient problem*. This issue predominantly arises in scenarios involving the processing of long time series data. As the gradient of the cost function with respect to the network's weights is propagated backward through time, its magnitude diminishes exponentially. This attenuation is a consequence of the repeated multiplication of gradients at each timestep, which are often less than one in magnitude. Consequently, this results in exceedingly small gradient values as one moves further back in time. The direct implication of this phenomenon is the negligible adjustment of weights corresponding to the earlier portions of the input sequence, thereby leading to slow convergence rates and a pronounced difficulty for the network in learning dependencies across extended sequences.

Vaniscing Gradient Problem The core of the issue comes from the characteristics of certain activation functions, notably the sigmoid function and the hyperbolic tangent function (tanh). The sigmoid function, represented as $\sigma(x)$, and the hyperbolic tangent function, denoted as $\tanh(x)$, are mathematically articulated as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{3.1}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{3.2}$$

effectively squeezing an extensive range of input values into a narrow band between 0 and 1, for the sigmoid, and -1 to 1, for the tanh .

This compression of input values means that large variations in input can result in disproportionately small changes in output. Consequently, the derivatives of these functions, crucial for the backpropagation algorithm, are significantly small in magnitude. For both functions, this effect is more pronounced for inputs of large magnitude, where the slope of the curve flattens out, leading to derivatives that approach zero.

The sigmoid function is characterized by its distinctive S-shaped curve, serving as a pivotal nonlinear activation function in neural networks.

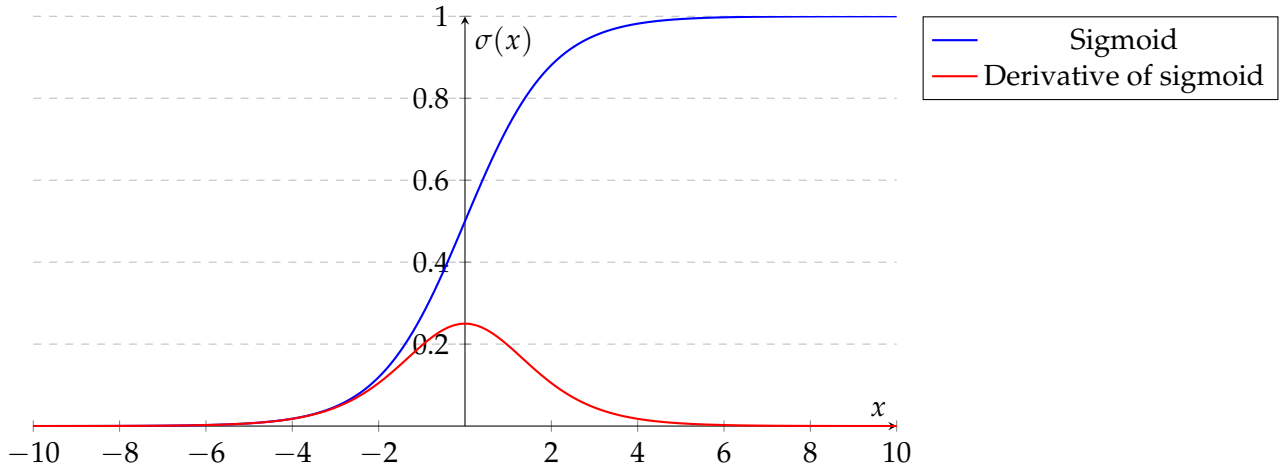


FIGURE 3.3: The sigmoid function and its derivative

One of its notable features is the relationship between the input magnitude and the rate of change in the output. As the absolute value of the input escalates, the incremental change in the output significantly diminishes, leading to a derivative that approaches zero.

Mathematically, the derivative of the sigmoid function, denoted as $\sigma'(x)$, can be expressed as:

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)), \tag{3.3}$$

where $\sigma(x)$ is the sigmoid function itself.

In networks with a shallow architecture and fewer layers employing these activations, the vanishing gradient is less of an issue. However, the problem escalates with increased network depth, making the gradient too small for effective training.

3.1.2 Long Short-Term Memory (LSTM) Networks

LSTMs, as discussed by A. Graves et al. [7], were designed to specifically address the challenges encountered with traditional RNNs, notably the vanishing gradient problem. Unlike standard RNNs, LSTMs incorporate a sophisticated gating mechanism to control the flow of information and gradients within the network.
This design significantly mitigates the vanishing gradient issue, enabling the network to learn from and remember information over extended sequences more effectively.

The figure below illustrates the internal structure of an LSTM cell, showcasing its unique components:

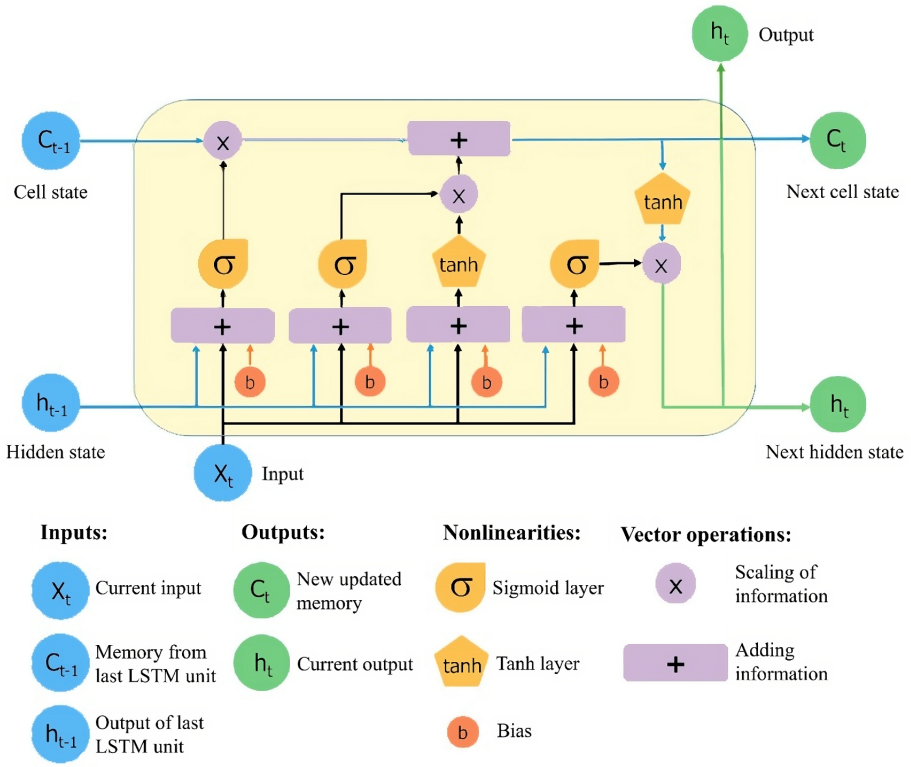


FIGURE 3.4: LSTM Cell Structure

LSTM cells include three critical gates: the input gate, the forget gate, and the output gate. These gates collectively manage the cell's memory, determining what information to retain, discard, and output.
This gating mechanism is pivotal for the LSTM's ability to modulate the gradient flow during backpropagation, directly addressing the vanishing gradient problem by maintaining gradient magnitude across long sequences. The LSTM cell's core components are as follows:

- **Forget Gate :**The forget gate plays a vital role in the LSTM architecture by deciding which information from the cell state should be kept or forgotten. It employs a sigmoid function to generate values between 0 and 1, with these values indicating the extent to which information is preserved or discarded.
  This selective memory process enables LSTMs to focus on relevant information while minimizing unnecessary data retention, aiding in the preservation of gradient significance.
- **Input Gate and Output Gate :**The input gate is responsible for updating the cell state with new information. It operates in two steps: a sigmoid layer determines which values will be updated, and a tanh layer generates new candidate values. Conversely, the output gate decides what the next hidden state should be, which includes the filtered cell state information to be used in output.
  These gates ensure the network's ability to capture essential details for the task at hand, enhancing its long-term dependency modeling capabilities.

The operation of both the input and output gates ensures that if the outcome of these gates is close to 1, gradients can pass through without significant attenuation.Conversely, outcomes near 0 block the gradient flow.
This dynamic allows LSTMs to mitigate the vanishing gradient problem effectively.

- **Memory Cell State :**The memory cell state is the core component that allows LSTMs to retain information across time steps. It integrates inputs from the current time step and previous cell state, facilitating the network's ability to maintain a continuous flow of relevant information through the sequence.
  The additive nature of the cell state updates helps to sustain gradient flow across long sequences, ensuring that LSTMs can capture and leverage long-range dependencies in the data.

- **Forget Gate:** The forget gate plays a vital role in the LSTM architecture by deciding which information from the cell state should be kept or forgotten. It employs a sigmoid function to generate values between 0 and 1, with these values indicating the extent to which information is preserved or discarded.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

  This selective memory process enables LSTMs to focus on relevant information while minimizing unnecessary data retention, aiding in the preservation of gradient significance.

- **Input Gate:** The input gate is responsible for updating the cell state with new information. It operates in two steps: a sigmoid layer determines which values will be updated, and a tanh layer generates new candidate values.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  Conversely, the output gate decides what the next hidden state should be, which includes the filtered cell state information to be used in output.

- **Cell State Update:** The memory cell state is the core component that allows LSTMs to retain information across time steps. It integrates inputs from the current time step and previous cell state, facilitating the network's ability to maintain a continuous flow of relevant information through the sequence.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

  The additive nature of the cell state updates helps to sustain gradient flow across long sequences, ensuring that LSTMs can capture and leverage long-range dependencies in the data.

- **Output Gate:** Determines the next hidden state, which includes the filtered cell state information to be used in outputs. The operation of the output gate ensures that gradients can pass through effectively, making it a key component in mitigating the vanishing gradient problem.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

3.1.3 LSTM Intent Recognition Architecture

The application of theoretical approaches to intent recognition necessitates an investigation into various methodologies, with a particular focus on Long Short-Term Memory (LSTM) networks. As detailed by Shao-Wen Wu et al. [12], LSTM networks excel at understanding the complex patterns in data indicative of human intent.
This research employs LSTM networks to estimate the parameters of a Beta distribution, demonstrating the networks' proficiency in accurately predicting human intentions, even in scenarios of incorrect teleoperation or obstacle avoidance, thus underscoring the model's dependability.

Despite the effectiveness and reliability of this approach, it is not devoid of limitations. A significant constraint is the assumption that objects remain stationary within their initial training environment.
Any movement of these objects outside the direct view of the camera compromises the reliability of the alpha and beta parameters of the Beta distribution, leading to diminished accuracy in predictions.

In response to these impediments,in this thesis, the network will be assume the role of a binary classifier. This model, through the analysis of the end effector's distance and direction relative to an object, determines the likelihood of the object being the intended target for grasping.
Such an approach not only permits the LSTM to discern the complex dynamics between the input and output parameters, thereby enhancing its generalization capabilities, but also enables the adaptation of the model to accommodate the displacement of objects within the spatial domain without necessitating retraining.

Furthermore, the introduction of modularity into the model represents a significant advancement; by dedicating a distinct LSTM to each object, as show in figure ... the framework achieves scalability, allowing for its replication across multiple objects within a scene.

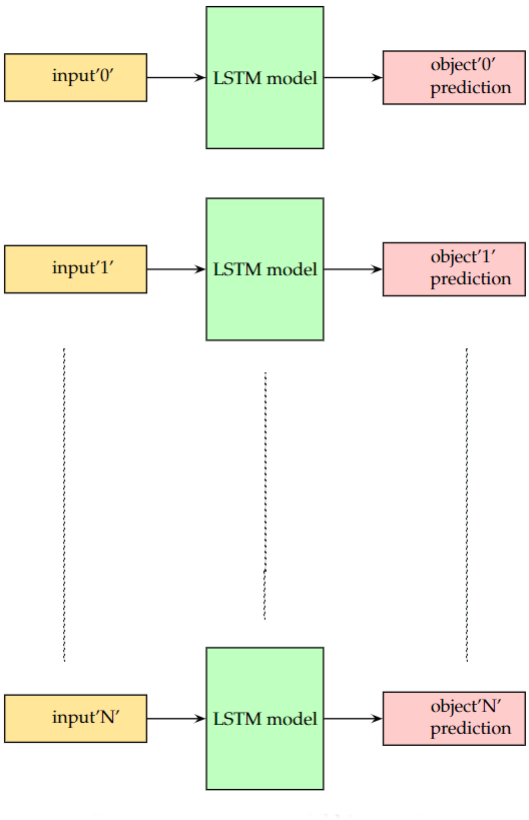


FIGURE 3.5: LSTM Model Modularity

This characteristic ensures the model's viability in real-world scenarios, which are typified by the simultaneous presence of multiple objects, though it also necessitates an acknowledgment of the potential limitations posed by environmental clutter and computational resource constraints.

Central to this enhanced methodological approach is the objective of minimizing the model's inherent bias, thereby granting the LSTM the autonomy to independently identify and learn the subtle interrelations between distance and direction and their impact on object identification. This capacity for autonomous learning is imperative for the development of an intent recognition system that is not only efficient and adaptable but also robust enough to operate effectively within the complexities of real-world environments.

Below a schematic of the LSTM model architecture is shown:

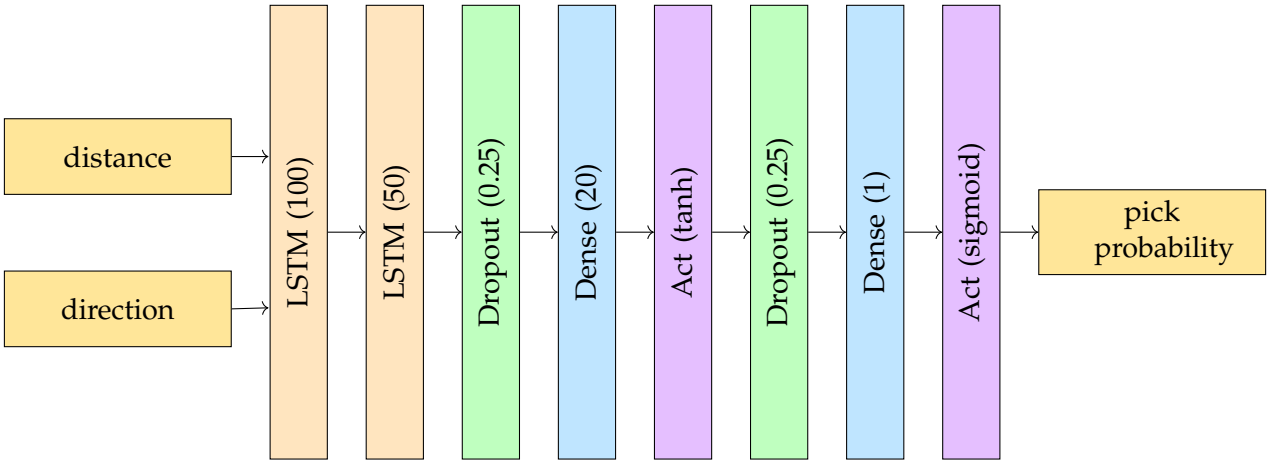


FIGURE 3.6: LSTM Model Architecture

To arrive a this final model, it was vital to carfelly plan evry step of the process, from the pre-processing of the data to modelling of each layer of the network.
since a minor change in the architecture of the network can lead to a significant change in the performance of the model, it was important to carefully plan each step of the process.

Pre-Processing of the Data The initial step in developing the model involved setting the number of neurons for the first LSTM layer. The LSTM, being a recurrent neural network, processes inputs sequentially through its repetitive cell structure. This means each cell within the layer handles a distinct input from the time sequence; specifically, as showcased in figure , the first cell processes the first input, the second cell the second input, and so on.

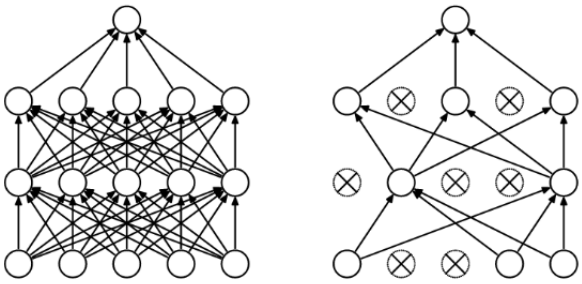
Selecting the appropriate number of cells in this layer is crucial. This choice is one of the first and most significant hyperparameters to be determined, directly impacting the model's capacity and computational demands. Once this hyperparameter is established, it sets a foundational aspect of the model's architecture that is not easily modified later.

Analysis of data from 26 teleoperation experiments revealed an average of 2820 timesteps per experiment, with each experiment lasting around 26 seconds on average. Given this data, and to efficiently balance model complexity with the available computational resources, the neuron count for the first layer was set to 100.

Stacked LSTM Recent studies on the application of Long Short-Term Memory (LSTM) networks to time series data, highlighted by [5] and [9], have revealed the efficacy of employing stacked LSTM layers in enhancing model performance. This design approach facilitates hidden layers in capturing progressively higher-level representations of the sequence data. Deep LSTM models, characterized by multiple LSTM hidden layers, create a structured flow where the output of one layer directly feeds into the subsequent layer, thereby significantly boosting the neural network's (NN) capability to learn.

The hyperparameter configuration for modeling complex relationships was determined to be a two-layer LSTM structure. This choice was guided by the recognition that, although additional layers may enhance the model's ability to identify complex relationships among feature variables, a point is reached where further layers contribute to diminishing returns and increased risk of overfitting. Considering the dataset characterized by a relatively small number of input timesteps, incorporating a third layer would likely lead to overfitting, undermining the model's generalizability and efficiency.

Other layers To further mitigate the risk of overfitting in models with multiple layers, such as stacked LSTM networks, dropout layers have been introduced. This method involves randomly removing a specified proportion of neurons, as determined by the dropout ratio, from the network during each training iteration, effectively preventing their contributions to downstream neuron activation and halting weight updates during backpropagation for that iteration. For this experiment, the dropout ratio was set to 0.25, meaning 25% of neurons in a layer are ignored at each training step, reducing the model's dependence on specific neurons and encouraging the learning of more robust features through forced feature redundancy. This approach not only helps in spreading out weights to diminish the likelihood of overfitting but also ensures the model does not rely too heavily on a small set of neurons, thereby improving its ability to generalize to unseen data. It is crucial to note that dropout is applied exclusively during training, with all neurons active during testing or evaluation phases, although their outputs are adjusted by the dropout rate to maintain consistency with the training environment



(a) Standard Neural Net (b) After applying dropout.

FIGURE 3.7: Effect of Dropout layer

Another crucial part of the model, is th Fully-Connected layer, often referred to as a Dense layer in neural networks, which ensures complete interconnectivity between the neurons of consecutive layers. In such a layer, each neuron from the previous layer is linked to every neuron in the next, facilitating the layer's ability to derive a comprehensive linear combination of the input features. This configuration is instrumental in the network's capability to discern complex patterns within the depth of neural network models.
As seen in the stacked LSTM, which saw a reduction of cell from 100 to 50, the first Dense layer was configured to contain 20 neurons. This configuration aims to refine the feature space, enabling the network to concentrate on the most significant patterns identified by the LSTM layers, culminating in a final output layer comprising a single neuron. Both the LSTM and Dense layers employ the *tanh* activation function.
The *tanh* function is frequently chosen for LSTM layers due to its effectiveness in

capturing complex patterns by introducing non-linearity into the network. Non-linearity is essential, as a purely linear model would be incapable of recognizing the intricate patterns present within the data.

Given the model's objective to classify the intention to pick or not pick an object based on the input, the task falls within the scope of a binary classification problem. Therefore, the architecture necessitates the inclusion of a Dense layer as the final component, distinguished by a single neuron utilizing a 'sigmoid' activation function. The 'sigmoid' function, characterized by its output range extending from 0 to 1, permits this concluding layer to map the features refined by antecedent layers into a predictive probability. This calculated probability serves to articulate the model's confidence in categorizing the input under one of two potential outcomes: the action to pick or the decision against picking the object.

Implementation of the model To deploy the LSTM model, it was done in python, more specifically using the keras library Keras [**chollet2015keras**] is a Python-based, open-source library designed to facilitate the development and training of deep learning models efficiently. Acting as an interface for TensorFlow, it simplifies complex processes with predefined layers and algorithms for rapid experimentation. Emphasizing user-friendliness and modularity, Keras supports various backend engines, operates on both CPUs and GPUs, and enables quick prototyping with minimal code. This makes it accessible to users at all levels in the fields of machine learning and artificial intelligence.

'CODE?'

3.1.4 Training the model

Appendix A

Support Code

A.1 Ground Truth Heatmaps

```
import numpy as np

def createHeatmap(landmark, vp, hmap_w, hmap_h, sig=1):
    hmap = np.zeros((hmap_height + 3, hmap_width + 3))
    x, y = landmark
    if vp:
        for i in range(y - 3*sig, y + 3*sig):
            for j in range(x - 3*sig, x + 3*sig):
                hmap[i,j]+=np.exp(-((i-y)** 2+(j-x)**2)/(2*sig**2))

    hmap = hmap[1:-2, 1:-2]

    return hmap

def coord2Heatmap(landmarks, vis, hmap_w=512, hmap_h=512, sig=1):
    hmaps = []

    for landmark, vp in zip(landmarks, vis):
        hmaps.append(createHeatmap(landmark,vp,hmap_w,hmap_h,sig))

    hmaps = np.array(hmaps).squeeze()

    return hmaps
```

LISTING A.1: Ground Truth Heatmaps

A.2 Landmark Location Selection

```
import numpy as np

def get_landmarks(landmarks_images, threshold1=-0.5, threshold2=-0.6):
    landmarks2D = []
    for img in landmarks_images:
        Lnd_found = False
        lndx = -1
        lndy = -1
        mask1 = img < threshold2
        img[mask1] = -1

        if np.max(img) > threshold1:
            Lnd_found = True
        elif np.max(img) > threshold2 and np.var(img) > 2e-6:
            Lnd_found = True

        if Lnd_found:
            p_lndy,p_lndx = np.where(img == np.max(img))
            lndx = np.round(np.mean(p_lndx))
            lndy = np.round(np.mean(p_lndy))
            if landmarks2D is not None or len(landmarks2D) > 0:
                for data in landmarks2D:
                    if data[2] == 1 and abs(lndx - data[0]) <= 3
                        and abs(lndy - data[1]) <= 3:
                        if np.max(img)>data[3] and np.var(img)>data[4]:
                            data[0] = -1
                            data[1] = -1
                            data[2] = 0
                            n_lnd=[lndx,lndy,1,np.max(img),np.var(img)]
                        else:
                            n_lnd = [-1,-1,0,np.max(img),np.var(img)]

            n_lnd = [lndx,lndy,1,np.max(img),np.var(img)]
        else:
            n_lnd = [-1,-1,0,np.max(img),np.var(img)]

        landmarks2D.append(n_lnd)

    return np.array(landmarks2D)
```

LISTING A.2: Landmark Location Selection

Bibliography

[1] Daniel Aarno and Danica Kragic. "Motion intention recognition in robot assisted applications". In: *Robotics and Autonomous Systems* 56.8 (2008), pp. 692–705. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2007.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0921889007001704.

[2] Joaquin Ballesteros et al. "A Biomimetical Dynamic Window Approach to Navigation for Collaborative Control". In: *IEEE Transactions on Human-Machine Systems* 47.6 (2017), pp. 1123–1133. DOI: 10.1109/THMS.2017.2700633.

[3] Sylvain Calinon et al. "Learning and Reproduction of Gestures by Imitation". In: *IEEE Robotics and Automation Magazine* 17.2 (2010), pp. 44–54. DOI: 10.1109/MRA.2010.936947.

[4] Coppelia Robotics. *CoppeliaSim*. https://www.coppeliarobotics.com/. [(cit. on p. 31)]. 2023.

[5] Zhiyong Cui et al. *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*. 2019. arXiv: 1801.02143 [cs.LG].

[6] Alberto Gottardi et al. "Shared Control in Robot Teleoperation With Improved Potential Fields". In: *IEEE Transactions on Human-Machine Systems* 52.3 (2022), pp. 410–422. DOI: 10.1109/THMS.2022.3155716.

[7] Alex Graves and Alex Graves. "Long short-term memory". In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[8] Shervin Javdani et al. "Shared autonomy via hindsight optimization for teleoperation and teaming". In: *The International Journal of Robotics Research* 37.7 (2018), pp. 717–742. DOI: 10.1177/0278364918776060. eprint: https://doi.org/10.1177/0278364918776060. URL: https://doi.org/10.1177/0278364918776060.

[9] Yangtao Li et al. "The Prediction of Dam Displacement Time Series Using STL, Extra-Trees, and Stacked LSTM Neural Network". In: *IEEE Access* 8 (2020), pp. 94440–94452. DOI: 10.1109/ACCESS.2020.2995592.

[10] Universal Robots. *UR5 User Manual*. Cited on pages 36, 41, 42.

[11] C. Warren. "Global path planning using artificial potential fields". In: *1989 IEEE International Conference on Robotics and Automation*. Los Alamitos, CA, USA: IEEE Computer Society, May 1989, pp. 316, 317, 318, 319, 320, 321. DOI: 10.1109/ROBOT.1989.100007. URL: https://doi.ieeecomputersociety.org/10.1109/ROBOT.1989.100007.

[12] Shao-Wen Wu. "Predicting user intent during teleoperation using neural networks". MA thesis. 2019.

[13] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (July 2019), pp. 1235–1270. ISSN: 0899-7667. DOI: 10.1162/neco_a_01199. eprint: https://direct.mit.edu/neco/article-pdf/31/7/1235/1053200/neco_a_01199.pdf. URL: https://doi.org/10.1162/neco%5C_a%5C_01199.

[14] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

Acknowledgements

Desidero esprimere i miei più sentiti ringraziamenti a tutte le persone che hanno contribuito al completamento di questa tesi.

Innanzitutto, desidero ringraziare il mio relatore, il prof. Marcello Chiaberge, e co-relatore, l'ing. Andrea Merlo, per avermi dato la possibilità di intraprendere questo progetto. Ringrazio l'ing. Marco Lapolla per la sua guida, disponibilità e dedizione nell'aiutarmi a sviluppare e perfezionare il mio lavoro.

Un ringraziamento speciale va alla mia famiglia che ha sempre sostenuto e incoraggiato il mio percorso accademico. Il loro sostegno e supporto sono stati la spinta necessaria per superare le sfide e raggiungere questo traguardo.

Ringrazio i compagni dell'università: Vale, Ali, Morgan, Matte, Nicoli, Franco, Luca e Nicco, che hanno alleviato le mie giornate e contro cui ho perso innumerevoli partite a bodriga.

Un ringraziamento agli amici di FORO, mia casa nell'ultimo periodo, con cui ho condiviso molte pause caffè.

Agli amici di una vita e compagni di mille avventure: Gian, Pier, Ale, Stol, Lollo e Chiara che, nonostante le nostre strade stiano prendendo direzioni diverse, sono e saranno sempre presenti al mio fianco.

Infine, il ringraziamento più importante va a Nau, mia compagna di viaggio, che mi ha sostenuto in questo percorso, credendo in me anche quando io non l'ho fatto.

Grazie mille a tutti.

Fede