FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Advanced 3D Computer Vision Approach to Clinical Motion Quantification for Neurological Diseases

Diogo Peixoto Pereira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: PhD João Paulo Trigueiros da Silva Cunha

July 25, 2020

Advanced 3D Computer Vision Approach to Clinical Motion Quantification for Neurological Diseases

Diogo Peixoto Pereira

Mestrado Integrado em Engenharia Informática e Computação

July 25, 2020

Abstract

The humans motor capabilities can be seriously compromised by neurological diseases like Parkinson and Epilepsy, which in turn worsen their quality of life. Human motion analysis can help to study and diagnose these diseases by providing information related to the gait and other physical motions of the patients. There are already systems being used in the fields of Parkinson and Epilepsy that study and provide very useful information about the neurological impairments of the patients using the Kinect V2 from Microsoft (RGB-D camera). These systems use one application to acquire the information from the camera (Kit - KinecTracker) and two other applications that analyse the data acquired, KiMA (Kinect Motion Analyzer) and KiSA (Kinect Seizure Analyzer). Until now, the cameras used on the KiT application has been the Kinect V2 and previously the Kinect V1.

In this thesis, a new system divided into two applications that use the new Azure Kinect from Microsoft is presented. This system has similar functions to the KiT but his adapted to the new functionalities of the Azure Kinect. The first application enables the acquisition and visualization of the images captured by the Kinect, namely the color, depth and infrared (IR) images. The second application segments the information gathered by the first application. The result is a portable system that can be used within the clinical context to gather the motion information of the patients with neurological diseases.

Keywords: Azure Kinect. RGB-D camera. computer vision. movement-related neurological diseases.

Resumo

As capacidades motoras humanas podem ser seriamente comprometidas por doenças neurológicas como Parkinson e Epilepsia, que por sua vez pioram a qualidade de vida do ser humano. A análise do movimento humano pode ajudar a estudar e diagnosticar essas doenças, fornecendo informações relacionadas à marcha e outros movimentos físicos dos pacientes. Já existem sistemas a serem usados nos campos de Parkinson e Epilepsia que estudam e fornecem informações muito úteis sobre as deficiências neurológicas dos pacientes, utilisando as capacidades do Kinect V2 da Microsoft (câmara RGB-D). Esses sistemas usam uma aplicação para obter as informações da câmera (Kit - KinecTracker) e outras duas aplicações que analisam os dados adquiridos, o KiMA (Kinect Motion Analyzer) e o KiSA (Kinect Seizure Analyzer). Até agora, as câmeras usadas na aplicação KiT foram o Kinect V2 e anteriormente o Kinect V1.

Nesta tese, é apresentado um novo sistema dividido em duas aplicações que usam o novo Azure Kinect da Microsoft. Este sistema tem funções semelhantes ao KiT, mas foi adaptado às novas funcionalidades do Azure Kinect. A primeira aplicação permite a aquisição e visualização das imagens capturadas pelo Kinect, nomeadamente, as imagens de cor, profundidade e infravermelho (IR). A segunda aplicação segmenta as informações adquiridas pela primeira aplicação. O resultado final é um sistema portátil que pode ser usado no contexto clínico para reunir as informações de movimento dos pacientes com doenças neurológicas.

Keywords: Azure Kinect. RGB-D camera. computer vision. movement-related neurological diseases.

Acknowledgements

I would like to express my sincere gratitude to my supervisor PhD João Paulo Cunha for rewarding my good efforts and for being tough on me when I needed to work harder. To the BRAIN members for being so welcoming and being always ready to help. A special thanks to the Machine Learning Researcher Tamas Karacsony for all the technical support during the development of the product.

A special thanks to my family for all the support. To my sister that worked next to me during the quarantine times. And to my parents for being always their, specially in the last couple of weeks before the deadline for this thesis.

I would also like to thank my friends Pedro Silva, António Almeida, Gonçalo Moreno, Ana Eulálio, Ana Gomes, Josefa Fonseca and Roberto Moreira that helped me keep my sanity during these complicated times. And also a special thanks to my friend Catarina Ribeiro for all the talks and laughs that we had.

Diogo Peixoto Pereira

“ *Those who dream by day are cognizant of many things which escape those who dream only by night*”

Edgar Allan Poe

Contents

List of Figures

Symbols and Abbreviations

KiT	KinectTracker
KiMA	Kinect Motion Analyzer
KiSA	Kinect Seizure Analyzer
PD	Parkinson’s disease
RGB	Red Green Blue
RGB-D	Red Green Blue and Depth
IR	Infrared
UPDRS	Unified Parkinson’s Disease Rating Scale
CAMSHIFT	Continuous Adaptive Mean Shift
EEG	Electroencephalogram
ToF	Time of Flight
SL	structured-light
AMCW	Amplitude Modulated Continuous Wav
WFOV	Wide Field of View
NFOV	Narrow Field of View
SDK	Software development Kit
FOI	Field of Interest
FOV	Field of View
UMI	Inertil Measurement Unit
MKV	Matroska Video File
EMU	Epilepsy Monitoring Unit
WPF	Windows Presentation Foundation
IDE	Integrated development environment

Chapter 1

Introduction

The world we live in is facing constant technological advancements in all areas and it's hard to keep up with the pace. The areas of computer sciences and medicine are no different. Combining both has been the main focus of professionals from both areas in order to provide better health care to the public. Using computer vision and artificial intelligence in neurological diseases diagnosis is just another step in this endeavour.

1.1 Context

Neurological diseases have a long history and they were many times misinterpreted and feared due to their symptoms.

Epileptic seizures is one of the world's oldest recognized conditions, with written records dating back to 4000 BC WHO. But the fear and stigma existing around this disease made it complicated for people suffering from it to live healthily and without being ostracized by the public. In many countries, epilepsy can be counted as a valid reason to ask for a divorce and even in the United Kingdom and in Northern Ireland, the laws related to this issue were only emended on 1971. Epilepsy is a chronic noncommunicable disease of the brain that affects around 50 million people worldwide and is characterized by recurrent seizures, which are brief episodes of involuntary movement that may involve a part of the body (partial) or the entire body (generalized) WHO (2006). Since disturbance of the movement of the body of the patient is a major factor on the correct diagnosis of epileptic seizures it is possible to use computer vision methods to acquire human motion information and analyse in order to help analyse and diagnose these types of neurological diseases.

Another disease that affects 4.5 to 19 persons per 100000 population per year is Parkinson's disease (PD). This disease is a chronic progressive neurodegenerative disorder characterized by the presence of predominantly motor symptomatology (bradykinesia, rest tremor, rigidity, and postural disturbances) WHO (2006). Just like epilepsy, PD is a neurological disease commonly associated with movement disorders and is usually diagnosed by observation of the patient.

These are just examples of many neurological diseases which cause neurological impairments and that benefit from motion capture systems.

1.2 Motivation

Studying the movement of the human body is performed in multiple industries and for multiple purposes Vilas-Boas and Cunha (2016). However, within the clinical context, especially related to neurological impairments context, although there have been many studies about this topic, performing movement analysis in complex environments such as hospitals is still very complicated Vilas-Boas and Cunha (2016). So, when analysing patients with neurological diseases, diagnosis is still mainly based on visual observation and subjective evaluations which is basically qualitative analysis.

Performing movement analysis is complex because it usually requires precise calibration of the equipment used details like lighting, noise and occlusion may affect the results making them inaccurate Roetenberg (2006). That's why the complex clinical environment is not the best place to realise these analyses. Another problem comes from the invasive equipment that is required to perform these analyses which many times requires the patient to use certain equipment in their bodies.

To tackle these problems mentioned above RGB-D cameras appeared. These cameras capture RGB images and also depth information. The Kinects from Microsoft are good examples of these types of cameras that are non intrusive to the human body which is more confortable for the patient and do not require complex calibration Vilas-Boas and Cunha (2016). The new Azure Kinect is also an RGB-D camera that will be used to acquire the data needed.

1.3 Objectives

To aid the analysis of neurological impairments this thesis will explore the sensors capabilities of the Azure Kinect from Microsoft with to see if it is possible and efficient to substitute the previous Kinect V2 in the application KiT. The KiT is an innovative computer vision application that acquires the information about human motion characteristics to help the diagnosis and evaluation of neurological diseases namely PD and epileptic seizures Cunha et al. (2016). If the analysis of the Azure Kinect and the comparison with the previous version prove that creating a new version of Kit is a better approach, it will be done. The data that need to be acquired and provided by the KiT are the RGB, the depth (measures the distance between the camera and the objects) and the infrared (IR) images. Besides this, the images information such as size, width, height and data stamps need to also be acquired in order to be processed by the applications KiMA and KiSA.

1.4 Thesis Outline

This thesis started of with an introduction in which the background of the idea was presented and also explaining why this thesis is important and relevant for the present world. Besides the introduction chapter this thesis also have 5 more chapters that will be outlined below.

On Chapter 2 this thesis presents the state of the art in which epileptic seizures and PD information is given. This section also addresses the motion capture algorithms, the systems that use these algorithms and also the new cameras that are crucial for this thesis.

On Chapter 3 a description of the analysis of the Azure Kinect and its possibilities is given. Furthermore, this chapter also approaches the KiT and explores the decision to improve it or do a new one.

Chapter 4 will provide a description of the method involved on the development of the product and what it entailed. This chapter will first introduce the requirements and the mockups defined for the product and then give an explanation of process of development of the product.

The chapter 5 will present the results obtained and discuss the achievements.

Finally chapter 4 will conclude the work by addressing the obstacles encountered during the thesis and the future work to be done.

Chapter 2

State of the art

This chapter is divided into three parts and gives an explanation about the topics needed to better approach the objectives of this thesis. In the first part a brief explanation of the two neurological diseases, in which the KiT e playing a part in acquiring data for analysis, is given. The two diseases are PD and Epileptic Seizures. The second part explains the background of motion capture systems and the systems that are already in use in two hospitals, on targeting patients with PD and the other targeting patients with Epileptic Seizures. The third part of this chapter describes the evolution of the Kinect over the years until the appearance of the Azure Kinect.

2.1 Diagnosis of Epileptic seizures and Parkinson's disease

The process of diagnosis in medicine has improved tremendously throughout the course of human history. The diagnosis of certain neurological diseases is still rather crude since its only based on patient observation and subjective analysis done by the doctor. However, recent developments have been made in order to introduce quantitative analysis into the diagnosis and study of these diseases.

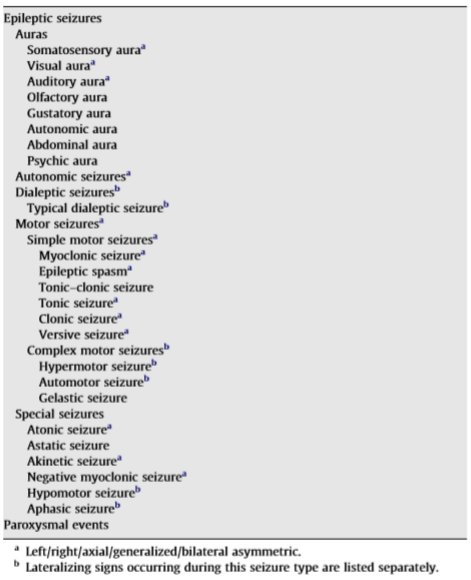
2.1.1 Epileptic seizures

The classification of epileptic seizures is based on clinical observations and detailed reports of seizure semiology (branch of medicine that studies the signs and symptoms of diseases) by the patient or other observers Noachtar and Peters (2009). To control the seizures, and therefore the disease, a clear definition of the seizure type is needed. An important thing to do straight from the beginning is to make sure that we differentiate an epileptic from a non-epileptic seizure. Modern video techniques are already utilised in the field to provide a more clear image and detail on observing patients. However, this still counts as a qualitative analysis since it's up to the subjectivity and observation skills of the doctor to determine whether they are seeing a seizure or something else. Some details may be overlooked which can cause an error on the diagnosis.

Because of this lack of quantitative analysis there are some methods developed to give this kind of information such has video-recordings O'Dwyer et al. (2007); Cunha et al. (2003). This method already helped create objective criteria for the analysis of seizure semiology which culminated in the identification of several seizure types and evolutions Noachtar and Peters (2009).

There are two main systems of seizure classification, one based on semiological criteria and another based on a priori distinction between focal and generalized seizures and semiological aspects in parallel Noachtar and Peters (2009). During an epileptic seizure, some features can be observed that belong to four categories: sensorial sphere, consciousness, motor sphere and autonomic sphere. Usually symptoms corresponding to multiple of the previous categories occur during a seizure but only one is predominant. The sensorial characteristics are difficult to measure and only the patient can describe it. Consciousness characteristics never appear alone and other types of phenomena are needed to classify the seizure. This being said, the most important characteristics for the present dissertation are the motor one's since they can be identified and measured using computer vision equipment.

The motor sphere related characteristics can be divided in two major groups, simple and complex motor seizures. Simple motor seizures are characterized by unnatural and relatively simple movements Noachtar and Peters (2009) and can be further divided in: myoclonic and clonic seizures, tonic seizures, epileptic spasms, versive seizures and tonic-clonic seizures. Complex motor seizures mix movements from different segments of the body that imitate the natural movements Noachtar and Peters (2009). Complex motor seizures can then be divided in hypermotor, automotor and gelastic seizures In fig. 2.1, we can see the semiological seizure classification as described above.



- Epileptic seizures
  - Auras
    - Somatosensory aura[a]
    - Visual aura[a]
    - Auditory aura[a]
    - Olfactory aura
    - Gustatory aura
    - Autonomic aura
    - Abdominal aura
    - Psychic aura
  - Autonomic seizures[a]
  - Dialeptic seizures[b]
    - Typical dialeptic seizure[b]
  - Motor seizures[a]
    - Simple motor seizures[a]
      - Myoclonic seizure[a]
      - Epileptic spasm[a]
      - Tonic–clonic seizure
      - Tonic seizure[a]
      - Clonic seizure[a]
      - Versive seizure[a]
    - Complex motor seizures[b]
      - Hypermotor seizure[b]
      - Automotor seizure[b]
      - Gelastic seizure
  - Special seizures
    - Atonic seizure[a]
    - Astatic seizure
    - Akinetic seizure[a]
    - Negative myoclonic seizure[a]
    - Hypomotor seizure[b]
    - Aphasic seizure[b]
- Paroxysmal events

[a] Left/right/axial/generalized/bilateral asymmetric.
[b] Lateralizing signs occurring during this seizure type are listed separately.

Figure 2.1: Semiological seizure classification has described in Noachtar and Peters (2009)

Joining together all these characteristics that can appear in an epileptic seizure it's easy to notice the complexity of this disease and the difficulty in doing an accurate diagnosis.

2.1.2 Parkinson's disease

Studies from Canada and the United Kingdom show that clinicians diagnose PD incorrectly in near 25% of the patients WHO (2006). This is very problematic since we have seen the large scope of this disease on previous sections. The diagnosis of PD is made exclusively on a clinical basis and is based on unified rating scales that focus on the observation of the patient. Therefore this is a qualitative analysis that relies on the subjectivity of the doctor and his capability to identify the disease.

Over the years there have been many scales for PD diagnosis but there was no universal scale that everyone could use. That's where the Unified Parkinson's Disease Rating Scale (UPDRS) comes into play. The UPDRS was developed with the cooperation of elements from the different existing scales to unify the diagnosis process of PD. This new scale is divided in four parts that were derived from the other scales which are: Part I (Non-Motor Aspects of Experiences of Daily Living (nM-EDL)), Part II (Motor Aspects of Experiences of Daily Living (M-EDL)), Part III (Motor Examination) and Part IV (Motor Complications) UPDRS (2003). As you can see, the motor deficiencies are a major part of the correct diagnosis of PD. This scale is considered as a fast and easy to use tool in PD's diagnosis and it becomes even faster by letting the patient do a self-administration of the first and second part of the UPDRS. This is only viable due to the personal nature of those two parts that clearly rely on the experiences of the patient and this way the neurologists can focus their efforts in the third and fourth part of the scale UPDRS (2003).

By analysing the second and third part of the diagnosis process of PD using UPDRS you notice the importance of detecting the smallest details in the patient motor capabilities and how using video recordings and later analysis using computer vision technologies could help in detecting them. There are already some motion capture systems that provide quantitative analysis to this area. One of those being a system that uses the Kinect V1 to study the gait of the patients. However, it was stated on the paper that explained this system that further tests needed to be made to validate the discoveries Rocha et al. (2014).

2.2 Motion capture technologies

The human motion has been a focus of studies since Aristotle first addressed the issue in is book *The Motum Animalium* Roetenberg (2006). However, motion capture technologies, although present before, were only improved considerably with the development of photography and cinematography Vilas-Boas and Cunha (2016). The introduction of video camera technologies has further improved the way we study human gait and other important human movements. Due to the high applicability of motion capture technologies in several areas of study their evolution has been fast and steady.

2.2.1 Motion capture algorithm

One of the crucial parts of motion capture technologies is the algorithm used that can be generalised and divided in three main components: extraction of the human body structure, tracking the movement of these structural elements and the recognition of movements Vilas-Boas and Cunha (2016); Aggarwal and Cai (1999). The extraction part of the algorithm is what will be the focus of this thesis since the KiT will be the only application worked on.

The extraction part can be further divided into two different approaches, one which uses an a priori model of the human body and another with no pre-determined model.

The first one is based on the segments of the human body and uses stick-figures, cylinder or contours to emulate the body parts. The model based on stick-figures considers that the human body is divided in joints that are grouped in subparts (head, torso, hip, arms and legs) Chen (1992). The example present on fig. 2.2 image A exemplifies this model and has 14 joints and 17 segments. The model in which the body is divided in cylinders is based on the Walker model. This model defines local Cartesian coordinate systems for each major articulation of the human body in order to facilitate the definition of the cylinders Hogg (1983). An example of this model is present on fig. 2.2 image B. The last model creates contours by means of ribbons. These ribbons are commonly used to represent 2D shapes and are abstract regions drawn from the outlined picture. These ribbons are then separated in two sets, one set for the parts of the human body and the other for the background aroud the body parts Leung (1995). On fig. 2.2 image c you can see this model only with the set of ribbons that represent the body parts.

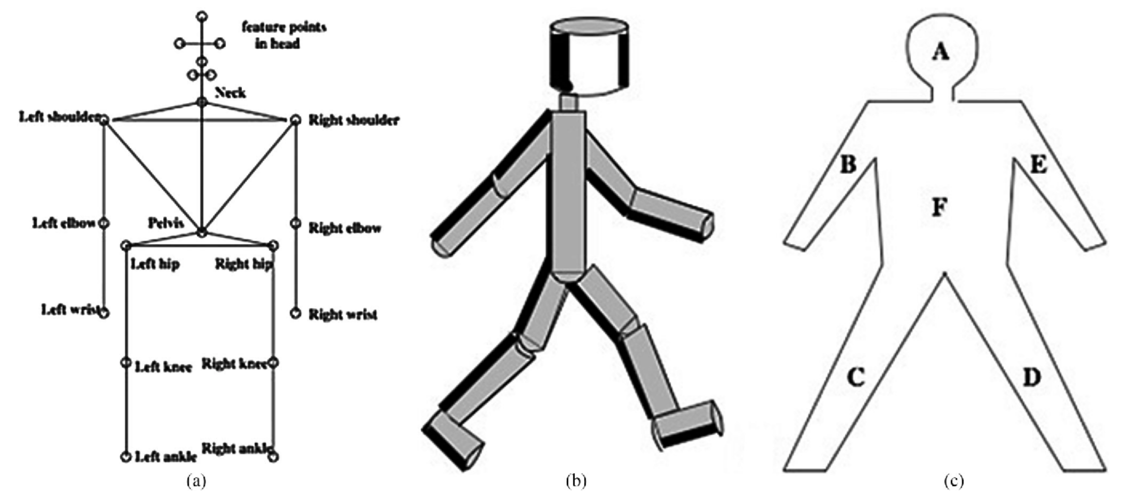


Figure 2.2: A priori models that represent the human body Vilas-Boas and Cunha (2016)

The second is based on the knowledge of how our eyes perceive motion. Studies show that human movement can be addressed by points representations or body contours without even assuming any a priori relative stiffness or joint biomechanics of the points or corresponding borders Johansson (1976). Consequently, some methods based on this were created namely correlation measurements between points on the body, demand for fixed rotation axes of joint points, iterative search of the joints through the movement it is being extracted or based on analysis of the optical flow between successive images Vilas-Boas and Cunha (2016).

The tracking process is used for establishing a correlation between the features extracted from the scene and the different frames of the movement and is based on ballistic trajectories or Kalman filters to predict the tracking object position, or use Mean-Shift inspired implementations such as the popular “Continuous Adaptive Mean Shift” (CAMSHIFT) Vilas-Boas and Cunha (2016). The number of cameras used for tracking plays a major role in having the best results and usually multiple cameras are only necessary when 3D imaging is necessary.

Recognition is usually associated with face recognition but it is also important in the identification of certain processes like frontal or backward movement. The algorithms for recognition that are used to identify these kinds of movements are based on phase space theory, motion dynamics constraints, and pattern recognition techniques Vilas-Boas and Cunha (2016).

2.2.2 Background of motion capture systems

The motion capture systems provide the data for the algorithms previously mentioned and are classified based on the sensors used and on the source of the signal that is received by the systems. Motion capture systems can be divided in *Inside-out* (sensors are placed on the body to detect natural or artificial external source), *Inside-in* (sensors and the sources are placed on the human body), *Outside-in* (external sensors that detect artificial or natural sources attached to the body) and *Outside-out* (computer vision based system). Another parameter that differentiates motion capture systems is the input signal of the sensors used which will determine the nature of the data recorded that can provide, for example, muscle activity information or inter-limb coordination. This new parameter can further classify motion capture systems as inertial, acoustic, mechanical, magnetic, optical and computer-vision based. Each of these systems belong to one of the classes previously mentioned in this section Vilas-Boas and Cunha (2016).

Inertial systems belong to the *Inside-in* class that utilise on-body sensors (gyroscopes, accelerometers and magnetometers) that are used to measure many kinematic components of motion. Gyroscopes measure the angular velocity of the of the body part to which it is attached. There are three types of gyroscopes from which the vibrating mass gyroscope was one of the most popular due to its lower size and weight compared to its peers Grimaldi and Manto (2010). The accelerometers however, uses Newtons second law as basis to measure the acceleration along the sensitive axis of the sensor Grimaldi and Manto (2010). The reliability of the accelerometers made it possible for them to be used within clinical context namely in analysing patients with PD. These systems were very disadvantageous in the past due to the big exoskeleton that was necessary to be used by the user and the difficulty of calibration. Nowadays it has evolved to the point that it can be used outside of a laboratory environment Vilas-Boas and Cunha (2016).

Acoustic systems can be *Outside-in* or *Inside-out* and are tracking systems that use ultrasonic pulses to determine the position of the body. The signal can be easily disturbed so unless there is a clear line of sight, this system should not be used Vilas-Boas and Cunha (2016).

Magnetic systems use magnetic field theory to produce 3D positions of the human body and rotational information. This is an *Outside-in* system that doesn't cause any problem to the human body but can be easily disrupted by any electric or magnetic interferences in the surroundings Vilas-Boas and Cunha (2016).Another problem that comes with these systems is the fact that they are expensive and low accuracy. On the bright side, these systems do not suffer from any occlusion problems because the human body is transparent to magnetic fields Gabriel et al. (1996). On fig. 2.3 you can see an example of an inertial and magnetic human motion tracking sensor.

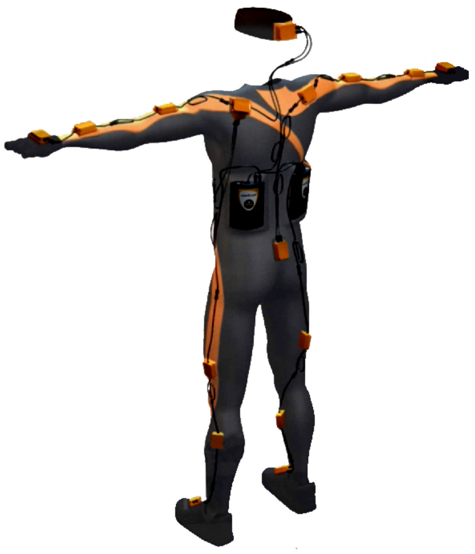


Figure 2.3: Human motion tracking system using inertial and magnetic sensors Roetenberg et al. (2009)

Mechanical systems are *Inside-in* systems and they make use of an exoskeleton to directly track body joint angles using the attached sensors, which provide occlusion free information of body posture Vilas-Boas and Cunha (2016). There are some good examples of these types of sensors namely the plantar pressure assessment that provides information about the foot and ankle functions during gait Orlin and McPoil (2000); Abdul Razak et al. (2012).

Video recordings have been present for a long time in medicine and the diagnosis of many diseases is made by observation of those recording. However, without a quantitative analysis of the data recorded, this method is purely qualitative and dependent on the observation skill of the doctor. So the optical systems were created to improve this technique. These systems are marker-based and are, therefore, part of the *Outside-in* class. The idea is to use either reflective or light-emitting markers that are identified by the cameras and are used to recreate a 2D or 3D image (dependent on the number of cameras) of the movement captured. These systems have some difficulties attached, namely the calibration, post-processing time, non-portability and problems related to occlusion Vilas-Boas and Cunha (2016).

Finally, we have the marker-less *Outside-out* new systems that can reliably measure a wide range of parameters from human motion such as 3D body shape and disambiguate poses. These systems are commonly known as Computer vision based systems. A good example of the utilization of this new technology is the RGB-D cameras that have already been tested on the field and are giving very good results. A study has been made that uses the capabilities of the RGB-D camera to study epileptic seizure of a patient by monitoring his bed 24/7 Cunha et al. (2016). As it was said before, this technology, without the usage of any kind of other sensors, was able to extract body motion information 87.5% faster and the images acquired had an average correlation with 3D motion trajectories of 84,2%. These systems can be divided based on the way they construct the 3D image. We have time of flight (ToF), which measures the depth of a scene by quantifying the changes that an emitted light signal encounters when it bounces back from objects in the scene, and structured-light (SL) systems, which the working principle is the projection, into the 3D scene, of a known light pattern viewed by the camera Vilas-Boas and Cunha (2016). An example of the architecture of these kinds of systems can be scene on fig. 2.4, in which a kinect camera is viewed and the steps of the recognition process is depict.

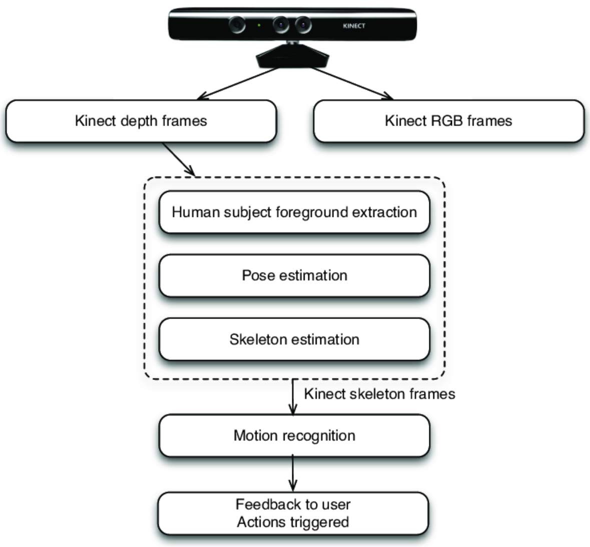


Figure 2.4: Basic kinect rgb-d architecture

2.2.3 Motion capture systems in clinical environments

The motion capture systems have suffered tremendous improvement through the years especially when related to the medical field. At the moment there are many systems used to study and diagnose epileptic seizures and the PD that use the capabilities of RGB-D cameras and the Kinect V1 and V2 to better capture the images that would then be analysed with the help of other medical means of examination Cunha et al. (2016).

The first system worth mentioning and analysed is the NeuroKinect employed on the detection of epileptic seizures that was developed and used in the University of Munich Cunha et al. (2016), INESC TEC and the Institute of Electronics and Informatics Engineering of Aveiro (IEETA). This system operates with an RGB-D camera (Microsoft Kinect), that sends the information to a computer running a KiT (software developed using the Kinect Software Development Kit v1.5) and is synchronized with an electroencephalogram (EEG) video system to acquire the data of the patient Cunha et al. (2016). The editing, storage and analyse of the data are done using the KiMA software (the architecture of the system can be seen on fig. 2). This system can provide sets of motions of interest which allow the discrimination between seizures arising from temporal and extra-temporal brain areas Cunha et al. (2016) which is crucial to the differentiation between seizures and therefore contributing for a better diagnose of this disease. Another important detail is that this system provides 3D imaging analyses which is a step forward compared to the original 2D imaging which is usually used in the clinical context.

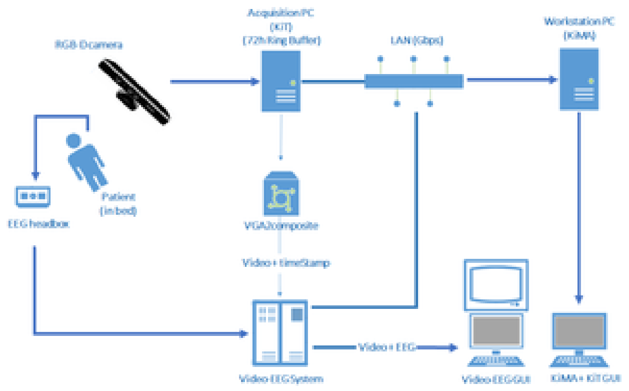


Figure 2.5: NeuroKinect architecture

There are also motion capture systems used to study PD and two of them are systems based on a Kinect V1 RGB-D camera and the kinect V2. The process to acquire data is very similar to the one used in epileptic seizures, in which the KiT is used to acquire the skeleton data of the patient. The data acquired need to be manually selected since the camera as a particular range in which it works better and then the gait cycles are identified. This is done based on the depth information that is received at the same time as the skeleton is acquired (Schema on fig. 3). After this process many parameters are computed (velocity, accelerations, distances ...) for each gait cycle and they are then used to distinguish between non-PD patients and PD patients Rocha et al. (2014). The second generation (Kinect V2) however, due to the new capabilities of the camera, such as, higher depth fidelity, infrared capabilities and the possibility to track more joints, made it more appropriate for the assessment of PD. One of the reasons is that it permits the usage of more gait parameters important for the diagnosis of PD. Finally it is important to say that this new system as an advantage over the old kinect V1 because it is more adaptable to new and different scenarios Rocha et al. (2015).

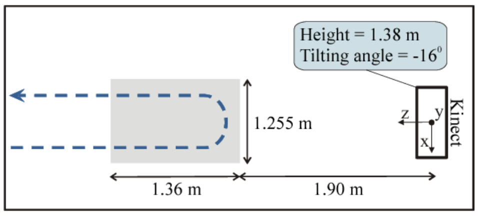


Figure 2.6: Setup used for data acquisition using Kinect V1

2.3 Kinect evolution

For the scope of this dissertation, we will focus on the RGB-D cameras that were developed by Microsoft. The reason why this is so is because this thesis is focused on the works done in the fields of PD and Epileptic seizures in the hospitals of Porto and Munich that previously used the Kinect V1 and after that the Kinect V2. Therefore this section will focus on the evolution of this cameras until the Kinect V2 camera.

2.3.1 Kinect from XBOX 360

The first Kinect was made by Prime Sense in collaboration with Microsoft and came with the gaming platform XBOX 360, own by Microsoft, as a peripheral device. This Kinect was equipped with with a RGB camera and an IR emitter and camera Cruz (2012). An image of the Kinect can be viewed on fig. 2.7. These two cameras made it possible to acquire complementary color images and depth information per pixel of the image, which brought many advantages in the fields of computer graphics, image processing, computer vision and so on. An example of the usage of this camera in computer vision and image processing is using it to identify peoples by their faces named Kinect Identity. This technology was developed for the XBOX 360 and with time it learns the faces of the players so that it can identify who is the player 1 and 2 during games by their faces Leyvand (2011). Within medical context there are also studies conducted using this camera that proved to be very successful such as a respiratory monitoring system using the Kinect (Xia and

Siochi (2012)) and also a sleep monitoring system that uses the depth information provided by the Kinect to detect sleep eventsYang (2014).

The great advancement of this new camera was the technology behind the depth sensor (IR emitter and camera), which was developed by group of researchers from Prime Sense (Zalevsky et al. (2013)). This sensor used the structured light method to measure the depth Cruz (2012), which was already explained in the section 2.2.2. Due to the increase in popularity of this Kinect for the XBOX 360, a version for Windows was later released for developers and researchers to use Cruz (2012). The resolution and other characteristics of this Kinect can be viewed in fig. 2.9 in comparison to its successor the Kinect V2.

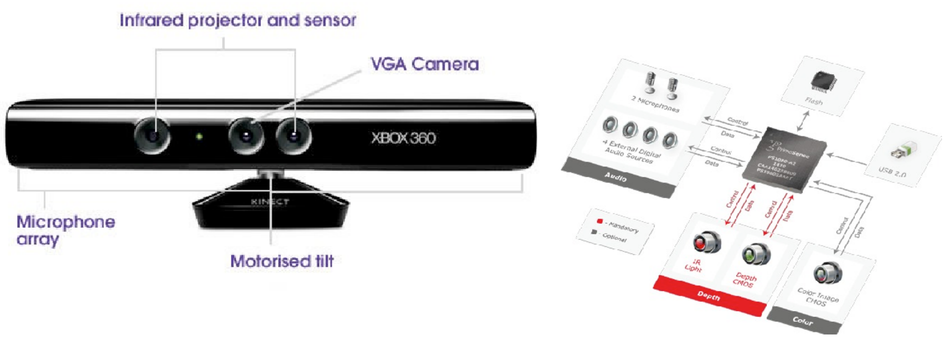
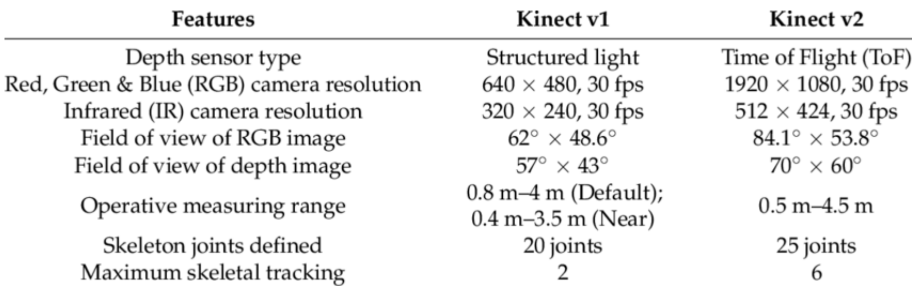


Figure 2.7: Kinect device and chip Cruz (2012)

2.3.2 Kinect V2

The second generation of the Kinect was the Kinect V2 which had some improvements in comparison to the previous version and also a major difference in the way the depth is measured. These advantages come in the form of better performance and accuracy, as well as a wider field of view due to this new way to measure depth Al-Naji et al. (2017). In Kinect V2 the depth is calculated using the ToF tecnhology that was described on the section 2.2.2 of this chapter. A study has been made that compares the two technologies (ToF and SL) using the Kinect V2 and the Kinect and explains in which environments each technology works better Sarbolandi (2015). In fig. 2.8 you can see the major differences between the two Kinects. As you can see, Kinect V2 has a wider field of view and better resolution for both the RGB camera and IR camera without lowering the frames per second (fps). Another thing worth notice is the increase in the number of joints that can be defined and the maximum number of skeletal tracking in the Kinect V2..

This Kinect comes equipped with an RGB camera, an IR sensor and projector that provide three types of frames, color, ir and depth. Besides these frames the Kinect V2 comes also equipped with a body tracking software that can reconstruct a 3D body and provide skeletal tracking, joint



Features	Kinect v1	Kinect v2
Depth sensor type	Structured light	Time of Flight (ToF)
Red, Green & Blue (RGB) camera resolution	640×480, 30 fps	1920×1080, 30 fps
Infrared (IR) camera resolution	320×240, 30 fps	512×424, 30 fps
Field of view of RGB image	$62^\circ \times 48.6^\circ$	$84.1^\circ \times 53.8^\circ$
Field of view of depth image	$57^\circ \times 43^\circ$	$70^\circ \times 60^\circ$
Operative measuring range	0.8 m–4 m (Default); 0.4 m–3.5 m (Near)	0.5 m–4.5 m
Skeleton joints defined	20 joints	25 joints
Maximum skeletal tracking	2	6

Figure 2.8: Specifications comparison between Microsoft Kinect V1 and Kinect V2 Al-Naji et al. (2017)

tracking and human recognition using the depth and color frames as basis Al-Naji et al. (2017). Overall the Kinect V2 is a better, more robust, version of the Kinect from Microsoft. Kinect V2 can be seen on fig. 2.9.

Figure 2.9: Kinect V2 sensor Al-Naji et al. (2017)

As mentioned in section 2.2.2 the application KiT was firstly programmed for the first version of the Kinect but later adapted to be used with the Kinect V2. This application reads the information from the Kinect and separates each of the frames (color, depth and IR) in different files to be further processed by the other software (KiMA and KiSA). The KiT also has a mode for gait analysis which retrieves the joints and skeletal tracking of the person being analysed.

2.3.3 Azure Kinect

The last instalment of the Kinect series is the Azure Kinect. And as you can see in fig. 2.10 the new Azure Kinect was downsized and it is much more practical. The features of the camera are also present on the image.

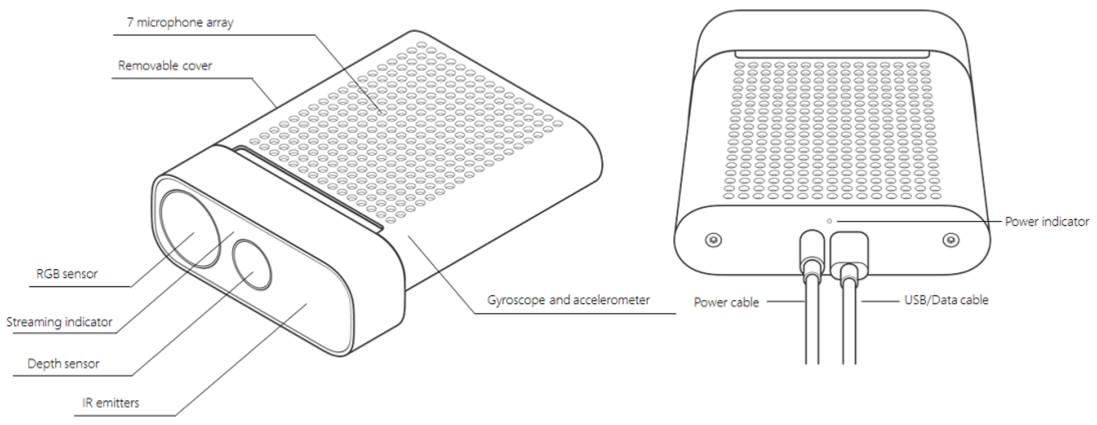
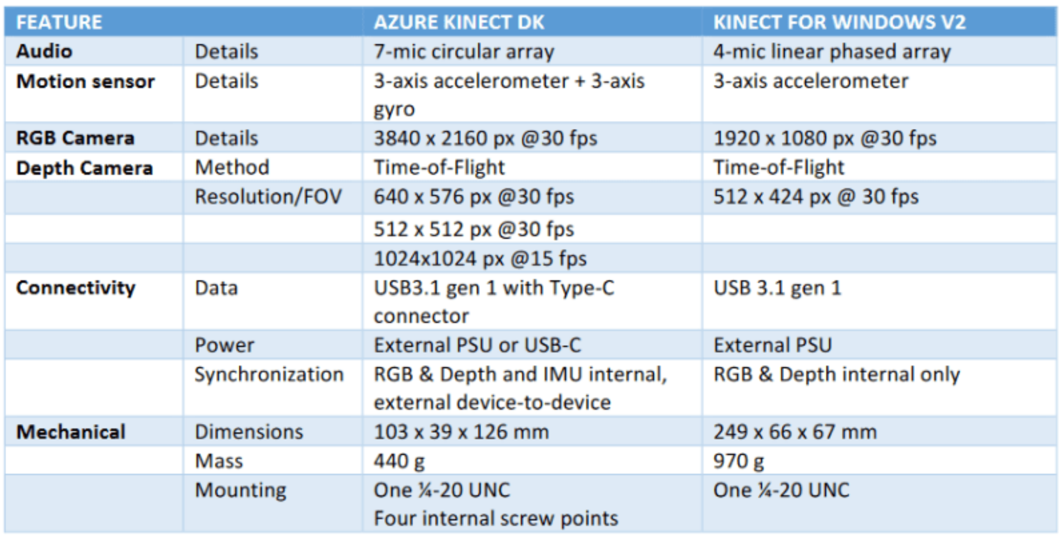


Figure 2.10: Azure Kinect features Tesych

The Azure Kinect made serious improvements in the depth camera by implementing the Amplitude Modulated Continuous Wave (AMCW) ToF principle (Tesych) that is based on the paper published in 2018 that cover the continuous wave ToF principle. Cameras using this principle emit light with an amplitude modulated light source that is deflected when it encounters an object. The delay between the the emission of the light and the reflection is used to calculate distance between the camera and the object making it possible to create the depth image Bamji (2018). This camera gives two types of images, a depth-map and a clean IR reading (which the result depends on the amount of light on the scene). The capabilities of the camera are quite good with 1-Megapixel ToF imaging chip with advanced pixel technology which enables higher modulation frequencies and depth precision, automatic per pixel gain selection enabling large dynamic range allowing near and far objects to be captured cleanly and low systematic and random errors. It's important to notice that all modes can be run at up to 30 fps with exception of the Wide Field of View (WFOV) unbinned mode that runs at a maximum frame rate of 15 fps Tesych.

The differences between the Kinect V2 and the Azure Kinect are quite significant, as you can see in fig. 2.11. The Audio and Motion sensors have been improved and a gyroscope was added. However for the scope of this thesis only the RGB and Depth camera are the one's that need to be looked at. Focusing on the RGB camera it can be seen the the max resolution was improved and the fps are still 30. It's important to note that this is not the only resolution available, as it will be shown on chapter 3. The same goes for the depth camera in which it can be notice that 1024*1024 resolution only with 15 or less fps. This mode corresponds to the WFOV unbinned mode that was mentioned above. These are the main differences and what will be the focus of this thesis.



FEATURE		AZURE KINECT DK	KINECT FOR WINDOWS V2
Audio	Details	7-mic circular array	4-mic linear phased array
Motion sensor	Details	3-axis accelerometer + 3-axis gyro	3-axis accelerometer
RGB Camera	Details	3840 x 2160 px @30 fps	1920 x 1080 px @30 fps
Depth Camera	Method	Time-of-Flight	Time-of-Flight
	Resolution/FOV	640 x 576 px @30 fps	512 x 424 px @ 30 fps
		512 x 512 px @30 fps	
		1024x1024 px @15 fps	
Connectivity	Data	USB3.1 gen 1 with Type-C connector	USB 3.1 gen 1
	Power	External PSU or USB-C	External PSU
	Synchronization	RGB & Depth and IMU internal, external device-to-device	RGB & Depth internal only
Mechanical	Dimensions	103 x 39 x 126 mm	249 x 66 x 67 mm
	Mass	440 g	970 g
	Mounting	One ¼-20 UNC Four internal screw points	One ¼-20 UNC

Figure 2.11: Specifications comparison between Kinect V2 and Azure Kinect Skarredghost (2020)

Chapter 3

Azure Kinect and KiT

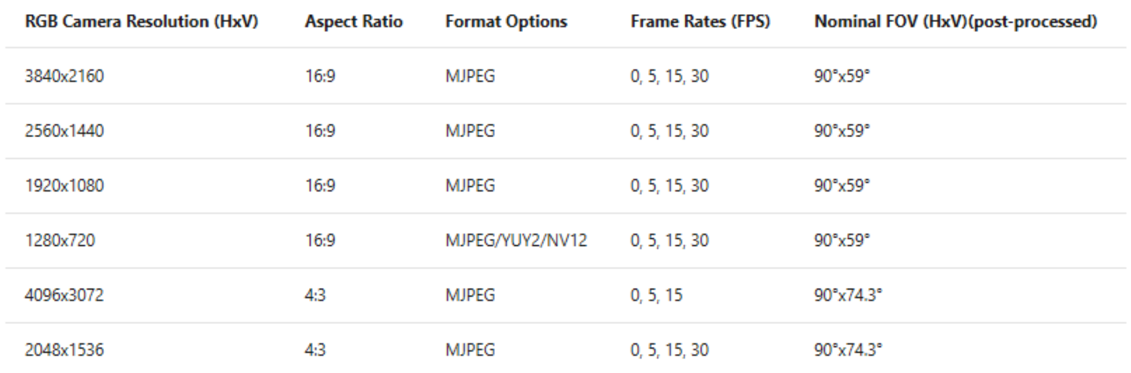
This chapter will be divided into two parts. The first part will have an in-depth analysis of the colour and depth camera to understand the conditions in which they work and what modes work better with each other. The second part will focus on the problems of the previous KiT as well as the compatibility with the Azure Kinect. Then the best modes of operation will be chosen in accordance to the specifications given by the Hospitals in which the KiT is currently functioning so that the best performance can be achieved. This study will result in deciding whether to update the old KiT so that it can work with the Azure Kinect, or make a new KiT specifically for the new camera and for the objective of this thesis.

3.1 Azure Kinect

This section will present an overview of the hardware capabilities of the Azure Kinect as well as the functionalities available within the software development kit (SDK) that comes with it.

3.1.1 Overview

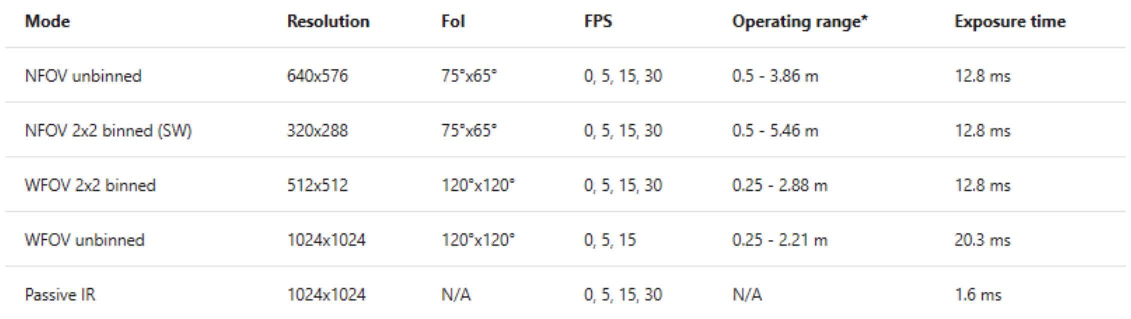
As described in chapter 2 the Azure Kinect comes equipped with a color camera and a depth camera. The color camera has six native operating modes in which the images are produced directly by the camera. These operating modes produce color images in three formats: MJPEG, YUY2 and NV12. However, due to the fact that the YUY2 and NV12 formats only work for the lowest resolution, they won't be addressed in this thesis. Besides these three formats, the Azure Kinect can also produce images in BGRA format but it needs the host CPU to convert the original image in MJPEG to this format Tesych (2019). So it is not a native mode. There is another problem that this format has that will be addressed in this chapter that will dismiss this format from the scope of this thesis. Another thing to take into account is that the mode with resolution 4096x3072 only achieves a frame rate of 15 which is not optimal to study neurological diseases in which the smallest detail can't be overlooked. And if 30 fps is not achieved, some information may be lost due to loss of frames. The characteristics of the color camera can be observed in fig. 3.1.



RGB Camera Resolution (HxV)	Aspect Ratio	Format Options	Frame Rates (FPS)	Nominal FOV (HxV)(post-processed)
3840x2160	16:9	MJPEG	0, 5, 15, 30	90°x59°
2560x1440	16:9	MJPEG	0, 5, 15, 30	90°x59°
1920x1080	16:9	MJPEG	0, 5, 15, 30	90°x59°
1280x720	16:9	MJPEG/YUY2/NV12	0, 5, 15, 30	90°x59°
4096x3072	4:3	MJPEG	0, 5, 15	90°x74.3°
2048x1536	4:3	MJPEG	0, 5, 15, 30	90°x74.3°

Figure 3.1: Native color operating modes of Azure Kinect Tesych (2019)

The depth camera also has different modes of operation. They can be divided in three groups: narrow field of view (NFOV), wide field of view (WFOV) and Passive IR. For this project, Passive IR will be dismissed since this mode turns of the iluminators of the camera and depends only on the ambient illumination of the space in which the camera is instaled. Another distinction between the modes is that both the NFOV and WFOV modes can be binned or unbinned. The binned modes are less precise then the other modes because the reduce the resolution of the images by combining two adjacent pixels into one bin. Like the color camera high resolution mode, the depth camera also has a mode that can only achieve 15 fps (WFOV unbinned) so it will be discarded for this project due to the same reason mention on the previous paragraph. The last thing to notice is the field of interest (FOI) that is way bigger in the WFOV mode. These modes and their characteristics can be seen on fig. 3.2.



Mode	Resolution	FoI	FPS	Operating range*	Exposure time
NFOV unbinned	640x576	75°x65°	0, 5, 15, 30	0.5 - 3.86 m	12.8 ms
NFOV 2x2 binned (SW)	320x288	75°x65°	0, 5, 15, 30	0.5 - 5.46 m	12.8 ms
WFOV 2x2 binned	512x512	120°x120°	0, 5, 15, 30	0.25 - 2.88 m	12.8 ms
WFOV unbinned	1024x1024	120°x120°	0, 5, 15	0.25 - 2.21 m	20.3 ms
Passive IR	1024x1024	N/A	0, 5, 15, 30	N/A	1.6 ms

Figure 3.2: Depth operating modes of Azure Kinect Tesych (2019)

The only thing that is missing is understanding how the fields of view (FOV) of both cameras are positioned depending on the modes selected. In fig 3.3 a representation of the FOV of the cameras can be observed. From the observation it can be seen that the WFOV mode is not that compatible with this project since there is a need for a good overlap of pixels between the color and depth images. From this point of view the best option would be a NFOV unbinned mode with a color camera mode with an aspect ratio of 4:3 for better overlap between images, better resolution and precision. However, a color camera mode with aspect ratio of 16:9 can't be overlooked since the overlap is only bad on the corners of the images. And those corners are usually discarded on the processing part done by the KiMA and KiSA applications.

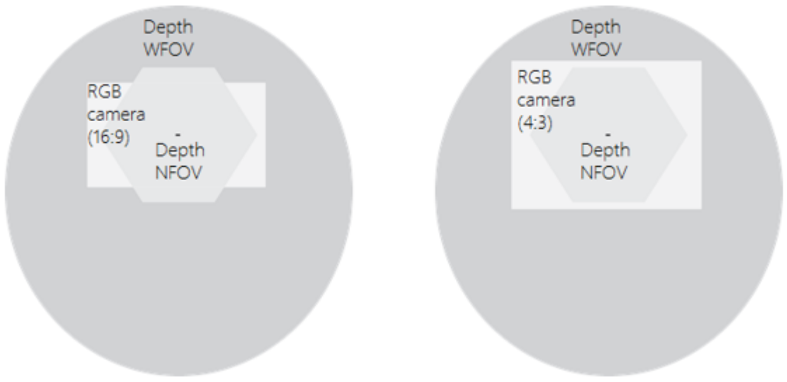


Figure 3.3: FOV of color and depth cameras Tesych (2019)

3.1.2 Azure Kinect SDK

The Azure Kinect comes equipped with a couple of tools and a SDK. The tools encompass a viewer, that lets you play a previously recorder file and can also show the images captured by the camera in real time, a recorder, that lets you record the type of images that you want with the format you choose, and a firmware tool to update your system.

The Azure Kinect SDK is the software provided by Microsoft that gives you access to the device configurations and the the hardware of the sensors so that you can develop your own code for the Azure Kinect. This version has some major differences compared to the previous version. In the Kinect V2 SDK you had access to the color and depth frames, as well as the body tracking frames in the same function. Basically you had one function two access both parts and it would provide synchronised frames. Now, the Azure Kinect SDK, separated these in two SDKs, the Sensor SDK and the Body Tracking SDK. Therefore, obtaining synchronised frames of both parts is more complicated. However, the color, depth and IR frames can still be obtained in a synchronised way. For the scope of this thesis the focus will be on the Sensor SDK.

The Sensor SDK has many core functionalities that are related with the hardware that composes the Azure Kinect. The cameras provide the images that can then be accessed by the SDK and can also be either transformed or saved. The Inertil Measurement Unit (UMI) can be accessed to get the samples related to the motion of the device provided by the accelerometer and the gyroscope. And the microphones can also be accessed. But what really matters in this project is the images acquired by the cameras. With the Sensor SDK we can change the color format and resolution, change the depth mode, select the frame rate, retrieve only the synchronised frames and even manage many Azure Kinects at the same time. This last function won't be addressed in this project. One interesting feature of this SDK is that it lets you transform the depth image into the color image and vice versa. This way you can have, for example, two images with the same dimensions in which one is a simple color image and the other is the transformation of the depth image into the color image. The last feature, and one that will also be used on this project, is the possibility to record the three images in a matroska file video (mkv) file. This file will contain instances of each type of image, with additional information beyond the image. It will save also the width, height, size, data stamp and other useful information. The file can be opened with a normal video player and you will see only the color video but if you use the viewer tool you can see the three frames in video. However, there is one color format that is not supported by this recorder function, the BGRA format. The thing that makes this function powerful is the fact the you also have access to a playback function that loads the mkv files and gives you access to the instances of the images and all the information within. This will prove crucial for this project.

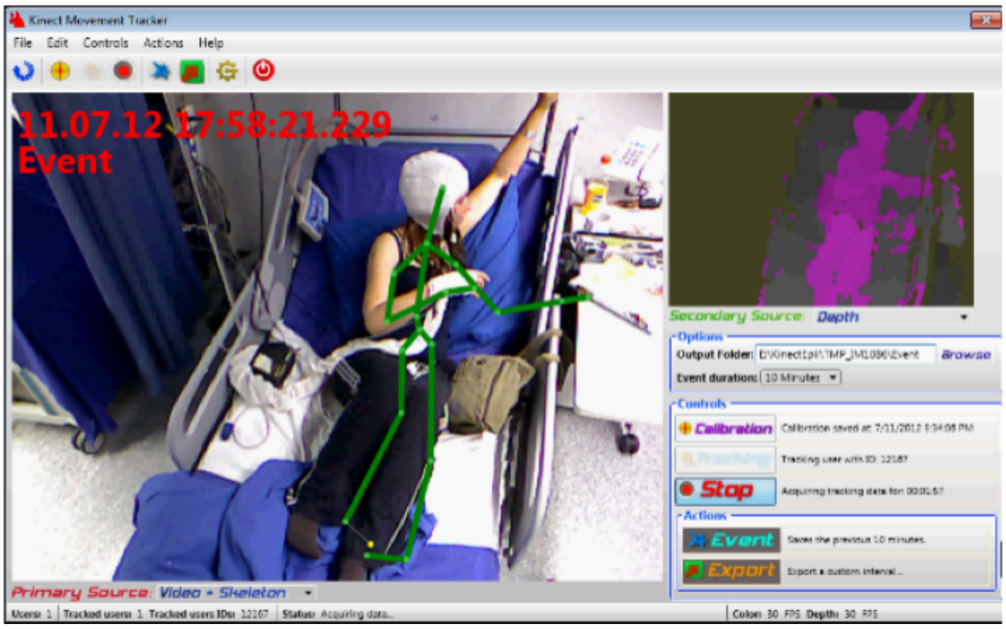
3.2 The KiT decision

In this section a simple review of the key concepts behind the KiT will be presented in order to better understand how the Azure Kinect can be integrated (or not) into the code and also the circumstances of the Hospitals in which the KiT is integrated. After that simple preliminary test was made to see how the camera worked using different combinations of the color camera and depth camera modes to see which one should be used in the environment that the KiT is placed. In the end it will be decided whether to continue using the original KiT or make a new one.

3.2.1 The KiT and the hospital environment

The KiT is an application that connects to the Kinect hardware, reads the information provided be it and then export it to specific files that can be later processed by other software applications like KiMA and KiSA, so that it can help analyse and diagnose neurological diseases. If we use the terms presented in the chapter 2, the KiT belongs to the extraction part of the motion capture algorithm. However this application also has some tracking process since it accesses the body tracking module of the Kinect V2 to extract the body skeleton and joint for further processing. For this project the focus will be on the extraction of the images and their exportation to files.

The flow of the application is pretty easy to understand. In order to start acquiring the frames, the application should receive some input from the user. First comes the information related to where to save the frames and the name of the session. Next you should select in the preferences the frames that you would like to save. It can be any combination between color, IR, depth, body and/or body index frames. Finally you have the option to select which frames you want to see while acquiring. This way you can keep track of what is being acquired at the same time that it is done. After receiving all this input the application can start acquiring the data. As mentioned in the previous section, the Kinect V2 provides all the frames in a multi frame reader so the KiT just saves each frame in a queue of their own to be dealt in a different thread each. Each frame that is acquired can be shown in the KiT interface and saved in the correspondent file that will be exported for processing in other applications. The input that the user made at the beginning is responsible for selecting the frames saved and shown on the interface. The frames shown can be selected while the camera is acquiring as shown on fig. 3.4. This figure is an image of the KiT working in the University of Munich Epilepsy Monitoring Unit (EMU) and is actually showing the skeletal body image acquired from the Kinect V2. As you can see by the image, the body tracking software of the Kinect V2 is not very precise because the woman has her left arm up, but the left arm of the skeleton is not. These are the main concepts behind the KiT software.



Primary Source: Video + Skeleton

Secondary Source: Depth

Output Folder: E:\KinectEpil\TMP_JM1080\Event

Event duration: 10 Minutes

Calibration saved at: 7/11/2012 5:34:08 PM

Tracking user with ID: 12167

Acquiring tracking data for: 00:01:57

Saves the previous 10 minutes.

Export a custom interval...

Users: 1 | Tracked users: 1 Tracked users IDs: 12167 | Status: Acquiring data... | Color: 30 FPS Depth: 30 FPS

Figure 3.4: KiT operating inside the University of Munich EMU Cunha et al. (2016)

The KiT application is not only being used in the epilepsy unit functioning but also in a PD unit in the São João University Hospital in Porto (Portugal) that also implements this application. In both cases the KiT was introduced to help the analysis of the diseases. The previous examinations that were done by the doctors continue to exist and the KiT just provides new information to the system. The idea that is used is that when a seizure is identified by the exams, the doctor can use the information captured by the KiT in that moment and compare it with their own exams. This information that was captured by the KiT was processed by the KiMA or KiSA software in order to be used. In fig. 3.5 we can se the EMU system functioning in which the EEG that detects seizures is compared to the images acquired by the KiT after being synchronised. The most important factor for the KiT to function correctly in these environments is to know the space available for saving all this information acquired. The KiT has access to a buffer of 10 TB of data for saving purposes. This is important for the tests conducted using the Azure Kinect in the next section, because it will help to decide the best approach for the KiT application.

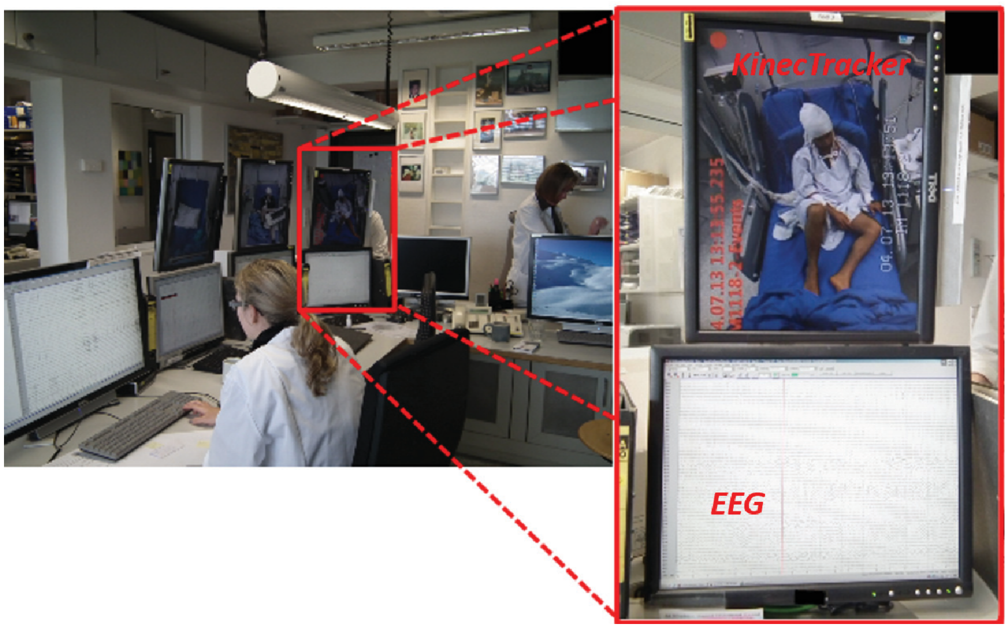


Figure 3.5: Aspects of the 3Dvideo-EEG system console installed in the routine of University of Munich EMU Cunha et al. (2016)

3.2.2 Azure Kinect Mode Preliminary Tests

The test done to the Azure Kinect is supposed to test the real frame rate acquired by the device in the different modes, as well as see the space occupied in disk by the images provided by the device. After getting the images a calculation using the 10 TB of memory available in the hospitals to get the time that the Kit can be functioning without running out of space. This preliminary test is a simple program in C that has the following flow:

- Start the Azure Kinect Device
- Configure the device to the mode being tested
- Start the cameras
- Get the images

- Save the images in files

This program was made using the C language since the source code of the Azure Kinect SDK is also written in C. This program is very standard and uses just the functionalities of the SDK for starting the devices and cameras with the correct configurations. As an example, if you want to configure the device to work at 30 fps, use the color format MJPEG with a resolution of 1280*720 and the depth mode NFOV Unbinned, you should do the following:

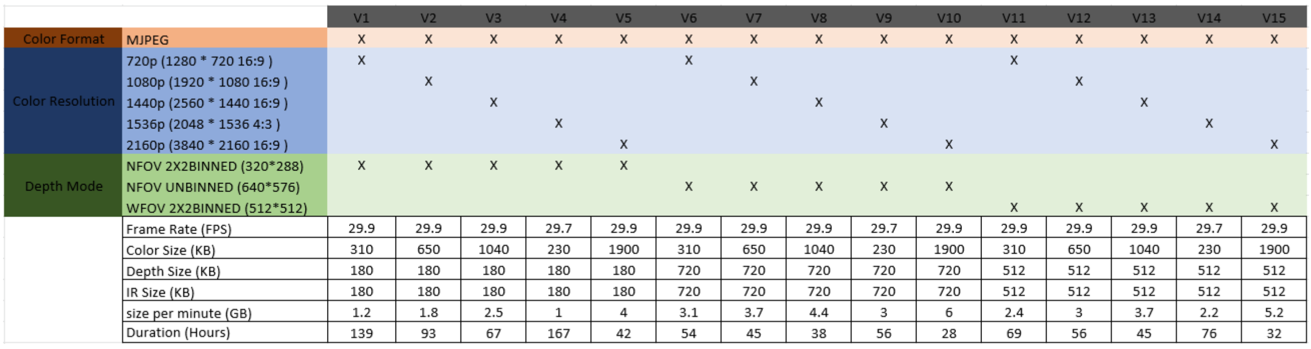
```c
k4a_device_configuration_t config = K4A_DEVICE_CONFIG_INIT_DISABLE_ALL;
    config.camera_fps = K4A_FRAMES_PER_SECOND_30;
    config.color_format = K4A_IMAGE_FORMAT_COLOR_MJPG;
    config.color_resolution = K4A_COLOR_RESOLUTION_720P;
    config.depth_mode = K4A_DEPTH_MODE_NFOV_UNBINNED;
    config.synchronized_images_only = true;
```

Listing 3.1: Azure Kinect device configuration in C

The last configuration makes it only possible to receive synchronised images from the device. This being said, there were some configurations that were left out due to certain reasons. For the color format, only the MJPEG and BGRA formats were tested since the other two (YUY2 and NV12) only worked in the lowest resolution. The color resolution 4096x3072 was also left out of the test because it only worked at a maximum of 15 fps. The depth mode WFOV Unbinned was also not tested because of the same fps issue. The test was divided in two parts, one centered around the color format MJPEG and the other on the BGRA format, because these formats were the cause of the major differences in the frame rate of the device.

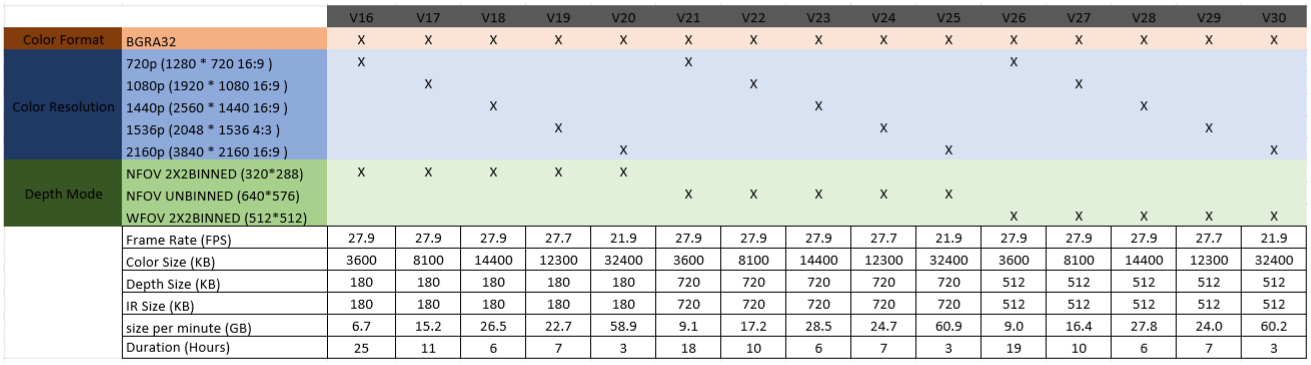
In the fig. 3.6 you can see the test results for the color format MJPEG. Looking into the results it can be seen that in this format the fps is very close to the 30 that was expected and only when the device was configured to the resolution that used the aspect ratio 4:3 would the value be slightly lower. The test also showed what was the configuration that had the best duration and the one with the worst. These durations were obtained by calculating the amount of images that could be saved on the entire 10TB disk and based on the frames per second obtained by each configuration. The best duration was achieved with the 720p color resolution and the NFOV 2x2 Binned depth mode. The worst duration was the 2160p resolution and the NFOV Unbinned depth mode.

The BGRA color format test results can be observed in the fig. 3.7. The results of this format were a very different from the previous. The fps in this mode was lower then 30, even if not by a great margin, and the resolution with aspect ratio of 4:3 was closer to the 15 fps mark than the 30. This result may be caused by the fact that BGRA images are converted from the MJPEG native format provided by the device. This means that before the images are acquired, the host CPU processes the original images in order to provide the BGRA images, which causes the fps to be lower. Another problem with this format is the size of the files. As it can be seen in the image, the size of the color images is so big that even in the lowest resolution, the KiT can only work for a day without running out of space.



		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
Color Format	MJPEG	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Color Resolution	720p (1280 * 720 16:9)	X					X					X				
	1080p (1920 * 1080 16:9)		X					X					X			
	1440p (2560 * 1440 16:9)			X					X					X		
	1536p (2048 * 1536 4:3)				X					X					X	
	2160p (3840 * 2160 16:9)					X					X					X
Depth Mode	NFOV 2X2BINNED (320*288)	X	X	X	X	X										
	NFOV UNBINNED (640*576)						X	X	X	X	X					
	WFOV 2X2BINNED (512*512)											X	X	X	X	X
	Frame Rate (FPS)	29.9	29.9	29.9	29.7	29.9	29.9	29.9	29.9	29.7	29.9	29.9	29.9	29.9	29.7	29.9
	Color Size (KB)	310	650	1040	230	1900	310	650	1040	230	1900	310	650	1040	230	1900
	Depth Size (KB)	180	180	180	180	180	720	720	720	720	720	512	512	512	512	512
	IR Size (KB)	180	180	180	180	180	720	720	720	720	720	512	512	512	512	512
	size per minute (GB)	1.2	1.8	2.5	1	4	3.1	3.7	4.4	3	6	2.4	3	3.7	2.2	5.2
	Duration (Hours)	139	93	67	167	42	54	45	38	56	28	69	56	45	76	32

Figure 3.6: Depth operating modes of Azure Kinect Tesych (2019)



		V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	V29	V30
Color Format	BGRA32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Color Resolution	720p (1280 * 720 16:9)	X					X					X				
	1080p (1920 * 1080 16:9)		X					X					X			
	1440p (2560 * 1440 16:9)			X					X					X		
	1536p (2048 * 1536 4:3)				X					X					X	
	2160p (3840 * 2160 16:9)					X					X					X
Depth Mode	NFOV 2X2BINNED (320*288)	X	X	X	X	X										
	NFOV UNBINNED (640*576)						X	X	X	X	X					
	WFOV 2X2BINNED (512*512)											X	X	X	X	X
	Frame Rate (FPS)	27.9	27.9	27.9	27.7	21.9	27.9	27.9	27.9	27.7	21.9	27.9	27.9	27.9	27.7	21.9
	Color Size (KB)	3600	8100	14400	12300	32400	3600	8100	14400	12300	32400	3600	8100	14400	12300	32400
	Depth Size (KB)	180	180	180	180	180	720	720	720	720	720	512	512	512	512	512
	IR Size (KB)	180	180	180	180	180	720	720	720	720	720	512	512	512	512	512
	size per minute (GB)	6.7	15.2	26.5	22.7	58.9	9.1	17.2	28.5	24.7	60.9	9.0	16.4	27.8	24.0	60.2
	Duration (Hours)	25	11	6	7	3	18	10	6	7	3	19	10	6	7	3

Figure 3.7: Depth operating modes of Azure Kinect Tesych (2019)

3.2.3 The KiT problems and the decision

There is one last thing to consider before making a decision regarding the KiT and the Azure Kinect, the current problems that the KiT has, which are:

- The frame rate of the application drops from time to time which causes loss of important information
- The files in which the frames are saved (binary files) have some issues that cause the information to be sometimes rewritten and as a result, loss of information
- The format in which the information is save is very complicated, which causes difficulties in reading the data in the KiMA and KiSA software.

These problems were identified through the years of utilization of the application by the supervisor of this thesis, and by a Machine Learning Researcher at INESCTEC. In view of all the information gathered in the previous sections and chapters, and with the information regarding the problems of the KiT decisions have been made.

The first decision made was to create a new KiT for acquiring the color, depth and IR frames. This decision was made due to two key factors. The loss of valuable data due to frame rate drops and and the binary files and also due to the new feature of the Azure Kinect SDK that lets you record the captures directly from the camera without frame drops. This last reason was also the one that molded the new KiT. Basically, to correct the frame drops, it was decided that the new KiT will be divided in two parts, or better said, two distinct applications. One part will just produce the recording video and show it in real time to the user, so that the program won't lose processing time with saving each image into files and all the additional information that caused the frame drops. The second part will open the video file (mkv file) using the Azure Kinect SDK and will seperate each type of frames (color, depth and IR) into separate files and create an additional file with the complementary information regarding each frame.

The Second decision was related to the main configuration that would be used for the device. Looking into the color format, the MJPEG format was chosen for three reasons. The size of the files were more appropriate since more then one day would be preferable. The frame rate is much more stable than the BGRA format. But the most important reason is that the BGRA format is not supported by the recording function of the SDK because it is not a native mode of the Azure Kinect. The depth mode selected was the NFOV Unbinned manly because the other two modes were binned and lost half of the information as mentioned in the beginning of this chapter. Therefore the depth and IR images acquired have better resolution which is great for the processing done by KiMA and KiSA. The last configuration chosen was the 720p color resolution so that the KiT could function nonstop for more than two days without running out of space. This configuration corresponds to the test V6 in the fig. 3.6.

These were the decisions made with the information gathered on the last chapters. In the next chapter the development methods applied in the making of the Azure KiT will be discussed.

Chapter 4

Azure KiT development

This chapter will be divided in three parts that will explain the process of development of the Azure KiT and also the programming language and tools used in the development. The first section will describe the tools and the other two sections will explain the development of each part of the KiT. The last section will present the system architecture of the application developed.

4.1 Development tools

The programming language chosen for this project was C$\sharp$, which is a multi-paradigm programming language, using the WPF (Windows Presentation Foundation) framework. This combination was chosen for two reasons. The previous KiT was already made using this technology and it provided a good graphic interface for the user. And also because a library in C$\sharp$ for the SDK was already made and the functionalities needed for the project were available.

To use C$\sharp$ and the WPF framework, the IDE (integrated development environment) Visual Studio was chosen, because all three technologies were developed by Microsoft making it easier to use and get support when needed.

In addition to these technologies three nuget packages belonging to the NuGet Gallery were used. The NuGet Gallery is a package manager for .Net and it gives access to many libraries with useful functions NuGetGallery (2020). The first nuget package that was used was the K4AdotNet package which is a library that provides the functions to work with the Azure Kinect devices using C$\sharp$ bibigone and baSSiLL (2020). Another package used was the Extended.Wpf.Toolkit that provides many WPF controls that are not present on the original framework Xceed (2020). The last package used was the Accord.Video.FFMPEG.x64 which contains a bundle of classes and methods to handle video files using FFMPEG Accord.NET (2017). This last package was specifically for 64 bit applications because the Azure Kinect SDK doesn't work with 32 bit applications.

4.2 Azure Kit part 1

In this section, the process of development of the first part of the Azure KiT will be presented, starting with the definition of the requirements for the application followed by the mockups of the user interface and the development process of the program.

4.2.1 Requirements and Mockups

The definition of the requirements for this part of the KiT was agreed upon by the candidate and the supervisor. The main focus of this part of the KiT was to provide a clear interface to the users and to correctly produce the mkv file with the color, depth and IR frames. The requirements were divided in three groups one for each simple window interface that the user had access. The windows are: initial window for the definition of the session, main window for the recording and a configuration window to let the user change the device configurations if needed. The requirements are as follows:

1. Initial Window
   - Get the folder in which the user wants to save the mkv file
   - Get the name of the session in order to separate from other sections that may have already been recorded in the same folder or even to save on the same session if the user wants it so.
   - Start the session and move to the main window
2. Main Window
   - The main window should have a screen to show the color image at the same time that it is being recorded
   - While being recorded the user should know that the images are getting acquired
   - The video recorded should be split in 2 minutes videos to facilitate the access to the videos in the times of the day needed and to not loose all the content if the Kit stops working.
   - A button to start the recording
   - A button to stop the recording
   - A button to get access to the configurations window
3. Configurations Window
   - Let the user select the color format
   - Let the user select the color resolution

- Let the user select the depth mode
- Let the user select the frame rate
- Let the user select the duration of the videos recorded
- Let the user confirm the selected options

Mockups were made for each of these windows. In fig. 4.1 the mockup for the initial window is presented. As it can be seen the user can select the folder and the name of the session before starting it.

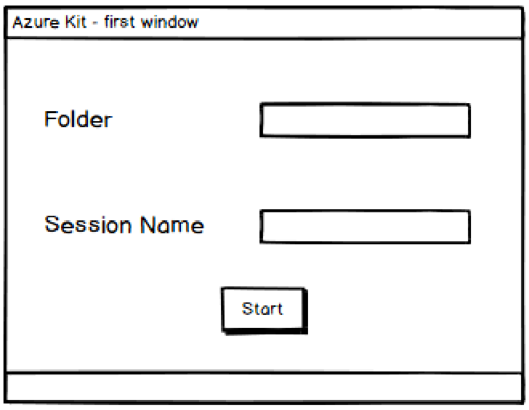


Figure 4.1: Mockup of the initial window of the Azure KiT

The main window can be observed in fig. 4.2. In the mockup you can see the screen that will show the real time color image and the buttons for the functions required in the requirements.

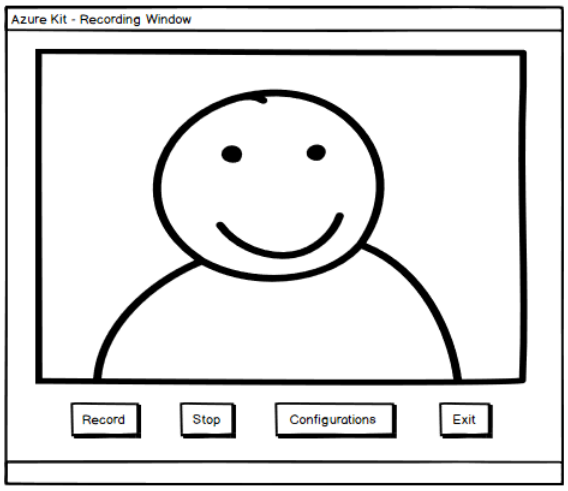


Figure 4.2: Mockup of the main window of the Azure KiT

The configurations window can be observed in fig. 4.3. The configuration window gives the user the option to change the configurations as demanded by the requirements as well as a button to confirm the new configurations.

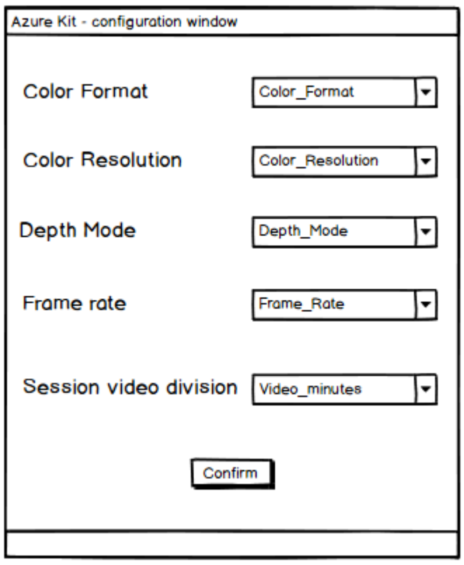


Figure 4.3: Mockup of the configurations window of the Azure KiT

4.2.2 Development Process

Having defined the requirements and mockups the development process of the program can be started. The first thing to look at when programming with the WPF framework is that you have windows that communicate between themselves and that contain elements that can have events associated. So the first thing developed was the layout of the three windows with the elements that were defined in the previous section. After this, the events for each element were created and the link between the windows was established.

In this part of the Azure Kit the main window was the core window, so it was the one that called the other windows. When the program starts the main window calls the initial window with the dispatcher and is hidden until the initial window ends the work it is supposed to do.

The work that the initial window needs to do is pretty simple, it just needs to select or create a new folder (if it doesn't exist yet) for the images to be saved in. The code that is shown on the listing 4.1 is what is used in the initial window.

```
// Create a Folder Browser
String folderName = null;
WinForms.FolderBrowserDialog dlg = null;
dlg = new WinForms.FolderBrowserDialog();
```

```
dlg.SelectedPath = System.AppDomain.CurrentDomain.BaseDirectory;

// Show open file dialog box
WinForms.DialogResult showResult = dlg.ShowDialog();

// Check if the directory exists
!Directory.Exists(folderName)

// Creates a new directory if it doesn't
Directory.CreateDirectory(folderName);

// Saves the name of the directory if it exist and can be opened
Directory.GetAccessControl(folderName);
```

Listing 4.1: Select the folder where the recording files are saved

After the initial window was completed the main window became the focus of the development. And this is where the K4AdotNet package. As mentioned before, this package provides the same functions that the Azure Kinect SDK provides but in C♯ language. When the main window is loading, the Kinect device is opened and configured with the modes chosen on the chapter 3. Next the recording event was made. This event would be activated when the user clicked on the record button. This event can be broken down in three main activities. The first activity was creating the recording file using the SDK functions, to where the captures obtained by the Kinect will be recorded. The second part was creating an event that would trigger when the 2 minutes (unless another video duration was selected) have passed in order to create a new recording file. The video division purpose is to make sure that the data being acquired is saved and not lost. If the data was being saved in only one file and the application stopped working due two external reasons or lack of this space, all the information might get lost. This way if it happens only the file being used at the time is lost. A sample of this second activity can be seen on the listing 4.2.

```
// Create a timer for 2 minutes
Timer saveTimer = new Timer(1000 * 2 * 60);

// Generate the event
saveTimer.Elapsed += saveTimer_Elapsed;
saveTimer.Start();

// Event function
private void saveTimer_Elapsed(object sender, ElapsedEventArgs e)
{
    // File name has the day an time
    currentDate = DateTime.Now;
    this.file_name = this.folderName + "/" + currentDate.ToString("
        yyyyMMdd_HHmmss") + ".mkv";
    // Create the new file with the SDK functions
    this.recorder[this.recorder_counter + 1] = new Recorder(this.file_name
       , this.device, this.config);
    this.recorder[this.recorder_counter + 1].WriteHeader();
    this.recorder_counter++;
    this.recorder[this.recorder_counter - 1].Dispose();
}
```

Listing 4.2: Timer and event responsible for the creation of the recording files

The last activity is receiving each capture, show the color frame on the screen and save the captures to the file. receiving the capture can be done using the nuget package that it is being used. To show the color image the buffer of the color image (that contains the raw image data) needs to be transformed into a bitmap source to be then shown on the screen. In listing 4.3 is the sample of the code responsible for showing the image on the screen. Note that this is not the only way to show an image from the image buffer. After showing the color image the full capture can be saved and disposed to give space to the next capture.

```
// Verify if the color image was obtained
if (capture.ColorImage != null)
{
    // Create an intermediate byte array buffer to save the image
    var color = capture.ColorImage;
    innerBuffer1 = new byte[color.SizeBytes];
    Marshal.Copy(color.Buffer, innerBuffer1, 0, innerBuffer1.Length);

    // Create and show the bitmap source
    using (var stream = new MemoryStream(innerBuffer1))
    {
        colorImageKinect.Source = BitmapFrame.Create(stream,
            BitmapCreateOptions.None,BitmapCacheOption.OnLoad);
    }
}
```

Listing 4.3: Show the color image on the screen

Only the configuration is left and this one has a very simple job. It only receives the input of the user related to the configurations of the device and changes it if the user confirms it. With this the Azure KiT part one is completed and the next part can be addressed.

4.3 Azure Kit part 2

This section follows the same pattern as the previous one. First the requirements and mockups are presented to give an overview of the Azure KiT part 2 and then a section dedicated to the development process of this program is given.

4.3.1 Requirements and Mockups

For this part, besides the candidate and the supervisor, another person helped to define the requirements. This person is a Machine Learning Researcher at INESCTEC and is working with the KiMA software so he was the on that made the decisions related to the type of files created and exported in this part so that the process work of the KiMA software could be facilitated. The main function of this part of the KiT is to receive the video files recorded during the KiT part one functioning and separate the video in three videos, one for each type of frame (color, depth and IR). Another important point is that the user will select the start and end time of the seizure in order for the program to only separate the files that belong in that time period. This part only has one window so the requirements are the following:

- Get the session where the recorded videos were saved
- Select the new folder in which to save the files
- Select the start time of the seizure
- Select the end time of the seizure
- Select the type of frames to be saved
- Start the processing of the data

For these requirements the mockup made is in fig. 4.4.

4.3.2 Development Process

The development process of this part of the Azure KiT was similar to the previous part except for the fact that it only has one window. This time, for the elements present in the layout of the window, one of them came from the nuget package Extended.Wpf.Toolkit. The element in question was the start and end date. These dates needed to be composed of a day and time of the day, and there is no element like this in the original WPF framework. This package provided a date time picker that did this job easily. During the development the was an element that was

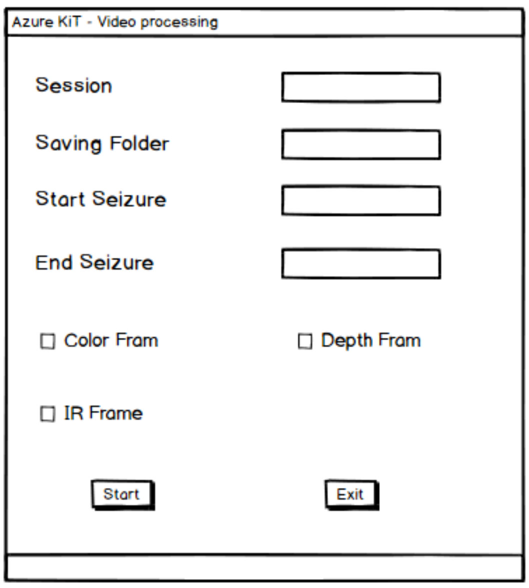


Figure 4.4: Mockup of the Azure KiT part two

added due to necessity that was the session name for the saving folder since there can be multiple different seizures in the same session. And it is important to separate the information. After the layout of the main window was made, the next step was to process the information given has an input. When the user clicks on the start processing button an event should start to do this process. This event started by selecting the video files that needed to be processed based on the start and end date. However, the folder contains many videos, so it is necessary to find the one's that are the target of the processing. To do this first all the names of the video files were copied to a string array and then the program should find the files that contained the start and end date within their videos. because the day and time were part of the name of the files it was very easy to do, has it can be seen in the code sampled present in the listing 4.4.

```
//Copy the file names to the string
string[] videoFiles = Directory.GetFiles(this.videosFolder, "*.mkv").
    Select(System.IO.Path.GetFileName).ToArray();

//Some useful counters
int firstFilePosition = 0;
int lastFilePosition = 0;
int filePosition = firstFilePosition

if (this.startDate.CompareTo(this.endDate) < 0)
{
    for (int i = 0; i < videoFiles.Length; i++)
    {
        //clean the names of the files
        videoFiles[i] = videoFiles[i].Substring(0, videoFiles[i].Length -
            4);
    }
    for (int i = 0; i < (videoFiles.Length - 1); i++)
    {
        if (!this.startFound)
        {
            if (this.startDate.CompareTo(videoFiles[i + 1]) < 0)
            {
                if (this.startDate.CompareTo(videoFiles[i]) >= 0)
                {
                    this.startFile = videoFiles[i];
                    this.startFound = true;
                    firstFilePosition = i;
                }
                else
                {
                    return;
                }
            }
        }
        if (this.startFound)
        {
            // similar to the code of the start file
        }
    }
}
```

Listing 4.4: find the start and end file

After this is done the last nuget package mentioned at the beginning of this chapter will be needed. The Accord.Video.FFMPEG.x64 package was used to save the frames into a video file format. This format would prevent the data from being overwritten has it happened with the previous version of the KiT. This package enabled the creation of video files with a stream of bitmaps with a predetermined format and size (width and height). After creating one videos file for each format, the mkv files recorded by the Azure KiT one that were selected before this process should be opened, one at a time in chronological order. To open these files and access the captures containing the three types of frames, the playback function of the SDK (using the K4AdotNet package) was used. Next comes the process of saving each frame. The color frame had to be converted into a bitmap in order to be saved into the new video file, like the sample on the listing 4.5.

```
// Receiving the color frame
K4AdotNet.Sensor.Image color = this.currentCapture.ColorImage;
```

```
// Only enters if the save color option was selected on the interface
if (colorframesEnabled)
{
    //Create an intermediate Byte array buffer
    innerBuffer = new byte[color.SizeBytes];
    Marshal.Copy(color.Buffer, innerBuffer, 0, innerBuffer.Length);
    Bitmap bmp;

    //Transform the buffer into a bitmap
    using (var stream = new MemoryStream(innerBuffer))
    {
        BitmapEncoder enc = new BmpBitmapEncoder();
        enc.Frames.Add(BitmapFrame.Create(stream, BitmapCreateOptions.None
            , BitmapCacheOption.OnLoad));
        using (MemoryStream outStream = new MemoryStream())
        {
            enc.Save(outStream);
            bmp = new System.Drawing.Bitmap(outStream);
        }
    }

    //To open the video file the width and height of the frames is
        required
    // so the first frame needs to be identified
    if (firstColorFrame)
    {
        streamColor.Open(this.colorFileName, color.WidthPixels, color.
            HeightPixels, 30, VideoCodec.Default);
        firstColorFrame = false;
    }

    //Write frame to the stream
    streamColor.WriteVideoFrame(bmp);
    bmp.Dispose();
}
```

Listing 4.5: Saving the color frames

The depth and IR images were a little bit different because of the way the raw data is saved. In these types of images, each pixel is a 2 Byte little endian unsigned depth data. For the depth image, each pixel corresponds to the distance (in millimeters) from the camera to the object and for the IR image, each pixel corresponds to the brightness level. The problem here is that the K4AdotNet package only has two types of conversion of the buffer. It can either convert to a byte array or a short array. So, in order not to lose any information and convert the buffer to a ushort array the code present in the listing 4.6 was done.

```
//create the buffers
byte[] depthIntermediateBuffer = new byte[depth.WidthPixels * depth.
    HeightPixels * 2];
ushort[] grayScaleDepthData = new ushort[depth.WidthPixels * depth.
    HeightPixels];
depth.CopyTo(depthIntermediateBuffer);
int counterShort = 0;

//Convert each two bytes of the buffer into a ushort value for the array
for (int index = 0; index < depthIntermediateBuffer.Length - 1; index =
    index + 2)
{
    grayScaleDepthData[counterShort] = BitConverter.ToUInt16(
        depthIntermediateBuffer, index);
    counterShort++;
}
```

Listing 4.6: Saving the color frames

This was the example for the depth image but the same is applied for the IR image. The rest of the code is similar to the color frame. The only difference is the that in order to not lose any information the bitmap created had to be in the format Grayscale 16 (Images composed with 16 bits per pixel). The last part of this application was to save the information of each frame (width, height, size and timestamp) into a text file.

After this being done, the development of the Azure KiT was completed.

4.4 System Architecture

The architecture of the system is very simple as it can be seen in fig.4.5. This system is divided in two applications, which were explained previously in this chapter. The first application, azure KiT 1, receives the stream of frames captured by the Azure Kinect device and also the information of the user has an input in order to produce the mkv multiframe video files that contain the color, depth and ir frames exactly how they are captured by the device. The video file is then passed to the azure KiT application two for further processing. In this part the videos recorded are chosen by the user (not every file might be needed) and then they are separated in videos containing only one type of frames. The videos are: color video file, depth video file and ir video file. Along with these files a text file is produced that contains the information related to the frames recorded, like the size and timestamps.

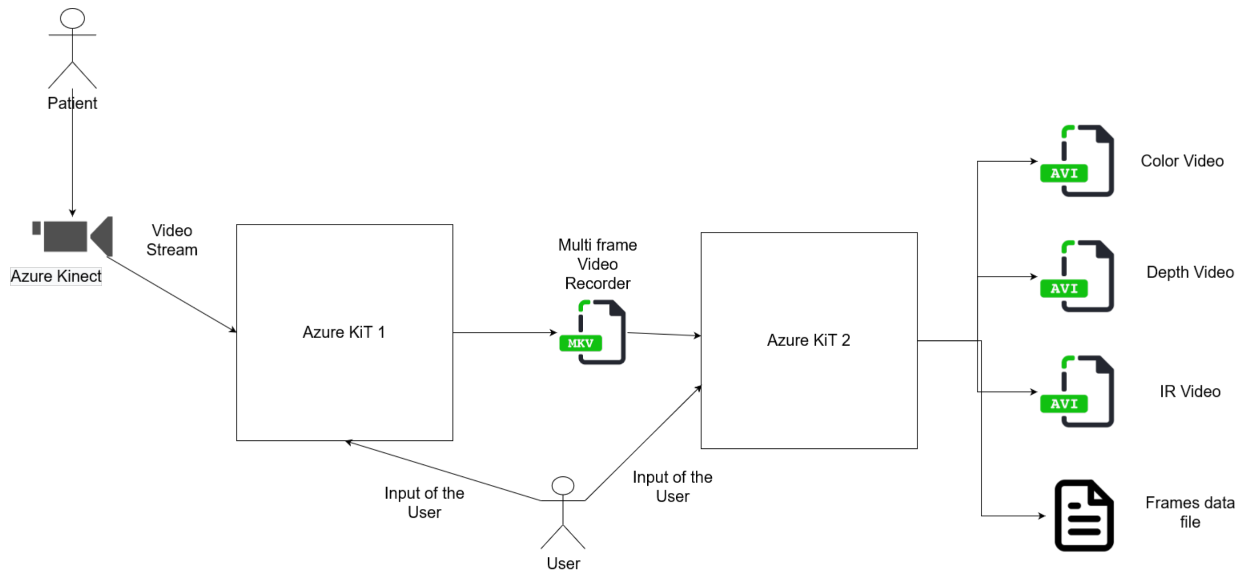


Figure 4.5: Architecture of the azure KiT application

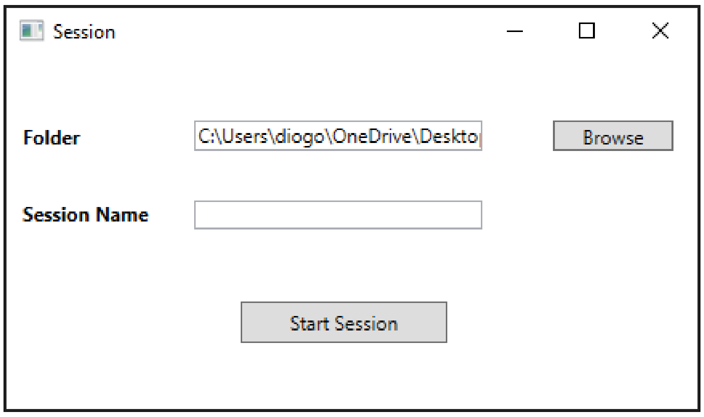
Chapter 5

Results and Discussion

In this chapter the results of the development of the Azure KiT will be presented. Images of the interface and of the results of the both programs will also be presented. This chapter is divided in three sections, one for each part of the Azure KiT and another for the discussion of the results.

5.1 Azure KiT part one results

The Azure KiT part one development resulted in a WPF application with three windows. Each window is very similar to the mockups made in the beginning of the development phase. In fig. 5.1 the initial window can be observed. As it was defined in the requirements, this window receives the target folder and name of the session before starting it.



Folder: C:\Users\diogo\OneDrive\Deskto
Session Name:

Figure 5.1: Azure KiT 1 initial window

In fig. 5.2 The main window of the Azure KiT one is presented showing the application running and acquiring the images. While the images are being acquired, a text saying the data is being acquired appears so that the user knows what is happening. the configurations button in the main window will open the configurations window (fig. 5.3) and is only enable when the Azure KiT is not recording.

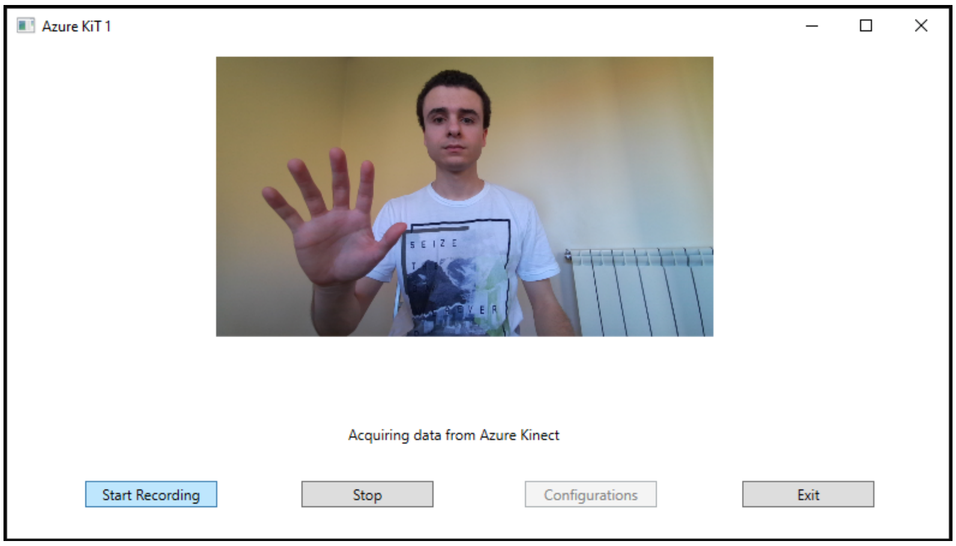
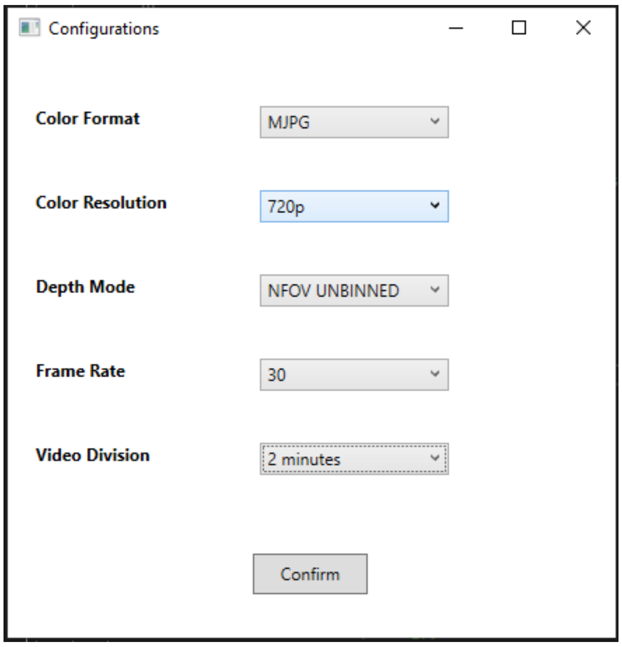


Figure 5.2: Azure KiT 1 main window



Color Format MJPG

Color Resolution 720p

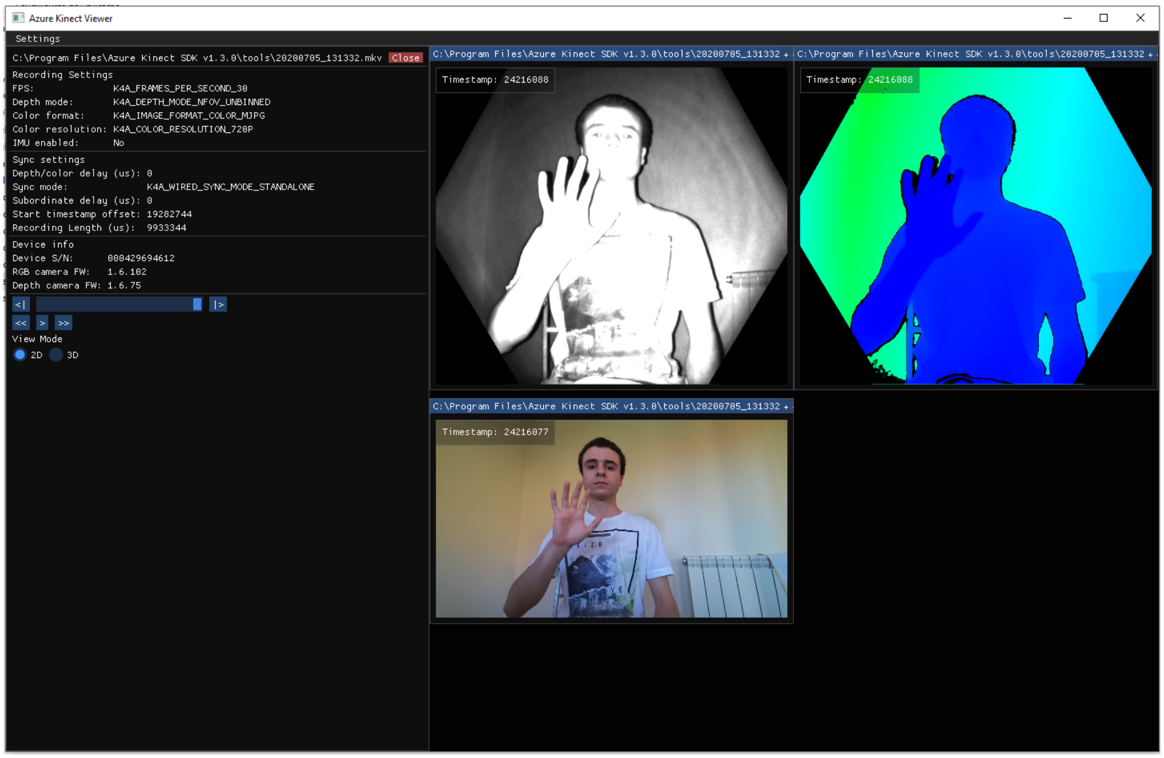
Depth Mode NFOV UNBINNED

Frame Rate 30

Video Division 2 minutes

Figure 5.3: Azure KiT 1 configurations window

After the Azure KiT application is terminated, the videos recorded during the process can be accessed in the folder selected by the user on the initial window. As it was explained on the chapter 3 the videos can be opened with a normal video player. However, only the color images will be played. To see the video with the three types of images the Azure Kinect has a viewer tool, that is installed with the SDK, that can show these recordings. The viewer tool showing the videos recorded can be seen on the fig. 5.4.



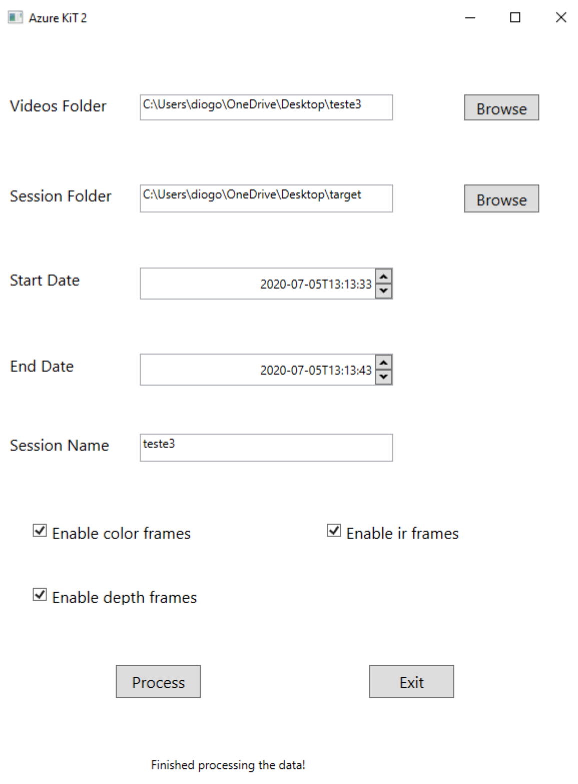
```
C:\Program Files\Azure Kinect SDK v1.3.0\tools\20200705_131332.mkv
Recording Settings
FPS:               K4A_FRAMES_PER_SECOND_30
Depth mode:        K4A_DEPTH_MODE_NFOV_UNBINNED
Color format:      K4A_IMAGE_FORMAT_COLOR_MJPG
Color resolution:  K4A_COLOR_RESOLUTION_720P
IMU enabled:       No
Sync settings
Depth/color delay (us): 0
Sync mode:              K4A_WIRED_SYNC_MODE_STANDALONE
Subordinate delay (us): 0
Start timestamp offset: 19282744
Recording Length (us):  9933344
Device info
Device S/N:      000429694612
RGB camera FW:   1.6.102
Depth camera FW: 1.6.75
```

Figure 5.4: The mkv file recorder by the Azure Kit viewed in the viewer tool provided by the Azure Kinect

These are the results of the Azure KiT part one application. Next comes the processing of this videos, which is done by the azure KiT part two.

5.2 Azure KiT part two results

The Azure KiT part two as an easier interface compared two the part one. This part is responsible for receiving the folder with the videos recorded in the Azure Kit part one and separate the frames into three video files, one for each type. However not all the videos are necessary. Only the videos that correspond to the time periods in which the patient being monitored by the has seizures matter. So the user must input the start and end date of the seizure. The last input needed from the user is the frames that they want to acquire from the recorded videos. Sometimes you just want to get the IR frames and so the application doesn't need to lose time doing unnecessary processing. In fig. 5.5 the interface of the Azure Kit part two is shown.


Azure KiT 2

Videos Folder C:\Users\diogo\OneDrive\Desktop\teste3 Browse

Session Folder C:\Users\diogo\OneDrive\Desktop\target Browse

Start Date 2020-07-05T13:13:33

End Date 2020-07-05T13:13:43

Session Name teste3

[✓] Enable color frames [✓] Enable ir frames

[✓] Enable depth frames

Process Exit

Finished processing the data!

Figure 5.5: Azure Kit 2 interface

After the application ends the process, the video files are recorded using the frames retrieved from the original video files and can be opened using a normal video player. The color video can be observed on the fig. 5.6. One thing to notice is that although the IR and depth videos can be opened the visibility is very limited, specially in the case of the depth video (images are almost completely black). This is because the videos use the Gray scale 16 bit format in which each pixel corresponds to an ushort value (values range from 0 to 65535) that are then transformed into a shade of grey. So without a mask to change the brightness or without colorizing the images it is difficult to really see something. On the fig 5.7 is an image of the IR video being played on a normal video player in a moment that the image can be perceived. The images were saved this way because the requirements asked for raw images and using masks would change the values of the pixels. The last file created is a text file that contains the information of each frame acquired (width, height, image size and datastamps). These files are now prepared to be inserted in the KiMA and KiSA software for further analysis.

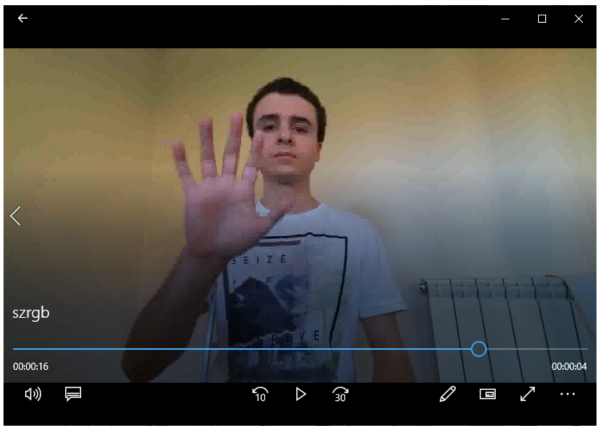


Figure 5.6: The color video separated from the original videos recorded by the Azure KiT 1

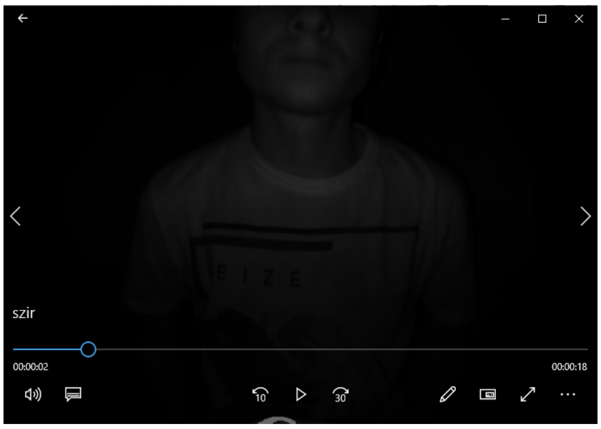
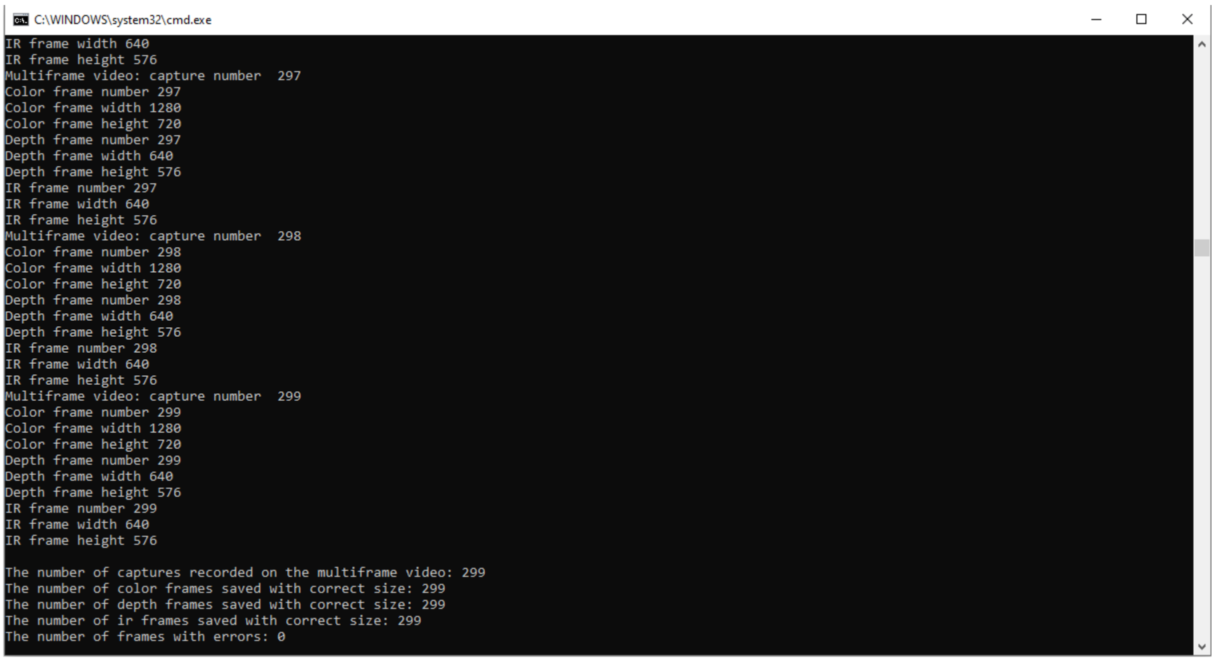


Figure 5.7: The IR video separated from the original videos recorded by the Azure KiT 1

5.2.1 Data Verification Test

With the development of the Azure KiT application completed there are some tests that need to be done in order to verify the data produced by the application. To do that a simple application was developed that could analyse the data and find out whether it was correctly segmented into the different types of frames and if there are any frames missing.

This test main objective is to compare the data from the multiframe video recorded by the first application (Azure KiT part one) with the color, depth and ir videos made from it. The test opens the multiframe video using the azure kinect SDK and also the three videos using the Accord.Video.FFMPEG.x64 package. The number of frames obtained from each video file is accounted in order to see if no frame is missing. While the frames as being obtained they are transformed into a bitmap to see whether they have the same size or not. In the end, any discrepancy with the number or the size of the frames is noted. In fig. 5.8 the test result done to a 10 second video is demonstrated. The number of frames accounted for were 299 that shows a frame rate of 30 fps (29.9) which is what was expected by the azure kinect device. This test also shows that all the frames obtained from the videos add the correct size and no frame was missing.



```
IR frame width 640
IR frame height 576
Multiframe video: capture number  297
Color frame number 297
Color frame width 1280
Color frame height 720
Depth frame number 297
Depth frame width 640
Depth frame height 576
IR frame number 297
IR frame width 640
IR frame height 576
Multiframe video: capture number  298
Color frame number 298
Color frame width 1280
Color frame height 720
Depth frame number 298
Depth frame width 640
Depth frame height 576
IR frame number 298
IR frame width 640
IR frame height 576
Multiframe video: capture number  299
Color frame number 299
Color frame width 1280
Color frame height 720
Depth frame number 299
Depth frame width 640
Depth frame height 576
IR frame number 299
IR frame width 640
IR frame height 576

The number of captures recorded on the multiframe video: 299
The number of color frames saved with correct size: 299
The number of depth frames saved with correct size: 299
The number of ir frames saved with correct size: 299
The number of frames with errors: 0
```

Figure 5.8: Test done to the azure KiT application that tests a 10 second video

5.3 Discussion

The results presented on the previous sections of this chapter represent what was achieved of the requirements established on the chapter 4. In the Azure KiT part one almost everything was achieved with success. The video files were correctly saved, as proven by the ability to open them with the viewer tool of the Azure Kinect. The interface provides the necessary tools for the users to correctly utilise the application. The application also correctly shows the color image at the same time that the images are being received from the Kinect device. The only point that this application may be lacking is showing the depth or IR image as well as the color image that is already being shown, like the KiT for Kinect V2 did. Right now the application shows that the color camera is working correctly but the state of the images acquired by the depth camera is unknown. However this is not a great problem since the configurations of the device were set to only capture the images that were synchronised, so if there is a problem with the depth camera, the color image won't be acquired as well.

The Azure Kit part two was equally successful to the part one and even improved the quality of the data acquired when in comparison to the KiT for Kinect V2. This application achieved all the requirements presented in chapter 4. The interface of the application, however, is a little bit different than the mockup presented. Besides asking the user for the folder where the videos were recorded in the first application and the folder where the program should save the files processed in this application, the user should also specify the name of the session because many seizures can be identified in a session and this will help in differentiating them. The rest of the interface is equal to what was depicted on the mockup. When the application terminates processing the videos, four files are created, on for each image type and the last one is the text file containing important information about each frame. As it was mentioned before, the frames come in triples because they are always synchronised, so the information on the text file is also organised in triples. Each line with the information related to the three frames that were acquired simultaneously. The thing to notice here is that in the previous KiT, the data was saved in binary files that were difficult to read and the information wasn't always correct (problems documented on chapter 3). However, now the information is saved in video files that can be easily opened and that can also be accessed with multiple programming languages without any problem.

To summarise, both the applications were completed with different degrees of success but in the end, the product that is generated by them goes along what was expected of the requirements defined for it.

Chapter 6

Conclusions and Future Work

The main objective of this thesis was to understand if the new generation of the Kinect from Microsoft (Azure Kinect) could be introduced into the existing systems that monitor the patients with neurological diseases that affect their normal movement. The two systems that were introduced, one in the field of epilepsy and the other in PD, both used the application KiT which utilised the functions of the Kinect V2 from Microsoft. The KiT, despite the problems described in this thesis, was already a very mature application since it started with the first Kinect developed by Microsoft. However, the Azure Kinect proved to be very different from its predecessors which doesn't mean that it is bad. Although those differences were some of the reasons to create a new KiT from scratch, the new possibilities created by the SDK of the Azure Kinect were also crucial in this decision. The possibility to record and then playback the files is exactly what the KiT needed in order to tackle some of the problems that were verified through the years.

With the development of a new KiT set in stone came new difficulties that were not expected in this thesis but overall the objective was achieved. Even though the results deviate from what was expected at the beginning, an application that gathers the necessary information for the applications that needed them was created. So it can be said that with some adjustments this application can potentially be inserted into the clinical environment alongside the previous KiT or alone.

6.1 Obstacles

During this thesis there were many obstacles that appeared and needed to be tackled. Some were resolved and actually opened new paths, others weren't and changed the way the work was being planned and done. Some of these obstacles should be addressed if future work is done with the results obtained in this thesis. The obstacles were the following:

- The fact that the Azure Kinect SDK separated the color, depth and IR frames from the body frames made it very difficult to integrate into the existing KiT since in the Kinect V2 those frames could be obtained simultaneously and synchronised.

- When dealing with the Azure Kinect device, the lack of experience in the field of signal processing and computer vision made more difficult to tackle some problems that appeared along the way.
- The programming language, the frameworks and the packages were all new technologies which added to the creation of an application that dealt with hardware that was introduced less then one year before the start of this thesis, proved to be very difficult and impacted the results of this thesis.
- The Azure Kinect device also only works with 64-bit programs which caused some constraints during the development process of the applications. Many of the libraries available were old and were made for 32-bit programs which normaly is not a problem due to the fact that the programs can function in both modes.

6.2 Future Work

Motion capture systems are already inserted in the clinical context and although they have already presented very promising results (chapter 2), they haven't yet reached their full potential. The work being done using the Kinects from Microsoft and the software KiT, KiMA and KiSA are great and are walking towards the goal of achieving a better understanding of neurological diseases. Each new advancement brings new information about these diseases that help to better diagnose and treat the patients that suffer from them.

This thesis presented a new tracking application using the Azure Kinect, based on the previous applications that worked with older versions of the Kinect and it still doesn't do everything that those applications did. There is always space for improvement and the application developed during this thesis is the same. In the first part of the application developed, showing the depth image could give much more feedback to the user and would assure them that the data was being acquired correctly. Although the color camera already assures that the images are being acquired, if the depth camera has problems (pixels that are not acquired correctly for example) it won't be known until the video is seen. The second part of the application should also let the user set multiple starts and end dates so that the user doesn't need to be constantly looking at the application, waiting for it to finish the process before typing the next set of dates.

In the future trying to integrate the body tracking SDK of the Azure Kinect into this application should also be considered since this is the big module of the KiT that is not currently in the Azure KiT developed in this Thesis. This way the gait analysis process present in the KiT could be integrated using the Azure Kinect.

References

M. d. C. Vilas-Boas J. M. Fernandes A. P. Rocha, H. M. P. Choupina and J. P. S. Cunha. System for automatic gait analysis based on a single RGB-D camera. *PLoS ONE*, 13, 2018.

Abdul Hadi Abdul Razak, Aladin Zayegh, Rezaul K Begg, and Yufridin Wahab. Foot plantar pressure measurement system: A review. *Sensors*, 12(7):9884–9912, 2012.

Accord.NET. Accord.video.ffmpeg.x64 3.8.2-alpha, 2017. URL https://www.nuget.org/packages/Accord.Video.FFMPEG.x64/3.8.2-alpha.

Jake K Aggarwal and Quin Cai. Human motion analysis: A review. *Computer vision and image understanding*, 73(3):428–440, 1999.

Ali Al-Naji, Kim Gibson, Sang-Heon Lee, and Javaan Chahl. Real time apnoea monitoring of children using the microsoft kinect sensor: a pilot study. *Sensors*, 17(2):286, 2017.

Swati; Thompson Barry; Elkhatib Tamer; Wurster Stefan; Akkaya Onur; Payne Andrew; Godbaz John; Fenton Mike; Rajasekaran Vijay; Prather Larry; Nagaraja Satya; Mogallapu Vishali; Snow Dane; McCauley Rich; Mukadam Mustansir; Agi Iskender; McCarthy Shaun; Xu Zhanping; Perry Travis; Qian William; Chan Vei-Han; Adepu Prabhu; Ali Gazi; Ahmed Muneeb; Mukherjee Aditya; Nayak Sheethal; Gampell Dave; Acharya Sunil; Kordus Lou; O'Connor Pat Bamji, Cyrus S.; Mehta. [ieee 2018 ieee international solid - state circuits conference - (isscc) - san francisco, ca, usa (2018.2.11-2018.2.15)] 2018 ieee international solid - state circuits conference - (isscc) - impixel 65nm bsi 320mhz demodulated tof image sensor with 3m global shutter pixels and analog binning. 2018. ISBN 978-1-5090-4940-0. doi: 10.1109/ISSCC.2018.8310200.

bibigone and baSSiLL. K4adotnet 1.4.0, 2020. URL https://www.nuget.org/packages/K4AdotNet/.

G. R. Bradski. Real time face and object tracking as a component of a perceptual user interface. In *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*, pages 214–219, Oct 1998. doi: 10.1109/ACV.1998.732882.

H.-J. Chen, Z.; Lee. Knowledge-guided visual perception of 3-d human gait from a single image sequence. *IEEE Transactions on Systems Man and Cybernetics*, 22, 1992. doi: 10.1109/21.148408.

Djalma; Velho Luiz Cruz, Leandro; Lucio. [ieee 2012 xxv sibgrapi conference on graphics, patterns and images tutorials (sibgrapi-t) - ouro preto, brazil (2012.08.22-2012.08.25)] 2012 25th sibgrapi conference on graphics, patterns and images tutorials - kinect and rgbd images: Challenges and applications. 2012. ISBN 978-0-7695-4830-2,978-1-4673-5091-4,. doi: 10.1109/SIBGRAPI-T.2012.13.

João Paulo Silva Cunha, Hugo Miguel Pereira Choupina, Ana Patrícia Rocha, José Maria Fernandes, Felix Achilles, Anna Mira Loesch, Christian Vollmar, Elisabeth Hartl, and Soheyl Noachtar. Neurokinect: A novel low-cost 3dvideo-eeg system for epileptic seizure motion quantification. *Plos One*, 11(1), 2016. doi: 10.1371/journal.pone.0145669.

JP Silva Cunha, C Vollmar, Zj Li, J Fernandes, B Feddersen, and S Noachtar. Movement quantification during epileptic seizures: a new technical contribution to the evaluation of seizure semiology. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439)*, volume 1, pages 671–673. IEEE, 2003.

Pietro; Cortelazzo Guido M Dal Mutto, Carlo; Zanuttigh. *[SpringerBriefs in Electrical and Computer Engineering] Time-of-Flight Cameras and Microsoft Kinect^{TM}* ||, volume 10.1007/978-1-4614-3807-6. 2012. ISBN 978-1-4614-3806-9,978-1-4614-3807-6. doi: 10.1007/978-1-4614-3807-6.

C Gabriel, S Gabriel, and E Corthout. The dielectric properties of biological tissues: I. literature survey. *Physics in Medicine and Biology*, 41(11):2231–2249, nov 1996. doi: 10.1088/0031-9155/41/11/001. URL https://doi.org/10.1088%2F0031-9155%2F41%2F11%2F001.

Giuliana Grimaldi and Mario Manto. Neurological tremor: sensors, signal processing and emerging applications. *Sensors*, 10(2):1399–1422, 2010.

David Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1, 1983. doi: 10.1016/0262-8856(83)90003-3.

G. Johansson. Spatio-temporal differentiation and integration in visual motion perception. *Psychological Research*, 38, 06 1976. doi: 10.1007/bf00309043.

M.K.; Yee-Hong Yang Leung. First sight: A human body outline labeling system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17, 1995. doi: 10.1109/34.385981.

C.; Yi-Chen Wei; Jian Sun; Baining Guo Leyvand, T.; Meekhof. Kinect identity: Technology and experience. *Computer*, 44, 2011. doi: 10.1109/mc.2011.114.

Soheyl Noachtar and Astrid S. Peters. Semiology of epileptic seizures: A critical review. *Epilepsy Behavior*, 15(1):2–9, 2009. doi: 10.1016/j.yebeh.2009.02.029.

NuGetGallery. Nuget gallery: Home, 2020. URL https://www.nuget.org/.

Rebecca O'Dwyer, Joao P Silva Cunha, Christian Vollmar, Cordula Mauerer, Berend Feddersen, Richard C Burgess, Alois Ebner, and Soheyl Noachtar. Lateralizing significance of quantitative analysis of head movements before secondary generalization of seizures of patients with temporal lobe epilepsy. *Epilepsia*, 48(3):524–530, 2007.

Margo N Orlin and Thomas G McPoil. Plantar Pressure Assessment. *Physical Therapy*, 80(4):399–409, 04 2000. ISSN 0031-9023. doi: 10.1093/ptj/80.4.399. URL https://doi.org/10.1093/ptj/80.4.399.

Ana Patricia Rocha, Hugo Choupina, Jose Maria Fernandes, Maria Jose Rosas, Rui Vaz, and Joao Paulo Silva Cunha. Parkinsons disease assessment based on gait analysis using an innovative rgb-d camera system. *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2014. doi: 10.1109/embc.2014.6944285.

Ana Patricia Rocha, Hugo Choupina, Jose Maria Fernandes, Maria Jose Rosas, Rui Vaz, and Joao Paulo Silva Cunha. Kinect v2 based system for parkinsons disease assessment. *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015. doi: 10.1109/embc.2015.7318601.

Hugo; Fernandes Jose Maria; Rosas Maria Jose; Vaz Rui; Silva Cunha Joao Paulo Rocha, Ana Patricia; Choupina. [ieee 2014 36th annual international conference of the ieee engineering in medicine and biology society (embc) - chicago, il (2014.8.26-2014.8.30)] 2014 36th annual international conference of the ieee engineering in medicine and biology society - parkinson's disease assessment based on gait analysis using an innovative rgb-d camera system. 2014. ISBN 978-1-4244-7929-0. doi: 10.1109/embc.2014.6944285.

Daniel Roetenberg. *Inertial and magnetic sensing of human motion*. s.n., 2006.

Daniel Roetenberg, Henk Luinge, and Per J. Slycke. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. 2009.

Damien; Kolb Andreas Sarbolandi, Hamed; Lefloch. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 5 2015. doi: 10.1016/j.cviu.2015.05.006.

M. Shah, K. Rangarajan, and P. . Tsai. Motion trajectories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1138–1150, July 1993. ISSN 2168-2909. doi: 10.1109/21.247894.

Skarredghost. All you need to know on azure kinect, Jan 2020. URL https://skarredghost.com/2019/02/25/all-need-know-azure-kinect/.

Tesych. Azure kinect dk depth camera, https://docs.microsoft.com/pt-pt/azure/kinect-dk/depth-camera. URL https://docs.microsoft.com/pt-pt/azure/Kinect-dk/depth-camera.

Tesych. Azure kinect dk hardware specifications, 2019. URL https://docs.microsoft.com/en-gb/azure/Kinect-dk/hardware-specification.

UPDRS. The unified parkinsons disease rating scale (updrs): Status and recommendations. *Movement Disorders*, 18(7):738–750, 2003. doi: 10.1002/mds.10473.

Maria Do Carmo Vilas-Boas and João Paulo Silva Cunha. Movement quantification in neurological diseases: Methods and applications. *IEEE Reviews in Biomedical Engineering*, 2016.

WHO. Epilepsy, world health organization, https://www.who.int/en/news-room/fact-sheets/detail/epilepsy. URL https://www.who.int/en/news-room/fact-sheets/detail/epilepsy.

WHO. *Neurological disorders: public health challenges*. World Health Organization, 2006.

Xceed. Extended.wpf.toolkit 4.0.1, 2020. URL https://www.nuget.org/packages/Extended.Wpf.Toolkit/.

Junyi Xia and R Alfredo Siochi. A real-time respiratory motion monitoring system using kinect: proof of concept. *Medical physics*, 39(5):2682–2685, 2012.

Gene; Chan Kevin; Stankovic Vladimir Yang, Cheng; Cheung. [ieee 2014 ieee international conference on multimedia and expo workshops (icmew) - chengdu, china (2014.7.14-2014.7.18)] 2014 ieee international conference on multimedia and expo workshops (icmew) - sleep monitoring via depth video compression analysis. 2014. ISBN 978-1-4799-4717-1. doi: 10.1109/icmew.2014.6890645.

Zeev Zalevsky, Alexander Shpunt, Aviad Malzels, and Javier Garcia. Method and system for object reconstruction, March 19 2013. US Patent 8,400,494.