SUMMER PROJECTS
2019

SMART GAMER

ELECTRONICS CLUB

SCIENCE AND TECHNOLOGY COUNCIL

IIT KANPUR

Final Project Report

July 9, 2019

Index

1. Acknowledgement

We are grateful to the Electronics club and Science & Technology council IIT Kanpur for providing us with the resources and motivation for this summer project. We would like to thank our mentors and co-ordinators of Electronics club:

- Soumya Ranjan Dash
- Jay Mundra
- Mudit Agrawal
- Nitish Deshpande

to guide us throughout the project

2. Introduction

Computer games are a popular consumer electronics item. The game players find it captivating to interact with games via joysticks, buttons, trackballs, or wired gloves. They may find it even more engaging to interact through natural, unencumbered hand or body motions. A computer vision-based user interface could provide these capabilities. Computer games represent a possible mass-market application for computer vision. So in this project we exploit this mass-market application and try to build and interactive game which can be controlled using a particular object and also build an AI which can play the game on its own.

3. Components

Intel Real Sense D435-Depth Sensing Camera

Figure 3.1: RealSense Camera D435i

Figure 3.2: Detailed RealSense D435i

3.1 Overview

The Intel RealSense Depth Camera D400-Series uses stereo vision to calculate depth. D435i version is USB-powered and consist of a pair of depth sensors, an RGB sensor, and an infrared projector. It is ideal for makers and developers to add depth perception capability to prototype development.

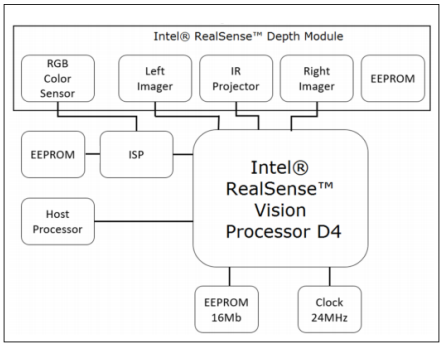

Figure 3.1.1: Working System RealSense D435i

3.2 Highlights

Depth Camera Includes :

1. A powerful vision processor that uses 28 nanometer (nm) process technology and supports up to five MIPI Camera Serial Interface 2 lanes to compute real-time depth images and accelerate output.
2. A powerful vision processor that uses 28 nanometer (nm) process technology and supports up to five MIPI Camera Serial Interface 2 lanes to compute real-time depth images and accelerate output.
3. A set of image sensors that enable capturing of disparity between images up to a 1280 x 720 resolution.
4. Support for the new cross-platform and open source Intel RealSense SDK 2.0.
5. A dedicated color image signal processor for image adjustments and scaling color data.
6. Active infrared projector to illuminate objects to enhance the depth data.

3.3 Advantages

1. With the rolling image shutter and narrow field of view, the Intel RealSense Depth Camera D435i offers high-depth resolution when the object size is small and more precise measurements are required.
2. With the global image shutter and wide field of view (91.2 x 65.5 x 100.6), the Intel RealSense Depth Camera D435i offers accurate depth perception when the object is moving or the device is in motion, and it covers more field of view, minimizing blind spots.

4. Approach

We started off this project by learning how an object detection model works and implementing it on Google Chrome's T-Rex game which many of us have played when our Internet connection goes down. Then we learnt how to use the pygame library of python and built a simple shooting game using this. Then we moved on to implementing game AI using behaviour tree and reinforcement learning which we implemented on the game we built on pygame. The last portion of our project was to play a game using our

gestures. For this we used a depth sensing camera and integrated it with a game (a little more complex) which was different from the one we built.

5 Modules Implemented

5.1 Object Detection

In this we first take an image through our webcam then we will convert our image data from RGB to HSV values. After this we will use thresholding in order to isolate our object (we have chosen a blue colour object to easily distinguish it from the background). Once we have done this we will use multiple rounds of erosion and dilation in order to remove noises. Then once we have removed major noises we draw contours around our filtered image and then find the contour of largest area, because in case any noise would be left after filtering then it will get neglected as the largest area will be that of our object. Once we have detected our object we find its centre of mass and term it as our centre. We will use this for our reference while controlling our key presses our game. Now we set reference line above or above which our key press simulation will work which we implemented simply by using pynput library.

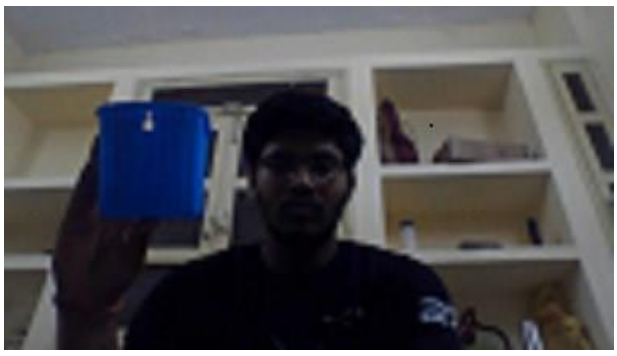

Figure 5.1.1: Original Image

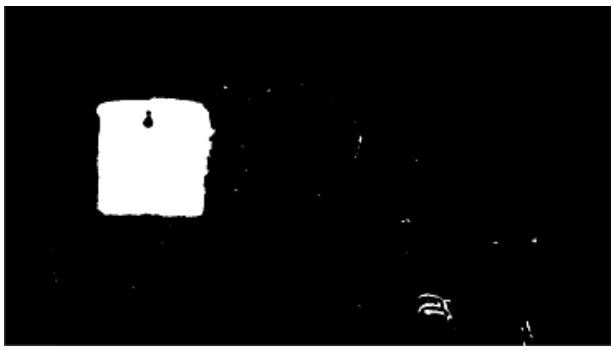

Figure 5.1.2: Threshold Image

Figure 5.1.3: Filtered Image

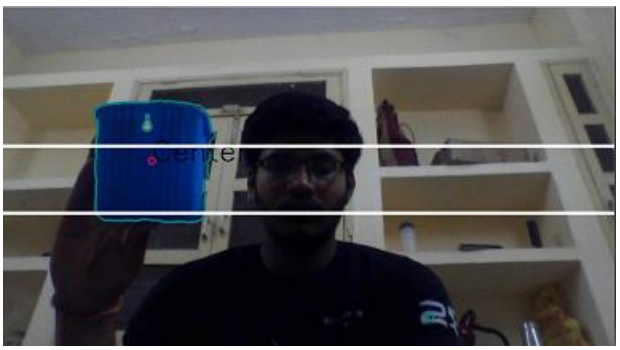

Figure 5.1.4: Final Image

5.2 PyGame

5.2.1 About PyGame

PyGame is a set of python modules, and these modules are specially designed for writing or coding games. Python is great for writing out ideas in one long block of code.

5.2.2 Game Using PyGame

Pygame is a good library to make simple 2-D games like space shooters etc. PyGame with Python OOP was used to make the game. OOP helped us clean up our code in spite of increasing the complexity in the game as an when required.

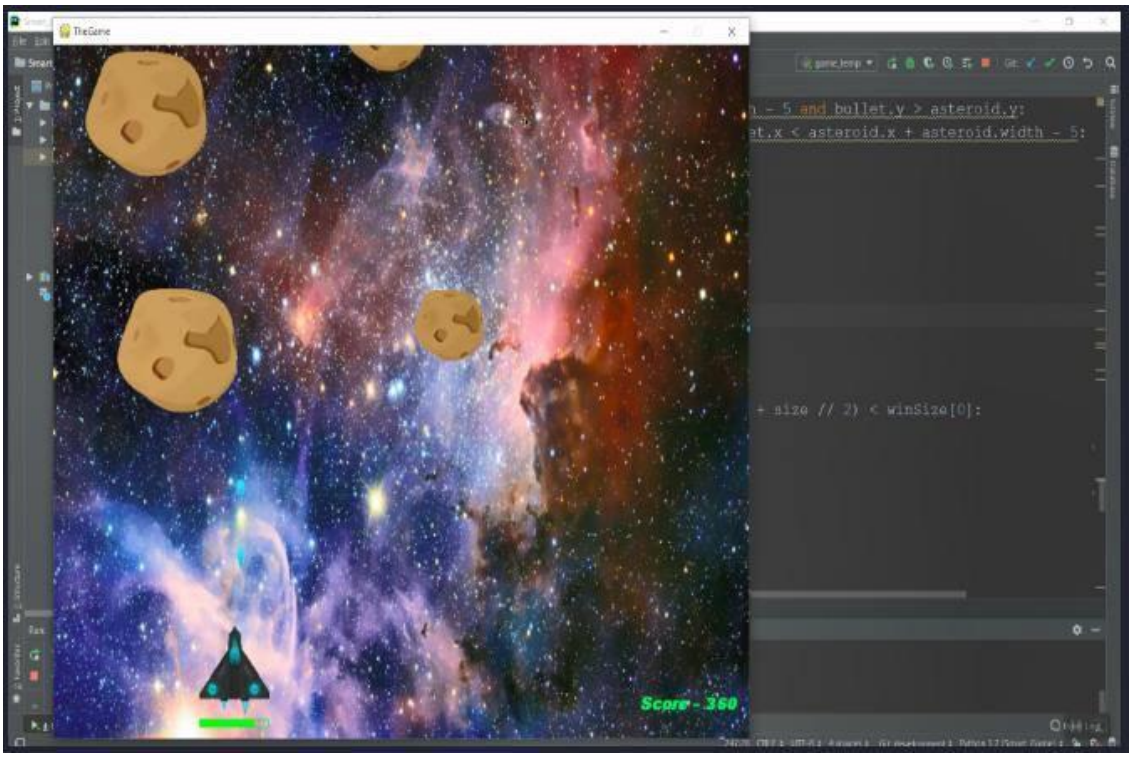

Figure 5.2.2.1: Game developed using pygame

5.2.3 Why PyGame

- One Long Block Of Code
- Global Variables are more obvious
- Pygame comes with less assembly required
- Game interface can be created using all defined variables which can easily be tracked

5.3 Game AI : Behaviour Tree

It was the most famous model to create game AI during last decade. Its popularity was due to the fact that it is easy to visualize, implement and provide decent performance. On the top of the tree there is the root node which is executed on each iteration(unless you want to make it more efficient). Each leaf node is generally a task node - related to some sort of task. All the intermediate nodes are one of selector, sequence, parallel, decorator, etc type which determine how to execute it child nodes. Each node returns either Success, Failure or Running status which decides the next operation. While behaviour tree works very good in simple to intermediate level situations, the increasing complexity of games lead to its continuing decline and other models like utility AI and machine learning based models are gradually taking over.

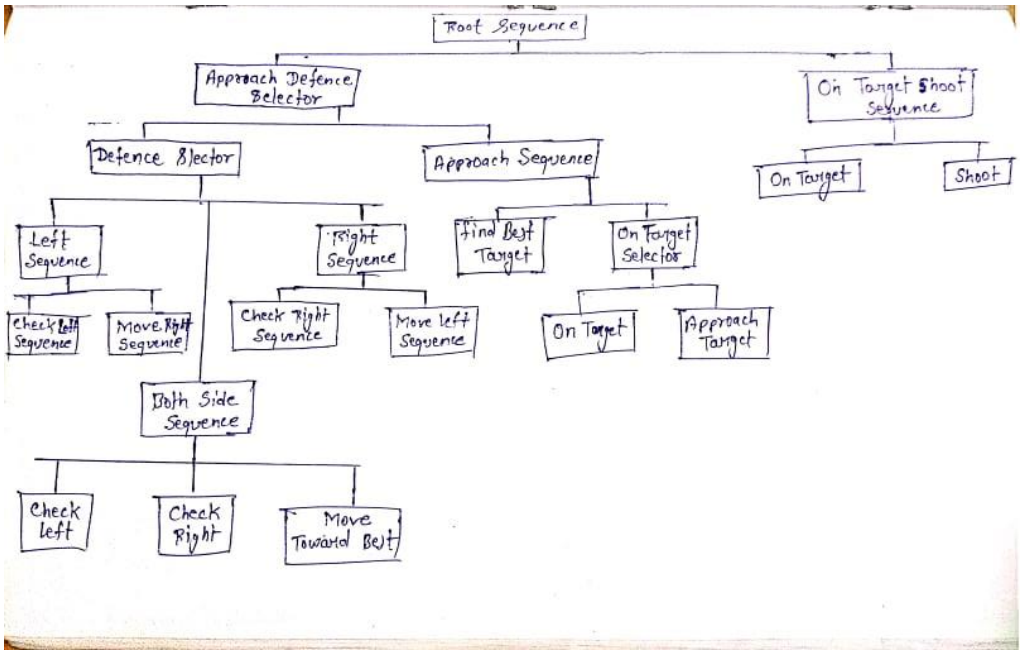

Figure 5.3.1: Behaviour Tree

5.4 Game AI : Reinforcement Learning

Reinforcement learning and games have a long and mutually beneficial common history. From one side, games are rich and challenging domains for testing reinforcement learning algorithms. From the other side, in several games the best computer players use reinforcement learning.

5.4.1 Q-Learning

Objectives

- To modify the model for Q learning from catch the ball game to make it work on the game of space shooters.
- To integrate the model with the game.

Model

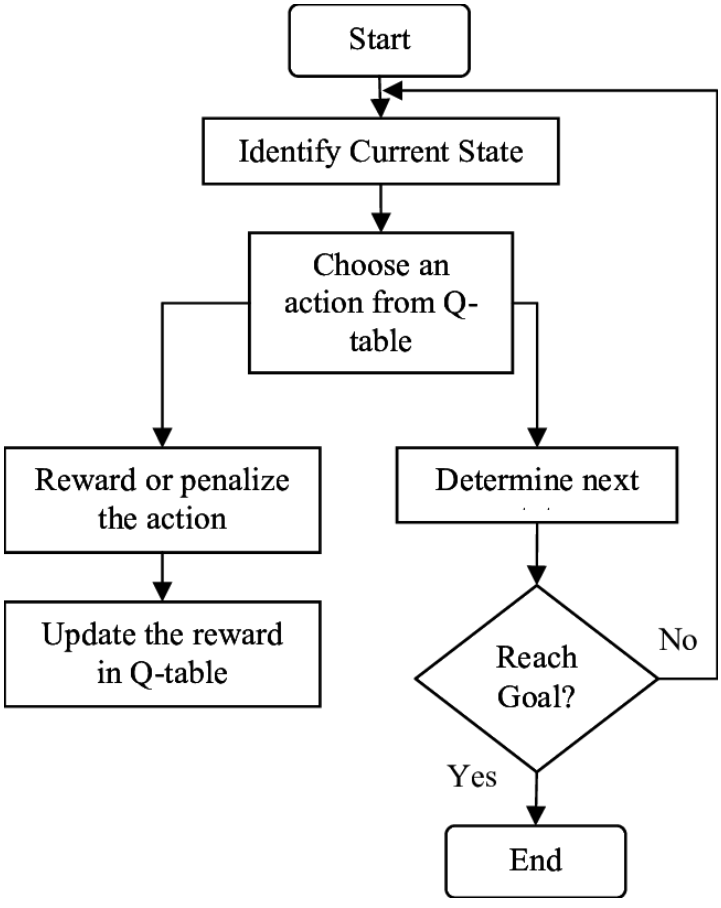

Figure 5.4.1.1: Q-Learning Model

Problems faced

- Multiple states mapped to the same value in the QI dictionary (Solved).
- Focused on a single enemy than the whole bunch (Solved). For the testing phase, the enemy count was reduced to only one per screen.
- Tried to figure out all the possible states of the game (Solved) but training around 11 lakh states for 3 possible actions (stay, left, right) will take around 15 days which is not possible so may apply pooling methods to reduce the number of states to few thousand.

Progress after mid-term evaluation

- **Reduced the number of states:**
  Number of states were reduced by sacrificing the uniqueness of states a little bit. It greatly increased the efficiency of filling the Q table and the spaceship has started moving to the right which it never did before.
- **Updated Reward System:**
  Changed the reward system for it to get hit than to dodge the asteroids (basically reversed it) for the testing phase and added an extra small reward along with the negative reward when the asteroids miss. When the asteroids miss the spaceship, the model will receive a negative reward but also a small positive reward depending upon the distance between the spaceship and the asteroid.

Results

- The model is training but it's not training well. The error could be in 3 places, the reward system, the state based model. Changing the rewards system can be a possibility. Need to define the states more clearly.
- When tested with asteroid falling in only a single place, the spaceship is not moving towards the asteroid but is oscillating in its own place.

5.4.2 Deep Q-Learning

Objectives:

Using neural networks to train an AI which can play ATARI games like Space Invaders.

Problems Faced:

- Training is taking too much time.

- Since training is taking too much time, we can't predict how well the current model is performing.

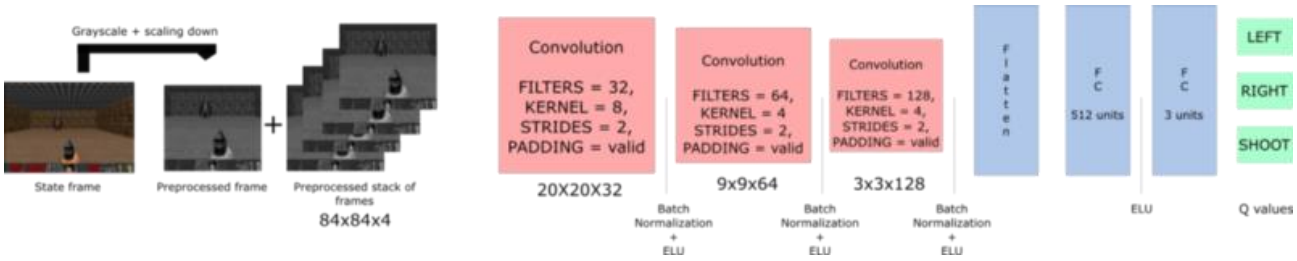

Figure 5.4.2.1: DQN Model

Final Product:

Testing with asteroid falling in only 2 places alternatively so that the spaceship learns to find the asteroid and shoot it. If this happens successfully we will try with asteroids falling in 4 places instead of 2 and if that works too we will continue with this model and the reward system or else will change the reward system.

5.5 Depth Sensing Using RealSense

Playing games using gestures can be real fun. For this purpose we used depth sensing technique with realsense camera and integrated this with a more complex game: ViZDoom.

5.5.1 ViZDoom

ViZDoom allows developing AI bots that play Doom using only the visual information (the screen buffer). ViZDoom is based on ZDoom to provide the game mechanics. We chose this game because we intended to build a game which can be played using our gestures and also develop its AI. But since all this had already been implemented in simple shooting game built using PyGame we chose this game to avoid the part of building a game again.

Figure 5.5.1.1: ViZDoom Game

5.5.2 Integration With RealSense

- Calibration Of RealSense Camera
- Using pyrealsense2 library in the ViZDoom game
- Gestures integrated in game using the object detection model

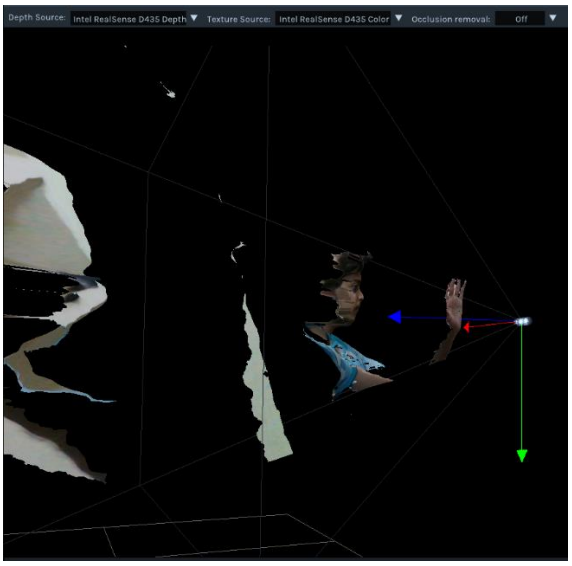

Figure 5.5.2.1: Hand Depth by RealSense

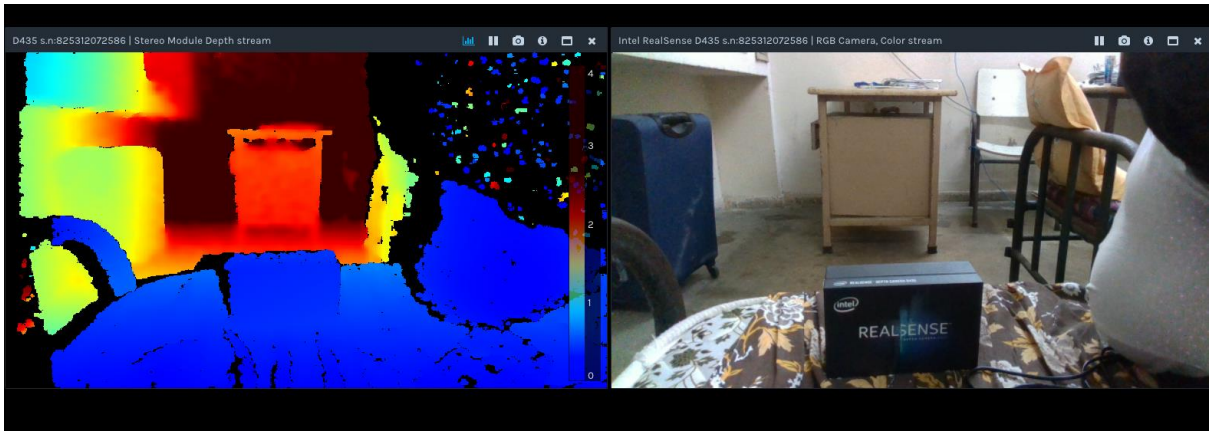

Figure 5.5.2.2: Depth Map by RealSense

5.5.3 Final Game

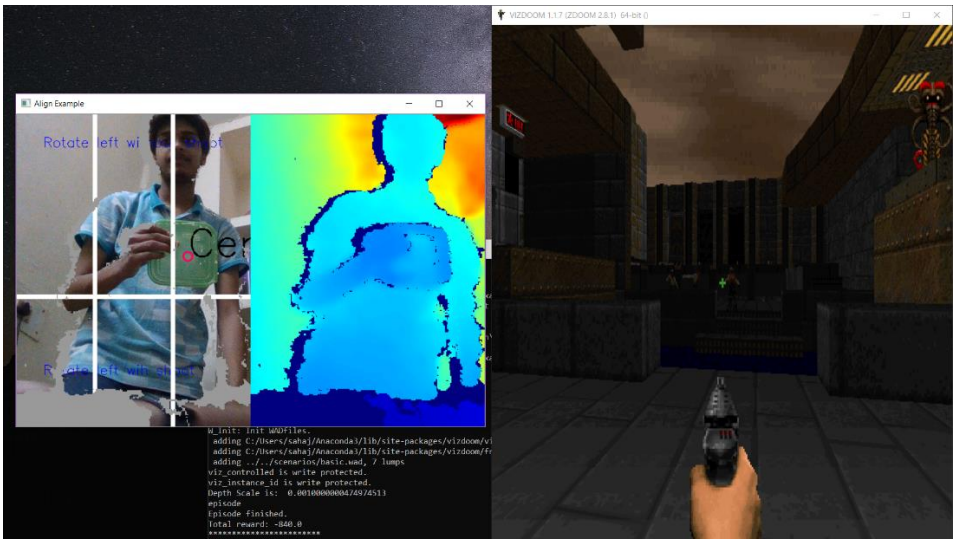

Figure 5.5.3.1: Final Game with RealSense

6 References

- Chrome T-Rex game using computer vision https://medium.com/@sulphurgfx/playing-chromes-dinosaur-game-using-computer-vision-105da2f3114f
- ViZDoom game https://github.com/mwydmuch/ViZDoom
- PyRealSense2 library of Intel RealSense https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python
- PyGame Tutorial https://youtu.be/i6xMBig-pP4

7 The Team

- Sahaj Agrawal
- Moksh Shukla
- Ritik Saxena
- Prashant Singh
- Priyanshu Agarwal
- Yuvraj Gagneja
- Sameer Mahadule
- Sparsh Agarwal
- Pushpesh Kumar