JEDID-JAH DORNELES DOS SANTOS

Framework for controlling automation devices based on gestures

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2021

JEDID-JAH DORNELES DOS SANTOS

Framework for controlling automation devices based on gestures

Master's Dissertation in Electrical and Computer Engineering

Work done under the supervision of:
Professor Doutor João Miguel Fernandes Rodrigues
Professor Doutor Ivo Manuel Valadas Marques Martins

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2021

Framework for controlling automation devices based on gestures

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

I hereby declare to be the author of this work, which is original and unpublished. Authors and works consulted are properly cited in the text and included in the reference list.

_____

(Jedid-jah Dorneles dos Santos)

Abstract

Nowadays people's routine is becoming more and more fulfilled, when we are at home or in the office we have constant activities, appointments, and meetings. With the growth of technology, there are several ways it can help and even replace people in certain tasks. Nevertheless, many mundane tasks are not yet possible to be done only by a computer or a robot, and machines and people must work together to achieve the objective.

Specifically, devices such as sensors or actuators have become very common in our daily life and they are a great help to have a more comfortable environment in our homes or our work, once they allow us to control the lighting, air conditioning, or multimedia, etc. With this in mind, it is important to develop adaptive interfaces that can instantly adjust to the needs and conditions of each user, making people’s activities more efficient.

This dissertation presents a framework that performs the integration of human actions, being gestures and/or poses to activate and control devices that have the standard KNX protocol. A pose detection algorithm is used to detect different gestures/poses, where each gesture or group of gestures integrated with KNX allows easy and universal communication with various types of existing automation devices. The algorithm has standard gestures/poses which are obtained by making comparisons between the coordinates (key points) obtained by the pose estimation, comparing the key points, for example, of the wrist and shoulder for detection of a gesture where the user raises the arm vertically. In addition to standard gestures/poses, it is possible to carry out training so that the algorithm can learn new gestures and, in this way, being adaptive for each type of user. This way, with the detection of the user's gesture/pose, an interaction is then made with the home automation different types of equipment by KNX protocol, each user's gesture performs an interaction on the equipment, such as activating, deactivating, changing its intensity or its mode of operation.

The results show that the framework is capable of effortlessly controlling different devices with different functionalities.

Resumo

Atualmente a rotina das pessoas está cada vez mais preenchida, quando estamos em casa ou no escritório cumprindo atividades, compromissos e reuniões constantemente. Com o crescimento da tecnologia, existem várias maneiras de auxiliar e até mesmo substituir as pessoas em certas tarefas. No entanto, muitas tarefas mundanas ainda não são possíveis de serem realizadas apenas por um computador ou um robô, e é necessário que a máquina e a pessoa trabalhem juntas para atingir o objetivo.

Especificamente, dispositivos como sensores ou atuadores tornaram-se muito comuns no nosso dia a dia e são de grande ajuda para termos um ambiente mais confortável nas nossas casas ou no nosso trabalho, uma vez que nos permitem controlar diferentes sistemas e equipamentos como a iluminação, ar condicionado ou multimídia, etc.

Com isso em mente, é importante desenvolver interfaces adaptativas que possam se ajustar instantaneamente às necessidades e condições de cada utilizador, tornando as atividades das pessoas mais simples e eficientes.

Esta dissertação apresenta um *framework* que utiliza a integração de ações humanas, gestos e/ou poses para ativar e controlar dispositivos de automação que possuam o protocolo padrão KNX. O algortimo possui gestos/poses standard onde são obtidos realizando comparações entre as coordenadas (pontos chave - *keypoints*) obtidas pela pose estimation, comparando os *keypoints*, por exemplo, do pulso e do ombro para uma deteção de um gesto onde o utilizador levanta o braço verticalmente. Além de gestos/poses standard é possivel realizar o treinamento para que o algortimo aprenda novos gestos e, desta forma, sendo adaptativo para cada tipo de utilizador. Deste modo, com a detecção de gesto/pose do utilizador é então feita uma interação com os equipamentos de domotica por protocolo KNX, cada gesto do utilizador realiza uma interação nos equipamentos, como ativar, desativar, alterar sua intensidade ou seu modo de funcionamento.

Os resultados mostram que o *framework* é capaz de controlar facilmente diferentes dispositivos com diferentes funcionalidades.

Acknowledgments

I would like to thank my advisors, Professor *João Rodrigues* and Professor *Ivo Martins*, for their constant support and availability, for their valuable feedback and guidance, for having the patience to always listen to my ideas. For always being readily accessible even with a global pandemic happening and most importantly for giving me the chance to work on something I'm passionate about and showing immense interest in all the developments along the way.

I thank my parents for always giving me total support to study, being by my side and fighting with me, always giving me the example of what it means to work to achieve your goals.

I also have a special thanks to my wife Naithá, for all her love and support, for letting me fill the house with computers, monitors, cameras, and making gestures and poses with me, for always inspiring me to be the best I can be, for believing in me and my potential and for being patient with me in all my madness in this trajectory.

Table of Contents

1 Introduction 14

1.1 Introduction 15

1.2 Objectives 17

1.3 Contents 18

2 State of the Art and Contextualization 19

2.1 Introduction 20

2.2 State of the art 20

2.3 Discussion 22

2.4 Pose Estimation and Face Recognition 23

2.4.1 OpenCV 23

2.4.2 Pose Estimation 24

2.4.3 Face Detection and Recognition 26

2.5 KNX Protocol 28

2.5.1 Topology 30

2.5.2 Communication Telegram 31

2.5.3 KNXnet/IP 32

2.5.4 Equipment configuration 34

3 Adaptive Control Devices Framework 37

3.1 Introduction 38

3.2 Recognize user 41

3.3 Recognition of human movements 46

3.4 Communication with devices 47

3.5 Association of movements with KNX devices 48
3.6 Navigation between devices 53
3.7 Learning gestures 55
3.8 Store and share actions the cloud 60

4 Tests and Results 62
4.1 Introduction 63
4.2 Initial Prototype 63
4.3 Final Prototype 65
4.4 Discussion 72

5 Conclusions and Future Work 74
5.1 Conclusions 75
5.2 Future Work 77
5.3 Publications 78

References 79

List of Figures

Figure 1: Left, Skeleton-based Model. In the middle Contour-based Model, and on the right Volume-based Model (adapted from Yucheng *et al.* (2020)). 25
Figure 2: Example BlazePose with 33 key points (adapted from Bazarevsky *et al.* (2020)). 26
Figure 3: Haar features on the left; Haar-like features scan in images on the right (adapted from OpenCV (2021)).....27
Figure 4: Example of different methods for Face Detection. Left, deection using Haar Cascate, middle using BlazeFace and right Kartynnik *et al.* (2019) method (see details in the text). 27
Figure 5: KNX architecture (adapted from KNX (2020)). 29
Figure 6: Topology KNX (adapted from KNX (2020)) 30
Figure 7: KNX Telegram Structure (adapted from Pimpare (2017)). 31
Figure 8: KNXnet/IP model OSI (adapted from (KNX, 2020)). 33
Figure 9: On the left, Example of KNXnet/IP routing: (adapted from (KNX,2020)). On the right a KNX IP Device.....34
Figure 10: KNX Individual Address.....34
Figure 11: KNX Group Address.....35
Figure 12: ACDf Model.....39
Figure 13: Steps to train face recognizer. 42
Figure 14: Example of image storage to perform training.....42
Figure 15: Applying face recognition with the trained model.....43
Figure 16: Example of a 4-camera environment for detecting gestures/poses.....44
Figure 17: Example detection with multiple cameras, from left to right: Camera 01; Camera 02; Camera 03; Camera 04.....45
Figure 18: The GluonCV tool application to detect movement.....46

Figure 19: Example of a simple connection of KNX devices (adapted from KNX, (2020)). 47
Figure 20: Example of movements used in the framework, from left to right: Right Arm Up; Right Arm Slide; Left Arm Slide; Right Arm Side; Left Arm Side; Left Arm Up. 49
Figure 21: Application of pose estimation in different users in different scenarios.....51
Figure 22: Navigation process between devices with auditory feedback. 54
Figure 23: The MediaPipe tool application to detect movement. 55
Figure 24: Hand landmarks (adapted from MediaPipe, (2021)). 56
Figure 25: Steps for learning new gestures 57
Figure 26: Examples collect frames to new gestures 59
Figure 27: Google Drive API Relationship Diagram 61
Figure 28: Top row, detailed image of the KNX IP router (left) and devices (right). Bottom row, detection of a swipe-up (left), and the activation of the KNX device – lamp (right). 64
Figure 29: Top row, detailed image of the switches used (left) and and complete installation with KNX equipment (right). Bottom row, Detection of some movements during testing. 66
Figure 30: Examples of samples collected for training new movements 68
Figure 31: Case study environment 69
Figure 32: Visualization of each of the cameras. Top row, Camera 01 (left) and Camera 02 (right). Bottom row, Camera 03 (left) and Camera 04 (right) 70
Figure 33: Examples of ACDf detection for the case study 71

List of Tables

Table 1: Types of communication media (adapted from Pimpare (2017)).....32
Table 2 : Example 3-level structure.....36
Table 3: Example data available in ETS5 software file.....50
Table 4:Correspondence between actions and movements. Movements marked with (*) are the same movement, if applied, makes the negation of the previous one.....52
Table 5: Table with the percentages of correct answers obtained during the tests.....64
Table 6: Table with the percentage of correct answers obtained during the tests of the first stage of the final prototype.....67
Table 7: Table with the percentage of correct answers obtained during the tests of the second stage of the final prototype.....69
Table 8: Table with the percentage of correct answers obtained during the tests of the third stage of the final prototype.....71

List of Acronyms

ACDf	Adaptive Control Devices Framework
AmI	Ambient Intelligence
AI	Artificial Intelligence
API	Application Programming Interface
BACnet	Building Automation and Control Networks
BC	Backbone Coupler
CEBus	Consumer Electronic Bus
CNN	Convolution Neural Netork
CSV	Comma Separated Values
CV	Computer Vision
DL	Deep Learning
Dlib	Face Detection Library
DVC	Bus Device
EC	Edge Computing
ETS	Engineering Tool Software – KNX Software
FD	Face Detector
FPS	Frames Per Second
GA	Group Address
GPU	Graphics Processing Unit
HMC	Human-Machine Cooperation
IA	Individual Address
IKLDA	Improved Kernel Linear Discriminant Analysis
IP	Internet Protocol
KNX	Home Automation Equipment Communication Protocol
LBPH	Local Binary Pattern Histogram
LC	Line Coupler
LonWorks	Local Operating Network

LSTM	Long Short-Term Memory
ML	Machine Learning
Modbus	Data communication protocol
NLP	Natural Language Processing
NN	Neural Network
OpenCV	Open Source Computer Vision
OSI	Open System Interconnection
PE	Pose Estimation
PNN	Probabilistic Neural Network
Pyttsx3	Text-To-Speech Library
RELU	Rectified Linear Unit
ResNet	Deep Residual Network
RNN	Recurrent Neural Network
TCP	Transmission Control Protocol
TP	Twisted Pair
UDP	User Datagram Protocol
XKNX	Library for IP communication with KNX protocol
XML	Extensible Markup Language

1 Introduction

ABSTRACT

Computer Vision and Home Automation are increasingly used in our daily lives. Human-Machine interaction is increasingly present in our activities, helping us. This chapter introduces the reader to the context and objectives of this dissertation, explaining the importance of a *framework* for interaction with home automation devices through gestures.

1.1 INTRODUCTION

Universal Access can be defined as the global requirement to deal with diversity in a target population of users, which includes people with disabilities, the elderly, and a population with cultural differences (Stephanidis, 2001). The purpose and nature of technological tasks and platforms and the effects of their proliferation at home, offices, and social enterprises is one of the important topics for research. With this in mind, special attention should be given to adaptive interfaces as presented in Rodrigues *et al*. (2017), i.e., interfaces that can adapt quickly to each user (Johnston *et al.,* 2019) to his/her preferences and needs.

The above is related to the great demand for comfort and versatility in the management of different types of devices, mainly at home or in the office. Nowadays, more and more, people want a comfortable environment for their leisure or at work, something practical, sustainable, and safe. Reaching this level of automation today is a real challenge, as it involves "a lot of" wiring, from sensors and actuators to control and monitoring centres. For professionals, the (amount of) "wiring" also means greater design and installation efforts, greater risk of fire, and rising costs.

Nevertheless, there are currently a wide variety of sensors and actuators that communicate in different ways (wiring, Wi-Fi, Bluetooth, etc.), with different equipment; it increasingly makes the communication among those devices of paramount importance to achieve the objective of an automated and adaptive place for the user. There are several solutions to achieve this, one of which is the KNX protocol for communication with and between devices (KNX, 2020).

In the market, there are numerous communication protocols for the application of home automation, such as X-10, CEBus, or LonWorks, which have different characteristics that can make their implementation easier or more difficult. The protocols are divided into two types, the open standard protocols, with their rules open to public knowledge, and the proprietary protocols with their rules being private, and it is only possible to verify the applications in their standard operation. KNX is associated with protocols of the open standard type (Pimpare, 2017).

In summary, the KNX protocol is a technology for home and commercial automation, which allows controlling different devices, such as lighting, security, energy

management, air conditioning systems, etc. KNX is the only open standard for residential and building control and complies with EN 50090, EN 13321-1, and ISO / IEC 14543. With the use of KNX, we can exclude the use of isolated sensors and actuators, making it possible to internet communication with all these devices. The protocol has a topology where it is possible to support several devices from around 500 manufacturers, enabling a wide range of automation applications (KNX, 2020). And one of the great achievements of KNX is its possibility to be used in various types of installation situations; it is possible to use KNX in media with interlocked pair, electrical network, radio frequency, infrared, and the modality used in this project, by Internet (Gonçalves, 2017).

In parallel with home automation and its constant evolution, there is the concept of Ambient Intelligence (AmI) which in turn is a concept intrinsically and fully connected with Artificial Intelligence (AI) (Gams *et al*., 2019). AmI normally refers to electronic environments that are sensitive and react to people's presence, performing integration between data analysis and automation devices (Martín *et al*., 2019).

With ambient intelligence, devices can work together to assist people in their tasks, activities, and routine in their lives in an intuitive way, using information and intelligence that are hidden in the network that connects these devices. This is also connected to the so-called Human-Machine Cooperation (HMC), which consists of the development of autonomous machines that learn to cooperate with humans. HMC does not always aim to improve our efficiency, it does not require absolute computational power, and it depends on intuition, cultural norms, emotions, signs, and pre-evolved dispositions, which is difficult to code on machines. The ultimate purpose of HMC is that the ability of man and machine to work together to achieve that common goal, thus facilitating each other's tasks (Crandall, 2018).

Thus, with the aforementioned, we created a loop effect where the adaptive interfaces “adapt” to each user in real-time so that it is possible to be connected with the HMC and at the same time, allow the creation of AmI environments with less difficulty. The dissertation has its focus on this loop, presenting an initial framework, where it is possible to recognize and adapt each user's predefined or learned movements to control many devices that in turn are using the standard KNX protocol for your internal communication.

The main contribution of the dissertation is the interconnection between human gestures or poses and how those can be organized and communicate with home/office automation equipment, i.e., how to realize a small number of body actions, that can be defined by the user, that can be used in any environment (home, office, …) to command a large different number of devices.
The above will allow creating an environment smarter for its users, bringing facilities to their daily activities.

1.2 OBJECTIVES

The main objective of this dissertation is to develop a framework that recognizes a person's actions/gestures and that through this recognition it is possible to interact/control many home automation devices, particularly using the KNX protocol.
The main objective is divided into the following sub-objectives:

- Analyze, study and write a consistent state of the art;
- Implement and/or adapt a pose and/or a face detector;
- Define and recognize specific “movements” for each user to interact with devices;
- Connect the user's “movements” to the devices using the open protocol standard KNXnet/IP;
- Adapt and/or propose an integrated structure that works with different types of users, including elderly users;
- Develop a *framework* that:
    - integrate all the above;
    - works as “edge computing”;
    - the same commands can be used in any environment around the world (“cloud-based”);
- Test the structure with real images and/or videos (or camera stream) in real environments with different types of users.

1.3 CONTENTS

This chapter presented the reader with the context and objectives of this dissertation. Chapter 2 includes the state of the art and contextualization of KNX control, pose detection methods, and some background concepts to help understand the subject matter in this dissertation. Chapter 3 presents the framework - Adaptive Control Devices Framework (ACDf), and its implementation, Chapter 4 consists of the tests and results obtained using the framework and finally, Chapter 5 presents the conclusion, together with an explanation of what is planned to be carried out in the future work.

2 State of the Art and Contextualization

ABSTRACT

Computer Vision and Home Automation have several applications for different scenarios, it is important to know some of their applications and tools. In this chapter, we present the state of the art on the theme, contextualization, and the main concepts necessary for a better understanding of this dissertation.

2.1 INTRODUCTION

In this chapter the state of the art is presented, bringing concepts necessary for understanding the subjects and the scope of this dissertation, namely concepts as Machine Learning (ML), Face Detector (FD), Pose Estimation (PE), and automation protocols, in this case, focused for KNX.

The concept of Machine Learning (Alpaydin, 2020) can be defined simply, as the study of computer algorithms to optimize a performance criterion using example data or experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or experience. They have been increasingly improved over the years. By connecting ML, including Deep Learning (DL) (Sachan, 2020), with Computer Vision (CV), it is possible to detect, define and quantify human movements and actions (Sawant, 2020), as well as to recognize people for their gait (Chikano *et al*., 2020) or, for example, by their face (Satake, 2020). Nevertheless, it is not yet trivial to apply these concepts to the so-called Edge Computing (EC) (Calheiros, 2020) using (or not) low-cost AI cards, such as Jetson (2019) or Coral (2019).

2.2 STATE OF THE ART

With the continuous development of automation, manual systems are being converted into “intelligent systems”, that assist in people's routines and activities. Simple devices that are being used in our daily lives such as lamps or fans, including equipment to obtain information about the quality of the environment, such as temperature sensors, flame or gas sensors, and countless others (currently available on the market). All these sensors and actuators are increasingly present in our daily lives and with this, the justification for the great growth of systems carry out the communication and transmission of this information between them and the user.

As already mentioned, nowadays it is possible with some low-cost devices to develop monitoring systems to obtain information from certain domestic spaces (Tejesh, 2018) presenting the user with information such as temperature, humidity, and/or other information that is necessary for that environment. More and more intelligent

environments are bringing services to people's routines, along with security. For this to continue growing, it is of utmost importance that communication between sensors and actuators be completely reliable. As presented by Sapundzhi (2020) the KNX system can act on devices that can be monitored or controlled remotely with great efficiency, being possible to control lighting, heating, and multimedia, among other devices.

The KNX, being a technology for home and/or commercial automation, can be implemented for the control and reduction of energy consumption, Atalaya *et al.* (2020) presents a KNX installation for a lighting control network where obtain a significant reduction in energy consumption. Feki (2019) shows that the KNX protocol can be used together with the Raspberry PI for a home monitoring system, being possible to check the temperature, humidity, presence sensor, CO2 sensor, and light control. KNX has also been tested and integrated with other types of systems where, for example, in a classroom, a Natural Language Processing (NLP) was implemented so that it was possible to control by voice commands equipment such as lights, printer and features TV (Yumang *et al.*, 2020).

Due to the constant development of home automation, cameras (IP cameras, for example) are becoming part of this system to make intelligent environments even more efficient. Cameras are one of the cheapest sensors that allow us to replace a variety of sensors used in home automation, such as motion detectors, touch sensors, light sensors, as well as all types of switches. For that, as already mentioned, we connect Computer Vision with modern Machine Learning methods, including DL, making it possible to detect, define and quantify human movements and actions, as well as to recognize people by their gait or by their face.

Sawant (2020) describes a method to recognize human activities, using a recurrent neural network (RNN) being possible to detect activities such as waving hands, jumping, and clapping. It is also possible, with Pose Estimation, to detect several people in real-time (Cao, 2021). Using this technology, it is possible to recognize the gait of a person, for example, an elderly person, and extract information to detect a risk of falling, bringing security and extending healthy life expectancy (Chikano, 2020).

In the case of facial recognition, it can be used for assisted robots that collaborate with people, as described by Baltanas *et al*. (2020), or in the case of Bellotto *et al*. (2017) to help the elderly with mild cognitive disabilities. Both have in common the development

of environments where it is possible to use these two technologies, Computer Vision and Home Automation (sensor/actuator devices), to help/facilitate everyday people, especially the elderly, where they have the most needs.

More examples of intelligent environments, monitored only by a camera, are shown by Daher *et al*. (2017), where they track and recognize the basic activities of an elderly person or as presented by Chen *et al*. (2020) that analyzes falls from people using OpenPose, where the key points of the human skeleton and the speed of descent in the centre of the hip joint, the angle of the body's central line with the ground and the width-height relationship of the external body rectangular is used to detect the fall.

2.3 DISCUSSION

The present state of the art (and additional literature, not presented) shows that different systems can interact with home automation devices using the KNX protocol, using different methods and tools. Likewise, Computer Vision presents numerous tools for detecting body, face, and movement. Different systems were presented showing the applications of this technology for monitoring people and interactions between machines and humans.

All systems presented uses only KNX tools to control devices or uses ML for detection and/or monitoring of people, but they do not work together, using the tools on the same system. Home automation has several applications, so it is of great interest that some technologies, such as CV, ML, and other technologies presented, can work increasingly together, emerging different forms of communication between them so that it is possible to bring even more applications to people's daily lives, bringing even more facilities and increasingly helping users' activities and leisure.

As of this, it is possible to develop touchless interfaces, which can control any wireless device in real-time. In addition, it can also allow the development of interfaces that can adapt in real-time to the natural actions of each user. For example, different users can turn a light on or off using different movements, or even the same user can consider different movements depending on which room in the house or office, etc. he/she is. It is important to note that this technology can also allow the user to program actions in a specific environment, for example at home, and export the same movement to a

different environment, for example to the office, without the need to reprogram the action for the same occupation.

Currently, with the sars-cov-2 pandemic and the COVID-19 disease, we have seen even more the need for automation systems that do not require touch activation, bringing even more safety and quality of life to users.

In summary, none of the systems presented has the characteristic of controlling/interacting with automation devices through gestures or poses, and in addition, these gestures/poses being different from person to person (adaptative to the person), in addition, they do not have the feature of different gestures/poses from different persons can activate the same functionality to the device that is being controlled. Complementing, the same person can also use the same gesture/pose to control different devices anywhere in the world without the need to change or teach (again) the system. This type of system is not yet available in the market.

To develop this system, it will be necessary to use Pose Estimation and communication via KNX protocol for home automation devices. We are going to detail this in the following sections.

2.4 POSE ESTIMATION AND FACE RECOGNITION

Before presenting the Pose Estimation and its characteristics, an initial explanation about the library OpenCV (OpenCV, 2021) is necessary for a better understanding of the tools.

In this section two technologies provided in the OpenCV library are presented: Pose Estimation, which is used to detect and analyze human posture, and Face Recognition, which makes it possible to recognize a particular person by their face.

2.4.1 OpenCV

As mentioned before, the simplified concept of Computer Vision is to model and replicate human vision using software and hardware. It is the science and technology of machines that see. CV investigators develop theory and technology for building artificial systems that get information from images or any multidimensional data. Thus,

there are several technologies to put this concept of CV into practice; one of them is the OpenCV library (OpenCV, 2021).

OpenCV is an open-source programming library to make Computer Vision more accessible. The library contains hundreds of methods that support capturing, analyzing, and manipulating visual information provided to a computer by webcams, video files, or other types of devices. With this tool, simple functions can be used to draw a line or other shape on a screen, while the more advanced parts of the library contain algorithms for face detection, motion tracking, and shape analysis (Antonello, 2018). Many of the algorithms in this library are related to specific uses of Computer Vision, including product inspection, medical imaging, robotics, facial and gesture recognition, human-computer interaction, and of course, also human pose estimation.

2.4.2 Pose Estimation

In more detail, pose estimation is a technology based on Computer Vision that analyzes and detects people's posture, its main component is the possibility of modelling the human body. Three types of models of the human body are most used (Yucheng *et al.*, 2020). Figure 1 shows an example of each of the models described below:

- ***Skeleton-based Model***: This model is a set of keypoints (or joints) such as ankles, knees, shoulders, and elbows that make up the skeletal structure of a human body. Two dimensions (2D) and 3D pose estimation techniques are used for this model due to its flexibility. See Fig. 1 left;
- ***Contour-based Model***: The contour model consists of the contour and approximate width of the trunk and limbs of the human body, where the body parts are presented with limits and rectangles of the silhouette of a person. See Fig. 1 middle;
- ***Volume-based Model***: In the volume model are the 3D shapes of the person's body being represented by volume-based models with meshes and geometric shapes. See Fig. 1 right.

Over the years there has been an increase in different datasets with different methods for the application of Pose Estimation. Bulat *et al*. (2020) present an MPII Human Pose dataset with the Soft-gated Skip Connections method, already in 2019 Feng *et al*.

(2019) show the COCO test-dev as a dataset and an HRNet-W48 + DARK method. Artacho *et al*. (2021) use the Leeds Sports Poses dataset with the OmniPose method. This is always due to the application in question and the criteria for using the different datasets or methods.

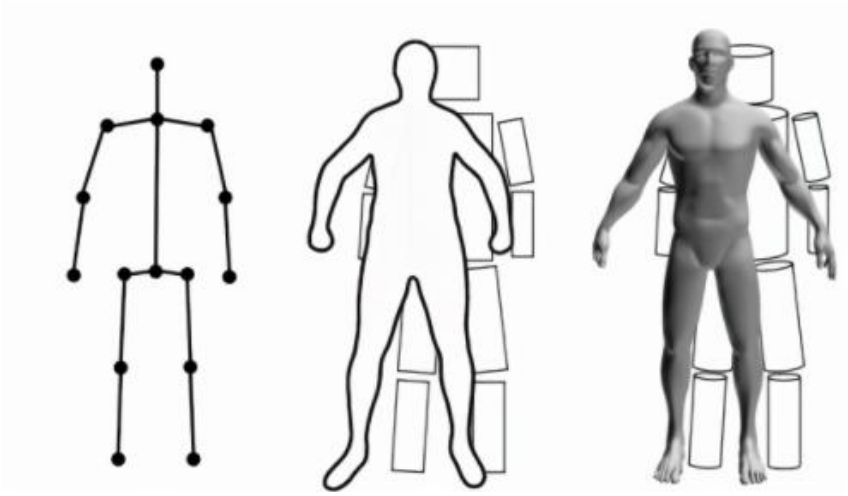

Figure 1: Left, Skeleton-based Model. In the middle Contour-based Model, and on the right Volume-based Model (adapted from Yucheng *et al.* (2020)).

Currently, we can find models already tested and pre-trained, ready to use, as is the case of the GluonCV tool (Guo *et al*., 2020; GluonCV, 2020) which is a deep learning kit for Computer Vision that works in Python where it is possible to find classification algorithms, object detection and pose estimation, among others. GluonCV type of model used for the pose estimation is "skeleton-based" as shown in Fig. 1, on the left. Another technology that uses the same type of model, that is more recent and faster is MediaPipe (MediaPipe, 2021; Grishchenko *et al.*, 2020) which has algorithms the face detection, face mesh, iris, hands detection, pose, hair segmentation, object detection and recognition, box tracking, instant motion tracking. MediaPipe is one of the tools where it is possible to integrate models such as face detection, detection of hands, and pose that, initially implemented separately, can be implemented at the same time bringing even more advantages for the implementation of computer vision.

These technologies have a topology consisting of n reference key points on the torso, arms, legs, and face, GluonCV $n = 17$. These key points are interconnected, thus creating the *Skeleton Model*. There are other models where there are more key points (Bazarevsky *et al*., 2020), to have a more accurate detection, such as the BlazePose

model (Fig. 2), for example, which has n = 33 key points but the philosophy remains the same.

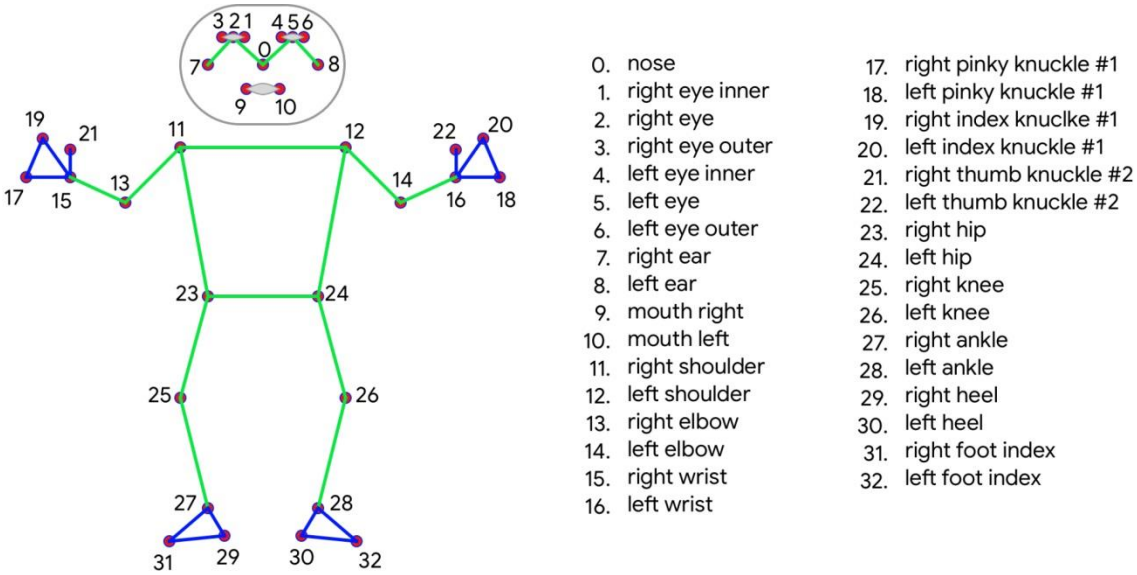

Figure 2: Example BlazePose with 33 key points (adapted from Bazarevsky *et al*. (2020)).

With the pose estimation we can have different applicability (see section 2.2), one of them is the recognition of human activities (Sawant, 2020) and for this, it is necessary to carry out training in the pose estimation so that this recognition of a certain movement or activity (see section 3.7).

2.4.3 Face Detection and Recognition

Face detection is a Computer Vision technology that helps to locate/view human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance in several areas.

OpenCV has, among its different technologies, a cascading classifier based on Haar resources (Fig. 3). This is a machine learning object detection method in which the cascade function is trained from many positive and negative images. It is then used to detect objects in other images (OpenCV, 2021).

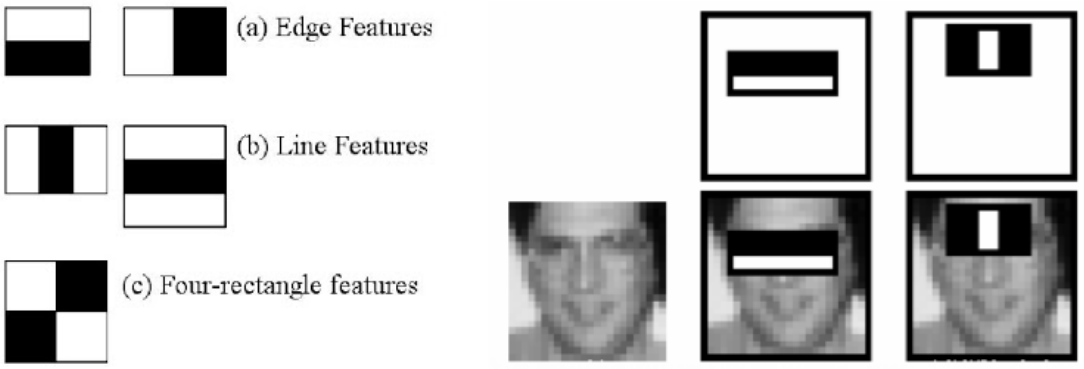

Figure 3: Haar features on the left; Haar-like features scan in images on the right (adapted from OpenCV (2021)).

For human face detection with this classifier, Haar features are the main part. Haar features are used to detect the existence of features in a given image. Simplifying, each feature produces a single value that is calculated by subtracting the number of pixels under the white rectangle from the number of pixels under the black rectangle. Haar features are rectangular for quick detection of the human face. Applying Face Detection with Haar Cascade is shown in Fig. 4 on the left.

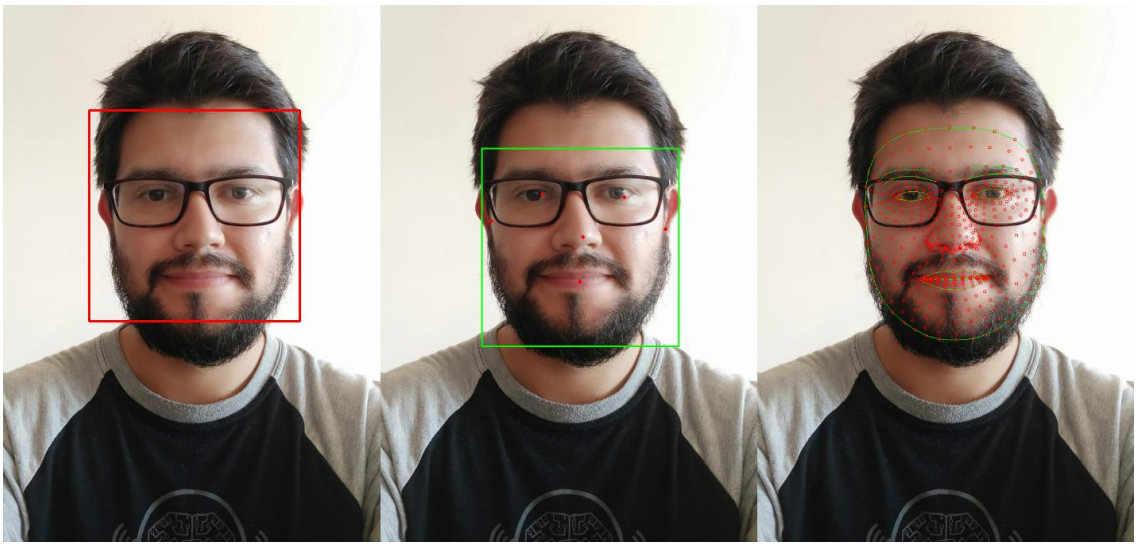

Figure 4: Example of different methods for Face Detection. Left, deection using Haar Cascate, middle using BlazeFace and right Kartynnik *et al.* (2019) method (see details in the text).

However, like pose estimation, there are different methods for face detection. Yang *et al.* (2018) use the YOLO tool for face detection using an input image size of 416 × 416 pixels (px) and after a series of batch convolution and normalization operations on the

input image, the input image is sampled three times, 32 times, 16 times, and 8 times, and the resource map at various scales is obtained.

With MediaPipe we also found tools for face detection, through Bazarevsky *et al.* (2019) with BlazeFace. A solution made for mobile GPU and presents 6 reference points in its detection (shown in Fig. 4 in the middle). Kartynnik *et al.* (2019) also present a different solution through a facial geometry that estimates 468 reference points that in addition to returning a person's face, it is also possible to detect eyes, mouth, nose, and eyebrows (shown in Figure 4 on the right).

For face recognition there are also several methods available, for instance, described by Dinalankara (2017) who uses Haar-Cascade for face detection and recognition uses the LBPH (Local Binary Pattern Histogram) method. Using similar tools Bah *et al*. (2020) combines the LBPH technique with contrast adjustments, bilateral filtering, histogram equalization, and image combinations for a higher precision search for face recognition. Ouyang *et al.* (2020) present a hybrid method that uses a combination of Probabilistic Neural Networks (PNNs) and Improved Kernel Linear Discriminant Analysis (IKLDA) to perform face recognition and Xu *et al.* (2020) show the possibility of using face detection with Dlib and face recognition using Convolution Neural Network (CNN) and Deep Residual Network (ResNet).

With these tools, we can then carry out training so that it is possible to recognize a specific user or different users at the same time (see section 3.2).

2.5 KNX PROTOCOL

In the market, there are different types of communication protocols for the home automation area. Each protocol has some specific characteristics that make them different and that can make them easier or more difficult to use. The home automation protocols are divided into two groups, the open standard protocols, where the rules are available to the public, and the proprietary protocols where it is only possible to verify the systems in operation, without having their access available. The KNX protocol fits into the group of open protocols.

The KNX protocol emerges through the European Home Systems Association to establish itself as a European and worldwide standard. The KNX Association has supports for systems before KNX such as Batibus, EIB, and EHS. For EIB systems,

KNX is compatible and for this reason, many devices may bear the EIB logo, although only the KNX designation is currently being used (Pimpare, 2017).

With KNX it is possible to control different devices, such as lighting, security, power management, HVAC systems, etc. KNX is also the only open standard for residential and building control and complies with EN 50090, EN 13321-1, and ISO / IEC 14543. With the use of KNX, we can exclude the use of isolated sensors and actuators, making communication possible over the internet with all these devices. The protocol has from around 500 manufacturers, allowing a wide range of automation applications (KNX, 2020).

Figure 5 presents a simple way the distributed architecture of the KNX protocol where the connection with the actuating devices, such as lamps or heaters and the sensing devices, as luminosity is made with a control cable in parallel with a 230V cable. This means that: the amount of cabling compared to conventional installation technology is considerably reduced when the bus devices are arranged in a decentralized manner, the number of possible system functions is increased, and the transparency of the installation is improved.

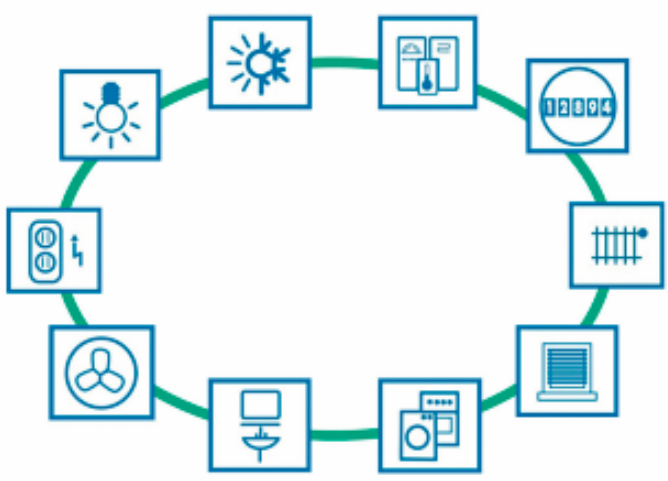

Figure 5: KNX architecture (adapted from KNX (2020)).

2.5.1 Topology

The topology of an installation is structured into areas, lines, and line segments (Fig. 6). Each bus device (DVC) can exchange information with any other device utilizing telegrams.

One line consists of a maximum of 4-line segments, each with a maximum of 64 bus devices. Each segment requires an appropriate power supply. The actual number of devices is dependent on the power supply selected and the power input of the individual devices.

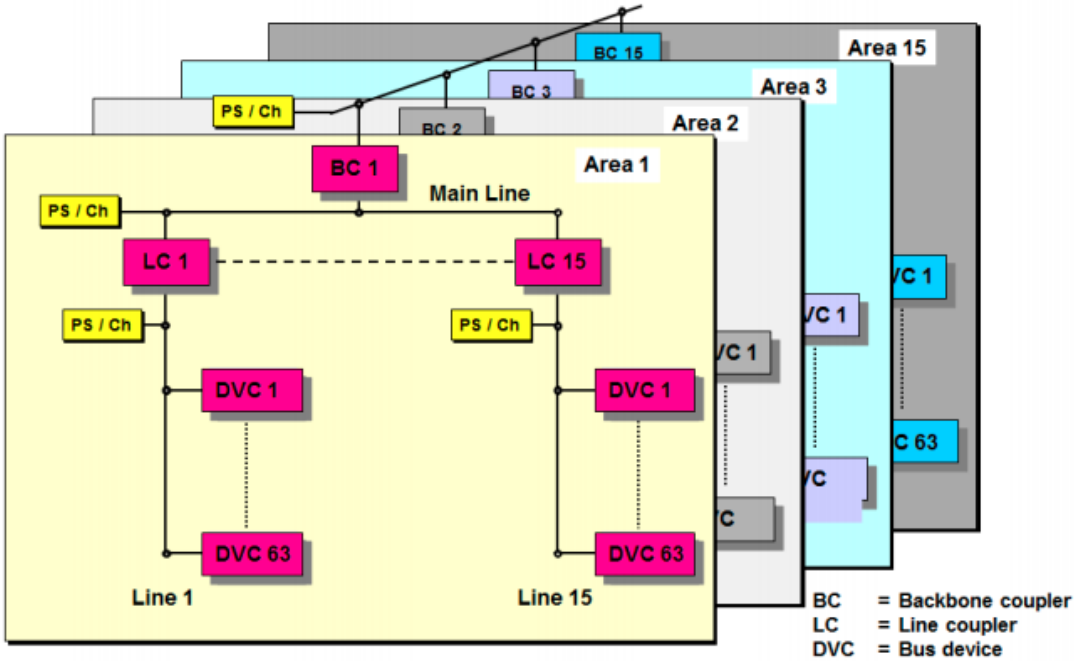

Figure 6: Topology KNX (adapted from KNX (2020)).

The smallest segment in the KNX topology is the line segment, which consists of the coupling of a power supply, with a maximum of 64 devices to the bus. The next segment is a line that can have a maximum of 4-line segments, being able to obtain up to 256 bus devices. If more than one line is used, they must be coupled to the mainline, called a line coupler (LC), making this set an area. In this area, up to 15 lines can be coupled.

In case more than one area is used, the connection between them is made by a connection called the area line or backbone (BC). For the connection between areas, an area coupler is required. The maximum number of areas that can be connected is 15 (KNX, 2020).

2.5.2 Communication Telegram

When an event occurs, for example, using a push-button a telegram is transmitted. Telegrams are essential for communication between devices and are parameterized to ensure compatibility between them. Each time an event occurs; the device generates and sends a telegram via the communication bus. The structure of a telegram consists of the following components, as shown in Fig. 7.

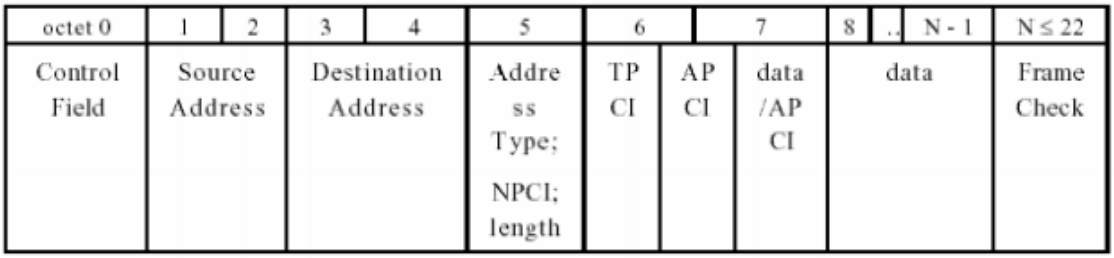

Figure 7: KNX Telegram Structure (adapted from Pimpare (2017)).

The structure of a telegram has four fields; first, it is the control field, with octet 0. After the telegram it has the direction field (octet 1 to 6); the data field (octets 7 to 16 bytes), where the data in this field can vary its length. The information is transmitted in its entirety is in the form of 8-bit characters, octets. The last field that appears in the telegrams is the error detection data, which guarantees a level of reliability in the transmissions (KNX, 2020). The fields are:

- ***Control:*** This field indicates the priority of a given telegram when it is sent by the device on the bus. If one of the bus devices has returned a negative acknowledgement the transmission of the telegram is repeated, a repeat bit 0 is set, this way it is guaranteed that bus devices that have already executed the appropriate command will not execute it again;
- ***Source Address:*** This field indicates the physical direction of the device sending the telegram (4 bits for the area, 4 bits for the line, and 8 bits with the device number);
- ***Destination Address:*** If the most significant bit is the value of "0", this field is a physical address, individual address, and the telegram is sent exclusively to one device. If the value is "1" it is a group address, and the telegram is addressed to all devices that have this address;

- ***Frame Check:*** This field consists of a byte that is obtained from the parity calculation, when a device receives the telegram, to check if it is correct from the control byte.

Knowing communication telegram, the next step is how to communicate.

2.5.3 KNXnet/IP

The KNX protocol has the possibility of various media for communication, the most common is by cable (twisted pair), but it is also possible to communicate by the power grid, radio frequency, and Ethernet (IP). Table 1 summarizes the types of communication media and their main areas of activity.

Table 1: Types of communication media (adapted from Pimpare (2017)).

Communication	Type	Areas of use
Twisted Pair	Dedicated cable	New constructions or profound refurbishment of facilities. Where large amounts of communication data are transferred.
Power Grid	Electric cables	In places where a bus cable does not exist, but there is a 230 V power cable.
Radio Frequency	Radio	In places where there is no wiring or where such wiring is not desired.
IP	Ethernet	In large installations where a structure is needed of interconnection. Residential installations without the need to change electrical connections already made.

Making the focus for the type of IP communication, which is related to this dissertation, Ethernet is an open, high-performance, local, and wide-area network, compliant with the international standard IEEE 802.3 (Ethernet). Ethernet is used for local area networks, especially in conjunction with the Internet. All over the world, there is a wide variety of different network structures; however, for communication between two (or more) devices these definitions are generally not sufficient; some other details need to be defined.

TCP/IP is a group of protocols or rules (protocol family) introduced in 1984 is now widely used. Although generally discussed in the form “TCP/IP”, TCP (Transmission

Control Protocol) and IP (Internet Protocol) are two distinct protocols. Strictly speaking, the TCP/IP Internet protocol package also includes a third equally important protocol: UDP (User Datagram Protocol) (KNX, 2020).

The IP protocol performs the role of ensuring that data packets are sent from one device to another. The TCP protocol is based on the IP protocol and is used to establish a permanent connection with error checking and ensures that all data packets are sent in the correct order. Connecting KNX to Ethernet has several advantages, such as:

- Existing network infrastructure in the building/house can be used for KNX backbone and main lines (higher speed, more cost-effective, and more convenient);
- Buildings/houses can be monitored and controlled via Ethernet from anywhere in the world;
- Several individual websites can be watched and maintained from a central location on the Internet;
- KNX client installations can be analyzed and programmed remotely over the internet by the KNX system designer.

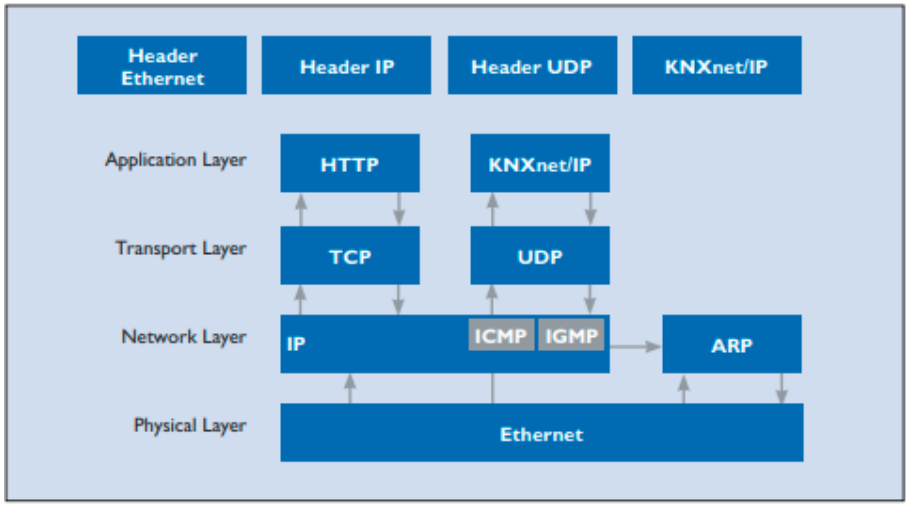

Figure 8: KNXnet/IP model OSI (adapted from (KNX, 2020)).

With this in mind, the KNX protocol has KNXnet/IP which is the KNX protocol via IP communication. The IP communication in KNX can be explained using the OSI reference model (Fig. 8) where communication takes place via the application layer (which generates the KNXnet/IP telegram), the transport layer (UDP), the network layer (IP), and Ethernet – the physical layer. Like with the TP protocol, additional

information for the respective layer (the header) is always added to the KNXnet/IP information.

This method has two types, tunnelling and routing communication. Tunnelling is used to access a local network bus to program the KNX installation, for example, and routing is used to exchange telegrams over an Ethernet network, connecting two KNX TP systems via Ethernet (Fig. 9, left), for example. IP communication KNX device is also shown in Fig. 9 right.

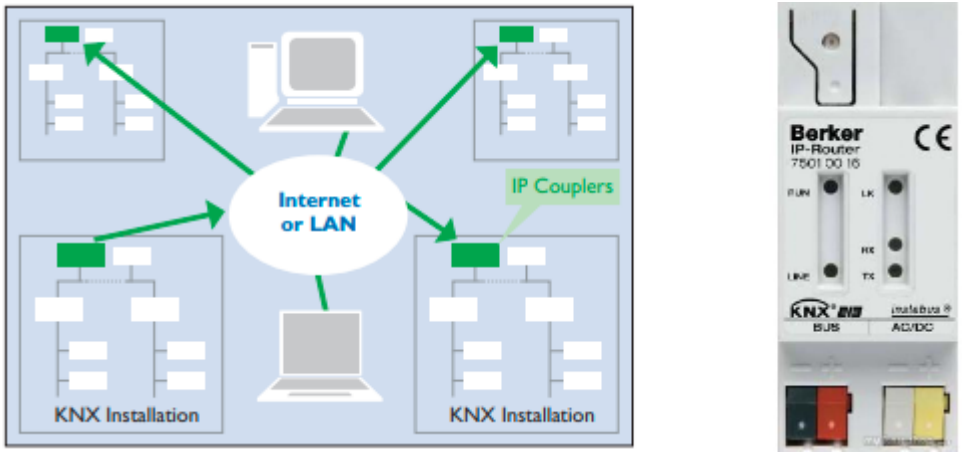

Figure 9: On the left, Example of KNXnet/IP routing: (adapted from (KNX,2020)). On the right a KNX IP Device.

2.5.4 Equipment configuration

For KNX devices connected to the bus to communicate, they must initially be configured to be recognized. KNX devices must have individual addresses and group addresses for the installation to function normally.

The ***Individual Address*** (IA) is unique for each KNX device. This address has the following format: area [4 bit] - line [4 bit] - device [1 *byte*] (Fig. 10).

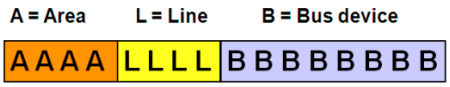

Figure 10: KNX Individual Address.

The area is the address where the device is located and values can be assigned from 1 to 15 as this is the number of possible areas. The second field means the line and can also

be assigned a value from 1 to 15 because it is the number of possible lines and the third and last field is the device and the value can be from 1 to 255 (KNX, 2020).

This address is used for the following purposes: diagnosis, error detection, installation modification by reprogramming.

Communication between KNX devices in an installation is done via ***Group Addresses*** (GA). The group address is defined via the *ETS* software and can be selected as a 2-level structure (main group/subgroup), 3-level (main group/intermediate group/subgroup), or a freely defined structure (Fig. 11). For the case of the specific address 0 / 0 / 0, it is reserved for telegrams to all devices available on the bus.

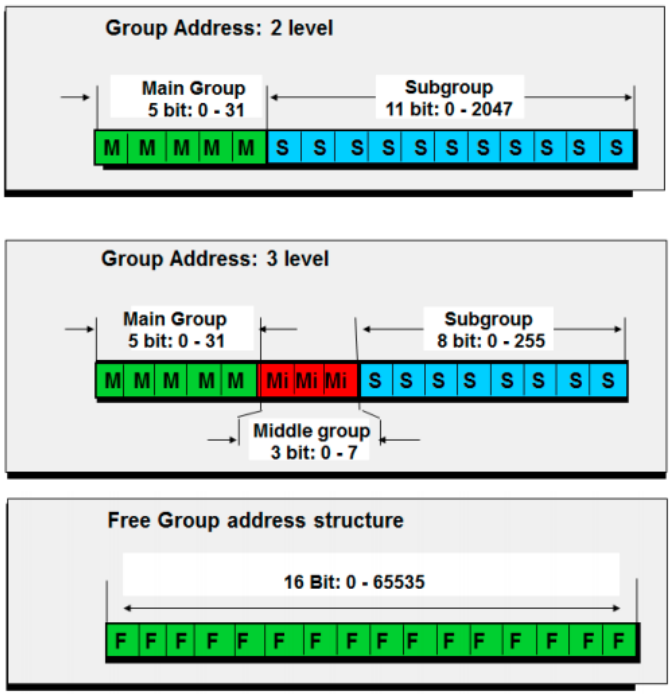

Figure 11: KNX Group Address

The ETS project engineer can define how the addressing levels will be used and must follow this standard for the entire project.

Each group address can be assigned to bus devices as needed, regardless of where the device is located on the system. Actuators can read multiple group addresses; however, sensors can only send one group address in each telegram. Table 2 shows an example of a group address assignment with a 3-level structure.

Using a standard where the main group refers to the type of KNX equipment, for example, lighting, blinds, electrical outlets, air conditioning, etc. The middle group using a larger area of the location of this equipment, for example, a kitchen, living

room, bedroom, bathroom, etc. And the subgroup being the specific location within that larger area, the electrical outlet, which is in the kitchen and is connected to the fridge. Thus, having a standard for structure and being possible to build a project with this standard.

Table 2 : Example 3-level structure

Main Group	Middle Group	Subgroup	Address
1: Lighting	1: Living room	1: Balcony	1/1/1
1: Lighting	2: Bedroom	1: Bed	1/2/1
2: Blinds	1: Living room	1: Balcony	2/1/1
2: Blinds	2: Bedroom	1: Balcony	2/2/1
3: Electrical outlet	1: Kitchen	1: Fridge	3/1/1

In the next chapter, it is presented the framework architecture.

3 Adaptive Control Devices Framework

ABSTRACT

Human-Machine interaction is increasingly present in our activities. This dissertation proposes a *framework* for communication with home automation devices using a user's gestures/poses. We demonstrate the structure and functioning of the *framework*.

3.1 INTRODUCTION

The Adaptive Control Devices Framework (ACDf) has the following 5 processes: (i) Recognize user; (ii) Detect movements/actions; (iii) Association between human movements and devices; (iv) Communication with devices and (v) feedback from the interface (system/devices) - usually by sound.

In the case of the process (i) and (ii), it can use any of the cameras available in the environment, and in the process (iii) the system automatically, for the first time a user is detected in that environment downloads the user movements/actions from the cloud, for the remaining times the system has local memory to store *N* uses (standard configuration is 6 users). Finally, to get the information from the cloud the option "cloud" must be activated, i.e., to check if the user is already in the global application.

It is important to stress, that a user is always checked if exists in local memory, only if does not exist is checked in the cloud, and if exists his/her information is brought to the local system.

For understanding ACDf, a diagram with its main functions is presented in Fig. 12, followed by a short explanation of each component. The ACDf has 7 main modules which consist of (a) recognizing the user, (b) detecting their movements, (c) associating these movements with certain actions to activate the equipment's functionalities, (d) having the possibility of navigating between the devices and their functionalities, (e) communicating with their equipment by KNX protocol, (f) learn new moves if necessary and (g) save these moves in the cloud, allowing access from other locations.

In more detail, the functionalities of the models are:

- ***Recognize User:*** This component recognizes the user who is using the framework, by face, so that it is possible to use their respective movements and thus allow more than one user to use the framework at the same time and each user uses their respective movements for the same actions. A face detection model is used to capture face images regions and a recognition model is used to train the face recognition model that deals with present users and new users.

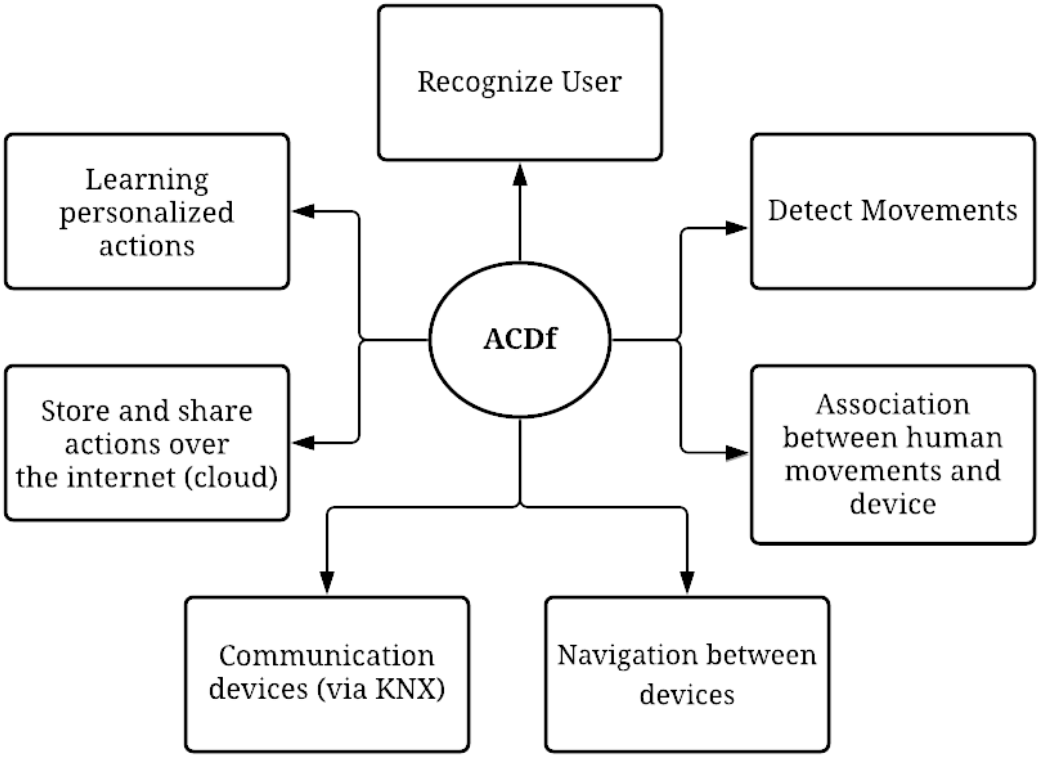

Figure 12: ACDf Model.

- ***Detect Movements:*** For movement detection, the GluonCV library was initially used, but throughout development, another library was found and obtained faster results. The new library is Mediapipe. In this component, pose estimation is applied so that it is possible to find the keypoints on the human body and from these points, comparisons can be made between the coordinates and check if the user has made a certain movement.
- ***Association between human movements and devices:*** According to the movement detected, an action is taken by the framework. In this model, pre-defined moves are defined to be used by the user. With these movements, it is possible to activate and deactivate the functionalities of the devices and navigate between the devices.
- ***Navigation between devices:*** When the installation has more than one device, the framework allows navigation between these devices so that it is possible to use the user's gestures. After each movement/navigation audio feedback is returned so that the user does not have to go to a screen to choose which device he wants to activate.

- ***Communication with Devices (via KNX):*** The communication with the devices will be with the KNX protocol, using a connection through a KNX IP device. To make this possible, along with the detection of gestures, a Python library is used to manage the devices and their addresses.
- ***Learning personalized actions*****:** In this component, new poses/gestures are learned for each user using the new MediaPipe library. The framework has standard pose/gestures, but it is possible to assign new movements if the user cannot perform a certain movement. There is a method where the framework collects a certain number of frames (videos of the user performing the new movement) and this information is trained by the algorithm and associated with a certain action. Making it possible for the user to make any move he thinks best for his equipment commands.
- ***Stores and share actions over the internet (cloud):*** In this component, it is possible to save data stored locally by the ACDf in the cloud, such as new learning movements, so that it is possible for the user to access this data in different environments. For this, a Google Drive API is used, where with the user's data it is possible to save this data in your account. In this way, the user can perform the gesture training for their equipment at home and if there is KNX protocol equipment installation in their office and the framework installed, they can download this data so that they use the same movements in their office, without having to undergo new training.

Along with the brief explanation above about each module of the ACDf, it is important to introduce some concepts before presenting each module in more detail, such as the difference between the *functionalities*, *actions*, and *movements* of the devices.

The ***functionalities*** of the device are "functionalities" that the selected device allows for its interaction, for example, a led lamp has the possibility of having four types of functionalities: *on*, *off*, *varying* its lighting, and *changing* its colour.

About the ***action***, it is an activation of specific functions of that device in question, such as the action of turning the device *on* or *off*. Therefore, the same action can be applied to different devices, for example, turning on a lamp, turning on a TV, and turning on an air conditioner.

Related to the ***movement***-(s), it corresponds to the movements performed by any part of the body of the person using the framework to perform the specific action. For example, lift an arm - slide up, rotate your arm from right to left - slide to left, rotate your head to the left - rotate to the left, stretch your arm to the left /right and hold horizontally for *n* seconds (s) - lateral *n*.

Finally, an *action* can be associated with a single body movement, for example, the activate action can be associated with the *slide-up* movement or action can be associated with a group of movements, for example, *increase* lighting or *increase* the volume of the TV, can be done by concatenating two movements: *slide upside* 3 (3 seconds) to select the action "increase", and then *slide to the left* once or several times to select the action "increase" the lighting/sound (*slide* or *swipe* are similar terms in this context).

With these concepts in mind, the remainder of this chapter presents in detail the modules to create the ACDf.

3.2 RECOGNIZE USER

For the framework to present its different functionalities for each user, it's necessary to recognize the user, in a way to associate the user *movements* to *actions* to the device *functionalities.*

For this, LBPH (Local Binary Pattern Histogram) face recognition method is used (Dinalankara, 2017), and the framework is trained to know specifically which user is performing the movement. The above allows different users to make the same *movements* (gesture) but for different *actions* making it possible that the framework adapts to each user.

For user recognition, the method is trained using the region of interest (RoI) extracted from a face detector. The face detector used is the Haar-cascade (see section 2.4.3) following the stages proposed by Dinalankara (2017) (see also Fig. 13). Where, initially, a collection of *x* number of images is made, and the face detector is applied to use the RoI of each one of these images. After applying the face detector, the features of this RoI are removed and the training of these resources is carried out.

With the training completed, a model (dataset) is saved in an XML file where this file is then used to predict the user's identity. For user prediction, the input image is being analyzed by the framework and compared with data the trained model.

Figure 13: Steps to train face recognizer.

The collection of images (RoIs) is associated with an *#id* that will be assigned by the user. The framework, when detecting a new user - requested to add a user, will capture approximately 7 seconds (s), i.e., x = 200 RoIs of the user face (detection), and storing these images with their respective *id* and username (Fig. 14).

In our case, the implementation used OpenCV (OpenCV, 2021) and the FaceRecognizer class, which allows the creation of XML files to store resources extracted from data sets. Here the face recognizer used has ***cv2.face.createLBPHFaceRecognizer()*** which is a type of visual descriptor used for classification in CV, using local binary pattern histograms and it accommodates the following parameters:

Figure 14: Example of image storage to perform training.

- ***Radius***: The radius used for building the Circular Local Binary Pattern. The greater the radius, the smoother the image but more spatial information you can get;

- ***Neighbors***: The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use 8 sample points. Keep in mind: the more sample points you include, the higher the computational cost;
- ***Grid_x***: The number of cells in the horizontal direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector;
- ***Grid_y***: The number of cells in the vertical direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector;
- ***Threshold***: The threshold applied in the prediction. If the distance to the nearest neighbour is larger than the threshold, this method returns -1.

Recognition objects are created, and images are imported, resized, converted to *numpy* arrays, and stored in a vector. Image *id* is obtained by dividing the file name and stored in another vector and using ***FaceRecognizer.train (NumpyImage, id)***, all objects are trained. The model is then saved as an XML file using ***FaceRecognizer.save (FileName)***.

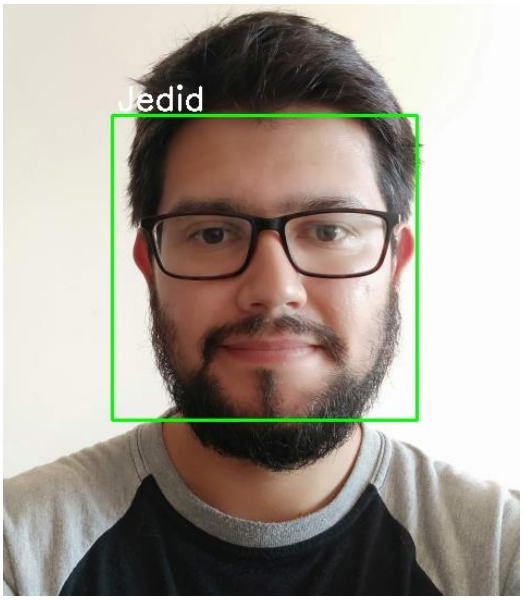

Figure 15: Applying face recognition with the trained model.

After collected images and performed the training of resources and saved in the file, it is possible to carry out the recognition of that user (Fig. 15). The same technique used for image capture is used initially and for face detected a prediction is made using ***FaceRecognizer.predict()*** which returns the class *id*. Finally, text file names with *id* are

used to have the username, if the forecast does not recognize the image it will be presented as unknown.

With the possibility of detecting the user's face, it is possible to make the framework even more dynamic, such as adding more than one camera in the environment where the framework is being applied. With this condition, if we did not have face recognition it would not be possible because we would have problems detecting gestures/poses with the application of pose estimation. After all, pose estimation does not detect if the person is in front, back, or sideways. If the algorithm would be applied this way the response matrix would not be with the complete information and thus we would not have the detection of the gesture/pose or we would have an incorrect detection.

In this way, using face recognition it is then possible to apply pose estimation only when this user is detected and recognized, thus making sure that the user is in a position considered adequate for the framework to act. With this it is also possible for the user to perform the movements without needing to be in a specific place, in front of a certain camera, considering that the framework has feedback via audio to inform the user of the changes (see section 3.6).

For a better understanding, Fig. 16 shows the example of a distribution of 4 cameras by a room of a residence where these cameras are all connected to the framework and then a loop is created between these cameras to check in which it is possible to have detection of the face and thus apply the pose estimation.

Figure 16: Example of a 4-camera environment for detecting gestures/poses.

Considering *Camera 01* being the one assigned next to the sofa, *Camera 02* being the one next to the entrance, *Camera 03* being the one next to the kitchen, and *Camera 04* being the one next to the second door, then we have a large area detection and user movement.

As shown in Fig. 17, it is possible to verify that the capture by different angles is viable because only when the face is detected is the pose estimation applied and thus verified which gesture/pose the user performed. In all of them, the presence of a person is detected, but only in one that the rest is detected, and the user is recognized and in this camera the pose estimation was applied, thus making it possible for the user not to depend on just one place to interact with automation equipment.

Figure 17: Example detection with multiple cameras, from left to right: Camera 01 to Camera 04.

In the next section, the process for recognizing gestures/poses with pose estimation will be presented and explained.

3.3 RECOGNITION OF HUMAN MOVEMENTS

To detect the user's movements, the GluonCV tool (see section 2.4) was initially used. It has a kit of Deep Learning tools for use in Computer Vision. As mentioned, in this kit, you can find classification algorithms, object detection, pose estimation, among others and all are possible to be used together with OpenCV. The tool has pre-trained and tested models, being ready for use in different projects.

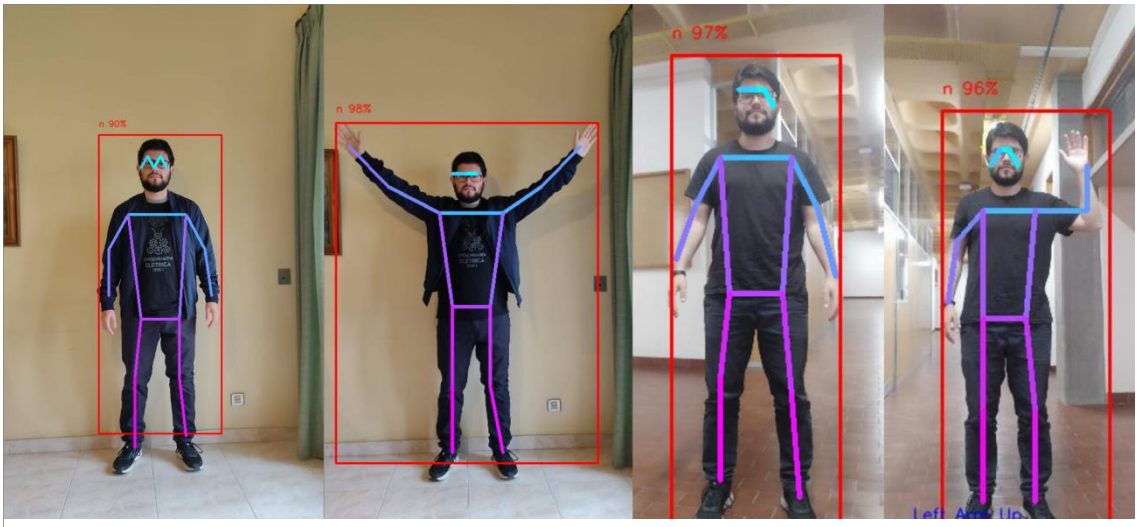

Figure 18: The GluonCV tool application to detect movement.

We use the Simple Pose for pose estimation, with a ResNet18 backbone (Xiao *et al.*, 2018) that is trained with an input size image of 128 × 96 px. This choice is made so that it is possible to obtain a satisfactory balance between precision and speed since it is intended that the framework works in any standard HD camera, cell phone, and laptop computer is used.

The feedback from this tool is a vector with the coordinates of the 17 keypoints that each point represents the position of the joints in the person, as shown in Fig. 18. In this figure, the detection of people with the respective percentage of average precision is presented and the joints in these images are connected by lines so that the user and the detection of the joints are better demonstrated.

With this response from the GluonCV tool, the data were processed so that it is possible to perform the detection of certain movements that will be used in the framework. For example, with the image received by the camera, it is verified if the keypoint of the

wrist is above the key point related to the shoulder and thus it is possible to check if the user has the arm raised or not, this for both arms, having taking into account that the keypoint of the left wrist with the left shoulder is always analyzed and the right shoulder with the right wrist in the same way (Fig. 18, on the right).

This initial approach where all the movements were "*hard-coded*", has previously changed in a way each user can teach to the system his/her movements. For that a new library, MediaPipe (MediaPipe, 2021), was used that allows the user to teach the movements, being the interface adapted to each different user and his preferences and needs. In addition, this library is much faster than GluonCV, and it is prepared for edge devices. The use of MediaPipe and the way the movements are learned is presented in section 3.7.

The next section presents the process for communicating with home automation devices that have the KNX protocol.

3.4 COMMUNICATION WITH DEVICES

Communication with the devices was done using the KNX protocol (see section 2.5). A KNX installation can be done in different ways, depending on the devices being used, but a simple installation consists of a KNX power supply (DC 30V), a sensor of any type, such as a switch, an actuator, and a power cable single twisted pair (TP) bus (shown in Fig. 19 on the left). For this project, a KNXnet/IP Tunneling was used for the interface with the system.

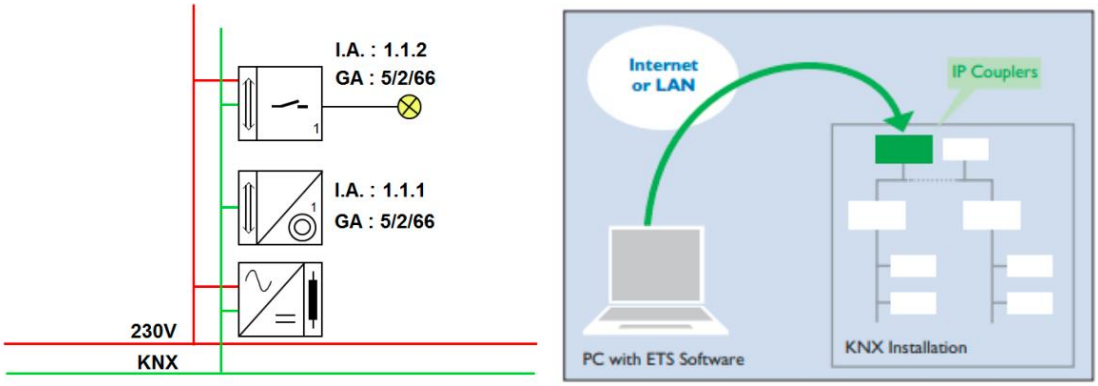

Figure 19: Example of a simple connection of KNX devices (adapted from KNX, (2020)).

In installations using KNX, each device has an individual address (IA) that must be unique for each piece of equipment, so that it is the identification of this device within the installation. For communication between devices, it is carried out through group addresses (GA). Group addresses can be assigned to devices as needed, making it possible to communicate between multiple devices at the same time. But, although actuators can hear multiple group addresses, sensors can send only one group address per telegram.

In this project, an open-source library, XKNX (2020) was used, making communication with devices simpler. This library was the whole framework that was coded in Python to simplify the association between codes. It is possible to make an IP connection with the KNX devices, specifying the IP address of the KNX IP interface that is connected to the system and sending the action that the KNX device (s) must perform.

To use the library for KNX communication, proceed as follows: (a) for initialization, (a.1) inclusion of the IP that is associated with the KNX IP interface, and (a.2) inclusion of individual addresses (s) or group (s) configured by the KNX software for the device (s) connected to this interface. (b) After initialization, (b.1) choose the device or group of devices, (b.2) send information to act - functionality according to the movement performed, and (b.3) receive recognition of the functionality enabled.

In the next section, the association of pre-defined movements with communication equipment via KNX protocol is presented.

3.5 ASSOCIATION OF MOVEMENTS WITH KNX DEVICES

The human movements used for the association with the devices have been predefined as demonstrated proposed by Palsa *et al*. (2020) and Hernández *et al*. (2019), which are a set of movements that are already used for virtual environments, and these movements served as a basis for the movements implemented in this project. Also as mentioned in section 3.3 they were "*hard-coded*". The movements used for the initial prototype are shown in Fig. 20.

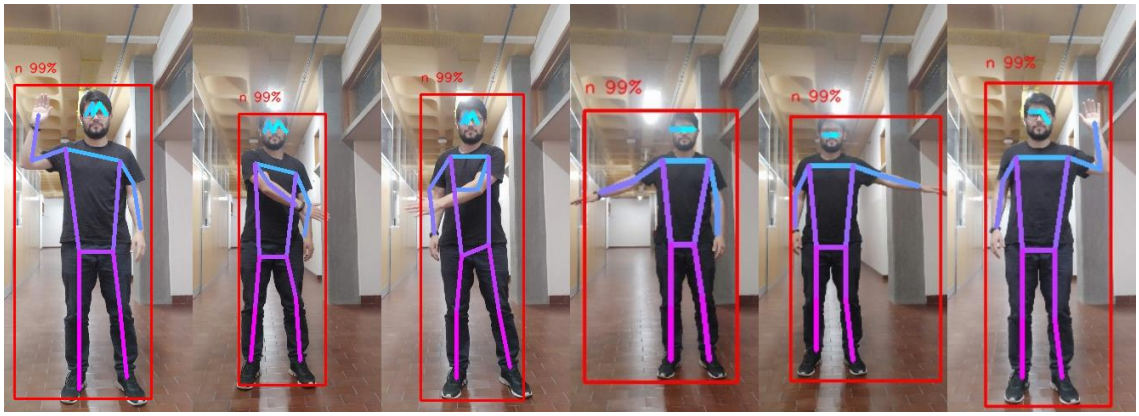

Figure 20: Example of movements used in the framework, from left to right: Right Arm Up; Right Arm Slide; Left Arm Slide; Right Arm Side; Left Arm Side; Left Arm Up.

It is important to emphasize, at this point, that the number of movements determined was made after studying the file that is exported by the ETS5 software after configuring the KNX installation.

The file exports information such as device type, device name, KNX address, and features (Table 3). With this information, the framework handles these data to assemble the navigation menu (see section 3.6). Thus, it was verified that these pre-determined movements are sufficient for all the necessary points in the framework, as it has a navigation menu, it is possible to determine movements for each situation and use the same movement for different actions, depending on what moment that is being used.

For example, the *Right-Arm-Up* is used to enter a selection in the menu, choosing which equipment you want to access the functionalities but is also used to turn on or off a lamp functionality, or an on/off of an air conditioner. *Left-Arm-Slide* and *Right-Arm-Slide* are only intended for menu navigation, between the types of equipment and its functionalities.

The lateral movement of the left arm (*Left-Arm-Side*) is for decreasing values, for example, to decrease the temperature of a setpoint of an air conditioner. The lateral movement of the right arm (*Right-Arm-Side*) is for the increase of values and, finally, the *Left-Arm-Up* means that a scenario determined as a favourite by the user will be activated, where various actions will be made on the automation equipment that is associated with this movement, such as turning on lighting the room, turning on a TV and closing the room window blinds.

Table 3: Example data available in ETS5 software file

Types	Equipment	Functionalities	Address
Illumination	Room Lamp Kitchen Lamp	On/Off/Increase/Decrease	1/1/1 1/3/1
Power Outlets	Room Outlet Kitchen Outlet	On/Off	3/1/1 3/3/1
Air-Conditioning	Room AC	On/Off/Mode/SetPoint	2/1/1
Blinds	Room Blind Kitchen Blind	Up/Down	4/1/1 4/3/1

Using the Kitchen Lamp equipment in Table 3 to increase its luminosity, the following process should be done: (i) *next/previous* movement to get to the illumination type; (ii) *active* (right-arm-up) movement to access lighting type equipment; (iii) *next/previous* movement to navigate to kitchen lighting equipment; (iv) *activate* movement to access the functionalities of this equipment; (v) *increase* movement (right-arm-side) to increase the luminosity intensity of this equipment. Remembering that the framework has an audio response in its navigation menu as explained in more detail in section 3.6.

For the detection of each movement, comparisons were made between the keypoints returned from the GluonCV tool (see section 3.3). GluonCV returns a matrix with 17 keypoints where each keypoint corresponds to a body part, as shown in the following list: *nose; left-eye; right-eye; left-ear; right-ear; left-shoulder, right-shoulder; left-elbow; right-elbow; left-wrist; right-wrist; left-hip; right-hip; left-knee; right-knee; left-ankle; right-ankle.*

From this, comparisons were made for each movement:

- ***Right-Arm-Up***: Right wrist position is compared with right elbow and right shoulder position, if the wrist is above the right elbow and right shoulder then right arm raised detection is made;
- ***Right-Arm-Slide***: The right wrist position is compared with the left hip position, along with the right elbow position with the right hip position. When the right wrist is after the left hip and the right elbow is after the right hip, slippage of the right arm is detected;

- ***Left-Arm-Slide***: The left wrist position is compared with the right hip position, along with the left elbow position with the left hip position. When the left wrist is after the right hip and the left elbow after the left hip, left arm slip is detected;
- ***Right-Arm-Side***: The position of the right wrist is compared to the position of the right shoulder and the right hip. When the position of the right wrist remains between the coordinates of the right shoulder and right hip, in height, and moves away, laterally, a lateral lift of the right arm is detected;
- ***Left-Arm-Side***: The position of the left wrist is compared to the position of the left shoulder and the left hip. When the position of the left wrist remains between the coordinates of the left shoulder and left hip, in height, and moves away, laterally, a lateral lift of the left arm is detected;
- ***Left-Arm-Up***: Left wrist position is compared with left elbow and left shoulder position, if the wrist is above the left elbow and left shoulder then left arm raised is detected.

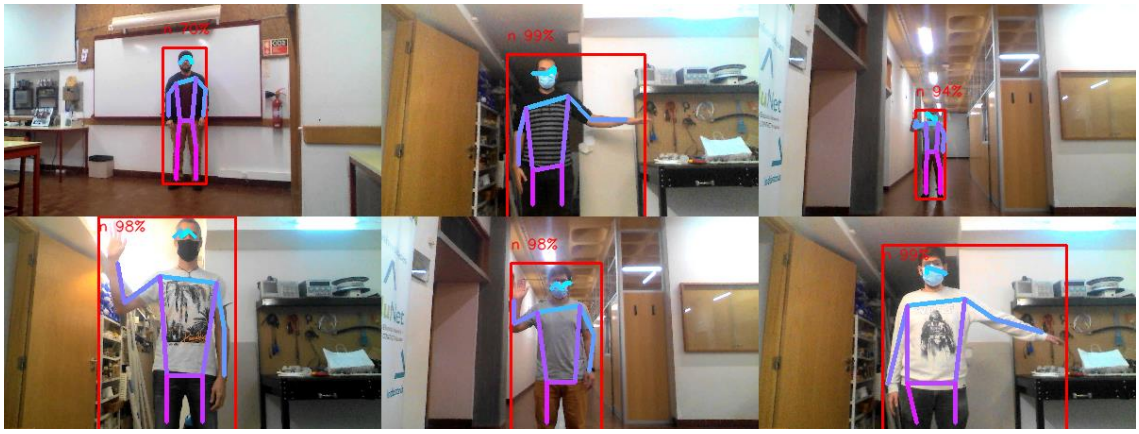

Figure 21: Application of pose estimation in different users in different scenarios.

Figure 21 shows different captures of the movements mentioned above, using different users and scenarios, varying the lighting, angle, and distance. These movements presented will be for the following actions: (i) ***start***, start interaction with devices, (ii) ***next***, and (iii) ***previous***, for when it is necessary to navigate between devices if there is more than one. ***Activate*** (iv) or ***deactivate*** (v) devices. If the equipment has the functionality to increase or decrease a certain parameter, the actions (vi) ***increase*** and (vii) ***decrease*** will be used.

Table 4:Correspondence between actions and movements. Movements marked with (*) are the same movement, if applied, makes the negation of the previous one.

Action	Movements
Start	static + arm-both + lateral + 3 (both arms laterally during 3s)
Next	swipe-right + arm-right + any + 0 (right arm move from right to left)
Previous	swipe-left + arm-left + any + 0 (right arm move from left to right)
Turn-on/Active* Turn-off/ Deactivate*	swipe-up + arm-right + any + 0 (raise right arm vertically)
Increase	swipe-up + arm-right + lateral + 0 (raise the right arm laterally)
Decrease	swipe-down + arm-left + lateral + 0 (raise the left arm laterally)
Favorite	swipe-up + arm-left + any + 0 (raise left arm vertically)

Table 4 shows the correspondence between actions and movements. The movements are coded as follows: type + direction + body part + position + duration. Specifically, (a) *type of movement*: static, theft, rotation, walking, etc.; (b) *direction of movement*: up, down, left and right; (c) *part of the body you are doing to move*: right arm, left arm, head, etc.; (d) *position*: lateral, frontal, any, and (e) *duration* (minimum) of the movement in *n* seconds.

In this case, actions with "*", meaning that they are complementary, the same movement performs both actions, making the negative of the action that is triggered. In the case of movements with duration "0", it means that there is no minimum time to make that movement.

In the case of ***favourite***, it is an action that performs a sequence of functionalities on different equipment that are considered favourites by the user. Using this action, all the equipment that is associated is "activated" performing the functionalities that are saved in the favourite action. Turn on the lamp, the TV and close the blinds, for example.

Again, it is important to stress that in the initial prototype of the system, all movements – actions were "*hard-coded*", this means that all users had to use the same movements to activate an action. In section 3.7 we will detail how to achieve these "*hard-coded*" default movements using the MediaPipe library and as well as how to allow each user to teach to the system the movement that he wants to associate with the action.

3.6 NAVIGATION BETWEEN DEVICES

This type of framework does not need a specific physical location in the environment where it will be used, nor a screen that shows to the user if the movements-actions are correctly being done, in addition, the recognition of movements was also extended so that it can be used in navigation between devices, being possible to navigate between different types of equipment (laps, TV, AC, etc.) and activate its functionalities.

The types of equipment with KNX protocol have ETS5 initialization software where it is necessary for the initial configuration and with this interface, it is possible to generate a *.csv* document that presents the list of equipment and their IA and GA addresses. With this document, it is possible to load in the ACDf so that the devices are added, with their respective addresses.

With the listing of these devices and using the movements presented in section 3.5, or the new movements that the user introduces to the framework, see section 3.7, it is possible to switch between the different types of devices and activate their functionalities. Thus, so that the user would not need to be in front of a graphical interface to check which device he is browsing on, the framework presents an auditory response where, as the equipment alternates, i.e., the user receives an audio response informing which equipment he is currently activating.

Figure 22 shows the navigation process for when the system has several devices in the same room. Once a user is detected and recognized in a room (almost instantaneously), the ACDf accesses the list of devices (available in the room). The user can now do the movements associated with the *Next* or *Previous* actions which will allow alternating between the devices, and the system informs the user with sound which device is selected at that instance. When the user is on the device he wanted to use, the *Active* action - movements will be made and with that, he will have access to the functionalities of that equipment.

To activate the functionalities, it works in the same way for the navigation between the equipment. i.e., the user, performing the *Next* and/or *Previous* actions-movements to navigate between the functionalities, receiving audio feedback for each functionality, and arriving at the desired functionality will perform the movement referring to that functionality.

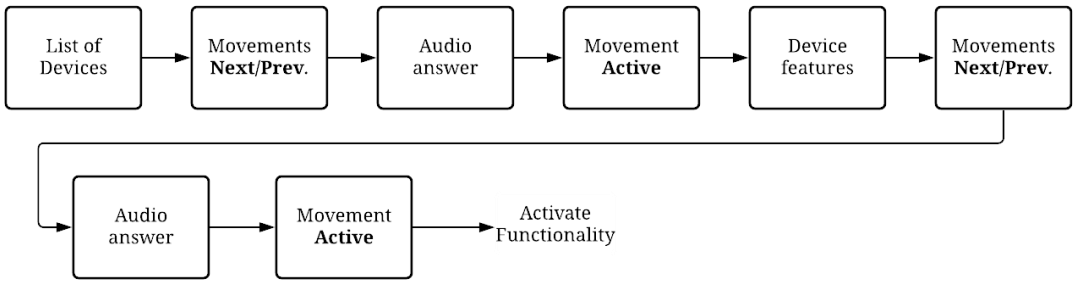

Figure 22: Navigation process between devices with auditory feedback.

To return the response to the user via audio, informing which action was performed or which equipment is being accessed to activate a functionality, a tool in Python is used. The *pyttsx3* (Pyttsx, 2021) is a text-to-speech library that is friendly to use and is compatible with Python 2 and 3.

The *pyttsx3* has some functions that are the most used for most systems:

- ***pyttsx3.init()***: Gets a reference to an engine instance that will use the given driver. If the requested driver is already in use by another engine instance, that engine is returned. Otherwise, a new engine is created;
- ***pyttsx3.getProperty()***: Gets the current value of an engine property;
- ***pyttsx3.setProperty()***: Queues a command to set an engine property. The new property value affects all utterances queued after this command;
- ***pyttsx3.say()***: Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue;
- ***pyttsx3.runAndWait()***: Blocks while processing all currently queued commands. Invokes callbacks for engine notifications appropriately. Returns when all commands queued before this call are emptied from the queue.

This library can be applied in numerous systems, Anas *et al.* (2020) and Yadav *et al.* (2021), for example, apply to the assistance of visually impaired people. Using sensors and computer vision, they propose two virtual assistant systems, making it possible to help using audio in different ways. The systems can inform the user if there is an obstacle in front of him, right or left through the sensors, inform what type of obstacle through object recognition. It is also possible to assist in document reading and currency detection.

Here, the use of the library is carried out as follows: (i) ***init()*** function to create the instance as soon as the framework is started; (ii) function ***say()*** which will be assigned the situation the user is in at the moment, what is the name of the equipment or what is the name of the functionality he is to select and (iii) the ***runAndWait()*** function to wait for all the information to be passed to the user and thus the framework flow can be followed.

3.7 LEARNING GESTURES

As already mentioned, (see section 3.3), for movement detection, the framework has standard conditions ("*hard-coded*") that were initially used with the GluonCV tool and later changed to the MediaPipe library.

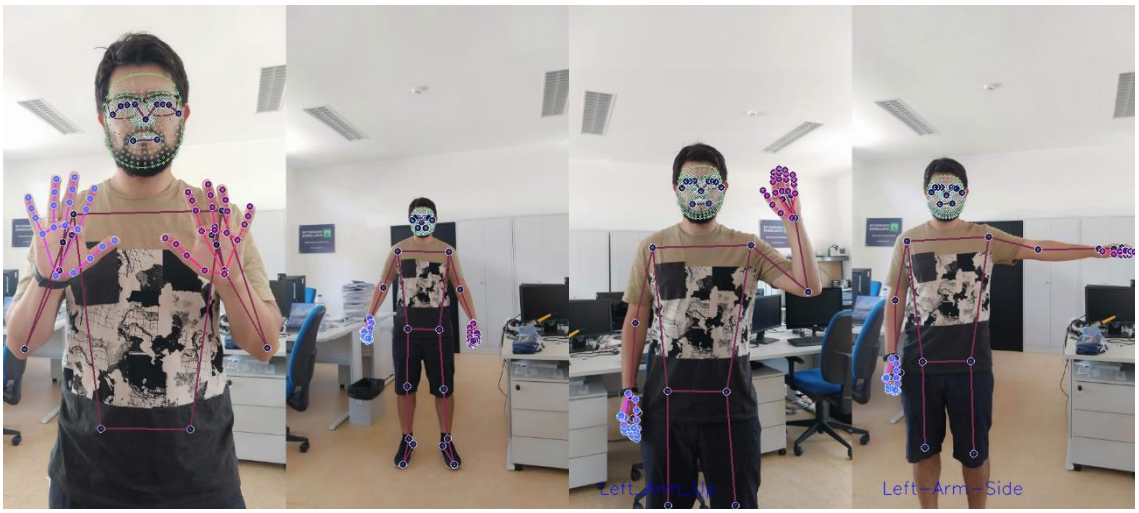

Figure 23: The MediaPipe tool application to detect movement.

In this section, it will be presented how the learning of new movements is performed, so that users can change them from the existing movements of the framework. For this, it is first necessary to explain the detection of movements (by default) with the new library.

In the same way it was used GluonCV, MediaPipe can be used, nevertheless, the last has a greater number of keypoints, so it is possible to perform conditions with more variables being analyzed.

Figure 23 shows the application of the MediaPipe library and the detection of movements with the new conditions for each movement. The Mediapipe library is being used with the Holistic function (Grishchenko *et al.*, 2020) where it is possible to simultaneously apply pose estimation (as shown in Fig. 2) hand detection (Fig. 24), and face detection (as shown in Fig. 4).

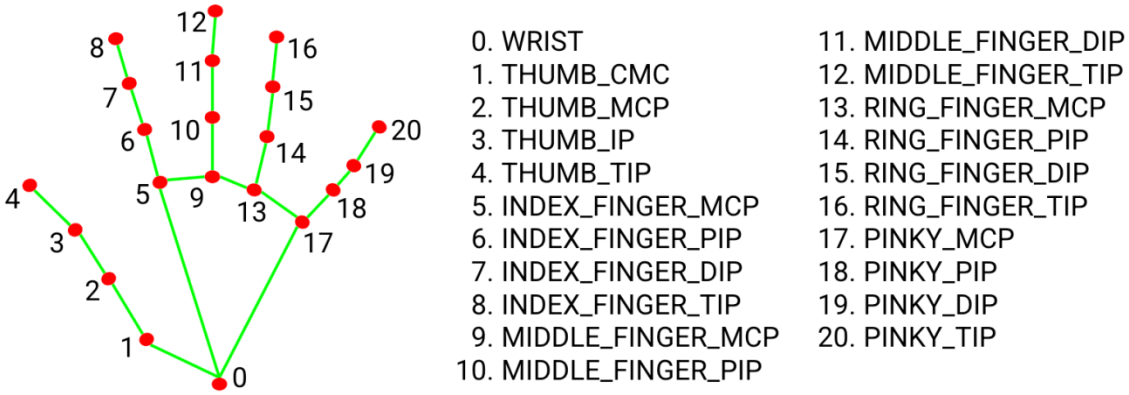

Figure 24: Hand landmarks (adapted from MediaPipe, (2021)).

The movements remain the same as presented in section 3.5 and are determined as follows:

- ***Right-Arm-Up***: Right wrist position together with right-hand points is compared with right elbow and right shoulder position if the wrist is above the right elbow and right shoulder then right arm raised detection is made;
- ***Right-Arm-Slide***: The right wrist position together with right-hand points is compared with the left hip position, along with the right elbow position with the right hip position. When the right wrist is after the left hip and the right elbow is after the right hip, slippage of the right arm is detected;
- ***Left-Arm-Slide***: The left wrist position together with left-hand points is compared with the right hip position, along with the left elbow position with the left hip position. When the left wrist is after the right hip and the left elbow after the left hip, left arm slip is detected;
- ***Right-Arm-Side***: The position of the right wrist is compared to the position of the right shoulder and the right hip. When the position of the right wrist remains between the coordinates of the right shoulder and right hip, in height, and moves away, laterally, a lateral lift of the right arm is detected;

- ***Left-Arm-Side***: The position of the left wrist is compared to the position of the left shoulder and the left hip. When the position of the left wrist remains between the coordinates of the left shoulder and left hip, in height, and moves away, laterally, a lateral lift of the left arm is detected;
- ***Left-Arm-Up***: Left wrist position together with left-hand points is compared with left elbow and left shoulder position, if the wrist is above the left elbow and left shoulder then left arm raised is detected.

With this library, it is possible to get better results compared to the GluonCV library, as presented in the test and results section (section 4), but like GluonCV the movements are still hard-coded. For the framework to be adaptive and learn new gestures, pose estimation training is used using a neural network architecture called Long Short-Term Memory (LSTM) (Sherstinsky, 2020).

Having the predefined movements or association of movements to each action, if we intend to allow the user to change these movements for more ergonomic ones, we have two possible solutions: (i) show the body landmarks to the user, with demonstration, and ask him which landmarks should be used for that intent movement. Of course, this solution is impracticable, once the user must have some "practical" skills to do this. (ii) the second solution is to ask the user to mimic which movement he whats to do, this was the solution adopted.

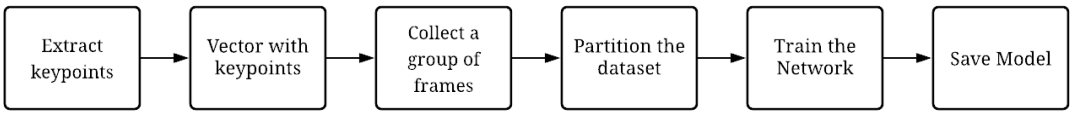

Figure 25: Steps for learning new gestures

To implement this solution the logical approach is to use a neural network (NN) and the user "teaches" the neural network the movements, by mimicking them. The results are stored and can be used in the future. By the results, we refer to the NN defined for this and the respective heights. For that several steps are necessary (Fig. 25):

(a) The first step is to extract all the landmarks and keypoints from the pose, face, left-hand and right-hand using the MediaPipe library (Lugaresi *et al.*, 2019), once we do not know if the user will use a "body" movement or a hand movement or a combination of several movements, including facial movements.

In the case of a pose, we have 33 landmarks, that go from *nose* to *right_foot_index*. For each of those landmarks, we have 4 values, the coordinates *x*, *y*, and *z*, where *z* determine whether parts of the user's body are in front or behind the users' hips. The last value is the *FrameLikelihood score*, where for each landmark, is a measure that indicates the probability that the landmark is within the image frame. The score has a range of 0.0 to 1.0, where 1.0 indicates high confidence. For the face, each face is represented as a list of 468 face landmarks, and each landmark is composed of *x*, *y*, and *z*. Where *x* and *y* are normalized to [0.0, 1.0] by the image width and height respectively. *z* represents the landmark depth with the depth at the centre of the head being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of *z* uses roughly the same scale as *x*. The hands (right and left) have 21 keypoints, 3D hand-knuckle coordinates inside the detected hand regions, again we have three values, *x*, *y*, and *z*. *x* and *y* are normalized to [0.0, 1.0] by the image width and height respectively. *z* represents the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of *z* uses roughly the same scale as *x*. In summary, we have a total of $tkp = 33 \times 4 + 468 \times 3 + 21 \times 3 + 21 \times 3 = 1662$ landmarks/keypoints.

(b) These landmarks/keypoints after-acquired are flattened to a single vector, that will be used later to feed the network. It is important to stress at this point that every time a landmark or a keypoint is not detected their values are put to 0.

(c) The following step is to collect a group of frames (movies) representing each movement, to teach the network the different movements (Fig. 26). In our case we are not interested in the row images/"movies", but the MediaPipe landmarks/keypoints.

We must acquire *na* movements, and for these movements, we have to acquire *ns* sequences for that movement (more sequences, more data, in principle better results, in the end, the counterpart it that can be tedious for the user), and the length of each sequence, in frames, *ls*. This means that we have for each action $nfa = ns \times ls$ (number of) frames. In our case $na = 6$, $ns = 10$ and $ls = 30$.

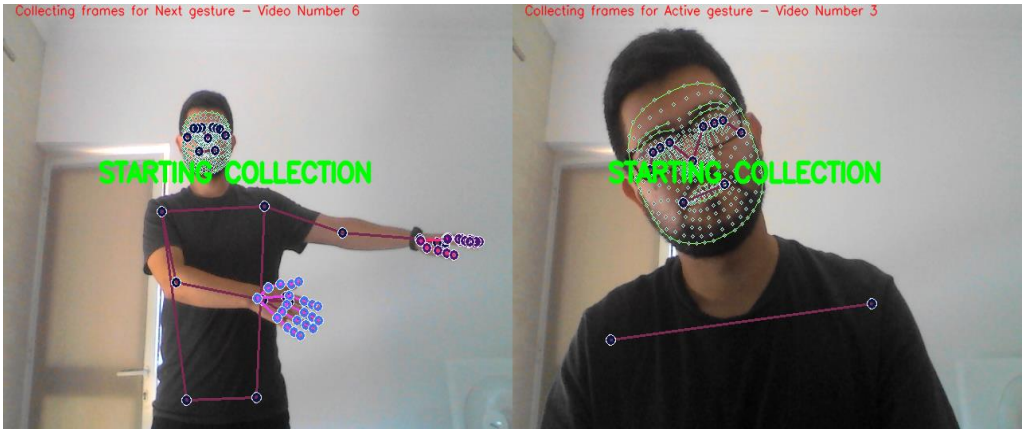

Figure 26: Examples collect frames to new gestures.

(d) Now having the dataset of the movements that we pretend to use, the next step is to partition the data in two: (a) *training data* and (a) *testing data*, giving them labels. It is important to remember that we had 6 initial movements/categories (Right Arm Up; Right Arm Slide; Left Arm Slide; Right Arm Side; Left Arm Side; Left Arm Up) that now are being converted to new movements, that we will call "action 1" to "action 6" which will do the same functionality, now with a different movement associated.

The first step (d.1) since we have categorical data ("action #") we have to convert this categorical data to numerical data, this is done by *one-hot encoding* (e.g., Rodríguez *et al.*, 2018). The next step (d.2) is to create an array with all the data (so far our landmarks/keypoint were stored in the disk or the memory, using a tree structure representing each movement). At this point, we have the condition too (d.3) partition the data, in 95% for training, 5% for testing (in reality validation). The reason for these values is that our dataset is very small, with bigger datasets the partition should be around 60% for training e 40% for testing.

(e) The following step is to train the network. Our network has 6 layers, being the input data the pair (*ls, tkp).* From the 6 layers, three are LSTM layers (Sherstinsky, 2020), which allows a temporal component to learning the movements. The first layer was 64 neurons/units the second 128 and the third 64, the activation function is a Rectified Linear Unit (RELU) (Chollet, 2017). The three initial layers are followed by 3 full-connected layers, with 64, 32, and the last layer with 6 neurons. The number of neurons in this layer is equal to the number of movements we want to learn, in our case 6. The two initial layers the

activation function is a RELU, the final layer is a Softmax (Chollet, 2017). The three 3 full-connected layers allow us to classify our movements, and the Softmax gives us the probability of the movement, it has a value between [0.0, 1.0] (being the sum of all results from the last layer be 1.0).

The reason to use LSTM layers is that they are a type of recurrent neural network capable of learning order dependence in sequence prediction problems, i.e., is a recurrent neural network in deep learning that has been specifically developed for the use of handling sequential prediction problems, more details can be found for instance in Sherstinsky (2020), or a brief explanation can be seen in Knight (2021). Finally, the reason to choose 6 layers and this number of neurons for our network, is to reproduce similar networks that do similar tasks and presents good results.

To train the network we use TensorFlow and Keras (Grattarola *et al.*, 2020) using an *Adam* optimizer (Chollet, 2017; Brownlee, 2017), *categorical crossentropy* (Chollet, 2017; Peltarion, 2021a) for the loss and for the metrics it was used *categorical accuracy* (Chollet, 2017; Peltarion, 2021b). The model was trained with 1000 epochs.

(f) Now it is possible to save the model and the heights for the movements. The final step is to load them (model and heights) every time our application starts and do the inference to know which movement is being executed by the user (instead of the predefined ones).

3.8 STORE AND SHARE ACTIONS THE CLOUD

In this section, the applicability of edge computing will be presented with a simple model (Calheiros, 2020; Chen *et al.*, 2019). Thus, the data that the framework has locally, such as information for the recognition of a specific user, the new gestures/poses stored by this user, and KNX addresses of the automation devices that are being processed by a Raspberry, for example, can be stored in the cloud and another environment can be accessed and connected to a new on-premises system. For example, if the same user has an installation of automation equipment in his home and also in his workplace, he can use the same gestures both at home and at work without having to perform the movement training again.

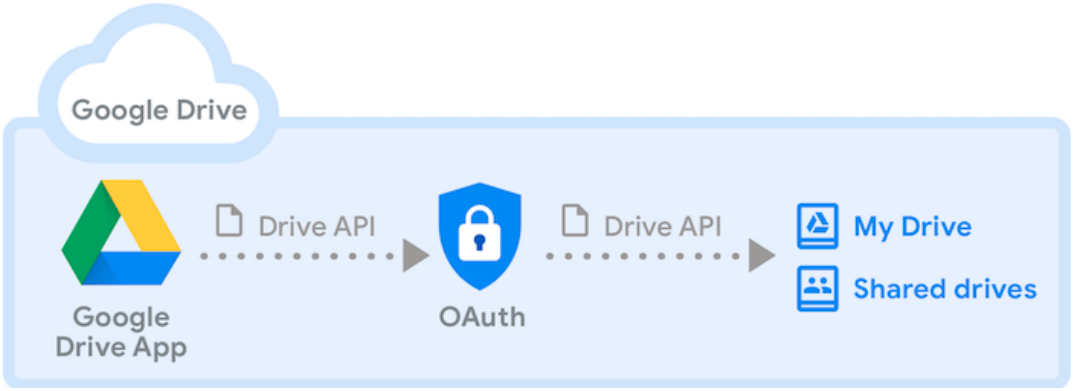

Figure 27: Google Drive API Relationship Diagram.

For this, an Application Programming Interface (API) provided by Google Drive (Google Drive, 2021) will be used, which is compatible with the use in Python, and this way it is possible to upload and download files in an account that the user has. The process is carried out in a simple way (Fig. 27) where an authorization (token) and an account on google drive are required so that it is possible to have the release to access and perform actions in the destination folder.

With the API it is possible to perform several features, including:

- Download and upload different types of files;
- Search for files and folders stored in Google Drive;
- Create complex search queries that return any of the file's metadata fields in the Files feature;
- Allows users to share files, folders, and drives to collaborate on content.

The features of this tool are used, for example, by Harish *et al*., (2019) who perform face recognition in a classroom, to store information about students, and this data is stored in the cloud through the API where it is possible then access this content anywhere you want. For the download, in another place that the user wants to use this data, the framework is informed that new models are to be downloaded and by informing their cloud credentials the framework will access and allocate the files in the correct locations in the directory so that ACDf can work correctly with the updated data.

4 Tests and Results

ABSTRACT

Using the methods described in the dissertation, we present some tests and results to demonstrate the performance of the proposed framework.

4.1 INTRODUCTION

This section presents the tests and results obtained with the methods mentioned in this dissertation. The tests are divided into two steps, the first step (a) was an initial prototype, for the validation of the method, using the GluonCV library and hard-code movements to interact with the KNX installation. In the second step (b) of the tests, for the final prototype, the same tests of the first stage were performed, but with the new MediaPipe library, and also performed tests with the final version of the framework, being possible to perform face recognition, navigation between KNX installation equipment, activation of favorite mode, feedback via audio from the framework and also learning new gestures.

4.2 INITIAL PROTOTYPE

The initial prototype (Santos *et al.*, 2021), was tested as proof of concept. The tests were done by 6 different users in 3 different environments: laboratory, corridors, and office. All with a different set of natural and artificial illumination and in different buildings. In all tests was used HD Webcams placed around 1 meter above the floor level, and i5 Laptops, 5-8th generation Intel Processors, with SSD and 8 Gbytes RAM. The tested KNX installation consists of a power supply 320mA RMD, a universal dim actuator 1gang, 50-500W/VA RMD, and a KNX IP router RMD, together with the ETS5 software for programming the KNX installation (Figure 28 top row).

The 6 users were divided into 3 groups of 2, each group test one of the three environments, doing 5 repetitions for each of the actions presented in Table 4. The test scenarios consist in the users being face forward to the camera, at a position approximately 0º (degrees) - frontal to the camera, and at distances of 1, 2, 3, 4, and 5m (meters) do the respective movements to activate the mentioned actions (Table 4). Them the users change to a position around ± 25º (degrees) to the right and the left and repeat the same distances. All users before performing the test were briefed with the movements-actions but were not induced to do it in a certain way, it was left to the user to perform the gesture as he considered the most correct way.

Figure 28: Top row, detailed image of the KNX IP router (left) and devices (right). Bottom row, detection of a swipe-up (left), and the activation of the KNX device – lamp (right).

The total number of test samples was 2.250, comprises 3 environments × 2 users per environment × 5 repetitions per action × 5 actions × 3 positions × 5 distances. This means that each of the five actions has acquired 450 test samples. Table 5 shows the percentage of correct actions achieved by the framework.

Table 5: Table with the percentages of correct answers obtained during the tests.

Action	**Accuracy (%)**
Next	76,7
Previous	80,0
Turn-on/Turn-off	83,3
Increase	66,7
Decrease	60,0

In terms of results in the phase (initial prototype), all tests present satisfactory results, with overall average correct actions of 73.3%. No major flaws were detected in the

framework. The results had better accuracy when the user has at distances of 2m because the pose estimator can detect the person's full body and presented the joints with good precision, and consequently, the movements to action classification were done with more accuracy. No error was detected when an action is well defined and communicated to the KNX devices.

As expected, the very few tests done with very little or no light (less than 1% of the overall tests) fail, the remaining of the fails were due to gestures done very quickly and gestures where the user is not doing the intended gesture for the intended action.

4.3 FINAL PROTOTYPE

As presented in the dissertation, the tool for using pose estimation was changed. In the initial prototype, GluonCV was used, but, as the prototype was developed, the library was changed to the final prototype, using the MediaPipe library. With this in mind, the tests with the final prototype are divided as follows: (a) The same test performed on the initial prototype, but with the new library so that it would be possible to detect/prove its improvements over the previous library; (b) Teach new gestures/poses for the framework and perform the equipment interaction with these new movements and finally (c) a case study with a user.

The first test of the final prototype was carried out in the same way as described in the initial prototype, containing the same type of installation of KNX equipment, different types of environments, and the same number of users. And in this step, the action test has already been performed to activate a favorite scenario, in this case when this action is detected, all installation devices are activated/deactivated at the same time.

Figure 29 shows (top row) the installation of equipment with the KNX protocol used. Initially, the test for the activation of the installation lamp was carried out, and then four switches (represented as *S1, S2, S3, S4*) that the installation has were used, where they are represented with a light warning when they are activated. In this way, the tests were performed performing the interaction with the lamp and with the four switches that can represent different actuated loads in an installation in a home or office.

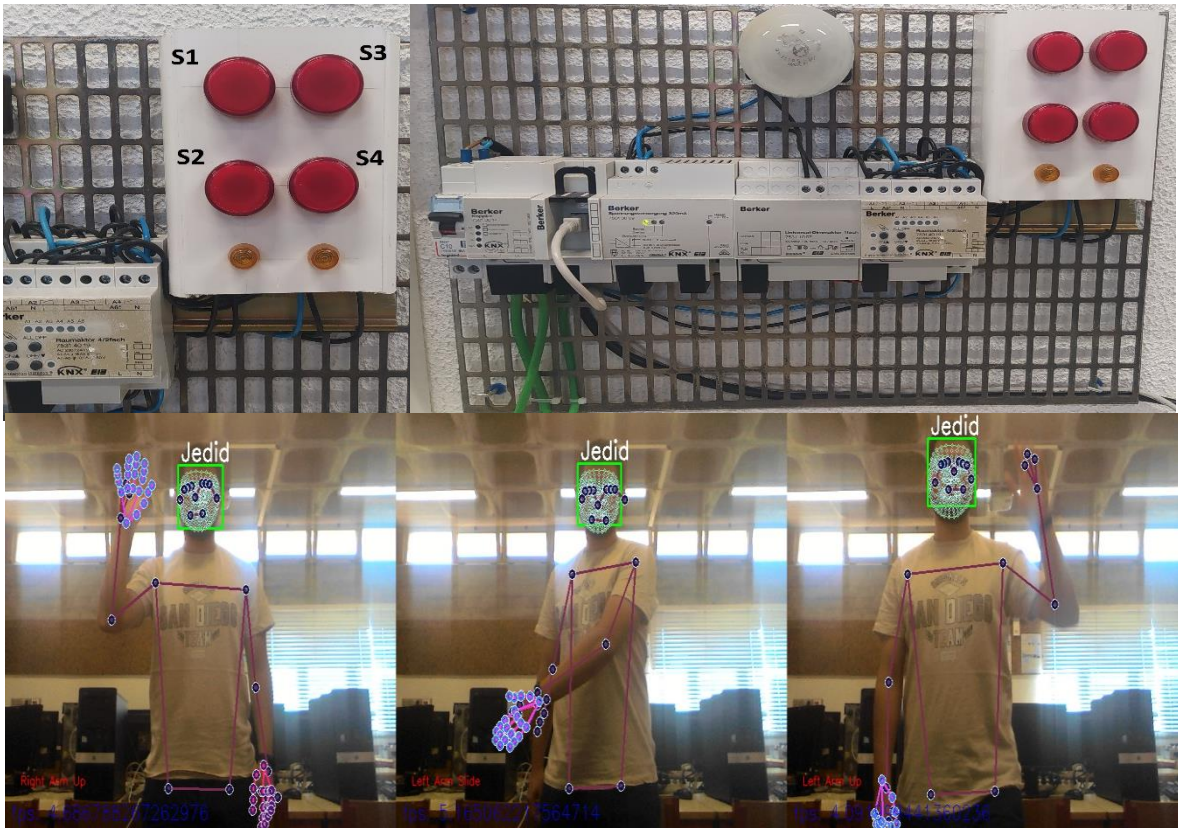

Figure 29: Top row, detailed image of the switches used (left) and and complete installation with KNX equipment (right). Bottom row, Detection of some movements during testing.

Table 6 shows the results obtained in this first stage of testing the final prototype. In terms of results in this first stage of the final prototype, all tests show good results, with an overall average of correct actions of 79.6%. No major flaws were detected in the framework. In the same way as in the initial prototype, when the user is at distances of up to 2m, the results obtained are more accurate.

Regarding communication with KNX devices, we didn't get any problem due to the change of the library, and, as in the initial prototype, the tests that were done with low light (or none) fail.

In comparison with the initial prototype tests, it was verified an improvement in the detection of the movements and mainly related to the speed of the pose estimation algorithm. Using the MediaPipe library we have frames per second (FPS) of *FPS*=5 so that the entire framework is working with all its other functionalities, if only the pose estimation was applied, its *FPS* would be higher. While in GluonCV we have an average of *FPS*=1.5 with the initial functionalities (initial prototype) together. This is

one of the biggest factors for changing the library as with a higher FPS we have a faster framework response, bringing benefits to the user.

Table 6: Table with the percentage of correct answers obtained during the tests of the first stage of the final prototype.

Action	**Accuracy (%)**
Next	80,0
Previous	78,8
Turn-on/Turn-off	89,5
Increase	70,0
Decrease	69,5
Favorite	90,0

In the second stage of testing the final prototype was to verify the functioning of the framework with the movements taught (see section 3.7) by the user. For this step, 6 users were used and divided into groups of 2 in 3 different scenarios (laboratory, home, and office). Each user assigned a new move (or the same move) for all actions (*Next, Previous, Active, Favorite, Decrease and Increase*). The choice of movements was determined by each user, being possible to perform movement only by analyzing a part of the body as only the face or movements above the waist and even movements with the whole body, if possible.

For training, it was only informed that it should be done up to a maximum distance of 2m, as the previous tests had better results up to this distance. Once the algorithm is trained with the new movements, the tests are then carried out in the same way as the previous ones, placing the camera in front of the user and performing the movements varying the distance between 1, 2, 3, 4, and 5m and then the user shifting his position by +- 25º (degrees) to the right and left. The installation of KNX equipment is the same as presented in the first test of the final prototype (Figure 29 top row).

Some samples of keypoint collections are shown in Figure 30 with different movements for the different types of action, which in total were collected 10,800 frames, being 6 users x 6 actions x 10 videos x 30 frames (each video).

Figure 30: Examples of samples collected for training new movements

Table 7 shows the percentage of correct answers performed with the movements trained by the users. In terms of results in this second stage of the final prototype, the tests show lower results (hit average of 61.7%) than the tests performed in the first stage or the initial prototype, but this was expected because it is now being treated as a data training and because no longer be analyzed static images, but a sequence of frames. But this does not make the results unsatisfactory, because, in the same way, that the communication with the pre-defined movements (hard-code) was carried out, it now happens with the movements trained by the user, but now more precision is needed to carry out the movements since it is predictions being made in sequence of frames.

Pieces of training were also carried out changing to movements with the face and it was also verified that without more specific treatment to this situation it becomes more complicated to apply as it is necessary to be very close to the camera to detect the action since the face mash of the holistic applies many keypoints and as the distance increases it becomes more difficult to differentiate if a certain movement has been performed. As the distance increased, differences were also detected between the

predictions, since as the distance increased, the predictions became more difficult to be correct.

Table 7: Table with the percentage of correct answers obtained during the tests of the second stage of the final prototype

Action	**Accuracy (%)**
Next	60,0
Previous	62,5
Turn-on/Turn-off	64,0
Increase	65,2
Decrease	62,0
Favorite	56,6

In the last stage of testing the ACDf was performed with a user doing a case study in a home. Where the room was used as the environment (Figure 31) and 4 cameras were installed for detection, with Camera 01 placed next to the television with an approximate height of 50cm, Camera 02 allocated near the sofa with an approximate height of 80cm, Camera 03 allocated in the office with an approximate height of 70cm and Camera 04 allocated near the window with an approximate height of 80cm.

Figure 31: Case study environment

In Figure 32, the visualization of each of the cameras used is presented and these cameras are all integrated into the ACDf via IP communication.

For this case study, the user moves in this environment, firstly carrying out the framework's hard-code actions in different places in the environment, such as a sofa, dining table, in front of the office, and the middle of the room. Afterwards, the training of new movements is carried out and the movement through the environment is carried out again, with the movements trained for the ACDf being carried out. And finally, the files are uploaded to the cloud with the user's registered account.

Figure 32: Visualization of each of the cameras. Top row, Camera 01 (left) and Camera 02 (right). Bottom row, Camera 03 (left) and Camera 04 (right).

In this step, all the features of the framework, interaction with automation equipment, activation, and deactivation are tested. Verification of the feedback via audio, which for this case is heard through the laptop that is being applied the ACDf.

Table 8 shows the percentages of correct answers in this case study, considering all the situations mentioned for this stage of testing for these data.

Table 8: Table with the percentage of correct answers obtained during the tests of the third stage of the final prototype

Action	Accuracy (%)
Next	62,0
Previous	64,5
Turn-on/Turn-off	63,0
Increase	64,5
Decrease	62,0
Favorite	58,5

The results presented in this third stage of testing the final prototype are satisfactory, with an average of 62,4%. In Figure 33 some images detected by the framework during the tests of this step are presented. It is possible to verify that ACDf can work with different cameras since it performs face detection and from this, it performs gesture recognition, but it is always necessary that the user is viewing the camera so that it is possible to interact with the equipment.

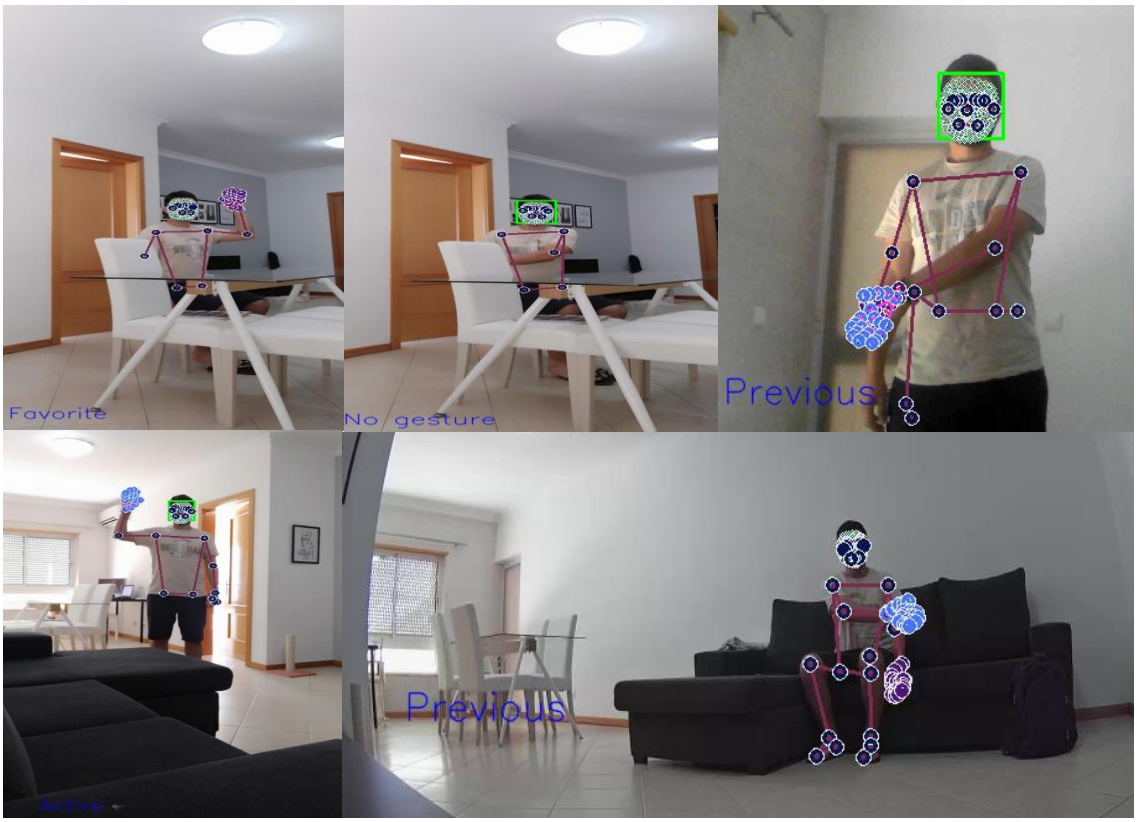

Figure 33: Examples of ACDf detection for the case study

With the tests, it was also verified that the ACDf can perform the detection with the user sitting on the sofa, for example, if the movement used is a movement with the arms, since trying to perform a movement, for example with the face, can become more difficult to detect because of the distance the camera is. It is also possible to analyze that when the user was at the dining table, depending on how his movement was made, it was not detected since the chairs could be in front of the user and thus covering some part of the body necessary for the detection of movement.

4.4 DISCUSSION

The obtained results from our tests were in line with what was expected based on what we had seen from intermediate tests during the development of the framework.

The first thing we can notice is that you didn't get exceptional results, but the implemented framework works. Pose/gestures are detected, communication with automation equipment is carried out, we receive responses via audio from the framework, navigation between equipment is carried out from its initial configuration, new gestures are learned and detected and are also stored in the cloud.

With the initial tests (initial prototype) it was possible to validate the proof of concept and continue the development of the framework for the other goals. During development, another library was discovered, called MediaPipe, where we detected a certain improvement about GluonCV that in the following tests (first stage of the final prototype) it is better functioning for the situation presented here was validated. With MediaPipe we obtained better precision results, due to more keypoints for detecting movements, while GluonCV works with n=17 coordinates, MediaPipe works with n=33 for pose estimation and, in addition, performs face and face detection at the same time. right and left-hand detection, making it have n=1662 keypoints. In addition to accuracy due to the large number of keypoints we also improved the framework in terms of speed, making the ACDf response faster for the user.

In the final prototype, we performed tests for learning new gestures/poses. We use an LSTM neural network where it is possible to carry out information sequence training. We obtained results lower than the previous tests with the predetermined movements, but it was expected since we have more variables possible to the error than in the initial tests. As the training of new gestures is being carried out, by the user's choice, the

probability of error is greater if the user, for example, makes two very similar movements for two actions so that when performing one of the movements the prediction is for the other. We also did not use a large dataset to have more data for training as this can become tiring for the user.

However, even with results lower than the previous tests, the results are satisfactory because it is possible to train new gestures for actions and the framework is able, after training, to recognize these gestures and perform the actions in the KNX installation.

In the case study, carried out in the final prototype, satisfactory results were also obtained. It was carried out in an environment only, a room, but it was already possible to detect the benefits of ACDf such as the possibility of operating with different cameras that brings the benefit of carrying out activations of equipment directly, for example, from the sofa or the desk, without the need of having a physical post within the environment where this framework will be implemented. The answer via audio, which in this case was being transmitted by the laptop that was the functioning of the framework, worked correctly being possible to hear perfectly the actions and the equipment that were being communicated. The training of new gestures was also applicable in this test and was used for navigation and activation of equipment and the information was saved in the user's cloud.

Some movement detection failures were detected, for example, when some furniture such as a chair or table is in front and the movement you want to perform depends on parts of the body that are behind this furniture, however, this type of failure is expected. It also fails when the user is at a very large distance from the camera that wants to detect movement, as described in the other tests, so the situation of having more than one camera in the environment makes the user always close to some and thus reduce the number of detection failures.

Although the tests were not exceptional, it was possible to reach all the goals proposed in this dissertation and we are very satisfied with the test results, especially since the framework can still be improved in many points.

5 Conclusions and Future Work

ABSTRACT

The Human-Machine interaction is accompanied by great advantages for everyday life. The possibility of automation equipment interactions through computer vision follows this great development, bringing great benefits to users.

5.1 CONCLUSIONS

In this dissertation, we present a framework that performs the activation and interaction of automation equipment, with KNX protocol communication, through a pose estimation. The communication with the automation devices being carried out with an open-source library for KNX data transmission through IP communication and the pose estimation algorithm, initially provided by the GluonCV library and after the development of the dissertation being changed to the MediaPipe library.

Initially, we present the contextualization and an art study related to the constant evolution of adaptive environments that can help people in their daily tasks and the benefits this can bring to the user. From this, we carried out a proof of concept to validate the interaction of KNX communication methods through body movements. Through the detection of pre-determined movements, actions were carried out in the installation of the equipment using the KNX protocol (for example, raising the right arm and turning on a lamp). Tests were performed (see section 4.2) where we obtained an average accuracy of 73.3% for the detection of poses and activation of KNX devices. Obtaining satisfactory results, we validated the possibility of performing actions on devices that use the KNX protocol through predetermined body movements.

With the proof of concept established, using the initial methods presented in this dissertation (see sections 3.3, 3.4, and 3.5), we continued the development of the framework so that it would have more functionality and so that the person using it becomes more adaptable. With the continuation of research and development, we found a Computer Vision library called MediaPipe (until now the GluonCV library was being used (see section 2.4)) which achieved better results in terms of accuracy and speed compared to the library that was being used previously, in section 4.3 we present an average of 79.6% of correct answers, for the predetermined movements. An increase of 6.3% with the use of this new library and not only about precision, but the detection speed was also increased, in GluonCV we had an average of FPS=1.5 and with MediaPipe it was possible to reach FPS=5 making a better library for the application we want because the faster and more accurate the detection, but the more reliable the data are also transferred to the algorithm and the faster the actions are carried out in the equipment installations.

With the new library for pose estimation tested, developments continued to be carried out, firstly being possible to differentiate users (see section 3.2) so that different users

can access the framework and control the equipment with their movements. In section 3.6 we present a method to make it possible for the user to navigate between different devices, if the installation has more than one piece of equipment, using body movements. So that the user didn't need to have a physical point with a camera or graphic environment for the interaction, a response via audio was added, where, with the initial information added from the KNX installation, a menu is set up where the user receives the response from which equipment is selected and what action was taken. Making it possible for the user only with cameras distributed throughout the environment to perform actions on devices from different locations in the environment.

For the framework to be adaptive to the user's needs, a method was used (see section 3.7) so that the user can teach new movements to the framework, not requiring the predetermined movements, in this way a person with a handicap or an elderly person who is no longer able to perform certain movements may use movements that he/she considers easy for their daily life. For this teaching, an LSTM network was used, which is used for solutions that require training in sequences, which in our case is applied because that way we can train all the movements that the user uses. Finally, using the concept of Edge Computing, the data stored by the ACDf locally can be saved in the cloud, in a user account (see section 3.8) and this data can be downloaded to another environment that has the ACDf configured and a KNX installation. In this way, the user can, for example, teach new moves to the framework at home and then can add these same moves in his office (if he has a KNX installation) without having to teach the neural network again.

With these methods defined and developed, we performed tests, firstly, with the training of new movements by different users and performing the interaction/action of the equipment with these new movements, and verifying the behaviour of the ACDf for this scenario. The results obtained were satisfactory, we obtained an average accuracy of 61.7%, which were expected results since they are learned movements by the framework and no longer pre-determined movements. In these tests, because we are performing with a considerably small dataset, created by us and not adding the information to an already created dataset, depending on whether the user trains two very similar movements for different actions, the predictions end up becoming more difficult and thus possible if get too many false positives in these cases. For movements performed only with the face, a greater difficulty of the framework for predictions is also detected, since if the movement is performed too far from the camera that is acting,

it becomes difficult to correctly predict, requiring the user to stay very close to the camera and thus losing some of its mobility in the environment.

Finally, we carried out a case study (see section 4.3) where an environment in a house was used, with 4 distributed cameras, where the user performed all the features developed in this dissertation. Using predetermined moves, user recognition, teaching new moves, interacting with the installation of KNX equipment, and saving local data in the cloud. In these tests, we obtained satisfactory results, with an average of 62.4% of correct answers in detecting the movements. In a real environment, we detected situations that we were waiting for, such as the possibility of performing the movement sitting on the sofa or standing in the middle of the environment, also the fact that we now have other objects in the environment was also analyzed that depending on the position of the user, with furniture in front of you or part of your body, depending on the movement, will not be detected.

With all this in mind, we got great results with ACDf, making possible the interaction of two technologies, Computer Vision and KNX protocol for automation devices. So, the tests carried out showed great benefits for the use of this technology and it is still open to different improvements depending on which applicability will be necessary.

5.2 FUTURE WORK

The framework reached the proposed goals, but due to the great development of adaptive frameworks and their benefits, as we presented in this dissertation, the ACDf has some points where it can still be improved.

One of the main points we can mention for future work is the use of this framework for communication with other communication protocols for residential and/or industrial automation equipment, protocols such as Modbus, BACnet, CEBus, LonWorks, etc. Thus, making the framework even more comprehensive for automation equipment with different functionalities.

Not least, as mentioned in this dissertation for the training of new movements, it is being carried out with a considerably small dataset, so as not to become something tiring for the user, in this way, we obtained good results, but which can be improved if a larger dataset is used. In this way, we can consider for future work to add the data collected by the framework in a large dataset and not create a dataset from the

beginning, to obtain a result with greater success. But without forgetting that this system has the objective of bringing ease to its user, so we need to find a balance between improving results but not causing time-consuming and tiring processes for the user. It is also interesting to carry out developments and research for the training of new movements with the only face since in the tests we verified that for this situation, it was necessary to be very close to the camera. It is of great interest to find algorithms/tools that can deal with this problem and thus bring more precision to situations like this. Still related to the user's face, for future work, it is also possible to associate the recognition of the user's facial expressions and set up scenarios in the environment, for example, the user is happy and to play high-spirited music. Also associate these scenarios with external situations, for example, the user arrives at home during the day and through external access to an API can detect if the temperature is high or low and thus activate equipment inside the house, such as downloading or climbing blinds.

Finally, we also believe that it will be of great help if the ACDf also recognizes accidents within the environment as it works from cameras around the environment and if a person falls indoors, for example, perform detection and can trigger the related organs to support the user.

5.3 PUBLICATIONS

Santos J., Martins I., Rodrigues J.M. (2021) ***Framework for Controlling KNX Devices Based on Gestures***. In: Antona M., Stephanidis C. (eds) Universal Access in Human-Computer Interaction. Access to Media, Learning, and Assistive Environments. HCII 2021. Lecture Notes in Computer Science, vol 12769. Springer, Cham. https://doi.org/10.1007/978-3-030-78095-1_37

Santos J., Martins I., Rodrigues J.M. (2021) ***Adaptive body interface to control devices using KNX protocol***. Submitted to 27th Portuguese Conference on Pattern Recognition, 5 Nov., Evora, Portugal

Santos J., Martins I., Rodrigues J.M. (2021) ***Adaptive Control Devices Framework***. In preparation for 16th International Conference on Universal Access in Human-Computer Interaction, 26-01 Jul., Gotherburg, Sweden

References

Alpaydin, E. (2020). Introduction to machine learning. MIT press.

Anas Majid, P., Nikhitha, V., Smrithi, K. M., & Basheer, V. P. (2020). Machine Learning Based Mobility Aid for Blind People. Machine Learning, 7(1).

Antonello, R. (2018). Introdução a Visão Computacional com Python e OpenCV. Santa Catarina, 1a edição.

Artacho, B., Savakis, A. (2021). OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation. arXiv preprint arXiv:2103.10180v1.

Atalaya, O. C., Porras, A. Q., Santillan, D. A. (2020). Lighting control network based on KNX protocol, for the reduction of energy consumption. In Indonesian Journal of Electrical Engineering and Computer Science. Vol. 19, No. 3, (pp. 1186-1193).

Bah, S. M., & Ming, F. (2020). An improved face recognition algorithm and its application in attendance management system. Array, 5, 100014.

Baltanas, S., Sarmiento, J., Jimenez, J.(2020). A Face Recognition System for Assistive Robots. In Procs of the 3rd Int. Conf. on Applications of Intelligent Systems, Art. 29, (pp. 1-6).

Bazarevsky, Valentin., Grishchenko, Ivan. (2020). On-device, Real-time Body Pose Tracking with MediaPipe BlazePose. (online on: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html, last accessed 2021/22/06).

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). Blazeface: Sub-millisecond neural face detection on mobile gpus. arXiv preprint arXiv:1907.05047.

Bellotto, N., Carmona, M., Cosar, S.(2017). ENRICHME Integration of Ambient Intelligence and Robotics for AAL. In Wellbeing AI: From Machine Learning to Subjectivity Oriented Computing, Technical Report SS-17-08.

Brownlee, J. (2017) Gentle Introduction to the Adam Optimization Algorithm for Deep Learning(online on: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/, last accessed 2021/09/07).

Bulat, A., Kossaifi, J., Tzimiropoulos, G., Pantic, M. (2020). Toward fast and accurate human pose estimation via soft-gated skip connections. arXiv preprint arXiv:2002.11098v1.

Calheiros, R. N. (2020). Fog and Edge Computing: Challenges and Emerging Trends (Invited Talk). In 2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y. (2021). OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. In IEEE Trans. on PAMI, vol. 43, no. 1, (pp. 172-186).

Chen, J., & Ran, X. (2019). Deep Learning With Edge Computing: A Review. Proc. IEEE, 107(8), 1655-1674.

Chen, W., Jiang, Z., Guo, H., Ni, X.(2020). Fall Detection Based on Key Points of HumanSkeleton Using OpenPose. Ing: Symmetry, 12(5), 744.

Chikano, M., Tomiyasu, F., Awai, S., Hirai, Y., & Konno, T. (2020). Person Matching Technology using Gait Information of 2D Pose Estimation. In 2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech) (pp. 140-144). IEEE.

Chollet, F. (2017). Deep learning with Python. Manning Publications Co. ISBN:978-1-61729-443-3.

Crandall, J.W., Oudah, M., Tennom, et al.(2018). Cooperating with Machines. Nat Commun 9, 233.

Daher, M., Najjar, M., Diab, A., Khalil, M., Dib, A., Charpillet, F. (2017). Ambient Assistive Living System Using RGB-D Camera. In 4th Int. Conf. on Advances in Biomedical Engineering (ICABME), (pp. 1-4).

Dinalankara, L. (2017). Face detection & face recognition using open computer vision classifies. Project, Faculty of Science and Engineering, Plymouth University.

Feki, E., Kassab, K., Mami, A. (2019) Integration of the small board computers Raspberry PI in Home Automation based on KNX protocol. In IEEE 19th Mediterranean Microwave Symposium (MMS), (pp. 1-4).

Feng, Z., Xiatian, Z., Hanbin, D., Mao, Y., Ce, Z. (2019). Distribution-Aware Coordinate Representation for Human Pose Estimation. arXiv preprint arXiv:1910.06278v1.

Gams, M., Gu, I., Härmä, A. et al (2019). Artificial intelligence and ambient intelligence. In Journal of Ambient Intelligence and Smart Environments, 11(1): 71-86

GluonCV, State-of-the-art Deep Learning Algorithms in Computer Vision. (online on: https://cv.gluon.ai/, last accessed 2020/11/12).

Gonçalves, J.P.A. (2017). Protocolos de Automação Doméstica - Solução de Automação Residencial e Vigilância Baseada em Protocolo Z-Wave. Instituto Superior de Engenharia do Porto.

Google Drive. (2021). Introduction to Google Drive API. (online on https://developers.google.com/drive/api/v3/about-sdk, last accessed 2021/08/06).

Grattarola, D., & Alippi, C. (2020). Graph neural networks in tensorflow and keras with spektral. arXiv preprint arXiv:2006.12138.

Grishchenko, Ivan., Bazarevsky, Valentin. (2020). MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device. (online on: https://ai.googleblog.com/2020/12/mediapipe-holistic-simultaneous-face.html, last accessed 2021/21/06).

Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Zhang, A. (2020). GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing. In Journal of Machine Learning Research, (pp. 1-7).

Harish, M., Chethan, P., Azeem, S. A., & Veena, M. G. (2019, October). A Smart Attendance System based on Machine learning. In 2019 Global Conference for Advancement in Technology (GCAT) (pp. 1-7). IEEE.

Hernández, D.,Calleros J.M., García J., Vizzuett L. (2019). Gesture-Based Interaction for Virtual Reality Environments Through User-Defined Commands. In Human-Computer Interaction. HCI-COLLAB 2018. Communications in Computer and Information Science, vol. 847, (pp. 143–157).

Johnston, V., Black, M., Wallace, J., Mulvenna, M., Bond, R. (2019). A Framework for the Development of a Dynamic Adaptive Intelligent User Interface to Enhance the User Experience. In Procs of the 31st European Conf. on Cognitive Ergonomics, (pp. 32-35)

Kartynnik, Y., Ablavatski, A., Grishchenko, I., & Grundmann, M. (2019). Real-time facial surface geometry from monocular video on mobile GPUs. arXiv preprint arXiv:1907.06724.

Knight, A. (2021) LSTM Neural Network: The Basic Concept, An High Level Introduction to Long Short Term Memory Neural Networks (online on: https://towardsdatascience.com/lstm-neural-network-the-basic-concept-a9ba225616f7, last accessed 2021/09/07).

KNX (2020). KNX Smart Home and Building Solutions. Global. Secure. (online on: https://www.knx.org/knx-en/for-your-home/, last accessed 2020/11/12).

Li, K., Zhao, X., Bian, J., & Tan, M. (2017, August). Sequential learning for multimodal 3D human activity recognition with Long-Short Term Memory. In 2017 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1556-1561). IEEE.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.

Martín, A.A.S., Guerrero, E.G., Santamaría, L.E.B. (2019). Prospective integration between Environmental Intelligence (AMI), Data Analytics (DA), and Internet of Things (IoT). In Cong. Int. de Innovación y Tendencias en Ingenieria (CONIITI), (pp. 1-6).

MediaPipe (2021). MediaPipe ML Solutions. (online on: https://google.github.io/mediapipe, last accessed 2021/21/06).

OpenCV (2021). OpenCV Modules. (online on: https://docs.opencv.org/master/index.html, last accessed 2021/06/11).

Ouyang, A., Liu, Y., Pei, S., Peng, X., He, M., & Wang, Q. (2020). A hybrid improved kernel LDA and PNN algorithm for efficient face recognition. Neurocomputing, 393, 214-222.

Palsa, J., Vokorokos, L., Bilanova., Z. (2020). User Interface of smart environment based on human body gestures. In IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), (pp. 165-170).

Peltarion (2021a) Categorical crossentropy (online on: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy, last accessed 2021/09/07).

Peltarion (2021b) Categorical accuracy (online on: https://peltarion.com/knowledge-center/documentation/evaluation-view/classification-loss-metrics/categorical-accuracy, last accessed 2021/09/07).

Pimpare, P.L.T. (2017). Internet das Coisas e a integração de sistemas domóticos residenciais: o protocolo KNX. Universidade da Beira Interior.

Pyttsx (2021). Pyttsx3 - Text-to-speech x - plataform. (online on: https://pyttsx3.readthedocs.io/en/latest/ 2021/11/03).

Rodrigues, J.M.F., Pereira, J.A.R., Sardo, J.D.P., Freitas, M.A.G., Cardoso, P.J.S., Gomes, M., Bica, P. (2017). Adaptive Card Design UI Implementation for an Augmented Reality Museum Application, In M. Antona and C. Stephanidis (Eds.): Universal Access in Human-Computer Interaction 2017, Part I, LNCS 10277, (pp. 433–443).

Rodríguez, P., Bautista, M. A., Gonzalez, J., & Escalera, S. (2018). Beyond one-hot encoding: Lower dimensional target embedding. Image and Vision Computing, 75, 21-31.

Sachan, A. (2020) Human pose estimation using Deep Learning in OpenCV, (online on: https://cv-tricks.com/poseestimation/using-deep-learning-in-opencv/, last accessed 2021/27/04).

Santos J., Martins I., Rodrigues J.M.F (2021) Framework for Controlling KNX Devices Based on Gestures. In: Antona M., Stephanidis C. (eds) Universal Access in Human-Computer Interaction. Access to Media, Learning, and Assistive Environments. LNCS 12769. Springer, Cham. https://doi.org/10.1007/978-3-030-78095-1_37.

Sapundzhi, F. (2020). A Survey of KNX Implementation in Building Automation. In: TEM Journal. vol 9, (pp. 144 - 148).

Satake, H., Tani, R., & Shigeno, H. (2020). A Task Placement System for Face Recognition Applications in Edge Computing. In 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.

Sawant, C. (2020). Human activity recognition with OpenPose and Long Short-Term Memory on real time images (No. 2297). EasyChair.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

Stephanidis, C. (2001). Adaptive techniques for universal access. User modeling and user-adapted interaction, 11(1-2), (pp. 159-179).

Tejesh, B.S.S., Neeraja S. (2018). A Smart Home Automation system using IoT and Open Source Hardware. In International Journal of Engineering & Technology, 7 (2.7), (pp. 428-432).

Xiao, B., Wu, H., Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In Procs of the European Conf. on Computer Vision (ECCV), (pp. 466-481).

XKNX(2020). Asynchronous Python Library for KNX, (online on: https://xknx.io/, last accessed 2020/11/12).

Xu, M., Chen, D., & Zhou, G. (2020). Real-Time Face Recognition Based on Dlib. In Innovative Computing (pp. 1451-1459). Springer, Singapore.

Yadav, A. V., Verma, S. S., & Singh, D. D. (2021). Virtual Assistant for Blind People. International Journal, 6(5).

Yang, W., & Jiachun, Z. (2018). Real-time face detection based on YOLO. 2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII).

Yucheng, C., Yingli, T., Mingyi., H. (2020). Monocular Human Pose Estimation: A Survey of Deep Learning-based Methods. In Computer Vision and Image Understanding (CVIU).

Yumang, A., Abando, M., Dios, E. (2020). Far-field Speech-controlled Smart Classroom with Natural Language Processing built under KNX Standard for Appliance Control. In: International Conference on Computer and Automation Engineering, (pp. 219-223).