Summer 2024

Scene Understanding and Spatial Analysis Using Scene Graph Enhanced by Hall's Proxemics Zones in Smart Homes

Debaleen Das Spandan
Missouri State University, Debaleen0010@MissouriState.edu

SCENE UNDERSTANDING AND SPATIAL ANALYSIS USING SCENE GRAPH ENHANCED BY HALL'S PROXEMICS ZONES IN SMART HOMES

A Master's Thesis

Presented to

The Graduate College of

Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science, Computer Science

By

Debaleen Das Spandan

August 2024

The author assumes responsibility for obtaining and maintaining the necessary rights and releases to legally publish this work, including the informed consent of any subjects identifiable in the photographic and/or illustrative work.

SCENE UNDERSTANDING AND SPATIAL ANALYSIS USING SCENE GRAPH ENHANCED BY HALL'S PROXEMICS ZONES IN SMART HOMES

Computer Science

Missouri State University, August 2024

Master of Science

Debaleen Das Spandan

ABSTRACT

Voice-controlled smart assistants have received widespread popularity. It plays a pivotal role in smart homes by providing a natural and convenient interface for interacting with smart devices. However, these assistants are unable to serve persons with physical disabilities and speech impairments. Therefore, non-verbal communication methods, such as eye tracking, gesture recognition, and context awareness can complement and overcome some of these limitations to enhance user experience in smart homes. To address this issue, I am investigating non-verbal communication methods to make smart home technology more accessible and intuitive. In this research, I focus on proxemics, i.e., the study of distance between smart home users and surrounding objects, to enable spatial awareness and intuitive automation in smart homes. I apply scene graphs to provide a structured representation, such as positions, relationships, and properties, of the static and moving objects in indoor home environments. The novelty of this approach lies in the application of proxemics via scene graph generation for extracting contextual information and scene understanding to automate smart home actions. This work adds a distance attribute to the scene graph predicate for quantifying human and object relationships. The key contribution lies in leveraging proxemics through scene graph generation to extract contextual information, facilitating the automation of intelligent actions.

KEYWORDS: distance calculation, human-computer interaction, non-verbal communication, object detection, proxemics, scene analysis, scene graphs, smart home, stereo vision

SCENE UNDERSTANDING AND SPATIAL ANALYSIS USING SCENE GRAPH ENHANCED BY HALL'S PROXEMICS ZONES IN SMART HOMES

By

Debaleen Das Spandan

A Master's Thesis
Submitted to the Graduate College
Of Missouri State University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science, Computer Science

August 2024

Approved:

Razib Iqbal, Ph.D., Thesis Committee Chair

Mohammed Y Belkhouche, Ph.D., Committee Member

Adnan Maruf, Ph.D., Committee Member

Julie Masterson, Ph.D., Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Razib Iqbal, for his invaluable guidance, support, and encouragement throughout this journey. His expertise and insights have been instrumental in shaping my research and bringing this work to fruition. Moreover, his unwavering mentorship and remarkable patience have been crucial in ensuring the success of my graduate studies. Overall, he has greatly improved my life for the better.

I am also profoundly grateful to my committee members, Dr. Yassine Belkhouche and Dr. Adnan Maruf, for their constructive feedback, suggestions, and unwavering support. Their contributions have significantly enriched the quality of my research.

I would like to extend my sincere thanks to the Computer Science department head, Dr. Ajay Katangur, for his continuous support and assistance during my academic endeavors. His guidance throughout my academic journey in Missouri State University has been essential to my success.

Additionally, I am deeply thankful to my parents, grandparents, my brother and immediate relatives for their unconditional love, support, and sacrifices, which have been my constant source of strength. I am also deeply grateful to my friends for their companionship, encouragement, and for always being there for me.

This journey would not have been possible without the support and encouragement from all these remarkable individuals. Thank you all from the bottom of my heart.

I dedicate this thesis to my grandparents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF EQUATIONS

1. INTRODUCTION

Voice assistants like Apple's Siri, Google Assistant, and Amazon's Alexa have become widely popular for controlling various smart devices using voice commands [1]. While verbal communication is popular and easier to process, there are some limitations, such as people with speech impairment or people who do not want to use verbal commands finding it difficult, if not impossible, to interact with these voice assistants. Therefore, I am interested in non-verbal communication techniques for smart home setups. There are several existing techniques, such as Kinesics, Oculesics, and Environment, which have previously been applied in smart home contexts. I will elaborate on non-verbal communication and its types in the 'Background & Related Works' chapter.

Kinesics, the study of body movements, has been explored exhaustively. Works such as [2], [3], [4], [5], [6], [7], attempt to use facial expressions, body gestures, sign language, hand movements, and body postures to control certain forms of IoT devices in smart environments. Researchers have also considered Oculesics, the study of eye movement, as a non-verbal method to control smart devices. Works by many researchers [8], [9], [10], [11] use eye movements, blinks, and iris tracking to generate commands to control smart devices.

Using sensors to identify human activity within a smart environment has also been explored extensively. Researchers in the work [12] offer an autonomous in-home healthcare monitoring system for a variety of applications. They leverage multiple sensors that provide the system with multi-modal data such as geriatric physiological and behavioral data, auditory data, environmental factors, and medical information. This multimodal fusion improves overall system dependability by recognizing many distress conditions. In another work [13],

researchers utilized an actual apartment (Health Smart Home) outfitted with cameras, infrared sensors, door contact sensors, temperature and humidity sensors, and a microphone to identify Activities of Daily Life (ADL). The study's subjects also wore a wearable kinematic sensor. They utilized the data from these sensors to classify each temporal frame into one of seven ADLs (hygiene, toilet use, eating, resting, sleeping, communication, dressing). Infrared sensors identify the presence of a person within the apartment. Door contacts are used to monitor refrigerator, cupboard, and drawer usage. For voice recognition, microphones are utilized. The kinematic sensor detects and categorizes postures and walk times (using frequency parameters). Wide-angle web cameras are solely used to date the different ADLs for supervised machine learning. The Health Smart Home is served by four computers. The first computer oversees sound and voice analysis, the second three-webcam data and the controller area network, the third other two webcams, temperature, and humidity, and the fourth kitchen and bedroom door contact sensors.

Incorporating sensors in smart environments makes it easy to monitor the surroundings. However, such sensors are sometimes pervasive. Setting up the sensor systems to effectively monitor ADL such as in [12] and [13] is not cost-effective. Using eye behavior such as blinks to control devices is intriguing, yet the generation of commands using eye blinks is challenging. As the number of commands increases, the number of eye blinks required also increases. This is evident from the works in [10], [11], and [6]. Also, to track eye gazes accurately, an expensive multi-camera system is required.

To reduce the cost and complexity of using multimodal sensors and multiple cameras and to avoid pervasive sensing, I decided to work with visual data captured from camera inputs.

Another type of non-verbal communication viz. proxemics refers to the study of how humans use and perceive personal space in various social situations. Proxemics was first introduced by American anthropologist Edward T. Hall. Hall defines proxemics broadly as "the study of how man unconsciously structures microspace – the distance between men in the conduct of daily transactions, the organization of space in his houses and buildings, and ultimately the layout of his towns" [14]. In the context of smart environments, it involves understanding how users interact with the space around them and how technology can respond to these interactions. Researchers have previously demonstrated the capability of proxemics to regulate both implicit and explicit interactions. Researchers have shown that proxemics can regulate implicit and explicit interactions by the movement of individuals in and out of specific zones. It also mediates interactions among multiple people and interprets their directed attention to others and objects [15]. In another research [16], authors examine four dimensions: position, locality, identity, and movement—in Kitchen-as-a-pal, an interactive smart kitchen equipped with a sonar network and Radio Frequency Identification (RFID) technology to sense and model proxemics between humans and objects. It is evident from these research works that proxemics offer a promising solution by leveraging physical proximity and movement to facilitate interactions. However, proxemics in smart settings remain under-explored and not widely applied in mainstream technology. I plan to investigate a novel approach for non-verbal communication in smart homes by employing a scene graph enhanced with the infrequently explored non-verbal communication method, proxemics, to represent and understand the context of indoor home environments.

A scene graph represents objects in a scene as nodes and the relationships between them as edges. Each node in the graph typically corresponds to an object, and the edges represent relationships like “on”, “next to”, “inside”, etc. [17]. Therefore, scene graph generation is the problem of detecting a collection of objects and predicting the relationship or predicate among every pair of objects.

The application of scene graphs together with proxemics offers new avenues for smart home automation. Contextual information retrieved through this approach might allow for the automatic adaptation of day-to-day operational policies and settings based on user presence and surrounding objects. For example, a user can control smart home devices simply by moving closer or farther away from them. Additionally, by defining specific movements or gestures within proxemic zones, users can perform tasks such as adjusting the thermostat or playing music. Moreover, proxemics allows the creation of personalized interaction zones for different users within the household. A user with limited mobility can have a defined zone near their bed or chair where simple gestures can control multiple devices, improving their comfort, independence, and safety. Another example is recognizing a child close to a knife or an individual approaching a firearm signifies dangerous contexts triggering an alarm, whereas a person walking past a bookshelf reflects a normal scenario. Effectively inferring or classifying context from such visual data necessitates a profound understanding of the scene, encompassing the identification of objects, discernment of object relationships, and analysis of spatial arrangements. Therefore, I focus on capturing structured information, such as objects, their positions, and relationships. The process entails extracting higher-level contextual

information from images or videos by harnessing the distance as a relationship between depicted objects.

By integrating scene graphs with quantified distance attributes, my research strives to unlock a deeper understanding of contextual nuances present in smart home environments. The core objective is to enable robust non-verbal communication within these environments by facilitating the identification of contexts as dangerous, normal, or concerning. Therefore, in this research, I try to answer the following research questions:

1. Can proxemics and scene graphs be combined as a method of non-verbal communication in smart home environments?
2. Will introducing a quantifying attribute (distance between objects) enrich the predicate in a scene graph triplet?

This research has an Institutional Review Board (IRB) approval with review type as 'Exempt'. The decision is shown in Appendix A. The following peer-reviewed paper has been accepted for publication which is based on the work performed in this thesis research:

- Debaleen Das Spandan and Razib Iqbal, "ProxeGraph: Scene Graph Generation Utilizing Proxemics for Smart Homes", in 2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR), San Jose, CA, USA, August 7-9, 2024.

The rest of the document is arranged in the following format: In the '2. Background & Related Works' chapter, I familiarize the reader with non-verbal communication, distance estimation using camera, object detection and its scope in this research, scene graph and its related works and scope in this research. In the '3. Proposed Method: ProxeGraph' chapter I elaborate on the proposed methodology and architecture. In the '4. Implementation' chapter I

discuss the experimental setup and deployment of the proposed architecture. In the ‘5. Performance Evaluation’ chapter I discuss the performance of the deployment and the proposed architecture in details. In the ‘6. Conclusion’ chapter I summarize the entirety of the research, discusses the limitations, and suggests directions for future work.

2. BACKGROUND & RELATED WORKS

In this chapter, I will discuss non-verbal communication and its different types. Following that, I will explain the experiments I conducted to estimate distance using a camera. Since distance estimation is an integral part of my research but is not the primary focus, I am including these experiments in this background chapter. After the experiments on distance estimation, I will give a brief introduction to object detection, and lastly, I will introduce the scene graph and its related works.

2.1. Non-verbal Communication

According to [18], non-verbal communication is the process of transmitting messages or information through means other than words. It includes various forms of body language, facial expressions, gestures, posture, and other physical behaviors that convey meaning. Non-verbal communication can complement, enhance, or even contradict verbal messages. Understanding non-verbal communication is crucial as it often provides more context and emotional insight than words alone [19]. Based on the information obtained from [18], [20], [21] and [22], I reach the conclusion that non-verbal communication can be broadly classified in to at least 8 different types. These types are explained in the following paragraphs.

Oculesics: As per [23], the study of the communicative role of the eyes in nonverbal communication is known as Oculesics. The primary behaviors in Oculesics include eye-contact/eye gaze and eye blinks. According to [18] and [20], eye contact helps build rapport and connection and signals when turn-taking or engagement in cognitive activities. Eye contact is utilized to convey information, regulate dialogue, and monitor interactions by observing feedback and other nonverbal cues. For instance, eye contact signals readiness to speak or

prompts others to talk, demonstrating the use of Oculesics in controlling communication. Through eye contact, hand gestures, facial expressions, and appearance can be interpreted. A speaker can evaluate the audience's eye contact to determine if they are confused, engaged, or bored, and adjust their message accordingly.

Vocalics: As per [20], the study of nonverbal voice cues is known as vocalics or paralanguage. Rather than focusing on the content of what we say, it examines how we express it. Unlike the verbal aspect of speech, which involves words and their meanings, vocalics is concerned with the vocal elements of speech. These elements include components such as intensity, pace, tone, fluency, and vocal patterns.

Haptics: According to [18], the study of touch in communication is known as Haptics. It refers to the touch behaviors that convey message during interactions. Depending on the nature of the contact and the relationship between individuals, touch can communicate support, comfort, control, or aggression.

Chronemics: According to [24], the study of the communicative role of time in nonverbal behavior is called chronemics. Chronemics also explores cultural differences in how individuals perceive time as either fixed and unchanging (monochronic) or flexible and adaptable (polychronic). Factors such as punctuality, speech speed, and the duration of conversations can convey messages about importance, interest, and respect.

Kinesics: According to [20], kinesics is the study and interpretation of human body movements that can be taken as metaphorical or symbolic in social interaction. It was coined by Anthropologist Ray Birdwhistell in 1952. Kinesics can be further classified into body movements, which include deliberate movements such as gestures and involuntary movements

like shifts in body position that convey messages about a person's feelings or intentions; gestures which are hand and arm movements that are used to express ideas, indicate direction, or emphasize verbal communication, such as waving, pointing, and using hand signs; facial expressions or the movement of the facial muscles that convey emotions such as sadness , happiness, surprise, and anger, often universal and recognized across different cultures; head movement, such as nodding in agreement or denial; and posture, or the way an individual positions their body while sitting or standing, which can indicate attitudes, emotions, and levels of engagement or interest.

Appearance: Personal appearance, artifacts, and objects are types of nonverbal communication used to adorn our bodies and surroundings to communicate meaning to others. Aside from clothes, artifacts like visible body art, hairstyles, jewelry, and other social, political, and cultural symbols convey messages to others about who we are [18].

Environment: According to [18], our verbal and nonverbal communication are influenced by the environment in which we interact. The location of furniture and objects in physical space can contribute to the creation of a formal, distant, friendly, or intimate atmosphere. For example, a room with a small fountain creating soothing water sounds, soft lighting, and a comfortable chair can enhance the interaction between therapists and patients. Therefore, environmental cues are a form of non-verbal communication.

Proxemics: According to [20], the study of intrepersonal distance and space in communication is called proxemics. This includes studying personal space, territoriality, crowding, and density.

Figure 1 gives a diagrammatic representation of various types of non-verbal communication.

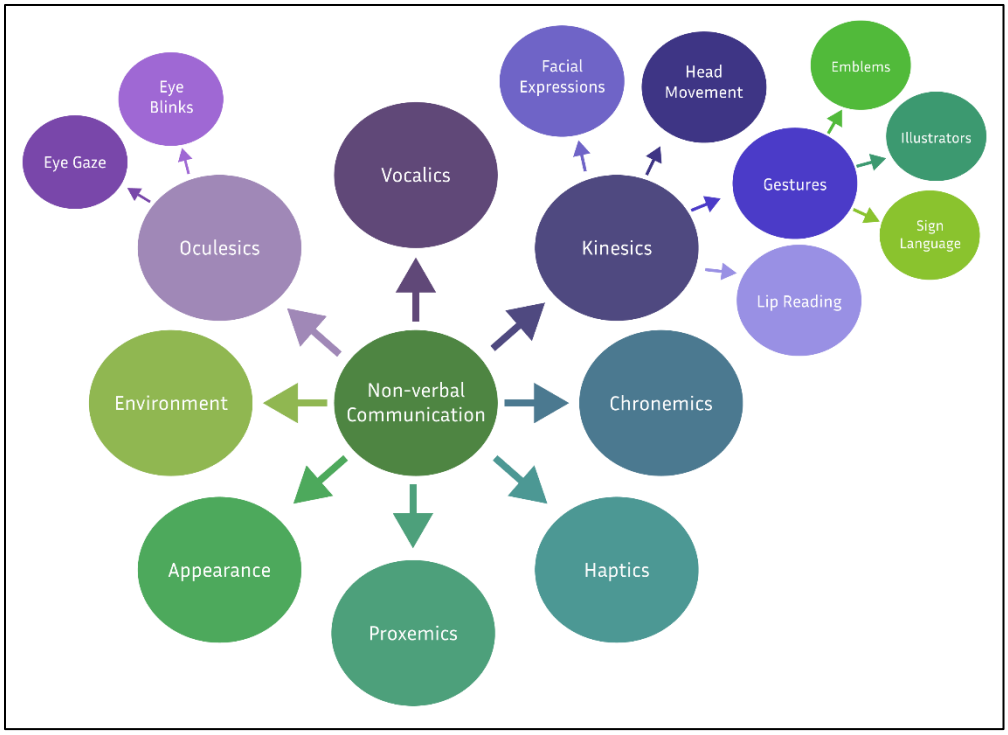


Figure 1 Different Types of Non-verbal Communication.

As discussed in the '1. Introduction' chapter, most of the different types of non-verbal communication that are mentioned earlier in this section have been widely explored in the field of smart-home communication or human-computer interaction. Proxemics is one of the intriguing and under-explored non-verbal communication methods with good potential. Since, proxemics is the study of distance, and since this research is focused on using a camera to extract contextual information from a smart-home environment, a suitable method for estimating the distance between smart-home elements is necessary. This research also requires being able to detect objects present in a smart home from the captured camera frames. In the next part of this chapter, I will explain different methods for estimating distance using vision explored in this research which will be followed by a brief introduction to object detection and literature review on scene graph.

2.2. Distance Estimation

To accurately estimate distances between objects, I first explored the efficacy of different methods which utilize mono-vision cameras, and then focused on stereo-vision cameras. Techniques utilizing mono-vision cameras estimate the distance of an object from the camera using a single image captured by a single-lens camera, whereas stereo-vision techniques use two images captured by two lenses to calculate the depth of a point object from the camera.

In this research work, I have explored various methods of depth estimation and distance calculation using mono-vision and stereo-vision cameras as described in the following subsections.

2.2.1. Method 1: Using a Reference Object

The approach presented in [25] uses a reference object present in each scene to calculate the distance between objects in the given scene. The reference object must meet two primary criteria – i) its real-world dimensions need to be known in units like inches, millimeters, or centimeters, and ii) it should be easily identifiable within the image based on appearance or location or size. For example, the only square object or the left-most object or the smallest object can be easily identified in an image.

To find the real-world distances among the objects present in the images/video, the author applied 2 steps. In the first step the author calculates a pixel-per-metric ratio as given in Equation 1 where, PPM = pixel-per-matric ratio, W_p = width of the reference object in pixels and W_r = width of the reference object in real-world. This ratio helps to determine the pixels that are needed to represent a specific real-world distance in metric units.

$$PPM = \frac{W_p}{W_r} \quad (1)$$

In the second step, bounding boxes are created for the objects and the pixel distance between these objects is calculated by computing the Euclidean distance between either the corners of these bounding boxes or their midpoint. Then the real-world distance between these objects is calculated by dividing the pixel distance by the pixel-per-metric ratio, as shown in Equation 2 where, D_r = real-world-distance, $PD_{1,2}$ = (Pixel distance between object 1 and object 2, and PPM = pixel-per-metric ratio.

$$D_r = \frac{PD_{1,2}}{PPM} \quad (2)$$

Figure 2 shows a successful experiment using this method to estimate the distance between a watch (target object) and a ring (reference object). Figure 3 shows an unsuccessful experiment where this method was unable to identify the target object because of a noisy background. While this method is computationally simple and does not require depth information, it relies on a reference object in the image which requires prior knowledge of the real-world dimensions of that reference object. Also, the orientation of the camera can introduce pixel distortion which may potentially lead to inaccuracies in the computed Euclidean distance. Therefore, my verdict is that this method is not suitable for smart-home environments as it is unable to give accurate results in complex scenarios.

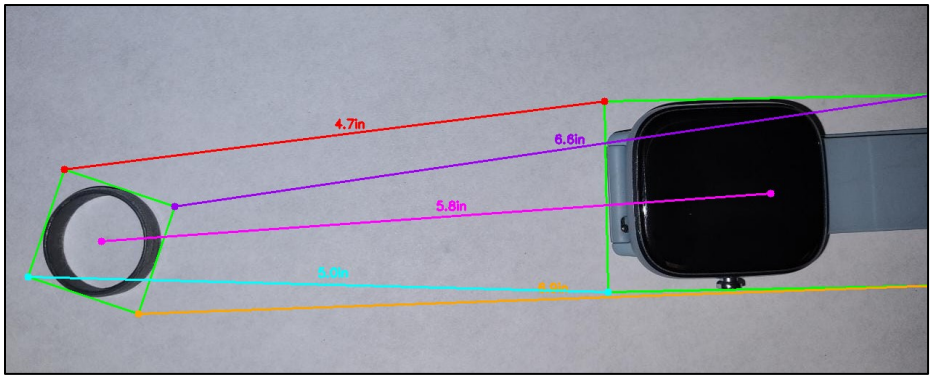


Figure 2 Experiment Using Method 1 with Accurate Results.

Figure 3 Experiment Using Method 1 with Inaccurate Results.

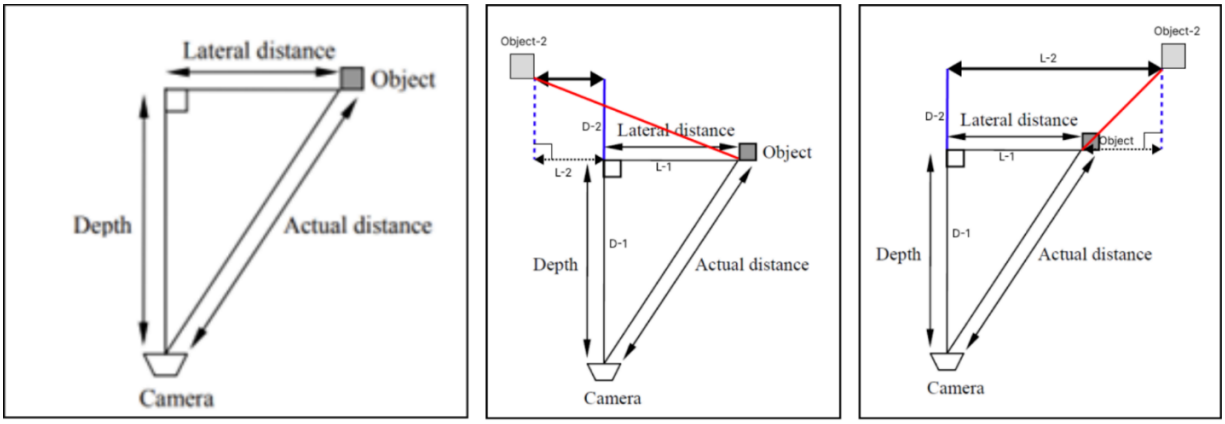
2.2.2. Method 2: Using Polynomial Equations

In this method [26], the authors proposed a method for distance estimation using a mono-vision approach using a polynomial equation (Equation 3) which is most suitable for rectangular objects. In this equation, δ represents the depth of the object, and ρ denotes the pixel height. The authors also proposed Equation 4 to estimate the 'physical-pixel lateral distance ratio'. Finally, to determine the actual distance of an object from the camera, the Pythagoras formula as given in Equation 5 and illustrated in Figure 4a, is applied, where D_o = Depth of the object from the camera and, L = lateral distance of the object from the camera.

$$\delta = 0.0033 \times \rho^2 - 1.9918 \times \rho + 372.83 \quad (3)$$

$$Ratio = 0.0004 \times \delta^2 - 0.1469 \times \delta + 18.119 \quad (4)$$

$$distance = \sqrt{D_o^2 + L^2} \quad (5)$$



a b c

Figure 4 Method 2. a) Theory; b) & c) Adaptation.

I adapted this method as shown in Figure 4b and Figure 4c to estimate the distance between objects. If there are 2 objects, the depth between the objects will be the difference between the depth of object 2 and the depth of object 1. The lateral distance between the objects will be either the sum of the lateral distances of the objects (Figure 4b) or the difference between the lateral distances between the objects (Figure 4c). However, in my experiment using this adapted method, as shown in Figure 5, it failed to correctly estimate the distance between a ring and a watch. Despite both objects being positioned at the same distance from the camera (40 cm), the calculated depths deviated significantly. The watch's distance from the camera was determined as 74.1 cm, while the ring's distance unexpectedly measured 133.1 cm. This discrepancy reveals a limitation in this method's precision for distance estimation in this experimental scenario. Also, the original method assumes the use of a rigid geometrical shape such as a rectangle, cylinder, triangular or circular objects only. This is a constraint that may

impact its adaptability to scenarios involving more complex or deformable objects present in smart-home environments.

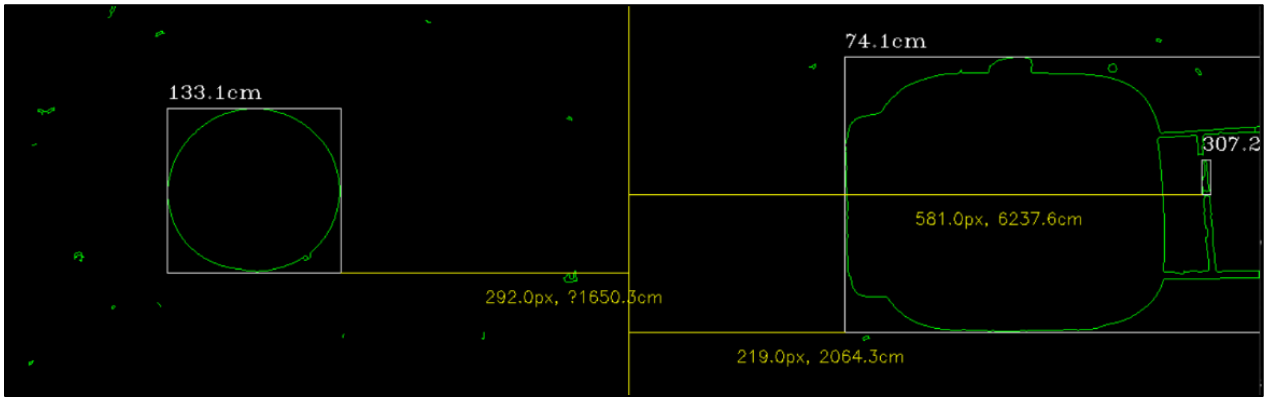


Figure 5 Method 2 Experiment Inaccurate Result.

2.2.3. Method 3: Using Real-World Dimensions

In [27] and [28], the authors use a mono-vision camera and OpenCV to estimate distances of objects from the camera. Both approaches consist of 5 steps - They mention a series of 5 key steps for precise distance estimation as given below: i) Capturing a reference image with a reference object – in this case a human face, ii) Measuring the distance of the reference object from the camera (*KD*) along with the width of the reference object (r_w), iii) Calculation of pixel width (f_w) of the reference object in the reference image, iv) Calculation of focal length of the camera, and v) Calculation of distance of other objects from the camera.

Using the measured (*KD*, r_w) and calculated (f_w) dimensions, the authors employ a mathematical formula to calculate the focal length of the camera. The equation is given in Equation 6. With the focal length established, the authors compute the actual distance of the object from the camera using the following formula given in Equation 7.

$$focal\ length = \frac{f_w * KD}{r_w} \tag{6}$$

$$distance = (r_w * focal\ length)/f_w \quad (7)$$

Figure 6 shows a sample from my experiments using this method where the distance of my face from the camera is calculated. The distance is calculated to be 21.93 cm while the actual distance was 40 cm.

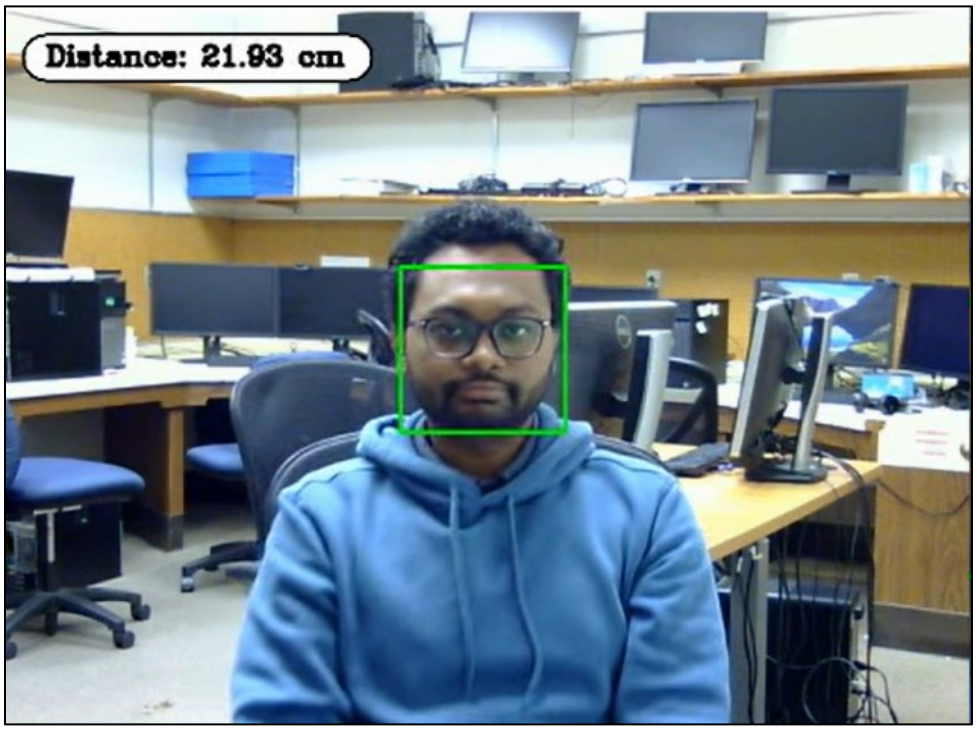


Figure 6 Distance Estimation Using Method 3.

While this approach offers several advantages, including its mathematical simplicity, efficiency, and independence from the need for depth information, its dependence on the presence of a reference image and a reference object introduces a potential limitation. Additionally, practical application of the method requires prior knowledge of the real-world dimensions of the reference object, presenting a potential challenge. Therefore, I conclude that this method is not suitable for my work.

Mono-vision approaches provide good estimation of distance with less computational complexity. However, these approaches require some prior knowledge such as a reference object and reference real-world dimensions. To overcome this drawback, I explored stereo vision approaches for distance estimation. The foundation of stereo vision is analogous to 3D perception in human vision, relying on the triangulation of rays from multiple viewpoints. Researchers in [29] introduced a novel approach to distance estimation for pedestrian identification utilizing a stereo camera (Intel RealSense D435). This approach creates a mapping of the pixels of the point object (i.e., a pedestrian) from the normal image frame with a depth map. This pixel-to-depth map mapping enables the computation of a median distance for each pedestrian. Depth maps are computed from the stereo images captured by the stereo camera's left and right lenses and then aligned with the corresponding RGB image. Considering the accuracy and precision of stereo vision cameras, this approach is a good candidate for my research. A comparison between distance estimation using the mono-vision approach and the stereo-vision approach is shown in Table 1. While the advantages of the stereo-vision approach are evident, the cost of a specialized camera is considered as a drawback. Therefore, to select a suitable stereo camera for my research, I have carefully examined 3 different stereo cameras, and the advantages and disadvantages are discussed below and also summarized in Table 2.

2.2.4. Stereo Camera-1: Intel RealSense Depth Camera D435

The Intel RealSense D435 is a depth camera powered by USB which consists of a pair of depth sensors, an RGB sensor, and an infrared projector [30]. The image sensors in this camera offer an excellent 1280 x 720 depth stream resolution along with 1920 x 1080 RGB resolution. The active infrared projector helps in illuminating objects to enhance depth data. The on-board

powerful vision processor enables this camera to run advanced stereo depth algorithms enhancing computation of real-time depth data and accelerating output. The camera can be programmed by an open-source SDK provided by the manufacturer. This camera was also used in the work by authors of [29]. However, with a $349 cost, this camera becomes too expensive for a smart-home implementation.

Table 1 Comparison Between Mono and Stereo Vision Approach for Distance Calculation.

Attribute	Mono-vision Approach	Stereo Vision Approach
Reference Object	Reference object required.	No reference object required.
Real-world Dimensions	Real world dimensions required.	No real-world dimensions required.
Special Camera	No special camera required.	Camera with stereo vision capabilities required.
Cost	Comparatively cheaper as no specialized hardware is necessary.	Since specialized hardware is required, the cost can be higher than its counterpart.
Accuracy	Less accurate than stereo vision.	More accurate than mono-vision.
Pixel Distortion	More prone to pixel distortion.	Less prone to pixel distortion.
Computational Complexity	Comparatively less complex.	Requires calculating the disparity between frames captured by each lens in the stereo setup and therefore increasing complexity more than the counterpart.

2.2.5. Stereo Camera-2: E-Con Systems' Tara Stereo Camera

According to [31], Tara is a UVC-compliant 3D stereo camera that uses the MT9V024 stereo sensor. It supports Wide Video Graphics Array (WVGA) resolution at 60 frames per second (fps) in an uncompressed format. It delivers two synchronized sensor frame data streams to the host machine though a USB 3.0 interface. While stereo camera algorithms can be developed using Tara, the proprietary Software Development Kit (SDK) causes inconvenience due to the lack of implementation of object detection algorithm. Although such an algorithm can be implemented by a user application, the inability of the camera to provide a color image introduces further obstacles for object detection. The Figure 7 shows an experiment using their SDK. We can see that the camera can successfully calculate the distance of the subject from the camera as shown in the middle bottom subfigure of Figure 7. Since the actual color frame was not available, object detection could not be performed. Therefore, this camera is not suitable for my research.

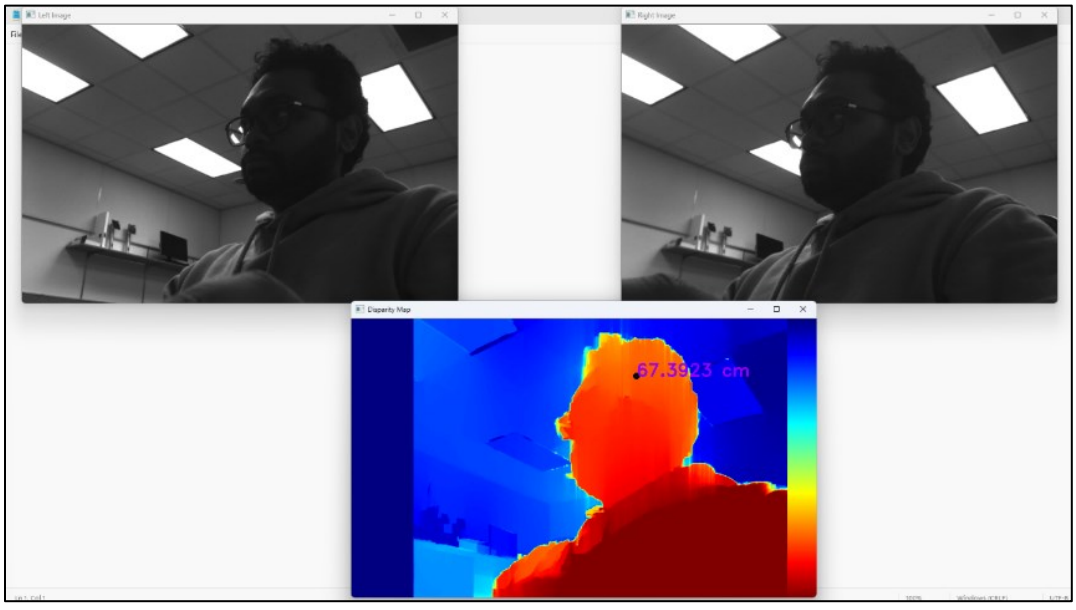


Figure 7 Using Tara Stereo Camera by E-con Systems.

2.2.6. Stereo Camera-3: Oak-D Lite Camera by Luxonis

The Oak-D lite Camera [32] is a specialized machine vision stereo camera with 4K color lens. It supports offloading of computer vision algorithms into the built-in Intel Myriad-X Visual Processing Unit. The SDK is an open-source DepthAI Application Programming Interface (API) with support for Python programming language. The camera is available for $149, which makes it cheaper than the previously mentioned stereo cameras. I experimented with this camera and concluded that I could successfully detect objects using the RGB image and an object detection neural network and then use the depth information from stereo lenses to calculate the distance between the objects. An experimental result of using Oak-D Lite camera is shown in Figure 8 which depicts an indoor scene where two chairs have been detected and labeled with bounding boxes, including their spatial coordinates (x, y, z) in millimeters. The image also shows the calculated distance between the two chairs, marked by a red line and labeled as 77.36 mm.

Figure 8 Using Oak-D Lite Camera by Luxonis.

As evident by the experiments and explorations conducted, a stereo camera is the most precise and accurate when it comes to estimating depth. However, there are a lot of stereo cameras available in the market and thus picking a suitable stereo camera for my research is crucial. Since smart home environments demand affordable, real-time, and fast inference systems, it is important to find a balance between cost and efficiency of the camera being selected. In Table 2, I provide a comparison of the 3 cameras I explored for my research.

Among the three stereo cameras, the Intel D435 is the costliest. Even if this research was to create an algorithm for real-world distance measurement with this camera, the cost would prevent it from being widely used. Conversely, the most affordable choice is the OAK-D Lite camera. In addition to being cost effective, it offers stereo vision and the extra flexibility to offload computer vision models to its hardware, possibly proving helpful for real-world implementations of my method. Although the Tara Stereo camera from the E-Con System is a good substitute, it is limited since it does not have the color frames that this research needs for object detection. Further, developing a flexible system is hampered by the SDK's proprietary nature. By comparison, the OAK-D Lite camera proves to be a more affordable option, including not just a color camera but also an in-built vision processing module. Therefore, for this research work, I decided to work with the OAK-D Lite camera.

Along with estimating distance between objects using a camera, this research work also requires detecting objects from a given camera input. In the next part of this chapter, I will briefly introduce object detection and establish the scope of object detection in this research work.

Table 2 Stereo-camera Comparison.

Device	Intel RealSense Depth D435	Tara E-Con Stereo Camera	OAK-D Lite
Cost	$314	$299	$149
Highlight	Specialized stereo camera with IR projector and 4K color lens	Stereo camera.	Specialized machine vision stereo camera with 4K color lens.
SDK	Open-source Intel® RealSense™ SDK	Proprietary SDK by E-Con systems	Open source DepthAI API
Color Camera Resolution	1920 × 1080	No color camera	4208 x 3120
Stereo camera Resolution	Up to 1280 × 720	1504x480	480P (640x480)
ML/AI capabilities	No custom model supported.	No hardware support	Image (Computer Vision) processing is offloaded to the camera hardware allowing us to use it on low-capability machines. Supports custom AI models.

Table 2 continued

Device	Intel RealSense Depth D435	Tara E-Con Stereo Camera	OAK-D Lite
Vision Processor	Built-in Intel RealSense Vision Processor D4.	No built-in processor	Built-in Intel Myriad-X Visual Processing Unit capable of pre-built Artificial Intelligence (AI) algorithms such as object detection. Can also be modified to run custom AI models.

2.3. Object Detection

Object detection is a computer vision method used to recognize and locate objects within an image or video frame. This process typically includes two main tasks: classification, where the type of object is identified, and localization, where the object's position is determined using bounding boxes. Object detection is an ongoing and elaborate area of research, encompassing a wide range of techniques and methodologies aimed at accurately identifying and localizing objects within images or video frames. Over the years, numerous advancements have been made in this field, including the creation of various deep learning models such as R-CNN [33], Fast R-CNN [34], Faster R-CNN [35], YOLO [36], SSD [37], RetinaNet [38], EfficientDet [39], MobileNet [40], and the Detectron2 platform [41].

Evaluating object detection models involves various metrics to measure their performance. Precision and recall are fundamental metrics, where precision is the ratio of true positive detections to the total number of positive detections, and recall is the ratio of true positive detections to the total number of actual positives. The F1 score, a harmonic mean of precision and recall, is often used to balance these metrics. IoU in object detection is a metric which evaluates the degree of overlap among the bounding box ground truth (gt) and predicted bounding box (pd). The bounding box can be any shape – rectangular, circular or irregular shape. IoU ranges between 0 to 1 where 0 indicates no overlap and 1 indicates complete overlap. A diagrammatic representation of IoU is shown in Figure 9. IoU at 0.5 means that the predicted bounding box overlaps the bounding box ground truth by at least 50%. Similarly, for IoU at 0.95 means the overlap is at least 95%.

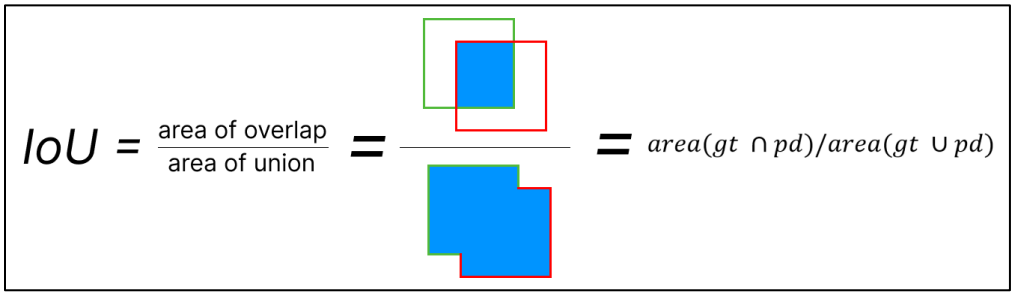


Figure 9 Intersection Over Union (IoU) Depiction.

Another important metric in evaluating object detection models is mean Average Precision at alpha or mAP@α. The formal definition of Average Precision (AP) is given in Equation 8.

$$AP@\alpha = \int_0^1 p(r)dr \tag{8}$$

In Equation 8, α is the IoU threshold at which the precision and recall are being measured, *p* refers to the precision which measures the model's exactness in identifying only the relevant objects and *r* is recall which is the ability of the model to detect all ground truths.

For object detection, precision (p) is given by the Equation 9. And recall (r) is given by Equation 10.

$$p = \frac{TP}{TP + FN} = \frac{TP}{all\ detections} \tag{9}$$

$$r = \frac{TP}{TP + FN} = \frac{TP}{all\ ground - truths} \tag{10}$$

In the above formulas, TP denotes true positive, and FN denotes false negatives.

The mean Average Precision is the average of AP values over all classes. It is given by Equation 11.

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i \ \ for\ n\ classes \tag{11}$$

A higher mAP value at a given threshold for IoU means the model can predict the bounding boxes and the class label for a given object more precisely and more accurately.

Despite the critical importance and rapid advancements in object detection, my research work does not aim to contribute to this particular domain directly. Instead, object detection is utilized as an intermediate process within a broader framework. The focus of my work lies in exploring novel methods of non-verbal communication in smart home environments by integrating proxemics and scene graph generation. While object detection provides the necessary foundational capability to identify and localize objects within a scene, the primary objective of my research is to leverage this information to infer context based on

spatial relationships and proximity. Thus, improving or enhancing object detection algorithms is beyond the scope of this research. In this research work, I have employed an existing state-of-the-art object detection model, YOLOv8 [42] to detect the objects from a given camera input.

2.4. Scene Graph

Researchers in [17] describes scene graph as a graphical depiction of a given scene where the nodes of the graph represent the objects present in the scene and the edges are the relationship between those objects. A more detailed definition of scene graph is provided in the 'Proposed Method: ProxeGraph' chapter. Recognizing context or describing a scenario from visual data such as images and videos has been an intriguing field of research. Previously, researchers developed a neural network to determine the hidden alignment between sentence segments and the corresponding regions in the image they describe [43] . This neural network takes an image as input and produces a textual description of it. Researchers have also investigated the extent to which scene graphs can enhance image captioning [44]. Scene Graphs are often regarded as an input for scene understanding [45]. The investigation of physical relations between objects to aid in scene understanding has been explored in [46], [47], [48]. Researchers in [49] introduced a Spatiality-Context-Appearance module designed to capture spatially aware contextual feature representations. Another research [45] proposed a novel method where the authors used a Convolutional Neural Network (CNN) on a red, green, blue color image with depth information (RGB-D) image for object detection and object segmentation. A super pixel image is created from object segmentation from the depth image. Bounding boxes of the detected objects, along with the super pixel map, are utilized to segment the objects. Surface normal is calculated in parallel to this process and it is used to generate the

scene graph. Using their framework, it is possible to create semantic scene graphs which consider the spatial relations between objects. Researchers in [50] proposes a novel model designed to learn features at different semantic levels by simultaneously addressing three vision tasks: object detection, scene graph generation, and region captioning. This approach includes a dynamic graph construction layer within the CNN to create a semantic graph. The semantic graph is built on semantic and spatial relationships among Regions of Interests (ROIs). The scene graph is defined as G (V, E) where each node in V represents specialized features of a ROI. The edge set E consists of undirected edges connecting captions and phrases. Researchers in [51] uses distance relation as a spatial information for scene graphs. However, their work uses overlap ration and union ration between two regions to calculate the distance. But this approach is not applicable to scenes with no overlapping regions between the subject and object. In another research, [52] researchers propose the use of depth maps for visual relation detection. The authors use the CNN trained on a depth map to improve the detection of spatial relations such as “holding”, “behind”, “above” etc. Although they use spatial relations, a quantifying attribute for the relationship labels is lacking.

The current works [17] in Scene Graph and Scene Graph generation uses labels like “has”, “is”, “contains”, “wearing” etc. Although, there are labels in the current works which specifies the spatial relationship between objects (labels such as “in-front of”, “behind”, “along”, “near”, “on” etc.), there is no attribute which can successfully quantify the relationship between the objects. The research works in scene understanding which considers spatial information [53], [54], [55], [56] have gained traction in the recent years. However, they mainly process spatial information in a way which is computationally expensive.

Previous works in this field have either neglected depth information, employed complex methods for processing depth data, or failed to utilize depth information to quantify spatial relationships effectively. To address these deficits, my research aims to reduce computational complexity by considering only the distance between objects, rather than identifying the spatial relationship labels as performed in other research works. This approach includes depth information between objects and simplifies computation by incorporating distance as a quantifying attribute in the predicates of a scene graph. The inclusion of distance attributes allows for a more accurate capture of spatial relationships between objects and subjects, thereby enhancing the reliability of contextual interpretation. For instance, consider the example of a knife in proximity to a child: while "Knife in front of child" conveys a certain level of concern, appending distance information such as "Knife 10 meters in front of child" introduces an added layer of contextual significance. By using distance as a quantifiable component in a scene graph, this work introduces proxemics as a mode of non-verbal communication in smart home settings. My research leverages scene graphs to incorporate proxemics, utilizing spatial relationships for contextual inference in smart environments. While extensive research exists on scene graph generation methodologies, my objective is to adapt the existing concept of scene graphs to use proxemics. In the next chapter, I will elaborate on my proposed methodology – ProxeGraph.

3. PROPOSED METHOD: PROXEGRAPH

In my proposed approach, I have employed **scene graph** enhanced with infrequently explored non-verbal communication method, **proxemics**, to represent and understand the scene and spatial arrangements of humans and objects in indoor home environments. In the next subsection, I elaborate on my proposed modification in scene graph definition to introduce proxemics. Following that, I propose a novel architecture to generate proxemics-enhanced scene graphs.

3.1. ProxeGraph Definition

In this research, I aimed to introduce proxemics as a non-verbal communication in smart homes with the help of scene graphs. To achieve my goal, I modified the scene graph definition to include a quantifying attribute as the predicate in a scene graph. To explain my approach, I first present the definition of scene graphs in the next paragraph.

As per [57]**,** a scene graph for a visual scene $S \in S$, such as an image or 3D mesh, is a set of visual triplets $R_s \subseteq O_s \times P_s \times (O_s \cup A_s)$, where O_s is the set of objects. The attribute set is denoted by A_s and the relation set is denoted by P_s. Each object $o_{S,k} \in O_S$ holds a semantic label $l_{S,k} \in O_L$ (The semantic label set is denoted O_L) and is grounded by a bounding box (BB) $b_{S,k}$ in scene S, where $k \in \{1, \ldots, |O_S|\}$. Each relation $p_{S,i\to j} \in P_S \subseteq P$ is the primary form of a visual relationship triplet $r_{S,i\to j} = (o_{S,i}, p_{S,i\to j}, o_{S,j}) \in R_S$ and $i \neq j$, where the third element $o_{S,j}$ could be an attribute $a_{S,j} \subseteq A_S$ if $p_{S,i\to j}$ is $p_{S,is}$. For one-way relationship, it is expressed $r_{S,i\to j}$ as $(s_{S,i}, p_{S,i\to j}, o_{S,j})$ to retain semantic accuracy where $s_{S,i}, o_{S,j} \in O_S$, $s_{S,i}$ is subject and $o_{S,j}$ is object. In the perspective of graph theory, a scene graph is a directed graph consisting of three distinct types of nodes - object, attributes, and relation. However, for semantic convenience, a node of a scene graph

refers to an object along with all its attributes, while the relation is represented as the edge between the nodes.

In this research work, I propose a modification of the relation set P_s, where the relation between the objects and subjects in a scene graph is depicted as a quantifiable label. Therefore, the modified visual relationship triplet becomes $r_{S,i\rightarrow j} = (o_{S,i}, p_{S,i\rightarrow j}, o_{S,j}) \in R_S$ and $i \neq j$, where $s_{S,i}$, $o_{S,j} \in O_S$, $s_{S,i}$ is subject and $o_{S,j}$ is object and $p_{S,i\rightarrow j}$ is a label derived from Hall's proxemics zones [14] as shown in Figure 10 which illustrates the zones, categorizing personal space into four distinct areas: Intimate (0-50 cm), Personal (0.5-1 m), Social (1-4 m), and Public (4 m or more). In terms of graph theory, the above-mentioned modification results in a scene graph where the nodes of the graph are detected objects in the scene and the edges between the nodes are a relation derived from Hall's proxemics zone division. I name this proxemics-enhanced scene graph as ProxeGraph. Generating a ProxeGraph involves object detection and the capture and processing of spatial information from a given scene. Figure 11 illustrates the steps in ProxeGraph generation. The process begins with capturing spatial information in the form of Cartesian coordinates and detecting the objects within the scene. In the next step, the spatial information is processed to calculate the distances between the detected objects. These calculated distances are then used to apply Hall's proxemics zones, which determine the labels and relationships between the objects based on their spatial proximity. In the next subsection, I discuss the proposed novel architecture for ProxeGraph generation in smart-home environments.

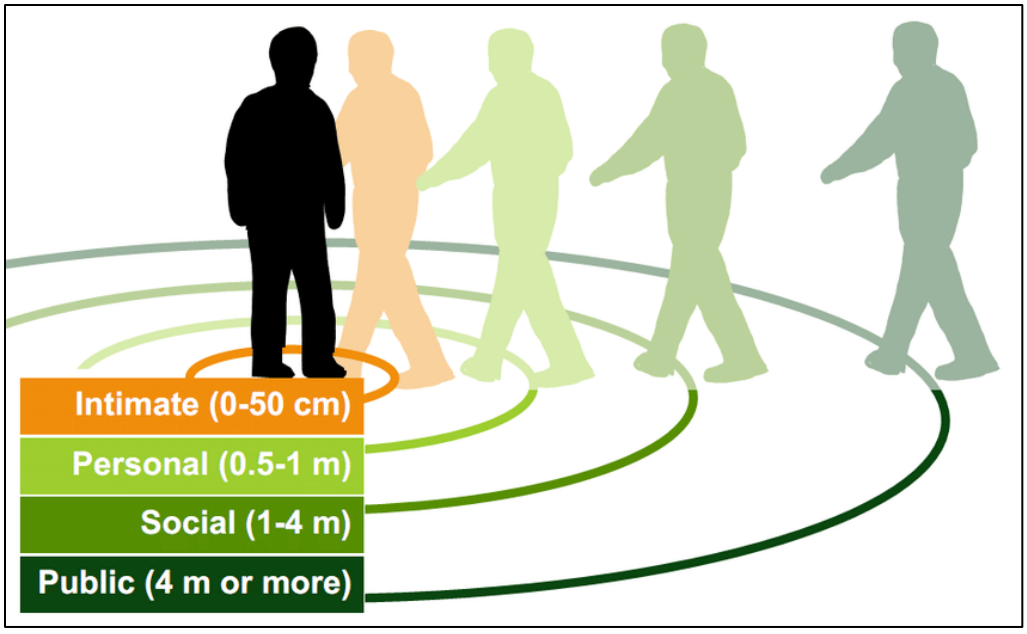


Figure 10 Hall's Proxemics Zones.

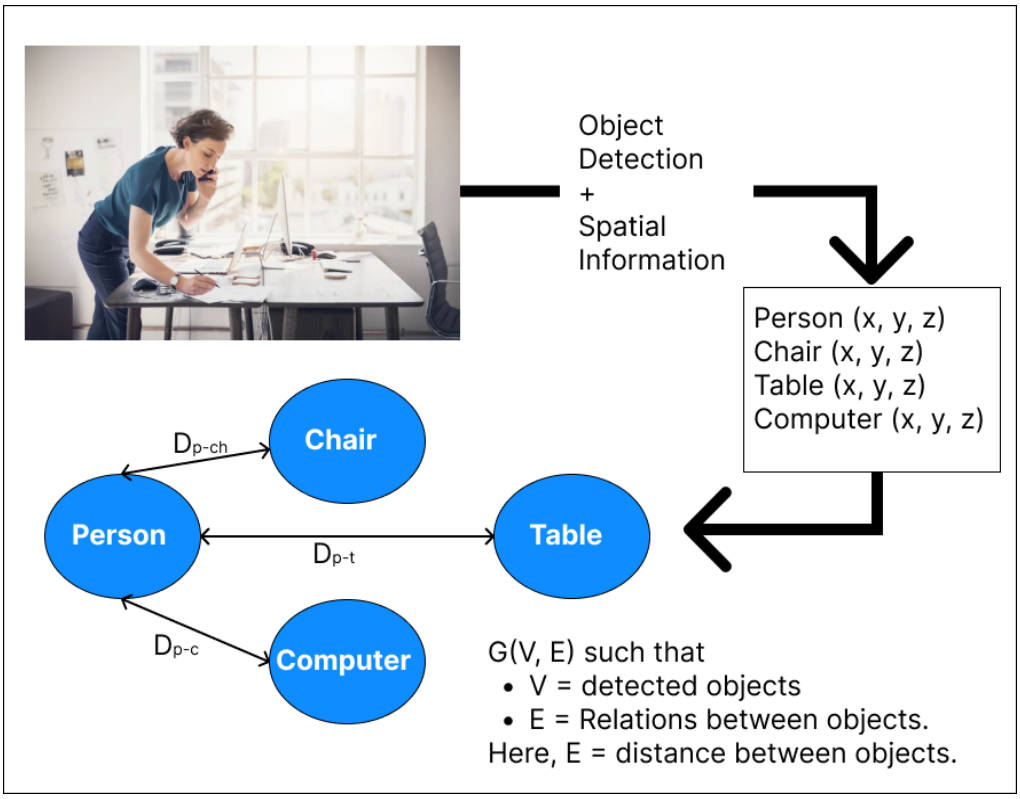


Figure 11 ProxeGraph Generation Steps.

3.2. Proposed Architecture

The proposed system architecture, as shown in Figure 12, accepts input from a stereo vision camera – the RGB frame and the stereo information. The object detection module uses the RGB frame to detect and calculate the bounding boxes of the objects present in the scene. The stereo information along with the detected object information is passed to the distance calculation module. The distance calculation module uses the stereo information to produce the depth map and then uses the depth map along with the object detections to calculate the distance between person and object pairs using Euclidean geometry. The object detection module consists of a neural network trained on smart home objects dataset. The neural network analyzes the image and gives the bounding boxes of detected images and their corresponding labels. Along with this information and the depth information from stereo camera, I calculate the distance between the corresponding subjects and objects. Finally, Hall's proxemics zone separations are applied on the distance information to get the label for the relation predicate between all possible pairs of subjects and objects. Since my focus is on smart-home environments, and humans play a significant role in a smart-home environment, the object is always a person, and the subject is a smart home device or object. The final output of my proposed approach is a scene graph with quantifiable predicate. I use this proxemics-enhanced scene graph to understand and analyze the spatial arrangements of humans and objects in smart homes. In the following paragraphs, I give more details about the various processes involved in the proposed architecture.

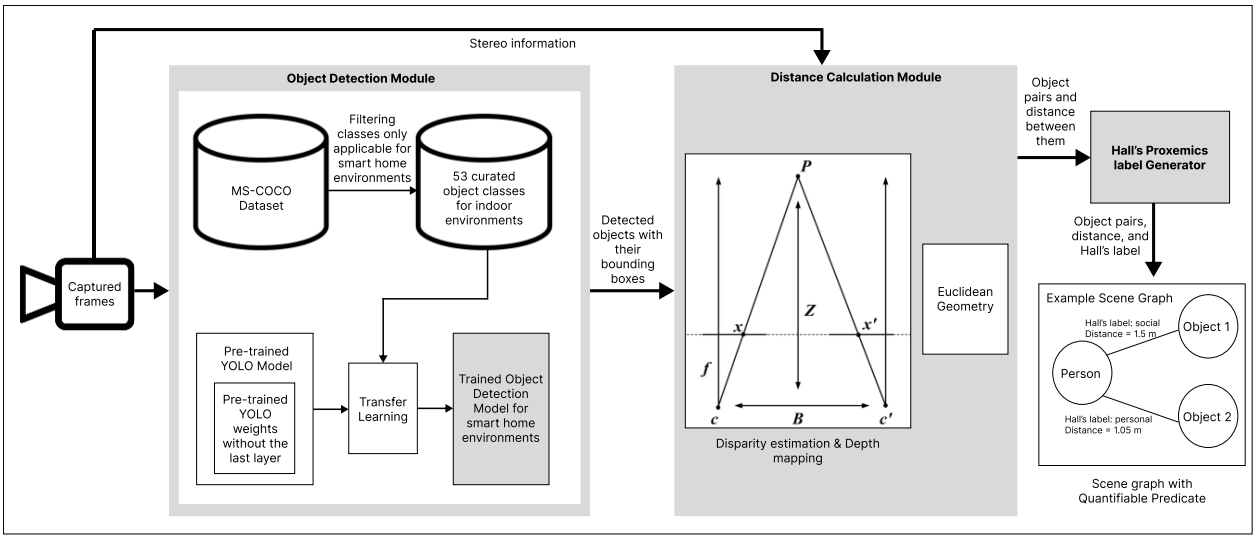


Figure 12 Proposed Architecture.

3.2.1. Object Detection

The object detection module is responsible for detecting the smart-home objects and persons from a given video input. As mentioned earlier, the object detection module consists of a neural network that is trained on a smart-home object dataset. I used transfer learning on a YOLO architecture [42] trained on the MS-COCO [58] dataset with 53 curated classes for smart home environments which is further detailed in the 'Implementation' chapter. This neural network calculates the bounding boxes of the detected objects and persons along with the detected object categories. Finally, this information is passed to the distance calculation module.

3.2.2. Distance Calculation

This module is responsible for calculating the distances between the detected objects and persons. As discussed in the '2. Background & Related Works' chapter, I explored 2 approaches for distance estimation. The first one is a mono-vision approach, and the second

one is a stereo-vision approach. I have experimented with 3 different methods that use the mono-vision approach and 3 different methods that use the stereo-vision approach. These experiments are detailed in '2. Background & Related Works' chapter. Based on these experiments, I concluded that stereo-vision approaches are better suited for distance estimation from images or videos. In the proposed framework, I use a stereo-vision camera to estimate the distance between the detected objects and persons received from the previous module. The stereo-vision approach calculates the depth of an object from the camera using the formula given in Equation 12.

$$Z = f.B/(x - x') \tag{12}$$

In the given Equation 12, *Z* is the depth between a point P in the real world and the camera. *f* is the focal length of the cameras. *B* is the distance between the two lenses of the stereo-vision camera and *(x-x')* is the disparity. After the depth of each object is calculated, the objects are given spatial coordinates (*x, y, z*). Using these spatial coordinates, the distance between the objects and persons is calculated using Euclidean geometry. The formula is given in Equation 13.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{13}$$

In the given Equation 13, *d* is the distance between a pair of subject and object with spatial coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) respectively. This distance information is passed to the 'Hall's Proxemics Label Generation' module to generate a predicate for the scene graph.

3.2.3. Hall's Proxemics Label Generation

This module is responsible for generating the label based on the distance calculated from the previous module. The proxemics zones as specified by Hall and as shown in Figure 10 are divided into 4 categories. Let us assume a living room scene where the detected objects are a chair and a Television (TV), and the subject is a person. Let us say that the distance between the person and the TV is 1.1 meters and the distance between the person and the chair is 2.6 meters. Therefore, in this scenario, the label between the person and the TV would be "*personal"* and the label between the person and the chair would be *"social"* as shown in Figure 13. Algorithm 1 given below is used to determine Hall's Proxemics label between a person and a detected object.

Algorithm 1 Hall's Proxemics Label Generation.

Input: *distance between object and person in meters*
Output: *Hall's Proxemics Label*

if *distance > 0 and distance <= 0.5* **then**
    label ← *"intimate"*
else if *distance > 0.5 and distance <= 1* **then**
    label ← *"personal"*
else if *distance > 1 and distance <= 4* **then**
    label ← *"social"*
else if *distance > 4* **then**
    label ← *"public"*
else
    label ← *"invalid"*
end if
return *label*

The final output from the proposed architecture is a scene graph with a quantifiable predicate, which consists of the detected objects and persons, the distances between them, and the corresponding Hall's proxemics label.

To summarize, I propose a modified definition of a scene graph by introducing a quantifiable predicate in the triplet relations. The predicate is quantified using distance and labeled according to Hall's proxemics zones. Based on this definition, I propose an architecture capable of generating proxemics-enhanced scene graphs, or ProxeGraphs, in smart indoor environments. The ProxeGraph and the proposed architecture highlight my research contributions in integrating proxemics into smart indoor environments, thereby enhancing scene understanding and spatial analysis of humans and objects in smart home settings. In the next chapter, I will discuss the implementation of the proposed architecture.

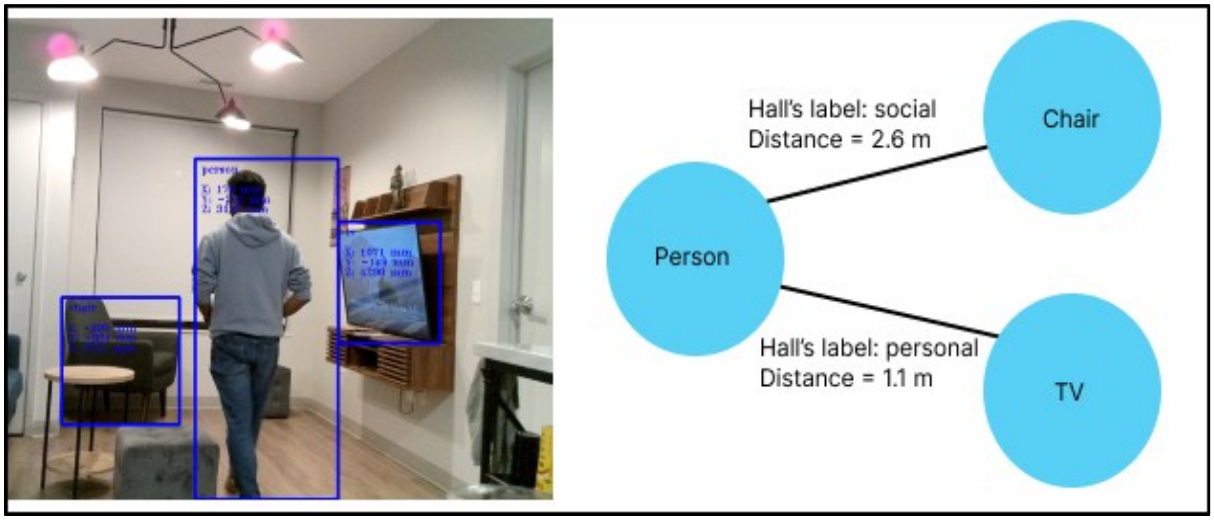


Figure 13 Example of Proxemics-enhanced Scene Graph.

4. IMPLEMENTATION

In this chapter, I will describe the implementation of the proposed approach using the Oak-D Lite camera. This camera is fully customizable using DepthAI's open-source API and SDK. According to the documentation provided by DepthAI [59], to set up an Oak-D Lite camera, a pipeline is required. To upload a custom pipeline to the Oak-D Lite camera, it needs to be connected to a host computer using a Universal Serial Bus (USB) cable. In this thesis work, the Oak-D Lite camera works as the input for the proposed architecture. The onboard Neural Network inferencing capability of this camera is used to detect objects and the depth capability of this camera is used to spatially locate the detected objects. As shown in Figure 14, the pipeline designed by me using Python and DepthAI's open-source Python API for the Oak Camera consists of the following classes: i) ColorCamera, ii) 2 MonoCamera, iii) StereoDepth, iv) SpatialDetectionNetwork, v) 3 XLinkOut.

I elaborate on the classes used in the pipeline in the following manner: In the 'Object Detection Module in Oak-D Lite Camera' paragraph of the 'Object Detection' subsection, I explain the usage of the 'ColorCamera' and 'SpatialDetectionNetwork' classes. In the 'Distance Estimation' subsection, I explain the usage of the 'MonoCamera' class, the 'StereoDepth' class, and the 'XLinkOut' class. The usage of the 'XLinkOut' class also extends to the 'ProxeGraph Generation Using Hall's Proxemics Label Generator' subsection.

The rest of this chapter is organized as: the first sub-section, 'Object Detection' describes the dataset curation, model training and model deployment in Oak-D Lite pipeline for object detection. The second sub-section, 'Distance Estimation' is dedicated to distance estimation and the third sub-section, 'ProxeGraph generation using Hall's Proxemics label

Generator' describes the implementation of the Hall's Proxemics label generation and scene graph generation with quantifiable predicates.

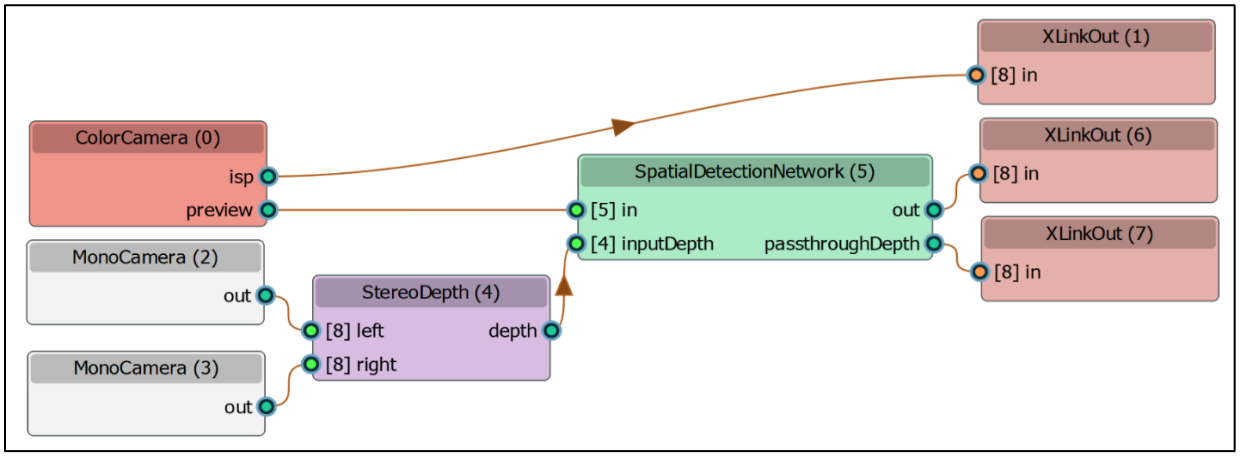


Figure 14 Oak-D Lite Camera Pipeline.

4.1. Implementing Object Detection

The object detection module of my proposed architecture is a crucial component, responsible for detecting, localizing, and classifying the various objects present in a given scene. In the following paragraphs, I will describe the formation of a dataset suitable for smart indoor environments, the training of an object detection model on this curated dataset, and finally, the deployment of the trained model on the Oak-D Lite camera pipeline.

4.1.1. Curating Dataset for Smart-home Environment

In this subsection, I will describe how I curated the MS-COCO [58] object detection dataset for smart-home environments. The MS-COCO dataset is a large-scale object detection, segmentation, and captioning dataset. The paper [58] defines 91 object categories; however, the actual dataset uses 80 object categories. The 80 object categories are classified into 12 superclasses as mentioned in Table 3. Out of these 12 superclasses, 8 superclasses are suitable

for smart home or indoor settings as the remaining superclasses contain objects typically not found in indoor environments, such as cars, airplanes, boats, traffic lights, and fire hydrants. Therefore, I included all the object categories from those 8 superclasses. Also, considering the most common pets and sports materials, I decided to include 2 classes from the “Animal” superclass and 4 classes from the “Sports” superclass. The selected object categories from the "Animal" superclass are “Dog” and “Cat” and the selected object categories from the "Sports” superclass are “Baseball bat”, “Baseball glove”, “Sports ball”, “Skateboard”. Thus, I have a smart indoor environment dataset with 53 object categories belonging to 10 superclasses. The curated dataset is shown in Table 4.

Table 3 MS-COCO Object Categories.

Superclass	Object Categories
Person	Person
Vehicle	Car, Airplane, Motorcycle, Bus, Train, Bicycle, Boat, Truck
Outdoor	Fire hydrant, Traffic light, Stop sign, Bench, Parking meter
Animal	Giraffe, Cat, Sheep, Horse, Cow, Elephant, Bird, Dog, Bear, Zebra
Accessory	Suitcase, Umbrella, Tie, Backpack, Handbag
Sports	Skateboard, Frisbee, Tennis racket, Sports ball, Surfboard, Baseball glove, Skis, Snowboard, Kite, Baseball bat
Kitchen	Bottle, Cup, Wine glass, Fork, Spoon, Knife, Bowl
Food	Orange, Donut, Broccoli, Hot dog, Cake, Banana, Sandwich, Carrot, Apple, Pizza

Table 3 continued

Superclass	Object Categories
Appliance	Toaster, Sink, Refrigerator, Oven, Microwave
Electronics	Keyboard, TV, Cell phone, Mouse, Remote, Laptop
Furniture	Potted plant, Chair, Toilet, Couch, Dining table, Bed
Indoor	Toothbrush, Teddy bear, Clock, Hair drier, Vase, Scissors, Book

Table 4 Smart-indoor Environment Dataset.

Superclass	Object Categories
Person	Person
Animal	Cat, Dog
Accessory	Tie, Suitcase, Umbrella, Backpack, Handbag
Sports	Baseball bat, Skateboard, Baseball glove, Sports ball
Kitchen	Bottle, Cup, Wine glass, Fork, Spoon, Knife, Bowl
Food	Orange, Donut, Broccoli, Hot dog, Cake, Banana, Sandwich, Carrot, Apple, Pizza
Furniture	Potted plant, Chair, Toilet, Couch, Dining table, Bed
Electronics	Keyboard, TV, Cell phone, Mouse, Remote, Laptop
Appliance	Toaster, Sink, Refrigerator, Oven, Microwave
Indoor	Toothbrush, Teddy bear, Clock, Hair drier, Vase, Scissors, Book

4.1.2. Training an Object Detection Model

In smart-home environments, real-time inference is crucial for ensuring swift and accurate responses to changing conditions. To meet this demand, deploying a real-time object detector is essential. Recent research [42] highlights YOLOv8 as the fastest and most accurate one-shot real-time object detector available, making it an ideal choice for efficient and effective smart-home systems. Its rapid processing capabilities enable timely recognition of objects, enhancing the overall performance and responsiveness of smart-home applications.

To optimize the functionality of smart home systems in real-time scenarios, I incorporated transfer learning into a pre-trained YOLOv8 architecture. Integrating transfer learning ensures a tailored and adaptive approach to object detection for the smart-home environment. The model was fine-tuned with a specific set of 53 object categories relevant to smart-home applications. These categories were curated from the MS-COCO dataset, ensuring that the selected objects are pertinent to the smart-home context. The selection process, detailed in the previous subsection, is crucial as the number of trainable parameters in a model is directly proportional to the number of classes selected. With more trainable parameters, the training time and the size of the trained model increase.

By leveraging transfer learning and focusing on a curated set of classes, the re-trained YOLOv8 achieved faster training and inference times, which are essential for real-time applications. Additionally, concentrating on specific object categories resulted in higher accuracy and better generalization. The curated selection from MS-COCO ensures that the model is not only efficient but also highly relevant to practical smart-home scenarios.

I used Ultralytics Hub [60] for model training as it provides cloud training support, which significantly enhances the training process. In the Ultralytics Hub's cloud training platform I used a single NVIDIA T4 16 GB GDDR6 Graphics Processing Unit (GPU) to train a pre-trained YOLOv8 model on my curated smart indoor dataset for 100 epochs with image size as 640x640 pixels, patience of 100 with RAM caching policy and automatic batch sizing. The training took a runtime of 26 hours 44 mins in the cloud. The results of the model training are discussed in the chapter '5. Performance Evaluation'. I have provided screenshots of the Ultralytics Hub dashboard and other training steps in Appendix B. In the next subsection, I describe how I deployed the YOLOv8 model trained on the curated indoor dataset in the Oak-D camera pipeline.

4.1.3. Object Detection Module in Oak-D Lite Camera

To deploy the trained model in the Oak-D Lite camera, I exported the trained model in ONNX format using Ultralytics Hub's export tool. During the export process, I used an image size of 640x640 pixels, applied FP16 quantization, and enabled model simplification. FP16 quantization ensures that the model works with 16-bit floating points, which helps scale the model size for the Oak-D Lite camera. A screenshot of the Ultralytics export tool is shown in Appendix B.

Next, I used Luxonis's Blob Converter tool [61] to convert the trained model into a supported format (.blob) for the Oak-D Lite camera. For model optimization, I used mean values of 127.5 and scale values of 255 for normalization of input values. These normalization values ensure that the input data is scaled appropriately, improving model accuracy. Screenshots of the Blob Converter tool is shown in Appendix C. After the conversion process,

the size of the trained model was reduced from 11.6 MB to 5.87 MB marking a 50% reduction in the model size. This significant reduction makes the model more lightweight and suitable for deployment on the Oak-D Lite camera. These steps were chosen to ensure that the model could run efficiently on the Vision Processing Unit (VPU) of the Oak-D Lite camera while maintaining high accuracy.

The custom model obtained through the training process described above is utilized by the 'SpatialDetectionNetwork' class in the Oak-D pipeline, as depicted in Figure 14 to detect objects from a given scene. The 'SpatialDetectionNetwork' class receives inputs from both the 'ColorCamera' and the 'StereoDepth' classes. It uses the RGB input frame from the 'ColorCamera' class to detect and classify the objects present in the scene.

Subsequently, it processes these detections by mapping the RGB frame to the depth frame provided by the 'StereoDepth' class. This mapping allows the 'SpatialDetectionNetwork' class to spatially localize the detected objects. The bounding boxes of the detected objects are mapped to the depth frame, providing spatial (x, y, z) coordinates for each detected object. This spatial information is crucial as these coordinates are used by the ‘Distance Calculation’ module in the proposed architecture. An output of the object detection and spatial mapping is shown in Figure 15. The figure shows an indoor scene where multiple objects, including a person, laptop, and chairs, have been detected and labeled using bounding boxes with their spatial coordinates (x, y, z) in millimeters. The labels provide detailed distance information for each object relative to the camera. In the next subsection, I describe the implementation of the ‘Distance Calculation’ module.

Figure 15 Output of Object Detection and Spatial Mapping.

4.2. Implementing Distance Calculation

As discussed in the ‘Background & Related Works’ and ‘Proposed Method: ProxeGraph’ chapters, I have used a stereo vision approach for estimating depth using cameras and I have selected Oak-D Lite stereo camera because of its stereo vision capabilities. In this subsection, I explain the working of the Oak-D Lite camera’s stereo depth capability and discuss the method of calculating distance between each pair of detected objects.

According to the DepthAI API documentation [59], the two 'MonoCamera' classes (2 and 3) shown in Figure 14 correspond to the left and right lenses of the Oak-D Lite camera and are responsible for capturing the frames and passing them to the 'StereoDepth' class. The 'StereoDepth' class accepts the frames captured by the left and right lenses as input. It then

performs rectification of the frame using a homography matrix. The rectified frames are then passed through a stereo matching algorithm and finally, post-processing filtering is applied to the frame. I have used default values for all the algorithms used in this class. The Oak-D Lite camera uses a left-handed coordinate system for all spatial coordinates. The output of this class is a disparity/depth map which is aligned with the RGB frame. This depth map is passed as an input to the 'SpatialDetectionNetwork'. The bounding box coordinates of the detected objects from the object detection neural network are mapped to the depth map, generating the spatial coordinates (x, y, z) for each detected object. A depth frame with detected objects and mapped spatial coordinates is shown in Figure 16. The figure depicts the same scene as in Figure 15, but with the addition of a depth image. The depth image includes the same bounding boxes and spatial coordinates of the detected objects as seen in the color image.

The 'XLinkOut' classes in the pipeline are responsible for transmitting these spatial coordinates, along with the captured frames and the list of detections, to the host computer to which the camera is connected. The host computer uses the list of detections and the corresponding spatial coordinates to calculate the distance between each pair of detected persons and objects using Euclidean geometry, as discussed in the ‘Distance Calculation’ module of the ‘Proposed Method: ProxeGraph’ chapter. The calculated distances are then passed to the ‘Hall’s Proxemics Label Generator’ module to label the distances between objects based on Hall’s proxemics zones using Algorithm 1.

Figure 16 Depth Map (right) from Oak-D Lite with Corresponding RGB Frame (left).

4.3. ProxeGraph Generation Using Hall's Proxemics Label Generator

As discussed in the 'Proposed Architecture' section of the 'Proposed Method: ProxeGraph' chapter, I use the list of detections and distance information received from the 'Object Detection' module and the 'Distance Calculation' module to form proxemics labels based on Hall's proxemics zones. This process generates a proxemics-enhanced scene graph. In this section, I will discuss the implementation of the generation of proxemics-enhanced scene graphs.

In my implementation using the Oak-D Lite camera, I receive the detected objects and their spatial information as a list of 'ImgDetection' objects. 'ImgDetection' is a class from DepthAI's Python library that contains the detected object label, its bounding box coordinates, and its spatial coordinates. The 'XLinkOut' class, numbered (6) in Figure 14, is responsible for transmitting this information from the camera to the host computer.

Using this list, I form pairs of person-object combinations according to Algorithm 2. Upon receiving the list of 'ImgDetection' objects from the Oak-D Lite camera, Algorithm 2 iterates over the list and modifies the label of each item by adding a count representing the quantity of that particular label present in the list. During the same iteration, it also separates the 'ImgDetection' objects with the ‘person’ class from other object classes, forming two separate lists—one containing only ‘person’ objects and the other containing non-person objects. At the end of the iteration, a Cartesian product of the list containing only ‘person’ objects with the list of non-person objects is taken. The output of Algorithm 2 is a list of pairs of person-object combinations present in the list of 'ImgDetection' objects obtained from the Oak-D Lite camera. For each of these pairs, I apply Algorithm 3 to calculate the distance between the subject and the object in the pair. Algorithm 3, upon receiving the list of person-object pairs from Algorithm 2, iterates over the list to calculate the Euclidean distance using the spatial coordinates of the person and the object. In the same iteration, this distance is used to create a proxemics label for the person-object pair using Algorithm 1. Following proxemics label generation, Algorithm 3 forms an instance of the ‘Triplet’ class, which I developed, where the attributes of the class are a subject, a predicate, and an object. The output of Algorithm 3 is a list of instances of the ‘Triplet’ class, where the subject is one of the detected persons, the object is one of the detected objects, and the predicate is a combination of the distance and a proxemics label representing each person-object pair. This list of triplets constitutes the proxemics-enhanced scene graph for a given scene.

The final output of the Hall's Proxemics Label Generator is a scene graph with quantifiable predicates, as described in the 'proposed architecture' section. This scene graph is

stored in the host computer's secondary memory encoded as a JavaScript Object Notation (JSON) file. A graphical representation of the list of triplets is shown in Figure 17. The figure depicts the same scene as in Figure 15, but with the addition of a graphical representation of the proxemics-enhanced scene graph. The right side of the figure shows the ProxeGraph, with the detected objects as nodes of the graph and the proxemic relationships as edges, illustrating their spatial relationships within the scene. A JSON representation is shown in Figure 18. The figure depicts the structure of a ProxeGraph in JSON encoding on the left side and a sample snippet of the ProxeGraph illustrated Figure 17. The JSON is structured as an object with 'time' as a string representing the timestamp of the data, 'triplets' as a list of containing multiple 'Triplet' objects, where each triplet represents a relationship between an object and a subject with a specific predicate. Each triplet contains an 'object', which is a detected object with a label (a string for the object's label), a 'bbox' (an object for the bounding box coordinates with four numbers: x1, y1, x2, y2), and 'spatialCoordinates' (an object for spatial coordinates with three numbers: x, y, z). The predicate is an object with distance - a number representing the distance in millimeters between the object and the subject and hall's label - a string representing Hall's proxemics label for the relationship. The subject is another object with the same structure as the triplet 'object'.

By integrating the object detection and depth estimation capabilities of the Oak-D Lite camera, this implementation of the proposed methodology applies proxemics in smart homes to offer a more intuitive and accessible way of interacting with smart home devices. This approach improves the overall user experience and makes smart homes more accommodating to a diverse range of users. To summarize, in this chapter I explained the implementation of the

proposed architecture as follows: the ‘Object Detection’ module is implemented by training a YOLOv8 architecture on a curated indoor environment and deploying it in the Oak-D Lite camera with the help of the 'ColorCamera' and 'SpatialDetectionNetwork' classes. The ‘Distance Calculation’ module is implemented using the 'MonoCamera' and 'StereoDepth' classes in the Oak-D Lite camera. The ‘Hall’s Proxemics Label Generation’ module is implemented on the host computer using Python code based on Algorithm 1, Algorithm 2, and Algorithm 3. The 'XLinkOut' classes in the Oak-D Lite camera are responsible for data transfer between the host computer and the camera. In the next chapter, I evaluate the performance of my implementation of the proposed architecture.

Algorithm 2 Form Person-Object Combinations.

Input: *list of ImgDetection objects from Oak camera*
Output: *list of Person-Object combinations*

1. *counts* ← empty mapping
2. *personList* ← *empty list*
3. *objectList* ← *empty list*
4. **for each** *item* **in** *ImgDetection list* **do**:
5. *counts[item.label]* ← *counts[item.label]* + 1
6. *item.label* ← ***concatenate***(*item.label*, *counts[item.label]*)
7. **if** *item.label* == *‘person’* **then***:*
8. add *item* to *personList*
9. **else**
10. add *item* to *objectList*
end if
11. **end for**
12. *combined_list* ← ***cartesian_product****(personList, objectList)*
12. **return** *combined_list*

Algorithm 3 ProxeGraph Generation.

Input: *list of combined person-object pairs*
Output: *ProxeGraph*

1. *proxeGraph* ← empty list
2. **for each** *pair* **in** *combined_list* **do**:
3. *distance* ← **calculate_euclidean_distance**(*pair.person.spatialCoordinate*, *pair.object.spatialCoordinate*)
4. *hall_label* ← **Hall's Label Generator**(*distance*)
5. *triplet* ← **Triplet**(*pair.person*, (*distance*, *hall_label*), *pair.object*)
6. add *triplet* to *proxeGraph*
7. **end for**
8. **return** *proxeGraph*

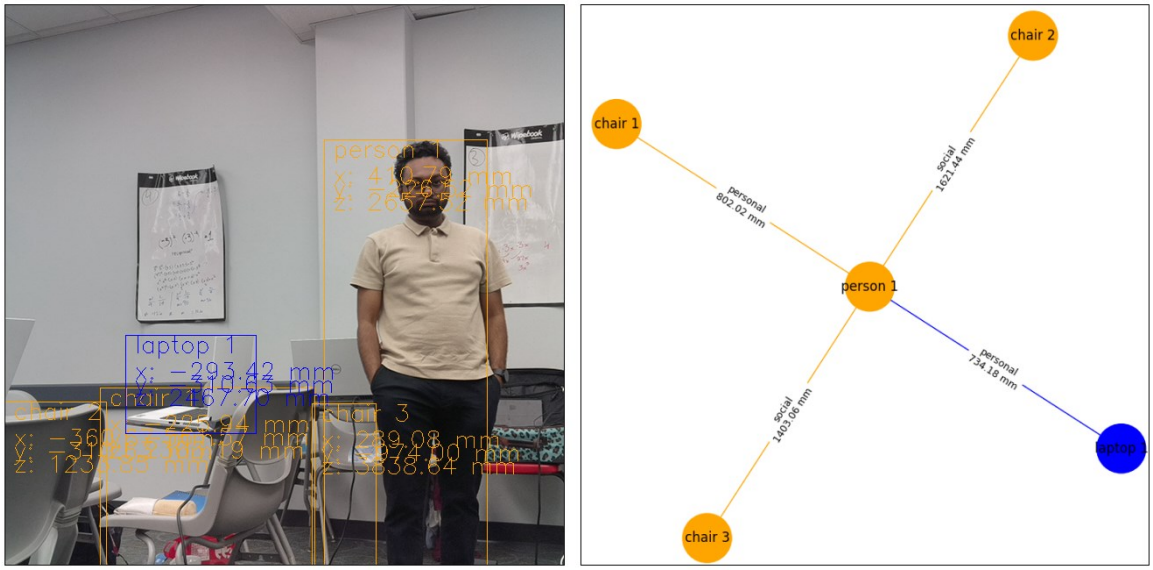
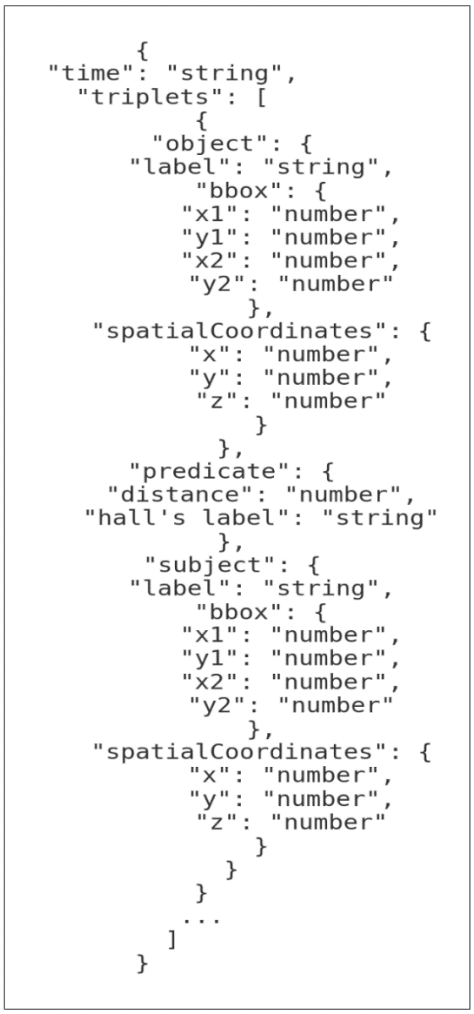


Figure 17 ProxeGraph - Proxemics-enhanced Scene Graph.



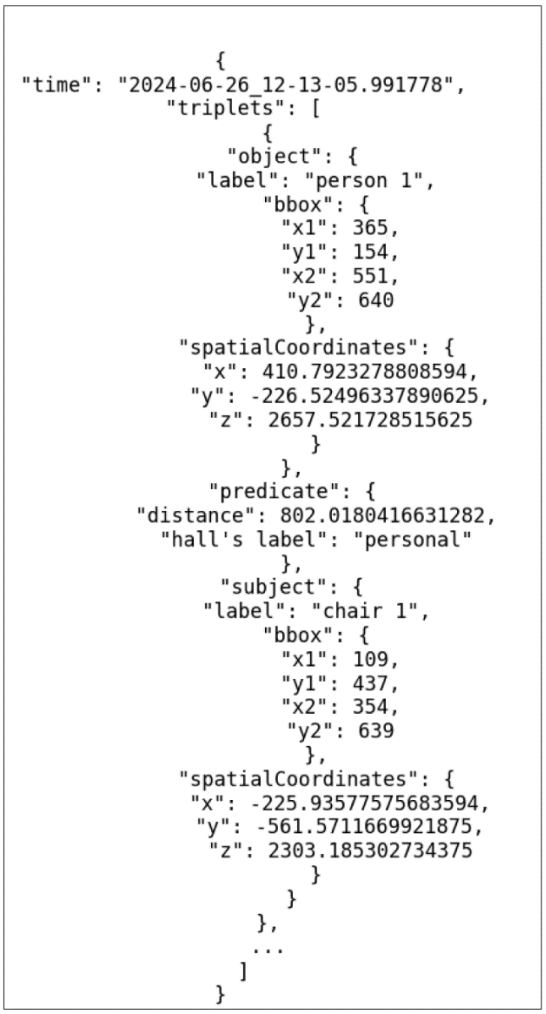


Figure 18 ProxeGraph JSON Representation.

5. PERFORMANCE EVALUATION

In this section, I will explain the results and performances of the different submodules of the proposed architecture and its implementation in this thesis work. This section is divided into 4 subsections – the first subsection describes the object detection training performance, the second subsection describes the proxemics-enhanced scene graph generation, the third subsection describes the impact of viewpoint in ProxeGraph and the final subsection describes the application of ProxeGraph towards pre-defined context detection in smart homes.

5.1. Evaluation of Object Detection

As described in 'Proposed Method: ProxeGraph' chapter, I used a YOLOv8 model trained on a dataset curated for indoor smart-environments object detection in the implementation of my proposed architecture. The curated dataset containing 53 classes, as detailed in 'Proposed Method: ProxeGraph' chapter, plays a significant role in finetuning the proposed architecture for indoor environments. I trained the model for 100 epochs. I measured 3 losses: box loss, classification loss and the deformable convolutional layer loss. The box loss (box_loss) is the bounding box regression loss measuring the error in predicted bounding box compared to the ground truth. Lower box loss means the predicted bounding boxes are more accurate. The error between predicted class probabilities and the ground truth classed for each detected object in the image is measured by classification loss (cls_loss). A model with lower classification loss means it is predicting the class of an object more accurately. The deformable convolutional layer loss (dfl_loss) measures the error in deformable convolutional layers which improves model's capability to detect objects across different scales and aspect ratios. A lower dfl_loss indicates that the model is better at handling object deformations and variations in

appearance. At the end of 100th epoch, the training loss for box (box_loss) is 1.11, for classification (cls_loss) is 1.175 and for deformable convolutional layer (dfl_loss) is 1.179. In case of validation, the losses are 1.125, 1.227 and 1.166 respectively. All three losses showed a trend of decreasing over epochs, which is a good sign indicating that the model is learning. There was a significant drop early in training (before epoch 5), followed by a plateau, which is common as the model starts to converge. The patterns for the validation loss are similar to the training losses, but it is important to note that the losses here are generally higher than the training losses. This is expected since the validation set consists of unseen data and serves as a good indicator of how well the model might perform on new, unseen data. Along with the losses, I also monitored metrices such as mAP, mAP50-95, precision and recall. After training for 100 epochs, I found that the mAP50 metrics is at 0.469. This means that the model is correctly identifying and localizing the objects about 46.9% of the time when a 50% overlap with the ground truth bounding boxes is considered a correct detection. At the same time, since the mAP50-95, which is the average measure of model's performance across IoU thresholds from 0.50 to 0.95 is at 0.331, it can be inferred that the performance of the model drops when a stricter localization criterion is applied. This is a common issue because it is more challenging to have a high degree of overlap for correct detections, but it shows that the model has room for improvement in terms of precision of bounding box predictions. In object detection, especially with a large number of classes (53 in this case), achieving high mAP values can be challenging. The mAP at IoU=0.5 is decent, suggesting that the model can detect objects with a fair amount of accuracy when a lower threshold for overlap is set. However, the mAP for IoU thresholds from 0.5 to 0.95 is lower, which is expected as it is a more stringent measure. This indicates

that while the model can detect objects, the exactness of the bounding boxes could be improved. The graphs for training and validation loss are given in Figure 19a and Figure 19b respectively and the graph for metrices is given in Figure 19c.

From Figure 19, it can be noticed that the model has improved over the epochs. Although a crucial part of the proposed architecture, improving object detection is not my primary goal. Therefore, I decided to use this decently trained model for carrying out further experiments which are described in the following subsections.

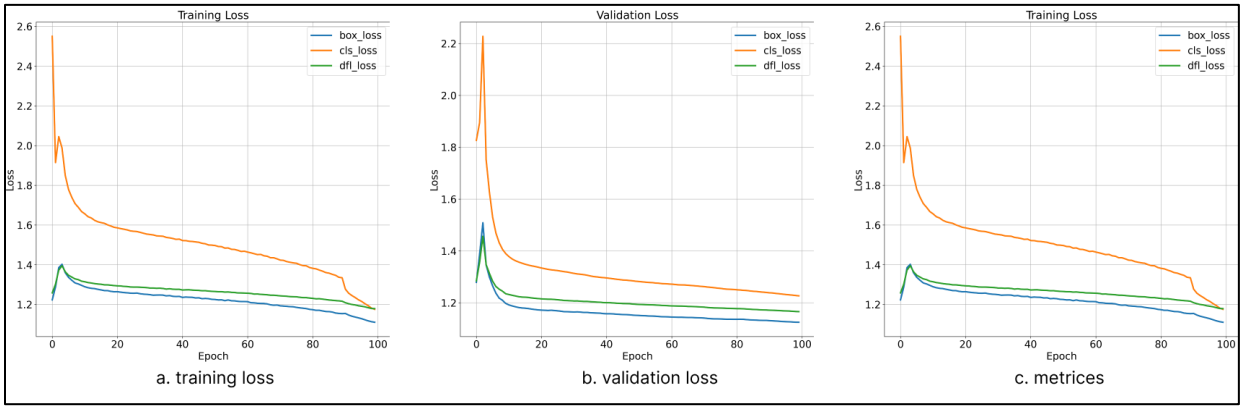


Figure 19 Object Detection Model Training on Curated Smart-indoor Dataset Using GPU for 100 Epochs Using Ultralytics Hub.

5.2. ProxeGraph Generation

I implemented the proposed ProxeGraph architecture in four different environments: my apartment kitchen, living room, my office, and a lounge area. The generated proxemics-enhanced scene graphs for these environments are shown in Figure 20- Figure 23. In Figure 20, a kitchen environment features a person, microwave, bottle, and oven. Figure 21 presents a living room with a person, TV, laptop, and chair. Figure 22 showcases an office setting with a person, multiple chairs, and TV screens. Figure 23 displays an indoor lounge area with several individuals, a chair, and a laptop. For each of these figures, the ProxeGraph shown on the right

subfigure of each figure, visualizes the relationships among the individuals and objects, conveying the spatial dynamics within the scene. These experimental environments contained furniture such as chairs and desks, electronics such as monitors and laptops, appliances such as microwaves and ovens, and accessories such as bottles. The lounge area also contained multiple people, providing a diverse set of proxemics zones between persons and objects. Each environment serves different functions and presents unique challenges for the proposed architecture.

The proposed architecture successfully detected most of the objects present in a frame and generated corresponding proxemics-enhanced scene graphs. The variety of these environments showcases the architecture's adaptability to different scenarios, object arrangements, types of objects, and the presence of multiple objects of the same kind. This variety implies robustness, enabling the system to maintain performance across different settings.

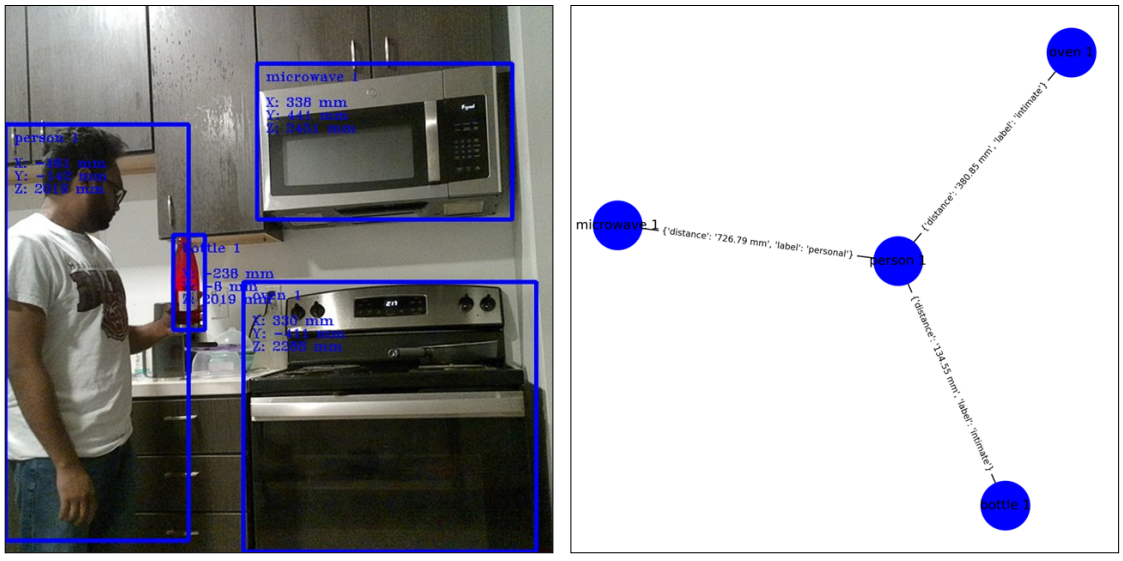


Figure 20 Kitchen Environment.

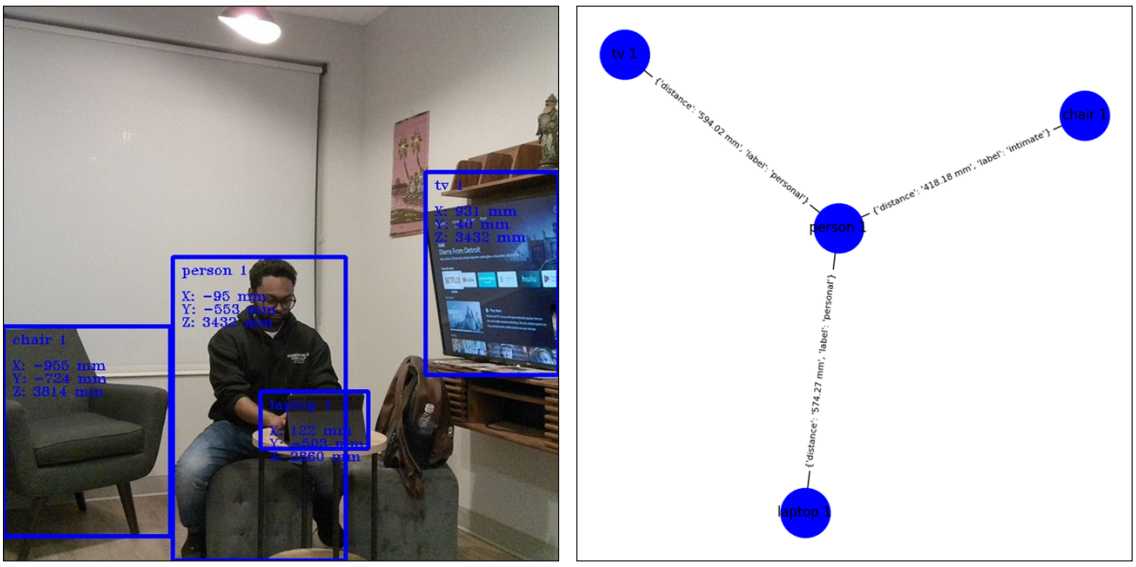


Figure 21 Living Room Environment.

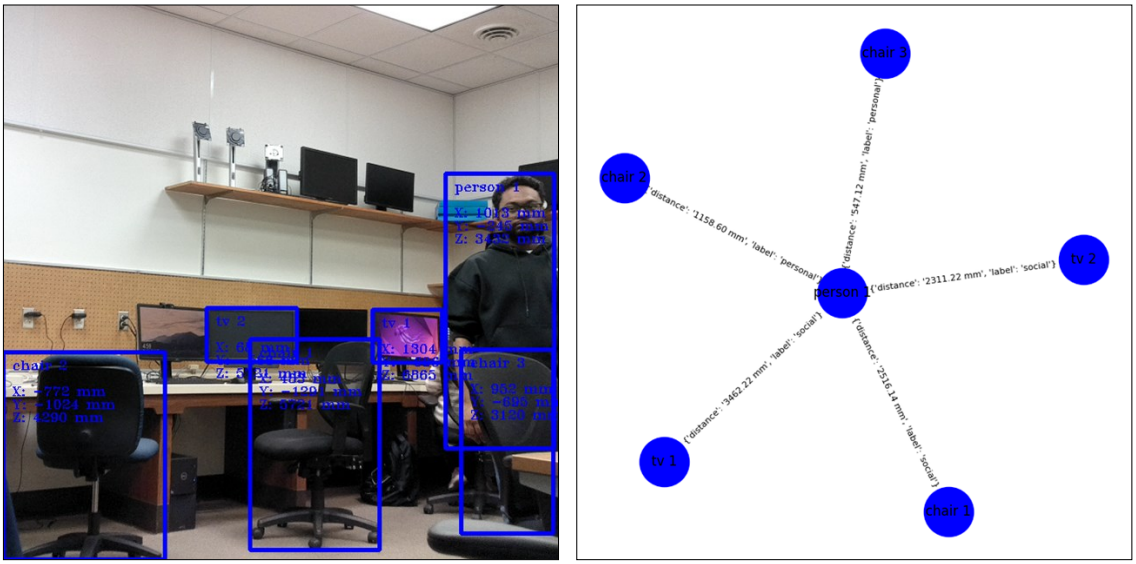


Figure 22 Office Environment.

These environments, with their complex and varying layouts, add to the diverse settings in which the ProxeGraph architecture has been observed to function effectively. This underscores the system's adaptability to various interior designs and its potential utility. The

architecture's ability to identify different objects and people, gather spatial information, localize detected objects in 3D space, calculate distances between pairs of persons and objects, and classify these distances to provide proxemics labels demonstrates its complexity, adaptability, and versatility.

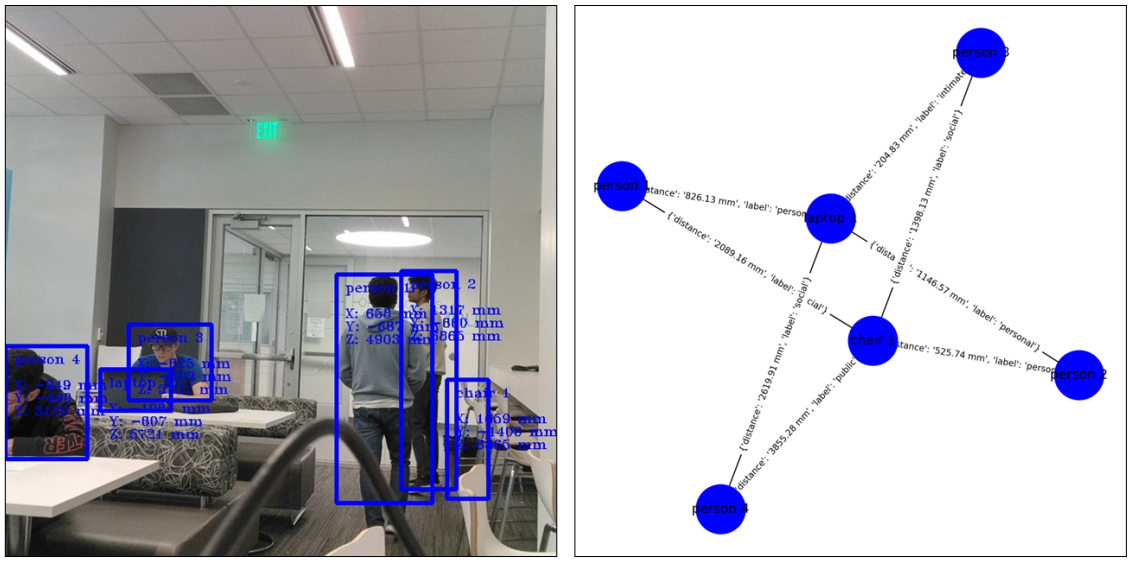


Figure 23 Lounge Environment.

To further evaluate the ProxeGraph architecture, I timed the deployed system by running the architecture on a host computer with 8 GB RAM and Intel i7 processor and recorded the time required to generate a scene graph corresponding to an input frame. The results of this evaluation are shown in Figure 24, which shows the mean time required to process varying numbers of triplets. The number of triplets is a combination of the number of persons detected in the scene multiplied by the number of objects detected in the scene.

The average processing time exhibits variability across different numbers of triplets. It is apparent that the processing time increases as the number of triplets grows, reflecting the expected computational load associated with handling more complex scenes. There is a

noticeable upward trend in the mean processing time with an increasing number of triplets. This trend suggests that the architecture may face scalability challenges, as more triplets result in longer processing times. The weighted average time required to generate a proxemics-enhanced scene graph across varying numbers of triplets was 0.000344 seconds, depicted by a red dashed line on the graph. This weighted average time indicates that, on average, the ProxeGraph architecture performs consistently.

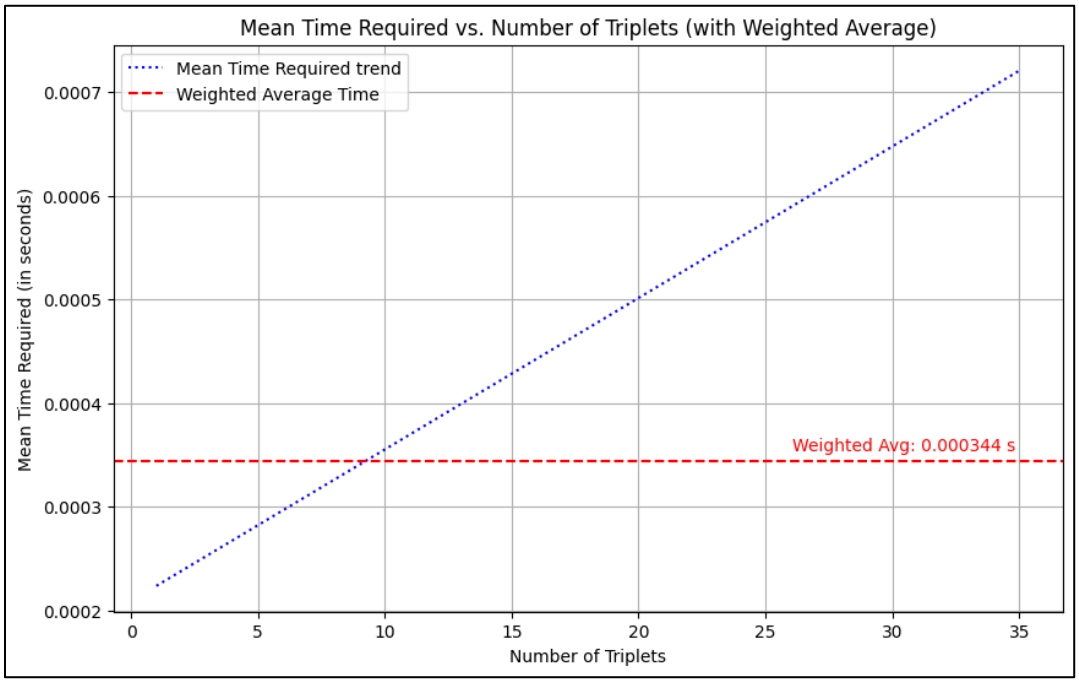


Figure 24 ProxeGraph Generation Timing.

The system's consistent performance, as indicated by the weighted average time, suggests robustness in various smart home scenarios. This reliability is crucial for practical applications, where consistent and predictable performance is necessary to maintain user satisfaction and system dependability. Additionally, the low weighted average time required suggests that the proposed ProxeGraph architecture is well-suited for real-time applications, which is crucial for smart home contexts.

5.3. Viewpoint Impacts on ProxeGraph

To evaluate the change in ProxeGraph and determine whether it remained consistent when the same scene was viewed from different perspectives, I deployed the ProxeGraph architecture using two different Oak-D Lite cameras positioned in two different corners of a living room environment. Figure 25 shows a sample of detected objects from the first camera perspective, which include ‘person 1’, ‘couch 1’, and ‘chair 1’. The corresponding ProxeGraph indicates that ‘person 1’ is at a ‘personal’ distance from ‘chair 1’ and at a ‘social’ distance from ‘couch 1’. Similarly, Figure 26, taken from the second camera perspective, detects ‘person 1’, ‘tv 1’, ‘chair 1’, and ‘chair 2’. The corresponding ProxeGraph shows that ‘person 1’ is at a ‘personal’ distance from ‘chair 1’ and at ‘social’ distances from both ‘chair 2’ and ‘tv 1’. Both images were captured simultaneously from different perspectives. Observations of the generated

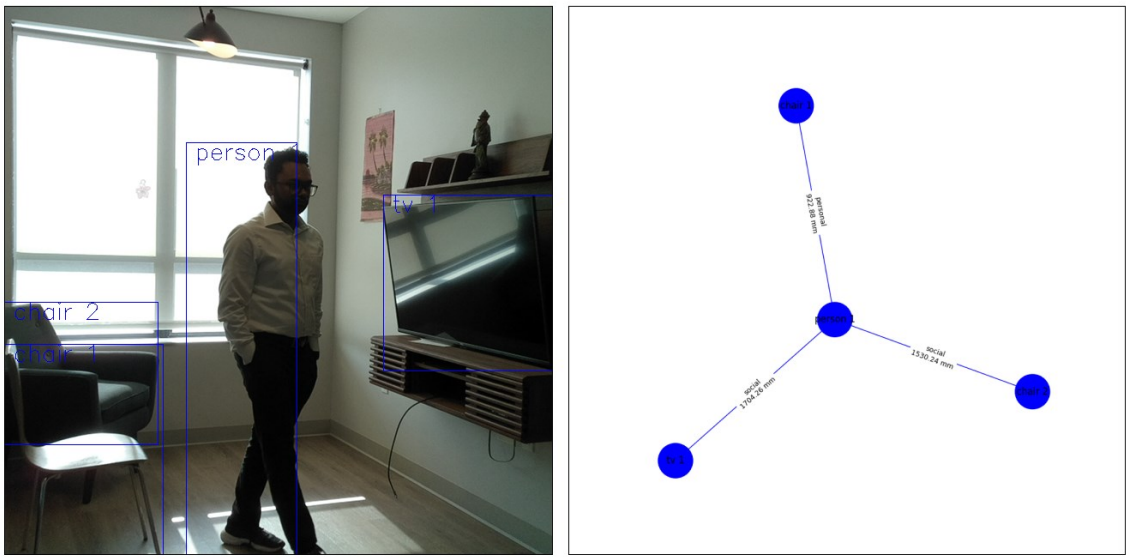


Figure 25 Viewpoint of a Living Room Environment from Oak-D Lite Camera 1.

ProxeGraphs revealed that while the detected objects varied based on the camera locations, the common objects in both perspectives maintained the same relationships derived

from Hall's proxemics zones. This indicates that the ProxeGraph architecture can maintain consistent spatial relationships for common objects across different viewpoints. However, it is evident that depending on the field of view and the position of the cameras, as well as the size of the environment, multiple cameras may be required to generate a comprehensive ProxeGraph. The observations of the generated ProxeGraphs also confirmed that the common objects across different perspectives had consistent relationships. The absolute distance variation between the detected objects was within the bounds of a 2% error margin, as specified in the Oak-D Lite camera's documentation [32]. The consistency across different perspectives, where common objects like 'person 1' and 'chair 1' maintain the same proxemics relationships as shown in Figure 25 and Figure 26, indicates the robustness of the ProxeGraph architecture in preserving spatial relationships despite changes in viewpoint. Although initial observations show promising results, an elaborate evaluation of this problem requires a more comprehensive study of object identification and scene regeneration, which is outside the scope of this research.

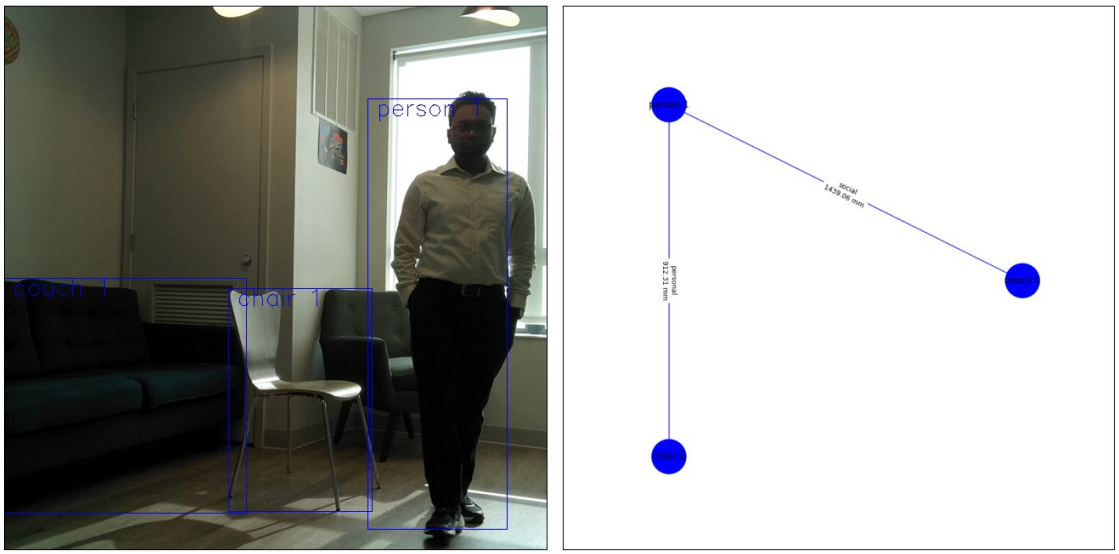


Figure 26 Viewpoint of a Living Room Environment from Oak-D Lite Camera 2.

5.4. Proof-of-Concept: Application of ProxeGraph towards Context Detection in Smart Home Environments

To evaluate the efficacy of ProxeGraph towards context detection in smart-home environments, I deployed the ProxeGraph architecture in two distinct indoor environments: a kitchen and a multi-person environment. I created separate context policies manually for these environments to define three contexts in decreasing order of importance: dangerous, concerning, and normal.

5.4.1. Kitchen Environment Deployment

For the kitchen environment, I defined the following policies:

1. If person is in intimate zone with an oven, it is a dangerous context.
2. If person is in intimate zone with a refrigerator, it is a dangerous context.
3. If person is in personal zone with a microwave, it is a normal context.

In the kitchen environment, as shown in Figure 27 and Figure 28, the ProxeGraph framework is correctly able to identify the dangerous and normal contexts as highlighted by red and blue colored nodes respectively. In Figure 27, ‘person 1’ is detected in close proximity to ‘oven 1’. The generated proxemics-enhanced scene graph identifies the distance as 389.16 mm, falling within the ‘intimate’ zone. However, the distance between the ‘person 1’ and the ‘microwave 1’ is 973.36 mm which falls within the ‘personal’ zone thereby making it a normal context between the person and the microwave. Similarly, in Figure 28, 'Person 1' is detected near ‘refrigerator 1’, with a measured distance of 435.33 mm, also within the ‘intimate’ zone and thereby making it a dangerous context.

5.4.2. Multi-Person Environment Deployment

For the multi-person environment, I defined the following policies:

1. If person is in intimate zone with a cup, it is a dangerous context.
2. If person is in personal zone with a chair, it is a concerning context.
3. If person is in social zone with a chair, it is a concerning context.
4. If person is in any zone with a television monitor, it is a normal context.
5. If person is in any zone with a bottle, it is a normal context.

In the multi-person environment, as shown in Figure 29, the ProxeGraph framework successfully identified the various contexts, despite the complex layout and the presence of multiple people and objects.

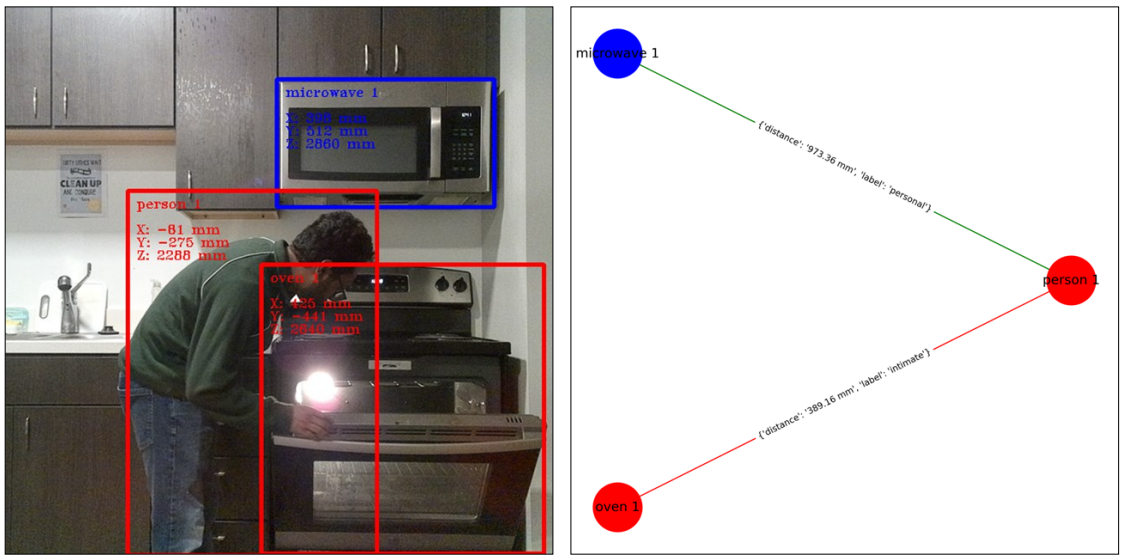


Figure 27 Kitchen Environment Deployment: Oven Interaction.

Additionally, ‘Person 1’ is in the ‘social’ zone with ‘Chair 1’, ‘Person 2’ is in the ‘social’ zone with ‘Chair 2’, and ‘Person 4’ is in the ‘public’ zone with ‘TV 1’, creating normal and concerning contexts as per the defined policies.

To further evaluate the architecture, I timed the context detection by the system based on the generated ProxeGraph. The timing graph is given in Figure 30. Although the average

context detection time depicted by the blue dotted line shows an increasing trend, the weighted average time of 0.00314 seconds depicted by the dotted red line, indicates consistent performance, underscoring the system's capability for real-time applications.

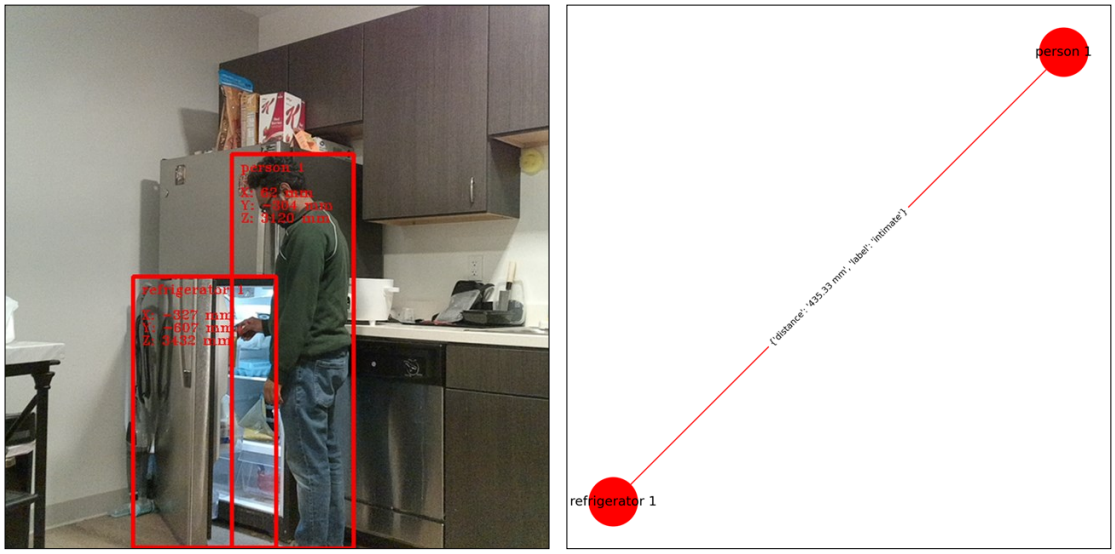


Figure 28 Kitchen Environment Deployment: Refrigerator Interaction.

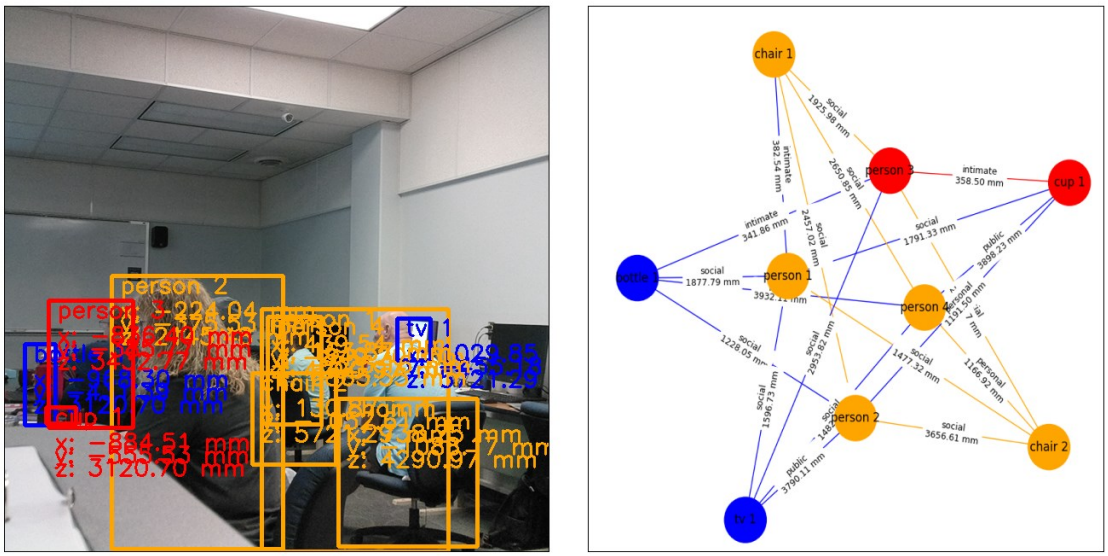


Figure 29 Multi-person Environment Deployment.

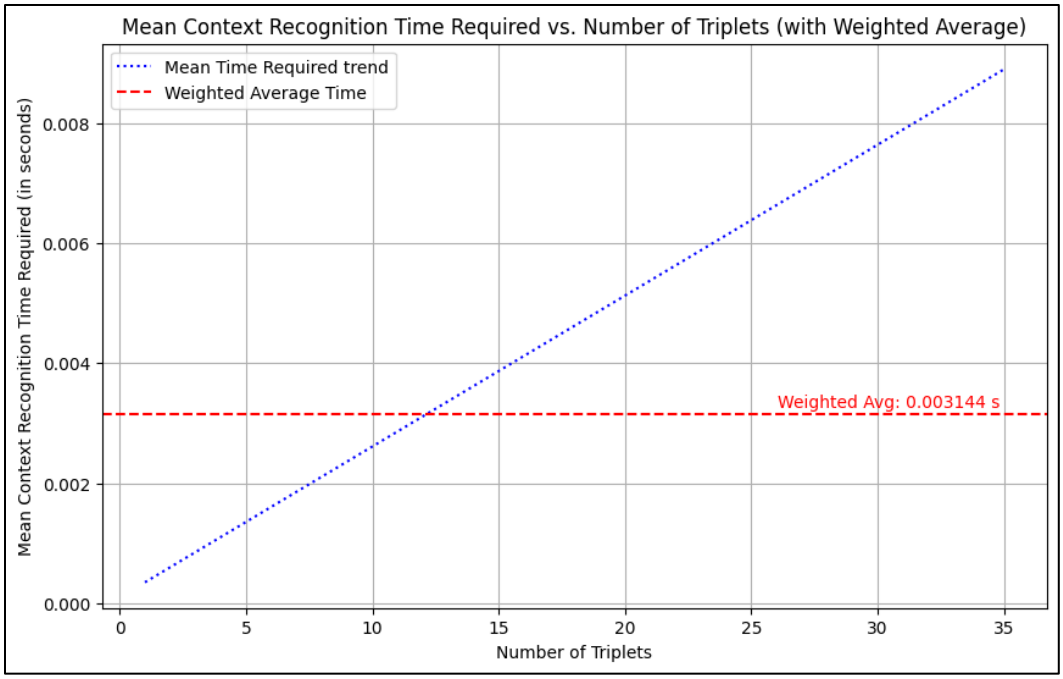


Figure 30 Context Recognition Timing.

The ProxeGraph architecture's deployment in both kitchen and multi-person environments demonstrates its adaptability and robustness in detecting simple pre-defined context such as ‘dangerous’, ‘concerning’ and ‘normal’. By accurately identifying objects, measuring proxemic distances, and applying predefined context policies, the system ensures safety and efficiency. The consistent performance and low policy generation time further emphasize its practicality for real-time applications in indoor-smart environments. This proof of concept highlights the potential of proxemics-enhanced scene graphs to enhance human-computer interactions, making the ProxeGraph architecture a valuable asset in diverse indoor-smart environments.

The results presented in this chapter demonstrate that combining proxemics with scene graphs is not only feasible but also effective for non-verbal communication in smart home environments. The proposed architecture effectively utilizes the spatial relationships between

users and objects to generate proxemics-enhanced scene graphs. By leveraging proxemic zones defined by anthropologist Edward T. Hall, the system can interpret predefined contexts based on users' physical proximity to various smart home devices. This novel integration provides a robust framework for enabling non-verbal interactions, particularly beneficial for individuals with speech impairments or those who prefer not to use verbal commands, thereby answering my first research question. From the proof of concept discussed earlier, it is evident that introducing a quantifiable distance attribute enriches the predicate in scene graph triplets. This addition allows for a more precise and detailed representation of spatial relationships, thereby improving scene understanding and spatial analysis in smart homes.

6. CONCLUSION

In this research, I attempted to find the feasibility of combining the non-verbal communication method proxemics with scene graphs to introduce a passive avenue of interaction between users and devices in smart homes, and to explore the application of this combination for context inference in smart-home environments. Additionally, I investigated whether introducing a quantifying attribute would enrich the scene-graph triplet. To this end, I proposed the ProxeGraph architecture, which generates proxemics-enhanced scene graphs by utilizing Hall's Proxemics Zones as relationship labels between smart home users and objects.

The primary goal was to determine if proxemics could be integrated with scene graphs to enhance non-verbal communication within smart homes. This integration aimed to provide a structured representation of spatial relationships between users and objects, enabling intuitive and automated interactions. The performance evaluations conducted across various smart home scenarios confirmed that this combination is not only feasible but also highly effective.

In the kitchen environment, the ProxeGraph architecture accurately detected and labeled interactions with appliances and utensils, which can enable context-aware automation such as adjusting cooking settings or turning off devices when not in use. In the living room, the system demonstrated its ability to generate proxemics-enhanced scene graph by correctly labelling the interaction zone between a user and the devices present. This can potentially be used to manage entertainment and relaxation settings by detecting user proximity to furniture and electronic devices, thereby facilitating automated adjustments like changing TV channels or dimming lights. The lounge environment, characterized by multiple users and social

interactions, further highlighted the architecture's robustness in generating proxemics-enhanced scene graph across different environmental layouts and varying number of users.

The inclusion of distance as a quantifying attribute in the scene graph triplets enriched the predicates by allowing for more precise spatial analysis and improved interaction labeling. For example, detecting a person near the oven in the kitchen as an "intimate" proxemics zone indicates potential interaction with cooking activities. The ProxeGraph system successfully generated proxemics-enhanced scene graphs across diverse settings, demonstrating robustness and adaptability. The average time to process and generate a scene graph was 0.000344 seconds, which indicates that the system is well-suited for real-time applications.

The applicability of the proxemics-enhanced scene graph in simple pre-defined context detection such as ‘dangerous’, ‘concerning’ or ‘normal’ was demonstrated by the proof-of-concept work provided in this research. These ProxeGraphs provided a comprehensive understanding of user interactions within the smart home, enabling the system to maintain spatial awareness and infer contexts. The ability to detect potentially hazardous situations, such as a person near an oven showcases the practical applications of the context detection capabilities. Using this proxemics-enhanced scene graph it is therefore possible to adapt settings based on user presence, such as turning devices on and off, adjusting room temperature or lighting or changing the operating mode of various devices based on the proxemics zone of the user to those devices. The average time required to detect context from a given proxemics-enhanced scene graph was 0.00314 seconds, which indicates that the system is fast and well-suited for real-time applications.

While this research has laid a foundation in integrating proxemics in smart home environments, several areas warrant further exploration. Deployment of the ProxeGraph architecture across two distinct viewpoints revealed that although the architecture is consistent in maintaining relationships between commonly detected objects, multiple cameras are required to capture a comprehensive scene, depending on the room size and camera viewpoints. In-depth evaluation of ProxeGraphs generated from different viewpoints in the same environment requires further exploration in object identification and scene regeneration. Future work could also focus on integrating the ProxeGraph architecture with traditional scene graph generation architectures, thereby enhancing the richness of spatial understanding in smart environments. Additionally, customizing the proxemics-based interactions to cater to individual user preferences and behaviors can further personalize the smart home experience such as adjusting the thermostat settings based on the proximity of users to heating or cooking stations, or automatically locking doors when children move towards unsafe zones. Combining proxemics with other non-verbal communication methods like gesture recognition could create a more comprehensive and intuitive interaction system. For example, a user with limited mobility can have a defined zone near their bed or chair where simple gestures can control multiple devices, improving their comfort, independence, and safety. A user can control smart home devices simply by moving closer or farther away from them. Moreover, by defining specific movements or gestures within proxemic zones, users can perform tasks such as adjusting the thermostat or playing music. Improving the performance of ProxeGraph generation is another future direction.

Despite its promising outcomes, this research has some limitations. The reliance on stereo vision cameras for distance estimation may limit the system's applicability in environments where such equipment is not feasible. Future research could explore alternative, cost-effective distance estimation methods such as Light Detection And Ranging (LIDAR) cameras or cameras with Infrared (IR) sensors. As the number of objects and interactions increases, the computational load for generating and processing scene graphs also grows. Ensuring scalability while maintaining real-time performance is a significant challenge that needs to be addressed. The current implementation uses a fixed set of proxemic zones as defined by Hall. Expanding and refining these zones to better capture the nuances of human-object interactions in diverse settings could enhance the system's accuracy and usability.

In conclusion, my research provides a novel approach to enhancing human-computer interactions in smart homes through the integration of proxemics and scene graphs. The ProxeGraph system demonstrates the potential of non-verbal communication method proxemics to create more intuitive, accessible, and responsive smart environments. Future advancements and refinements in ProxeGraph can further bridge the gap between technology and user needs, making smart homes smarter.

7. REFERENCES

[1] D. Ruby, "65 Voice Search Statistics For 2023 (Updated Data)," Demand Sage, April 2023. [Online]. Available: https://www.demandsage.com/voice-search-statistics/. [Accessed April 2023].

[2] S. A. Khowaja, K. Dahri, M. A. Kumbhar and A. M. Soomro, "Facial expression recognition using two-tier classification and its application to smart home automation system," in *2015 International Conference on Emerging Technologies (ICET)*, 2015.

[3] M. R. Abid, E. M. Petriu and E. Amjadian, "Dynamic Sign Language Recognition for Smart Home Interactive Application Using Stochastic Linear Formal Grammar," *IEEE Transactions on Instrumentation and Measurement,* 2015.

[4] X. Han and M. A. Rashid, "Gesture and voice control of Internet of Things," in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, 2016.

[5] D. Kim and D. Kim, "An Intelligent Smart Home Control Using Body Gestures," in *2006 International Conference on Hybrid Information Technology*, 2006.

[6] M. Ghaffar, S. R. Sheikh, N. Naseer and F. Ahmed, "Assistive Smart Home Environment using Head Gestures and EEG Eye Blink Control Schemes," in *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, 2021.

[7] K. Zhang and Y. Zhang, "SmartPose: An Intelligent Household Appliances Controller Based on Visual Recognition of Human Postures," in *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 2020.

[8] M. O. A. Aqel, A. Alashqar and A. Badra, "Smart Home Automation System Based on Eye Tracking for Quadriplegic Users," in *2020 International Conference on Assistive and Rehabilitation Technologies (iCareTech)*, 2020.

[9] A. Klaib, N. Alsrehin, W. Melhem and H. Bashtawi, "IoT Smart Home Using Eye Tracking and Voice Interfaces for Elderly and Special Needs People," *Journal of Communications,* July 2019.

[10] Z. A. Wassan, M. S. H. Talpur, A. Oad, R. Sarwar, A. L. S. H. Talpur, F. Talpur, T. Nuzhat and A. Oad, "IoT Based Smart Home for Paralyzed Patients through Eye Blink," *International Journal of Advanced Trends in Computer Science and Engineering,* April 2021.

[11] H. Malik and A. Mazhar, "EyeCom-An Innovative Approach for Computer Interaction," *Procedia Computer Science,* 2019.

[12] H. Medjahed, D. Istrate, J. Boudy, J.-L. Baldinger and B. Dorizzi, "A pervasive multi-sensor data fusion for smart home healthcare monitoring," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, 2011.

[13] A. Fleury, M. Vacher and N. Noury, "SVM-Based Multimodal Classification of Activities of Daily Living in Health Smart Homes: Sensors, Algorithms, and First Experimental Results," *IEEE Transactions on Information Technology in Biomedicine,* 2010.

[14] E. T. Hall, The silent language, Anchor Press, 1959.

[15] T. Ballendat, N. Marquardt and S. Greenberg, "Proxemic interaction: designing for a proximity and orientation-aware environment," *ACM International Conference on Interactive Tabletops and Surfaces,* 2010.

[16] D. Surie, B. Baydan and H. Lindgren, "Proxemics Awareness in Kitchen As-a-Pal: Tracking Objects and Human in Perspective," in *2013 9th International Conference on Intelligent Environments*, 2013.

[17] H. Li, G. Zhu, L. Zhang, Y. Jiang, Y. Dang, H. Hou, P. Shen, X. Zhao, A. Shah and M. Bennamoun, "Scene Graph Generation: A comprehensive survey," *Neurocomputing,* 2024.

[18] U. of Minnesota, Communication in the Real World: An Introduction to Communication Studies, U. of Minnesota Libraries Publishing, Ed., University of Minnesota Libraries Publishing, 2016.

[19] C. K. Goman, The Silent Language of Leaders: How Body Language Can Help--or Hurt--How You Lead, John Wiley & Sons., 2011.

[20] S. Littlejohn and K. Foss, Encyclopedia of Communication Theory, Thousand Oaks: SAGE Publications, Inc., 2009.

[21] D. Chandler and R. Munday, *nonverbal communication,* Oxford University Press, 2011.

[22] L. A. Samovar, E. R. McDaniel and R. E. Porter, "Nonverbal Communication: The Messages of Action, Space, Time, and Silence," in *Communication between cultures*, Wadsworth/Cengage Learning, 2010.

[23] D. Chandler and R. Munday, *oculesics,* Oxford University Press, 2011.

[24] D. Chandler and R. Munday, *chronemics,* Oxford University Press, 2020.

[25] A. Rosebrock, *Measuring distance between objects in an image with opencv,* PyImageSearch, 2021.

[26] A. Rahman, A. Salam, M. Islam and P. Sarker, "An Image Based Approach to Compute Object Distance," *International Journal of Computational Intelligence Systems,* March 2012.

[27] asadullahdal, *Realtime distance estimation using OpenCV - Python,* GeeksforGeeks, 2023.

[28] AiPhile, *Distance(webcam) estimation with single-camera opencv-python,* MLearning.ai, 2023.

[29] A. Tupper and R. Green, "Pedestrian Proximity Detection using RGB-D Data," in *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2019.

[30] Intel® RealSenseTM Depth and Tracking Cameras, "Depth Camera D435," [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/. [Accessed February 2023].

[31] e-consystems, "Tara - 3D Stereo Camera for Depth Sensing," [Online]. Available: https://www.e-consystems.com/3D-USB-stereo-camera.asp#. [Accessed February 2023].

[32] Luxonis, "OAK-D Lite," [Online]. Available: https://shop.luxonis.com/products/oak-d-lite-1?variant=42583102456031. [Accessed February 2023].

[33] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[34] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[35] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

[36] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *arXiv*, 2015.

[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[39] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," in *arXiv*, 2017.

[41] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, "Detectron2," in *Facebook Research*, 2019.

[42] G. Jocher, A. Chaurasia and J. Qiu, "Ultralytics YOLO," January 2023. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed March 2023].

[43] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[44] D. Wang, D. Beck and T. Cohn, "On the Role of Scene Graphs in Image Captioning," in *Proceedings of the Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, 2019.

[45] M. Y. Yang, W. Liao, H. Ackermann and B. Rosenhahn, "On support relations and semantic scene graphs," *ISPRS Journal of Photogrammetry and Remote Sensing,* 2017.

[46] N. Silberman, D. Hoiem, P. Kohli and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Computer Vision – ECCV 2012*, Berlin, 2012.

[47] F. Xue, S. Xu, C. He, M. Wang and R. Hong, "Towards efficient support relation extraction from RGBD images," in *Information Sciences*, 2015.

[48] Z. Jia, A. Gallagher, A. Saxena and T. Chen, "3D-Based Reasoning with Blocks, Support, and Stability," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[49] G. Yin, L. Sheng, B. Liu, N. Yu, X. Wang, J. Shao and C. C. Loy, *Zoom-Net: Mining Deep Feature Interactions for Visual Relationship Recognition,* 2018.

[50] Y. Li, W. Ouyang, B. Zhou, K. Wang and X. Wang, *Scene Graph Generation from Objects, Phrases and Region Captions,* 2017.

[51] Y. Zhu, S. Jiang and X. Li, "Visual relationship detection with object spatial distribution," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 2017.

[52] S. Sharifzadeh, S. M. Baharlou, M. Berrendorf, R. Koner and V. Tresp, "Improving Visual Relation Detection using Depth Maps," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[53] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik and S. Savarese, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[54] J. Wald, H. Dhamo, N. Navab and F. Tombari, "Learning 3D Semantic Scene Graphs From 3D Indoor Reconstructions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[55] Z. Wang, B. Cheng, L. Zhao, D. Xu, Y. Tang and L. Sheng, "VL-SAT: Visual-Linguistic Semantics Assisted Training for 3D Semantic Scene Graph Prediction in Point Cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[56] A. Rosinol, A. Gupta, M. Abate, J. Shi and L. Carlone, "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans," in *arXiv*, 2020.

[57] M. Qi, W. Li, Z. Yang, Y. Wang and J. Luo, "Attentive Relational Networks for Mapping Images to Scene Graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[58] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, *Microsoft COCO: Common Objects in Context,* 2015.

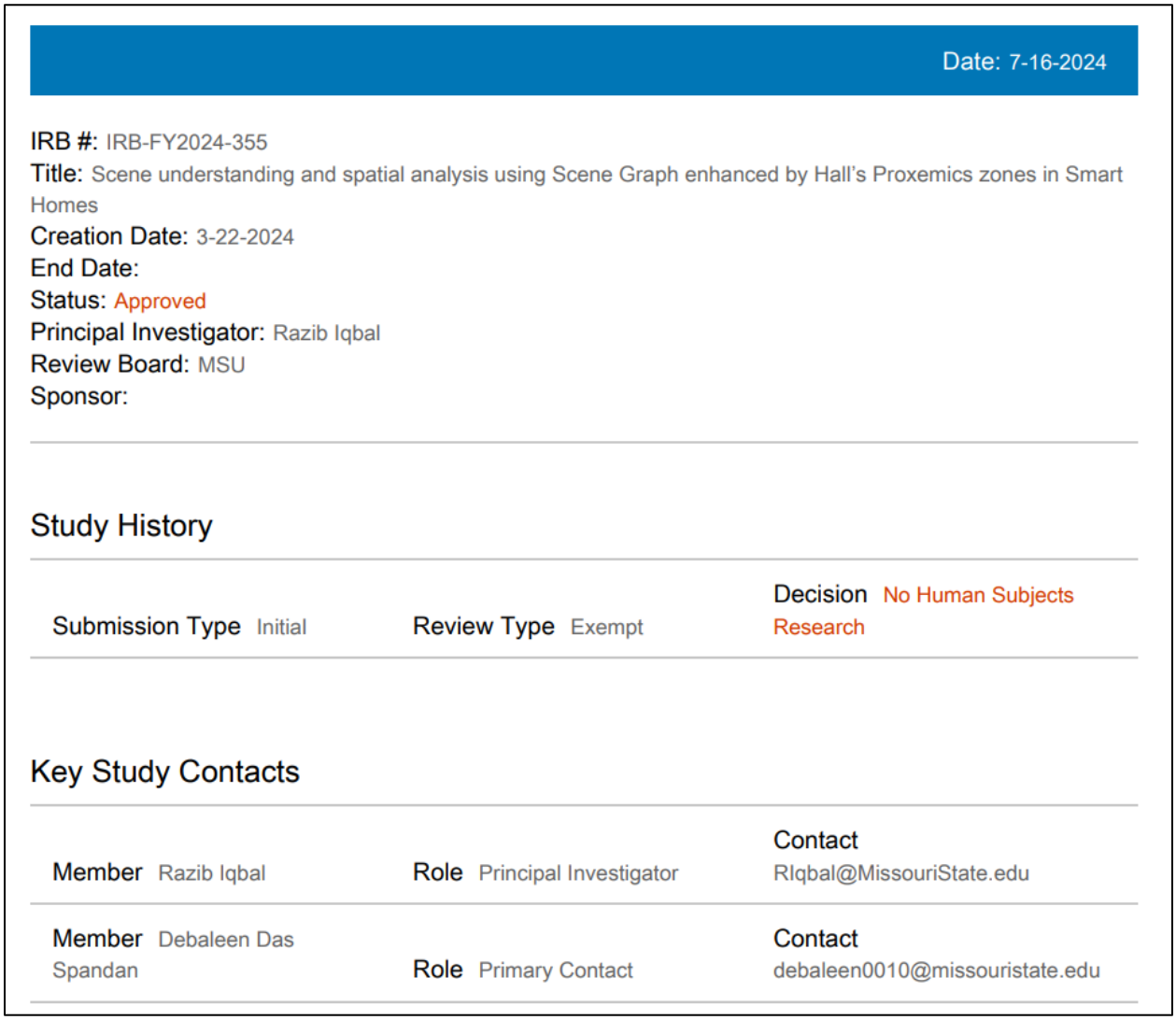
[59] Luxonis, "First Steps with DepthAI," [Online]. Available: https://docs.luxonis.com/software/depthai/manual-install#first-steps-with-depthai. [Accessed February 2023].

[60] Ultralytics, "Ultralytics Hub," [Online]. Available: https://hub.ultralytics.com/. [Accessed January 2024].

[61] Luxonis BlobConverter, "Luxonis Blob Converter," [Online]. Available: https://blobconverter.luxonis.com/. [Accessed July 2023].

8. APPENDICES

8.1. Appendix A: IRB Approval



Date: 7-16-2024

IRB #: IRB-FY2024-355
Title: Scene understanding and spatial analysis using Scene Graph enhanced by Hall's Proxemics zones in Smart Homes
Creation Date: 3-22-2024
End Date:
Status: Approved
Principal Investigator: Razib Iqbal
Review Board: MSU
Sponsor:

Study History

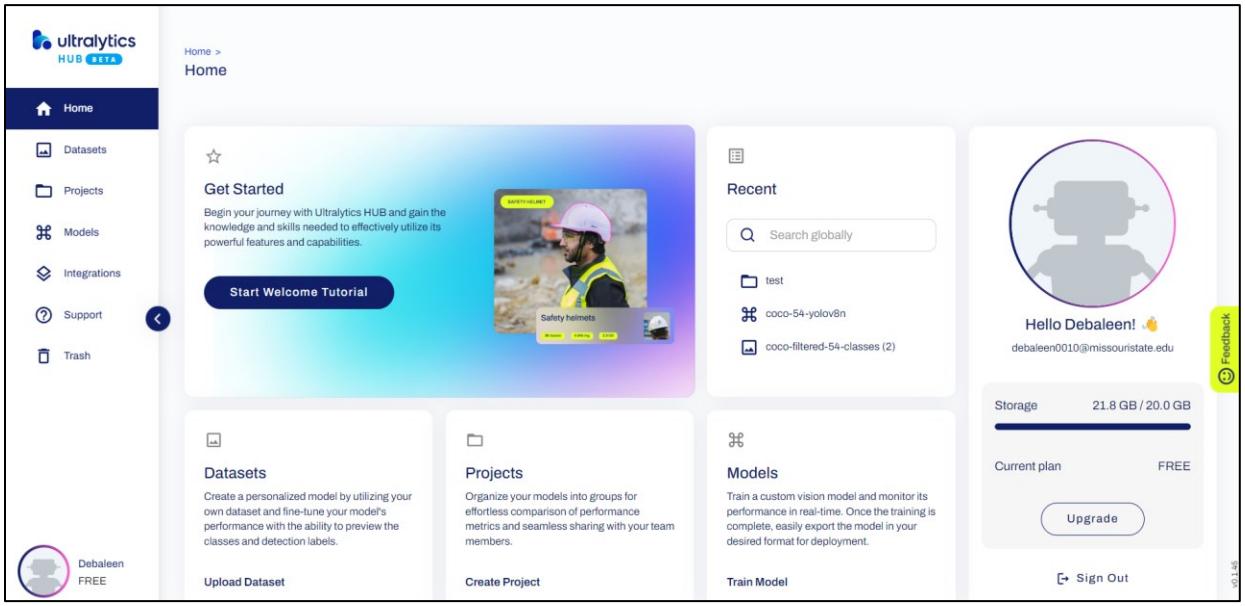
Submission Type	Review Type	Decision
Initial	Exempt	No Human Subjects Research

Key Study Contacts

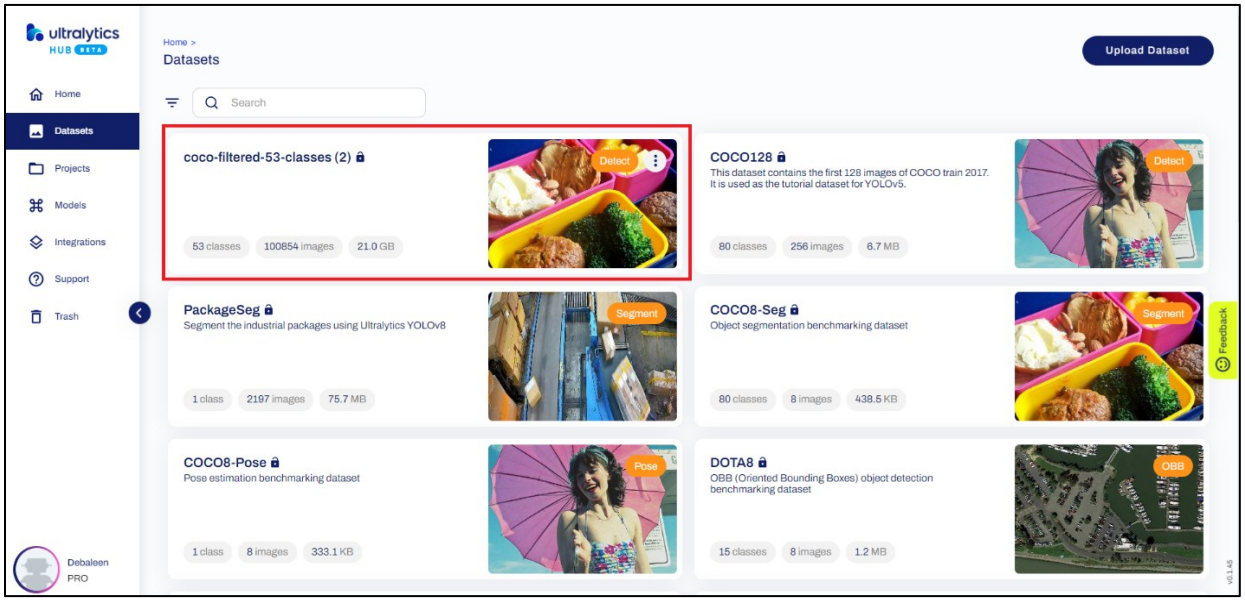
Member	Role	Contact
Razib Iqbal	Principal Investigator	RIqbal@MissouriState.edu
Debaleen Das Spandan	Primary Contact	debaleen0010@missouristate.edu

Appendix A 1 IRB Exemption

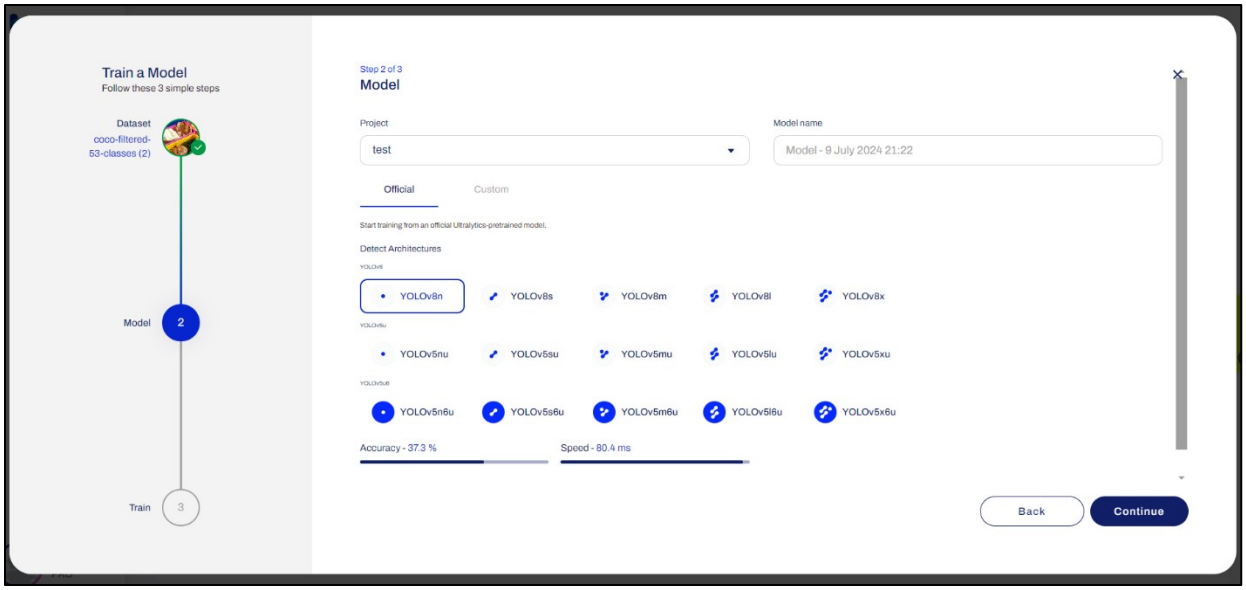
8.2. Appendix B: Ultralytics Hub Screenshots



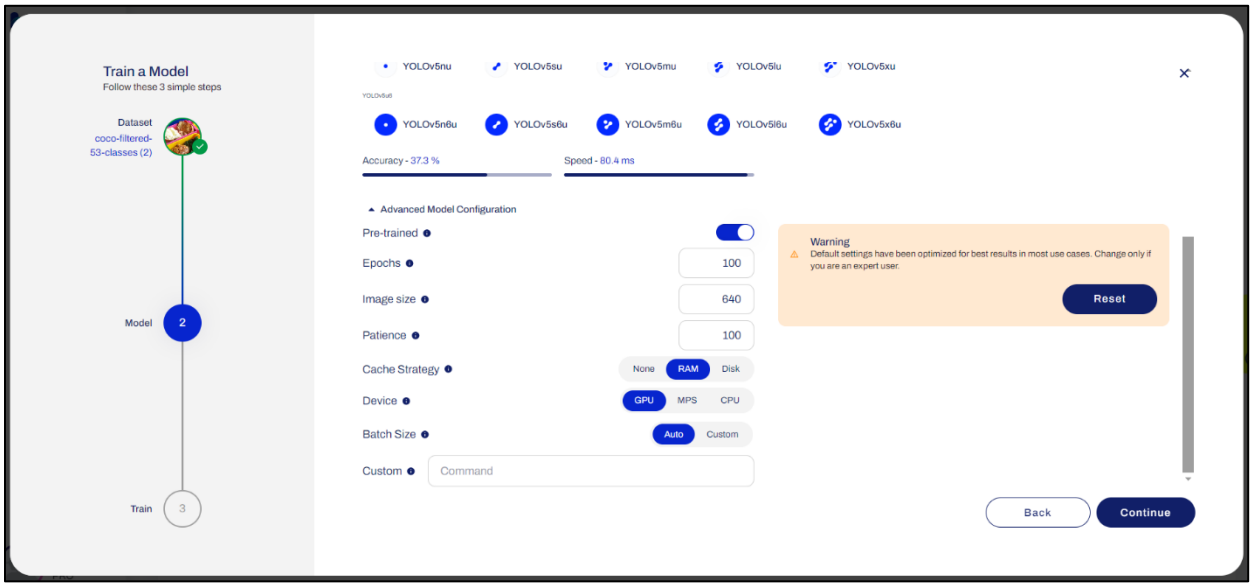
Appendix B 1 Ultralytics Hub - dashboard.



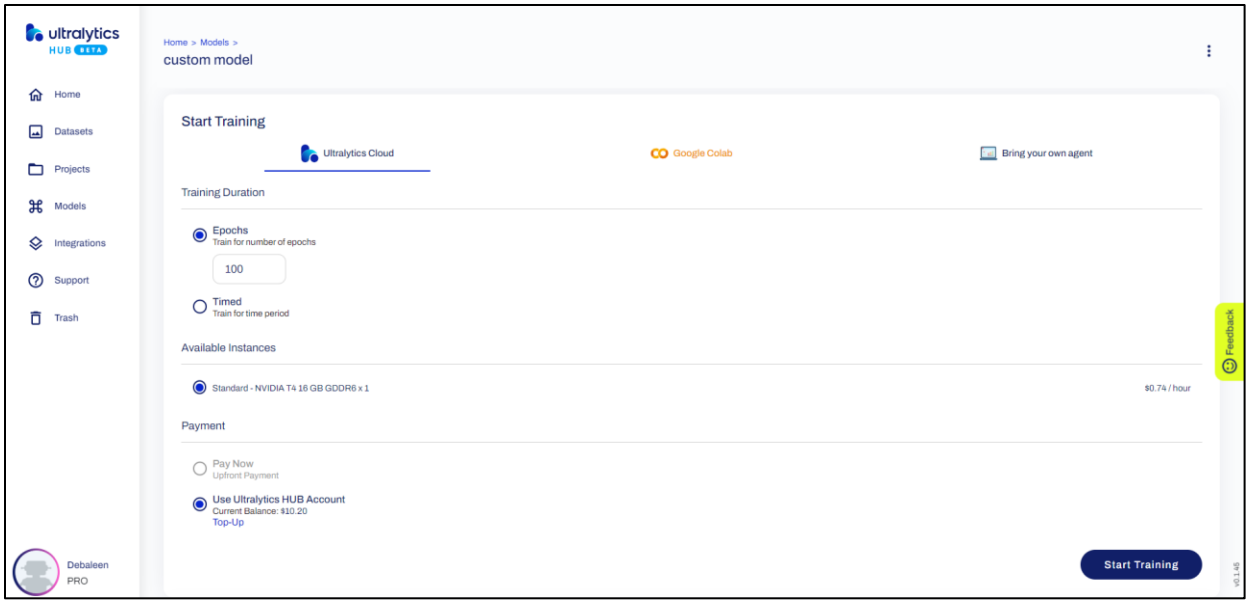
Appendix B 2 Ultralytics Hub - Datasets dashboard with the curated indoor dataset highlighted.



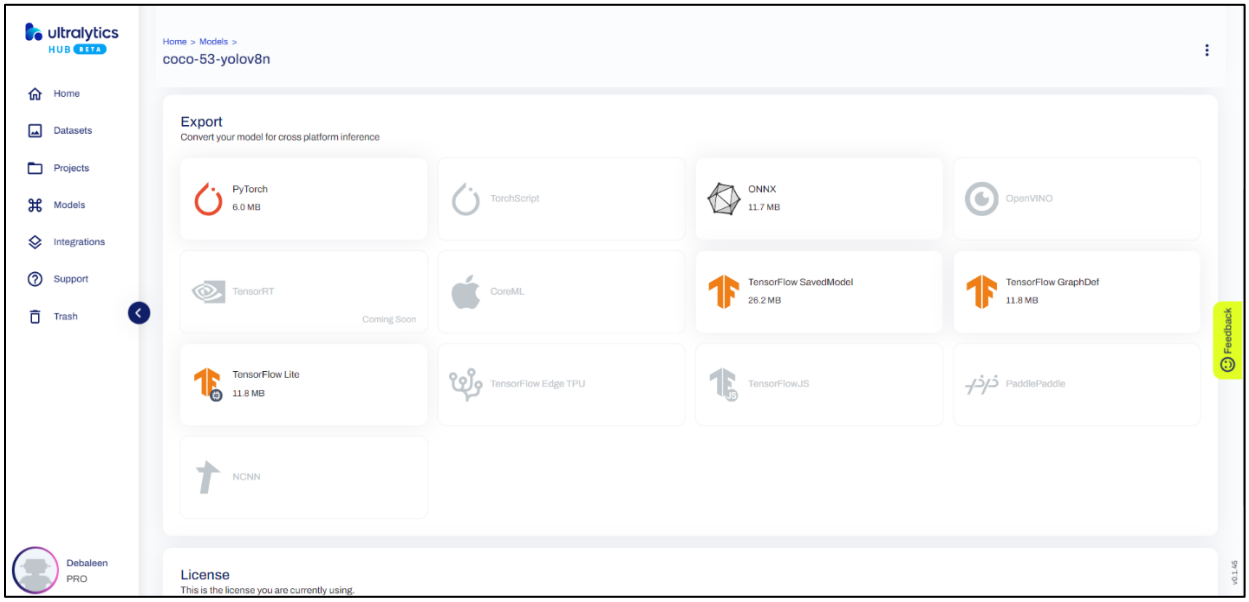
Appendix B 3 Ultralytics Hub - Training a YOLOv8 model step 1.



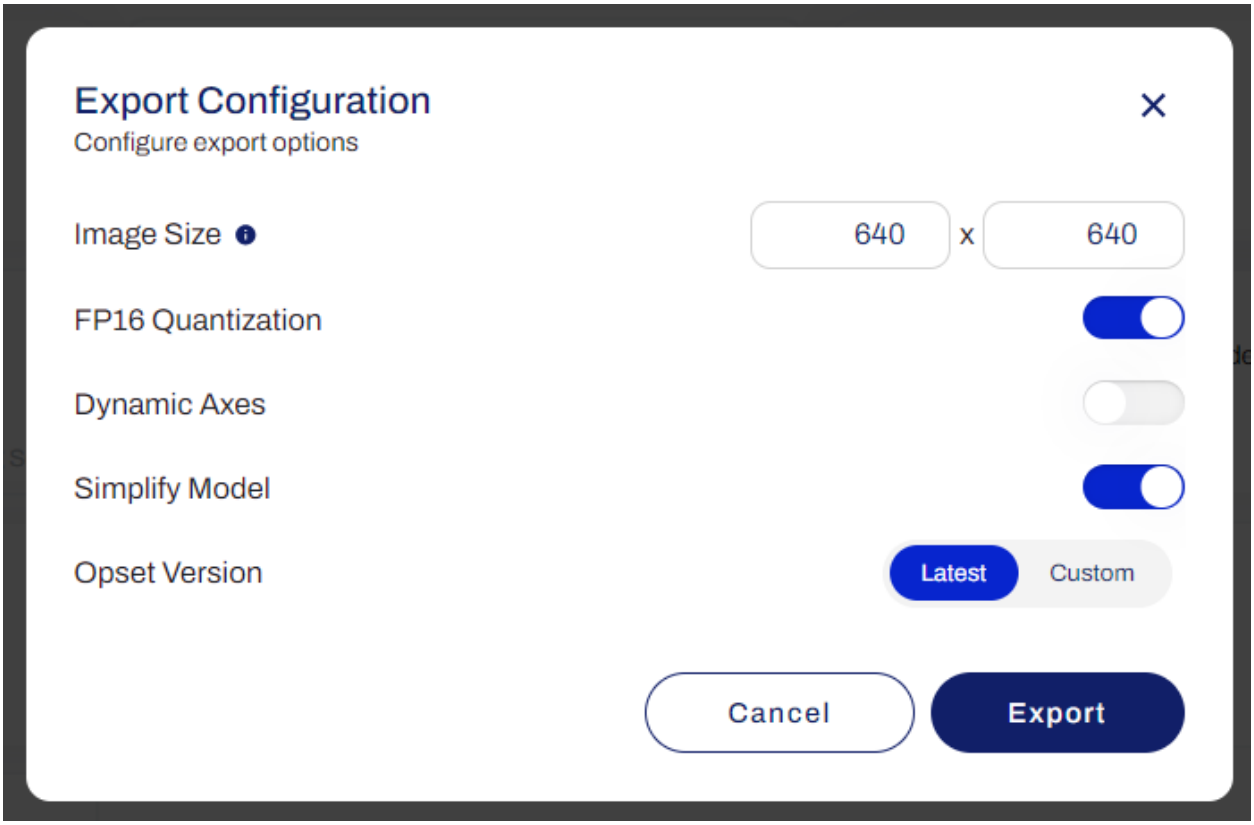
Appendix B 4 Ultralytics Hub - Model training parameters.



Appendix B 5 Ultralytics Hub - Cloud GPU selection and final step to start training.

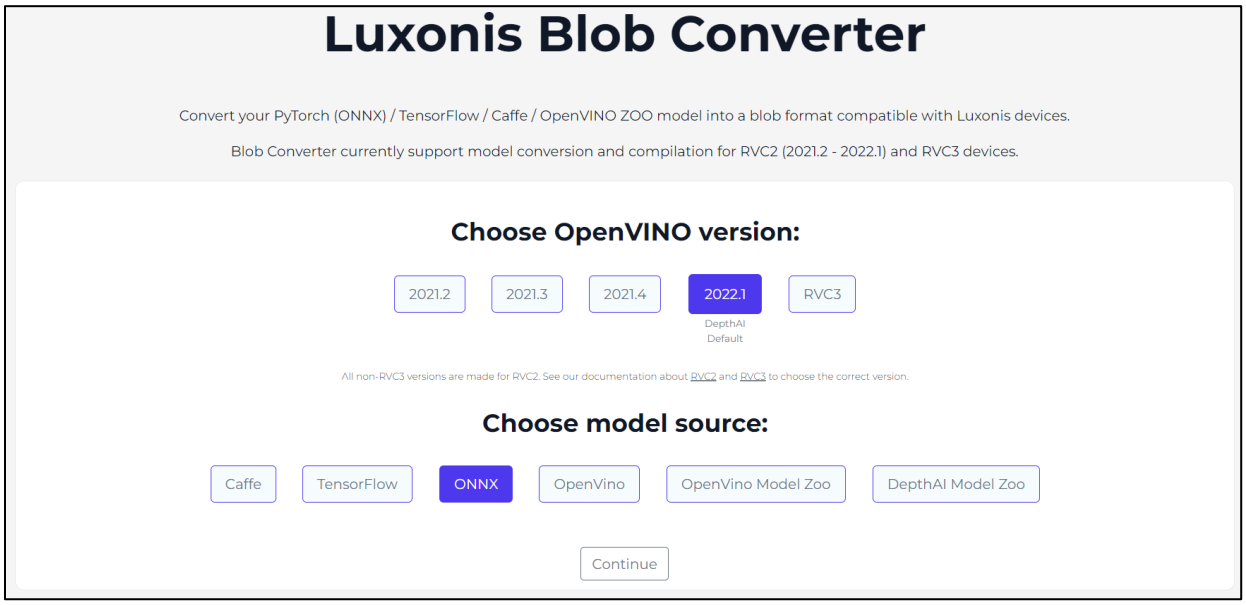


Appendix B 6 Ultralytics Hub - Model export dashboard.

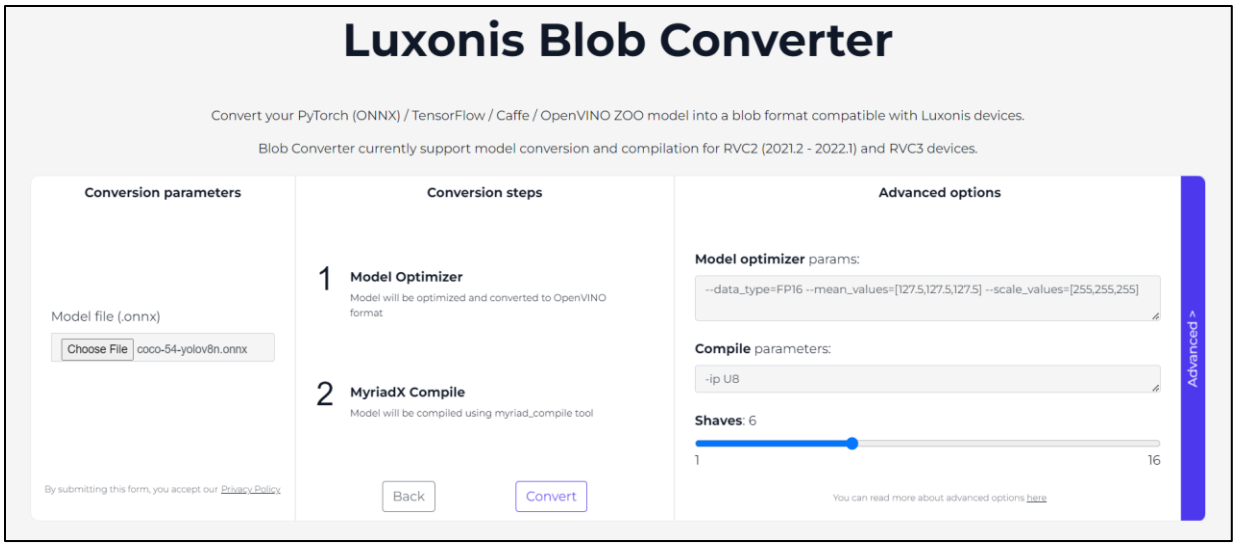


Appendix B 7 Ultralytics Hub - Model export parameters.

8.3. Appendix C: Luxonis Blob Converter Tool Screenshots



Appendix C 1 Luxonis Blob Converter Tool dashboard.



Appendix C 2 Luxonis Blob Converter – model conversion parameters.