Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería Mecatrónica

Sistema de predicción del error de modelado para una tarea de manipulación de objetos de la vida diaria para un robot humanoide, utilizando técnicas de aprendizaje de máquina

Informe de Proyecto de Graduación para optar por el título de

Ingeniero en Mecatrónica con el grado académico de Licenciatura

Orlando Moisés Solís Villalta

Cartago, 27 de abril, 2018

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Orlando Moisés Solís Villalta

Cartago, 30 de abril de 2018

Céd: 2-0733-0508

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería Mecatrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

MSc. Marta Vílchez Monge
Profesora Lectora

Dra. Gabriela Ortiz León
Profesora Lectora

Dr. Juan Luis Crespo Mariño
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por el Área Académica de Ingeniería Mecatrónica.

Cartago, 27 de abril de 2018

Resumen

Los robots han ganado mucha popularidad como facilitadores de tareas a los seres humanos, no solo en la industria, sino que se incorporan a ambientes de la vida cotidiana (ambientes poco controlados). Los robots tipo humanoide prometen ser los siguientes asistentes en el hogar, en donde la interacción con todo tipo de objetos es indispensable. En diferentes laboratorios alrededor del mundo se plantean distintas técnicas que permiten a los robot humanoides interactuar y manipular objetos en diferentes escenarios. Una de estas técnicas se basa en que el robot se concentra en las características del objeto para poder manipularlo, por lo que se plantea un modelo analítico del objeto para así poder conocer (predecir) su comportamiento.

Estos modelos analíticos consisten en una representación matemática con ecuaciones dinámicas que describen el comportamiento del objeto ante diferentes efectos. Para obtener una descripción ideal (igual a la real) del comportamiento del objeto, se requiere un modelo completo (modelo real) que incluya todas las variables que afecten la dinámica del objeto. En la práctica, por diversas razones, estos modelos son difíciles de obtener completos de forma analítica. En aplicaciones reales, se acude a modelos incompletos que aproximen el comportamiento real del objeto.

En el Arcos-Lab se plantea un modelo de objeto (una caja) el cual es utilizado por un robot humanoide para manipularlo en varios escenarios, este modelo permite al robot conocer (predecir) la pose del objeto mientras lo utiliza. Este modelo de la caja, es un modelo incompleto, por lo que la predicción de la pose presenta un error. En esta tesis se plantea un sistema compensador de error para este sistema de predicción de pose de objeto con un modelo de objeto incompleto. En la solución propuesta se estudian las fuentes de error del sistema de predicción de pose y se propone un modelo con redes neuronales que permite calcular el error en la pose. Con este modelo de error se logra no solo calcular el valor de error, sino también utilizar este valor para compensar el error presente en el sistema de predicción de pose. Se evalúa el modelo propuesto mediante validación cruzada, utilizando datos experimentales obtenidos con un robot humanoide. El modelo propuesto muestra resultados satisfactorios al disminuyendo del error del sistema de predicción de pose.

Palabras clave: Robot humanoide, redes neuronales, compensador de error, modelo incompleto.

Abstract

Robots have gained popularity as facilitators of tasks to human beings, not only in the industry, but also they are incorporated into daily life environments (low-controlled environments). Humanoid robots promise to be the next assistants at home, where interaction with all kinds of objects is essential. In different laboratories around the world, different techniques are proposed to allow humanoid robots to interact and manipulate objects in different scenarios. One of these techniques is based on the fact that the robot focuses on the characteristics of the object in order to manipulate it, so an analytical model of the object is proposed in order to know (predict) its behavior. These analytical models consist of a mathematical representation with dynamic equations that describe the behavior of the object when different effects are presented. In the theory to obtain an ideal description of the behavior of the object, it requires a very complex model (real model) that includes all the variables that affect the dynamics of the object, in practice, for various reasons, these models are difficult to obtain analyticaly. In real world applications, incomplete models are used to approximate the real behavior of the object. At Arcos-Lab an object model is proposed (a box model) which is used by a humanoid robot to manipulate it in various scenarios, this model allows the robot to know (predict) the pose of the object while using it. This model of the box is an incomplete model, so the prediction of the pose presents an error. In this thesis, an error compensation system is proposed for this object pose prediction system with an incomplete object model. In the proposed solution, the sources of error of the pose prediction system are studied and a neural network model is proposed to calculate the error in the pose. With this error model it is intended not only to calculate the error value, but also to use this value to compensate the error present in the pose prediction system. The model is evaluated using cross validation with experimental data obtained with a real humanoid robot. The proposed model shows satisfactory results by decreasing the error of the pose prediction system.

Keywords: Humanoid robot, error compensator, neural network, incomplete model.

...a mi familia
que siempre me ha sido mi motivación, para obtener este logro.

...a mis padres, quienes con mucho esfuerzo, hicieron posible mi sueño de estudiar una carrera universitaria.

...a mis hermanos por el apoyo a lo largo de mis años de estudio.

...a mis amigos por hacer de mi experiencia en la universidad algo inolvidable.

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin la ayuda de mi familia y amigos, quienes fueron un apoyo indispensable para llegar hasta el final de mi carrera.

A los profesores Federico Ruiz y al Dr. René Mayorga por brindarme la oportunidad de trabajar en sus laboratorios, donde viví una experiencia de mucho crecimiento, y además fueron guías importantes en el cumplimiento de los objetivos del proyecto.

A los profesores quienes durante la carrera me ayudaron a formarme no solo académicamente sino como persona. Por todas las lecciones que aprendí dentro y fuera de clase.

A los diferentes funcionarios del Tecnológico de Costa Rica quienes me ayudaron en mi estadía en la institución. Un especial agradecimiento a los funcionarios del comedor institucional.

Orlando Moisés Solís Villalta

Cartago, 30 de abril de 2018

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

ANN	Red neuronal artificial
MO	Modelo de Objeto
SPP	Sistema de predicción de pose

Notación general

$\mathbf{A}$	Matriz. $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$
$\underline{\mathbf{x}}$	Vector. $\underline{\mathbf{x}} = [x_1 \; x_2 \; \ldots \; x_n]^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
y	Escalar.

Capítulo 1

Introducción

En robótica se desea desarrollar sistemas que interactúen con el ambiente de una manera cada vez más precisa, lo que requiere mejores modelados del entorno. Dadas diferentes condiciones del ambiente físico, en muchas ocasiones no es posible obtener un modelo analítico que cumpla con los requisitos de la tarea, por lo que es necesario recurrir a otra técnica para compensar las deficiencias de los modelos. En esta tesis se presenta el desarrollo de un sistema con aprendizaje automático utilizado para compensar las deficiencias de un modelo de objeto utilizado por un robot humanoide para manipularlo.

El proyecto se desarrolló en el Laboratorio de Robots Autónomos y Sistemas Cognitivos, Arcos Lab. Actualmente el proyecto principal del laboratorio, es desarrollar un robot humanoide, que pueda ser utilizado como asistente en diversas tareas. En la figura 1.1 se tiene un modelo en 3D del robot humanoide en el que se está trabajando. Esta tesis se enfoca en el problema que existe con un modelo de objeto utilizado por el robot, para manipular objetos con una de las extremidades. Además, como parte del proyecto se realizó una pasantía en el Wise Lab, de la Universidad de Regina en Canadá.

El proyecto se planteó para realizarse durante una pasantía en el Wise Lab, en donde se propuso solucionar un problema de elección libre en el área de ingeniería mecatrónica, para resolverse utilizando aprendizaje automático. Se propuso la idea de Arcos Lab en donde se escogió el problema relacionado al robot humanoide.

Es importante mencionar, que aunque el proyecto se desarrolla en dos laboratorios diferentes, estos laboratorios no tienen ninguna relación, ni ha existido ninguna cooperación anterior a este proyecto. El contacto con este laboratorio en la Universidad de Regina lo realiza mi persona, en donde el Dr. René Mayorga me permite realizar una pasantía en su laboratorio.

Arcos-Lab

El laboratorio de Robots Autónomos y Sistemas Cognitivos [7] se encuentra en la Escuela de Ingeniería Eléctrica de la Universidad de Costa Rica, y es dirigido por el Dr. Federico

Ruiz Ugalde. El laboratorio trabaja en la creación y mejoramiento de sistemas cognitivos y robots autónomos que permitan ayudar a los humanos en diversas tareas. Principalmente aquellas tareas que sean tediosas, peligrosas o sumamente complicadas, que a su vez requieren de capacidades avanzadas de inteligencia cognitiva.

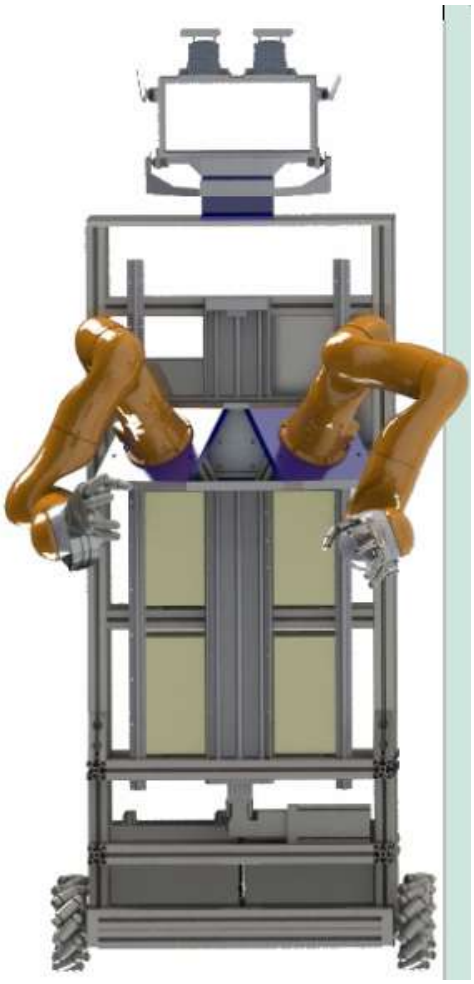


Figura 1.1: Robot Humanoide del Arcos-Lab.

En este laboratorio se trabajan varias áreas con diferentes proyectos, donde el proyecto principal del laboratorio es el robot humanoide que se muestra en la figura 1.1. Este robot se está diseñando bajo el concepto de robot asistente, donde se plantean diferentes escenarios de trabajo. Todos estos escenarios requieren altos niveles de cooperación entre el robot y los seres humanos, así como destreza a la hora de la manipulación de objetos.

El proyecto se realizó específicamente en una de las extremidades superiores del robot humanoide, este sub-sistema está compuesto por el brazo KUKA LBR IV+, y una mano DLR Hit II [8] como se muestra en la figura 1.2. Esta extremidad del robot cuenta con un sistema de control diseñado por el Dr. Federico Ruiz [1], para manipular objetos. Este sistema de control se concentra en el comportamiento del objeto para controlar su trayectoria.

Figura 1.2: El brazo KUKA LBR IV+, y una mano DLR Hit II.

Wise-Lab

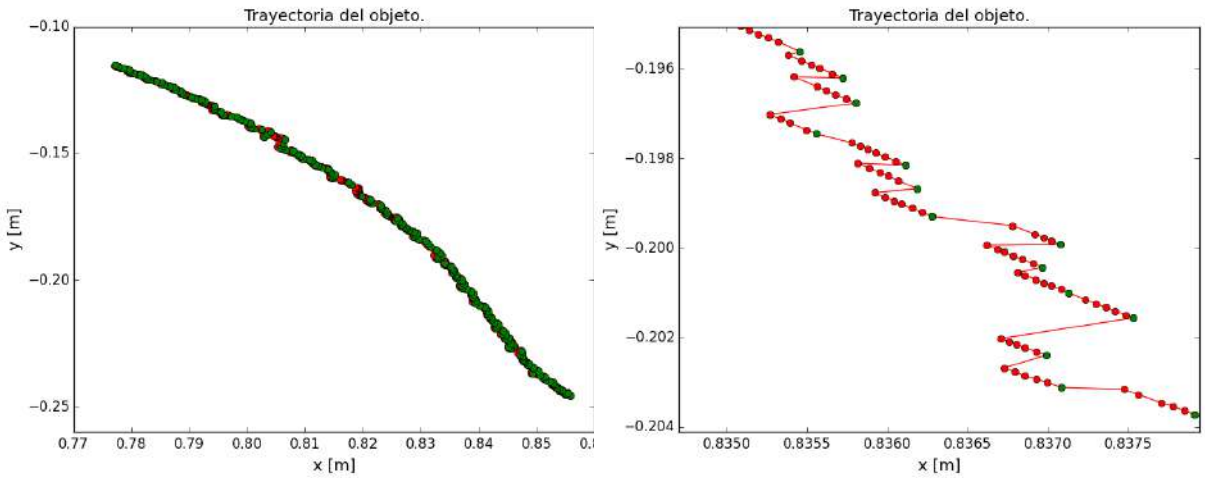
El Wise Lab (Wise, Intelligent, Systems and Entities, Lab)[9] trabaja bajo la dirección del Dr. René Mayorga. Este laboratorio se encuentra en la Universidad de Regina en Canadá y se enfoca en la investigación en temas relacionado con Sapiencia Artificial y Sistemas de Sapiencia. En el Wise Lab se realiza la etapa de diseño del sistema de aprendizaje automático bajo la asesoría del Dr. René Mayorga.

1.1 El proyecto

Como se menciona anteriormente el robot cuenta con dos extremidades las cuales le permiten realizar diferentes tareas con objetos con destreza. En [1] Federico Ruiz desarrolla una tarea de manipulación de objeto para el robot humanoide. Esta tarea consiste en desplazar una caja que se encuentra sobre una superficie horizontal desde una pose **A** de la caja, hasta lograr una pose **B** deseada, utilizando la mano del robot. Para lograr esto el robot desliza la caja sobre la superficie utilizando un único punto de aplicación de fuerza. La fuerza es aplicada con uno de los dedos de la mano DLR Hit II.

El sistema de control utilizado para realizar esta tarea se explica ampliamente en [1]. El control propuesto se concentra en el modelo de la caja para poder controlarla. Para ejecutar el sistema de control es necesario conocer la pose de la caja en cada iteración, por lo que parte del sistema propuesto por Federico Ruiz en [1] incluye un sistema de predicción de pose (SPP). Este sistema de predicción utiliza el modelo de la caja para predecir su comportamiento mientras esta se desliza sobre la superficie. El sistema de predicción utiliza los sensores del dedo de la mano para estimar la pose de la caja.

Este sistema de predicción presenta algunas deficiencias por lo que no logra estimar la pose



(a) Trayectoria. (b) Sección de la trayectoria.

Figura 1.3: Desplazamiento de la caja.

con la precisión deseada. Estas deficiencias introducen un error entre la pose estimada por el sistema de predicción y la pose real del objeto. El objetivo de este proyecto es proponer un módulo en el sistema de predicción, que permita disminuir el error de la pose estimada por el sistema de predicción.

En la figura 1.3 se ilustra el error que hay en la estimación de la posición del objeto utilizando el sistema de predicción descrito. En la imagen se muestra con puntos verdes la posición real del objeto y en rojo la estimación de posición que hace el sistema de predicción. Como se puede ver existe un error en el cálculo de la posición, en donde la predicción no coincide con los datos de la posición real.

En esta tesis se propone un módulo de software que permita disminuir el error de la estimación de la pose. Para atacar el problema se plantea que este módulo sea capaz de aproximar la señal de error en la pose estimada por el sistema de predicción. La solución propuesta se plantea de manera que al lograr una aproximación de la señal de error, y posteriormente utilizar esta aproximación para compensar la estimación realizada por el sistema de predicción.

El módulo que se plantea permite al sistema de predicción de pose, compensar algunas deficiencias del modelo de objeto utilizado. En este modelo del objeto no contemplan algunos fenómenos físicos presentes durante el deslizamiento del objeto sobre la superficie. Además el modelo hace algunas simplificaciones de algunos fenómenos físicos que también puedan influir en errores en los cálculos de pose.

En la tesis se exploran algunos de estos fenómenos de los cuales se sospechan como principales fuentes de error. Se analiza como pueden afectar los diferentes efectos a la dinámica del sistema y como pueden generar errores en el modelo planteado. Se explora la naturaleza del error entre la estimación del modelo de predicción y el comportamiento real del objeto.

Para el módulo de compensación propuesto se plantea utilizar aprendizaje automático para aproximar la señal de error. Específicamente se aborda el problema de aprendizaje con la técnica de redes neuronales. Esta red neuronal se entrena con datos experimentales para que este aprenda el comportamiento del error del sistema de predicción de pose.

Como parte adicional del proyecto fue necesario implementar un sistema para capturar datos de la pose real de la caja. Por lo que se incorpora una cámara para estimar los datos de la pose de la caja. Estos datos que se toman de la cámara se utilizan como los datos reales de pose de la caja. Esta información de la pose real es utilizada para obtener información de error al comparar la pose real de la caja con la pose estimada desde el sistema de predicción.

1.2 Hipótesis

"Es posible aprender el error de un modelo analítico predictivo, utilizado en la predicción de tareas de manipulación de una caja, utilizando aprendizaje automático, para mejorar la predicción del comportamiento de dicha caja."

1.3 Contribuciones

En la solución planteada se exploran las ventajas de obtener modelos de sistemas utilizando técnicas mixtas. La tesis parte de un modelo matemático que describe la dinámica de un objeto, y se propone un sistema de aprendizaje automático para "completar" las deficiencias de este modelo matemático. Esta combinación permite aprovechar las ventajas de cada técnica al aplicarlas en un mismo problema. El modelo analítico describe el comportamiento dinámico del objeto, utilizando conocimiento científico para obtener las ecuaciones físicas que gobiernan el movimiento del objeto. Se utiliza un modelo con red neuronal para mejorar el desempeño del modelo matemático del objeto.

En esta tesis se quiere demostrar que mediante el uso de redes neuronales, se puede generar un modelo que aprenda el error de cálculo de un sistema (sistema de predicción de pose) que se creó para estimar la pose de un objeto, el cual utiliza un modelo matemático de dicho objeto. Además, se comprueba que esta red neuronal una vez entrenada, puede utilizarse para disminuir el error en el cálculo del sistema que estima la pose del objeto.

Se brinda una validación cuantitativa de la disminución del error obtenida con este sistema propuesto. Se valida el modelo de error con diferentes datos y se mide su desempeño utilizando como métrica la disminución del error relativo respecto al modelo de predicción de pose original (modelo matemático).

1.4 Objetivos

En esta sección se describe el objetivo general y los objetivos específicos planteados para el desarrollo de esta tesis.

Objetivo General

Crear un sistema de aprendizaje de error para compensar las debilidades de un modelo de objeto incompleto utilizado para la predicción de pose del objeto, para una tarea de manipulación de cajas prismáticas mediante su deslizamiento sobre una superficie horizontal.

Objetivos Específicos

1. Proponer una lista de experimentos con el robot y un objeto, ante situaciones criticas, donde se pruebe la habilidad de un algoritmo de aprendizaje automático para disminuir el error del modelo de objeto, utilizado para predecir el comportamiento del movimiento de una caja sobre una mesa.
2. Implementar un sistema de visión artificial utilizando ROS, para capturar la pose de una caja.
3. Diseñar un algoritmo de aprendizaje automático capaz de compensar el error de cálculo asociado a un sistema de predicción de pose de objeto.
4. Validar la capacidad del algoritmo de aprendizaje automático para disminuir el error, utilizando datos experimentales obtenidos con un robot humanoide.

1.5 Estructura del documento

Este documento está organizado por capítulos donde se detallan los pasos para obtener la solución. En el capítulo 2 se detallan conceptos clave referentes a las distintas tecnologías y mecanismos de diseño y experimentación utilizados durante el desarrollo del proyecto. En el capítulo 3 se explica la infraestructura que se utilizó para realizar el proyecto, además se explica la configuración que se realizó con la mano del robot así como los pasos realizados para la implementación del sistema de visión. En el capítulo 4 se describen las etapas de diseño para realizar el módulo compensador de error de pose. En el capítulo 5 se exponen las distintas pruebas realizadas sobre el módulo de predicción de pose, así como los resultados de estas, acompañados de un análisis para determinar que estos resultados validan la obtención de los objetivos planeados. Finalmente en el capítulo 6 se exponen las principales lecciones aprendidas en el desarrollo del proyecto así como proyecciones de como podría el concepto crecer en el futuro.

Capítulo 2

Marco teórico

En este capítulo se presenta una serie de conceptos relevantes para el desarrollo del proyecto. Las ideas aquí expuestas se organizan en 3 secciones principales. En la primera parte se brinda una explicación sobre el sistema de predicción de pose, propuesto por Federico Ruiz en [1], aquí se mencionan sus principios de funcionamiento así como una explicación de las deficiencias que se detectaron en el modelo de objeto que se utiliza en el sistema de predicción. Luego se discuten diferentes trabajos donde utilizan aprendizaje automático para a la compensación y corrección de error de modelado en diferentes sistemas. Finalmente se muestra teoría relacionada al aprendizaje automático, así como algunas de las técnicas mas populares en la actualidad.

2.1 Sistema de Predicción

El sistema de control que propone Federico Ruiz en [1] para un robot humanoide, permite a un robot manipular un objeto deslizándolo sobre una superficie. El robot utiliza una de sus manos y aplicándole una fuerza con uno de sus dedos, es capaz de deslizar un objeto desde un punto inicial hasta un punto final deseado. El objetivo de esta tarea es mover un objeto de su pose **A** inicial, hasta una pose **B** deseada.

Durante el ciclo de control, el robot necesita conocer la pose del objeto para guiar la trayectoria de este. Como parte de este sistema de control también se propone un **Sistema de Predicción (SPP)**, el cual le permite al robot utilizar información sensorial de los dedos de la mano para conocer la pose del objeto durante la trayectoria.

El sistema de predicción propuesto por Federico Ruiz [1] es el encargado de estimar la pose del objeto que se está manipulando, y comunicar esta información al sistema de control. Este sistema de predicción utiliza un modelo del objeto para conocer su comportamiento mientras esta se desliza. El modelo matemático del objeto en el cual se concentra este trabajo es el modelo de una caja prismática, el cual se desarrolla ampliamente en [7].

Este modelo incorpora diferentes efectos físicos que intervienen mientras la caja es em-

pujada por el dedo. En él se utiliza información sobre los parámetros físicos de la caja, tales como las dimensiones y su masa. También incorpora coeficientes de fricción estática y dinámica existentes entre el dedo y la caja así como entre la caja y la superficie sobre la cual se desliza. Finalmente el modelo permite estimar la velocidad de la caja cuando se le aplique un fuerza puntual.

Utilizando este modelo se puede predecir cual es la velocidad de la caja al aplicarle una fuerza determinada, y a su vez integrando los valores de velocidad de la caja es posible calcular su pose. De esta manera se puede calcular la pose de la caja durante la trayectoria utilizando la magnitud de la fuerza aplicada en la caja. Para el caso del humanoide se aprovecha la información de los sensores de torque de los dedos para estimar la fuerza que aplica el dedo sobre la caja. Finalmente se tiene un modelo que relaciona la pose de la caja con la fuerza que aplican el dedo.

Una parte importante de este modelo, es la influencia de la fricción de la caja sobre la superficie. Esta fricción determina mayormente como rota y se desplaza la caja durante la trayectoria. En el trabajo de Federico Ruiz para modelar la fricción en [1] se utilizó una representación conocida como el Limit Surface, la cual se explica en la sección 2.1.1. Esta representación se genera con las fuerzas y torques presentes al haber dos superficies en contacto.

2.1.1 Limit Surface

Para modelar la interacción friccional entre dos superficies se utiliza el **Limit Surface** (LS) propuesta en [10][11][12]. El LS es una generalización de la ley de fricción de Coulomb para el contacto entre superficies. Esta representación se obtiene de la relación que existe entre los valores de fuerzas y el torque experimentado por un objeto causado por la fricción al ser deslizado sobre una superficie. Calculando los valores de fuerza y torque para todos los posibles centros de rotación entre las caras en contacto, se obtiene esta superficie conocida como **Limit Surface**.

El Limit Surface brinda información muy importante en los cálculos del modelo de predicción. Este provee las ecuaciones para calcular la velocidad de la caja a partir de las fuerzas medidas por los dedos de la mano DLR Hit II.

En la figura 2.1 se muestra la forma que tendría el LS de una caja con área en contacto cuadrada. Esta superficie muestra la relación que existe entre las magnitudes de las dos componentes de fuerza y el torque aplicados al centro de masa, presentes en un ejercicio de deslizamiento de la caja.

Teniendo el **Limit Surface** de la caja en estudio, se puede calcular la velocidad a partir de esta representación, tal como se explica en [1] basándose en los aportes de [10][11][12]. Para calcular la relación entre la velocidad y la fuerza aplicada, se debe calcular el gradiente del Limit Surface en ese punto. El gradiente representa la dirección de la velocidad resultante. De esta forma se logra obtener una ecuación que relaciona la velocidad del objeto con los

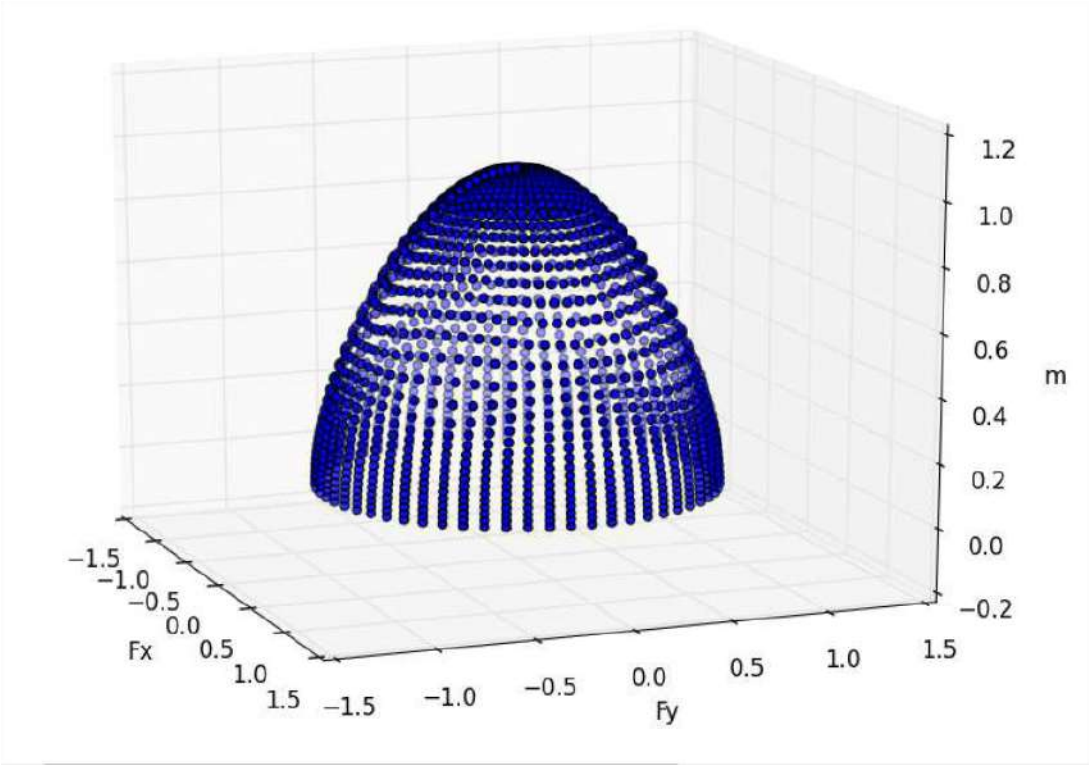


Figura 2.1: Representación gráfica de Limit Surface. Relación entre las fuerzas Fx, Fy y un momento m.

[1]

valores de fuerza y torque que están siendo aplicados. Una vez que se tenga un valor de velocidad es posible predecir las poses integrando desde una pose inicial conocida.

Tal como se explica en [1], para implementar el LS se utiliza una aproximación elíptica propuesta por [13] como se muestra en la ecuación 2.1.

$$\frac{F_{xo}^2}{f_{max}^2} + \frac{F_{yo}^2}{f_{max}^2} + \frac{M_o^2}{m_{max}^2} = 1 \tag{2.1}$$

Donde f_{max} y m_{max} es la máxima fuerza (como el resultado de la traslación pura) y torque (como resultado de la rotación pura), respectivamente y F_{xo} y M_{xo} son la fuerza y torque aplicado en el centro de masa del objeto.

Esta simplificación permite operar (matemáticamente) el LS más fácilmente, ya que proporciona la ecuación de la superficie que relaciona los valores de fuerza y torque aplicado. Pero cuenta con la desventaja que presentan un error asociado respecto al cálculo hecho con el LS verdadero, tal como se muestra en la figura 2.2.

En la figura 2.2 se muestran los ejemplos de LS de unas cajas al deslizarse sobre una superficie horizontal. Como se logra ver la superficie aproximada (en verde) con la ecuación 2.1 es diferente a la real (azul). A pesar de que tienen un comportamiento similar, existe un error asociado que se va a propagar en los cálculos posteriores.

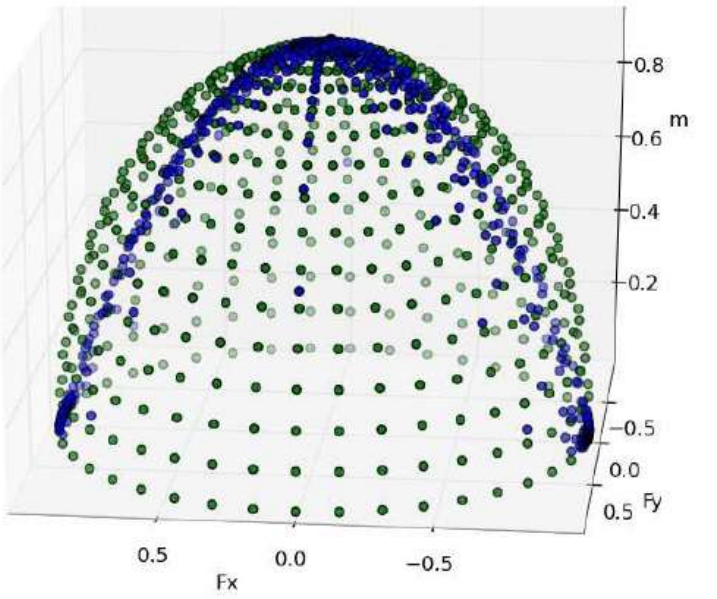


Figura 2.2: Ejemplo gráfico del error presente al utilizar la aproximación elíptica. En azul un LS verdadero y en verde su aproximación, para un área de contacto cuadrada [1].

Se utiliza la simplificación de elipsoide para derivar las ecuaciones de velocidad a partir del Limit Suface. A continuación se presenta un breve resumen de las ecuaciones necesarias para obtener las relaciones deseadas. Los procedimientos completos se pueden encontrar en la tesis de Federico Ruiz [1]. En la ecuación 2.2 se muestra como se calcula la velocidad v_o en el centro de la caja.

$$v_o = \mathbf{M} v_{cxy}, \mathbf{M} = m_{max}^2 \mathbf{ACB}^{-1} \tag{2.2}$$

La expresión que se presenta en la ecuación 2.2 muestra la relación que existe entre la velocidad del punto de aplicación de la fuerza v_{cxy} y la velocidad del centro de masa de la caja v_o. En este caso la velocidad v_{cxy} la aplica el robot con el dedo. La matriz (M) resume las propiedades de la caja, así como el $LimitSurface$, que transforma las fuerzas en velocidades. En donde **A**, **B** y **C** corresponden a la siguientes matrices,

$$\mathbf{A} = \begin{bmatrix} \frac{2}{f_{max}^2} & 0 & 0 \\ 0 & \frac{2}{f_{max}^2} & 0 \\ 0 & 0 & \frac{2}{m_{max}^2} \end{bmatrix} \tag{2.3}$$

$$\mathbf{B} = \begin{bmatrix} 2\left(\frac{m_{max}^2}{f_{max}^2} + r_y^2\right) & -2r_x r_y \\ -2r_x r_y & 2\left(\frac{m_{max}^2}{f_{max}^2} + r_x^2\right) \end{bmatrix} \tag{2.4}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -r_y & -r_x \end{bmatrix} \tag{2.5}$$

De estas matrices los valores de r_x y r_y, representan las coordenadas del punto de aplicación de la fuerza respecto al centro de masa de la caja. Los parámetros f_{max} y m_{max}, representan los valores de fuerza y torque máximo provenientes del elipsoide, utilizado como simplificación del LS presentes en la ecuación 2.1. La magnitud m_{max} se calcula como se muestra en la ecuación 2.6.

$$m_{max} = \frac{\mu\rho}{l_2}\left(l_2^3 arcsinh\left(\frac{l_1}{|l_2|}\right) + l_1^3 arcsinh\left(\frac{l_2}{l_1}\right) + 2l_1 l_2\sqrt{l_1^2 + l_2^2}\right) \tag{2.6}$$

$$\rho = \frac{f_n}{l_1 l_2} \tag{2.7}$$

Donde los valores l_1 y l_2 son las dimensiones de la base de la caja, el parámetro μ, es el valor del coeficiente de fricción en la caja y la mesa, y f_n es la fuerza normal que la caja ejerce sobre la superficie ($masa * gravedad$).

$$f_{max} = \mu_{ot} f_n \tag{2.8}$$

También se sabe que f_{max} se calcula con la ecuación 2.8. Donde μ_{ot} es el coeficiente de fricción entre la mesa y la caja.

2.1.2 Deficiencias del Sistema de Predicción

Como se a mencionado anteriormente el sistema de predicción permite estimar la pose de la caja mientras esta es deslizada por el robot. Pero como se explica en [1] existe un error presente en el cálculo de la pose. Este error se puede explicar por que existen efectos físicos omitidos en el modelo de objeto utilizado y además, las simplificaciones matemáticas en el Limit Suface puede afectar los cálculos. En esta sección se mencionan alguno de estos efectos omitidos y como estos pueden afectar la magnitud del error en la predicción.

Entre los efectos físicos no incluidos se puede realizar una gran lista de diferentes situaciones que afectan el comportamiento de la caja mientras esta es manipulada. A continuación se presenta una lista de aquellos efectos que se consideraron importantes fuentes de error del modelo.

1. Diferencias entre Elipsoide y Limit Surface: Esta es una de las principales fuentes de error del modelo, ya que está presente en todas las cajas que se trabajan. Esta simplificación del Limit Surface genera errores en los valores de predicción. El elipsoide es una aproximación al Limit Suface por lo que presenta diferencias respecto a la representación real. Estas diferencias incluyen errores en los cálculos.

   Pero además, es importante considerar que el elipsoide es una aproximación estática del Limit Suface real, mientras el Limit Suface es una representación dinámica

que puede cambiar bajo ciertas condiciones. Cualquier alteración que exista en la condiciones de fricción entre la caja y la mesa, hacen que el Limit Surface cambie, y estos cambios no son contemplados por el Elipsoide. Al utilizar el elipsoide se asume que las condiciones de fricción entre la caja y la superficie sobre la cual se desliza no cambian, y esto no siempre es así.

Las condiciones de fricción puede cambiar por diversas razones. Como se menciona en los puntos siguientes, durante la trayectoria de la caja existen diferentes condiciones que cambian la relación de fricción. Pequeños cambios en los coeficientes de fricción de la base de la caja o de la superficie, cambios en la masa de la caja o movimientos indeseados de la caja, pueden provocar cambios en la geometría del Limit Surface.

2. Distribución de peso irregular: El modelo esta diseñado considerando que la caja a manipular tiene una distribución de masa homogénea y por lo tanto el centro de masa coincide con el centro geométrico de la caja. El modelo asume que el centro geométrico de la caja y el centro de masa coinciden, por lo que esta variante puede provocar un comportamiento no predecible de la caja. La distribución de peso irregular afecta la fuerza de fricción presente en la base, por lo que implica un cambio en la geometría del Limit Surface.

3. Cambio de fricción de las superficies: este es otro efecto que no está considerado en el modelo, el cual también va a afectar el comportamiento de la caja mientras esta se desliza. El modelo asume los coeficientes de fricción como valores constantes. Si la caja no tiene un coeficiente de fricción constante sobre toda la superficie de la base, o si la superficie sobre la cual se desliza tampoco tiene un coeficiente de fricción constante en todas sus dimensiones, se presentarían cambios en la relación de fricción entre las superficies. Esto provoca que el Limit Suface cambie constantemente mientras la caja es desplazada.

4. Inclinación del objeto: como parte del modelo se asume que cuando la caja es desplazada, esta se mantiene siempre en contacto con la superficie sobre la cual se desliza, y se sabe que esto no siempre ocurre de esta manera, ya que el dedo aplica una fuerza paralela a la superficie, provocando que la caja tienda a inclinarse. Y esto junto con que la base de la caja no es totalmente plana afectan los valores de fricción esperados entre la base y la superficie sobre la cual se desliza el objeto.

5. Grandes perturbaciones: El sistema de control es capaz de reaccionar ante perturbaciones, como fuerzas externas que se apliquen a la caja. Pero el sistema de predicción es sensible si estas perturbaciones son muy grandes.

6. Deformaciones en las paredes de la caja: Esta característica afecta cuando el dedo aplica una fuerza sobre alguna de las caras de la caja y esta se deforma. Esta deformación genera cambios en los valores de fuerzas de fricción entre el dedo y la pared de la caja, ya que el dedo no esta tocando una superficie completamente

plana. Estas deformaciones de la caja, también pueden darse en la base de la caja, afectando la interacción de esta base con la superficie sobre la cual se desliza.

También se tiene una lista de efectos que tampoco fueron tomados en cuenta en el desarrollo del modelo de la caja, pero se consideró que no son relevantes en la magnitud del error.

1. Efectos inerciales: los efectos inerciales se omiten porque durante la tarea de manipulación no se alcanzan altas velocidades ni grandes aceleraciones.

2. Efectos de la temperatura: durante las tareas típicas de manipulación no se trabajan con cambios considerables de temperatura, que puedan incluir variaciones en el comportamiento del modelo.

3. Cambios de fricción con la velocidad: Se trabaja con el modelo de fricción de Coulomb, que para los rangos típicos en los que se desplaza la caja, no se esperan cambios en el comportamiento de la fricción.

Este ejercicio de identificar las fuentes de error permite plantear una serie de experimentos para medir el desempeño del modelo de predicción de pose ante estas circunstancias. Además, estos experimentos generan datos que nos permiten utilizar aprendizaje automático para plantear el módulo de composición de error. A modo de ejemplo se tiene que si se sospecha que una caja con una distribución de masa no uniforme representa una fuente de error, se busca hacer un experimento en el cual se utilice una caja con una masa no uniforme y se mide el error del modelo de predicción al intentar manipular esta caja. El siguiente paso es lograr que el sistema de compensación logre disminuir este error.

Como parte de los resultados expuestos en la tesis de Federico Ruiz en [1], se ejemplifica el error de predicción del modelo con un gráfico de la posición del sistema de predicción contra la posición real. Dadas estas imperfecciones del modelo, la implementación actual del sistema de control requiere de un sistema de respaldo para conocer la pose real de la caja. En el caso de TUM Rosie [14], este contaba con una cámara de vídeo que fue utilizada para conocer la pose real de los objetos. Por la velocidad de procesamiento del sistema de visión, la frecuencia del sistema de predicción de pose es mayor que la frecuencia de respuesta del dato que otorga la cámara.

En la figura 2.3 se muestran los datos de posición de la caja mientras esta es deslizada por el robot TUM Rosie. En este experimento la posición inicial es x=0.9655 y=-0.4015 y la posición final es x=0.7 y=-0.1. En este ejercicio la pose es estimada por el sistema de predicción, pero además, se utiliza la cámara para conocer la pose real del objeto. Así que se utiliza el sistema de predicción para estimar la pose, pero cada vez que el sistema de visión tenga un nuevo dato de pose, se utiliza este como real. Además, el sistema de predicción lo utiliza para predecir el siguiente dato hasta que el sistema de visión logre medir un nuevo dato de pose. En este caso la cámara tiene una velocidad de captura

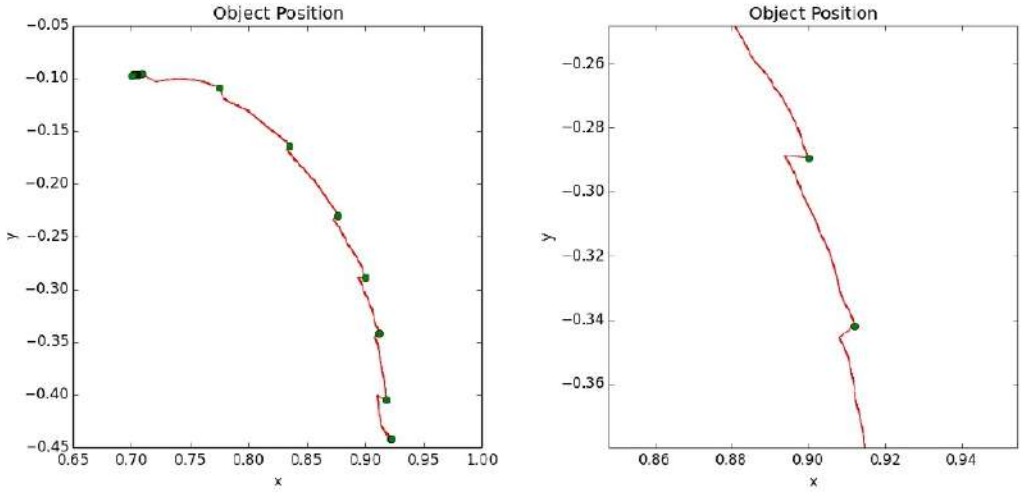


Figura 2.3: Error presente en el sistema de predicción. Los puntos rojos son los puntos calculados por el sistema de predicción, los verdes son la actualización de pose realizada por la cámara. A la Izquierda se muestra toda la trayectoria y a la derecha una vista en detalle de un tramo de la misma [1].

de imagen de 15 fps contra una velocidad de cálculo del sistema de predicción que varía entre 60 a 120 muestras por segundo.

En la figura 2.3 se ejemplifica el error de cálculo del sistema de predicción, y se ve como este aumenta conforme pasa el tiempo y el valor difiere aún más del valor real. La magnitud de este error va a estar afectado principalmente por los efectos descritos anteriormente en esta sección. Con el desarrollo del presente proyecto se pretende disminuir el error entre los valores del sistema de predicción y los valores de pose real.

2.2 Compensación y reducción del error de modelado

En esta sección se mencionan algunos trabajos que se desarrollaron para reducir el error de modelado de diferentes sistemas. Cuando se quiere controlar un sistema, una de las etapas más importantes es encontrar un modelo matemático del sistema en estudio. En esta etapa se busca traducir a ecuaciones matemáticas los efectos físicos que intervienen en el proceso a controlar. Pese al esfuerzo de quienes diseñan el modelo, muchas veces el modelo matemático planteado del sistema no se comporta como el sistema real, por lo que se acude a diferentes técnicas para compensar estas diferencias.

El uso del aprendizaje automático en el proceso de modelado de sistema se ha convertido en una técnica muy popular por los resultados satisfactorios que ha mostrado para solucionar problemas complejos. El aprendizaje automático se utiliza en diferentes etapas del modelado y brinda al diseñador una solución alternativa en el modelado de sistemas. El aprendizaje automático presenta algunas ventajas ante las técnicas tradicionales, ya que en muchos casos permite simplificar el modelado de fenómenos complejos.

En el área de robótica los sistemas interactúan con una gran cantidad de fenómenos físicos que muchas ocasiones son difíciles de modelar. Se encuentran numerosas aplicaciones en donde se aprovechan las ventajas del aprendizaje automático para solucionar problemas complejos. En [15] se presenta el uso de redes neuronales para modelar el comportamiento de un robot de dos vínculos con articulaciones flexibles. En casos como estos no se utiliza una representación matemática del robot, sino que se utiliza la red neuronal para modelar completamente el comportamiento del robot.

Un caso similar donde se modela completamente un sistema se presenta en [16]. Aquí se trabaja una extremidad de un robot humanoide donde se utilizan actuadores neumáticos. El arreglo de actuadores neumáticos utilizados requiere un modelado complejo. En [16] proponen una red neuronal para extraer el modelo que relaciona los voltajes aplicados a los actuadores y los ángulos medidos en la extremidad del robot.

La cinemática inversa es uno de los modelos más utilizados en robótica que permite conocer la pose en el efector final dada las posición en las diferentes articulaciones. Existen algunos tipos de robots que dada su configuración hacen difícil la obtención de este modelo. En [17] se utiliza aprendizaje automático para obtener el modelo se cinemática inversa en un robot continuum, el cual posee vínculos flexibles.

Las aplicaciones de robótica como en operaciones quirúrgicas se quiere una gran precisión. En robots tele-operados como en [18], [19] se utiliza cámaras de visión artificial para estimar la fuerza que aplica el robot sobre le tejido. El modelo planteado pretende medir con las cámaras la deformación del tejido y estimar la fuerza aplicada, lo cual presenta dificultades para implementar un modelo analítico que funcione correctamente en la realidad. En [18], [19] se propone un sistema de aprendizaje automático para obtener el modelo que mapeé los desplazamientos en el tejido a datos de fuerza aplicada.

La soluciones propuestas en los ejemplos anteriores, plantean utilizar aprendizaje automático para obtener un modelo completo. Esta solución aprovecha las ventajas del aprendizaje automático, pero no se cuenta con un modelo explícito del sistema. El modelo explícito tiene la gran ventaja de que permite escalar con facilidad la solución a otro sistema similar, mientras que el aprendizaje automático requiere un re-entrenamiento para un nuevo sistema.

Otros autores proponen utilizar el aprendizaje automático junto con un modelo analítico, para aprovechar las ventajas de ambos métodos. En estos trabajos que se presentan a continuación se desarrolla un modelo explícito del sistema y además se utiliza aprendizaje automático para mejorar el desempeño de este modelo analítico. Con esta solución se obtienen las ventajas de ambas soluciones.

Muchas aplicaciones de manufactura requieren un control que trabaje a altas velocidades y con gran precisión. En [20] y [21] se utiliza aprendizaje automático para compensar los efectos de la fricción e inercia en el modelo de un robot de un grado de libertad, mejorando así el desempeño del modelo. En [22] se utiliza aprendizaje automático para compensar la incertidumbre de no-linealidades y parámetros desconocidos del modelo, para un robot

móvil.

Para robótica cooperativa, en [23] se presenta el modelo de la interacción entre el efector final y la viga a manipular. El modelo de esta interacción es crítica para operar, por lo que se incorpora aprendizaje automático para compensar las incertidumbres del modelo al operar en el mundo real. En [24] se presenta otro ejemplo similar donde se utiliza aprendizaje automático para compensar las no-linealidades del control de un actuador un robot.

En [25] se propone un método de solución similar al que se propone en esta tesis. En [25] el problema es la disminución del error en la reconstrucción de imágenes de profundidad inversa (2D) a partir de modelos 3D generados desde una cámara estéreo.

Utilizan una red neuronal para aprender el error en el cálculo de profundidad a partir del modelo generado con una cámara estéreo. Para lograr esto se comparan los valores de profundidad calculados desde una cámara estéreo con los valores de profundidad calculados con un sensor más avanzado, como lo es el LIDAR. El LIDAR es un sensor que permite obtener los valores de profundidad con mayor confiabilidad.

El error aprendido con redes neuronales se utiliza para mejorar las reconstrucciones de imágenes de profundidad inversa a partir del modelo obtenido de la cámara estéreo. Esta técnica permite mejorar la calidad de estas imágenes de profundidad inversa sin la necesidad de invertir en hardware más costoso como lo son los LIDAR.

Esta forma de resolver los problemas de modelado, en donde se utiliza un modelo analítico en combinación con aprendizaje automático, presenta muchas ventajas. Ambas técnicas tienen sus ventajas y desventajas, una frente a la otra, pero en estos trabajos se propone utilizar ambas para compensar las debilidades de cada una.

Utilizar un modelo analítico para describir un fenómeno con ecuaciones matemáticas, permite comprender a profundidad el problema en estudio. La principal ventaja de estos modelos es que se pueden generalizar a muchos casos con facilidad. Los modelos matemáticos al ser paramétricos, se les puede cambiar estos parámetros (constantes físicas del problema) para cada caso específico. Otra ventaja es que al utilizar una descripción matemática, se tiene una comprensión clara entre la relación de las variables de salida y las variables de entrada del modelo.

Como desventaja se puede resaltar que para que el modelo analítico se comporte igual al fenómeno real, se requiere de un modelo muy complejo que tome en cuenta todas las variables físicas que intervienen. Estos modelos de alta complejidad son difíciles de obtener y en muchos casos no son deseables, por su dificultad de trabajarlos.

Por otra parte, los modelos con aprendizaje automático no se basan en ecuaciones matemáticas (ecuaciones físicas) del problema, sino, están basados en datos experimentales. Esta característica presenta una gran ventaja frente a modelos que sean extremadamente complejos de modelar matemáticamente. Los modelos con aprendizaje automático mapean la relación entre los datos de entrada y los datos de salida sin necesidad de profundizar en las ecuaciones matemáticas que relacionan ambas variables.

La calidad de un modelo obtenido con aprendizaje automático, depende principalmente de la calidad de los datos. El modelo obtenido es válido siempre y cuando se utilicen datos de prueba que se comporten de la misma manera que los datos de entrenamiento. Es por esta razón que estos modelos no suelen ser generalizables a varios casos. Para generalizar este modelo a un nuevo caso, se debe volver a entrenar el modelo incluyendo datos de este nuevo caso.

Como se ha mencionado, cada método presenta ciertas ventajas y la selección de cada uno depende de cada aplicación. En esta tesis se propone aprovechar un modelo analítico ya existente, y se plantea mejorar su desempeño incorporando un modelo con aprendizaje automático. Aprovechando las ventajas de ambos métodos.

Estas ventajas se pueden apreciar en un caso específico. Con el modelo matemático de la caja que se tiene, se puede conocer la dinámica de esta, para predecir su comportamiento, y estimar su pose. Este modelo puede ser utilizado con diferentes cajas, simplemente ajustando los parámetros (dimensiones, peso y coeficientes de fricción) en el modelo para la nueva caja.

Como ya se ha expuesto ampliamente en la sección este modelo matemático de la caja tiene algunas debilidades al estar incompleto, pero es lo suficientemente robusto para poder controlar esta caja. La incorporación del modelo con aprendizaje automático en muy útil, en el caso que se tenga algún objeto que el robot utilice con frecuencia y tenga que manipular este con mucha destreza. Se puede entrenar al modelo de aprendizaje automático para que puede compensar el error del modelo analítico con esta caja especifica.

Otro ejemplo en donde es importante la implementación de ambos modelos es cuando se conoce de antemano que el robot tiene problemas (errores de predicción grandes) para predecir el comportamiento con algún objeto especifico, utilizando únicamente el modelo matemático. De la misma manera se puede entrenar al modelo de aprendizaje automático para que compense el error en este tipo de casos.

2.3 Técnicas de aprendizaje automático

En esta sección se exponen algunos conceptos importantes relacionados a las técnicas de aprendizaje automático utilizadas en la solución propuesta. Se mencionan algunos conceptos que expone Tom Mitchel en su libro [6], sobre cómo plantear un problema a tratar con aprendizaje automático. Además, se explicarán algunos conceptos de la técnicas de Redes Neuronales Artificiales.

2.3.1 Resolución de problemas en términos de aprendizaje automático.

Para poder aplicar las técnicas de aprendizaje automático a un problema, es necesario comprender la naturaleza del mismo. Tom Mitchel, propone en su libro [6] dividir el problema en tres partes fundamentales para plantear una solución. Descomponiendo el problema en estos tres componentes se puede aplicar las técnicas de aprendizaje automático con facilidad. En este libro Mitchell propone definir una tarea T, una medida de desempeño P y la experiencia E.

La variable T es la parte principal del problema a solucionar. Se refiere a la tarea que se quiere lograr con el sistema. Se plantea como una actividad o un objetivo que se quiere alcanzar.

La variable P representa la medida del desempeño del sistema en la tarea T. Se tiene que elegir un parámetro medible con el cual se pueda evaluar si se están obteniendo los resultados deseados en la ejecución de la tarea T.

E es la experiencia que se utiliza para mejorar el desempeño P en la tarea T. Se busca definir una serie de experimentos que generen experiencia, de la cual el sistema pueda aprender.

Luego de definir los parámetros T, E y P, Mitchel [6] propone los tres pasos para definir la técnica de aprendizaje automático a utilizar.

1. **Elegir la función objetivo:** Este paso se refiere a reducir la tarea que se quiere lograr a una función F que sea computable. Es decir, se quiere definir una función más sencilla en donde se definan los parámetros de los cuales se va a aprender. En este punto se debe definir cuales serán el o los datos de entrada de la función, así como cual será la salida o las salidas de esta función.

2. **Elegir la representación de la función objetivo:** La función F, es totalmente desconocida. Lo que se quiere es encontrar una aproximación a esta utilizando aprendizaje automático. En este paso se debe definir una representación $\hat{F}$ de la función F definida en el paso anterior. Esta representación $\hat{F}$ podría ser una función de cualquier tipo, ya sea una función polinomial, una red neuronal artificial o una máquina de soporte vectorial, el modelo de k-vecinos más cercanos, entre muchas otras.

3. **Elegir un algoritmo de aproximación de la función objetivo:** Este paso consiste en escoger un algoritmo que permita ajustar los parámetros de función $\hat{F}$ para aproximar la función objetivo F. Un algoritmo se encarga de minimizar el error de la función $\hat{F}$ respecto a la respuesta deseada. Esta respuesta deseada se toma de los datos de entrenamiento.

2.3.2 Técnicas de aprendizaje Automático

En la actualidad las aplicaciones del aprendizaje automático se han expandido a diversas áreas del conocimiento. Lo que ha hecho que se amplíen las herramientas disponibles para utilizar estas técnicas. Actualmente se cuentan con diversas técnicas existentes, las cuales permiten solucionar problemas diversos. En esta sección se mencionan algunos de los algoritmos mas utilizados de aprendizaje automático.

En figura 2.4 se muestra un diagrama de algunas de las principales técnicas de aprendizaje automático y como esta se clasifican. El área de aprendizaje automático se puede dividir en dos grandes grupos. Estos son aprendizaje supervisado y aprendizaje no supervisado. En el caso de la técnica de aprendizaje supervisado, es aquella donde se tiene información de cómo se desea que se comporte el modelo a entrenar. Por otra parte en el aprendizaje no supervisado, el sistema determina la forma en que el modelo debe comportarse basado en la información con la que se entrena.

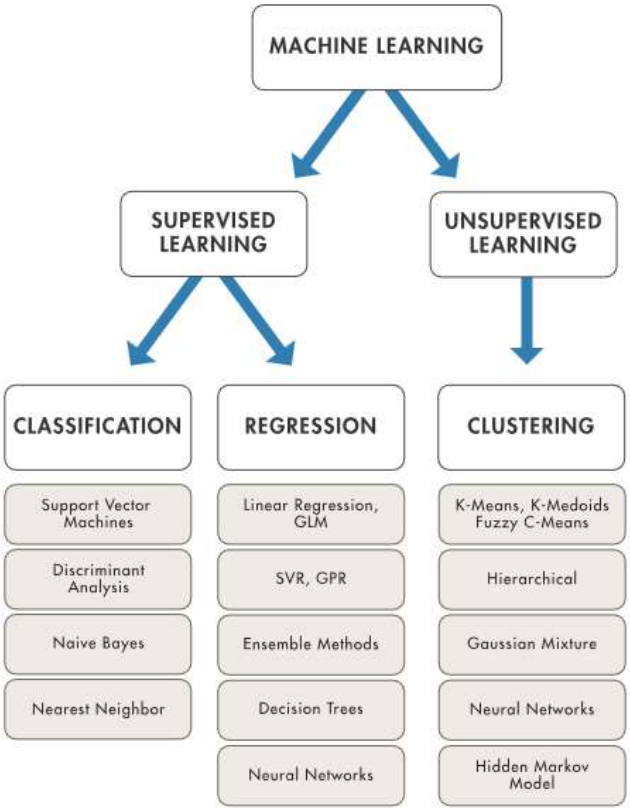


Figura 2.4: Diagrama de técnicas de aprendizaje automático [2].

Además, las técnicas de aprendizaje supervisado se dividen en dos subgrupos, estos son técnicas de regresión o de clasificación. Los problemas de clasificación como técnica de aprendizaje supervisado, consisten en agrupar las entradas del sistema dadas sus características. Es decir, el sistema aprende a discernir a qué grupo pertenece una entrada. De antemano se conocen la cantidad de grupos, el sistema aprende a reconocer las características de los elementos en cada grupo. El otro grupo de técnicas corresponde a

regresión. Estas técnicas funcionan para aproximar una función continua, por lo que se busca aprender la relación que existe entre las entradas al sistema y sus salidas.

El problema que se trabaja en esta tesis se clasifica como un problema de regresión. Esto porque se sabe de antemano, cuál debe ser la salida para cada entrada al sistema. El aprendizaje se utiliza para descubrir cuál es la función que relaciona la entrada con cada salida. Específicamente se tiene como entrada una estimación de pose calculada con el sistema de predicción (SPP)y la salida es el error de esta respecto al dato de pose estimado por la cámara. A continuación se exponen algunas de estas técnicas de regresión.

Modelos de regresión lineal.

Los modelos de regresión lineal son muy populares y se han utilizado en muchas aplicaciones. Su concepto es fácil de comprender y ha sido muy popular en diferentes campos. Esta técnica consiste en utilizar una función lineal sobre los valores de entrada tal como se muestra en la ecuación 2.9.

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b \tag{2.9}$$

Como se explica en [3], en esta ecuación $x[0] y x[p]$ representan variables de entrada del sistema de aprendizaje, w y b son parámetros que el modelo aprende, y $\hat{y}$ es la predicción hecha por el modelo. En el caso mas simple donde se utilice una sola variable de entrada $x[0]$ la ecuación tiene la forma $\hat{y} = w[0] * x[0] + b$ donde w seria la pendiente y b el corte con el eje y.

En este ejemplo mostrado en la 2.5 se utiliza un modelo de regresión lineal con una sola variable. Los puntos azules representan los valores de entrenamiento y la línea azul representa los valores del modelo.

k-vecinos más cercanos.

Este es uno de los algoritmos utilizados en aplicaciones de regresión. Este algoritmo consiste en analizar los vecinos cercanos para estimar el valor de un nuevo punto. Lo que hace es considerar los valores de los puntos más cercanos, y el valor del punto corresponde al promedio de estos valores. Para mayor claridad ver figura 2.6.

En la figura 2.6 se muestra un ejemplo de regresión utilizando el algoritmo k-vecinos más cercanos. En este caso se utiliza para calcular el valor de 3 puntos, estos puntos son $x_0 = -1.5$, $x_1 = 0.9$ y $x_2 = 1.5$. La estrella verde representa el valor buscado y la estrella azul representa la predicción. Los puntos azules representan los datos de entrenamiento. En este caso se utilizan los 3 vecinos más cercanos, es decir $k = 3$, por lo que el valor buscado se calcula como el promedio de los valores de los 3 vecinos más cercanos.

Esta técnica requiere de dos parámetros principales para configurar el algoritmo. Estos parámetros son la cantidad de vecinos k que se tomarán en cuenta y el otro parámetro

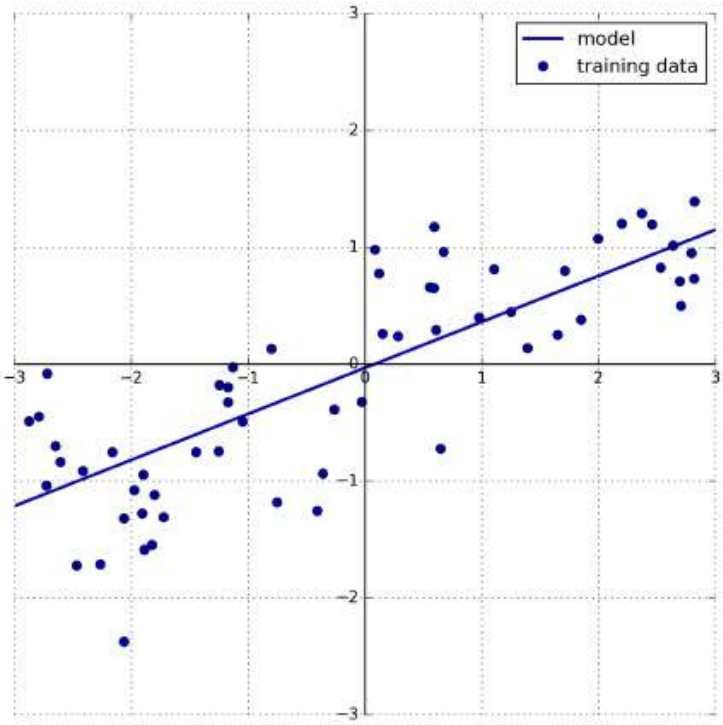


Figura 2.5: Predicciones realizadas con un modelo lineal [3].

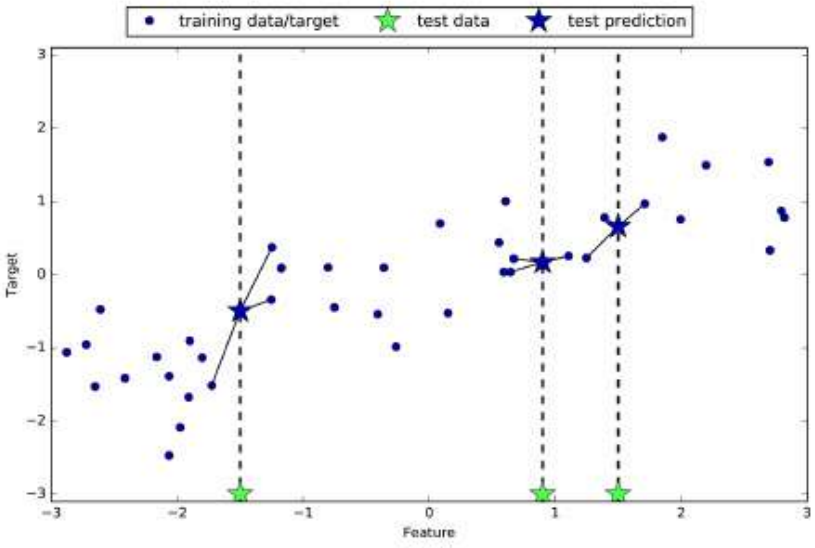


Figura 2.6: Vecinos más cercanos, con $k = 3$ [3].

es la forma en que se decida medir la distancia entre los puntos. Como desventaja de este algoritmo se puede mencionar que es lento para realizar predicciones y además no es eficiente cuando se utilizan muchas entradas.

Máquinas de Soporte Vectorial-Regresión

La técnica de aprendizaje automático conocida como Máquinas de Soporte vectorial fue propuesta por Vapnik [26] y [27]. Esta técnica se propone inicialmente para resolver problemas de clasificación y luego se extiende a problemas de regresión. En [28] se presenta el desarrollo matemático del método y de algunas de sus características. Se explica la técnica partiendo desde un modelo sencillo linear, en donde se busca encontrar el modelo que relacione un grupo finito de datos (x_i, y_i), $x_i \in R^n$,$y_i \in R$. En donde se quiere encontrar una función $f(x)$ tal que esté desviada máximo una distancia ε del objetivo y_i y que sea lo más "plana" posible (valores pequeños de w).

$$f(x) = \langle w, x \rangle + b \tag{2.10}$$

En caso de una función lineal, se tiene el modelo que se muestra en la ecuación 2.10. Donde $\langle , \rangle$ representa el producto interior. Como se buscan valores pequeños de w el problema se reduce a minimizar $\|w\|^2$ sujeto a las condiciones de las ecuaciones 2.11a y 2.11b.

$$y_i - \langle w, x_i \rangle - b \leq \varepsilon \tag{2.11a}$$
$$\langle w, x_i \rangle + b - y_i \leq \varepsilon \tag{2.11b}$$

Generalizando para cuando el problema de regresión es no factible, se reformulan las ecuaciones 2.11a y 2.11b. Y se agregan variables de holgura ξ_i, ξ_i^* como se muestra en las ecuaciones 2.12b y 2.12b. Y el problema a minimizar es $\|w\|^2 + C \sum_i (\xi_i, \xi_i^*)$, donde $\xi_i, \xi_i^* \geq 0$.

$$y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \tag{2.12a}$$
$$\langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \tag{2.12b}$$

La constante C determina el compromiso entre el plano f y la cantidad de desviaciones mayores a ε permitidos. Luego de resolver el problema de optimización sujeto a las condiciones 2.12a y 2.12b se obtiene el modelo de la ecuación 2.13. En [28] se muestra el desarrollo matemático completo.

$$f(x) = \sum_i (\alpha_i - \alpha_i^*) \langle w, x \rangle + b \tag{2.13}$$

Donde α_i, α_i^* se conocen como vectores de soporte. Como se explica en [28], los vectores de soporte son aquellos puntos x_i en los cuales el error de interpolación es mayor o igual a ε. Los puntos en los que el error de interpolación es menor que ε nunca son vectores

de soporte, y no se toman en cuenta en la solución. Cuando se encuentran los valores que no clasifican para ser vectores soporte, estos pueden ser eliminados del conjunto de datos, y si se resuelve el problema de regresión nuevamente sobre el conjunto reducido se encuentra la misma solución.

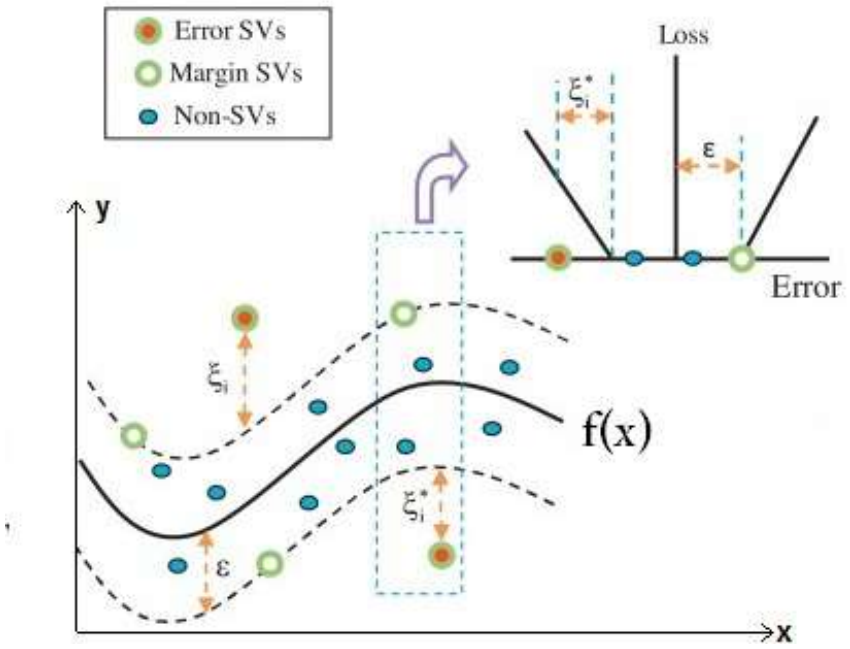


Figura 2.7: Diagrama del funcionamiento de la técnica de Máquinas de Soporte Vectorial para Regresión. Adaptada de [4].

En la figura 2.7, se explica el principio de funcionamiento del algoritmo de máquinas de soporte vectorial para regresión. En línea discontinua se representan las funciones del borde del tubo que se construyen a partir de los vectores soporte. En el centro de este tubo se tiene en línea continua la función $f(x)$ que representa al modelo buscado. Los puntos azules representan aquellos puntos del conjunto total de datos que no se consideran como vectores soporte, son todos aquellos que se encuentran dentro del tubo. En verde con el centro blando se representan los puntos que están justamente en el borde del tubo, con aquellos puntos que están a una distancia ε_i de la función f. Finalmente aquellos puntos fuera del tubo se representan en verde con el centro rojo, los cuales están a una distancia ξ_i del borde del tubo. Todos los puntos verdes se consideran vectores soporte.

En la ecuación se tiene el modelo para problemas lineales. Cuando se tratan con problemas no lineales, se suele cambiar el operador del producto interior (x,y) por una función núcleo k(x,y). A continuación se muestran algunos ejemplos de núcleos utilizados que se mencionan en [28].

a) Polinomio: $k(x, y) = (\langle x, y\rangle + 1)^d, d = 1, 2, 3...$

b) Funciones de base radial: $k(x, y) = e^{-|x-y|^2/\sigma^2}$.

c) Redes neuronales de dos capas: $k(x, y) = tanh(b\langle x, y\rangle - c)$, (para ciertos valores de b y c).

Redes Neuronales Artificiales

Esta técnica está basada en la estructura de las neuronas que se encuentra en el cerebro. En la figura 2.8 se tiene la estructura de una neurona tal como se encuentran el cerebro humano. Cada una de estas partes tiene una función, las dendritas es la entrada de información de la neurona, el cuerpo de la célula es el encargado de procesar estos datos de entrada y finalmente generar una salida que se envía por el axón. Las neuronas en el cerebro comunican la información por medio de señales electroquímicas.

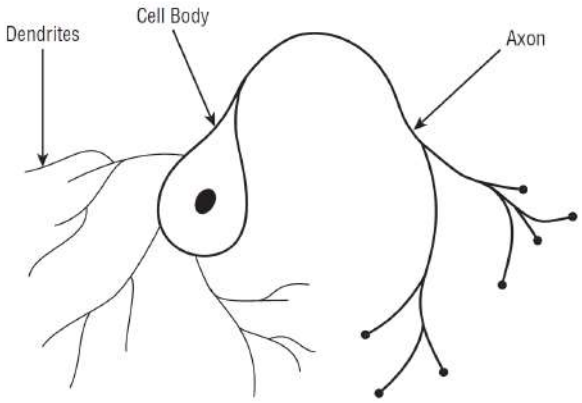


Figura 2.8: Estructura básica de una neurona [5].

En el cerebro humano existen redes de millones de neuronas conectadas de cierta manera para poder realizar todas sus funciones. Estas grandes redes de neuronas le permiten al cerebro procesar toda la información sensorial que se percibe del ambiente y tomar decisiones. La capacidad de aprender, de tomar decisiones, la consciencia y todos los atributos cognitivos del ser humano ocurren gracias a estas estructuras complejas que existen en el cerebro.

Con esta estructura en mente se planteó un técnica de aprendizaje automático que intenta copiar este comportamiento de las neuronas [6], para aprovechar su capacidad de procesamiento. Es de esta manera como nacen las Redes Neuronales Artificiales, que se han convertido una de las técnicas mas populares por su gran potencial para resolver problemas y su gran variedad de aplicaciones.

Con el fin de lograr aproximar esta arquitectura se plantea una unidad básica de cálculo llamada perceptrón, la cual se comporta más o menos como una neurona. En la figura 2.9 se muestra la estructura del perceptrón, donde al igual que las neuronas consta de tres partes principales en su funcionamiento. Se tiene un vector de entradas $\mathbf{x} = [1, x_1, x_2, ..., x_n]^T$, una salida $\mathbf{o}$ y una sección de procesamiento compuesta por la suma net y la función de activación $\sigma(net)$.

El perceptrón realiza un mapeo de un vector de valores de entrada y a una única salida en un intervalo cerrado que generalmente es [-1,1] o [0,1]. Para logralo primeramente se calcula la suma **net**, que consiste en la suma de la multiplicación de cada uno de los elementos del vector $\mathbf{x}$ por los elementos del vector $\mathbf{w} = [w_0, w_1, w_2, ..., w_n]^T$. El vector

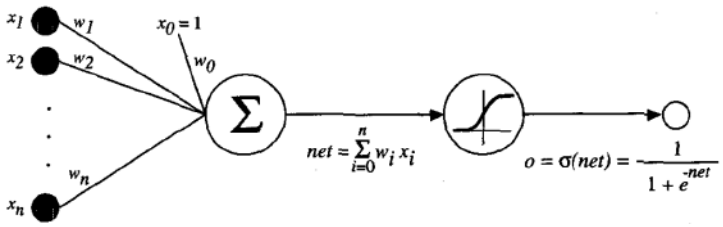


Figura 2.9: Estructura del perceptrón con función de activación tipo sigmoide [6].

w se conoce como el vector de pesos, este vector es calculado utilizando los datos de entrenamiento. Posteriormente, el valor de la suma *net* se mapea a un valor dentro del intervalo [-1,1] o [0,1] utilizando lo que se conoce como la función de activación. Esta función generalmente es la función signo, la función sigmoidea o la función tangente hiperbólica.

Esta estructura del perceptrón tal como se muestra en la figura 2.9, no es muy efectiva en la tarea de aproximar funciones complejas. La verdadera efectividad de esta técnica se obtiene cuando se utiliza un arreglo de varios perceptrones en forma de red. Esta red de perceptrones es lo que se conoce como Red Neuronal Artificial. En la figura 2.10 se tiene una red simple, se dice que se tiene una red de 3 capas, una capa de entrada de 4 neuronas, una capa oculta de 3 neuronas y una capa de salida de 1 neurona.

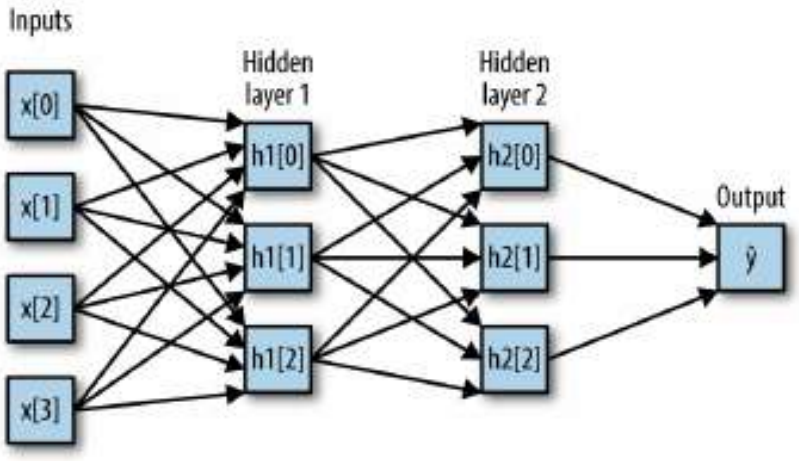


Figura 2.10: Red Neuronal Artificial con dos capas ocultas [3].

Esta técnica es más compleja que las que se mencionan anteriormente, ya que cuenta con una gran cantidad de parámetros a configurar. Algunos de estos parámetros principales del modelo son la cantidad de capas a utilizar, la cantidad de neuronas por capa y la función de activación en cada capa.

Capítulo 3

Infraestructura de experimentación

En este capítulo se brinda una descripción de la infraestructura que fue necesario configurar para el desarrollo de este proyecto. Se describen aquellos sistemas con los cuales se realizan trabajos previos y arreglos necesarios para realizar este proyecto. En la primera sección se describen los trabajos realizados en la calibración de algunos parámetros físicos de la mano DLR Hit II. Luego se menciona el papel del simulador del robot humanoide en el desarrollo del proyecto. Finalmente se presenta una descripción sobre los pasos realizados en la implementación de un sistema de visión para la captura de pose del objeto.

3.1 Configuración del robot

Para poder poner a funcionar el robot, fue necesario configurar algunos parámetros físicos en el sistema de control propuesto por Federico Ruiz. Esto porque su tesis [1] fue desarrollada con el robot TUM Rosie, y este usa como extremidad el mismo Brazo KUKA LWR+, pero utiliza la mano DLR Hit y no la mano DLR Hit II como la que se tiene en el Arcos Lab. Estas manos tienen características físicas diferentes. Este sistema de control que plantea Federico Ruiz en su tesis, utiliza control por impedancia, por lo que el robot debe tener conocimiento sobre los parámetros físicos del efector final a utilizar, en este caso la mano DLR Hit II. Los parámetros que interesan en este caso son la masa total de la mano y las coordenadas de posición de su centro de masa.

Para estimar estos parámetros de la mano se utilizan los sensores que tiene disponibles el brazo KUKA del robot. Así que se conecta la mano en el brazo y se miden los valores de torque y fuerza en la trama final del brazo y se calcula la información deseada. Se aprovechan los sensores del robot para estimar estos parámetros, ya que se asegura que los datos sean consistentes con la información utilizada durante el control. Ya que se recolecta la información con los mismos sensores que se utilizan para realizar el control.

Para este experimento de toma de datos, se desplaza el brazo en varias posiciones, con el fin de alcanzar las poses de la mano DLR Hit II deseadas para realizar medición. Las

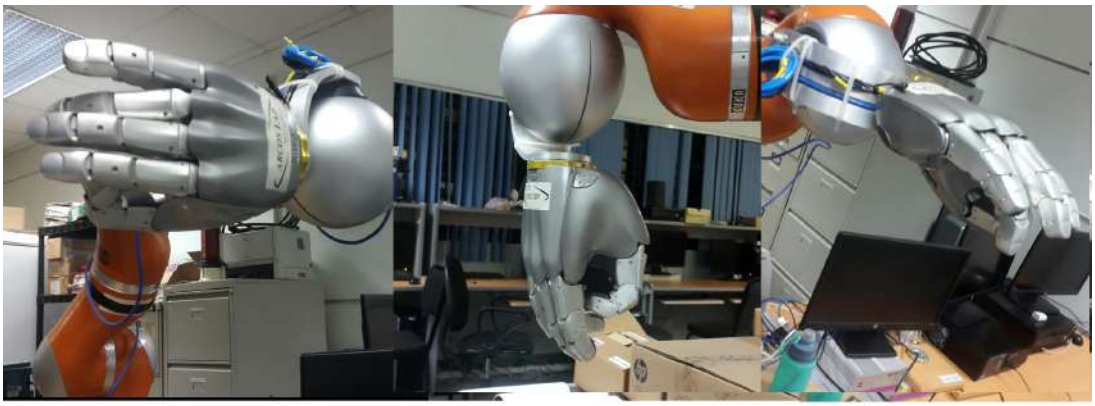


Figura 3.1: Poses de la mano DLR Hit II, para el cálculo del centro de masa en una configuración definida.

poses que se buscan son aquellas en donde uno de los ejes (de la trama de la mano) esté paralelo a la dirección del vector de gravedad. En este paso se logra estimar la masa directamente, ya que la fuerza de gravedad se mide únicamente en uno de los ejes, además el cálculo de torque también se simplifica.

En la figura 3.1 se muestran tres diferentes poses en las que se colocó la mano para recolectar datos, se tiene que en cada una de estas poses se tiene un eje diferente alineado con el vector de gravedad. De una sola pose se pueden obtener todos los datos de la mano deseados. Se utilizan las poses con ejes diferentes para evitar la influencia de la imprecisión de alguno de los sensores en una de las poses, se busca generalizar los datos para todo el área de trabajo.

Tabla 3.1: Datos de primera orientación

Resultado	F_x	F_y	F_z	T_x	T_y	T_z
Promedio	1.029	1.445	17.836	2.995	0.224	0.123
Desv.Est	1.54	1.09	0.39	0.09	0.46	0.13
Desv.Est %	150.06 %	75.34 %	2.17 %	2.96 %	207.18 %	104.70 %

*La promedio de la fuerza tienen unidades de N, el torque tienen unidades de $N \cdot m$. Los datos de fuerza y torque tienen incertidumbres iguales a ± 0.001.

Para la toma de datos se mantiene la orientación con uno de los ejes paralelos a la dirección del vector de gravedad y se mueve a diferentes posiciones. Para cada pose que se realiza en una orientación se toma los datos que reporta el Brazo KUKA LWR+ y se tabulan tal como se muestra en las tablas 3.1, 3.2 y 3.3. En donde cada tabla representa el resumen de valores que se obtuvo al escoger una orientación y tomar datos de diferentes posiciones.

En las tablas 3.1, 3.2 y 3.3 se puede ver en algunos casos la magnitud de el porcentaje de desviación estándar (Desv.Est %) presenta valores muy elevados, pero son condiciones esperadas. Esta situación se da en aquellas magnitudes que debería ser cero y por imprecisiones y ruido en los sensores no lo son. Por ejemplo en la tabla 3.1, se tiene la mano en una orientación en la cual el eje z se encuentra alineado con la dirección del vector de

Tabla 3.2: Datos de segunda orientación

Resultado	F_x	F_y	F_z	T_x	T_y	T_z
Promedio	0.07	20.11	1.50	1.86	0.03	0.27
Desv.Est	0.81	0.41	1.16	0.11	0.19	0.16
Desv.Est %	1137.92 %	2.04 %	77.60 %	6.07 %	606.25 %	68.62 %

*La promedio de la fuerza tienen unidades de N, el torque tienen unidades de $N \cdot m$. Los datos de fuerza y torque tienen incertidumbres iguales a ± 0.001.

Tabla 3.3: Datos de tercera orientación

Resultado	F_x	F_y	F_z	T_x	T_y	T_z
Promedio	1.50	18.47	1.40	1.92	0.86	0.52
Desv.Est	0.64	0.28	0.09	0.17	0.16	0.07
Desv.Est %	51.16 %	1.52 %	6.31 %	8.80 %	18.45 %	13.59 %

*La promedio de la fuerza tienen unidades de N, el torque tienen unidades de $N \cdot m$. Los datos de fuerza y torque tienen incertidumbres iguales a ± 0.001.

gravedad, por lo que idealmente solo esta componente de fuerza debería ser diferente de cero, pero no es así. En este caso de la tabla 3.1 las fuerzas F_x y F_y se ignoran para los cálculos.

Con estos datos y utilizando las ecuaciones 3.2 y 3.1 se pueden obtener los datos que se requieren para el sistema de control. En la tabla 3.4 se muestra el resumen de los datos estimados. Se tiene el valor de masa de la mano así como los valores de las coordenadas del centro de masa respecto a la última trama del brazo.

$$m = \frac{F}{g} \tag{3.1}$$

Donde para este caso se utilizó $g = 9.81 m/s^2$.

$$L = \frac{T}{F} \tag{3.2}$$

Donde F y T representan los datos de fuerza y torque medidos por el brazo, respectivamente. La variable m es la masa calculada y L una coordenada del centro de gravedad.

Tabla 3.4: Tabla resumen de resultados

Resultado	$Masa$	L_x	L_y	L_z
Promedio	1.936 ± 0.001	0.01810 ± 0.00005	0.1537 ± 0.0002	0.0958 ± 0.0002
Desv.Est	0.110	0.00891	0.0203	0.0071
Desv.Est %	5.68 %	49.23 %	13.03 %	7.360 %

*La masa tiene unidades kg y las distancias a los centros de masa unidades m.

3.2 Simulador

Durante el desarrollo de este proyecto fue necesario hacer uso del simulador [7] del robot desarrollado por Federico Ruiz. Esto porque facilita realizar las pruebas de los algoritmos de toma de datos, sin necesidad de utilizar el robot real. Las instrucciones para poder instalar este simulador se encuentran en la página del Arcos-Lab [7].

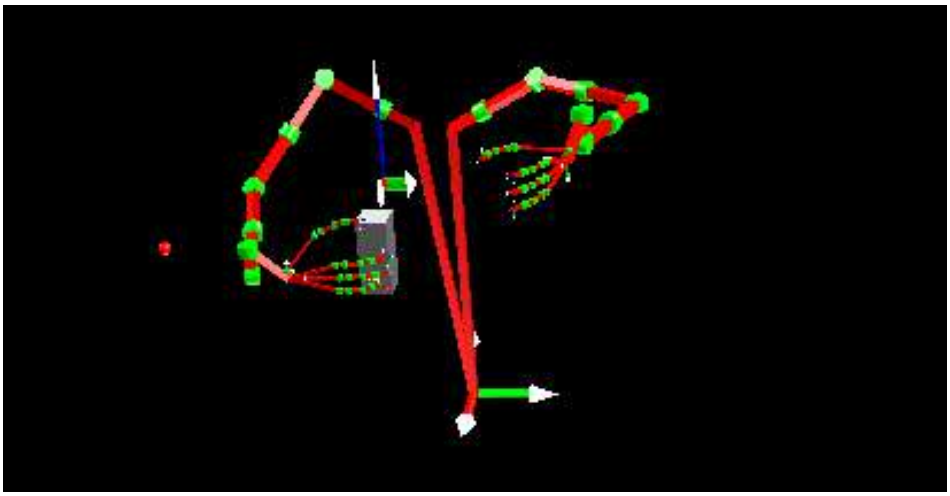


Figura 3.2: Imagen de la pantalla de visualización gráfica del simulador.

Este sistema de simulación presenta la ventaja de que se utiliza exactamente el mismo sistema de control que se utiliza en el robot real. La simulación hace uso de todos los módulos que usa el robot mientas hace rutinas de manipulación de objetos. De esta manera, es posible desarrollar el software de adquisición de datos que se logre comunicar con el sistema de la simulación y posteriormente este sistema de adquisición de datos se conecta al robot real.

Entre las principales ventajas que esto brinda, está la protección al robot, ya que al encontrarse en una fase de prueba inicial, en la simulación se trabaja en un ambiente controlado y sin riesgo de accidente. Si se hicieran las pruebas directamente en el robot físico, algún fallo en el sistema podría provocar un movimiento indeseado que dañara al robot o atentara contra la integridad física de algún miembro del laboratorio.

Otra ventaja importante es que el robot es utilizado por otros miembros del laboratorio para diferentes proyectos, por lo que su disponibilidad es limitada. Utilizando el simulador, se pueden desarrollar aplicaciones y hacer pruebas sin interferir con otros grupos de trabajo. Además, el uso del robot esta sujeto al horario de uso del laboratorio, lo que limita las horas de trabajo.

3.3 Implementación del sistema de visión artificial.

El sistema de control planteado por Federico Ruiz [1] requiere la implementación de un sistema de medición de pose alternativo, esto porque como ya se ha mencionado la estimación de pose desde el sistema de predicción presenta errores. Este sistema de medición alternativo consta de una cámara de vídeo que utiliza un sistema de visión artificial para estimar la pose del objeto.

Como parte de este proyecto se incluye como objetivo la configuración e implementación de este sistema de visión, ya que no se ha probado en el nuevo robot del Arcos-Lab. En esta sección se explica el procedimiento para instalar la cámara y además se implementa una pequeña interfaz de comunicación con el resto de sistema de control.

Esta interfaz permite comprobar el correcto funcionamiento del sistema de visión así como su compatibilidad en la comunicación de datos con el sistema de control. Este módulo de interfaz se utiliza únicamente para comprobar el funcionamiento del sistema de visión y de la comunicación correcta con otros módulos del robot, no se utiliza dentro del sistema de control.

3.3.1 Sistema de visión artificial

Para la captura de la pose real del objeto, se implementa un sistema de visión artificial utilizando **ROS (Robot Operating System)** [29], y diferentes paquetes que se explicarán a continuación para la toma y procesamiento de las imágenes para calcular el dato de pose.

La función del sistema de visión es tomar las imágenes directamente de una cámara, procesarlas y producir como salida un vector de pose. Desde el punto de vista de software, se puede subdividir este sistema de visión en varias etapas principales, tal como se muestra en la figura 3.3.

Este diagrama de flujo de información presentado en la figura 3.3 se implementa completamente en ROS. Se utiliza ROS por su flexibilidad y variedad de herramientas disponibles para aplicaciones de visión artificial. Además que brinda una gran compatibilidad y facilidad de comunicación entre diferentes paquetes. También es un software muy utilizado en el Arcos-Lab, al ser de código abierto, este se utiliza en aplicaciones de robótica dentro del laboratorio. El sistema de visión consiste en una serie de paquetes de ROS que se utilizan en conjunto para estimar la pose de un marcador utilizando imágenes obtenidas desde una cámara. El flujo de información entre los paquetes de ROS se muestra en el diagrama de flujo mostrado en 3.3. El inicio de este diagrama es la cámara, aquí se utilizó una cámara web de la marca havit modelo HV-N606 , para recolectar imágenes. El paquete **Gscam** [30] se encarga de comunicar dentro del ROS la imagen de la cámara y publica además la información de calibración de la cámara. Esta información de la cámara la toma de un archivo de calibración llamado **Calibratiom_info**. El archivo de calibración

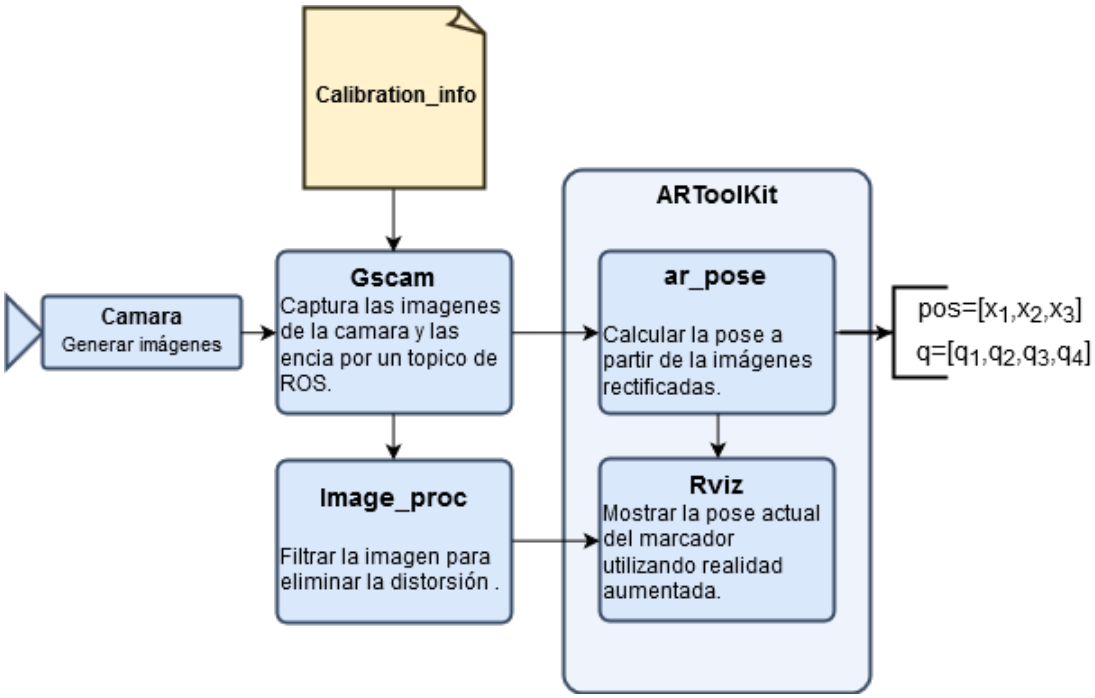


Figura 3.3: Diagrama del sistema de visión.

Calibratiom_info se genera automáticamente con un procedimiento que se explica en la sección 3.3.2.

El paquete **Image Proc** [31] se utiliza para corregir la distorsión de las imágenes de la cámara. Y comunicar una imagen sin distorsión al paquete **Rviz**, el cual ofrece una herramienta de visualización de la pose del objeto. Este paquete utiliza realidad aumentada para sobreponer un eje de referencia sobre le marcador para ilustrar la pose del objeto, tal como se muestra en la figura 3.5.

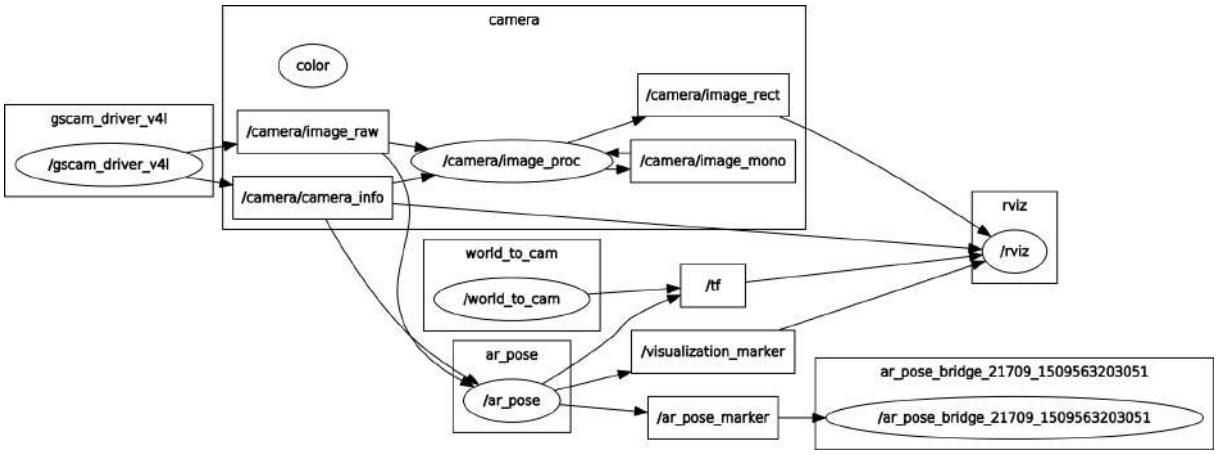


Figura 3.4: Diagrama de los nodos en ROS del Sistema de Visión.

El paquete **Ar pose** [32] se encarga de estimar la pose a partir de imagen y la información de calibración que obtiene del paquete Gscam. Ar pose utiliza dos vectores, para publicar la pose en un tópico de ROS. Se publica un vector de posición de la forma

pos= $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$, donde cada x_i representa una de las tres coordenadas de posición del centro del marcador respecto a la cámara en un espacio 3D. Por otra parte la orientación del marcador se publica como quaternio de la forma **q**= $\begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$, en donde posteriormente deberemos transformar a ángulos de rotación.

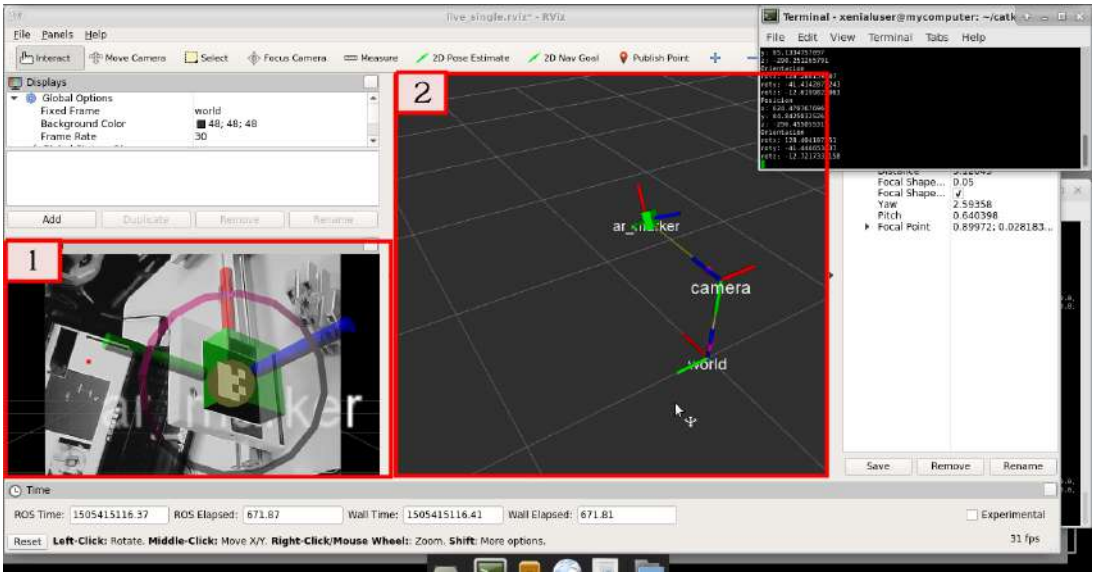


Figura 3.5: Visualizaciones de la pose de un objeto en realidad aumentada, en Rviz

Junto al paquete Ar pose, se utiliza **Rviz** [33], un paquete que brinda un ambiente de visualización en 3D. En este caso Ar pose utiliza Rviz para brindar al usuario una visualización en realidad aumentada de la pose del objeto, junto con una representación de las tramas del objeto y la cámara, lo que facilita la compresión de la información y permite comprender el correcto funcionamiento del cálculo de la pose.

En la figura 3.5 se muestra la vista general de Rviz, que contiene las dos ventanas principales de visualización. En la ventana 1 se tiene una vista de realidad aumentada de la pose del marcador, se ilustra con tres líneas los ejes de la trama. En la ventana dos se representan las tramas del objeto y de la cámara unidas por una linea amarilla, esta representación muestra en tiempo real los movimientos del objeto en el espacio, manteniendo la trama de la cámara estática. Esta visualización se muestra en tiempo real mientras se mueve el objeto.

3.3.2 Procedimiento calibración de la cámara

Como parte del procesamiento de las imágenes en el sistema de visión, es necesario realizar una rutina de calibración de la cámara. Esta rutina se explica en detalle en [34], donde el algoritmo realiza la calibración al tomar diferentes muestras de fotografías capturadas desde la cámara.

En la figura 3.6 se muestra el patrón utilizado para calibrar la cámara. Este patrón es un tablero de 9x7 cuadrados, el cual se puede imprimir en cualquier dimensión e indicarle al

software la medida real de la longitud del lado de unos de los cuadros en el papel impreso. Este patrón es conocido por el software por lo que al tomar muestras de imágenes logra estimar la distorsión del patrón en las muestras.

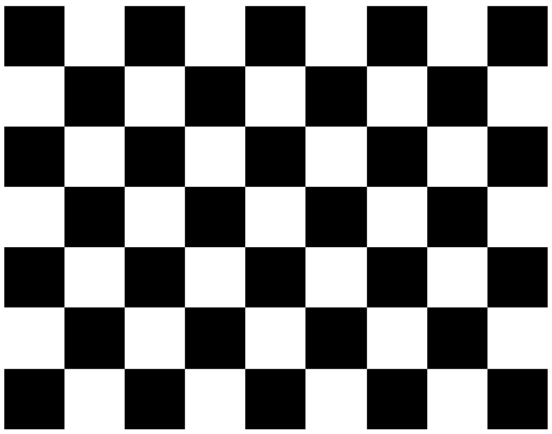


Figura 3.6: Checkerboard 8x6 para la calibración de una cámara.

El software es capaz de interpretar la imagen y medir los niveles de distorsión sobre la imagen. Tomando diferentes fotos de muestras sobre el patrón la aplicación es capaz de calcular como se distorsionan las lineas rectas del patrón utilizado. Esta información le permite calcular las constantes $k_1, k_2, k_3, k_4, k_5, k_6$ y p_1 y p_2 del filtro explicado en [34].

En la figura 3.7 se tiene la ventana de la aplicación de calibración en donde se le presenta el patrón. Como parte del procedimiento, la aplicación coloca puntos de colores en los vértices internos del patrón y los une con lineas, de esta manera se asegura que la aplicación está identificando el patrón de manera correcta.

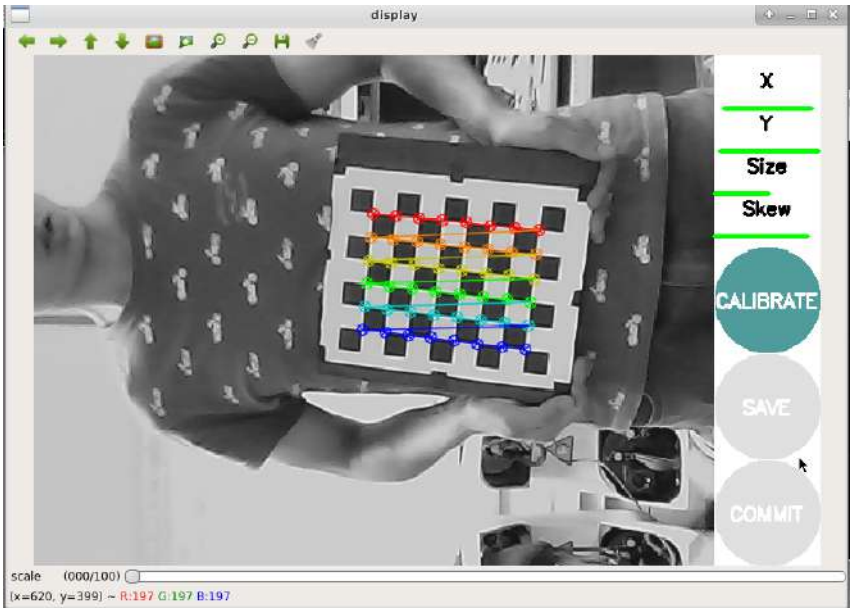


Figura 3.7: Aplicación de calibración de la cámara.

Para obtener resultados correctos de la rutina de calibración es importante mover y rotar el patrón impreso a lo largo del campo visual de la cámara, para que la aplicación tenga la mayor cantidad de información posible para calcular la información de calibración. Esta aplicación cuenta con 4 indicadores en la derecha de la ventana, que muestran cuando ya se tiene información suficiente para estimar los datos de corrección de la imagen. Para lograr suficiente información en los indicadores **X** y **Y** es necesario desplazar el patrón de izquierda a derecha y de arriba hacia abajo sobre todo el campo visual de la cámara. Para mejorar el indicador **Size**, se tiene que alejar y acercar el patrón de la cámara, y finalmente el indicador **Skew**, se logra mejorar al rotar el patrón en diferentes direcciones.

Cuando se tiene información suficiente el botón **Calibration** cambia a color verde como se muestra en la figura 3.7. Posteriormente se presiona este botón en la ventana y la aplicación se toma unos segundos en calcular la información de calibración de la cámara. El paso final es presionar el botón **Save** con el cual la aplicación guarda la información automáticamente en el archivo **Calibration_info**.

3.3.3 Módulos de prueba del sistema de visión.

En esta sección se explica el sistema planteado para realizar pruebas de comunicación entre el sistema de visión y el resto del sistema de control. Este sistema consiste en diferentes módulos que se comunican entre ellos utilizando **Yarp** [35]. El sistema que se plantea realiza la tarea de almacenar la pose que calcula el sistema de predicción y la pose que se estima desde el sistema de visión. Yarp es el sistema para comunicación que se utiliza en el sistema de control del brazo del robot humanoide propuesto por Federico Ruiz [1].

En la figura 3.8 se muestra el diagrama de flujo de información entre los diferentes módulos. El sistema se diseña para establecer comunicación entre el sistema de visión y el control del brazo. El sistema conecta y comparte información entre el sistema de visión (módulo **ar_pose**) , el sistema de predicción (módulo **planar_slider**) y el sistema de control (módulo **slider_control**) , y finalmente genera un archivo donde se almacena la información de pose del sistema de predicción y la pose que estima la cámara.

El sistema esta compuesto de canales de comunicación entre los módulos, que permiten tener tareas ejecutándose simultáneamente y compartiendo información entre ellas. Cada módulo está diseñado para realizar una tarea específica. De manera general un módulo recibe información, la procesa y envía el resultado al siguiente módulo para ejecutar otra tarea.

3.3.4 Módulo: planar_s_bridge

Este módulo se creó para llevar la información desde el sistema de predicción hasta el módulo **data_bridge** encargado de almacenar los datos en un archivo. Su tarea es recibir

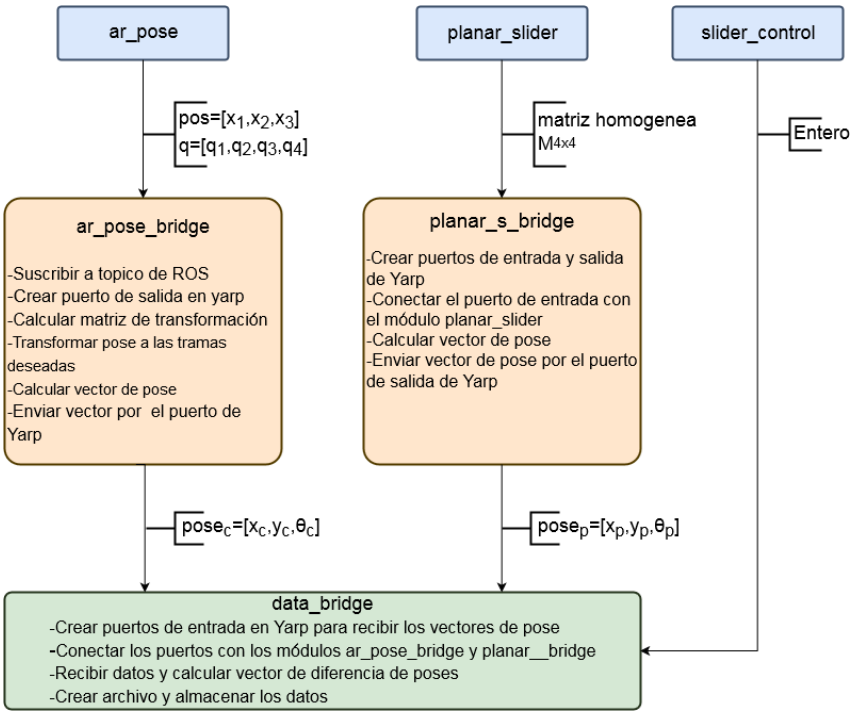


Figura 3.8: Diagrama de bloques para la adquisición de datos de error de pose.

la matriz de transformación desde el módulo de predicción y calcular los valores de traslación y rotación de la caja, para luego enviarlos preparados hasta el nuevo módulo. Para realizar esta comunicación se aprovecha la infraestructura que ya existe en el módulo de predicción utilizando Yarp.

En la figura 3.9 se explica como está diseñado el funcionamiento de este módulo. Al ser un módulo que funciona con Yarp, se deben crear y configurar nuevos puertos, capaces de recibir la matriz de transformación desde el sistema de predicción. Para que este módulo funcione, es necesario antes realizar algunos cambios en el sistema de predicción **planar_slider**. Estos cambios consisten en crear un nuevo puerto en Yarp capaz de poder transportar la matriz 4x4 de transformación.

También es necesario crear un puerto de salida para comunicar el dato de pose hasta el módulo data_bridge. En cada iteración del algoritmo se revisa si en el puerto de entrada del módulo hay un nuevo dato, si es así, se procede a hacer lectura del dato. Este dato es una matriz de transformación de la forma:

$$\mathbf{T}^{\mathbf{caja}}_{\mathbf{origen}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = \begin{bmatrix} cos(\theta_c) & -sen(\theta_c) & 0 & x \\ sen(\theta_c) & cos(\theta_c) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Al ser un movimiento en el plano, se tiene que la matriz de transformación $\mathbf{T}^{\mathbf{caja}}_{\mathbf{origen}}$ va desde el origen hasta el centro de la caja. Esta tiene la forma de una matriz con una

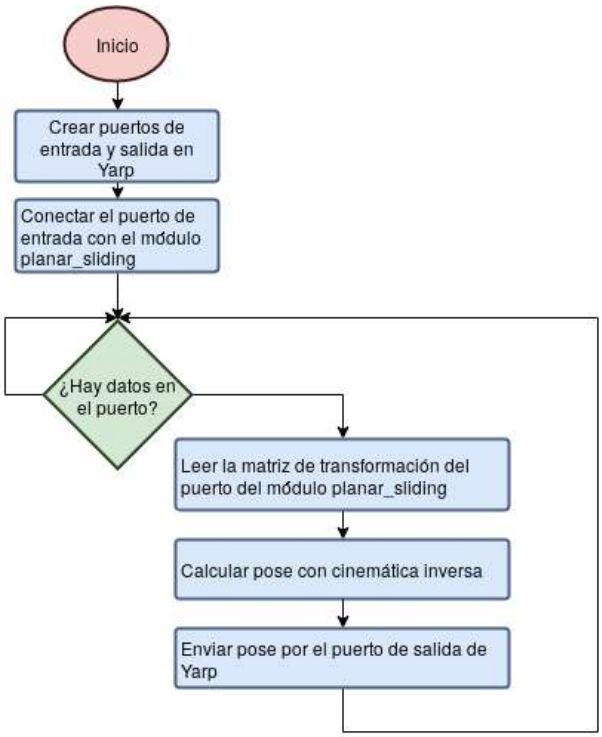


Figura 3.9: Diagrama de flujo del módudo: predictor data bridge.

única rotación en el eje z y traslaciones en el eje x y en el eje y. El origen es una trama fija y se utiliza como referencia universal para todo el sistema.

Teniendo la matriz disponible se calculan los valores de traslación y rotación de la caja respecto a la trama origen. Para lograr esto se utiliza cinemática inversa sobre la matiz de transformación:

$$x_c = r_{14} \tag{3.4}$$

$$y_c = r_{24} \tag{3.5}$$

$$\theta_c = \arctan 2(r_{21}, r_{11}) \tag{3.6}$$

Después de utilizar las ecuaciones 3.4, 3.5 y 3.6 se almacenan los 3 datos en un vector de la forma $\underline{\mathbf{pose_c}}$=[$x_c$,$y_c$, θ_c]. Finalmente, en el último paso del algoritmo se envía este vector por le puerto de salida y se vuelve a esperar un nuevo dato de entrada.

3.3.5 Módulo: ar_pose_bridge

Este módulo establece un puente de información entre el sistema de visión y el módulo encargado de generar el archivo de datos. Su tarea es recibir los vectores **pos** y **q** y

obtener de estos los valores de pose de la caja. Al ser un movimiento en un solo plano, se calculan únicamente dos coordenadas de traslación y una de rotación.

El módulo ar_pose_bridge, debe ser capaz de comunicarse con el módulo ar_pose del sistema de visión artificial , implementado en ROS, y con el módulo que almacena los datos que utiliza Yarp. Por lo que para poder integrarse al resto del sistema, debe comunicarse usando ambos métodos.

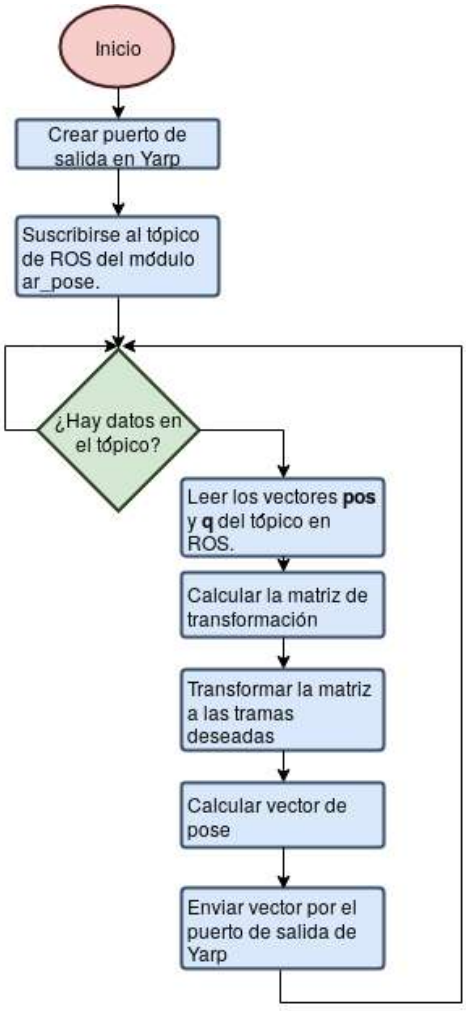


Figura 3.10: Diagrama de flujo del módulo: Camera data bridge.

El primer paso en el algoritmo será crear un puerto de comunicación en Yarp para la salida de datos al siguiente módulo. Este puerto debe ser capaz de manejar datos de la forma $\underline{\mathbf{pose_c}}=[x_c, y_c, \theta_c]$. Luego debe suscribirse al tópico */ar_pose_marker* en ROS del Sistema de Visión, para poder recibir los datos en forma de vector $\underline{\mathbf{pos}}$ y $\underline{\mathbf{q}}$, tal como se amplia en la sección 3.3.1.

La siguiente etapa del algoritmo es esperar a que se envíen datos al tópico */ar_pose_marker*, para poder tomarlos y calcular la matriz de transformación a partir de los vectores $\underline{\mathbf{pos}}$ y $\underline{\mathbf{q}}$. Como se explica en [36], el cálculo de la matriz se realiza como se muestra en la ecuación 3.7.

$$\mathbf{T}^{\mathbf{marcador}}_{\mathbf{cámara}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = \begin{bmatrix} 1-2q_3^2-2q_4^2 & 2q_2q_3-2q_1q_4 & 2q_2q_4+2q_1q_3 & x_1 \\ 2q_2q_3+2q_1q_4 & 1-2q_2^2-2q_4^2 & 2q_3q_4-2q_1q_3 & x_2 \\ 2q_2q_4-2q_1q_3 & 2q_3q_4+2q_1q_3 & 1-2q_2^2-2q_3^2 & x_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Donde los valores provienen de los vectores $\underline{\mathbf{pos}}=\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$ y $\underline{\mathbf{q}}=\begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$. tomados del módulo *ar_pose* del sistema de visión.

Con la matriz homogénea $T^{marcador}_{cámara}$, se logra conocer la relación entre la trama de la cámara hasta la trama del marcador pegado en la caja. Es decir, se conoce la pose del marcador respecto a un sistema de referencia de la cámara, por lo que se realizan las transformaciones necesarias para obtener la matriz desde la base del robot hasta el centro de la de la caja.

Transformaciones de las tramas

Para que la información de pose de la caja que brinda el sistema de visión, coincida con la información que brinda el sistema de predicción, es necesario dar las coordenadas de la caja desde su centro como se muestra en la figura 3.11. Por esta razón hay que realizar una serie de transformaciones de la información que nos brinda la cámara. En la ecuación 3.8 se muestran las transformaciones propuestas para lograr obtener T^{caja}_{origen}.

$$T^{caja}_{origen} = T^{cámara}_{origen} T^{marcador}_{cámara} T^{caja}_{marcador} \tag{3.8}$$

Del sistema de visión se conoce $T^{marcador}_{cámara}$, y conociendo la orientación y la posición del marcador en la caja, se puede calcular la matriz $T^{caja}_{marcador}$ con facilidad.

En la figura 3.11 se muestran ambas tramas, en el marcador y en el centro de la caja. La trama del marcador se tiene por defecto desde el sistema de visión que se implementó. Por lo que utilizando los ángulos de Euler sobre ejes fijos para la transformación, es necesaria una rotación de 180° respecto al eje x y una rotación de 90° respecto al eje y. Además, se hace una traslación de $-d/2$ sobre el eje z, donde d es el ancho de la caja.

Conociendo las rotaciones y traslaciones necesarias se calculan las matrices de transformación correspondientes. La rotación de 180° respecto al eje x se obtiene como se muestra en la ecuación 3.9, la rotación de 90° respecto al eje y se muestra en la ecuación 3.10 y finalmente se tiene una traslación de $-d/2$ sobre el eje z que se muestra en la ecuación 3.11.

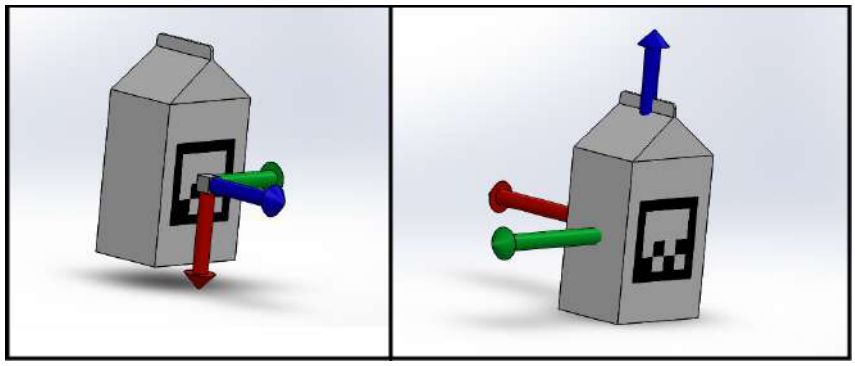


Figura 3.11: Transformaciones de la trama de la caja hasta la trama de la caja.

$$\mathbf{R_x(180^\circ)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{R_y(90^\circ)} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$\mathbf{T_z(d/2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Utilizando las ecuaciones 3.9, 3.10 y 3.11 se tiene que $T^{caja}_{marcador}$ se escribe como:

$$\mathbf{T^{caja}_{marcador}} = T_z(d/2)R_x(180^\circ)R_y(90^\circ) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -d/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

De la ecuación 3.8 para calcular T^{caja}_{origen}, ya se calcularon la matriz $T^{marcador}_{cámara}$ y la matriz $T^{caja}_{marcador}$ y es necesario ahora conocer la pose de la cámara respecto al origen para poder calcular $T^{cámara}_{origen}$. Para calcular esta transformación se tienen que es particularmente difícil determinar la pose de la trama de la cámara respecto a la trama origen, esto porque no se conoce con certeza donde esta la trama de la cámara. Además, no se cuenta con un

método confiable con el cual se pueden medir los ángulos entre ambas tramas, por lo que se decide calcular esta matriz de manera indirecta.

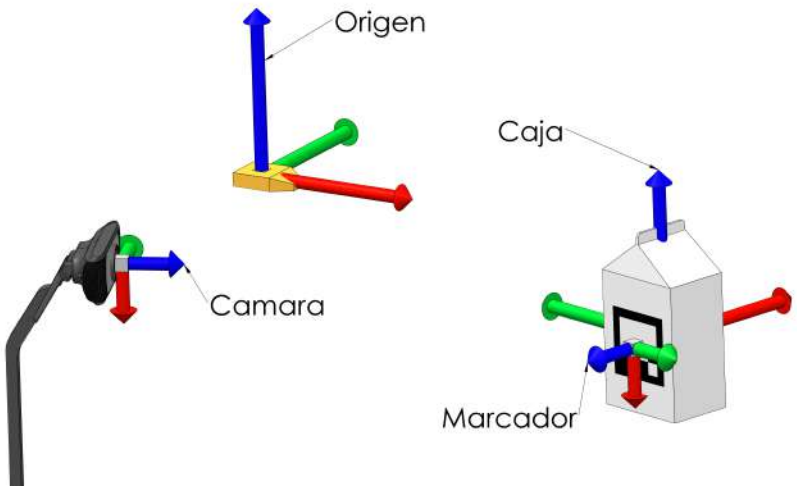


Figura 3.12: Tramas importantes del sistema.

El método que se plantea para estimar la matriz $T^{cámara}_{origen}$ es colocar la caja en una pose conocida respecto al origen. De la caja se conoce con exactitud la ubicación de su trama y dada su geometría prismática permite con facilidad colocarla en una pose deseada. Conociendo la pose de la caja se tiene una matriz T^{caja}_{origen} instantánea conocida que se denotará como $\bar{T}^{caja}_{origen}$.

Conociendo $\bar{T}^{caja}_{origen}$ utilizando una pose conocida, la matriz $T^{marcador}_{cámara}$ del sistema de visión y la matriz $T^{caja}_{marcador}$ como se explicó anteriormente en la ecuación 3.12, se puede calcular la matriz $T^{cámara}_{origen}$ como se muestra en la ecuación 3.13 que se obtiene al despejar $T^{cámara}_{origen}$ de la ecuación 3.8.

$$T^{cámara}_{origen} = (T^{marcador}_{cámara})^{-1}(T^{caja}_{marcador})^{-1}\bar{T}^{caja}_{origen} \tag{3.13}$$

Este método permite que cada vez que se cambie la cámara de posición, se podrá estimar la matriz con facilidad. Se puede establecer cualquier posición y orientación conveniente de la cámara, siempre y cuando se tenga la caja dentro del campo de visión. Se puede calcular $T^{cámara}_{origen}$ sin conocer con certeza la posición de la cámara.

3.3.6 Módulo: slider_control

Para poder recolectar los datos de pose de los módulos *ar_pose_bridge* y el módulo *planar_s_bridge* es necesario además obtener información del sistema de control diseñado por Federico Ruiz [1]. El módulo *slider_control* tal como está planteado, es el encargado de manejar el flujo de acciones y comandar señales al robot para que este se mueva.

En este módulo se describe un algoritmo que describe las etapas para poder manipular un objeto. El primer paso de esta rutina es llevar el brazo a una posición inicial preestablecida, la cual sea favorable para el robot. El siguiente paso es aproximarse a la caja hasta lograr contacto con esta. Luego el robot realiza la manipulación de la caja hasta llevarla hasta la pose final. Finalmente , cuando se logra la pose final, se lleva el brazo a una posición lejos de la caja y termina la rutina.

Del módulo *slider_control* es necesario saber la información de cuando inicia y cuando termina la labor de manipulación, es decir, cuando el robot entra en contacto con la caja y cuando termina este contacto. Por lo que es necesario realizar una modificaciones a este módulo para habilitar la comunicación de estos datos hacia el módulo *data_bridge* descrito en la sección 3.3.7.

Estas modificaciones consisten en crear este método de comunicación utilizando Yarp. En este caso se requiere enviar una variable desde el módulo *slider_control* indicando en qué etapa de la rutina se encuentra. Por lo que se crea un puerto de Yarp para transportar esta variable hacia el módulo Save Data. Estudiando el algoritmo del módulo *slider_control* es posible determinar el momento en que el robot entra en contacto con la caja y cuando se aleja de esta. Conociendo esta información es posible editar el algoritmo para que en estos puntos se envíe la variable correspondiente por el puerto de Yarp.

3.3.7 Módulo: data_bridge

La función de este módulo es recibir los datos de los diferentes módulos y almacenar la información en un archivo. En este algoritmo el primer paso es crear tres puertos de entrada de Yarp para recibir los dos datos de pose del módulo *planar_s_bridge* y del módulo *ar_pose_bridge*, y el dato de control proveniente del módulo *slider_control*. El siguiente paso es conectar estos puertos con los puertos de salida de estos tres módulos para asegurarse que hay comunicación desde el inicio.

Una vez que se logran conectar los puertos, la siguiente etapa es hacer lectura del puerto del *planar_slider* para evaluar si la variable que se obtiene indica si el robot ya entró en contacto con el objeto. Si la variable indica que aun no hay, este vuelve a revisar el puerto hasta que la variable cambie de estado. Cuando esta variable indica que si ya hay contacto, se entra a un ciclo que recolecta la información de los otros puertos que comunican la pose, y además hace lectura del puerto del *planar_slider* hasta que la variable cambie de estado e indique que se alcanzó la pose final deseada y el robot pierde contacto con la caja. Cuando esto ocurre se detiene la lectura de los puertos y se almacenan los datos en un archivo.

Del módulo *planar_s_bridge* se recibe un vector de la forma $\underline{\mathbf{pose_p}}=[x_p, y_p, \theta_p]$ y del módulo*ar_pose_bridge* se recibe un vector de la forma $\underline{\mathbf{pose_c}}=[x_c, y_c, \theta_c]$. Estos datos recibidos se almacenan en el formato de dato tipo diccionario. Cuando la caja alcanza su pose final, se detiene la recolección de datos. Finalmente, se almacena este diccionario en un archivo que pueda ser utilizado posteriormente para estudiar los datos guardados.

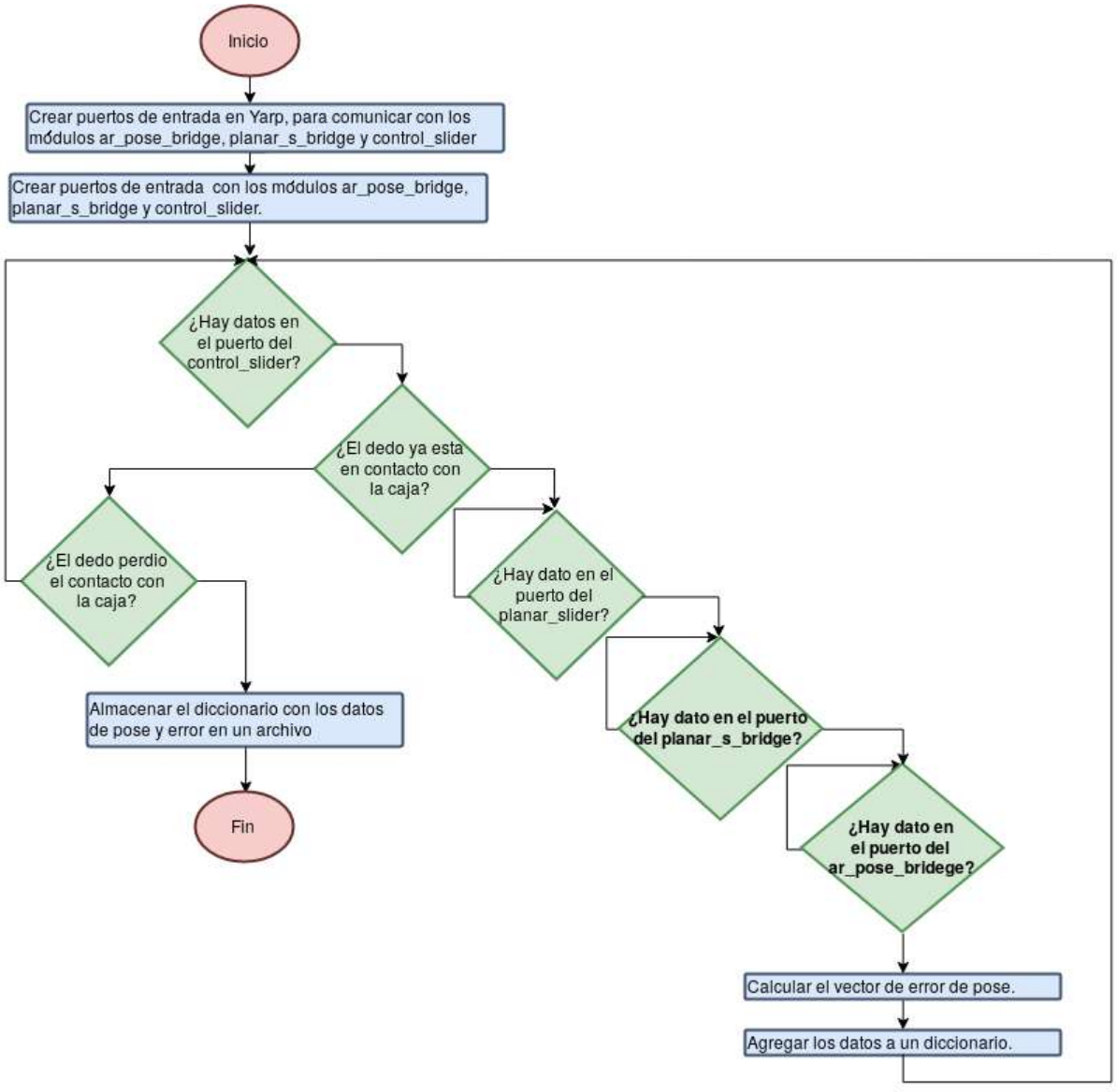


Figura 3.13: Diagrama de flujo del módulo Save Data.

Capítulo 4

Sistema compensador de error.

Se cuenta con un sistema de control que permite que un robot humanoide pueda deslizar una caja sobre una superficie utilizando el dedo de una de sus extremidades. Este sistema de control incluye un sistema de predicción de pose que tiene función de estimar la pose de la caja mientras está es deslizada por el robot. Como se ha expuesto anteriormente, este sistema de predicción utiliza un modelo de la caja con algunas debilidades, que causan que los valores de pose calculados contengan un error.

En este capítulo se explican los pasos que se realizaron para el desarrollo del sistema compensador de error para disminuir el error de este sistema de predicción de pose. En el capítulo se describen inicialmente los detalles sobre cómo se plantea la solución propuesta en este módulo compensador. Luego se profundiza en el planteamiento del algoritmo de aprendizaje automático. Finalmente se describen varios experimentos donde se utilizan diferentes escenarios de prueba para medir el desempeño del sistema de predicción de pose, y además estos experimentos buscan evaluar la habilidad del módulo compensador para corregir el error en estos escenarios.

4.1 Solución propuesta.

El objetivo del trabajo fue desarrollar un sistema que permita disminuir el error que tiene el sistema de predicción de pose de un objeto, propuesto por [1]. Se quiere que el sistema de predicción (SPP) logre estimar la pose de caja correctamente disminuyendo el error que existe en el modelado del objeto. Para atacar este problema se plantea un módulo de compensación de error que permita al SPP calcular con menor error la pose. Este sistema de compensación pretende reducir las deficiencias del sistema de predicción en la estimación de pose.

La solución propuesta en esta tesis consiste en obtener un modelo del error de predicción de pose. Este modelo permite no solo calcular la magnitud del error, sino que se pretende estimar también su dirección, para poder de esta manera compensar futuras predicciones del SPP. Como se ilustra en la figura 4.1 el objetivo del modelo de error es mejorar el

cálculo del sistema de predicción de pose (SPP), y mejorar la aproximación del valor real de la pose.

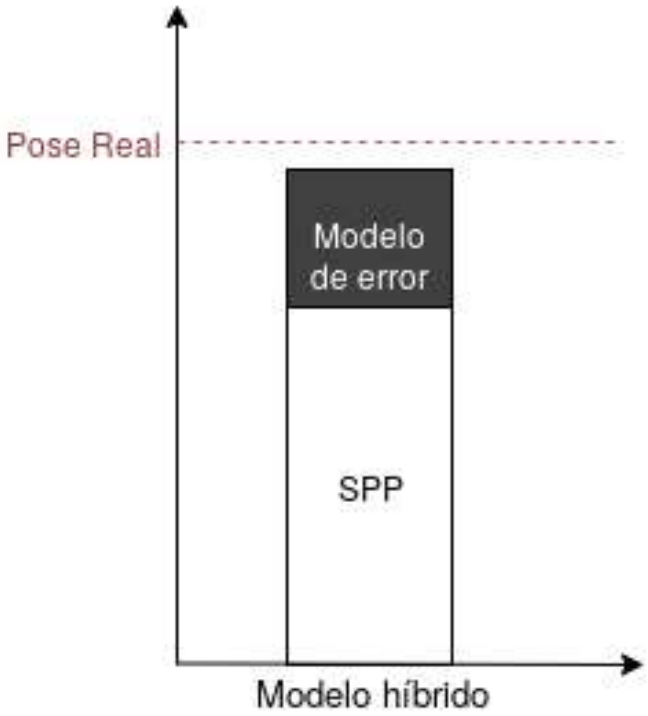


Figura 4.1: Modelo híbrido para el cálculo de la pose de un objeto.

Como se ha mencionado en el capítulo 2, la naturaleza del error no es conocida con total certeza, pero se listan las posibles fuentes de este error del sistema de predicción. Estos errores están ligados a efectos físicos que fueron desestimados en el modelo matemático utilizado para predecir el comportamiento del objeto. Obtener un modelo analítico del error, presenta un gran reto desde el punto de vista matemático y de implementación. Esto principalmente porque este error esta ligado a algunos fenómenos matemáticamente complejos que además involucran variables difíciles de medir para el robot con los sensores actuales.

Como caso de ejemplo, se puede acudir a uno de los casos mencionados en el capítulo 2. Si se quiere modelar el efecto que tiene la deformaciones de las paredes del objeto en la magnitud del error, se necesitaría una representación matemática de como se deforman estas paredes del objeto, y además encontrar la relación matemática de estas deformaciones con el valor de error. Modelar la relación de estas deformaciones y el valor del error, puede incluir efectos como la rigidez del material del objeto, las fuerzas aplicadas, el contenido del objeto (si se trata de algún recipiente), entre otras.

Para lograr esto se requiere conocer las propiedades de los materiales de cada objeto con el que se quiere trabajar, propiedades que no son brindadas directamente por el fabricante y requieren de experimentos adicionales para extraerlas. Depender de este tipo de propiedades tan especificas del objeto, hacen que el modelo sea poco practico para ser utilizado con el robot humanoide. Como se menciona en [1], se quiere que el modelo sea

compacto y fácil de adaptar a nuevos objetos, para que el robot pueda interactuar con objetos nuevos fácilmente.

Modelar esta y otras situaciones, es necesaria una investigación profunda del sistema y requiere de modelos matemáticos complejos. En donde además, implicaría la incorporación de nuevos sensores para lograr medir la deformación de las paredes u otra variable importante. Esta complejidad en modelar el error de forma analítica, guían la solución de esta tesis a otros métodos. Se quiere utilizar un método para modelar el error sin necesidad de acudir a modelos matemáticos complejos, y evitar la incorporación de nuevos sensores al robot.

Como se ha expuesto en la sección 2, una de las técnicas, que se ha popularizado en aplicaciones para modelar sistemas complejos, es el aprendizaje automático. El aprendizaje automático permite generar un modelo a partir de datos (datos de entrada/salida del modelo deseado), y no requiere de formulaciones matemáticas (ecuación del modelo como tal), como en si es el caso de los modelos analíticos.

Compensación del error.

Para plantear el módulo compensador es necesario analizar cómo se calcula la pose desde el sistema de predicción. Estudiando este sistema se sabe que este error es el resultado de los errores presentes en los cálculos anteriores. De esta manera es posible atacar los cálculos anteriores para así obtener un mejor cálculo en la pose.

Se propone inicialmente utilizar aprendizaje automático para aprender el error asociado a la pose. Pero dado el funcionamiento del sistema de predicción, se sabe que el error de pose proviene de un error previo en el cálculo de la velocidad. Por lo que se considera más conveniente atacar directamente el error que existe en el cálculo de velocidad. Si se logra mejorar la precisión en la estimación de la velocidad, se logra directamente una mejora de la precisión del cálculo de la pose.

Como se ha explicado en capítulos anteriores, la simplificación del elipsoide utilizada para el Limit Surface, se asume como una de las fuentes más importantes de error en los cálculos. Como se recuerda la velocidad es un cálculo que proviene del gradiente del Limit Suface, por lo que la velocidad se ve afectada directamente por esta simplificación con el elipsoide. Conviene por esta razón intentar corregir el cálculo de la velocidad del objeto, ya que esta contiene errores de la principal fuente de error que se considera del modelo, el Limit Surface.

De esta forma se determina que la mejor señal sobre la cual aplicar el sistema de compensación de error, es la señal de velocidad. Por lo que el sistema compensador se concentra en disminuir el error en la señal de velocidad. Como se tiene en la ecuación 4.1, se plantea que existe una función compensadora de error de velocidad v_{comp}, que permite aproximar la velocidad real v_{real} a partir de la velocidad del sistema de predicción v_{pred}.

$$v_{real} = v_{pred} + v_{comp} \tag{4.1}$$

El problema del compensador se reduce a encontrar una aproximación de la función v_{comp} que permita compensar el error en la velocidad del sistema de predicción. Para aproximar esta función es necesario generar datos de entrenamiento. Estos datos se obtienen de repetidos experimentos donde el robot manipula una caja.

Como se muestra en la figura 4.2, se presentan datos de la velocidad en el eje y de la caja tomados durante la trayectoria de la caja. En esta imagen en la parte superior se muestra en azul la velocidad real, y en rojo la velocidad estimada por el sistema de predicción (SPP). Como es evidente en esta imagen ambas velocidades presentan un comportamiento diferente por las deficiencias de este sistema de predicción. En la parte inferior de la imagen se muestra el error entre ambas señales.

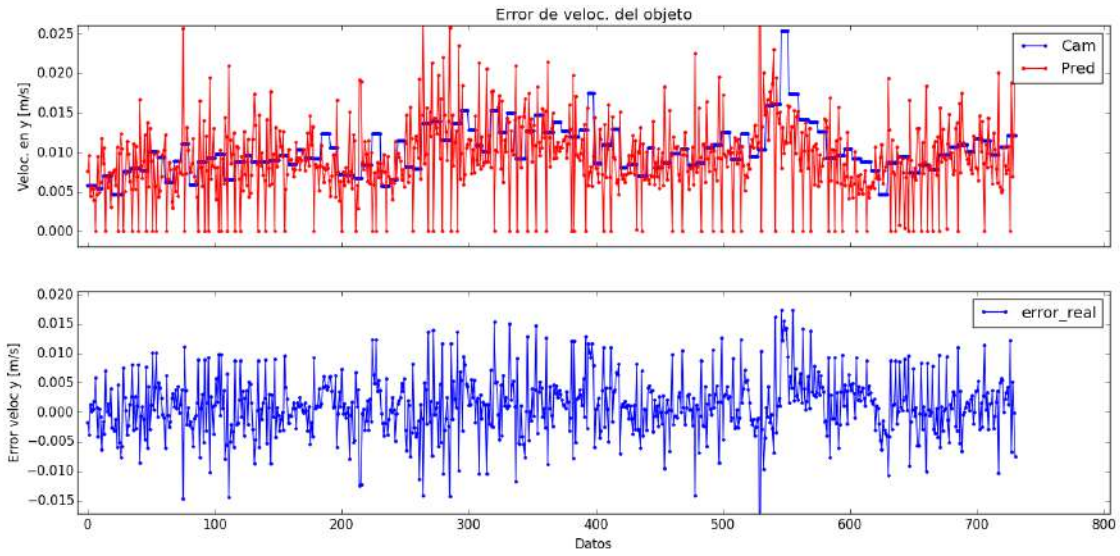


Figura 4.2: Señales de posición angular.

Para calcular los datos de aprendizaje de error es necesario conocer la pose real de la caja. Para lograr esto se utilizan los datos de pose calculados desde la cámara y se derivan respecto al tiempo para calcular la velocidad. Este dato de velocidad se toma como la velocidad real de la caja.

De esta manera lo que se quiere finalmente es que el sistema compensador genere una señal igual a la diferencia entre la señal de velocidad calculada desde la cámara y la señal de velocidad estimada por el sistema de predicción (SPP), como se muestra en la ecuación 4.2.

$$f_{comp} = f_{cam} - f_{pred} \tag{4.2}$$

Donde f_{comp} es la señal de salida del sistema compensador, f_{cam} es la señal de velocidad

calculada desde la cámara y f_{pred} es la señal de velocidad estimada desde el sistema de predicción.

4.2 Algoritmo de aprendizaje automático.

Dadas las características del problema a solucionar y tomando en cuenta las sugerencias realizadas en el trabajo de Federico Ruiz [1], se decide abordar la solución con técnicas de aprendizaje automático. Esto hace principalmente que el origen de los errores del sistema de predicción provengan de fenómenos muy complicados de modelar o muy difíciles de medir experimentalmente. Como se expuso anteriormente en el capítulo 2, el uso del aprendizaje tiene un gran alcance en múltiples áreas, en donde se atacan problemas que serían muy complejos de resolver con las técnicas tradicionales de modelado.

Las áreas de aplicaciones del aprendizaje automático son muy variadas, y en los últimos años se ha ampliado su uso gracias a que se han creado nuevas herramientas de fácil acceso para disciplinas no relacionadas con el desarrollo de software. Para el desarrollo de la solución planteada se utiliza **Keras** [37], una librería compatible con python que brinda una gran flexibilidad en la implementación de redes neuronales.

4.2.1 Planteamiento del problema en términos de aprendizaje automático.

Para el planteamiento del problema del aprendizaje del error en un sistema de predicción de pose, se acude a la definición que aporta Tom Mitchell [6]. En este libro Mitchell propone que para plantear un problema a solucionar con aprendizaje automático, a este se le deben definir tres partes fundamentales. Se define una tarea T que se quiere ejecutar, la cual se realiza con un desempeño P y este desempeño se mejora con la experiencia E. Estas tres variables describen una labor de aprendizaje, en donde se utiliza la experiencia E para mejorar el desempeño P en la tarea T.

Aplicando esta idea al problema de este trabajo se definen estas variables de la siguiente manera.

- T: La tarea consiste en disminuir el error del sistema de predicción propuesto por Federico Ruiz para la estimación de pose de una caja.
- P: El desempeño se mide como el porcentaje que se logra disminuir del error en los valores de estimación de pose del sistema de predicción.
- E: La experiencia de la cual se aprende corresponde a los datos que obtienen de los experimentos descritos en la sección 4.3 de este capítulo. Donde el robot adquiere experiencia al manipular una caja en varias ocasiones.

Cuando se define cual es la tarea T, y se elige como se va a medir el desempeño P con el cual se evalúa la tarea, y se establece una fuente de experiencia E, se procede al siguiente paso. Para definir el problema en términos concretos es de suma importancia determinar cual es función objetivo. Una función explicita que se quiera abordar con aprendizaje automático. Finalmente se elige una representación para esta función.

4.2.2 Función objetivo

Para el caso del sistema de predicción de pose (SPP), el sistema cuenta con el modelo matemático de la caja para predecir su comportamiento. Por lo que la función objetivo no es el modelo matemático de la caja como tal, sino más bien, se busca modelar aquellos efectos que incorporen error en los valores de pose de la caja estimados por el sistema de predicción. La función objetivo es una función que modela los efectos que no se incorporaron en el modelo de la caja propuesto por Federico Ruiz.

El objetivo del proyecto es disminuir el error en la predicción de pose de la caja. Como se discutió anteriormente, para lograr esto no se trabaja directamente con la corrección de la pose, sino con la estimación de la velocidad de la caja por parte del sistema de predicción. Se toma esta decisión porque el sistema de predicción no calcula pose directamente, sino que se desarrolló para calcular inicialmente la velocidad y luego la posición. Al trabajar en corregir la estimación de la velocidad, se trabaja directamente con la salida del sistema de predicción.

$$f_{Real} = f_{pred} + f_{error} \tag{4.3}$$

En la ecuación 4.3 se tiene que existe una función f_{Real} que es capaz de calcular la velocidad real de la caja. Esta se calcula como f_{pred}, que corresponde al valor de velocidad estimado por el sistema de predicción de Federico Ruiz, más la función f_{comp} que se encarga de modelar el error del SPP. La función f_{error} compensa las deficiencias del modelo f_{pred}, y permite calcular una aproximación de la velocidad real del objeto. Con la infraestructura actual del robot f_{Real}, corresponde a las mediciones de velocidad obtenidas desde la cámara.

$$f_{error} = f(x_1, x_2, ..., x_n) \tag{4.4}$$

Por lo que la función objetivo en este proyecto es f_{error}. y este depende de un vector $x_1, x_2, ..., x_n$, como se muestra en la ecuación 4.4. Como se discutió anteriormente, la hipótesis es que el vector de entradas al sistema de aprendizaje $x_1, x_2, ..., x_n$ contiene variables como (F_x, F_y, m) fuerzas aplicadas a la caja, punto de aplicación de la fuerza a la caja r_x, r_y, r_z, y la velocidad del dedo, entre otras variables.

4.2.3 Representación de la función.

Luego de definir la función objetivo, se debe elegir una representación para esta. Para tarea de aprendizaje de error de velocidad, planteada en esta tesis, se utilizan técnicas de regresión. Como se representa en la figura 4.3 se quiere que el sistema de aprendizaje automático genere una función continua que permita mapear las variables de entrada en valores de error. Esa función corresponde a la relación que existe entre una pose estimada $(x_1, x_2, ..., x_n)$ y su error asociado a cada coordenada $(\Delta v_x, \Delta v_y, \Delta\omega)$.

Figura 4.3: Entradas y salidas del sistema de aprendizaje automático propuesto.

Selección de la técnica.

Como se expuso en el capítulo 2 existen varias técnicas para realizar regresión. Cada una de ellas tiene un principio de funcionamiento diferente en la forma matemática de abordar el problema. Seleccionar una técnica de aprendizaje puede ser una tarea difícil, ya que existen muchas opciones diferentes y estas tienen ventajas y desventajas para cada problema a resolver.

Clasificado por tareas, el problema de esta tesis se trata como una tarea de aprendizaje supervisado y de regresión. Para tareas de regresión existen también varias opciones que se pueden utilizar. Se han consultado varios trabajos [38], [39], [40] [41],[42] como base, para escoger un criterio de selección de la técnicas más adecuada. La mayoría de autores coinciden en que la selección de una técnica depende de cada problema y los datos que se tengan, y en muchos casos, la única forma de determinar cual es la mejor técnica es experimentando con cada una de ellas.

El problema cuenta con datos entradas/salidas que son variables continuas y presentan una relación no-lineal, además el modelo a entrenar cuenta con múltiples entradas y múltiples salidas. Para este mismo problema se pueden aplicar varias técnicas, en donde destaca entre las más populares Máquinas de Soporte Vectorial-Regresion (SVR) [43] y Redes Neuronales (ANN) [44]. Usar ANN presenta la ventaja sobre SVR, que suele resolver problemas más complejos. Sin embargo, con la información que se tiene actualmente no se puede decir cual es la mejor técnica a utilizar entre SVR y ANN (también pueden

usarse técnicas como arboles de decisión o sus variantes [45]).

En muchas aplicaciones, las redes neuronales suelen ser mejores para resolver problemas de alta complejidad frente a SVR (u otras), pero este argumento no descarta que otras técnicas también sean utilizables para este problema, ya que además, no se conoce la complejidad para cada algoritmo de aprender este problema. La falta de experiencia previa en este problema específico hace que la única manera de determinar cual es el mejor algoritmo, es utilizar varios y evaluar los resultados. Para los alcances del proyecto se escoge una técnica (redes neuronales), y se demuestra que esta técnica sí se puede utilizar (logra aprender el error), pero el problema queda abierto a trabajos futuros para poder ser resuelto con otras técnicas.

No existen trabajos previos con estos datos de entrenamiento específico. Se consultan trabajos similares en donde se utilizan ANN con éxito para diversos problemas de aprendizaje de error, pero no se puede extrapolar esta experiencia a nuestro trabajo ya que los datos son diferentes. Como se menciona en [38], [39], [40] [41],[42], la forma para determinar cual es la técnica más adecuada, es implementarlas con los datos disponibles y escoger una basados es los resultados.

Redes Neuronales.

En el área de redes neuronales artificiales existen varias clases de redes. Las aplicaciones de las redes neuronales son muy diversas y han dado origen a una gran cantidad de tipos de redes. Existen algunos tipos que se han vuelto muy populares en algunas aplicaciones especificas. Como es el caso de las redes Convolucionales (CNN), que se popularizan por su uso en aplicaciones [46],[47],[25] que implican procesamiento de imágenes. También se tienen las redes Recurrentes (RNN) y LSTM (RNN), que se han utilizado ampliamente en aplicaciones que involucren el procesamiento de texto y audio [48],[49],[50], en donde se trabajan con datos secuenciales. Esto no limita estas redes a ser utilizadas únicamente en estas áreas, ya que sus aplicaciones son diversas.

Para el caso se regresión donde se utilizan variables de entrada/salida continuas, en varios trabajos utilizan redes del tipo redes de Función de Base Radial (RBFN) [51],[22], Perceptrón Multicapa (MLP) [52] o incluso RNN [25]. Pero, no se encuentra información en los trabajos consultados, que garantice que un tipo especifico de red neuronal es el mejor para la aplicación en esta tesis. Nuevamente el tipo de red más adecuado, depende fuertemente de la aplicación y de los datos disponibles [53],[54],[55],[56],[57],[58]. Por lo que por ejemplo, para decidir entre MLP y RBFN es necesario probar con ambas y analizar el resultado. En los resultados se de esta tesis se demuestra que con la arquitectura de MLP utilizada, es posible aprender el error. El problema de la tesis queda abierto a comparar estos resultados presentados, contra otros tipos de ANN.

En la tesis se muestran los resultados obtenidos con redes neuronales tipo MLP, y se demuestra con validación cruzada que esta elección de algoritmo permite aprender el error deseado. Esta solución no descalifica en trabajos futuros la utilización de otras técnicas

de aprendizaje automático, u otro tipo de red neuronal. Mas bien, ya que se demostró que se puede aprender el error, una segunda parte para la investigación, puede estar en la línea de realizar una comparación entre diversos métodos para abordar el problema.

4.2.4 Topología de la red neuronal.

Como se ha mencionado las redes neuronales tipo perceptrón multicapa (MLP) están conformadas por varias capas de neuronas. Se necesita una capa de salida, una capa de entrada y al menos una capa oculta. Dos de los parámetros más importantes de configurar en este tipo de redes neuronales son la cantidad de capas y de neuronas, y la función de activación que se utilice en las neuronas de cada una de las capas. Estos parámetros afectan la capacidad de la red MLP de aprender el modelo deseado.

La selección de estos parámetros depende de la naturaleza de cada problema a resolver. La selección de la(s) función(es) de activación, así como de la cantidad de neuronas y cantidad de capas, depende de los datos de entrenamiento propios de cada problema. En [54] y [53] se utilizan métodos similares para determinar cuales parámetros se ajustan mejor a los datos disponibles.

Entradas al sistema.

Las fuentes de error del sistema no son conocidas con certeza, pero con el análisis de las posibles fuentes de error del modelo descrito en la sección 2.2, se plantean varias hipótesis. Como candidato principal de fuentes de error en el sistema de predicción, la simplificación matemática realizada sobre el Limit Surface, es de gran importancia. La diferencia geométrica entre el Limit Surface Real y la simplificación de elipsoide, se plantea como el generador más importante de error a considerar.

Tomando esta hipótesis sobre el Limit Surface en cuenta, se estudiarían como señales influyentes en la magnitud del error, aquellas que afecten la geometría del Limit Surface, ya que cualquier cambio en esta geometría provoca que la aproximación de elipsoide difiera aun más de la representación real. Estas señales de fuente de error son mapeadas por el sistema de compensador a un valor de error.

Como se menciona el la sección 2.1.1 la geometría real del Limit Suface depende de las fuerzas aplicadas al objeto en su centro de masa. Por lo que se propone el vector $\underline{\mathbf{F_o}} = [F_x, F_y, m]$, fuerza en el eje x, fuerza en y y momento respectivamente, como señales fuentes de error del sistema. Este vector corresponde a las fuerzas aplicadas por el dedo al objeto, trasladadas al centro de masa del objeto. Este vector permitiría estimar la señal de error en los diferentes puntos del Limit Suface.

El Limit Surface también se ve afectado por el punto en el objeto sobre el cual se aplica la fuerza. La posición del dedo respecto al centro de masa de la caja afecta directamente el cálculo de las fuerzas en el centro de masa del objeto. Esta posición sobre la caja también

podría provocar movimientos indeseados durante la trayectoria del objeto, también puede variar la fricción dependiendo del punto que se toque del objeto, o generar inclinaciones indeseadas del objeto dependiendo de la altura a la que se aplique la fuerza. La posición del dedo sobre la caja puede generar varios cambios en el Limit Surface esperado. En el sistema de compensación de error se incluye el vector $\underline{\mathbf{r}}$=$[r_x, r_y, r_z]$ que corresponde a las tres coordenadas de posición del dedo respecto al centro de masa del objeto.

El control sobre la caja es un control por velocidad, así que la trayectoria que describe la caja se controla con la velocidad que le aplica el robot con el dedo. Es por eso que la velocidad que aplica el dedo $\underline{\mathbf{v_d}} = [vd_x, vd_y]$,se considera como una de las variables importantes de incluir como entradas dentro del modelo de aprendizaje de error.

Para determinar cuales otras entradas pueden aportar información sobre el valor del error, se aplicó un método probabilístico conocido como **Dependencia Mutua** [59]. Esta herramienta permite evaluar el grado de relación entre dos variables. Un grado alto significa que existe una mayor relación entre las variables. De este gráfico se descartan todas las variables que tienen un valor inferior a 0.1.

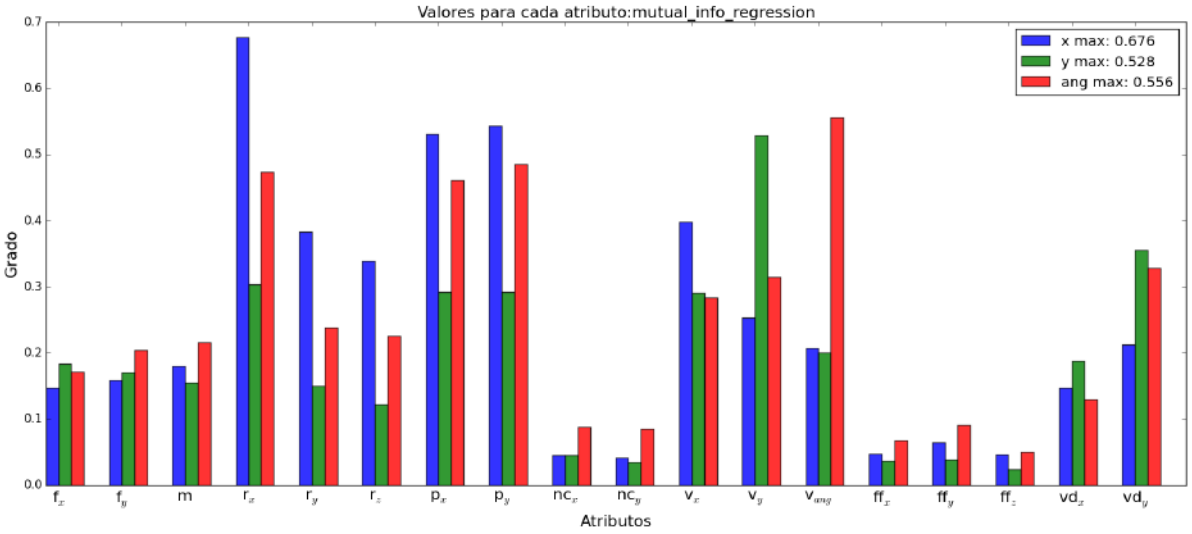


Figura 4.4: Valores de dependencia mutua.

En la imagen 4.4 se tiene un gráfico de barras donde se ilustra los valores de dependencia mutua entre diferentes variables y los valores de errores en las tres componentes (x, y, ang)de velocidad. En total se analizaron 18 entradas al sistema y se evaluó el grado de relación según su dependencia mutua.

Como se ve el el gráfico, se analiza la fuerza en el centro de la caja donde f_x es fuerza en x, f_y es fuerza en y, y m es el momento. La posición del dedo respecto al centro de la caja r_x,r_y,r_z, coordenada x,y, z respectivamente. La posición de la caja sobre la mesa p_x,p_y posición en x y en y respectivamente. La dirección de la velocidad de la caja en x y y nc_x,nc_x. La fuerza aplicada por el dedo a la caja ff_x, ff_y, ff_z en x, y, z. La velocidad del de dedo vd_x, vd_y ,vd_z en x, y, z. Finalmente también se agrega la velocidad calculada por el sistema de predicción v_x, v_y ,v_{ang}, velocidad en x en y y angular respectivamente.

De los valores de Dependencia Mutua presentados en la figura 4.4, se extraen conclusiones importantes. El primer dato es que los resultados de este método sugieren incluir como entrada del sistema de aprendizaje, los valores de velocidad del centro de masa de la caja (calculados por el SPP), y la posición de la caja sobre la mesa. Este vector de velocidad, obtiene valores altos de dependencia mutua con los valores de error. Otro dato que aporta este gráfico es que se confirma que las entradas propuestas inicialmente ($\underline{\mathbf{F_o}}, \underline{\mathbf{v_d}}, \underline{\mathbf{r}}$), aportan información importante para calcular el error de velocidad.

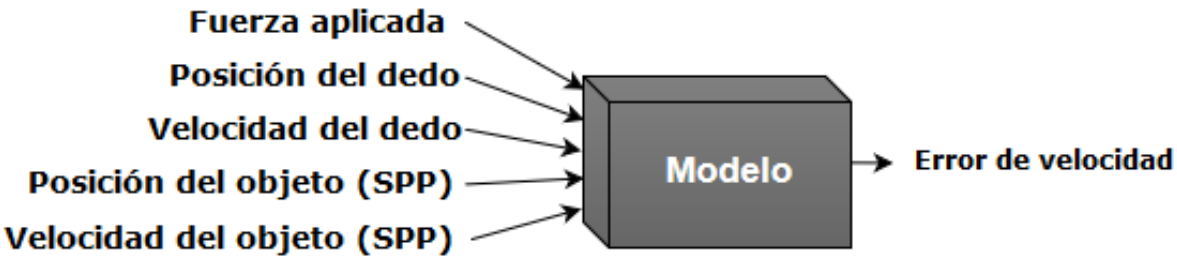


Figura 4.5: Esquema de entradas y salidas del modelo propuesto.

En la figura 4.5 se tiene la forma del modelo propuesto en esta tesis. La red neuronal se utiliza para aprender la relación que existe entre estas variables de entrada y los valores del error dados por el SPP.

4.2.5 Función de activación.

El primer parámetro que se selecciona en la topología de la red neuronal es la(s) función(es) de activación a utilizar. Para determinar cual es la función de activación más conveniente, se utiliza un método perimétrico como el empleado en [54]. Este método consiste en definir una topología de red y entrenar el modelo para diferentes funciones de activación. La elección de la función de activación más adecuada, se basa en las curvas de error de entrenamiento de cada modelo.

Para los experimentos en este trabajo se define una red con una única capa oculta de 100 neuronas y se entrena durante 10000 iteraciones. Los resultados de estas pruebas se muestran en la figura 4.6, en donde se grafica el valor del error cuadrático medio de cada modelo durante los 10000 ciclos de entrenamiento. Se utiliza escala logarítmica en el eje vertical para ilustrar de una mejor manera las diferencias entre las curvas de error.

Durante los experimentos se cambiaron las funciones de activación tanto en la capa oculta como en la capa de salida. Se utilizaron 4 funciones de activación diferentes y se generaron 9 modelos diferentes. Se utilizó las funciones sigmoide (S), lineal(L), tangente hipérbola (T) y la función lineal rectificada ReLU (R). Los modelos mostrados se nombran de la forma **S_L_**, que significa que usa la función sigmoide en la capa oculta y la función lineal en la capa de salida, de la misma forma el modelo **T_R_** utiliza la función tangente hiperbólica en la capa oculta y la función lineal rectificada en la capa de salida.

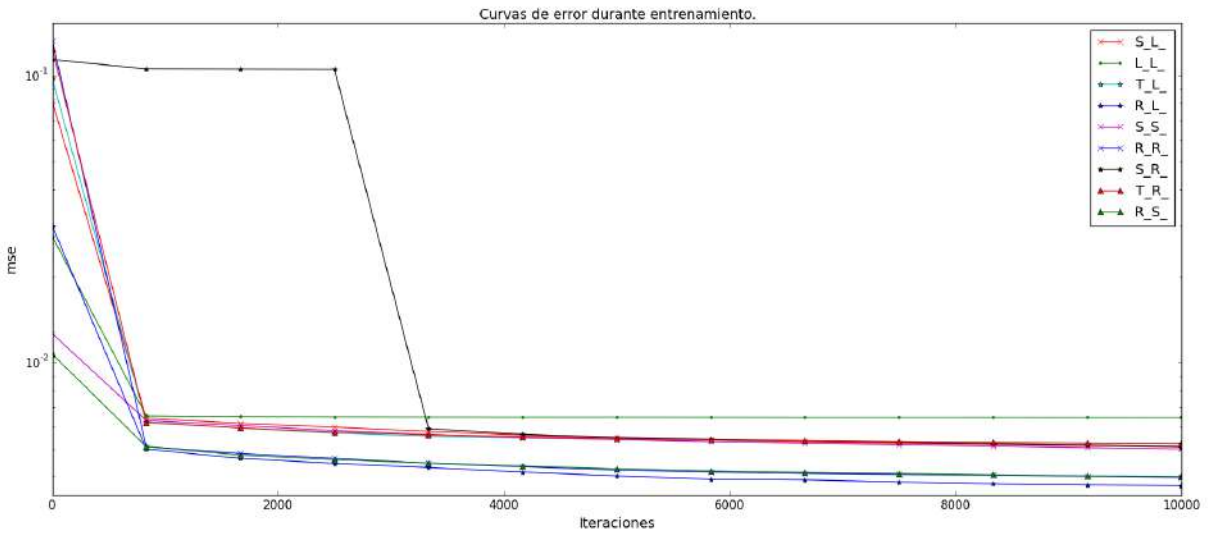


Figura 4.6: Valores del error durante el entrenamiento para diferentes funciones de activación en las capas de la red MLP.

Como se muestra en la figura 4.6, después de las 10000 iteraciones todos los modelos parecen aproximar su valor de convergencia. De las curvas mostradas de entrenamiento, se escoge el modelo **R_L_**, ya que este converge a un valor menor de error. Este modelo presenta un valor final de error cuadrático medio de entrenamiento de 0.0037.

4.2.6 Cantidad de capas y de neuronas.

Las redes neuronales tipo MLP están conformadas por una única capa de entrada y una única capa de salida, pero la cantidad de capas ocultas puede variar según lo requiera el problema. Como se menciona en [53], para funciones continuas, utilizar una única capa oculta con la cantidad óptima de neuronas es suficiente para modelar muchos problemas prácticos.

La cantidad de neuronas en las capas ocultas afecta en gran medida el desempeño de la red neuronal. Como se menciona en [53], si se seleccionan pocas neuronas, la red podría no ser capaz de aprender adecuadamente y presentar un mal desempeño cuando se le presenten datos nuevos. Por otra parte, seleccionar demasiadas neuronas puede hacer que la superficie aprendida por la red desarrolle falsos atributos entre los patrones de entrenamiento.

En esta etapa se generan modelos de una sola capa y se entrena con diferentes cantidades de neuronas para encontrar la cantidad óptima. Para estos experimentos se utiliza una red MLP con única capa oculta. Se utiliza la función de activación lineal rectificada para la capa oculta y la función lineal para la capa de salida, y se entrena durante 10000 iteraciones. Para realizar el entrenamiento se utiliza el 40 % (2330 pares) del total de datos disponibles. De estos datos de entrenamiento se utiliza el 80 % para entrenar la red y el 20 % para probar el modelo entrenado. Se quiere escoger una configuración que

logre minimizar el error pero que además dé buenos resultados para predecir sobre valores nuevos.

Para escoger la configuración óptima se plantea una función de costo, como la que utiliza en [54]. Esta función de costo planteada para esta tesis, se modifica respecto a la utilizada en [54], Como se muestra en la ecuación 4.5 se agrega un término que toma en cuenta el error del modelo para predecir nuevos valores.

$$fc = 0.5 * \frac{mse_{tr}}{MAX(mse_{tr})} + 0.3 * \frac{mse_{test}}{MAX(mse_{test})} + 0.2 * \frac{N}{MAX(N)} \tag{4.5}$$

Donde mse_{tr} es el valor final del error durante el entrenamiento, mse_{test} es el valor de error obtenido al evaluar el modelo sobre los datos de prueba y N es la cantidad de neuronas. Se entrenaron 19 modelos variando la cantidad de neuronas desde 5 hasta 100 neuronas. En la figura 4.7 se muestran los valores de la función de costo fc obtenido para cada cantidad de neuronas.

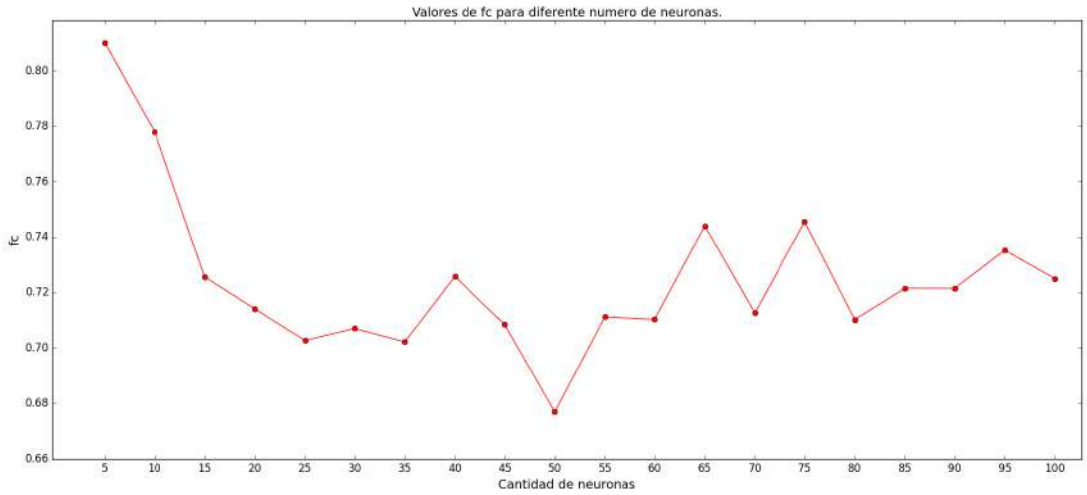


Figura 4.7: Valores obtenidos de la función costo para diferentes números de neuronas en la capa oculta de la red MLP.

Para esta función de costo se le da un peso de 50% al error de entrenamiento, 30% al error de prueba y 20% a la cantidad de neuronas. Los valores de mse_{tr} y N permiten escoger una red con un valor bajo de error de entrenamiento, pero que además se mantenga el numero de neuronas bajo. Se agrega el parámetro mse_{test} para escoger un modelo que no sea valido solo para los datos de entrenamiento,sino, que se pueda generalizar a nuevos datos.

Según las características evaluadas con la función de costo, la configuración más adecuada es la red con 50 neuronas en la capa oculta. Para esta red neuronal se obtuvo un error de entrenamiento (mse_{tr}) de 0.0044 y un error al evaluar sobre los datos de prueba (mse_{test}) de 0.0055.

4.2.7 Selección de parámetros para el entrenamiento.

Una vez definidos los parámetros más importantes de la red MLP, es conveniente determinar las configuraciones más adecuadas de entrenamiento. Para esto se debe seleccionar el método de optimización de los pesos de la red, así como la cantidad de iteraciones para el entrenamiento. Para determinar estos parámetros se realizan entrenamientos con diferentes métodos de optimización y se analizan las curvas de error.

Para estos experimentos se utiliza la red MLP con una capa oculta de 50 neuronas, con función de activación lineal rectificada en la capa oculta y la función lineal en la capa de salida. Se entrenó durante 10000 ciclos utilizando todos los datos disponibles (5826 pares). Del total se utiliza 80 % para entrenamiento y 20 % para evaluar el desempeño de la red ante datos nuevos.

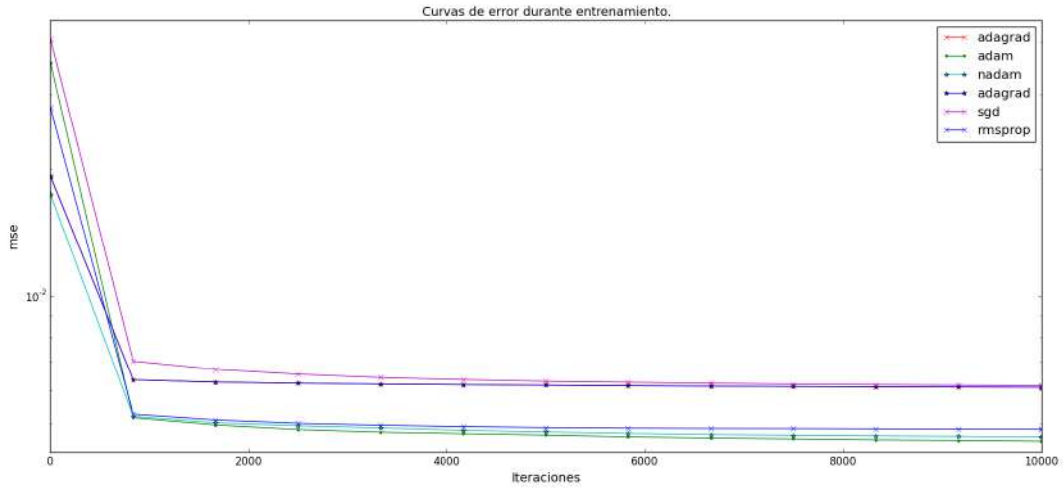


Figura 4.8: Valores de error durante el entrenamiento para diferentes métodos de optimización para la red ML.

Para los entrenamientos se utilizaron como métodos de optimización, gradiente adaptativo (**adagrad**), Adam (**adam**) [60] y [61], Nadam (**nadam**) [62], gradiente descendiente estocástico (**sgd**) y el método RMSprop (**sgd**). La descripción de cómo implementar estos métodos en la biblioteca Keras en Python se encuentra en [37].

En la figura 4.8 se muestran los resultados obtenidos con los diferentes métodos durante el entrenamiento del modelo. En donde se puede apreciar que el método Adam presenta una convergencia más rápida y además es el método con el cual se obtiene un valor de error menor de entrenamiento.

También se comprueba el desempeño de cada modelo ante datos nuevos, datos de prueba. Los modelos se prueban ante estos datos también durante el entrenamiento, y se muestran los resultados en la figura 4.9. En este caso los resultados coinciden con la gráfica anterior, y en este caso el método que converge más rápido y además ofrece un valor menor de error es el método Adam.

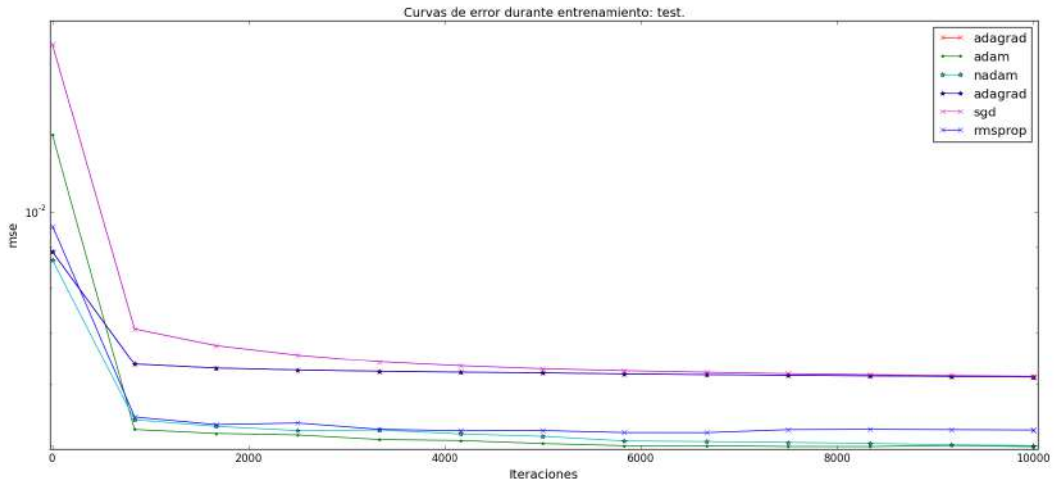


Figura 4.9: Valores de error durante el entrenamiento con diferentes métodos de optimización para la red MLP, con los datos de prueba.

Condición de parada del entrenamiento.

Para estimar cual es el numero adecuado de iteraciones para entrenar el modelo, se analizan las curvas de error durante el entrenamiento. En este caso se pone especial atención a evaluar el error ante los datos de validación. Tal como se muestra en la figura 4.10, durante el entrenamiento del modelo, después de una cierta cantidad de datos, el error ante los datos de validación empieza a aumentar. Este comportamiento se conoce como sobre-ajuste, en donde el modelo se adapta a los datos de entrenamiento, pero deja de ser válido ante nuevos datos.

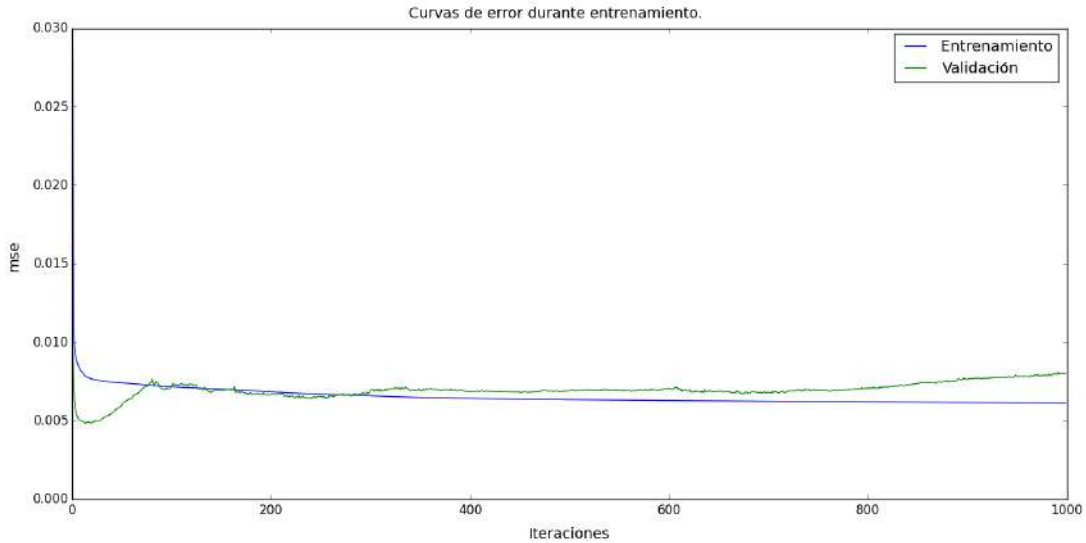


Figura 4.10: Valores de error durante el entrenamiento del modelo para pruebas de validación

Para evitar este comportamiento de sobre-ajuste, se define la condición de parada en base al comportamiento de la curva de error de los datos de validación. Este método

consiste en monitorear la predicción del modelo ante los datos de validación durante el entrenamiento. La condición de parada se define como la iteración en la cual el error de entrenamiento en los datos de validación, deje de disminuir y empieza a aumentar su valor.

En la implementación se da un margen de tres iteraciones, lo que quiere decir que se espera hasta tres iteraciones en donde el modelo aumente este error y luego el entrenamiento se detiene. Con este método se garantiza que el modelo sea válido para los datos nuevos (datos de validación), y se evita el sobre-ajuste del modelo ante los datos de entrenamiento.

4.3 Experimentos y toma de datos.

En esta sección se explican los experimentos que se plantean para la toma de los datos que se usan en la etapa de entrenamiento de los sistemas de aprendizaje automático. Estos experimentos consisten en utilizar el sistema de control diseñado por Federico Ruiz en [1], para manipular una caja. Durante la etapa de manipulación el brazo desliza la caja sobre una superficie desde el punto inicial hasta llevar la caja al la pose final. En la figura 4.11 se muestra la configuración en que se coloca la caja, y como el robot aplica la fuerza para deslizarla. Durante los experimentos, el brazo se encarga de deslizar la caja del punto inicial hasta el punto final, y se toman los datos de velocidad calculados por el sistema de predicción y los datos de velocidad que se miden con la cámara.

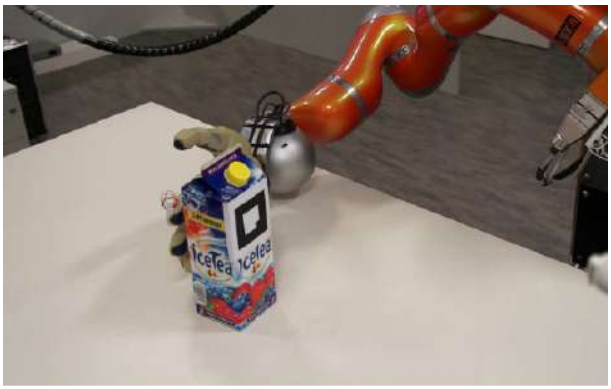


Figura 4.11: Configuración de brazo y mano con la caja para realizar experimentos [1].

En la sección 2.2 se explican los efectos principales que afectan la precisión del sistema de predicción planteado por Federico Ruiz. Basados en estos efectos se plantean tres experimentos para evaluar el desempeño del algoritmo al enfrentar diferentes tipos de cajas. Se busca presentar al robot cajas que se comporten diferente a lo que el modelo predice. Y utilizando el modelo de aprendizaje de error de la función f_{error} de la ecuación 4.4, se pretende modelar estas diferencias entre lo que el modelo planteado por Federico Ruiz predice, y lo que realmente ocurre con la pose de la caja. A continuación se presentan estos tres experimentos que se plantean, para evaluar 3 casos diferentes.

EXP1: Diferencias entre Elipsoide y Limit Surface

Para el primer experimento que se plantea, se utiliza una caja a la cual no se le realiza ninguna alteración. La cual debería tener un comportamiento similar a lo que predice el modelo. Pero como se menciona en la sección 2.1.1, el modelo incorpora una simplificación matemática en cálculo del Limit Surface, y esta simplificación matemática incorpora un error en el cálculo de la velocidad de la caja. Lo que indica que para todas las cajas a utilizar existe un error en la predicción de su velocidad.

Con este experimento se trata de que el algoritmo logre compensar el error del modelo dada la simplificación matemática en el Limit Suface. Aprender este error es importante ya que está presente en todas caja. La hipótesis es que en este experimento las fuentes de error provienen principalmente de esta simplificación matemática.

EXP2: Distribución de peso irregular

En este segundo experimento se realiza una modificación a la caja. Como se menciona en la sección 2.2, en el modelo de la caja se asume que el centro de masa de la caja coincide con el centro geométrico de la misma. Por lo que el modelo no contempla que el caso del centro de masa se encuentre en otro lugar. Dada esta diferencia, el modelo de la caja va a predecir un comportamiento diferente de lo que ocurre realmente.

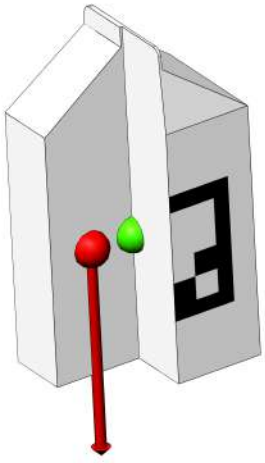


Figura 4.12: Caja utilizada en la sesión de experimentos EXP2.

Como se muestra en la figura 4.12, para este experimento se agrega una masa a una de las caras de la caja, para forzar a que el centro de masa se desplace del centro geométrico de la caja. Realizando los experimentos con esta caja, se quiere evaluar el comportamiento del modelo planteado por Federico Ruiz ante esta situación, y posteriormente evaluar el desempeño del algoritmo en aprender el error provocado por este cambio en el centro de masa de la caja.

EXP3: Cambio de fricción de las superficie

En este tercer experimento también se realiza una modificación a la caja. Basado en los problemas del modelo descritos en la sección 2.2, se tiene que en el modelo de caja que se plantea, se asume que el coeficiente de fricción entre la caja y la superficie sobre la cual se desliza, es uniforme. Este modelo asume que el coeficiente de fricción en toda la base de la caja es uniforme. Se sabe que por razones de fabricación y geometría de la base, la distribución del coeficiente de fricción sobre toda la superficie no es completamente uniforme.

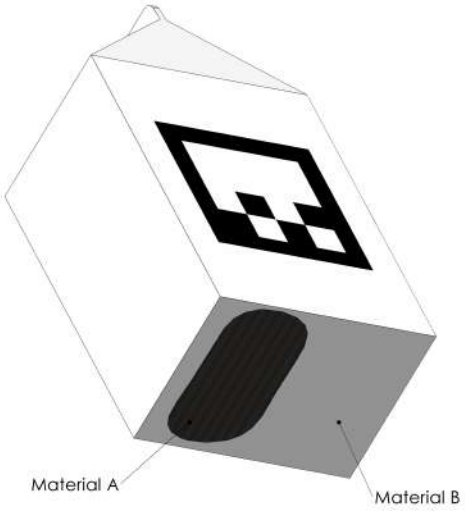


Figura 4.13: Caja utilizada en la sesión de experimentos EXP3.

Además de que originalmente la superficie de la caja no tiene una distribución de fricción uniforme, se decide forzar a que esta sea no uniforme. Como se muestra en la figura 4.13, se adhiere otro tipo de material en algunas secciones de la base de la caja, para generar diferentes coeficientes de fricción. Al utilizar esta caja para realizar experimentos, se quiere evaluar el comportamiento del modelo planteado por Federico Ruiz, ante esta situación de fricción no homogénea en la base, y posteriormente evaluar el desempeño del algoritmo en aprender el error provocado por este cambio en la fricción de la caja.

4.4 Aspectos económicos.

En esta sección se presentarán los costos asociados a la realización de este proyecto. Para esto se toma en cuenta las horas invertidas de trabajo, costo de la pasantía en Canadá y además el hardware y software utilizado propiamente en el desarrollo del sistema de compensación planteado. Se excluyen los costos de hardware propio del robot-humanoide, ya que este pertenece al Arcos-lab y no se adquirió propiamente para este proyecto.

En cuestión de hardware, para el proyecto se adquirió únicamente la cámara utilizada en

el sistema de visión. Se utilizó una cámara marca havit modelo HV-N606, la cual cuenta con conexión USB para computadora. Esta cámara es compatible con el sistema utilizado y además cumple con los requisitos mínimos para operar el sistema de visión. La cámara tuvo un costo de 10 500 colones.

Como parte del proyecto se utilizaron viarios software en las diferentes etapas. Como es costumbre en el Arcos-lab, todo el software implementado se utiliza bajo licencia libre. En este proyecto no fue necesario comprar ninguna licencia de software ya que todos los programas utilizados son de software libre. En la tabla 4.1 se listan los programas principales utilizados así como el tipo de licencia libre que utiliza.

Tabla 4.1: Software utilizado

Nombre	Licencia
Python	GPL-compatible
Keras	The MIT License (MIT)
ArToolKit	GNU (LGPLv3)
ROS	Varias (todas libres)

El desarrollo del proyecto requiere también la inversión de horas laborales, las cuales se contabilizan entre los costos finales del proyecto. Este proyecto se realiza dentro de un ambiente de investigación en la universidad, y no se contó con ningún tipo de subsidio por parte de los laboratorio en relación a las horas trabajadas en el desarrollo del proyecto. Por lo que para cuestión de cálculo del valor final del proyecto, se toma como valor de las horas invertidas, el rubro de salario mínimo para un licenciado universitario, el cual corresponde a 629 395 colones mensuales según el Ministerio de Trabajo y Seguridad Social [63] para el periodo 2017. Lo que representa un total de 2518000 colones durante los 4 meses de trabajo.

Como parte del proyecto se realizó una pasantía en el Wise-Lab [9], en la Universidad de Regina. Durante la duración de esta pasantía se generaron varios gastos para cubrir los costos del viaje, alimentación, hospedaje y derechos de estadía en la Universidad de Regina. Estos gastos también se toman en cuenta en el costo total de la realización del proyecto. En la tabla 4.2 se presenta un resumen de los gastos durante la pasantía.

Tabla 4.2: Gastos durante la pasantía en la Universidad de Regina

Rubro	Monto (10^3 colones)
Boletos de avión	400
Alimentación y hospedaje	1400
Cargos en la Universidad de Regina	350
Otros	270
Total	2420

Finalmente en la tabla 4.3 se listan los diferentes gastos que implicaron la realización de este proyecto de graduación. Cabe resaltar que los principales gastos del proyecto los

representan la pasantía en la Universidad de Regina y el valor representativo del pago de horas de trabajo. La inversión realizada en adquisición de hardware y software es mínima en comparación de los otros rubros.

Tabla 4.3: Costos totales

Rubro	Monto (10^3 colones)
Pasantía en el Wise-lab	2420
Remuneración por horas trabajadas	2518
Hardware (cámara)	10.5
Total	4948.5

Dadas las circunstancias actuales del proyecto del robot humanoide del Arcos-Lab, no es posible cuantificar los beneficios económicos que implica el sistema de compensación propuesto en este proyecto. El robot humanoide se encuentra en etapas de ensamble y paralelamente se trabaja en investigación y desarrollo de nuevas partes. No existen planes a corto plazo de comercializar ningún producto donde se involucre el sistema de compensación de error.

Como actualmente no se contempla la comercialización de ningún producto, no es posible cuantificar el impacto económico del desarrollo de este proyecto, en un posible producto. En este punto los beneficios de desarrollar el sistema de compensación de error, se cuantifican con parámetros técnicos que mejoran la destreza del robot al manipular objetos. Dentro del marco del proyecto del robot humanoide, la importancia del sistema compensador de error, es precisamente mejorar el desempeño del sistema de predicción de pose para el control, y no se buscaba un beneficio económico directo.

Capítulo 5

Resultados y análisis

En este capítulo se exponen los resultados obtenidos en las pruebas sobre los algoritmos de aprendizaje automático. Se presenta una comparación entre dos variantes del algoritmos de aprendizaje prepuestos, y se emite un criterio de selección basados en el desempeño. En capítulo se presentan los resultados obtenidos con datos experimentales acompañados de un análisis.

5.1 Los datos experimentales

Los datos experimentales utilizados para realizar las pruebas, provienen de experimentos que fueron realizados por Federico Ruiz durante el desarrollo de su tesis [1]. Estos datos provienen del robot humanoide TUM-Rosie [14] mostrado en la figura 5.1 el cual es parte del Grupo de Sistemas Autónomos Inteligentes (IAS-TUM) de la Universidad Técnica de Munich y se tomaron en diferentes experimentos . Se utilizan estos datos ya que por diferentes razones técnicas no fue posible concluir a tiempo la puesta en marcha del brazo del robot humanoide del Arcos Lab. Estos inconvenientes en el funcionamiento del robot del Arcos-Lab hicieron que no fuera posible obtener los datos experimentales a tiempo, por lo que se opta por utilizar datos ya existen pero con otro robot.

Estos datos fueron tomados durante diferentes experimentos que se realizaron con el sistema de control de Federico Ruiz [1] con el robot Tum-Rosie. Los datos son válidos para probar los algoritmos de aprendizaje automático ya que se recolectaron de la misma forma que se planean recolectar con el humanoide del Arcos-Lab. Para capturar los datos, el robot desliza una caja desde una pose inicial hasta un pose final, y almacena las variables utilizadas por el sistema de predicción en archivos.

Los datos disponibles corresponden a los que se obtendrían con el experimento planteado como **EXP1: Diferencia entre el Elipsoide y el Limit Suface**, descrito en la sección 4.3. Donde se experimenta con una caja sin alteraciones, y se busca compensar el error generado por la simplificación del elipsoide utilizada como Limit Surface.

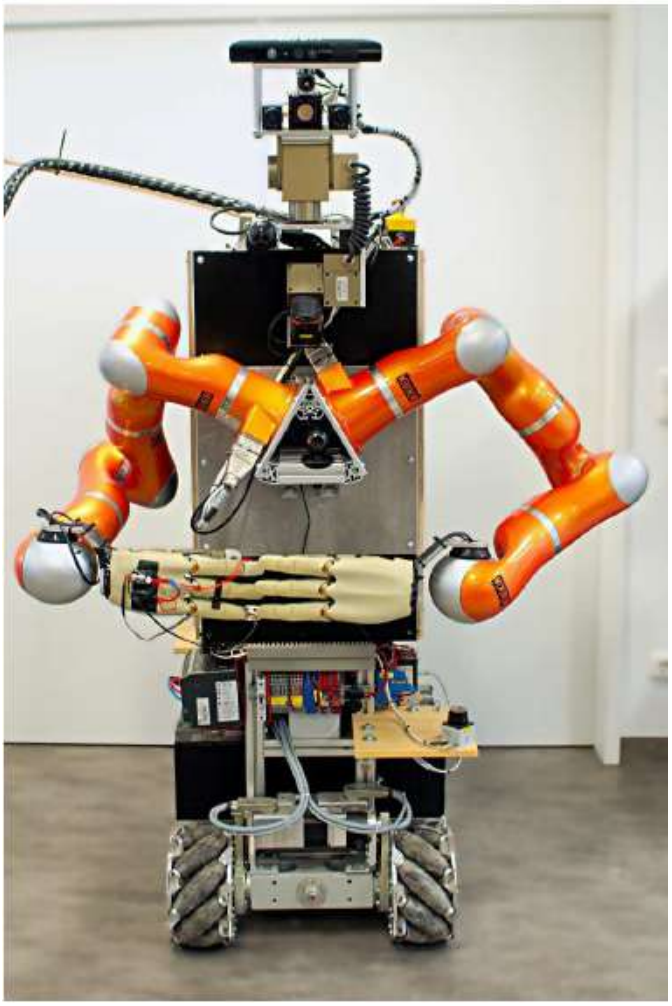
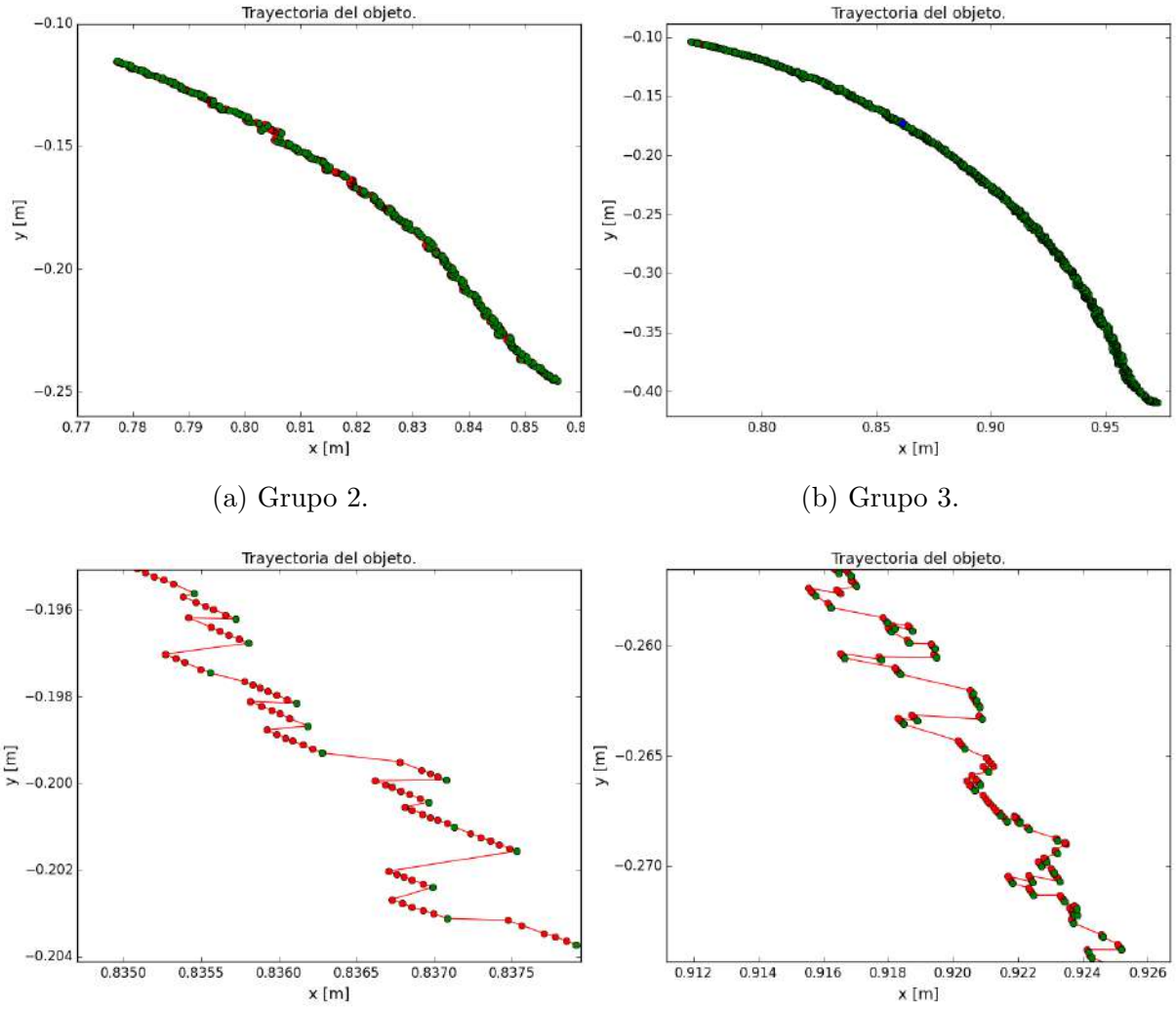


Figura 5.1: Robot TUM-Rosie [1].

Los datos disponibles para entrenamiento y validación, corresponden a 6 grupos de datos, donde cada grupo representa un experimento realizado. Cada experimento se realiza utilizando las mismas condiciones, utilizando la misma caja en todos ellos. Donde el grupo 1 tiene 731 muestras de datos, el grupo 2 con 614 datos, el grupo 3 con 1183 datos, el grupo 4 con 1076 datos, el grupo 5 con 1012 datos y el grupo 6 con 1020 datos

Estos 6 grupos de datos se utilizan de la siguiente manera: para lograr una validación redundante sobre los 6 grupos de datos, se realizan varios entrenamientos. Para evaluar la validez sobre los datos disponibles, se entrena el algoritmo con 5 grupos de datos y se realiza la validación con el sexto, por lo que rotan los datos de entrenamiento de manera que se valide con los 6 grupos de datos. Se realiza un total de 6 entrenamientos, en donde el grupo de datos utilizado para validar no se utiliza durante el entrenamiento. De los datos utilizados para entrenar, se utiliza 80 % propiamente para entrenamiento, y el otro 20 % se utiliza para prueba.

En la figura 5.2 se ilustra la trayectoria que realiza la caja en dos grupos de datos diferentes. Cada grupo muestra las coordenadas de la caja mientras esta es desplazada sobre



(a) Grupo 2.

(b) Grupo 3.

(c) Acercamiento en el grupo 2.

(d) Acercamiento en el grupo 3.

Figura 5.2: Desplazamiento realizado por la caja en dos experimentos diferentes realizados por el humanoide TUM-Rosie.

la mesa. Los puntos verdes representan los datos de posición que se estiman desde la cámara, mientras que los puntos rojos son valores de posición estimados por el sistema de predicción de pose. Como se ve claramente, los datos de posición calculados con el sistema de predicción tienen una tendencia diferente a la posición estimada desde la cámara.

Como se menciona anteriormente, se busca compensar el error presente en la velocidad calculada por el sistema de predicción, por lo que el sistema de aprendizaje automático se encarga de aprender este error en los cálculos de velocidad. Este error se calcula entre la velocidad estimada desde la cámara y la velocidad estimada por el sistema de predicción.

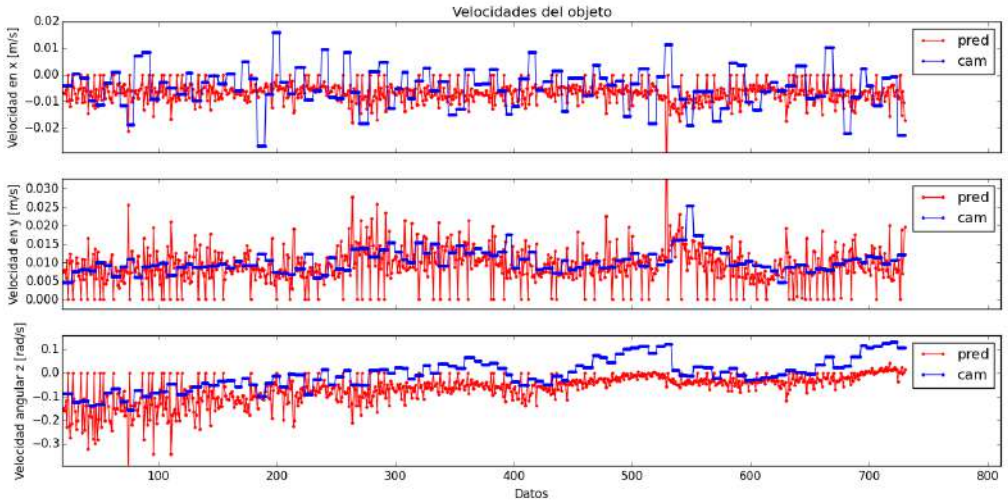


Figura 5.3: Velocidades de la caja.

En la figura 5.3 se tiene el comportamiento de las tres componentes de velocidad que nos interesan sobre la caja. Las velocidades lineales en x y y, así como la velocidad angular en z. Dado que el sistema de predicción genera datos mas rápido que la cámara, se compara la velocidad del sistema de predicción con la velocidad promedio calculada desde la cámara. Con el sistema compensador de error, se busca que la señal roja (sistema de predicción) se comporte de manera similar a la señal azul (cámara).

5.2 Validación cruzada.

Para conocer la validez del modelo de error planteado en esta tesis se somete a una prueba de validación cruzada utilizando los seis grupos de datos disponibles. Este ejercicio consiste en entrenar el modelo con los datos de 5 grupos diferentes y evaluar como se desempeña este modelo ante el grupo de datos ausente en el entrenamiento. La validación cruzada consiste en repetir este método para todos los grupos de datos disponibles. Con este método de validación se garantiza que el modelo se generalice ante datos nuevos.

En la figura 5.4 se presenta una curva típica de error durante el entrenamiento. La curva de error de validación alcanza un valor mínimo rápidamente, y a las pocas iteraciones

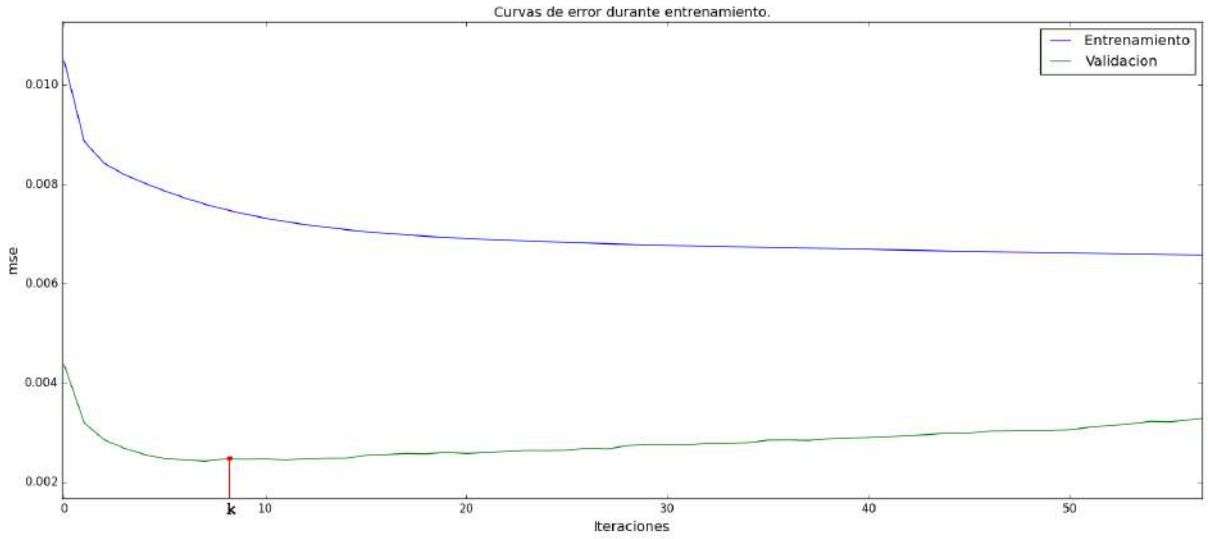


Figura 5.4: Curva de error de y criterio de parada de entrenamiento.

después, el error comienza a incrementar. Como se explicó anteriormente, el criterio de parada consiste en detectar este aumento en el error y detener el entrenamiento. En este ejemplo el entrenamiento se detuvo en la iteración **k**, ya que después de este valor, el error de validación incrementa.

Topología de red neuronal

Para las pruebas de validación se utiliza una red de tipo MLP (red de perceptrón multicapa), con una capa oculta de 50 neuronas, con función de activación tipo lineal rectificada (ReLU). En la capa se salida se tienen tres neuronas, que corresponden a las tres componentes del vector de error en la velocidad, en esta capa se utiliza la función lineal. Como método de optimización se utiliza **Adam** [60][61] y la función de pérdida es **Error absoluto medio**. La red neuronal se implementa en python utilizando la biblioteca Keras [37]. Esta biblioteca permite mucha flexibilidad en la configuración de redes neuronales.

Para evaluar el sistema de compensación de error, se quiere cuantificar el grado de mejora en el sistema de predicción. Esta cuantificación se hace como se muestra en la ecuación 5.1 en donde se calcula el error del sistema de predicción respecto a la cámara, y este se compara con el error que existe en el sistema de predicción luego de sumarle el sistema de compensación de error. Para calcular este error se utiliza la fórmula de raíz cuadrada del error cuadrático medio. Con esta ecuación se cuantifica qué porcentaje de mejora existe al implementar el compensador.

$$\%Error = \frac{Error_{original} - Error_{corregido}}{Error_{original}} * 100 \tag{5.1}$$

Donde $Error_{original}$ es el error del sistema de predicción respecto a la cámara. Y el $Error_{corregido}$ es el error calculado cuando se suma la corrección del sistema de aprendizaje

automático al sistema de predicción y luego se compara contra el valor de la cámara. Para todos los entrenamientos realizados se calculó el $\%Error$ de los datos de prueba y de los datos de validación.

Tabla 5.1: Porcentaje de error de validación para los diferentes grupos de datos

Modelo	$\%vel.x$	$\%vel.y$	$\%vel.angular$
Grupo1	16.16	44.80	62.22
Grupo2	65.83	41.21	62.28
Grupo3	5.46	24.29	46.03
Grupo4	6.36	26.81	28.28
Grupo5	7.52	24.60	43.95
Grupo6	29.37	51.52	29.22

5.2.1 Pruebas del compensador.

En esta sección se ilustran los resultados obtenidos con el modelo entrenado. Se presentan diferentes gráficos en donde se utiliza la red neuronal para disminuir el error en uno de los grupos de datos de validación. Además, se muestran algunos gráficos de posición donde se muestra la influencia del compensador en estos cálculos. Para todas las pruebas ilustradas en esta sección se utiliza el grupo 1 de datos.

Velocidad.

En el caso de la velocidad se presentan dos gráficos por cada componente de velocidad. Como se muestra en las imágenes 5.5, 5.6, y 5.7, el gráfico superior presenta tres señales de velocidad. Para el grupo 1 de datos se obtuvo una reducción de 16.16 % del error de velocidad en x. Este gráfico muestra en azul la velocidad medida desde la cámara, en rojo la velocidad estimada por el sistema de predicción, y en verde la velocidad del sistema de predicción al sumarle la velocidad calculada por el compensador. De este gráfico se quiere que la señal verde tenga un comportamiento muy similar a la señal azul.

En el gráfico de la parte inferior de la imagen, se tienen las señales de error presentes en cada componente de velocidad. Este gráfico muestra el error real del sistema de predicción en azul, y en rojo se tiene la señal de error estimada con el compensador. El éxito del compensador se puede ilustrar en esta gráfica como la capacidad de la señal roja de seguir a la señal azul.

La figura 5.5 representa las velocidades en x obtenidas sobre el grupo 1 de datos. Del gráfico superior destaca como el comportamiento de la señal de velocidad real (azul) y la velocidad del sistema de predicción (rojo), se comportan muy diferente. Estas diferencias son más notorias en los valores pico de magnitud de velocidad real, donde la velocidad estimada desde el sistema de predicción no tiene ese comportamiento.

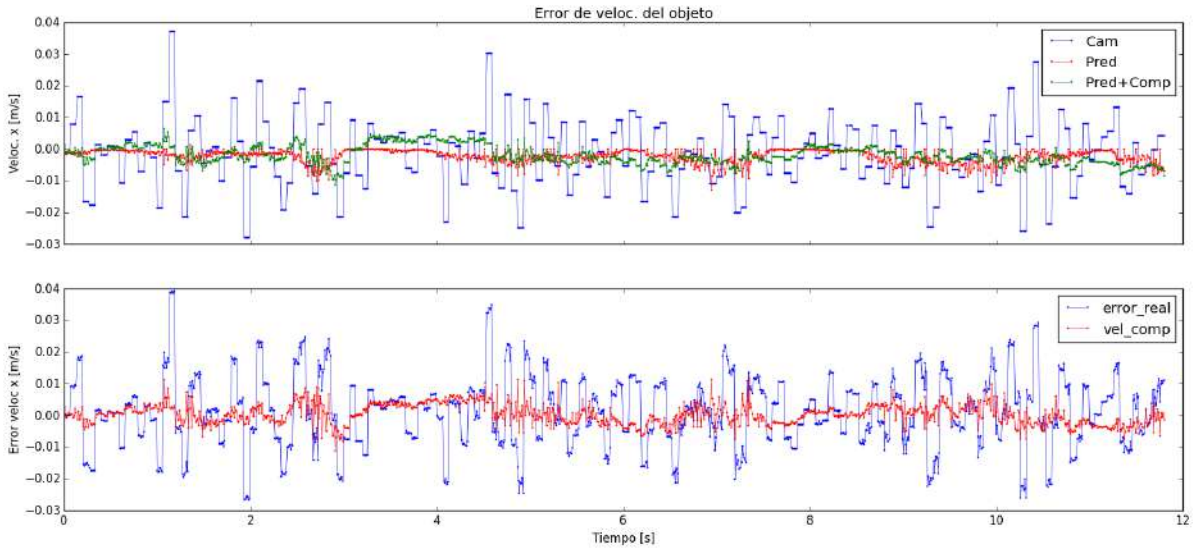


Figura 5.5: Señales de velocidades en x.

Como es de esperarse, dados los porcentajes bajos de disminución de error en la velocidad en x, al incluir el compensador no se logra una notable mejora en la señal del sistema de predicción con compensador. Se evidencia que la señal verde no sigue el comportamiento de la señal azul.

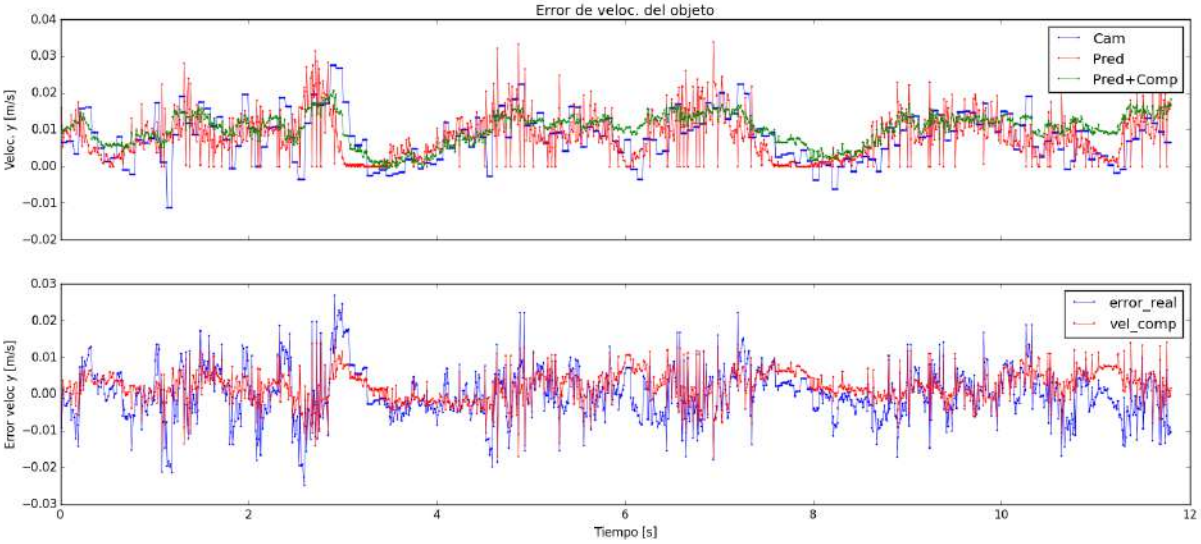


Figura 5.6: Señales de velocidades en y.

También es importante analizar el comportamiento de la gráfica inferior en la figura 5.5 donde se muestra el error de la velocidad del sistema de predicción. En este gráfico se muestra que la señal de error obtenida del compensador no logra seguir en muchos trayectos al error real. Esto se puede ver principalmente en los valores máximos de error donde el compensador no sigue ese comportamiento.

En el caso de la figura 5.6 de velocidad en y se ilustran mejores resultados que en el caso anterior. Para el grupo 1 de datos se obtuvo una reducción del 44.80 % del error

de velocidad en y. En el gráfico superior (figura 5.6) se tiene que la velocidad calculada por el sistema de predicción (SPP) se comporta diferente a la velocidad real. Existen varios picos en el sistema de predicción que no existen en la señal real, pero es importante notar que excluyendo esos picos, el sistema de predicción logra una tendencia similar a la velocidad real en algunos tramos.

En este mismo gráfico se tiene que la señal del sistema de predicción con compensador (verde) no presenta estos picos que presentaba el compensador (rojo) inicialmente. También se puede ver cómo esta señal verde del sistema de predicción compensado logra un comportamiento similar en gran parte del trayecto a la velocidad real.

En esta figura 5.6 se tiene en el gráfico inferior el error de velocidad en y. La gráfica muestra cómo la señal del compensador en azul, tiene un comportamiento similar a la gráfica del error real.

Finalmente se tiene la gráfica de velocidad angular en la figura 5.7. El compensador logra un mayor porcentaje de disminución de error en la velocidad angular. Para el grupo 1 de datos se obtuvo una reducción del 62.22 % del error de velocidad angular. En el gráfico superior de velocidad se aprecia cómo la señal del sistema de predicción con compensador (verde) logra una tendencia muy similar a la velocidad real (azul). El compensador logra disminuir el "off-set" que existe entre la señal de la cámara y el compensador.

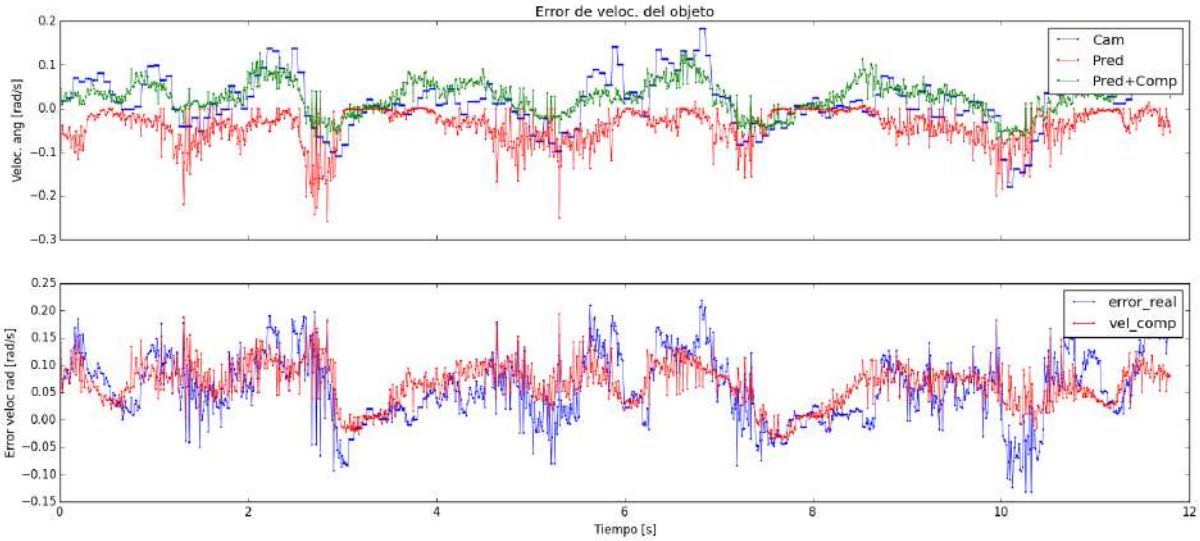


Figura 5.7: Señales de velocidades angulares.

En la gráfica inferior donde se tiene el error del sistema compensador, la señal de error estimada con compensador tiene un comportamiento muy similar a la gráfica de error real. El sistema compensador logra predecir los valores de error satisfactoriamente en muchos de los trayectos de la caja.

Existen algunos factores importantes que pueden afectar los resultados obtenidos con la solución propuesta. Hay que tomar en cuenta que para estos experimentos se confía en la señal de la cámara como valores reales de la posición y velocidad de la caja. La medición

de la cámara se puede ver afectada por diversos factores durante los experimentos.

Uno de estos factores son las oscilaciones indeseadas de la inclinación de la caja. Durante la trayectoria, la caja presenta pequeñas oscilaciones sobre su base. Esta oscilaciones pueden ser interpretados por la cámara como desplazamientos sobre la superficie. También, la pequeñas deformaciones del marcador pueden afectar la lectura de la cámara. Estas deformaciones pueden darse por deformaciones de las paredes de la caja por las fuerzas aplicadas.

Es importante recordar que los cálculos de pose desde la cámara, son mas lentos que los datos del sistema de predicción (SPP). Esto hace que se tengan mas datos del sistema de predicción que de la cámara. Para resolver esto, para los datos de velocidad se utiliza el promedio para los valores intermedios que faltan de la cámara, como se ilustra en la figura 5.8. Esto puede afectar los resultados ya que se están evaluando los modelos contra un valor promedio y no contra el valor real.

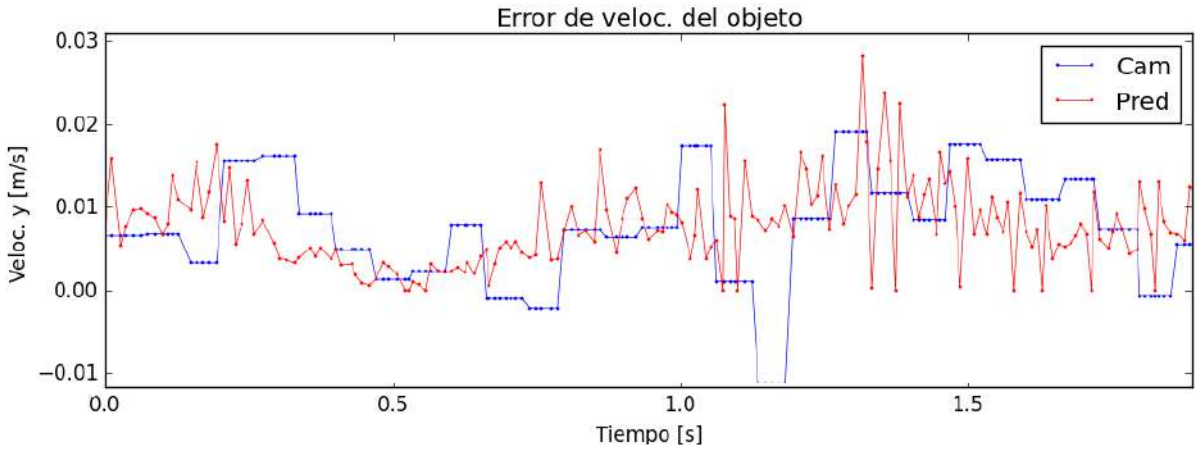


Figura 5.8: Señales de velocidad calculadas por el SPP (rojo) y por la cámara (azul).

Posición.

Para ilustrar los resultados obtenidos con el compensador, se generaron gráficas de cómo se ve reflejada la disminución del error de velocidad en la posición. Estas gráficas muestran con puntos azules las posiciones reales estimadas desde la cámara. En rojo se tiene la predicción de posición estimada por el sistema de predicción, y en verde se muestra la estimación de posición desde el sistema de predicción con compensador. Estas posiciones se calculan integrando la velocidad. El sistema hace predicciones a partir de un dato real de posición conocido (punto azul), los puntos verdes y rojos a la derecha de cada punto azul son predicciones. De la misma manera que con la velocidad también se calcula el porcentaje de error que se disminuye al utilizar el sistema compensador

En la figura 5.9 se muestran las posiciones en x durante la trayectoria realizada para los datos del grupo 1. Es este caso se puede ver como los resultados pueden variar a lo largo del trayecto. Para los datos de posición de x se logra disminuir el error en 5.41 %. Las

mejoras en el sistema de predicción (SPP) se notan cuando los puntos verdes tiene una tendencia similar a los azules. Estas mejoras se pueden ver la figura 5.9, entre el segundo 3 y el segundo 4.

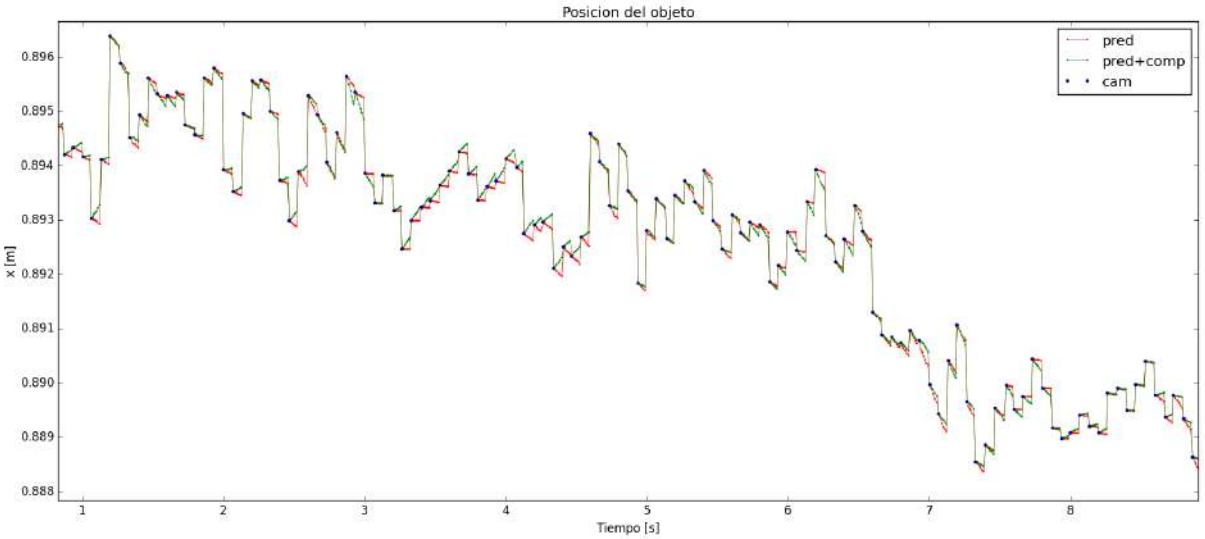


Figura 5.9: Señales de posición en x.

En la figura 5.10 se muestran las posiciones en y en la trayectoria realizada para los datos del grupo 1. El porcentaje de disminución de error de posición en este caso es de 24.05%. En varios tramos del recorrido se logra que los puntos verdes tengan la misma tendencia que los puntos azules. Cerca de los valores del segundo 3, se pueden ver estas mejoras.

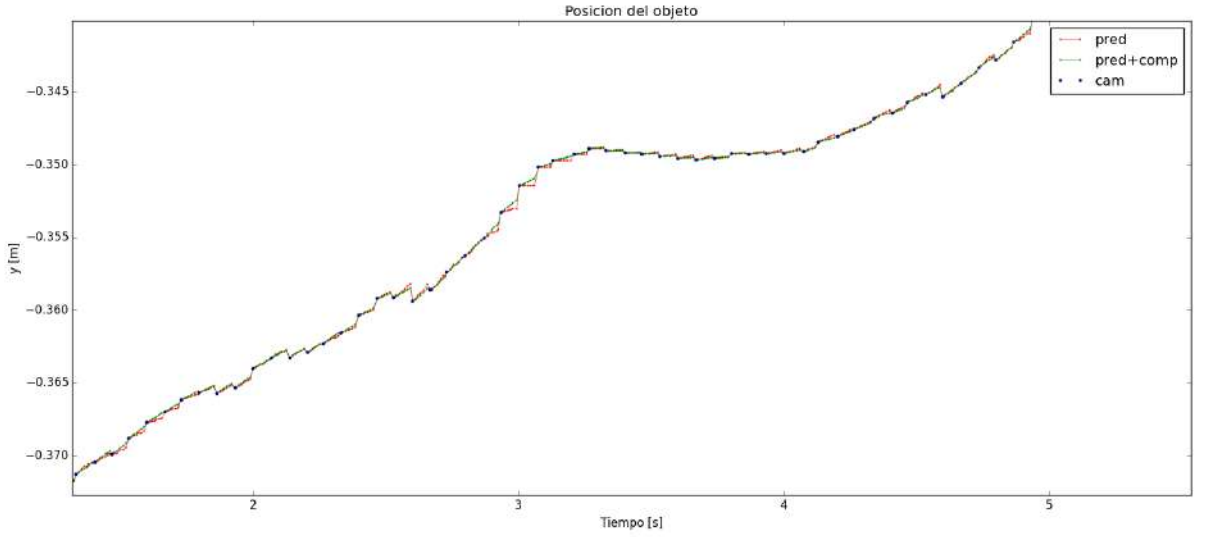


Figura 5.10: Señales de posición en y.

Finalmente en la figura 5.11 se muestran las posiciones angular durante la trayectoria realizada para los datos del grupo. Con el compensador de error se logra disminuir el error en 45.61%. Como se evidencia en la gráfica, la mejora en la estimación de pose es

clara a lo largo de los 11 segundos mostrados. Para este caso se ve cómo la estimación del ángulo por el sistema de predicción (puntos rojos) es muy pobre. El ángulo calculado por el sistema de predicción con compensador (puntos verdes) tiene un comportamiento muy similar al ángulo real de la caja (puntos azules) en la mayoría del trayecto.

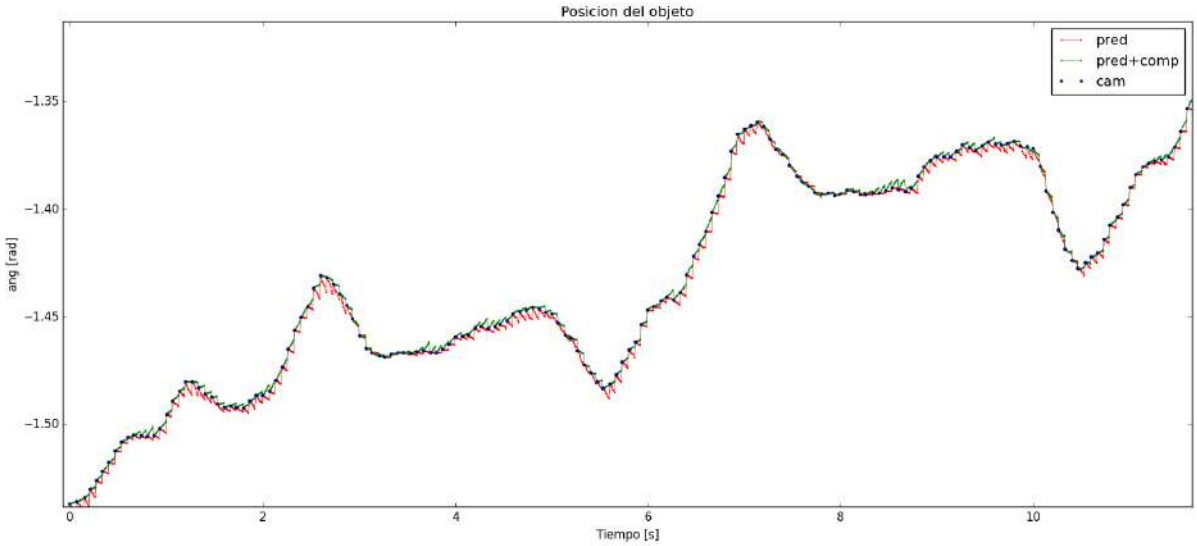


Figura 5.11: Señales de posición angular.

Capítulo 6

Conclusiones y Recomendaciones

- Se propone una lista de experimentos con el robot y una caja, donde se modifican condiciones experimentales, para probar la habilidad de un algoritmo de aprendizaje automático para disminuir el error del modelo de objeto, utilizado para predecir el comportamiento del movimiento de una caja sobre una mesa.

- Como parte de las configuraciones previas necesarias para poner en funcionamiento el sistema de control en el robot del Arcos-Lab. Se obtienen experimentalmente algunos parámetros físicos de la mano DLR Hit II necesarios para el control y además, se implementa un sistema de visión en ROS que permite estimar la pose del objeto.

- Se propone un modelo de error con redes neuronales para compensar las debilidades de un modelo matemático de un objeto, utilizado dentro de un sistema de predicción de pose de objeto, para un robot humanoide.

- Se demuestra que se puede aprender el error de la velocidad de un objeto calculada con un sistema de predicción de pose, utilizando redes neuronales.

- Se prueba mediante validación cruzada, que la red neuronal entrenada logra disminuir el error de la velocidad de una caja calculada por un sistema de predicción. En los 6 casos evaluados se logra disminuir en más de 5 % el error de la velocidad en x, en más de 24 % el error en la velocidad en y y en más de 28 % el error en la velocidad angular.

- Se logra disminuir el error en el valor pose de una caja calculado con el sistema de predicción de pose, utilizando una red neuronal. Se disminuye el error de pose indirectamente al utilizar la red neuronal para estimar y compensar el error en el cálculo de velocidad de una caja.

Recomendaciones

Para aumentar el porcentaje de disminución de error se propone experimentar con otros modelos de redes neuronales donde se agregue otras entradas que puedan afectar el valor de la magnitud del error. Además, se debe comprobar experimentalmente la sensibilidad de la red neuronal ante cada una de las variables de entrada utilizadas actualmente, con el fin de descartar entradas que no influyan en la salida de la red o sean variables redundantes. Para medir esta sensibilidad se puede utilizar el método que se menciona en [25] en donde se deshabilita cada entrada y se mide el efecto que esto tiene en el desempeño del sistema.

Un paso importante en la continuación de esta investigación es comprobar la capacidad de la red neuronal de aprender el error en casos donde el modelo de predicción presente dificultades. Algunos de estos casos difíciles se presentan en la sección 4.3, en donde se realizan modificaciones en el ambiente del experimento, para generar situaciones que el modelo de la caja no contempla.

Para el caso en donde se obtuvieron valores bajos de disminución de error, se pueden considerar como *outliers*. Para este problema se puede aumentar la cantidad de grupos de entrenamientos, para generalizar el modelo a más situaciones que puedan afectar los datos de entrenamiento.

Desde el punto de vista de técnica de aprendizaje automático utilizada, el problema de aproximación de la función de error queda abierto a implementarse con otra técnica u otra variante de red neuronal.

Bibliografía

[1] F. Ruiz, "Compact models of objects for skilled manipulation," Ph.D. dissertation, TUM, 2014.

[2] I. The MathWorks. Aprendizaje automático. [Online]. Available: https://es.mathworks.com/discovery/machine-learning.html

[3] S. G. Andreas C. Müller, *Introduction to Machine Learning with Python: A Guide for Data Scientists*, 1st ed. O'Reilly Media, 2016.

[4] S. R. N and P. C. Deka, "Support vector machine applications in the field of hydrology: A review," *Applied Soft Computing*, vol. 19, no. Supplement C, pp. 372 – 386, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614000611

[5] J. Bell, *Machine Learning: Hands-On for Developers and Technical Professionals*, 1st ed. Wiley, 2014.

[6] T. M. Mitchell, *Machine Learning*, 1st ed., ser. McGraw-Hill series in computer science. McGraw-Hill, 1997.

[7] (2013) Arcos-lab. arcos-lab wiki. [Online]. Available: https://wiki.arcoslab.eie.ucr.ac.cr/doku.php?id=star

[8] H. Liu, K. Wu, P. Meusel, N. Seitz, G. Hirzinger, J. Minghe, Y. Liu, S. Fan, T. Lan, and Z. Chen, "Multisensory five-finger dexterous hand: the dlr/hit hand ii," pp. 3692–3697, 09 2008.

[9] R.Mayorga. (2003) Wise-lab. wise, intelligent, systems and entities, lab. [Online]. Available: http://uregina.ca/~mayorgar/2W_WISE.html

[10] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit surface and moment function descriptions of planar sliding," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE, 1989, pp. 794–799.

[11] S. Goyal, "Planar sliding of a rigid body with dry friction: limit surfaces and dynamics of motion," Ph.D. dissertation, Cornell University Ithaca, New York, USA, 1989.

[12] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991.

[13] R.Howe and M.Cutkosky, "Practical force-motion models for sliding manipulation," *The International Journal of Robotics Research*, pp. 557–572, 1996.

[14] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic roommates making pancakes," pp. 529–536, 2011.

[15] W. Chatlatanagulchai and P. H. Meckl, "Motion control of two-link flexible-joint robot with actuator nonlinearities, using backstepping and neural networks," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2511–2516.

[16] B. Ulutas, E. Erdemir, and K. Kawamura, "Application of a hybrid controller with non-contact impedance to a humanoid robot," in *2008 International Workshop on Variable Structure Systems*, June 2008, pp. 378–383.

[17] O. Lakhal, A. Melingui, and R. Merzouki, "Hybrid approach for modeling and solving of kinematics of a compact bionic handling assistant manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 3, pp. 1326–1335, June 2016.

[18] A. I. Aviles, S. M. Alsaleh, J. K. Hahn, and A. Casals, "Towards retrieving force feedback in robotic-assisted surgery: A supervised neuro-recurrent-vision approach," *IEEE Transactions on Haptics*, vol. 10, no. 3, pp. 431–443, July 2017.

[19] A. I. Aviles, A. Marban, P. Sobrevilla, J. Fernandez, and A. Casals, "A recurrent neural network approach for 3d vision-based force estimation," in *2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Oct 2014, pp. 1–6.

[20] Y. H. Kim and F. L. Lewis, "Reinforcement adaptive learning neural-net-based friction compensation control for high speed and precision," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 1, pp. 118–126, Jan 2000.

[21] M. K. Ciliz and M. Tomizuka, "Modeling and compensation of frictional uncertainties in motion control: a neural network based approach," in *American Control Conference, Proceedings of the 1995*, vol. 5, Jun 1995, pp. 3269–3273 vol.5.

[22] K. Shojaei, "Neural adaptive output feedback formation control of type (m, s) wheeled mobile robots," *IET Control Theory Applications*, vol. 11, no. 4, pp. 504–515, 2017.

[23] P. Esmaili and H. Haron, "Adaptive synchronous artificial neural network based pi-type sliding mode control on two robot manipulators," in *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, April 2015, pp. 515–519.

[24] Y. Lu, J. K. Liu, and F. C. Sun, "Actuator nonlinearities compensation using rbf neural networks in robot control system," in *The Proceedings of the Multiconference on "Computational Engineering in Systems Applications"*, vol. 1, Oct 2006, pp. 231–238.

[25] M. Tanner, S. Saftescu, A. Bewley, and P. Newman, "Meshed up: Learnt error correction in 3d reconstructions," *arXiv preprint arXiv:1801.09128*, 2018.

[26] V. N. Vapnik, *Statistical Learning Theory*, ser. Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.

[27] ——, *The Nature of Statistical Learning Theory*, 2nd ed., ser. Statistics for Engineering and Information Science. Springer, 2000.

[28] J. Goddard Close, S. G. de los Cobos Silva, B. R. Pérez Salvador, and M. Gutiérrez Andrade, "Un algoritmo para el entrenamiento de máquinas de vector soporte para regresión," *Revista de Matemática: Teoría y Aplicaciones Vol. 7 Núm. 1-2 2009*, 2009.

[29] O. S. R. Foundation. (2000) Ros. [Online]. Available: http://www.ros.org/about-ros/

[30] G. J.Bohren and C.Crick. gscam. [Online]. Available: http://wiki.ros.org/gscam

[31] K. P.Mihelich and J.Leibs. Image proc. [Online]. Available: http://wiki.ros.org/image_proc?distro=kinetic

[32] B. I.Dryanovsk and G.Dumonteil. Augmented reality marker pose estimation using artoolkit. [Online]. Available: http://wiki.ros.org/ar_pose

[33] D. D.Hershberger and J.Faust. 3d visualization tool for ros. [Online]. Available: http://wiki.ros.org/rviz

[34] J.Bowman. Camera calibration. [Online]. Available: http://wiki.ros.org/camera_calibration?distro=kinetic

[35] P. G.Metta and L.Natale, "Yarp: Yet another robot platform," *Journal of Software Engineering for Robotics*, pp. 80–88, 2014.

[36] L.Sciavicco and B.Siciliano, *Modelling and Control of Robot Manipulators*. McGraw Hil, 1996.

[37] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[38] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 161–168.

[39] S. Ahmed, M. Khalid, and U. Akram, "A method for short-term wind speed time series forecasting using support vector machine regression model," in *2017 6th International Conference on Clean Electrical Power (ICCEP)*, June 2017, pp. 190–195.

[40] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[41] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[42] C. Fernández, I. Huerta, and A. Prati, "A comparative evaluation of regression learning algorithms for facial age estimation," in *Face and Facial Expression Recognition from Real World Videos*. Springer, 2015, pp. 133–144.

[43] M. Sanchez-Fernandez, M. de Prado-Cumplido, J. Arenas-Garcia, and F. Perez-Cruz, "Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2298–2307, Aug 2004.

[44] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive mimo fingerprint-based positioning," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct 2017, pp. 1–6.

[45] G. K. Tso and K. K. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007.

[46] A. K. Fattal, M. Karg, C. Scharfenberger, and J. Adamy, "Saliency-guided region proposal network for cnn based object detection," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–8.

[47] L. H. Hughes, M. Schmitt, L. Mou, Y. Wang, and X. X. Zhu, "Identifying corresponding patches in sar and optical images with a pseudo-siamese cnn," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, no. 99, pp. 1–5, 2018.

[48] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, "Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2164–2168.

[49] S. An, Z. Ling, and L. Dai, "Emotional statistical parametric speech synthesis using lstm-rnns," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec 2017, pp. 1613–1616.

[50] Y. H. Tu, J. Du, L. Sun, and C. H. Lee, "Lstm-based iterative mask estimation and post-processing for multi-channel speech enhancement," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec 2017, pp. 488–491.

[51] F. C. Sun, Z. Q. Sun, R. J. Zhang, and Y. B. Chen, "Neural adaptive tracking controller for robot manipulators with unknown dynamics," *IEE Proceedings - Control Theory and Applications*, vol. 147, no. 3, pp. 366–370, May 2000.

[52] M. L. Vaughn and J. G. Franks, "Explaining how a multi-layer perceptron predicts helicopter airframe load spectra from continuously valued flight parameter data," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 2, July 2003, pp. 1059–1064 vol.2.

[53] M. Rafiq, G. Bugmann, and D. Easterbrook, "Neural network design for engineering applications," *Computers Structures*, vol. 79, no. 17, pp. 1541 – 1552, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045794901000396

[54] J. Gónzalez-Gómez, "Identificación de la temperatura electrónica de un plasma frío por medio de redes neuronales artificiales," Master's thesis, Instituto Tecnológico de Costa Rica, Escuela de Electrónica., Cartago, Costa Rica, 2015.

[55] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Information Sciences*, vol. 364, pp. 146–155, 2016.

[56] S. Bayram, M. E. Ocal, E. Laptali Oral, and C. D. Atis, "Comparison of multi layer perceptron (mlp) and radial basis function (rbf) for construction cost estimation: the case of turkey," *Journal of Civil Engineering and Management*, vol. 22, no. 4, pp. 480–490, 2016.

[57] N. B. A. Aziz and W. F. H. Abdullah, "Comparison between mlp and rbf network in improving chemfet sensor selectivity," in *Computer Applications & Industrial Electronics (ISCAIE), 2015 IEEE Symposium on.* IEEE, 2015, pp. 165–170.

[58] M. A. Ghorbani, H. A. Zadeh, M. Isazadeh, and O. Terzi, "A comparative study of artificial neural network (mlp, rbf) and support vector machine models for river flow prediction," *Environmental Earth Sciences*, vol. 75, no. 6, p. 476, 2016.

[59] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E*, vol. 69, p. 066138, Jun 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.69.066138

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[61] Anonymous, "On the convergence of adam and beyond," *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[62] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[63] MTSS. (2011) Lista de salarios. Artículo web. [Online]. Available: http://www.mtss.go.cr/temas-laborales/salarios/lista-salarios.html