|  |  |  |
| --- | --- | --- |
|  | **Departamentul Automatică şi Informatică Industrială**  **Facultatea Automatică şi Calculatoare**  **Universitatea POLITEHNICA din Bucureşti**  Splaiul Independenţei 313, 060042, Bucureşti, România  Sala ED 412, Tel. 021/402.92.69  [www.aii.pub.ro](http://www.aii.pub.ro), email: secretariat@aii.upb.ro |  |

|  |  |
| --- | --- |
|  |  |

**Sesiunea de Comunicări Şiinţifice Studenţeşti**

**Ediţia 2021**

**Interacțiunea cu programele de grafică moleculară folosind gesturi naturale**

**Autori: Alexandru TOMESCU, Andrei CAPOTA, anul III, grupa 332AA**

**Adresa e-mail: alexandrutomescu23@gmail.com**

**Îndrumător ştiinţific : Prof. dr. ing. – dr. sc. nat. Cătălin BUIU**

Table of Contents

[**Abstract:** 3](#_Toc71748829)

[1. Introduction 3](#_Toc71748830)

[2. Gestures 4](#_Toc71748831)

[2.1 Gestures and computer interaction introduction 4](#_Toc71748832)

[2.2 Temporal classification 5](#_Toc71748833)

[2.3 Contextual classification 5](#_Toc71748834)

[2.4 Prescribed gestures 6](#_Toc71748835)

[2.5 Head motion 6](#_Toc71748836)

[3. Technology 6](#_Toc71748837)

[3.1 Types of application 7](#_Toc71748838)

[3.1.1 3D modelling 7](#_Toc71748839)

[3.1.2 Assistive application 8](#_Toc71748840)

[3.1.3 Data input/authentication 8](#_Toc71748841)

[3.1.4 Manipulation/navigation 8](#_Toc71748842)

[3.1.5 Touchless control 9](#_Toc71748843)

[3.2 Hardware used 10](#_Toc71748844)

[3.2.1 Microsoft Kinect 10](#_Toc71748845)

[3.3 Software used 12](#_Toc71748846)

[4. Methodology 13](#_Toc71748847)

[4.3 UCSF Chimera Application 14](#_Toc71748848)

[4.3.1 Body recognition and head position tracking workflow 14](#_Toc71748849)

[4.3.2 Chimera script workflow 14](#_Toc71748850)

[4.3.3 Results 16](#_Toc71748851)

[4.3.4 Conclusions for this application 17](#_Toc71748852)

[4.4 PyMol Application 17](#_Toc71748853)

[4.4.1 Graphical User Interface 17](#_Toc71748854)

[4.4.2 Head Tracking 18](#_Toc71748855)

[4.4.3 Hand Tracking (Zoom) 18](#_Toc71748856)

[4.4.4 Hand Tracking (Fingers Count) 19](#_Toc71748857)

[4.4.5 Application Flow 20](#_Toc71748858)

[5. Conclusions and Further Developments 21](#_Toc71748859)

[6. References 22](#_Toc71748860)

# **Abstract:**

Gestures are considered as the natural mode of interaction with the environment, have already been used in many applications for interacting with computers. In this paper we want to explore the idea of using head and hand tracking to interact with software applications that are used for three-dimensional manipulation and analysis of biological macromolecules. We will use a Kinect v2 (One) for head tracking, computer’s camera for hand tracking and Pymol for applying that movement to a molecule.

Keywords: molecular graphics, human-computer interaction, gesture recognition, Kinect, Chimera, Pymol, human motion recognition, interactive application

# Introduction

Gestures are a natural form of human expression and, in particular, hand gestures, including finger and arm movement, can be widely used as an interaction interface. Gestures are often used for communication and many times are accompanying speech to give deeper meaning or to add another layer of explanation for words or actions. Since the early 1980s [1] they have been considered as an alternative way for interacting with computers and other appliances because of the ease of use it brings or to aid certain groups in their work.

Molecular graphics can be defined as a technique of studying molecules and their properties on a graphical display device. Molecular graphical programs have become very powerful tools for people working in this field. Such programs are UCFS Chimera, PyMOL[2], VMD [3], NAMD[4] or Jmol [5] and all share in a common powerful user interface with capabilities for both mouse input and commands. The developments in graphics card technology helped this application develop even more, having a wide range of functionality [6].

Advancements in development in other fields such as computer science and Internet of Things contributed to bringing to the market more affordable and powerful hardware for researches to developed interactive applications for different areas of interest. Microsoft Kinect became very popular, an example can be [7], where Kinect-based games are used to see how effective they can be in language teaching.

Visualizing and manipulating three-dimensional objects proved to be one of the most popular fields along with touchless control where the natural movement of the hand feels natural for a human to interact and control different objects. In [8] we can see how Kinect is used to recognize gesture and based on them give commands to an ultrasonic robotic system.

Another use for gesture input can be seen in the introductory chemistry and biochemistry courses where a more immersive experience, easier control over picking, moving, exploring bond-breaking and forming can draw in more students. Results for such an experiment were seen in [9], where students that have seen a demo for movement of a molecule using head tracking grew their interest for bioinformatics and molecular graphics.

The application we will present in this paper is the Interactive Graphical Visualization of Biomolecules using Real-Time Head Tracking. Simple natural gestures, like head motion (head up and down, tilted left and right) can be an easy way to interact with molecular graphics programs. Head motion is a simple and natural gesture, but also suggestive, and it can be tracked live (real-time tracking). As [9] relates, the target group for this application is represented by biochemistry students or maybe even scientists willing to visualize some elements in a more intuitive environment. The hypothesis is that the use of natural gestures while interacting with molecular graphics programs will increase the motivation and interest of its users. According to the survey presented in [9], the first results indicated the education efficiency of this interactive application. The main goal is to use gestures such as the ones presented above in order to work with programs for the interactive visualization and analysis of molecular structures and related data such as Chimera, Pymol, etc.

The overall goal of the research in the current paper it to explore the idea of developing a tool for interacting in a natural manner with molecular graphics programs. The goal is to achieve something simple and intuitive that would maybe spark the interest for new people to explore this domain. The technologies used in this research are presented in detail, and so is the flow of the program. This paper ends with the conclusion and current limitations for the software and hardware used and recommendations for further research.

# Gestures

## Gestures and computer interaction introduction

Gestures are movements that communicate a feeling or instruction, a practical and natural form of human expression, and hands a natural mode of interaction with the physical world and objects in it. Gestures are used on a large scale to help humans express easier and faster, but they can also be used as encoded messages or even as a language itself for people with disabilities in this regard. They vary from gestures that do not convey a specific meaning and simply follow the rhythm of the speech or just simple actions used to designate, point at, or manipulate objects, to those with meaning and symbolizing specific concepts, things that would be difficult to convey by speech alone. Visual information and verbal information are processed in different ways, and by different parts of the brain. Each has its own strengths, and often both should be combined in a presentation.

In recent years, gestures in human communication have been increasingly studied. Unlike other modalities such as speech, gestures are bi-directional, acting directly on the environment and reacting to modifications of this environment. As [10] relates, the gesture modality includes a semiotic function - the ability to create representations of things so that we can communicate with each other - because gestures and body movements are capable of driving messages.

Nowadays, with this huge development in information technology and tools associated with it, human beings are trying to reduce workload and increase productivity by using machines. Task automation by finding better ways to communicate with these machines is one of the key goals we are trying to achieve. In Human-Computer Interaction (HCI), hand gesture recognition is broadly applied in many practical applications, such as sign language recognition, virtual controllers, remote control, or immersive gaming technology. Therefore, gesture recognition of the human hand or head from images or even live videos has been one of the goals of computer vision and gestures used in human-computer interaction interfaces for several decades. According to [11], development of Kinect and LEAP sensors, which are portable and supported by Software Development Kits (SDKs) enabling simpler implementation, seems to have contributed significantly to the expansion of the field on gesture-based interfaces since 2013.

With the massive development in this direction, new systems try to take advantage of the expressive power of gestures. At first, gesture interaction had been reduced to simple command interfaces. More recently, techniques such as capturing body movements, recognizing, and interpreting human actions and animating virtual humans or molecular interaction using gestures, have given rise to several virtual reality applications with more natural interfaces.

## Temporal classification

Based on their temporal characteristics, gestures can be classified as static or dynamic. Static gestures are those where only the static position of a hand is observed. These types of gestures are also called postures. These types of gestures are easier to track, but they are not that expressive. Dynamic gestures are those in which a hand moves between several positions to form a full gesture. A dynamic gesture is considered to have three to five phases, depending on the specific gesture performed: preparation, prestrike hold (if specific gesture contains it), stroke, post-stroke hold (if specific gesture contains it), and retraction. Dynamic gestures can be characterized as multiple frames of static gestures that give full meaning after it is completed.

## Contextual classification

Based on what they are used for, and the context they are used in, gestures can be classified as communicative or manipulative. Applications that use communicative gestures can be used for example to help children with Down's syndrome: the role of gestures as a 'bridge' between word comprehension and word production; and the predictive role of gestures, in association with comprehension, on vocabulary development.

According to [11], these communicative gestures can be classified in multiple groups that will be presented below.

The first group of gestures are independent communicative gestures. Even if they are still having a communicative purpose and most frequently are used concurrently with speech, they can be independent from speech and do not require it to convey a meaning. Symbolic gestures represent a symbolic object or concept, devoid of any morphological relation with what is being referred to. They have a direct translation into words, are used deliberately to send a particular message, and have a widely accepted meaning, some examples are "thumbs up" to indicate approval or hand waving as a greeting.

Another group is speech related and complement with what is being communicated verbally. Iconic gestures "represent meaning closely related to the semantic content of the speech". This form of nonverbal communication is used to emphasize the message. Speech-related gestures are intended to provide supplemental information to a verbal message such as pointing to an object of discussion and illustrate what is being said.

On the other hand, manipulative gesture classification is not that vast. Gestures are generally considered manipulative if they interact with and modify a spatial component of an object in an interface. [11] states that in manipulative gestures "hand motion indicates a path or a placement". Some gestures can be used to both communicate and manipulate as some manipulative gestures communicate the spatial position of objects, their size or even the ways they are manipulated.

## Prescribed gestures

Based on levels of instruction given to guide the gesture performance, gestures can be classified as prescribed or free-form gestures. Prescribed gestures use a predefined dictionary in which each instruction is explained. The problem here is that users of an application have to learn these gestures and the performance of a predefined gesture triggers a predefined action. Prescribed gestures may increase the cognitive load as [11] and [12] specify, their learning rates depend on the users' cognitive skills, and their use forces the users to learn and use gestures they perhaps would not choose themselves. The advantage of using these prescribed gestures is in implementation because you know what input to expect and that makes implementation easier.

## Head motion

When we talk about head movements, an innate head gesture assumes a headshake back and forth in order to say “no”. These kinds of movements are acquired naturally at birth, even before we get the ability to understand their corresponding meaning. We also move our heads when we want to indicate interest or approval. For example, a head up typically indicates an engaged or neutral attitude. Also, a head down signal can have the meaning of a negative or aggressive attitude.

# Technology

The types of application in which the gestures are used appear to influence the type of technology used to capture, track, and recognize these gestures. Some examples of the types of application are medical imagery, where the risk of contamination is reduced by using touchless interaction, driving, where people need to stay focused, or for industrial workers with dirty hands that need information from their equipment. Depending on the application and the accuracy needed, one can use infrared cameras, such as Kinect, depth Camera, Leap sensors or wearables, such as accelerometers, gyroscopes.

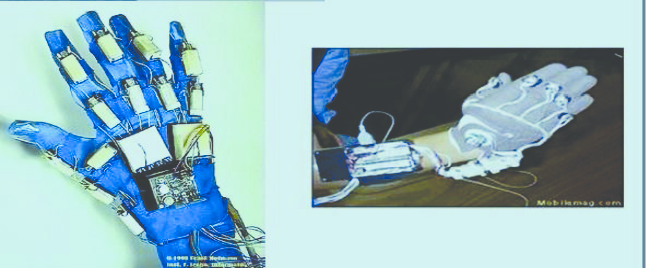
 However, the history of hand gesture recognition for computer control, according to [13], started with the invention of the glove-based control interfaces. Researchers realized that gestures inspired by sign language can be used to offer simple commands for a computer interface. This gradually evolved with the development of much more accurate accelerometers, infrared cameras, and even fiber-optic bend-sensors (optical goniometers).

Figure 1. Wearable glove for gesture

Some of those developments in glove-based systems eventually offered the ability to realize computer vision-based recognition without any sensors attached to the glove. These are the colored gloves or gloves that offer unique colors for finger tracking ability that had a huge impact on developing computer vision based on gesture recognition.

## Types of application

First gesture-based application appeared in literature from 1980 [1], where voice and gesture were used to command simple shapes on display. After this occurrence, gestures began to be used in multiple domains, from 3D/VR object manipulation to robot control and Computer Aided Design (CAD) and further technological developments in technology in the form of an infrared camera, depth-sensing camera or motion sensors made the research more affordable and easier to get into. Thus, the interest in this domain started raising.

Another domain that became very popular and had seen a lot of interest was gaming, where 3D games became more widely spread and companies began searching for more ways of interacting with the games. One of the earliest gesture-based controllers was the Wii Remote developed and released by Nintendo in 2005, followed in 2010 by the Kinect, but more information on those in Data input/authentication.

How hand or head gestures are used in a specific interface appears to be influenced, to a certain level, by the type of the application and the purpose it serves, type of technology facilitating its implementation, and the underlying type of gesture capture, tracking and recognition supporting it. A general schema for human computer interaction interfaces can be represented as following:

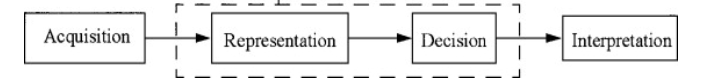


Figure 2 image extracted from [14]

These four steps presented below represent: acquisition is the process of receiving data from a device and converts the physical gesture data into numerical data. The main system of this workflow consist of two processes: the representation and the decision processes. The representation process converts the raw numerical data into a form adapted to the decision process, which then classifies the data. The interpretation process gives the meaning of the symbol series coming from the decision process.

According to [11] application can be separated into five categories : 3D modelling, assistive application, data input/ authentication, manipulation/navigation and touchless control.

### 3D modelling

This sort of application revolves in using gesture to make new forms, modify them by changing geometric characteristics or changing their position and shape in 3D space.

Such an example is shown in [15] where simple hand movement is used for the manipulation of objects. Hands are the best alternative because the connection between the user and computer is wireless, the movement of the hands has no constraints in the 3D space. But this implies other problems, because the user can move freely there needs to be a robust recognition of the gestures. Therefore, there must be a careful selection when it comes to the gesture spectrum because there is a fine balance between gesture complexion and functionality.



In applications like the one mention before, prescribed gestures are often used because they are connected to certain actions in the program, which is easier to manage and implement a feasible system that can recognize them.

Figure 3. User interacts with a 3D virtual model by using non-instrumented hand gestures [15].

### Assistive application

Assistive applications are being targeted at elderly people or people with some impairment to help them interact with electronic devices, computers providing additional aid in assistive living environments. For example, in [16], gestures are used to aid blind people by implementing an online newsreader. Prescribed gestures are chosen for this application too for the same reason as seen in 3D modelling, having functions like stop, pause, restart and resume for communicating with the application.

### Data input/authentication

Data input/ authentication applications are generally used for easing the interactions between the user and computes. Another application is recognizing handwriting or sign language which can be used in educational purposes too.

### Manipulation/navigation

These types of applications offer the user a more intuitive way to interact with an object or navigate with some unique environments. Some examples of the usage of gesture for manipulation/navigation can be seen in:

* Interaction with a display or a projection.
* Interaction with augmented reality enables overlapping 3D objects with the environment that you are working in.
* Navigating in an application
* Robot interaction

As pointed out in [17], using and interacting with machines using gestures has various applications, including Virtual Reality (VR) or Augmented Reality (AR). Gesture interaction is one of the most efficient interaction methods in AR and can be used to navigate through content, interact with virtual objects or play games. Augmented Reality provides a new user experience in smartphones where solutions are needed to solve how to interact efficiently and conveniently. Using gestures allows more of the screen to be dedicated to the AR experience and makes the experience more immersive. Also, when a user interacts with overlaid AR objects on a smartphone, gesture interaction provides a free line of sight and the ability to easily move objects in 3 dimensions.

These use cases use gesture control more like an add-on because of the hardware and software limitations and mostly use prescribed gestures for actions.

### Touchless control

A person holding a video game controller

Description automatically generated with medium confidence Touchless control applications have a wider spectrum of usage, from games, home appliance control to interacting with car controls and robots.

As discussed before, touchless control in games has seen a lot of interest. Nintendo Wii was one of the earliest implementations of such a technology. The Wii Remote controller had as inputs an accelerometer and a gyroscope and paired with a motion sensor bar mounted onto a television screen enabled motion tracking.

More gaming companies followed soon after and released their own versions in the form of PlayStation Move (Sony) and Kinect 360 (Microsoft)

Despite the initial interest in those technologies, the implementations in games were poorly designed and were not very successful in the long run. This fall was accentuated by the rising popularity of VR which had much more potential in the gaming industry, but the efforts in improving the technology helped in bringing the more affordable options for research using infrared cameras and motion sensors, especially Kinect which was refreshed in 2014 with an improved performance in tracking and furthermore in 2019 was released Azure Kinect which is a developer kit for the use of artificial intelligence (AI) sensors for computer vision and speech models.



Another use for gestures touchless controls is interacting with car entertainment system, for example, BMW implemented such a system that helps drivers be more focused on the road and use gesture for music volume control, managing calls, or selecting infotainment settings [18].

Figure 4. Example of a use case for BMW gesture controls

## Hardware used

The technology used for gesture tracking can be divided into two categories:

* Use of cameras: gestures are recorded at a distance by a camera or sensor.
* Use of wearable: users wear certain devices (gloves, bracelets, rings) with sensors.

Primarily the cameras are more popular because it is easier to work with them and despite the superior accuracy that wearables provide, cameras have a lot more flexibility and functionality being only dependent on the ones that use them.

For our specific research, we will focus more on the Microsoft Kinect because it is our camera of choice and one of the most popular for gesture tracking and recognition.

### Microsoft Kinect

Microsoft Kinect was initially developed for use in combination with an Xbox console to eliminate the game controller.

A picture containing diagram

Description automatically generatedThe Kinect 360 was originally launched in 2010 and incorporated an RGB camera, infrared projectors and detectors that mapped depth through though structural light calculations, a microphone array, along with software and artificial intelligence to allow real-time gesture and speech recognition and skeletal body detection, up to 4 people.

However, Kinect had found an unexpected interest from the academic and commercial application because was cheaper and more robust compared to other depth-sensing technology on the market, being thus considered for application in robotics, medicine, and health care. In 2012, Microsoft released Kinect for Windows, a development kit for commercial applications.

Graphical user interface

Description automatically generated In 2014, Microsoft released a refresh in the form of Kinect One, which had improved RGB camera, improved Depth Camera, and a new way to calculate depth: time-of-flight. In the past, it was hard to use more than one Kinect since they interfere a lot with each other because they calculate depth using an IR light pattern projection, whereas the Kinect One computes the depth of objects it has in front of it throwing some infrared light rays and looking how much time these rays need to bounce on surfaces and come back. This method is more stable, precise, and less prone to interferences.

 Ultimately Kinect line for Xbox was discontinued, and Microsoft released a non-gaming version as the Azure Kinect, which incorporates Microsoft Azure computing applications among the device's functionalities.

 We briefly mentioned that Kinect uses a depth-sensing camera, but what is that and how does it work? According to [19], the IR emitter on Kinect 360 beams structured light with predefined patterns to the objects in the field of view. By observing the unique pattern, the depth sensor can infer the line from the IR emitter to the pixel with the pattern. Hence, the depth sensor can calculate the vertical distance between the IR emitter-depth sensor line to the pixel using trigonometry, which is the depth reading of the pixel. However this had its flows because for the depth sensing to be perfect you need a unique visible pattern for each pixel and there had to be enough space between two adjacent dots for the depth sensor to distinguish and that leads to having about 1 in 20 pixels with a true depth measurement, the others being interpolated [20].On the other hand, Kinect One uses the time-of-flight technology [21] and can be broken down to a fairly standard camera sensor that adds a timing generator to it and connects this to an IR light source so the sensor can tell when the light is on and when it is off. This technology is based on that developed by Canesta, which was acquired by Microsoft in 2010 [22].

Figure 5. Depth Sensor Kinect

 Another important feature for Kinect is the Human skeleton estimation, a technique that recognizes the human body and identifies its specific joints. It does this by using a version of skeleton estimation which is included in the Kinect SDK. Firstly, it retrieves the stream from the depth frames, on which it performs a subject foreground extraction and detection. After the matching, the extracted humans subject against a trained model in overlays the skeleton joins once the current pose is estimated and.

This feature has facilitated Kinect application development because its robust human pose estimation frees the application developers from dealing with it and thus making research and software development in the gesture field easier.

Figure 6. Kinect skeleton estimation

## Software used

As presented before, there are a lot of biomolecular visualization programs, but we will focus particularly on UCSF Chimera [23] because it was the one that we use for this part of the research.

UCSF Chimera is a program for the interactive visualization and analysis of molecular structures and related data, including density maps, trajectories, and sequence alignments, developed by the Resource for Biocomputing, Visualization, and Informatics (RBVI) at the University of California, San Francisco and partially supported by the National Institute of Health.

One of the reasons why we used Chimera is that it has a built-in Python interpreter and support for use of specific commands and methods in Python scripts, thus the development for this "should" be an easy task, but we later discovered that being an older distribution of Python, Python 2.7, which caused some incompatibility issues that we will discuss in a later section.

Chimera offers two methods for scripting:

* Using Chimera commands in Python code, means you can use Python to record commands or give them to Chimera in code.
* Using Midas module, which is more flexible to use if you know programming and in general, there is a one-to-one correspondence to the basic commands.

PyMol is another alternative for molecular visualization, and its main advantage is that it offers support for the latest versions of Python, Python 3.7, and has an alternative way to embed directly into python projects. We used this program later as an alternative for Chimera because it offers the same benefits, but is more up to date because it can be integrated directly into a python project thus makes the integration seamless.

Another software technology that we talked about was Python. Python is an interpreted, high-level, and general-purpose programming language. The use of it was selected for this project because of the ease of use and generally good support and developments in AI and machine learning which we plan to use for gesture recognition and interpretation.

# Methodology

The application described in this paper is based on interacting with molecular graphics programs using head and hand gestures. The app is separated into an earlier version that uses UCSF Chimera which has a workaround in order to run and a later version that uses PyMOL with both hand and head gestures.

In order to work more interconnectedly, we used the GitHub code versioning system. The link to our project is here [24].

* 1. Useful libraries

**PyKinect2** library enables writing Kinect applications, games, and experiences using Python. It is inspired by the original PyKinect project on CodePlex. Only color, depth, body, and body index frames are supported in this version. PyKinectBodyGame is a sample game. It demonstrates how to use Kinect color and body frames. We followed tutorials presented in [22] to get familiar with this tool and implement body recognition.

**Mediapipe** framework offers customizable ML solutions for live and streaming media. It contains pretrained models for hand tracking, object detection and tracking, helping us building the hand tracking and interpretation modules.

**PyQt5 and Pygame** framework helps us to create a friendly interface in which users can interact with Kinect and Hand Module, see its body/hand and how our application recognizes different patterns of it.

* 1. Prerequisites

The easiest way to get most of the prerequisites is to use Anaconda which includes NumPy. You will then need to pip install comtypes. The PyKinectBodyGame sample requires PyGame, which needs to be manually installed.

1. Download [Anaconda and get](https://store.continuum.io/cshop/anaconda/) the 32-bit version.
2. Install the [Kinect for Windows SDK v2](http://aka.ms/k4wv2sdk)

Full List of Dependencies

1. Python 3.7
2. mediapipe
3. PyQt5
4. [Kinect for Windows SDK v2](http://aka.ms/k4wv2sdk)
5. [Kinect v2 sensor and adapter](http://aka.ms/k4wv2purchase) Note: you can use a Kinect for Xbox One as long as you also have the Kinect Adapter for Windows
6. [PyGame](http://www.pygame.org/) - for running PyKinectBodyGame sample

## UCSF Chimera Application

### Body recognition and head position tracking workflow

First, the workflow consists of body recognition and live head position tracking using human skeleton estimation, the technique described in section 3.2.1.

Class **BodyGameRuntime** is responsible for skeleton tracking. The main attributes are **self.kinect** that represents the Kinect runtime object in which we only select color and body frames. We also declare **self.bodies**, **self.joint, self.tblJoins** in which we will store skeleton data, joints data and the coordinates of each joint, respectively. We have also declared some attributes used for the Pygame window and frames.

**run** method is the main program loop where we have implemented the whole logic. Using predefined methods from pykinect2 library like **has\_new\_color\_frame** and **get\_last\_color\_frame**, we fill out the back buffer surface with frame's data using the **draw\_color\_frame** method. Now that we have a body frame, we can get the skeleton and save it in **self.\_bodies** attribute using **get\_last\_body\_frame** . After that, we iterate through the **self.\_bodies** frames indicated by Kinect sensors and convert joint coordinates to color space and use the internally implemented method **draw\_body** to draw the torso, left arm, right arm, right leg and left leg.

\*All the skeleton joints were taken into consideration in order to create a general application that can be modified or reused to track other gestures as well.

We get all joints using the specific attribute that corresponds to each body in **self.\_bodies** frames and then we create the **self.tblJoin** list that consists of (x,z,y) coordinates for each joint. The head joint is the third one in **self.joint** list. **self.joint**[i].Position.x, **self.joint**[i].Position.y and **self.joint**[i].Position.z tuple is the input data for our next workflow and it will be saved in head\_position.csv file which will be read by the next workflow continuously as it receives data from the current workflow.

### Chimera script workflow

Because of the nature of writing and reading data, it is advisable that we first run, for a couple of seconds the script that does the body recognition and head tracking because we need some initial data for the Chimera's script. This method is sure to introduce some input lag, but at the same time, it enables us to run the scripts in parallel for longer.

In order to move the actual molecule using Chimera, we created a specific script that continuously reads data from the previous workflow that contains the x,y,z coordinates of the head. These coordinates are absolute, but the molecule needs to move relative to the previous position, so we create another variable that will store it. A feature Chimera library has is that we can run Chimera commands directly into Python scripts using **runCommand** method. We used a standard defined molecule ‘ribbon’ and in order to move it, we use “move {x}, {y}, {z}; wait {frame}”. x, y, z, as we said before, need to be relative to the previous position, so we use the following formula: absolute\_x \* scaled\_cst - previous\_x\_position. scaled\_cst represents a constant applied to our coordinates in order to scale it to Chimera's resolution.

A more general schema of how these workflows join together is presented below.

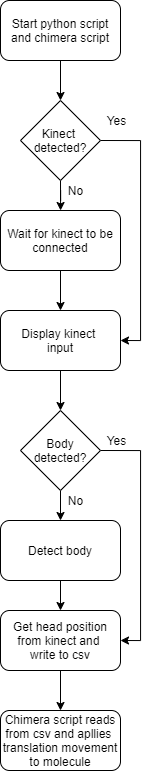


Figure 7. The logical flow of the application

### Results

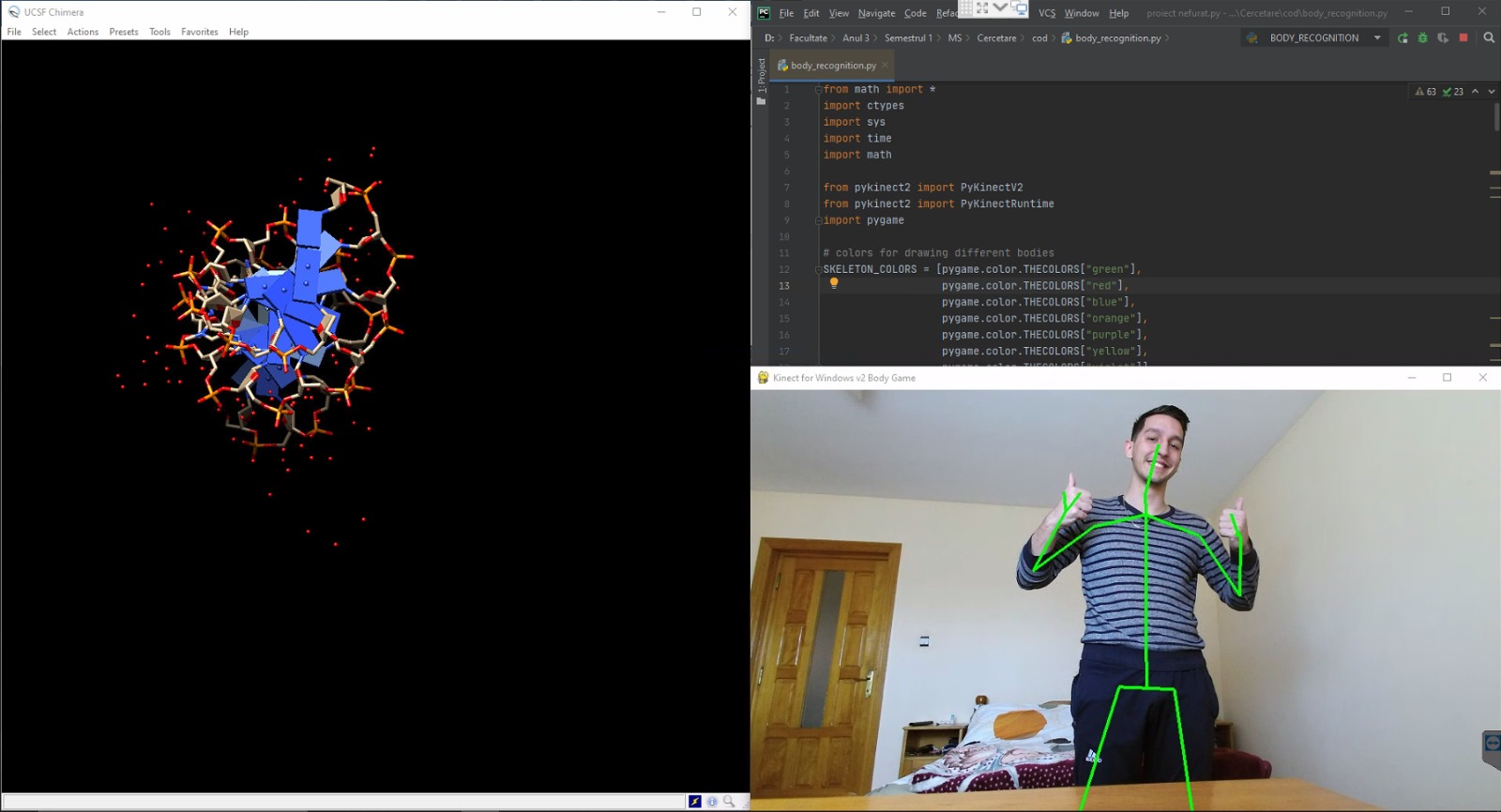


Figure 8. Kinect view along with the visual representation in Chimera

Running the first script opens up a new window that shows the camera feed from the Kinect and if a body is detected it overlays on top of it the calculation and representation for the skeleton. After saving the data taken from the Head joint and running the Chimera script, we will have a new window opened by Chimera where it loads a standard molecule, '1d86', this can be easily changed in the script with another, and the runCommand feature of the Chimera allows us to change the view or other characteristics before beginning the motion. Running the two side-by-side shows how the molecule moves according to the position of the head, is not in because of the method we used to give it the head data, but it offers a relevant perspective on what is possible to do with this technology. At times the molecule looks like it's moving too fast or too slow, that can be because of our scale factor that we used to translate the coordinates received from the Kinect for them to provide a good enough motion for Chimera.

A benefit that this implementation introduces is that all Chimera's features are still available through the graphical user interface during runtime.

Comparing the result of our system to the ones in [9], we can see that ours is not in a finalized state as the other and need further improvements for it to achieve the same fluid motion and to incorporate all the features into one application. On the other hand, the fact that we use full-body tracking brings a lot of potential in the future, because of the ability to implement a system that identifies and interprets hand gestures to perform specific actions and further improve the functionality that it can offers.

### Conclusions for this application

There are some limitations in this approach that are worth mentioning. First of all, is the fact that Chimera has no longer development support, and we were not capable of installing extra Python libraries like pykinect2 or pygame in order to merge those 2 workflows together. This problem also has consequences in the next limitation regarding how these two workflows interact. The fact that the first workflow is writing the information in a CSV file and the second one is reading the information written in that file affects the synchronization between user and molecule movements because of the fact that writing in a file is much slower than reading. The last limitation is about the coordinates given in Chimera. An improvement that we consider is adjusting or changing the way we translate the coordinates from Kinect in order to provide a smoother and more responsive motion. These limitations re addressed in an updated version of this application in which we also offer the possibility to interact with molecular graphics programs using hand gestures.

## PyMol Application

### Graphical User Interface

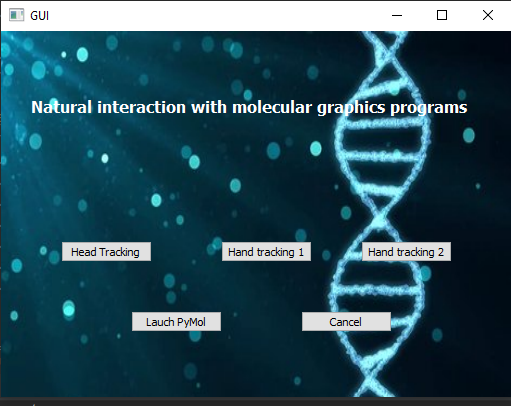


Figure 9. GUI

We create this GUI using QT5 and QT Creator for Python[25], because we implemented three separate ways to interact with PyMol, which will be explained later in the paper. The main thing to point out about the implementation of this is that we used multithreading for each module, implementing a Producer Consumer as follows:

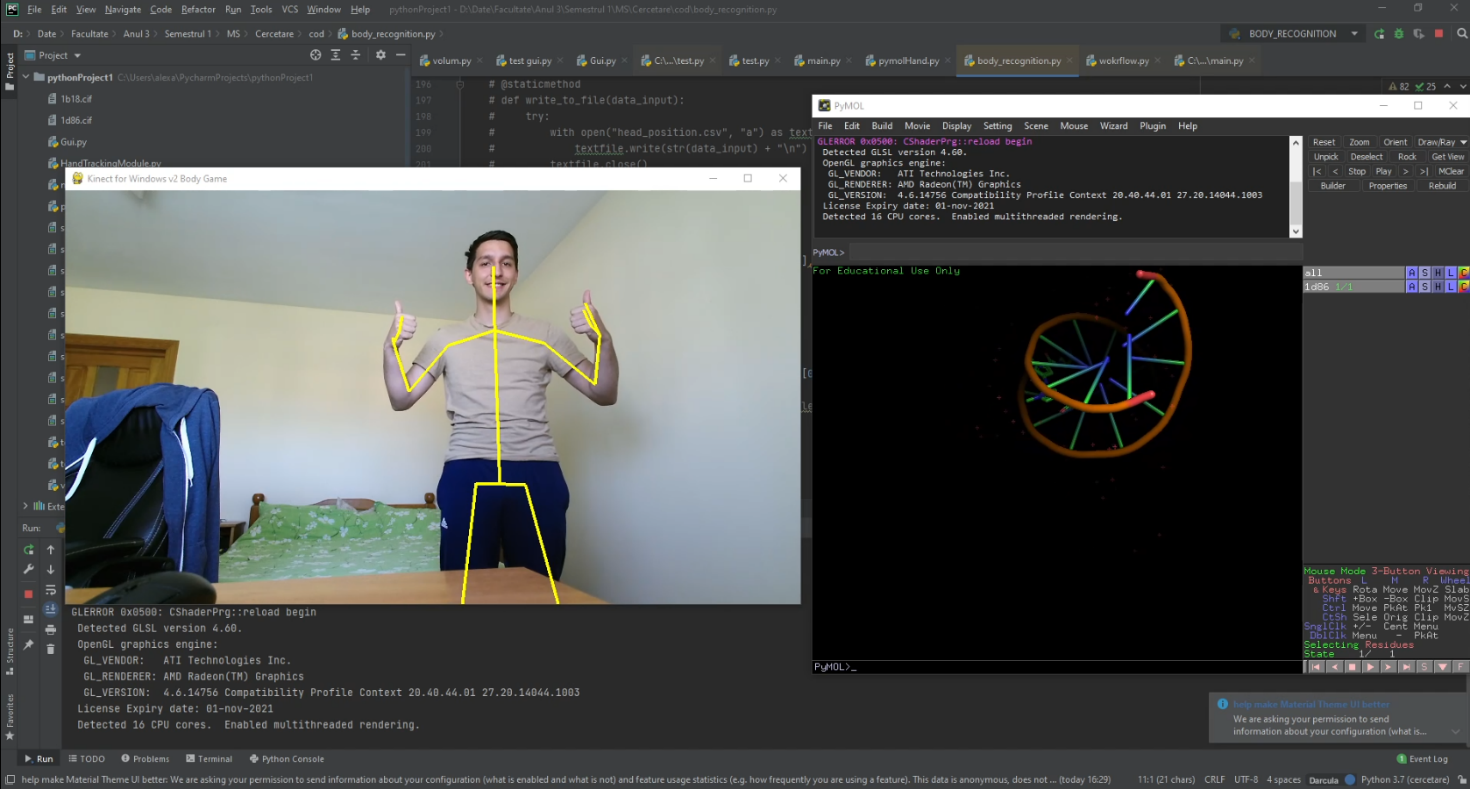
* One Consumer Thread that opens the PyMol and waits to receive different commands that it can feed to the molecule.
* Three Producer Threads, one for each tracking module, that give to the consumer specific commands

Before continuing any further we should mention that the first thing we should do is to launch Pymol and then continue on with the tracking modules.

### Head Tracking

In this first module we used exactly the same technique to track the head like in 4.3.1, adding rotation for the molecule in accordance with the Head Yawn Angle.

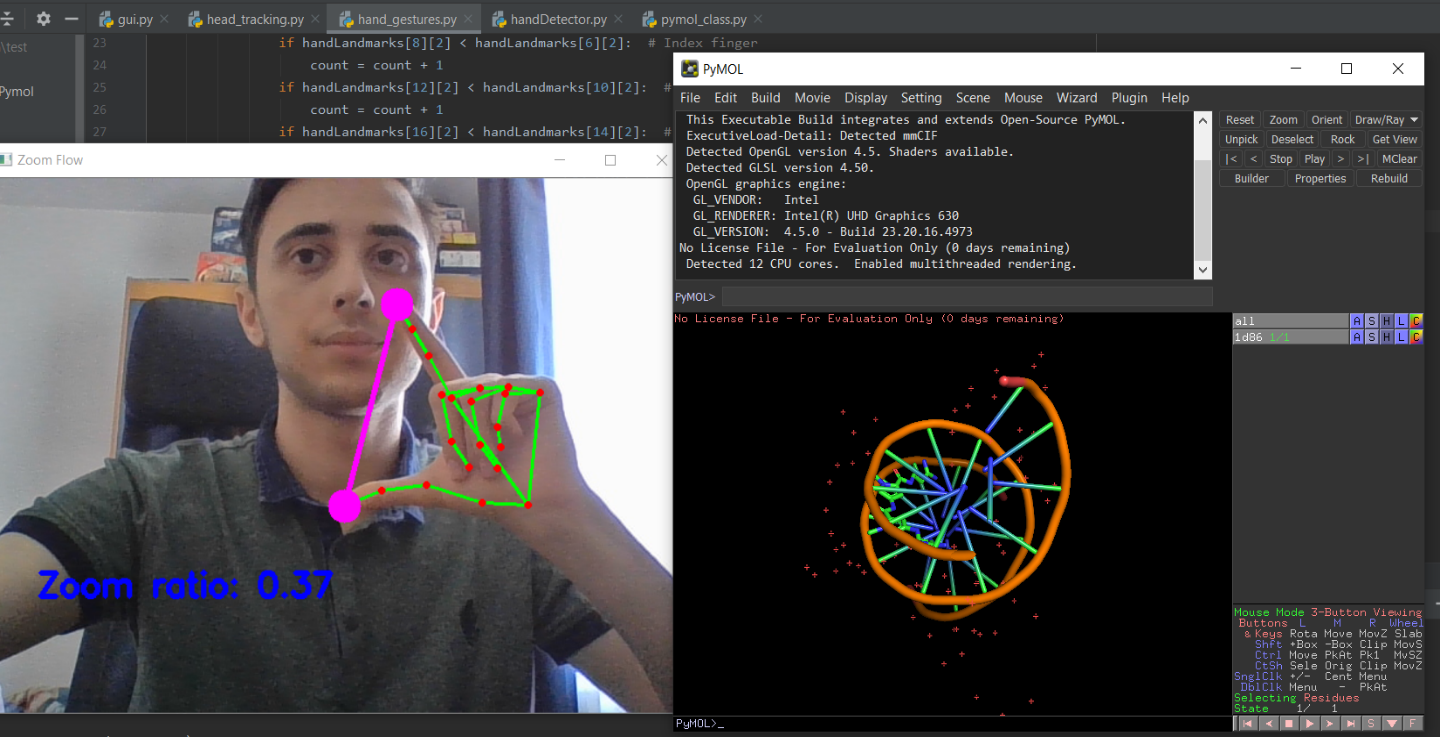
But, in comparison with the first implementation, Pymol did not suffer from the same problems as Chimera, we could send the coordinates from Kinect directly, no longer needing the workaround with writing to and reading from a CSV file.



## Hand Tracking (Zoom)

In this module, we use hand gestures like Zoom IN/OUT or the number of fingers raised and give them as input for molecule movement.

Through the distance between the thumb and the index finger, we manage to transpose that distance in Zoom IN/OUT commands.

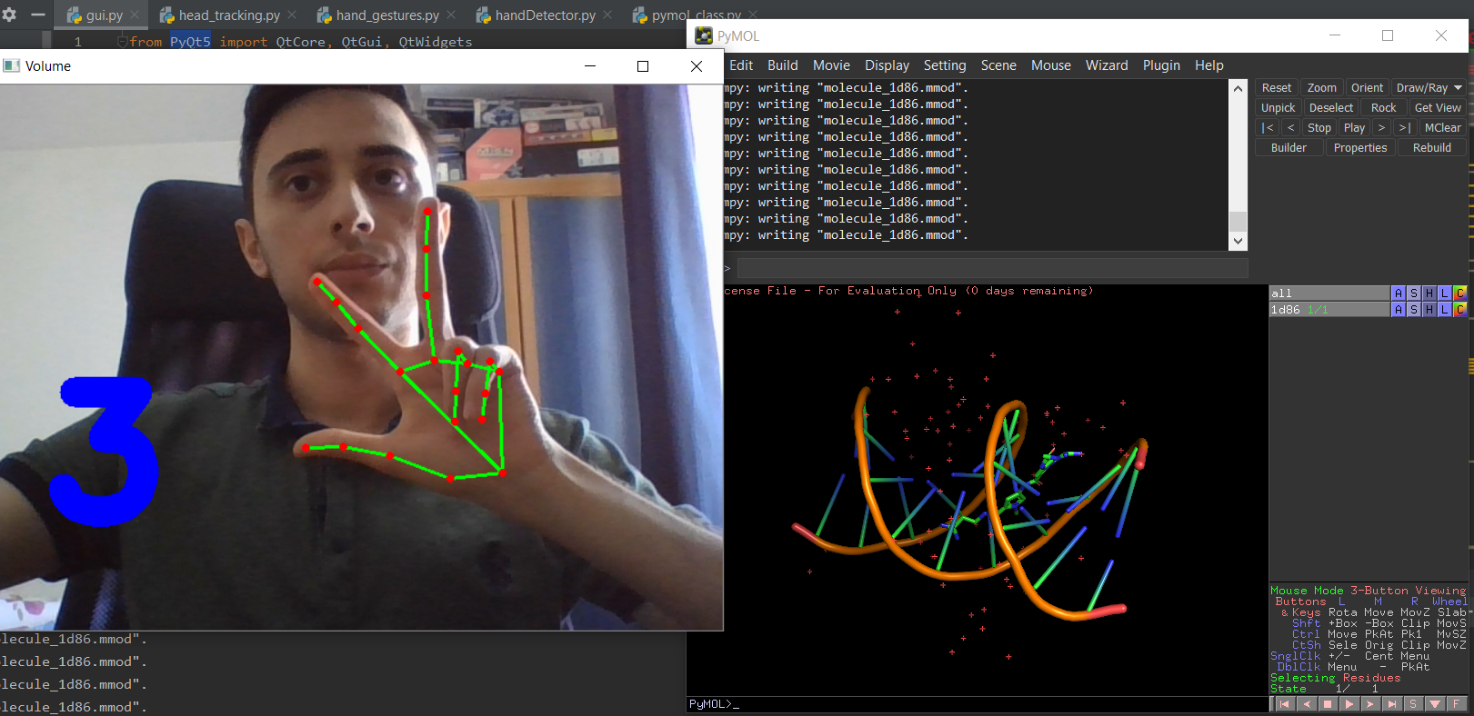


## Hand Tracking (Fingers Count)

In this module, we use the number of fingers raised. A legend to interpret this number is presented below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of fingers raised** | 1 | 2 | 3 | 4 | 5 |
| **Command** | Save the molecule state | Rotate right as long as 2 fingers are raised | Rotate left as long as 3 fingers are raised | Rotate up as long as 4 fingers are raised | Rotate down as long as 4 fingers are raised |

Based on this legend, we give the input to the PyMOL module and move the molecule consequently



### Application Flow

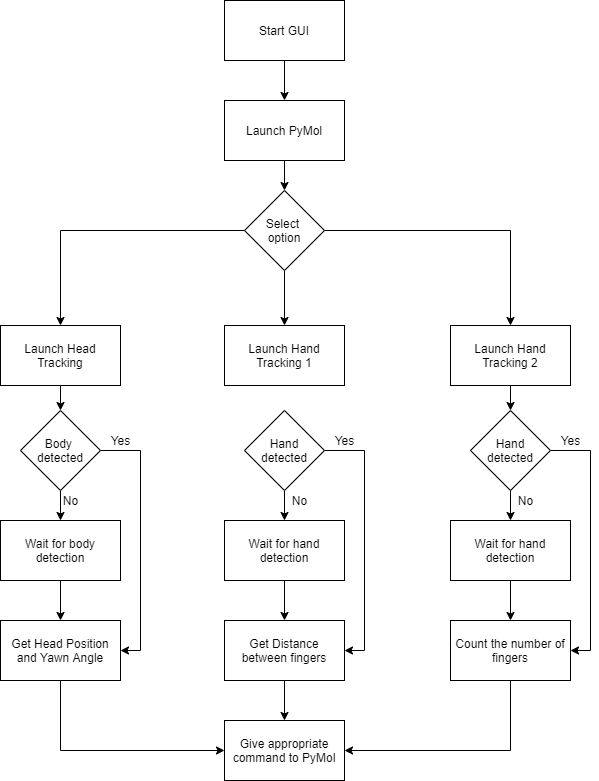


Figure 10. Logical Flow

# 5. Conclusions and Further Developments

This paper concludes by formulating some directions for further software developments, including a discussion of the limitations of the current version.

The main goal achieved within this project is the fact that human-computer interaction can be realized through gesture recognition. More precisely, using head and hand movements, we can control how a chosen molecule is moving with a software application that is used for three-dimensional manipulation and analysis of biological macromolecules, Pymol. Results were promising; most of these molecular programs have a bridge to connect to programming languages like Python. For further development, we want to create a framework fully capable of distinguishing not only head and hand gestures, but also complex gestures and manipulating molecules within these gestures decoded.

As a long time goal, as we said before, we plan to implement a more robust solution for using a wider variety of gestures to interact with the molecular graphics programs that bring real functionality and ease-of-use for those programs and ultimately exceed the level of a gimmick that which it's currently on.

# 6. References

[1] R. A. Bolt, “‘Put-that-there’: Voice and gesture at the graphics interface,” in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1980*, 1980, pp. 262–270, doi: 10.1145/800250.807503.

[2] A. Honegger, “Introduction to PyMOL,” no. February, 2017, doi: 10.13140/RG.2.2.19625.80483.

[3] W. Humphrey, A. Dalke, and K. Schulten, “VMD: Visual molecular dynamics,” *J. Mol. Graph.*, vol. 14, pp. 27,33-38, Mar. 1996, doi: 10.1016/0263-7855(96)00018-5.

[4] M. T. Nelson *et al.*, “NAMD: A parallel, object-oriented molecular dynamics program,” *Int. J. High Perform. Comput. Appl.*, vol. 10, no. 4, pp. 251–268, 1996, doi: 10.1177/109434209601000401.

[5] R. M. Hanson, “Jmol SMILES and Jmol SMARTS: Specifications and applications,” *J. Cheminform.*, vol. 8, no. 1, pp. 1–21, 2016, doi: 10.1186/s13321-016-0160-4.

[6] Z. Fan, W. Chen, V. Vierimaa, and A. Harju, “Efficient molecular dynamics simulations with many-body potentials on graphics processing units,” *Comput. Phys. Commun.*, vol. 218, no. September, pp. 10–16, 2017, doi: 10.1016/j.cpc.2017.05.003.

[7] M. F. Urun, H. Aksoy, and R. Comez, “Supporting foreign language vocabulary learning through kinect-based gaming,” *Int. J. Game-Based Learn.*, vol. 7, no. 1, pp. 20–35, 2017, doi: 10.4018/IJGBL.2017010102.

[8] Y. Yang *et al.*, “Ultrasonic robotic system for noncontact small object manipulation based on Kinect gesture control,” *Int. J. Adv. Robot. Syst.*, vol. 14, no. 6, pp. 1–7, 2017, doi: 10.1177/1729881417738739.

[9] C. Buiu and S. Avram, “Interactive Graphical Visualization of Biomolecules Using Real-Time Head Tracking. Technical Implementation and the Assessment of the Pedagogical Impact,” *INTED2020 Proc.*, vol. 1, no. March, pp. 730–738, 2020, doi: 10.21125/inted.2020.0280.

[10] S. Gibet, T. Lebourque, and P. F. Marteau, “High-level specification and animation of communicative gestures,” *J. Vis. Lang. Comput.*, vol. 12, no. 6, pp. 657–687, 2001, doi: 10.1006/jvlc.2001.0202.

[11] T. Vuletic, A. Duffy, L. Hay, C. McTeague, G. Campbell, and M. Grealy, “Systematic literature review of hand gestures used in human computer interaction interfaces,” *Int. J. Hum. Comput. Stud.*, vol. 129, no. September 2018, pp. 74–94, 2019, doi: 10.1016/j.ijhcs.2019.03.011.

[12] J. Cassell, “A FRAMEWORK FOR GESTURE GENERATION AND INTERPRETATION.” https://www.media.mit.edu/gnl/publications/gesture\_workshop/gesture.wkshop.html (accessed Jan. 17, 2021).

[13] P. Premaratne, “Historical Development of Hand Gesture Recognition,” 2014, pp. 5–29.

[14] T. Allevard, E. Benoit, and L. Foulloy, “Hand posture recognition with the fuzzy glove,” *Mod. Inf. Process.*, pp. 417–427, 2006, doi: 10.1016/B978-044452075-3/50035-2.

[15] H. Kim, G. Albuquerque, S. Havemann, and D. W. Fellner, “Tangible 3D: Hand gesture interaction for immersive 3D modeling,” in *9th International Workshop on Immersive Projection Technology - 11th Eurographics Symposium on Virtual Environments, IPT/EGVE 2005*, 2005, pp. 191–199, doi: 10.2312/EGVE/IPT\_EGVE2005/191-199.

[16] X. Li, M.-K. Tang, C.-Y. Wong, W.-M. Pang, A. Kong, and J. Tang, “A Gesture Assisted Online News Reader for the Visually Impaired,” *Int. Conf. Innov. Connect. World Smart Living*, no. October, 2016.

[17] Y. YAN, X. YI, C. YU, and Y. SHI, “Gesture-based target acquisition in virtual and augmented reality,” *Virtual Real. Intell. Hardw.*, vol. 1, no. 3, pp. 276–289, Jun. 2019, doi: 10.3724/sp.j.2096-5796.2019.0007.

[18] B. Rommel, “BMW gesture control: How it works? How to use it? | BimmerTech,” Oct. 19, 2020. https://www.bimmer-tech.net/blog/item/124-bmw-gesture-control (accessed Jan. 17, 2021).

[19] R. Lun and W. Zhao, *A Survey of Applications and Human Motion Recognition with Microsoft Kinect*, vol. 29, no. 5. 2015.

[20] O. Kreylos, “The Kinect 2.0 | Doc-Ok.org,” May 2013. http://doc-ok.org/?p=584 (accessed Jan. 17, 2021).

[21] C. Demerjian, “A long look at Microsoft’s XBox One Kinect sensor - SemiAccurate,” 2013. https://semiaccurate.com/2013/10/15/long-look-microsofts-xbox-one-kinect-sensor/ (accessed Jan. 17, 2021).

[22] N. Kolakowski, “Microsoft Acquires Canesta, 3D Tech Patents,” Nov. 2010. https://www.eweek.com/news/microsoft-acquires-canesta-3d-tech-patents (accessed Jan. 17, 2021).

[23] E. Pettersen *et al.*, “UCSF Chimera – A Visualization System for Exploratory Research and Analysis,” *J. Comput. Chem.*, vol. 25, pp. 1605–1612, Oct. 2004, doi: 10.1002/jcc.20084.

[24] I. A. T. A. Capota, “Interaction with molecular graphics programs using Kinect,” Jan. 2021. https://github.com/capotandrei/interaction-with-molecular-graphics-programs-using-Kinect (accessed Jan. 18, 2021).

[25] “Qt | Cross-platform software development for embedded & desktop.” https://www.qt.io/ (accessed May 11, 2021).