Jenny Han

jennyh1

Recitation K

**Project Proposal**

Emotion Detector

**Problem**

Marketing products needs to be personal in order to be successful. Human emotion is challenging to quantify, however doing so has many benefits. In particular, companies waste money advertising toward people who do not respond to their message and then do not buy products.

**Solution**

Creating a program that is able to detect emotions through machine learning in real time would allow companies to better market toward audiences. Dynamically changing advertisements based on how the user is feeling could optimize the money spent on advertisements. This would also save companies money by redirecting wasted funds from uneffective marketing strategies as well as create more effective forms of advertisements. Smiley intends to detect the two user emotions, happiness and sadness, through photo and video inputs. Smiley will also rate the amount of each emotion the user is feeling and display it on the screen.

**Task Analysis**

*Video and Photo*

1. ML for detecting emotions
   1. Using CMU’s Cohn-Kanade Database to help the machine learn the difference between various emotions
   2. Use OpenCV Machine Learning algorithms
2. Happy/sadness rating scale
   1. Scaling the happiness/sadness that the machine detects

*Video*

1. Face Tracking
   1. Use:
      1. OpenCV face detection program to detect faces in the video
      2. OpenCV video processing to make the algorithm run faster
      3. ML for detecting emotions to calculate in real time

**Algorithm**

I will be using OpenCV for this project as well as CMU’s Cohn-Kanade Database for emotion. OpenCV has a few pre-trained classifiers for facial, object, eye detections as well as machine learning programs. Object detection in OpenCV uses Haar cascade classifiers to detect objects, which splits classifications into multiple layers. Each layers check for given features and if the image doesn’t pass at one level then it is not the object in question. This makes it much faster to run object detection programs because it does not check every single feature in question. Using the prewritten facial detection code and the Cohn-Kanade Database, I will train my program detect faces with emotions. This trained data will be used to analyze what the user is feeling both in real time and in photos.

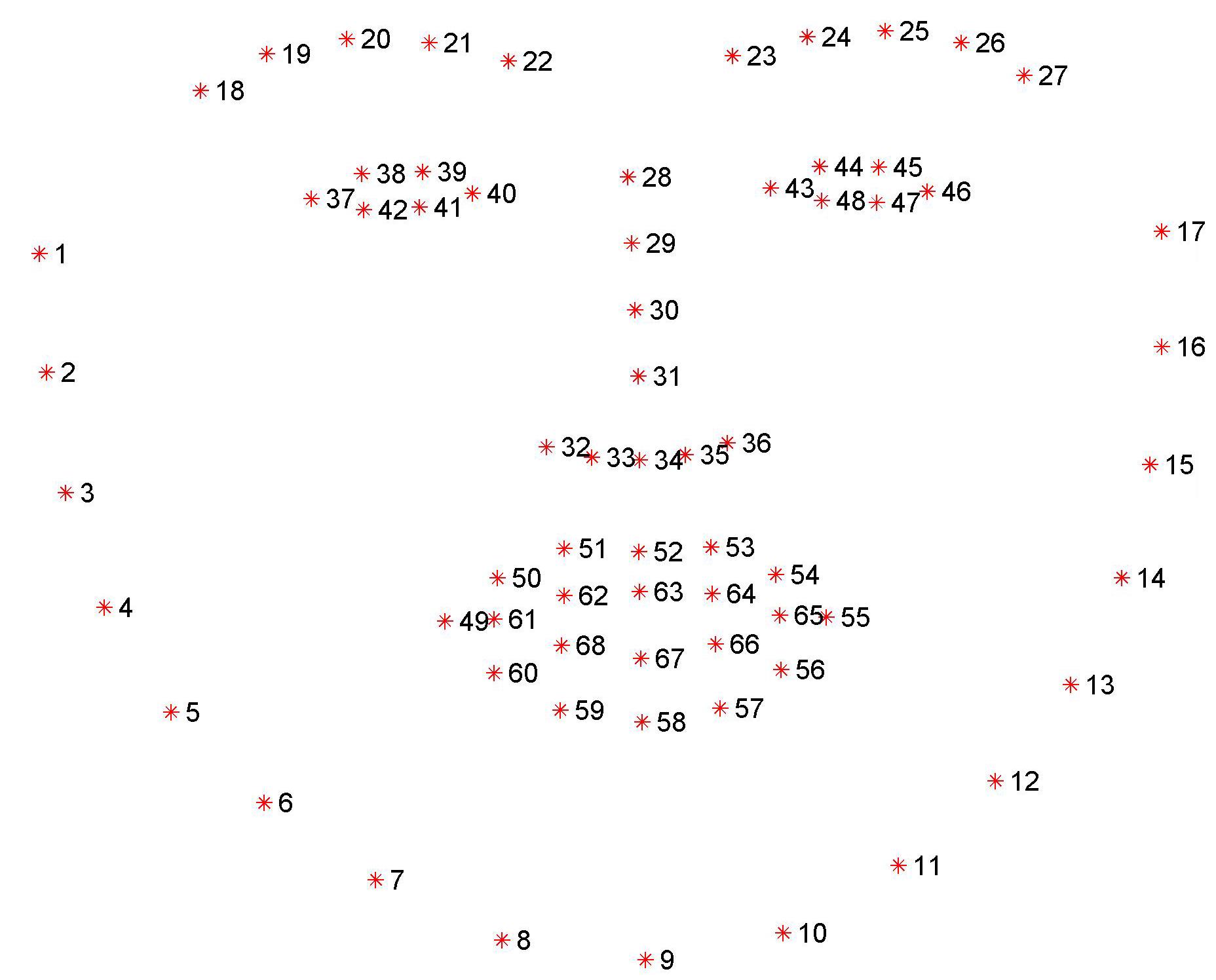
**Modules/Tools Used**

1. OpenCV
2. Cohn-Kanade Database

**Update 1**

Instead of incorporating machine learning and the Cohn-Kanade Database, I will just be using a combination of OpenCV(for Haar Cascades and locating features), dlib (for facial landmarking), and tKinter(for interface and timer). Currently, I have a dictionary of x,y points of 68 facial landmarks and visual framing for the face and mouth. I am working on calibrating a neutral face to compare the facial landmarks with in order to come up with accurate thresholds for movement personalized to each user. After I finish the calibration function, I will be measuring the slope and distance between facial landmarks to track movement of individual features. Various combinations of these feature movements will result in a different emotion classification. Instead of just smiling, I will also explore emotions like anger, disgust, sadness, neutral, etc.

Below is the positioning of the facial landmarks on a generalized facial layout.



**Update 2**

I incorporated the Tkinter framework with OpenCV and dlib as a way to present my project. Using this, I also added two start screens to introduce and explain how to calibrate. Also, I aimed to improve the users experience.

During the first 10 timer fired, the program “learns” the average coordinates of each landmark(the 1-68 key values) of the users face. It does this by taking 10 averages of a moving face. After, it compares the current expression with the calibration expression. I have also accounted for movement by the user left, right, up, or down on the screen by taking the change I X and change in Y from the current face landmarks to the calibration landmarks. It then updates the landmark key to match where the calibration key is. Thus, the user can move anywhere on the screen and the program will still be able to match the current face to the calibration face.

I tried around 17 functions to test facial movements, but was restricted in whether or not that movement was detectable or not in the program to create accurate results. I ended determining the top ten most reliable functions as the following:

1. eyebrow raised (up /T or down / F)
2. eyebrow together (yes/T or no/F)
3. eyes open (yes/T or no/F)
4. mouth open vertically small(yes /T or no /F)
5. mouth open vertically big (yes/T or no/F)
6. mouth open horizontally(yes / T or no /F)
7. mouth corners down (down/T or not/F)
8. one eyebrow up (up/T or neutral/F)
9. brow slopes down (down/T or neutral /F)
10. brow slopes up (up/T or neutral /F)

I would detect these traits by using a combination of distance, slope, and movement differences between the calibration key and the landmark realtime key. To improve accuracy, I would also take averages of multiple points to see how they moved as a whole – this reduced the error when the values for certain landmarks jumped around. To further improve accuracy of certain functions, I would change the threshold values (when the function detects a change in movement) to correspond to a more accurate result. For example, I would amplify some movents because the landmark key could not detect such small actions. To get these values, I would test the functions on my face and others to get an accurate threshold for movement.

Using true or false values from these movements, I combined together a tuple of them to correspond to an emotion. Again, I was limited to what emotions could be expressed due to the unreliability of the facial landmarks. Thus, I tested around 10 emotions and discovered that the most reliable ones were 1. Anger 2. Sad 3. Happy 4. Neutral and 5. Surprised.

In each frame, I would return a tuple with the 10 T/F values corresponding to the 10 motions being detected in real time. I would compare the tuple to the emotions set T/F tuple to determine the which emotion had the highest matches to what the user was feeling. This percentage was then printed onto the screen to show the user what ranges that the person felt.

Often times, I would notice that the user would equally match two or more emotions at a time. To improve accuracy, I then decided have emotions prioritize certain facial movements in the situation of two or more matches. For example, anger prioritized the 9th facial movement (brow slopes down) and surprise prioritizes the 5th facial movement (mouth opened vertially big).

With this, the most likely emotion would be found using a combination of these algorithms and displayed to the user.

# **Competitive Analysis**

# **Description of Your Planned Project**

My project is going to take a picture of someone’s face and then compute the emotion that that person is feeling at the time. It will be done through a webcam and there will be a set amount of emotions that the computer can recognize such as happy, sad, and content.

**Evaluating Your Competition**

AffdexMe

* App
* Starts with the camera on and subjects face
* 6 emotions to detect (Anger, Disgust, Fear, Joy, Sadness, Smirk) with percent lables on how much each emotion is detected
* Emojis that show around the face that correspond with what the person is feel

nViso

* Software sold to help advertisements/ user experience
* Can detect multiple faces at once in action
* Anonymously detect individuals
* Head pose tracking, face tracking
* Gives scale of emotions person feeling
* Emotions: happy, sad, fear, anger, disgust and sadness
* Estimate gender and age

Kairos

* Software sold to help companies with advertisements/better understand their market
* Attention measurement
* Emotion, Age, Gender, Facial Expression detection
* Emotions: joy, anger, disgust, sadness, fear and suprise
* Can compute multiple faces at once
* Face identification

**Identify Comparison Dimensions**

1. Camera: does the software include camera for live motion
2. Emotion Range: number of emotions detected
3. Multiple People: can the software detect multiple people at once
4. Usefulness: how useful is the product
5. Other Features: gender, age, ethnicity etc detection features

# **Comparison Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Camera** | **Emotion Range** | **Multiple People** | **Usefulness** | **Other Features** |
| AffdexMe | Yes | 6 | No | Personal | No |
| nViso | Yes | 6 | Yes | Commercial | Yes |
| Kairos | Yes | 6 | Yes | Commercial | Yes |

# **Summary**

All the different versions of emotion detection solve the core function of detecting emotion, however it is either done in a commercial way or in a personal, interactive way. nViso and Kairos are intended for companies to better understand their market, while AffdexMe is more about having fun with the app. Thus, nViso and Kairos have more interesting demographic information in the software and can detect multiple people at once. I think that my project will be more like AffdexMe in terms of how it is used. Thus, I think that incorporating the percentage range for emotions and having them written on the screen will enhance user experience as well as the ability to detect more than one face at a time.