INTERACTIVE 3D USER INTERFACE FOR SENSOR PLACEMENT ON ON-BODY LOCATIONS

Sukriti Bhardwaj

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science
College of Computing

Georgia Institute of Technology

Approved by:

Dr. Thomas Ploetz
College of Computing
Georgia Institute of Technology

Dr. Gregory Abowd
School of Engineering
Northeastern University

Mentored by:

Dr. Hyeokhyen Kwon
Department of Biomedical Informatics
Emory University

May 7, 2022

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

IMUTube generates on-body sensor-based human activity recognition data by leveraging the abundance of videos on platforms like YouTube as a dataset for training human activity recognition models. This thesis deals with the development of a user interface for IMUTube which will allow a user, such as a computer vision researcher or healthcare workers, to generate human pose detection data automatically using IMUTube by inputting custom data while hiding the nuances of the state-of-the-art computer-vision and signal-processing based back-end system. Thus, the user interface creates an interactive web application that makes IMUTube accessible to users from different backgrounds and requires no coding experience.

CHAPTER 1

INTRODUCTION

On-body sensor-based human activity recognition is used for labelling actions such as running, walking and dancing using data collected from sensors. This is a growing field in computer science, especially in computer vision, with a significant variety of applications in fields such as security, surveillance, and healthcare [1, 2, 3, 4]. One of the significant hurdles in the development human activity recognition algorithms is the lack of a large sensor-based labeled dataset. IMUTube [5] is an attempt at tackling this issue and this project deals with the user interface and 3D-modeling front-end of the application.

IMUTube solves this problem by using YouTube as a dataset where users can select videos and set parameters to extract human movement data in a video. YouTube is used as a dataset since it is has many videos demonstrating different actions and activities. The videos selected from YouTube, along with a 3D human model with functionality to place virtual sensors will be used to generate a large pose detection dataset. We are developing a front-end for the application with the goal of being easy to use, intuitive and familiar. We considered making a user-interface over a text-based interface as research has shown the advantages of using a GUI. Using a GUI, experts required 51 percent lesser time to complete tasks compared to using a text-based interface [6]. According to research done by Davis and Bostrom, novice users learned and performed tasks faster using a GUI [7].So, the user interface will make IMUTube accessible to people from different backgrounds, for purposes such as education and healthcare, including those with no coding experience.

The goal of the 3D interface is to create a simple and easy way for the users to attach sensors on the human model. A 3D interface design was chosen as it replicates the process of attaching sensors to humans very closely. From a development perspective, some factors we had to take into account included easy scalability of the user interface as well as ease of integration with the front-end and backend of the application.

CHAPTER 2

LITERATURE REVIEW

3D models in a web-interface are a relatively recent development driven by the growth of advancements in web render engines like WebKit and development of libraries like Three.js. Adding a 3D model to a user-interface adds another dimension to the user interface and with this extra dimension comes the ability to have multidimensional motion, more than just scrolling, but this brings a lot of challenges. Some of these challenges include varying capabilities of various browsers to display 3D models and poor display resolutions, and some of these challenges can be alleviated by using a cross-platform library. Some of the most popular techniques are Three.js and Unity3D[8].

Three.js was initially developed for ActionScript and uses WebGL, Canvas and SVG as some of the rendering engines. It is arguably the most popular 3D modeling library for the web which enables high-level browser based 3D scenes. Unity3D is a popular game rendering engine that is widely used to create web-based games. It is very easy-to-use and compatible with most modern web browsers. These two frameworks differ in two important criteria: required plug-in and customizable pipelining [8].Unity3D requires a plug-in to run in the web-browser and unlike Three.js it lacks a customizable pipeline. A customizable pipeline allows the programmer to have low-level control over the render targets and order, materials and associated shaders. Since the front end of our application is made using React.js, another factor to take into consideration is the integration of the 3D user interface with the React application. Unity3D lacks the ability to directly integrate with the React.js based front-end and using this framework would require the front-end to be developed as a game. Three.js is more powerful and more suitable for our project as it is developed in JavaScript and can be directly integrated into a React.js application using the react-three-fiber, which is a React renderer for Three.js.

Three.js and Unity3D act as a layer between the user and the rendering engines that actually displays the 3D polygon mesh on the website. A polygon mesh is a collection of edges, vertices and the surface that make an object [9].Rendering engines render this mesh when the browser loads a website. Some of the more popular rendering engines are WebGL, X3D and SVG. SVG is usually used more for 2D rendering so we focused on WebGL and X3D for this project. X3D is a standard for directive representing 3D graphics that uses OpenGL for web-based rendering [10]. WebGL uses OpenGL Embedded Systems shading library [11] and in essence is a modern "derivative" of OpenGL and is highly popular with modern web browsers, and because of this we are using this rendering engine for our project over X3D.

The development part of this project focuses on cross-platform compatibility and popularity of a framework in-order to make the code easier to use with the rest of the application, and we decided to pick Three.js with a WebGL rendering engine to reach the most users and to integrate better with the rest of the front-end of IMUTube. We plan on using the latest iterations of the frameworks described above in order to make our user-interface more fluid, and future-proof.

CHAPTER 3

METHODOLOGY

The front end of IMUtube application was developed in React.js which is an open source JavaScript framework created by Facebook used for developing front end applications[12]. The 3D model and the sensors was created in Three.js which is also a JavaScript library used for developing computer graphics for the internet. The 3D user interface consists of a 3D human model and functionality to move the camera, add sensors, and remove sensors. This 3D human model is a `glb` file, which is a 3D computer modeling file format which supports motion and animation in web applications, which is imported from Three.js examples. A skeleton helper, a type of skinned mesh which consists of bones, is used for visualizing a skeleton and added to our 3D model for testing.

The 3D human model is rotatable and zoomable using the Orbit Controls library which rotates the viewport camera. The sensors are added and removed on click. This process makes use of raycasting. Raycasting is casting rays in different directions from a particular point, in our case the mouse clicks, and then storing all the objects the ray intersects with. In our case, when removing the sensor, when the ray intersects with our sensor, we remove the sensor. Similarly, when the ray casted on clicking a point intersects the human model, we add a new sensor object to the location. To increase the accuracy of our raycasting, we increased density of triangles in the triangular mesh of the 3D model. The 3D model will be displayed using a web graphics API called WebGL [13].The user has the option to rotate the sensors along the x and y axis and this is done by changing the rotation vector of the cubes along the x and y axis by inputting the rotation of the cube in the user interface created using `dat.GUI`. `dat.GUI` is a tool used for interaction features with the 3D objects.

This 3D interface has been connected with the backend of the application where the position of the sensors will be used to calculate the pose of the human model. The position of sensors is calculated by finding the euclidean distance between world coordinates of the bones and the world coordinates of the sensors. Then the closest bone is chosen to describe the location of the sensor and this is communicated to the machine learning based backend system.

CHAPTER 4

RESULTS AND DISCUSSION

Now, let's consider the various components of the front end of the application: a home page, a search parameters page, an activity parameter page, a sensor placement page, a video selection page and an output page. The user starts at the home page where they can search for various activities such as singing, dancing, and typing as shown in fig. 4.1.

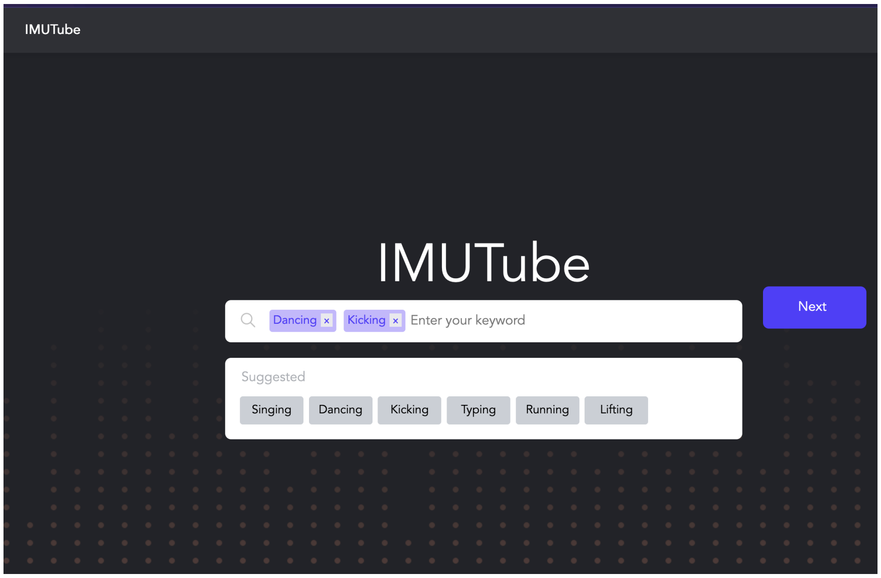
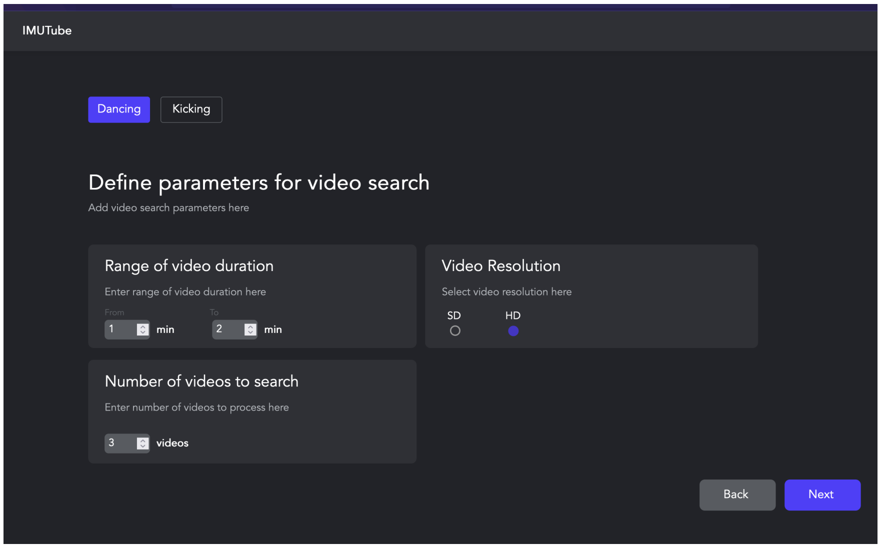


Figure 4.1: This is the home page of the IMUTube interface. Here, the user can search for activities to generate IMU data.

Then, the user is redirected to the search parameters page where they can set parameters for YouTube video selection such as number of videos, range of video duration and video resolution as shown in fig. 4.2. They can set the same or different search parameters for each activity selected.



Define parameters for video search

Add video search parameters here

Range of video duration

Enter range of video duration here

From 1 min To 2 min

Video Resolution

Select video resolution here

SD ○ HD ●

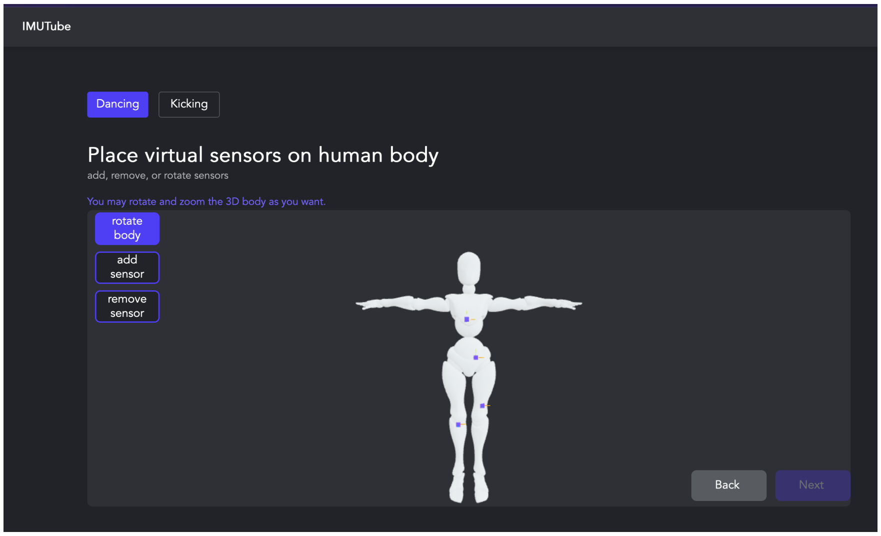
Number of videos to search

Enter number of videos to process here

3 videos

Figure 4.2: This is IMUTube's Activity Parameter page. Here the user can set the video parameters

Next, the user is redirected to the sensor placement page where they can place virtual IMU sensors on the 3D human model. The fig. 4.3 shows the integration of the 3D model with the rest of the user interface. The next stage is the video selection page where the user can select videos associated with the activities, based on parameters entered in the search parameter page, from a set of pre-selected videos from YouTube as shown in fig. 4.4.

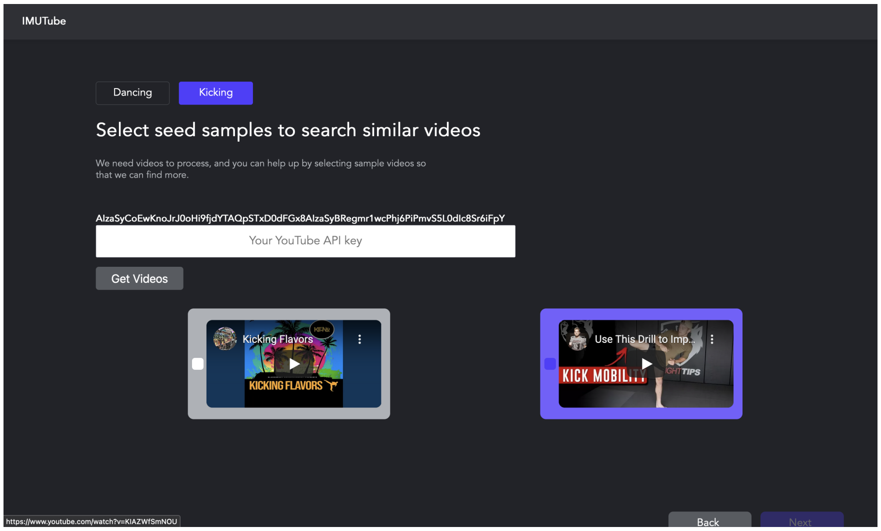


Place virtual sensors on human body

add, remove, or rotate sensors

You may rotate and zoom the 3D body as you want.

Figure 4.3: Sensor Placement. Here, the user can attach sensors to the humanoid model.



Select seed samples to search similar videos

We need videos to process, and you can help up by selecting sample videos so that we can find more.

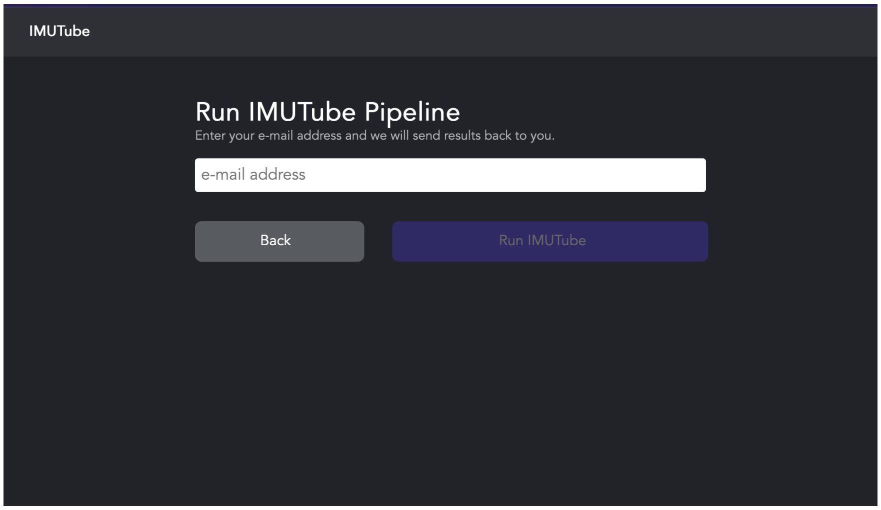
AIzaSyCoEwKnoJrJ0oHi9fjdYTAQpSTxD0dFGx8AIzaSyBRegmr1wcPhj6PiPmvS5L0dIc8Sr6iFpY

Your YouTube API key

https://www.youtube.com/watch?v=KIAZWfSmNOU

Figure 4.4: Video Selection. Here, the user can select videos to be used for generating IMU data.

Then, the backend of IMUTube searches and downloads the selected videos, extracts virtual sensor data by tracking human motion in the video, and then sends the generated data to the email specified in the output page. The output page is shown in fig. 4.5.



Run IMUTube Pipeline

Enter your e-mail address and we will send results back to you.

e-mail address

Figure 4.5: Output page. Here, the users can input their email to receive the resulting data from IMUTube pipeline.

This undergraduate thesis project deals with the 3D human model and relevant interface in the sensor placement page of the user interface for IMUTube. This interface consists of a 3D human model where the user can place various configurations of virtual inertial measurement unit (IMU) sensors. IMU sensors are used to track human motion and pose without the use of cameras making them particularly useful for outdoor activities and recordings[14]. The interface consists of a skeletal outline of human and virtual sensors. The sensors is rotatable, zoomable, and provide the user with an easy and interactive way to place sensors on the desired locations of the skeletal human model by clicking on the location. Shown in fig. 4.3 is the current user interface of the 3D model which is being used for IMUTube with the sensors in purple. This 3D user interface proposes a novel approach that lets the user attach sensors on the model as they would in real-time, in an easy, intuitive way while capturing the essence of the sensor placement process.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Data collection for pose detection is an expensive and time consuming process. Using a 3D human model and virtual sensors makes data collection cost effective. This user interface gives the user the option to attach the sensors to any part of the model just like the user would in real time.

This 3D user interface can be extended to a wide range of applications which use virtual sensors for dataset generation. One of the limitations of our current user interface is that the model does not represent different body types and height ranges. Creating an auto-scaling model might make the process of placement of sensors by the user more effective. Another drawback of our current setup is that the model represents a general human and does not address different age groups or genders. Giving the user options to select different models which represent different genders and age groups would help solve this problem. Currently, the sensors placed on the model do not adjust accordingly to the curving of the model. Another future direction would be to auto-adjust these sensors on the 3D model.

REFERENCES

[1] E. Kim, S. Helal, and D. Cook, “Human activity recognition and pattern discovery,” *IEEE pervasive computing*, vol. 9, no. 1, pp. 48–53, 2009.

[2] Y. Kim and B. Toomajian, “Hand gesture recognition using micro-doppler signatures with convolutional neural network,” *IEEE Access*, vol. 4, pp. 7125–7130, 2016.

[3] C. Torres-Huitzil and A. Alvarez-Landero, “Accelerometer-based human activity recognition in smartphones for healthcare services,” in *Mobile health*, Springer, 2015, pp. 147–169.

[4] W. Lin, M.-T. Sun, R. Poovandran, and Z. Zhang, “Human activity recognition for video surveillance,” in *2008 IEEE international symposium on circuits and systems*, IEEE, 2008, pp. 2737–2740.

[5] H. Kwon *et al.*, “Imutube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 1–29, 2020.

[6] M. Rauterberg, “An empirical comparison of menu-selection (((cui) and desktop (gui) computer programs carried out by beginners and experts,” *Behaviour & Information Technology*, vol. 11, no. 4, pp. 227–236, 1992.

[7] S. Davis and R. Bostrom, “An experimental investigation of the roles of the computer interface and individual characteristics in the learning of computer systems,” *International Journal of Human-Computer Interaction*, vol. 4, no. 2, pp. 143–172, 1992.

[8] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat, “3d graphics on the web: A survey,” *Computers & graphics*, vol. 41, pp. 43–61, 2014.

[9] Wikipedia contributors, *Polygon mesh — Wikipedia, the free encyclopedia*, [Online; accessed 13-April-2022], 2022.

[10] ——, *X3d — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=X3D&oldid=1076757809, [Online; accessed 13-April-2022], 2022.

[11] ——, *Webgl — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=WebGL&oldid=1076428654, [Online; accessed 13-April-2022], 2022.

[12] A. Mardan, *React quickly: painless web apps with React, JSX, Redux, and GraphQL.* Simon and Schuster, 2017.

[13] P. Vepakomma, D. De, S. K. Das, and S. Bhansali, “A-wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities,” in *2015 IEEE 12th International conference on wearable and implantable body sensor networks (BSN)*, IEEE, 2015, pp. 1–6.

[14] T. Von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, “Sparse inertial poser: Automatic 3d human pose estimation from sparse imus,” in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 349–360.