# Using computer vision to play the chrome's Dino game with facial expressions

Ahmed Elghabry, Amr Mohamed, and Esraa Gaber
University of science and technology at Zewail city, s-ahmedgabry, s-amrmoohamed, s-esraa.gaber@zewailcity.edu.eg

*Abstract* – **One of the computer vision applications is to make gaming funnier and more interesting. In our project, computer vision techniques such as face recognition, facial landmarks detection are used ti play the chrome's Dino game with our facial expressions instead of the keyboard.**

*Index Terms* – Dlib, Face detection, Facial land marks detection, Dino game.

## INTRODUCTION

Computer vision applications are applied in many fields and continue spreading. One of its applications is in gaming. Many games will be easier and more interesting if it could be played using our facial gestures.

In our project we implemented some computer vision techniques to play the Chrome's Dino game with our facial expression instead of the keyboard shortcuts. The game idea is based on making the Dino jump or crouch in order to avoid the obstacles it faces. Normally, this is done using the space and down button from the keyboard. But these actions can be mapped to some facial gesture like the mouth is open or closed, the eye bow's position, the eye is open or closed or any other convention.

In our implementation the Dino will jump when the player opens his mouth and continue running while the mouth is closed, Also, it will crouch when the player gets closer to the camera. This will be done through the upcoming steps:

- Real-time face recognition
- Facial landmarks detection
- Build the jump control mechanism for the Dino
- Build the crouch control mechanism
- Perform calibration
- Keyboard Automation
- Build the final application.

## REAL TIME FACE DETECTION

The first step in our implementation is to detect the face of the player in order to use his expressions in playing the game Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

First, the possible areas of the human eye are detected by testing all valley areas in the gravel level image. Then genetic algorithms are used to generate all possible facial regions, including eyebrows, iris, nostrils, and corners of the mouth. Each possible face candidate is then normalized to reduce the lighting effect caused by uneven lighting; and the wrinkle effect caused by head movement. The fitness value of each candidate is measured according to their projection on their face. After multiple iterations, all candidates with high fitness values are selected for further verification. At this stage, measure the symmetry of the face and check if each candidate face has different facial features.

There are two main open source libraries for face detection which are dlib and openCV, the one we used in our implementation is dlib and its face detection function dlib.get_frontal_face_detector(). This function returns the default face detector that can be used to detect the faces in any image.

## FACIAL LANDMARKS DETECTION

### *I. Overview*

The purpose of this step is to mark the most important face landmarks such as Mouth, Right eyebrow, Left eyebrow, Right eye, left eye, Nose, and Jaw. There are a variety of shape predictor algorithms depending on many factors, The algorithm used in dlib library uses ensemble of regression trees approach using the method described by Kazemi and Sullivan in their 2014 CVPR paper. The algorithm first examines a sparse set of input pixel intensities (i.e., the "features" to the input model), then passes the features into an Ensemble of Regression Trees (ERT), finally, it refines the predicted locations to improve accuracy through a cascade of regressors.

To train the shape predictor model many datasets with different features can be used such as iBUG 300-W dataset which estimates the location of 68 (x, y)-coordinates that map to facial structures on the face. Another dataset is HELEN dataset that can estimate 194 points of facial landmarks.

*II. Our implementation*

In our project we built our own neural network to train the model on an image dataset that highlights 15 facial landmarks which are left_eye_center, right_eye_center, left_eye_inner_corner,left_eye_outer_corner,right_eye_inner_corner, right_eye_outer_corner, left_eyebrow_inner_end, left_eyebrow_outer_end,right_eyebrow_inner_end, right_eyebrow_outer_end,nose_tip,mouth_left_corner, mouth_right_corner,mouth_center_top_lip, mouth_center_bottom_lip.
After training the model with this dataset, it was tested on our real time images and the results are shown in figure I below.

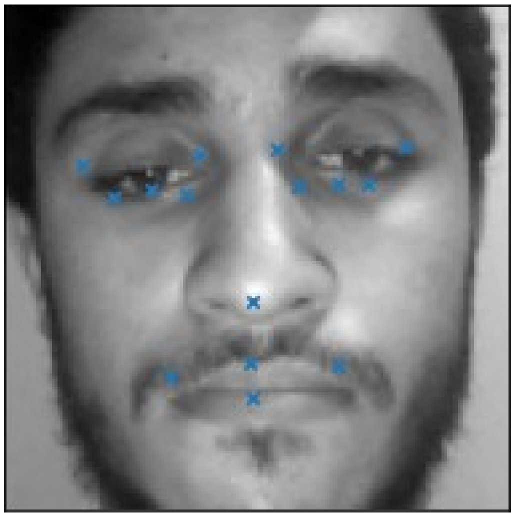

FIGURE I
FACIAL LANDMARKS EXTRACTED USING OUR TRAINED MODEL

Unfortunately when we used this model with our application it could not detect the mouth when it was closed because the dataset used for training the model contains only photos with the mouth is closed so when it was tested on images with mouth is open it could not detect that the mouth is open so it gave errors while playing the game. A solution for this problem is to train the model on a different dataset that contains images of people with open and closed mouth such as iBUG or HELEN datasets.

In order to make the game run, we used a dlib.shape_predictor function with pretrained landmarks detector model to detect 68 points of the player's face. These landmark points are shown in figure I. This model uses more key-points to define most important facial landmarks so it will give better accuracy than the 15 points model.

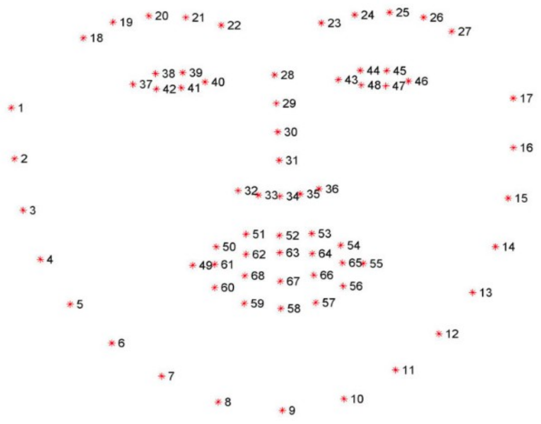

FIGURE II
FACIAL LANDMARKS EXTRACTED USING IBUG DATASET

**JUMP AND CROUCH CONTROL MECHANISMS**

The jump control mechanism is simple, The Dino should jump when the player's mouth is opened and continue running when the player's mouth is closed. This is done by calculating the euclidean distance between a pair of landmark points indicated in figure II to calculate a ratio of mouth height to its width. Using a threshold value for comparison the mouth can then be evaluated as being close or open.

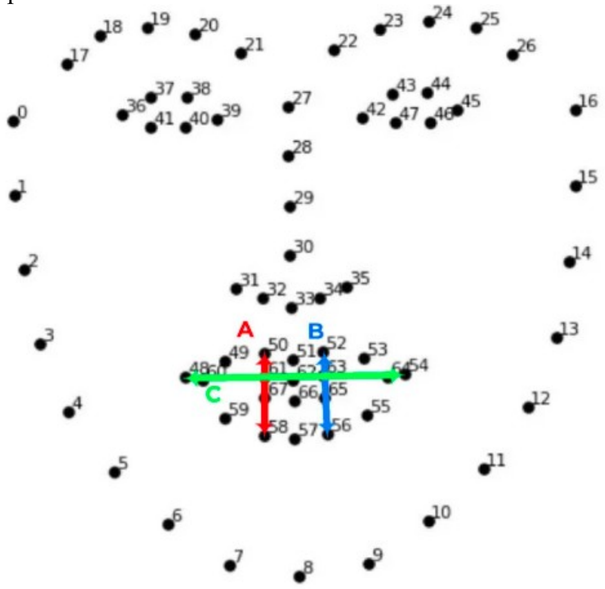

FIGURE III
THE EUCLIDEAN DISTANCE BETWEEN A PAIR OF LANDMARKS POINTS

The crouch control mechanism depends on the distance between the player and the camera, when it is near the camera the Dino should crouch and when it is far the Dino should continue running. This is done by calculating the euclidean distance between the top-left corner and bottom-right corner of the face bounding box. When the face is near the camera the distance will be greater, and when the face is close enough a key down event will be triggered causing the Dino to crouch.

## CALIBRATION

Authors do not add header and footer information. The Publications Chair will add the standard IEEE headers and footers as part of preparing the papers for publication.

## KEYBOARD AUTOMATION

The purpose of this step is to map the facial expressions extracted from the is_mouth_open() function that detects the state of the mouth and face_proximity() function that detects the distance between the face and the camera. To do this, PyAutoGUI API is used. PyAutoGUI is a simple API that lets a python script control the mouse and keyboard, it's used to automate processes.

## FINAL APPLICATION

The final application step brings all the steps together. The keyboard events causing the Dino to jump or crouch are determined based on the output of two functions is_mouth_open() and face_proximity().

## EVALUATION

The evaluation in our project was done based on three metrices:
1. IOU "intersection over union" which is an evaluative metric, it is used to tell how good our face detection algorithms works, we simply compare our predicted face algorithm, and ground truth from trained model. it is simply the ratio between the intersection of two boxes from two models over the area of union of them, a good IOU score will be > 0.5. figure IV shows the IOU of the face deteced by the algorithm.

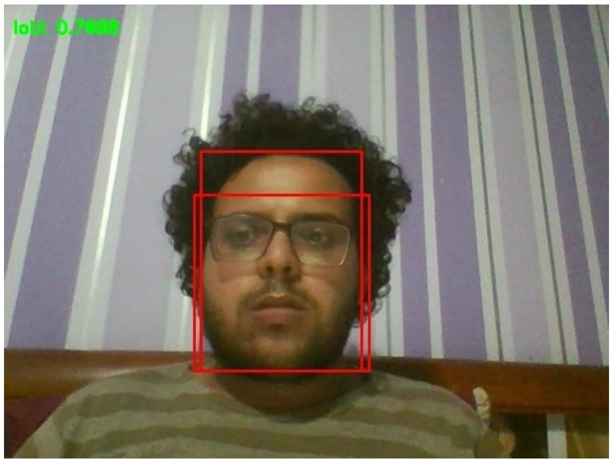

FIGURE IV
THE IOU OF THE DETECTED FACE

2. we have recorded a 20 second video ,the First 10 second the mouse is close, the second 10 second the mouth is open .we took those
# as our ground TRUTH label and Compared the model Results with Them.
# Results The True Positive are 238 The False Negative are 68 The False positive are 57 The true negative are 250

3. we record a 20 second video to calculate how much time the pipeline will take to process this video,The video consists from 613
# Frame ,it takes 60 second to process all frames with frame rate 10.178 Frame per second

## REFERENCES

[1] Taha Anwar ( BleedAI.com ) Momin Anjum (BleedAI.com), T. A. ( B. A. I. ), and M. A. (BleedAI.com), "Playing Chrome's T-Rex Game with Facial Gestures: Learn OpenCV," *Learn OpenCV | OpenCV, PyTorch, Keras, Tensorflow examples and tutorials*, 03-May-2021. [Online]. Available: https://learnopencv.com/playing-chromes-t-rex-game-with-facial-gestures/. [Accessed: 04-Jul-2021].

[2] R. Biswas, A. Rosebrock, J. Lawrence, O. Dalvi, Pranav, David, D. Lipcsey, Ajinkya, Itai, Kevin, S. Rahman, F. Norys, T.-S. Nguyen, Julien, A. Raza, Rj, R. John, W. Stevenson, Mike, Q. N. Tran, AI_Developer_OZ, Joel, Dexter, D. Hui, Lea, tigon7476, Cibya, P. Yin, Peter, and Nikos, "Training a custom dlib shape predictor," PyImageSearch, 17-Apr-2021. [Online]. Available: https://www.pyimagesearch.com/2019/12/16/training-a-custom-dlib-shape-predictor/. [Accessed: 04-Jul-2021].

[3] Anto, A. Rosebrock, Elliot, F. Ajibade, J. Shim, Danny, Mário, A. Orto, Dimitri, T. Amaratunga, T. Ideali, Hany, Sai, Khayam, T. Researcher, D. Addo, et al, Real-time facial landmark detection with OpenCV, and Face Alignment with OpenCV and Python - PyImageSearch says: "Facial landmarks with dlib, OpenCV, and Python," PyImageSearch, 03-Jul-2021. [Online]. Available: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/. [Accessed: 04-Jul-2021].

[4] "Facial Keypoints Detection," Kaggle. [Online]. Available: https://www.kaggle.com/c/facial-keypoints-detection/data. [Accessed: 04-Jul-2021].

[5] David, A. Rosebrock, and Rahul, "Tuning dlib shape predictor hyperparameters to balance speed, accuracy, and model size," PyImageSearch, 17-Apr-2021. [Online]. Available: https://www.pyimagesearch.com/2019/12/23/tuning-dlib-shape-predictor-hyperparameters-to-balance-speed-accuracy-and-model-size/. [Accessed: 04-Jul-2021].