

**MINOR PROJECT: FINAL REPORT**

TITLE: CRONUS 1.0 (A step forward in Gesture technology)



**SUBMITTED TO: Mr. Prashant Kaushik**

**PROBLEM STATEMENT: -**

1. **Gesture Based Input: The first achievement of the project is to successfully get the correct Input via Gesture and further process it.**

**In this first version of the initiative CRONUS, we have tried to input gestures as the brush strokes of a paint like app. Further the strokes are correctly recorded and communicated over servers and programs for processing.**

**Moreover, the tech used to achieve this result is minimum possible as no additional censors or cam is used apart from the standard webcam that comes by default with any Laptop.**

**This also solves the problem of the painful job of handling the very famous paint app using the trackpad!**

1. **Image Processing and Matching: The project elaborates on the concept of evaluating and comparing hand made pictures which is a TOTALLY NEW CONCEPT in the field of image processing. The Algorithm used to compare hand drawn images(doodles) is totally exclusive to CRONUS 1.0 and can be called ‘THE CRONUS ALGO’.**
2. **Constantly Learning Application (AI): The application system is made such that it never stops learning! Every new User Input teaches the application to be more and more precise in its job of evaluating the gesture based hand strokes and calculating their resemblance to real world objects.**

**Technology Used:**

1. **Unity:**
2. **Custom Skybox: For Background animation**
3. **Line Renderer: For recording paint strokes.**
4. **Core C# Network Libraries (Not Unity Network manager): For Network management**
5. **Python 3.6: Libraries Used:**
6. **OpenCV 2: for Image processing**
7. **Math, Random: mathematical utilities**
8. **Scikit-learn: For creating and handling decision trees**
9. **Scikit-image: for similarity index score**
10. **Numpy : for image matrix (3d) manipulation**
11. **Socket: for creating UDP server to communicate with the app**
12. **Os : for file handling**
13. **Matplotlib: for plotting results**
14. **Front Web Cam of laptop (Standard/Default)**
15. **C++: For creating DLLs using CV namespace which makes Unity identify Gestures.**
16. **Preferred IDEs: a) Monodevelop: For C#**

**b) Visual Studio 17: For C++**

**c) Pycharm or IDLE: For Python**

**DIVISION OF WORK AMONG GROUP MEMBERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Title: CRONUS 1.0** | | | |
| **S.NO.** | **Enroll Number** | **Student name** | **Contribution**  **(percentage)** |
| ***1.*** | **15104002** | **Abhishek Tiwari** | **26.0** |
| ***2.*** | **15104011** | **Harsh Chawla** | **22.0** |
| ***3.*** | **15104020** | **Sajal Anand** | **26.0** |
| ***4.*** | **15104023** | **Sarthak Sehgal** | **26.0** |

**WHY THIS PROBLEM STATEMENT IS NOVEL?**

Many painting software is available in the software industry. Each of them has its own pros and cons. One of the main disadvantages of these painting software is the need of accurate pointing devices. Here we introduce the Cronus 1.0, which is a gesture based painting software. It uses the hand movements or movement of a particular color for drawing. User doesn't need an external hardware pointing device to draw. Instead the movement of a red colored object is used for drawing brush strokes. We designed this system as software product which aims to provide entertainment to the user. The software can be further upgraded by improving the current features. The disadvantage of the traditional drawing software is that they use traditional drawing devices like mouse light pens etc. Now it has been overcome by touch screen technology. **The next evolutionary technology that will take over the world will be the gesture technology**.

*By using gestures for communicating with system, we have unlimited ways to pass different messages to the system.*

Now moving deeper inside Cronus 1.0…This is not just any simple app that let your paintings go in some forgettable folder. Cronus is made to understand the user and his/her strokes. **The App trains on each user input and tries to guess what real world object resembles to the user’s drawing** (provided the object is available in our database through some previous input)!

The Idea is to let Users across different locations, systems and platforms use the app and feed it with their inputs OR paintings and tell the system what they have drawn. The database which is global is basically the training dataset which is used to predict what the current painting looks like.

There are several image comparison algorithms available on the internet but **none is for comparing deformed or basically “Hand Drawn Images”.** All the algorithms focus on image retrieval from well formed digital images with well defined image matrices. The Cronus Algorithm passes the images through a series of filters and tests and compare them for similarity as well as generates a score for **quantitative evaluation**!

The Project is inspired from the AI experiment game of Google which is still under research which evaluates a user drawn image as “correct” or “wrong” comparing it to other user drawn images of the same object.

***Our Application is a limited extension of this experiment by google.***

Why Limited? Because We don’t have as many users as google so our dataset is limited. But this isn’t a real limitation as if exposed to enough users the app is expected to give enough accuracy for practical usage!

Why Extension? Because we don’t restrict our users to draw a particular object primarily. The App has two parts, one is same as what Google’s experiment is where user gets an instruction to draw something and then the drawing is evaluated. The 2nd part called the “teaching mode” tells user to draw anything he wants and then tries to guess what he has drawn and saves the painting in the database.

As a summary:

The problem statement is novel because –

1. The Gesture technology is relatively new and majorly untouched.
2. There are no well-known applications for laptops to “process” gesture inputs.
3. There are literally zero algorithms for comparing human drawn images for similarity.
4. THERE IS NO PAINTING APP WHICH COULD GUESS WHAT IN THE WORLD YOU HAVE DRAWN.

**How our implemented work will be different/ enhanced than existing work.**

**The Existing works are –**

1. **Paint app**
2. **Gesture Based smart paint app**
3. **Vision API for image comparison**

**How Better Than Paint app: -**

**Paint app is based on wired input media which is not as comfortable as the movements of human hand.**

**How Better Than Smart-Paint app: -**

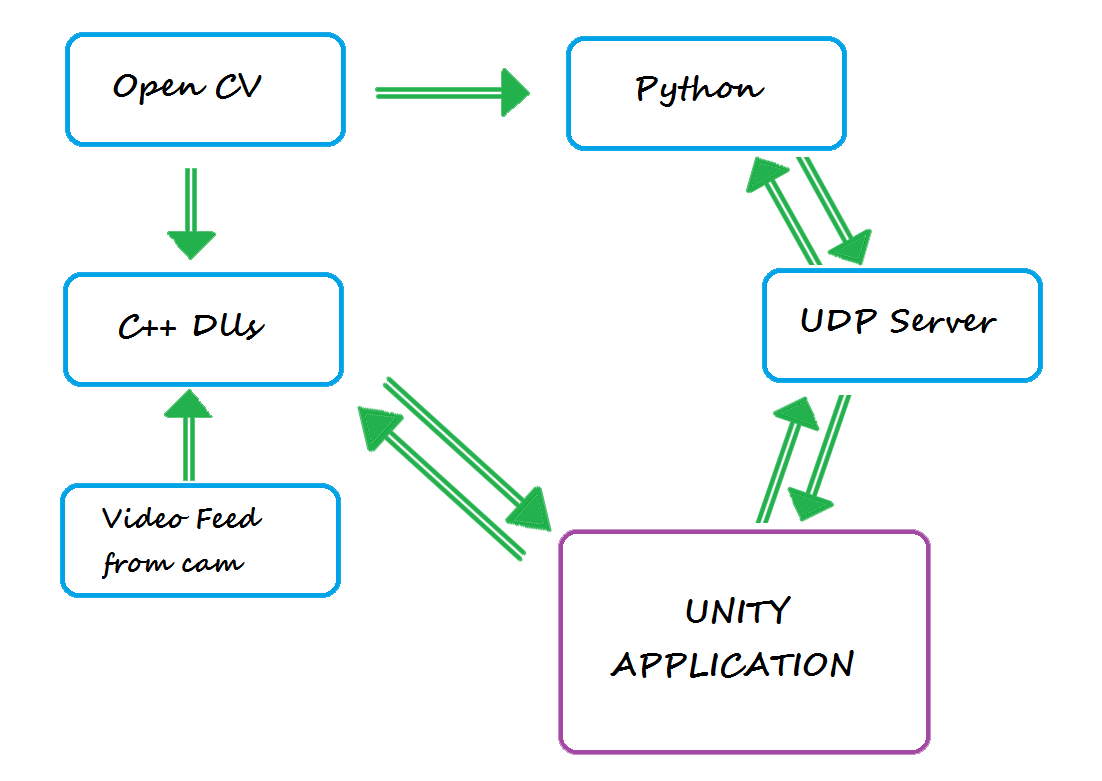
**Smart paint merely takes gesture inputs and saves the result without the scope of any further processing whereas CRONUS 1.0 further processes the images to make it more interactive and entertaining.**

**How Better Than Vision API: -**

**Clearly Vision API is a far more superior product by google with supernatural image processing capabilities but still it does not provide any service to process and compare hand drawn images which are going to be a big part of the future input tech.**

**Algorithms/ Approach used and Implemented:**

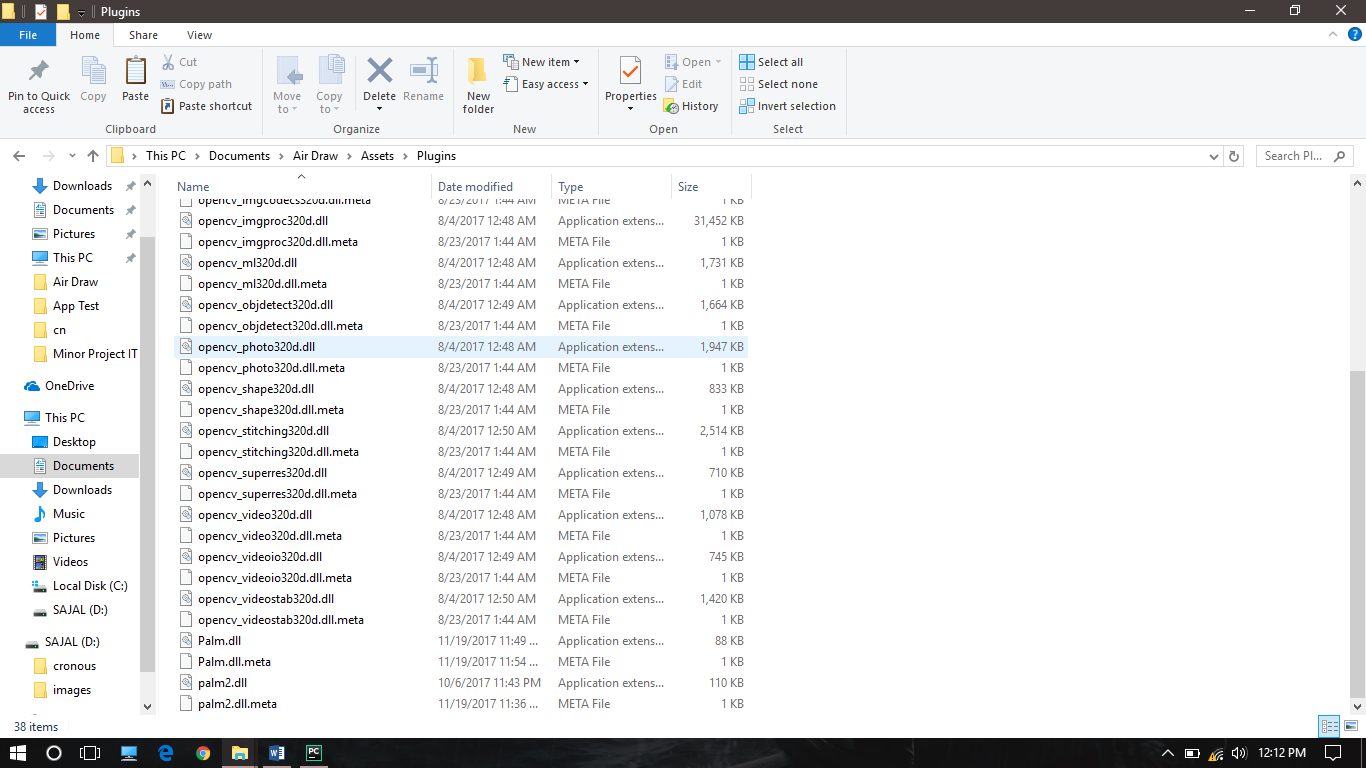
**Basic Structure of The App: -**



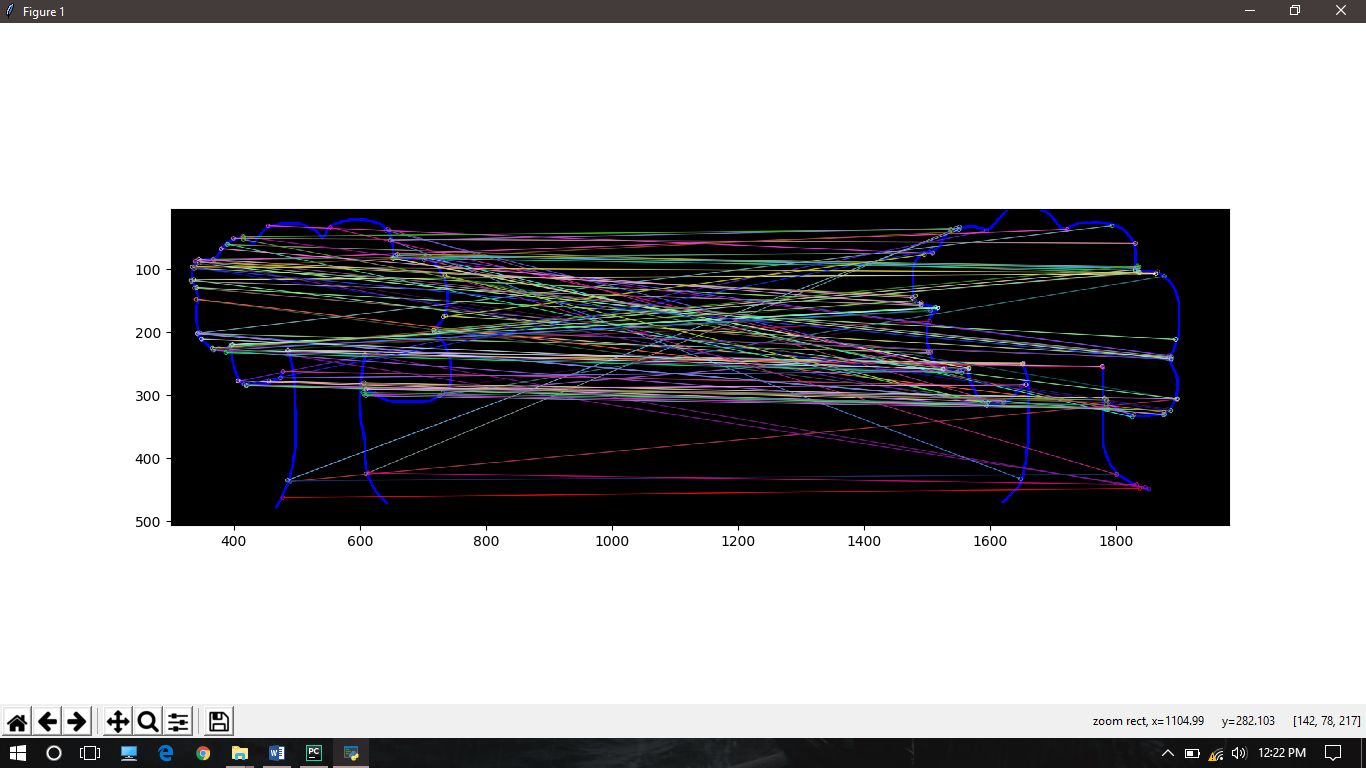
**The Flow Chart and detailed algorithm and process is further explained:**

**OpenCV: This Library comes in various packages for various languages. We have exploited two of these packages.**

1. **C++ OpenCV – The namespace cv and the OpenCV Library is used in C++ to Generate DLL files which contain the color detection features. Visual Studio is used to build the DLL files.**



1. **Python OpenCV – OR the cv2 library is used for image processing in following ways: -**
2. **Reading and writing images from a directory to the Program.**
3. **Creating ORBs for feature matching.**



**Fig.) Matching features of a tree with another tree.**

1. **Splitting Images into exact two halves. (Caution: Looks easy but it’s not)**

**No matter how big your screen is, the app works just fine in cropping and processing image. Also the resolution doesn’t matter when it comes to accuracy!**

**How the App Works:**

**i)Run the UDP server that will create a connection between t the Air Draw app and the algorithm containing program.**

**ii) Now User is shown a blank screen with 4 buttons: -**

**1.) Game: to toggle game mode on and off**

**2.) Test: to compare and check what the image is or what the result is in the game mode.**

**3.) Correct and Wrong: to guide the app about the guess made.**

**iii) By default, game mode is OFF which is also called the teaching mode. In Teaching mode, you can draw whatever you want and the app will try to guess what you have drawn using a learning algorithm which is explained below-**

1. **Image which is input using gestures is communicated from the app to the python script via the UDP server which is constantly running.**

**b) In the script firstly 2 scores namely MSE (or the mean squared error) and SSIM (or the structural similarity) are calculated which basically show how deviated the drawn image is from the closest matching image in the existing dataset.**

**c) Now the image is further taken on to the function which matches features of the image to all other images in the dataset. This is not as plain as it sounds.**

**First the drawn image is split in two equal halves to remove the redundant matches on wrong sides of the images.**

**Then left part of the drawn image is compared to left part of all the images in the dataset and the features are matched using the ORB descriptor with Brute Force Matcher!**

**Similar thing happens for the right part. Now the average of both the sides is taken and this is the value of the feature match result.**

**Now the entire data is fed to a dataframe. The dataframe consists of 2 features namely the feature array length and the overall shape.**

**Now the dataframe is trained on a decision tree to generate a similarity score and predict the label which is the name of the guessed image. Also, the guessed image is plotted using Matplotlib!**

**Notice that contribution of each image is counted while comparing a particular object with the dataset!**

**d) Now with the game mode on. There is already a target image similar to which the user is required to draw. Similar process happens but the score generated in the part c) which are MSE and SSIM are used to Evaluate a quantitative score of the image drawn by the user relative to the target image.**

**When the app thinks that you have drawn what it asked you to, you get a perfect score of 100 which is the motive of the game!**

**RESULTS:**

1. **openCV lib. was successfully compiled to DLL (dynamic linked**

**libraries) to be used for color detection purpose in unity directly.**

1. **openCV in Python is being used for image processing via \**

**Feature Matching Algorithm with good accuracy in a relatively**

**small Data set of images.**

1. **Python Algorithm is successfully setup to learn from input and**

**increase its Data set and hence accuracy gradually.**

1. **A Python server was setup to create a communication link**

**between the unity application and python image processing**

**module.**

1. **The application has 2 modes namely "Game" mode and**

**"Teaching" mode.**

1. **In game mode people draw what Cronus asks them and then**

**there drawing is rated on the basis of Cronus data set.**

1. **In teaching mode people draw what they want, and they can**

**give feedback to Cronus so that it can learn about new shapes**

**and images.**

1. **The biggest achievement of this project was, only using a red**

**color "thing" and a laptop which is available to people easily.**

**No extra hardware is needed for this application to work.**

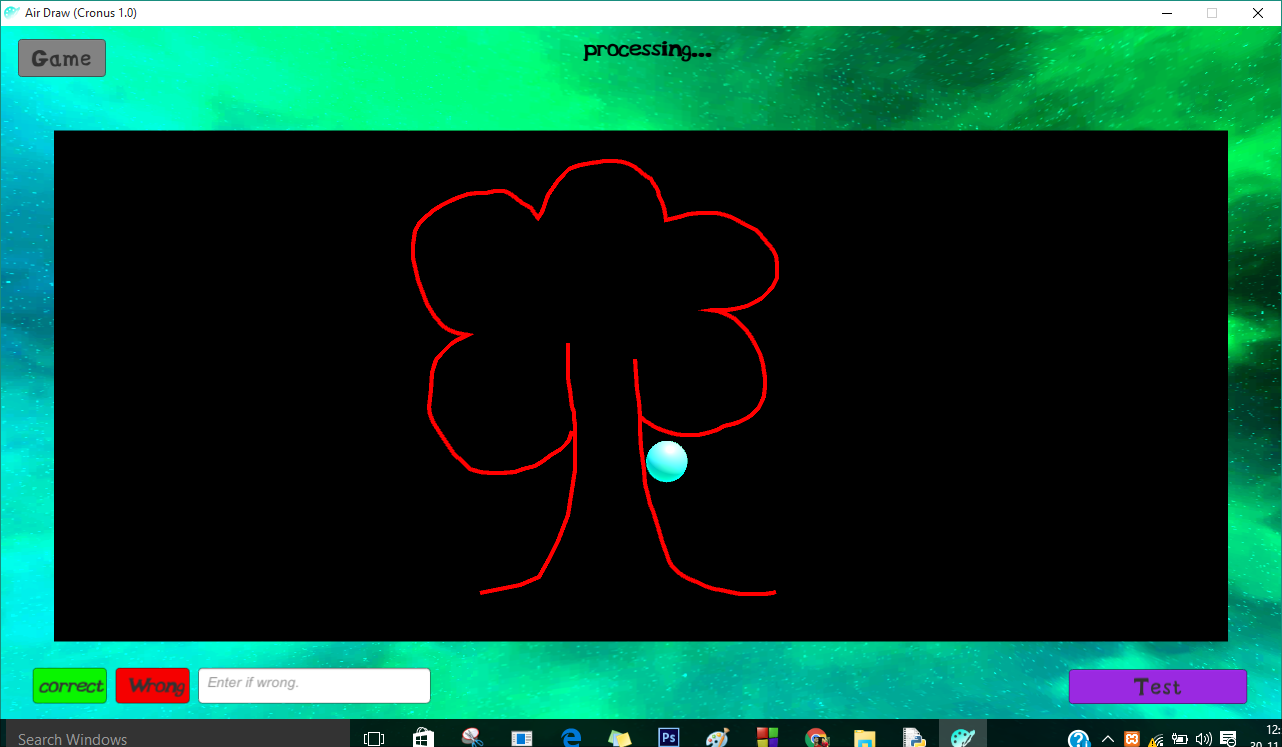
**RESULT VALIDATION/ TESTINGS:**

**We have taken 3 cases for testing the APP.**

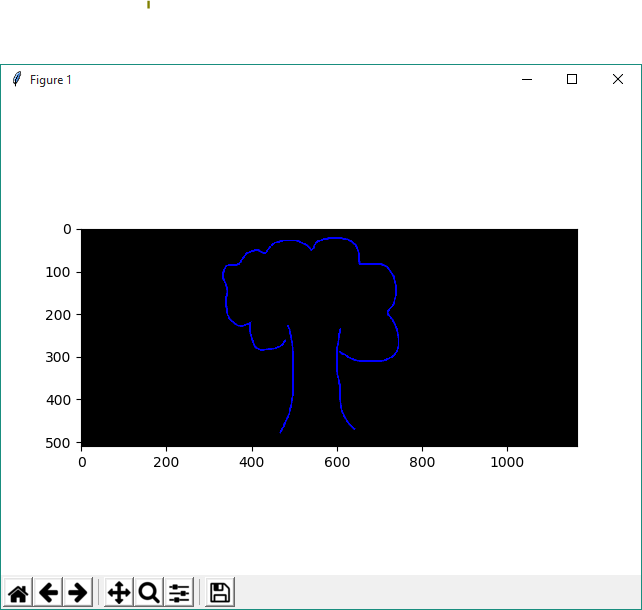
**Our Dataset consists of roughly 100 images.**

**Case1: Correct match (image drawn has enough matching images in the dataset)**

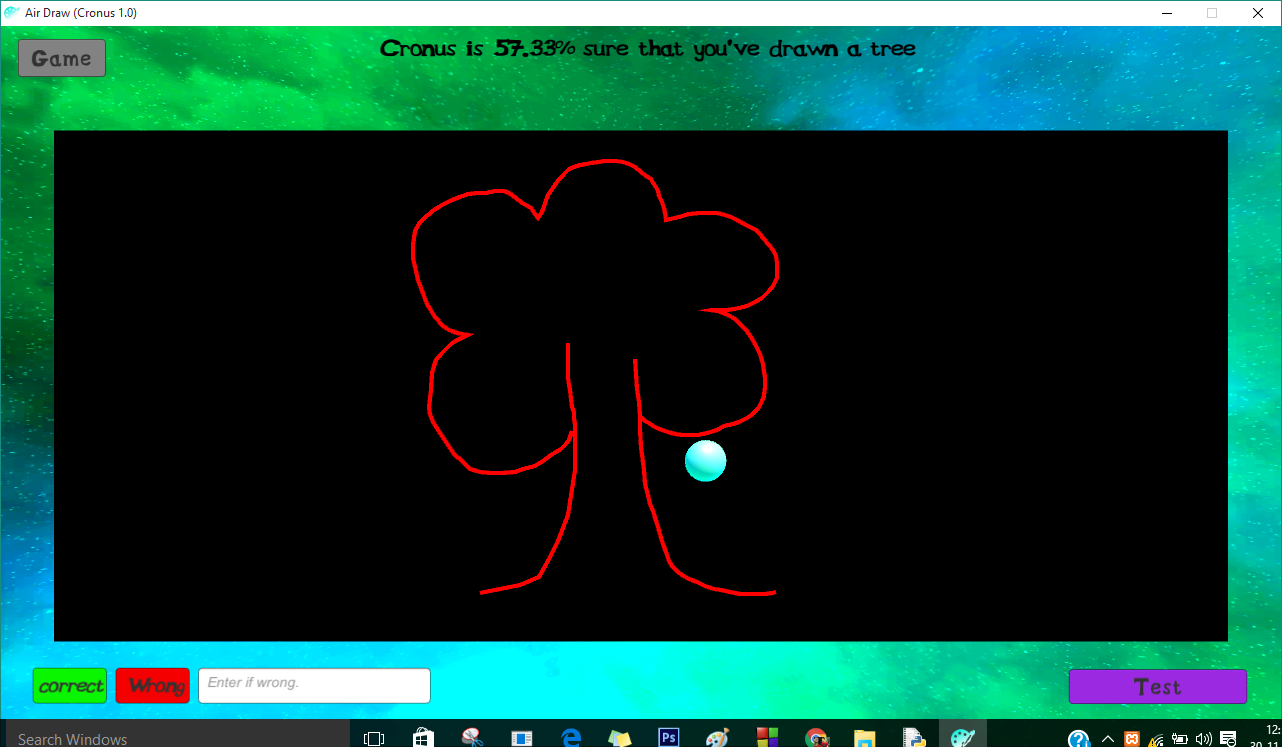
**Drawn Image**



**Resultant matching image:**

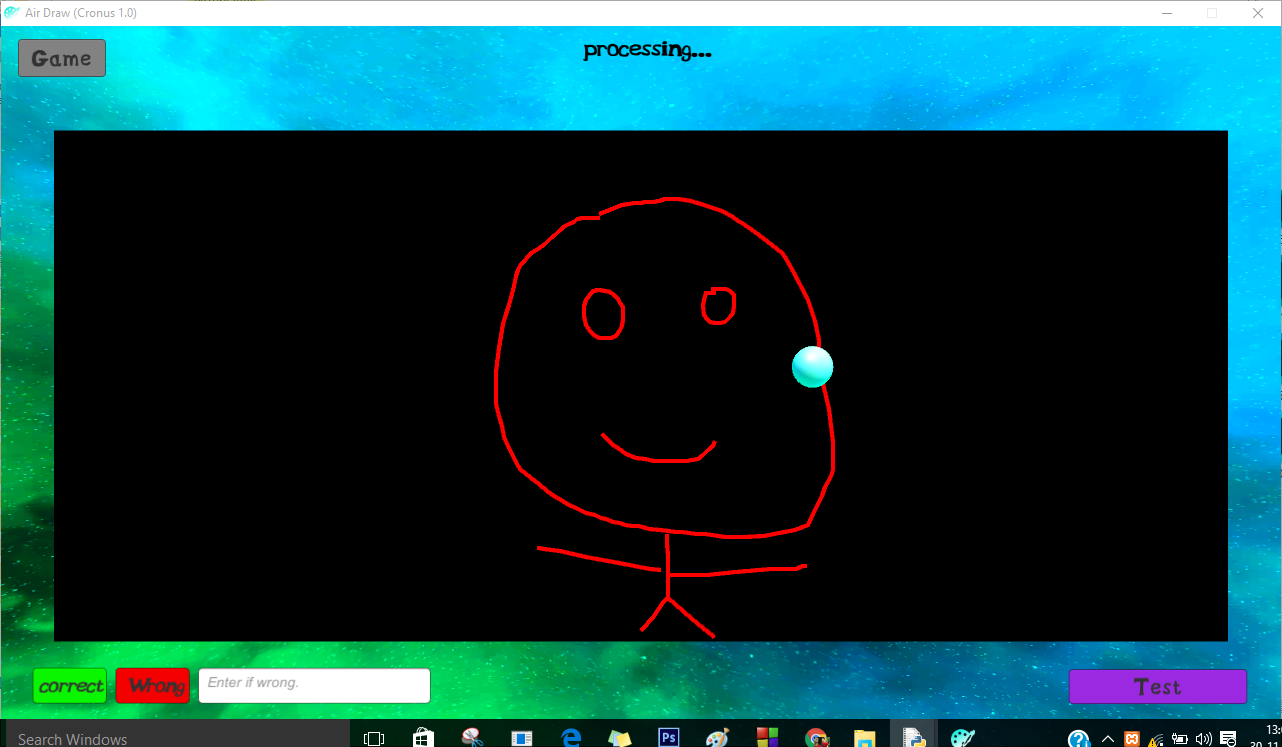


**Result screen in APP:**



**Case2: When not enough images similar to drawn image in dataset but still some similarity found!**

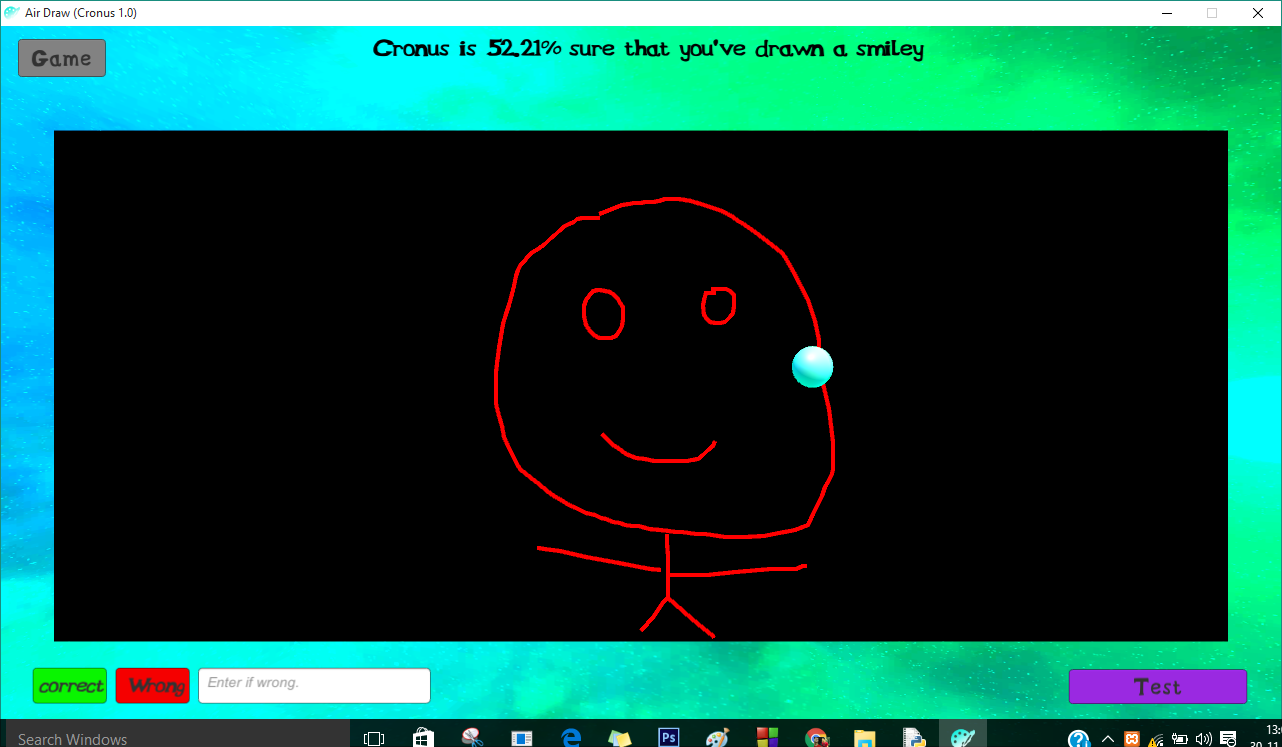
**Drawn Image:**



**Resultant image: (BEST POSSIBLE MATCH IN DATASET)**

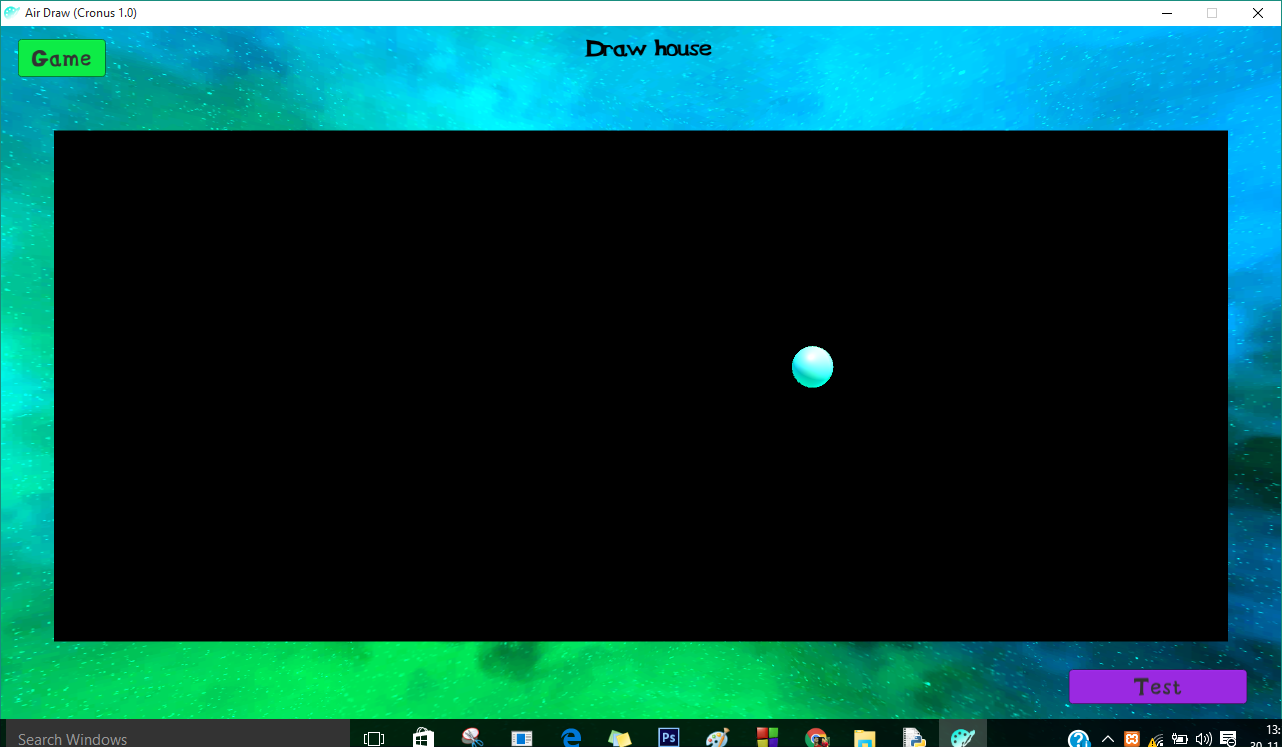


**Result on APP:**



**Case3: Game Mode ON:**

**Instruction On App:**



**Result On APP:**



**Future Scope:**

**1) We are in an era where the industries in the field of architecture and engineering are expanding at a very fast rate. The interaction between machine**

**and human thus needs to be done in a manner which requires least number of I/O devices giving high accuracy indeed. Gesture based interactions can**

**bring a lot of ease in such a scenario. It can help in the field of architectures and engineering which requires model designing and analysis.**

**2) Apart from industrial use, it can also be used at our homes and help us with a lot of gadgets we are surrounded with.**

**Examples: a) Reduce/Increase TV volume.**

**b) Turn ON/OFF AC.**

**c) Adjust time in a digital wall clock.**

**d) Change Fan speed.**

**PS: This all can be done using a main frame CPU having camera facility or your smart phones communicating to all these smart devices via WIFI connections.**

**3) In future, this form of image search based on visualization may reduce the barrier created by languages as an object can be referred differently by**

**people knowing different languages but it's visual form remains same for each and every individual.**

**4 ) Taking the gesture based image search a little further, this technology can be installed in the shops which can save a lot of time and work. Suppose you**

**go to a shop to buy a flower pot, now instead of wasting time in looking for the flower pot which you want or explaining that to the shop keeper, this**

**gesture based search can be used by the customers to immediately lookup at the images of all the flower pots resembling the one drawn by the customer.**

**References:**

1. <https://docs.opencv.org/3.3.0/dc/dc3/tutorial_py_matcher.html>
2. <https://experiments.withgoogle.com/ai/quick-draw>
3. <https://cmake.org/cmake-tutorial/>
4. <https://docs.opencv.org/2.4/doc/tutorials/introduction/linux_gcc_cmake/linux_gcc_cmake.html>
5. <https://docs.opencv.org/2.4/index.html#>