Amirkabir University of Technology
(Tehran Polytechnic)

Emotion Recognition via Neural Networks Classifiers

Supervisor

Dr. Mohammad Akbari

By

Sepideh Niktab

Winter 2020

Index

Introduction

Facial expressions play an important role in recognition of emotions and are used in the process of non-verbal communication, as well as to identify people. They are very important in daily emotional communication, just next to the tone of voice .They are also an indicator of feelings, allowing a man to express an emotional state. [8]

Reading emotional expression is one of the most difficult tasks for humans, let alone computers. [7] When two people look at the same photo they might have different opinions about the emotion of the person in the photo. Until recently computers weren’t good at detecting emotional expression but in the recent years we can see significant advances in deep learning has brought up speed, efficiency and accuracy in detecting people’s emotions in photos and videos. In this article first we will understand what exactly emotion recognition is, see some strong APIs In this field, know more about emotion detection applications, Startups which are pioneer in this field and we can see name of some good articles and datasets for emotion detection projects.

What is Emotion Recognition?

Emotion recognition is the process of identifying human emotion. People vary widely in their accuracy at recognizing the emotions of others. Use of technology to help people with emotion recognition is a relatively nascent research area. [6]

The Emotion Recognition algorithm gives you the emotion in the given photo or video with its corresponding confidence interval.

Emotion recognition algorithms are based on Convolutional Neural Networks. CNN’s are an algorithm design that reflects a network similar to the human visual cortex.

Even though CNN’s have been around for decades, it’s only recently that computing power has caught up with the algorithm’s capabilities. This enables CNN’s to run in scaled, production environments. [7]

Why we need Emotion Recognition?

If businesses could sense emotion using tech at all times, they could capitalize on it to sell to the consumer in the opportune moment. The truth is that it’s not that far from reality. Machine emotional intelligence is a burgeoning frontier that could have huge consequences in not only advertising, but in new startups, healthcare, wearables, education, and more.

There’s a lot of API-accessible software online that parallels the human ability to discern emotive gestures. These algorithm driven APIs use facial detection and semantic analysis to interpret mood from photos, videos, text, and speech. [9]

How do Emotion Recognition APIs work?

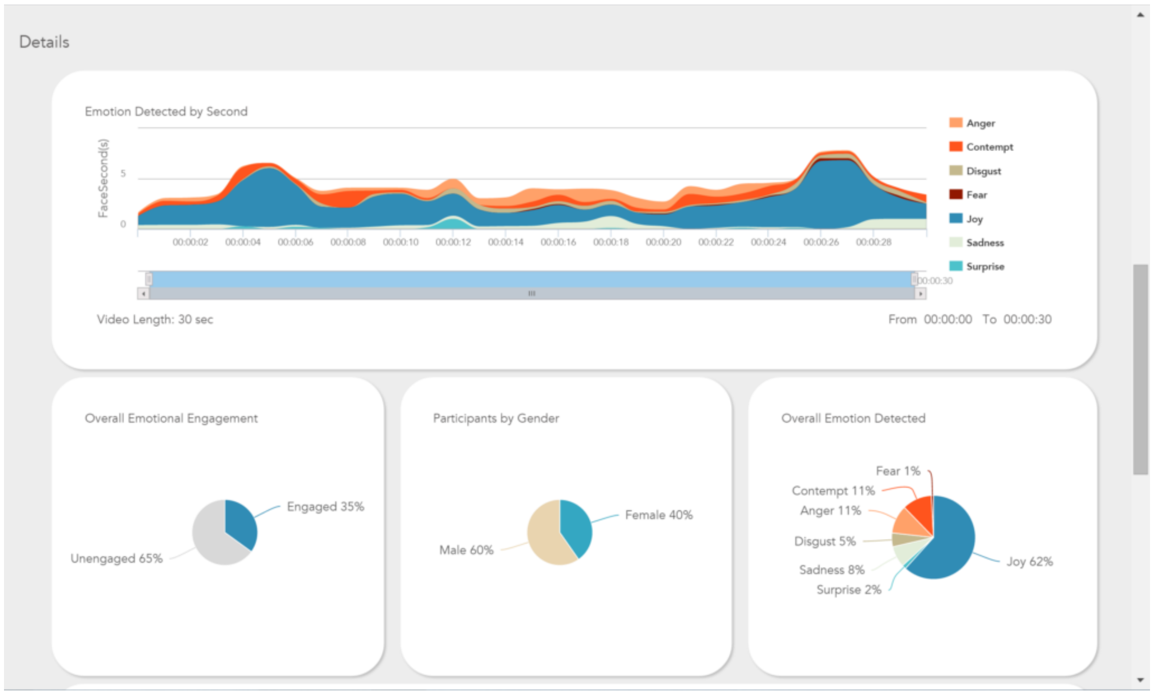
Emotive analytics is an interesting blend of psychology and technology. Though arguably reductive, many facial expression detection tools lump human emotion into 7 main categories: Joy, Sadness, Anger, Fear, Surprise, Contempt, and Disgust. With facial emotion detection, algorithms detect faces within a photo or video, and sense micro expressions by analyzing the relationship between points on the face, based on curated databases compiled in academic environments. [9]

Facial Detection APIs

These computer vision APIs use facial detection, eye tracking, and specific facial position cues to determine a subject's mood**.** There are many APIs that scan an image or video to detect faces, but these go the extra mile to spit back an emotive state. This is often a combination of weight assigned to 7 basic emotions, and valence — the subject's overall sentiment. [9]

1. Emotient

Emotient is an image recognition technology. Emotient allows faces, emotions, and other features to be recognized and identified in photos and images. The Emotient API allows developers to access and integrate the functionality of Emotient with other applications and to create new applications. Public documentation is not available.[10]

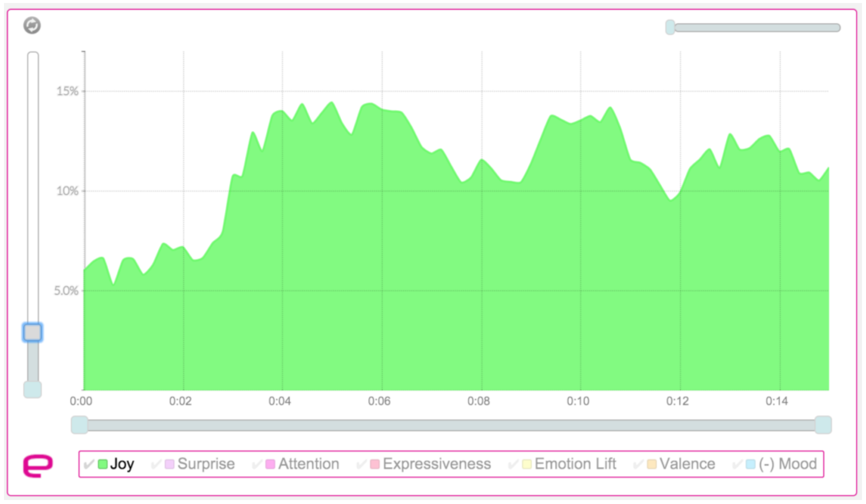


2. Affectiva

Affectiva is an emotion measurement technology company that grew out of MIT's Media Lab. Affectiva has developed software to recognize human emotions based on facial cues or physiological responses. Among its commercial applications, this emotion recognition technology is used to help brands improve their advertising and marketing messages. Another major application has been in political polling. [11]

3. EmoVu

EmoVu specializes in creating emotionally intelligent tools that can perceive the emotions of people by analyzing micro expressions using webcams. Supporting standard video, image, as well as image sequence formats, this emotive predictive analysis can help determine a user's opinion on the content they are viewing in real time. [12]



4. Nviso

NVISO provides artificial intelligence solutions (AI) for organizations to deploy applications that understand human behavior. It develops Applications and Software Development Kits (SDKS) as well as operates a Data Service platform that enables businesses to launch and incubate AI solutions. The company also operates a research and development laboratory that conducts research to deploy AI to detect and predict human behavior. NVISO was founded in 2009 and is headquartered in Lausanne, Switzerland with additional offices in Japan, UAE, and Australia. [13]

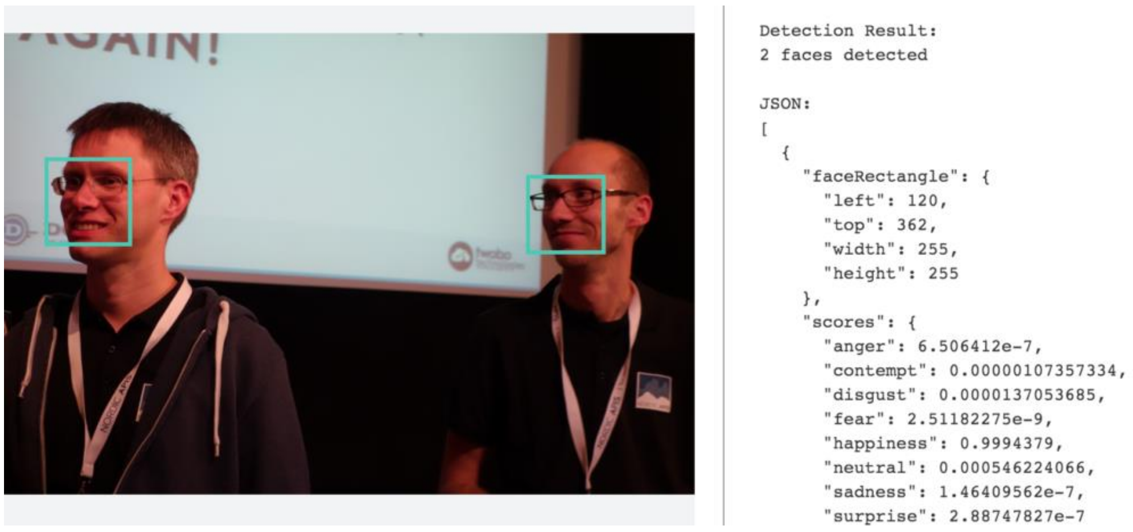
5. Kairos

Kairos is a software that understands people with face recognition technology. It contains human analytic features that can be used to gather, real-time insights about people interacting with the system. These key features include face detection which can find and track faces in any video, photo or image.

Kairos also does face identification by matching faces to their owners. It then does face verification to confirm who the user is .It detects emotions like anger, sadness , fear, among others. And also detects ages like being young or an adult and also detects if the user is male or female (gender). Kairos has the ability to detect multiple faces in crowds and groups.[14]

6. Project Oxford by Microsoft

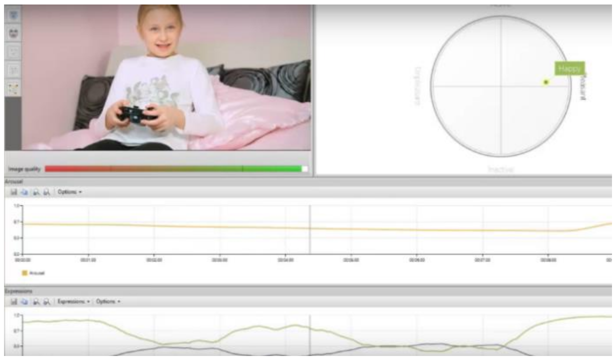
Microsoft’s Project Oxford is a catalogue of artificial intelligence APIs focused on computer vision, speech, and language analysis. The API only works with photos. It detects faces, and responds in JSON. [9]



```
Detection Result:
2 faces detected

JSON:
[
  {
    "faceRectangle": {
      "left": 120,
      "top": 362,
      "width": 255,
      "height": 255
    },
    "scores": {
      "anger": 6.506412e-7,
      "contempt": 0.00000107357334,
      "disgust": 0.0000137053685,
      "fear": 2.51182275e-9,
      "happiness": 0.9994379,
      "neutral": 0.000546224066,
      "sadness": 1.46409562e-7,
      "surprise": 2.88747827e-7
```

7. Face Reader by Noldus

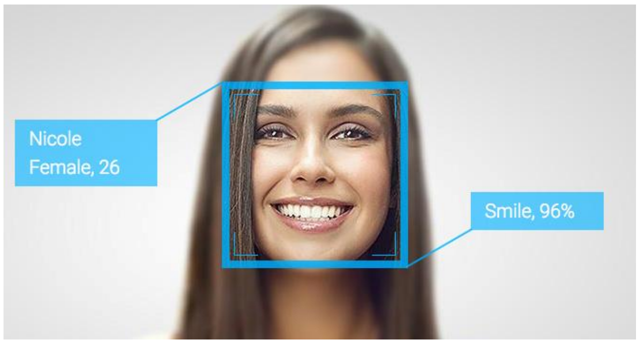
The FaceReader API by Noldus analyzes emotions and allows integration with 3rd party software to respond to the results. Used in the application of surveys, this can be a good application for experiment and research.the Face Reader API by Noldus is based on machine learning, tapping into a database of 10,000 facial expression images. The API uses 500 key facial points to analyze 6 basic facial expressions as well as neutral and contempt. Face Reader also detects gaze direction and head orientation. [15]



8. Sightcorp

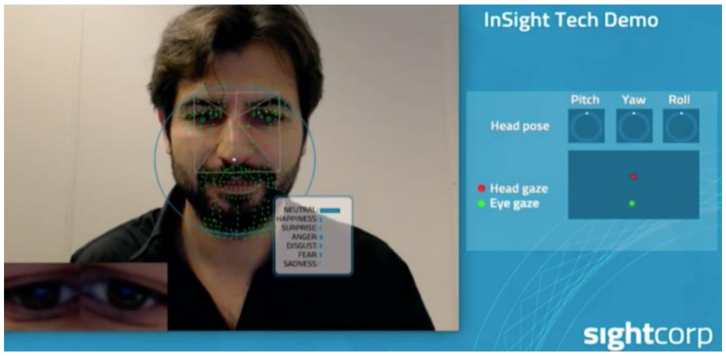
Sightcorp is an AI spin-off from the University of Amsterdam (UvA) specializing in face analysis and face recognition software. Through computer vision and deep learning, Sightcorp can analyze faces in images, videos, and real-life environments.

Their Face Analysis Technology is a pioneer in the real-time video analytics and audience measurement designed for digital signage, retail and other industries. They offer anonymized automated analysis of facial expressions like happiness, together with age, gender, head gaze and attention. [16]



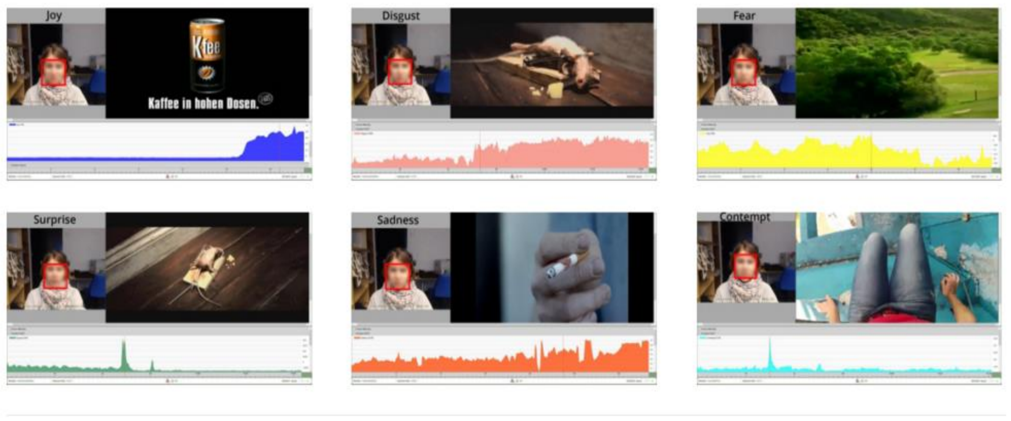
9. Face++

Face++ is a facial recognition and detection service that provides detection, recognition and analysis for use in applications. The Face++ API provides the detection and analysis of facials including two subsets of 23, and 81 points, as well as, age, gender, glasses, race, and others. The API uses REST calls and returns JSON, an account is required with services. [17]



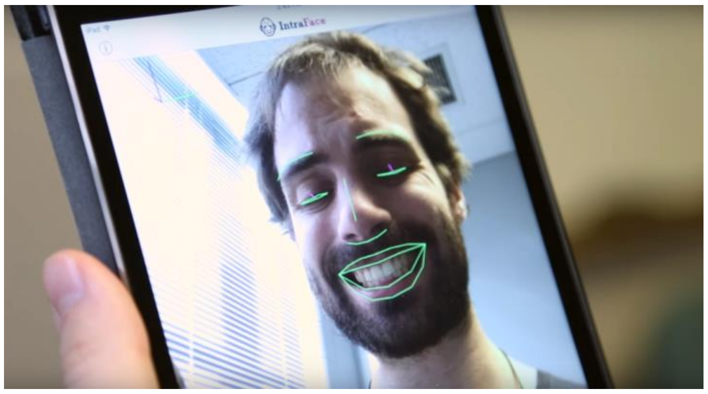
10. Imotions

Imotions is a biometrics research platform that provides software and hardware for monitoring many types of bodily cues. Imotions syncs with Emotient’s facial expression technology, and adds extra layers to detect confusion and frustration.[9]The combination of input and output API in iMotions allows for a closed data loop experience. iMotions is able to send out data received by an external application and in turn process the data and send it back into iMotions.[18] The Imotions API can monitor video live feeds to extract valence, or can aggregate previously recorded videos to analyze for emotions.[9]



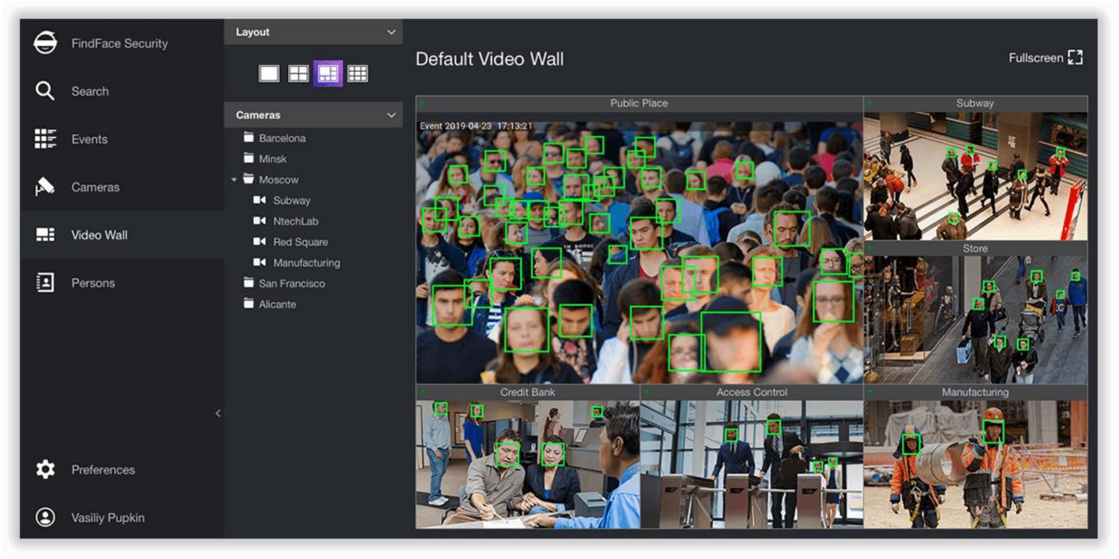
11. FacioMetrics

Founded at Carnegie Mellon University (CMU), FacioMetrics is a company that provides SDKs for incorporating face tracking, pose and gaze tracking, and expression analysis into apps. Their demo video outlines some creative use cases in virtual reality scenarios. The software can be tested using the Intraface iOS app.[9]



12. Findface

The Findface software utilizes the NtechLab face recognition algorithm to recognize 7 basic emotions as well as 50 complex attributes. It purportedly has a degree of 94% accuracy recognizing 7 emotions: joy, surprise, sadness, anger, disgust, contempt, and fear. Findface does not offer a web API for the emotive recognition, however, it does provide a powerful SDK. [9]



Emotion recognition applications [19]

1. Ads and emotion

In a world which each time is more saturated with audiovisual content, to generate advertising campaigns for impact, is a challenge which brands face every day. Numerous neurological studies have shown, that advertising that appeals to emotions has a greater impact on long-term memory, essential element in the launch of a product or service, or to consolidate a brand image. 80% of our decisions of purchases are based on emotions. Measuring the emotional impact of the advertising content is without doubt one of the main challenges. To know what the impact of a certain advertising video in your target audience will be.

Ads & Emotions allows you to predict what is the emotional impact of an advertising campaign in the audience through the technology of facial recognition of emotions .The software allows to know the levels of emotional activation, the engagement generated or the relevance, understood as an emotional memory of a video advertising, to know their strengths and weaknesses regarding their target audiences. Ads and emotions also have the online platform where the emotional evolution of the sample is set in motion, before the advertising stimulus. It is reflected in the 6 main emotions, 5 emotional metrics and 8 types of moods.

What information is obtained

GLOBAL EMOTIONAL IMPACT

• Mood generated by the stimulus.
• Levels of activation, engagement and satisfaction which generates the stimulus.
• Degree of emotional recall generated through of the calculation of relevance.
• Comparison of impact generated by various advertising videos about the sample.

CONTENT ANALYSIS

• Identification of the strong points of the announcement that generate greater satisfaction and engagement.
• Identification of rejection points generated for the stimulus.
• Analysis of the emotional impact of the stimulus in various segments of the sample (gender, age and others).

2. Emotional product test

The product test is the main research tool used to test the impact of all types of products on the market. This helps to predict the success of a new release or to introduce improvements to existing products. Although survey methodologies are getting more sophisticated in this field, it is not always easy for a standard panel to accurately move sensory experience into scales or exact categories. In this aspect, non-intrusive emotion measurement techniques help the researcher to evaluate the sensory impact of the product which reverts to the quality of the study.

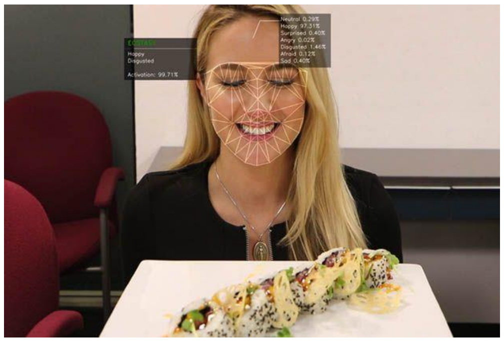
Emotional Product Test is the tool that allows knowing the emotional and unconscious response of the consumer to a particular product. Emotional Product Test is adapted to the research methodology in the laboratory with panels of consumers, offering accurate data to evaluate the experience with the product .The tool can be adapted to the following sectors:

1. Food

As has been demonstrated scientifically**,** taste and smell are the senses that send the most information to the brain, in fact, they are the senses that generate the most memories in the consumer. The facial coding technology of Emotion Research Lab transforms this impact into main and secondary emotions to detect the pleasure or dislike for different products or the emotional activation produced by different flavors and textures.

What information is obtained

• Level of satisfaction with the product through basic and secondary emotions.
• Emotional activation produced by different types of products and their characteristics: texture, smell and appearance.
• Evaluation of the coherence between the verbal and emotional respon

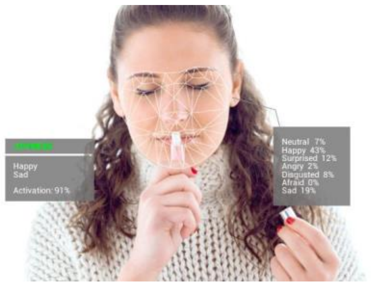


2. Cosmetics

Cosmetics and personal hygiene products have a special sensory impact on the tact and smell of consumers. Facial coding technology measures the consumer’s reaction to the test and trans-forms it into emotions, allowing them to know their reaction at the time of its application and in later moments.

What information is obtained

- Identify the sensitive impact on the consumer in terms of effect, touch and smell.
- Evaluation of the coherence between the verbal and emotional response during and after the test.

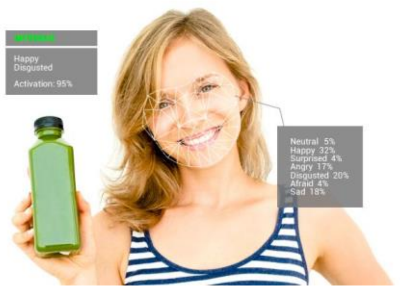


3. Packaging

The emotional test of packaging allows measuring the visual and user experience of a particular package. In this way, the emotional analysis allows to exactly know the reaction of a panel of consumers in the manipulation of the package and to evaluate the emotional performance of the different tasks performed in the test.

What information is obtained

• Emotional impact of the package in terms of design.
• Emotional impact of the use, detecting those elements that reveal greater satisfaction.



3. User experience

The investigation about the interaction between the individual and the computer has been limited just to the measure of effectiveness and efficiency of the use of a specific interface. At the same time, they recognize the data collection's limitations by asking the participant, right after the experience, the evaluation of it. Those limitations are due to the rationalizing's bias that pretends the participant's identification of their feelings and mood during the test.

What it is about

Emotional User Experience allows you to know which of the interface elements of your web or device are more attractive to the users. The emotional facial recognition software translates the facial micro-expressions into the 6 basic emotions, allowing the measurement of metrics such as activation, engagement and experience (valence) as well as to make comparisons between types of interface. In parallel, Emotional User Experience has the technology of eye tracking which allows the identification of the zones or elements that receive greater attention by the users, being able to associate the main emotional peaks with certain concentrations on the heat map.

How it is works

Emotional User experience is able adapted to any format on which the test is performed, either on PC, Tablet, or Smartphone (iOS and Android) in the following ways:

1. Online

Use the application to perform the test, recording, through the built-in camera, the facial expressions of the participant and tasks that the participant performs on the screen. The data is stored automatically in the Emotion Research Lab cloud.

2. Offline

Send two videos with the facial recording of the participant and the performed navigation on the device screen. The videos will be processed by the Emotion Research Lab software to get the emotional data. Mp4 video file recording, minimum size 640 x 360 (16:9).

What information is obtained?

GLOBAL EMOTIONAL IMPACT

• Elements that generate greater activation, engagement and satisfaction while the navigation.
• Main sub-emotions associated with specific moments while the navigation.
• Comparison of different interfaces and the emotional impact associated.
• Global emotional profile of the user experience.

ANALYSIS OF THE TASKS

• Identification of tasks that generate more positive emotions.
• Identification of the tasks that cause the greatest degree of rejection.

• Analysis of the emotional impact of the stimulus in different segments of the sample (gender, age and others).

4. Emofocus

The focus group is one of the qualitative techniques, that is most used today in the commercial investigation. It applies to positioning studies, brand and corporate identity, evaluation of packaging, advertising, usage habits or for knowing new opinions on some product or service.

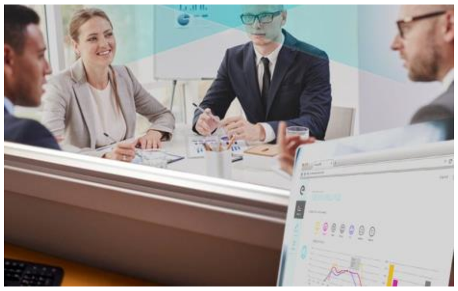
Emotion measurement technology for qualitative research

This technique allows the researcher to obtain valuable information to know the reactions of consumers in a group, although it requires a process of further analysis while away from the emotional situation in which the discussion takes place.
Since much of the exchange of ideas move in concrete situations and social contexts, understand the feelings that cause or provoke a certain type of affirmations and arguments to help to better understand what has happened in the group.

Emofocus is the first application of facial recognition of emotions that adapts to the technique of the discussion group through two forms:

1. EMOFOCUS live

Through installed cameras in the group room, the real-time reactions of the participants are followed on the screens in the observation room. This allows the observers to know the main and secondary emotions of each member in the group. Emotional information is displayed in real time and can be used to improve the research notes, warn the moderator to redirect the speech to the full availability of the client (present or online).



2. EMOFOCUS analytics

The session is recorded and we analyze the most relevant fragments to study. We develop a detailed analysis of the fragments based on a previous briefing. The resulting report makes it possible to compare the key moments, helping the research team in the analysis of the group.

What information is obtained

Emotion pattern of the group

• To know which emotional profile accompanies the different interventions and to evaluate their coherence with the discourse.
• To detect the intensity of the consensuses and disagreements to which the group comes to weigh the intensity in the overall discussion.

Level of implication connected to the participants in the discussion

• Identify which topics emotionally activate the participants and which do not (activation and neutrality).
• Emotional profile of the participants and impact of the different topics on them.
• Get the mood of the participants throughout the discussion.

Information on group training

• Information on the emotional profile of leaders and followers.
• Real-time data for the client and for the research team.
• Real-time and offline data to improve group moderation.

Report

EMOFOCUS live
Real-time recording with the participants of the group in the observation room and/or in a tablet for the moderator of the group.

EMOFOCUS analytics
Report with results and the emotional profile of the participants at different times and their comparison with the speech. Detailed analysis after the hand in of the brief by the client.

5. Emotional branding impact

Every brand understands the importance of knowing, what image they want to show around the world. The size of the logo, the color or even the packaging's design determines the relation between the product and the consumer.

Emotions and brands

Dominating the symbolic languages means to connect with the emotions, which are the ones that go along with the ideas and the expectations we have about the brands and the products. Due to the importance of our emotions in our perceptions, it is important to know and understand the emotional impact of a design to be able to reach the full potential of the success.

The Emotional Branding Impact records the emotional impact of the brand, the advertising message or the packaging design. Thanks to the technology of the Eye Tracking, it allows us to know which are the areas of the image , where the consumer focuses more and which are the associated emotions to the main focus of attention. This means being able to make comparisons between different designs and to know the preference of different types or segments of consumers. The Emotional Branding Impact uses the online platform for measuring emotions, which can be adapted to either PC, tablet or smartphone (iOS and Android)

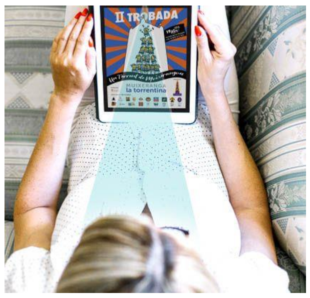
What information is obtained

EMOTIONAL PREFERENCES

• Which images generate greater activation, engagement and satisfaction.
• Which is the emotional profile generated by the brand throughout the corporative images, the packaging or the advertising.
• Testing the emotional impact of slogans.

ELEMENTS ANALYSIS

• Identification of the elements where the users focus more.
• Identification of the elements that cause the greatest degree of rejection.
• Analysis of the emotional impact of the stimulus in various segments of the sample (gender, age and others).



6. Attention and emotions analytics

Retail is the scenario in which brands have the opportunity to interact directly with the final consumer. It is concerned with the physical space in which consumers make their last purchase decision. Attention and emotion analytics give us the metrics we need to optimize our buying strategy. Given the large volume of stimuli to which the consumer is faced at the point of sale, it's essential to know what the visual impact of the products as well as the advertising elements associated with these spaces on the linears are**.** Non-intrusive techniques of emotion measurement help measure the emotional reaction of users. It is based on the products and advertising in real-time shopping environments**.** This implies a revolution in retail research since it avoids biases in the traditional research at the point of sale when registering the user's reaction to the linears in real-time. A simple built-in camera can analyze crowds and their spontaneous behaviors in any social context.

The facial coding technology of Emotion Research Lab measures the customer's reaction at the point of sale or showcase and translates it into data of the 6 basic emotions and the 4 emotional metrics**.** The software allows the researcher to obtain the knowledge of the number of people that have passed in front of the linears or areas of the point of sale. The platform reveals the number of people who are focusing on the product, as well as the time that they are showing attention. In addition, the software recognizes the sex and age of those who are paying attention to the linears or sales area.

What information is obtained

• The emotional impact of the products through metrics as activation, engagement, experience, satisfaction and the basic emotions with greater impact (happiness, surprise, anger, disgust, fear and sadness).
• The number of people walking in front of the sales areas and the proportion of active viewers.
• Sociodemographic profile of the viewers and the activities.

7. Artificial empathy: IoT

The Internet of Things (IoT) involves the interconnection of everyday objects with each other through the Internet. This opens the door to a greater interconnection between different types of objects and devices, but it also generates a more complex communication between people and everyday technological objects. Artificial empathy: IoT for humans.

The objects connected to the IoT can be classified into three groups depending on their function: those that function as sensors, those that are responsible for carrying out active actions and those that combine both functions. Currently, the IoT is already applied in household devices, home smart devices, robotics or vehicles. However, the challenge of this new technological horizon lies in achieving a more bidirectional connection, making objects able to empathize with human behavior.

What it is about

Artificial empathy is the application based on facial recognition technology. It allows technological devices to capture human emotions. A video camera is embedded in any object or device. It is possible to capture and store data of the 6 basic emotions and 4 metrics in real-time. The devices adapt to the states of mind of the users, performing actions or communicating according to the emotional information recorded. What information is obtained

• Basic sociodemographic data: sex and age.
• Basic emotions: happiness, anger, disgust, sadness, surprise and fear.
• Emotional metrics: activation, engagement satisfaction and experience.
• The emotional impact of the real environments through basic and secondary emotions.

Application

API and SDK available for Android, iOS, Linux and Windows

8. Political audience

The projected image of the political parties and their candidates is strongly conditioned by the audiovisual material distributed in the media and social networks. In a world increasingly dominated by audiovisual content, the tools of neuropolitics gain ground when it comes to addressing the impact of messages.

Neuropolitics study the behavior of citizens and political leaders using neuroscientific techniques to predict future electoral behavior and make political communications more efficient.
Emotions plan a key role in the political behavior, leaving aside rational processes and focusing on reactions such as sympathy, empathy or closeness, which ultimately decide the vote.

Political Audience is an Emotion Research Lab tool that allows knowing the emotional impact caused by videos or electoral posters as well as any audiovisual content generated in social networks**.** With it, you get to know what is the emotional impact on a given population sample to predict the success of a particular campaign.
Along with the emotional metrics, Political audience integrates eye-tracking technology that allows you to obtain heatmaps, where you can see which elements focus the attention of the audience.

What information is obtained

Audiovisual content analysis

• Mood generated by the stimulus.
• Levels of activation, engagement and satisfaction generated by the stimulus.
• Degree of emotional recall generated through the calculation of Relevance.
• Comparison of the impact generated by different versions of the same stimulus on the sample.
• Emotional reaction to certain messages and images identification of rejection points generated by the stimulus.
• Identification of rejection points generated by the stimulus.

Analysis of posters and images

• Testing the emotional impact of slogans.
• Identification of the elements that focus more attention on the poster.
• Identification of the elements that cause a greater degree of rejection.
• Analysis of the emotional impact of the image in different segments of the sample (sex, age and others).
• Comparison of the emotional impact between different versions of the electoral poster.

9. Cars

The Emotion AI applications for cars are endless: from drowsiness detection for driver safety, to identifying whether driver is attentive or distracted while driving, and to building a highly personalized and intimate in-cab experience. The ability to accurately detect emotions is being viewed as a significant enhancement to the human-machine interface in vehicles.

Being able to recognize those actions for drivers or passengers can help smart cars to better understand what's going on inside the car. Then based on those observations, control can pass between the smart car and driver - or triggers can be put in place, where an alarm goes off to the driver if he becomes drowsy before he drifts from his lane. These features not only can enhance the road safety but it will enhance the driving experience of people for next generations of family cars and public transportation. Affectiva is one of the pioneers' company in Driver emotion recognition and real-time Facial analysis for the Automotive Industry which has achieved significant advances. [20]

Startups pioneering the new field of emotional analytics [21]

1. Affectiva [22]

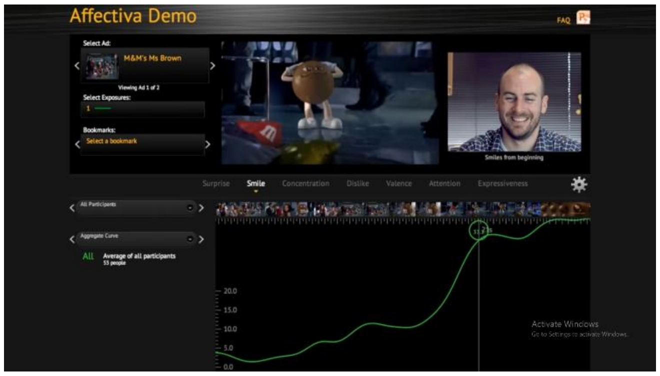
Website: Affectiva
Founded: 2009
Headquarters (and other locations): Waltham, Massachusetts
Amount raised (latest round): $33.72 million ($14 million Series D, May 2016)
Investors: WPP, National Science Foundation, Fenox Venture Capital, Horizons Ventures, Kleiner Perkins, Caufield & Byers (KPCB), Myrian Capital
Founders: CEO Rana el Kaliouby, Rosalind Picard

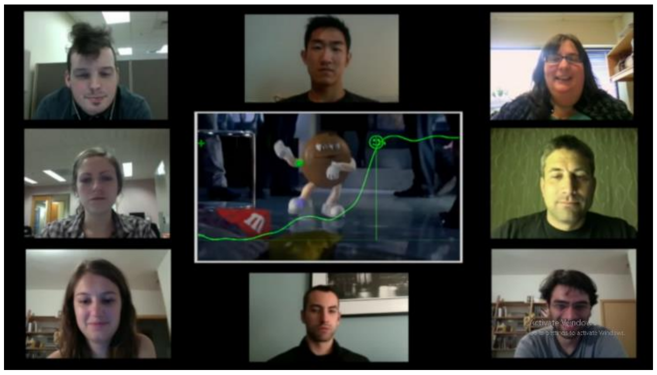
An MIT Media Lab spin-off, Affectiva calls itself the pioneer in Emotion AI. They use computer vision and a massive emotions database with 4 million separate facial expressions to track and summarize expressed emotions. The database includes samples from 75 countries, helping account for possible differences across cultures.

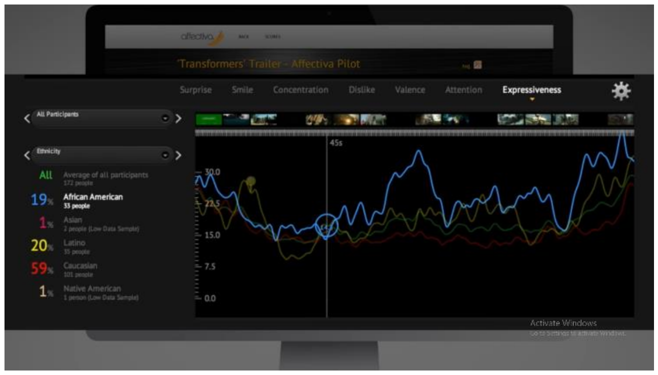
“We envision a future where our mobile and IoT devices can read and adapt to human emotions, transforming not only how we interact with hyper-intelligent technology, but also how we communicate with each other in a digital world,” Rana el Kaliouby said last year after their last fundraising round. “We build artificial emotional intelligence that senses, models and adapts to human emotion and behavior. It is a big, exciting vision for artificial intelligence, as it realizes the practical business application of AI and fuels innovation in many global markets.” [21]

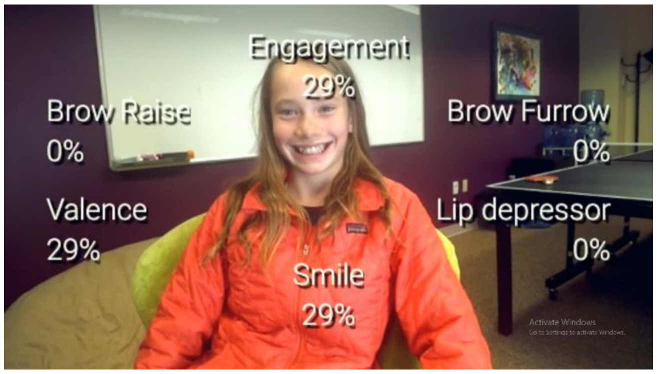
Affectiva Emotion Recognition Technology

Affectiva's emotion recognition technology allows developers to create hyper-personalized experiences across multiple industries such as gaming, advertising, healthcare and automotive. Game developers can create adaptive games that change based on a player's mood, clinical researchers can develop applications that respond to a patient's emotional state, and video communication platforms can optimize webinars so speakers can modify their presentations in real-time, based on an audience's engagement.









Affectiva future plan

According to affetiva manageres:” We envision a world where technology understands all things human. To achieve that, we’re pioneering Human Perception AI: software that can detect not only human emotions, but complex cognitive states, such as drowsiness and distraction. And, in the future, it will be able to understand human activities, interactions and objects people use.

Building AI that understands all things human is no small feat. But with our proven approaches to deep learning, computer vision and speech science, and the massive amounts of real-world people data we continue to collect and annotate, we’re well on our way. What’s more, by taking a multi-modal approach to human perception -- analyzing both facial and vocal expressions -- we’re able to get a more comprehensive and accurate understanding of human states.

It’s easy to get wrapped up in what the future of AI holds, but we need to remember the motivation behind AI: people, and the potential to improve the way we work, live and interact with one another. It’s critically important to take a human-centric approach to AI, and at Affectiva, we believe that Human Perception AI will help to bring the focus of AI back to the people who will benefit from it. Building AI that can understand us for what makes us human is the only way we’ll begin to trust in the technology.

We’re excited to embark on this next chapter, building on what’s already possible with Emotion AI, and taking it a step further to replicate the power of human perception in artificial intelligence. But there’s still much to be explored, developed, and discussed. Specifically, there are important considerations to address for the ethical development and deployment of AI systems -- especially those designed to interact with humans. Ethical AI hinges on diversity -- of teams, data, use cases and more -- but that’s a larger conversation for another blog post.”

Some of effetiva’s achievements

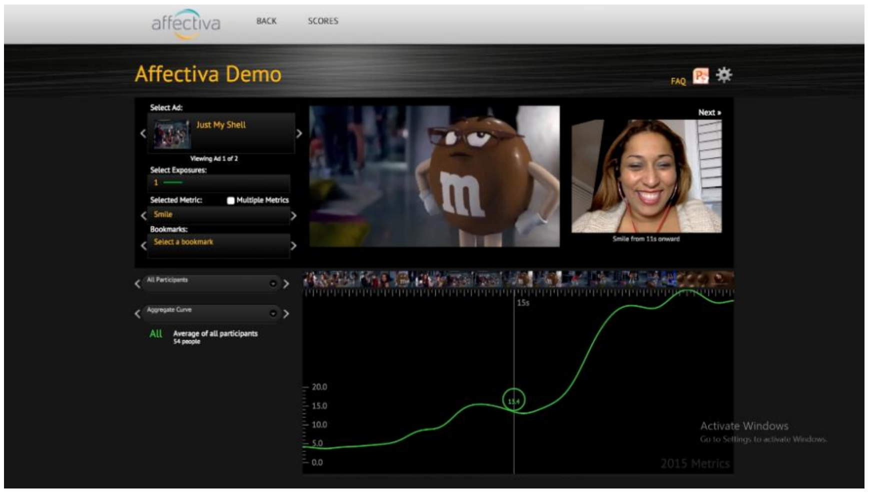
Affective Automative AI

While AI in automotive has historically focused on what’s going on outside of the vehicle, OEMs and Tier 1 suppliers are beginning to turn cameras and sensors inwards, using AI to understand all things human within a vehicle. New regulations such as the European New Car Assessment Programme (Euro NCAP) are requiring next-generation safety features including advanced driver state monitoring, child presence detection and more. And looking ahead to the future of mobility, consumers will shift away from assessing a vehicle’s driving performance to focusing on the best in-cabin experience. Affectiva Automotive AI powers in-cabin sensing systems that perceive all things human inside of a vehicle. OEMs, Tier 1 suppliers, ridesharing providers and fleet management companies are using Affectiva’s patented technology to understand drivers’ and passengers’ states and moods, in order to address critical safety concerns and deliver enhanced in-cabin experiences. Affectiva Automotive AI unobtrusively measures, in real time, complex and nuanced emotional and cognitive states from face and voice. The technology will scale with evolving safety standards and future of mobility needs, with the potential to perceive passengers, activities, interactions, and all facets of the human experience inside of a vehicle.

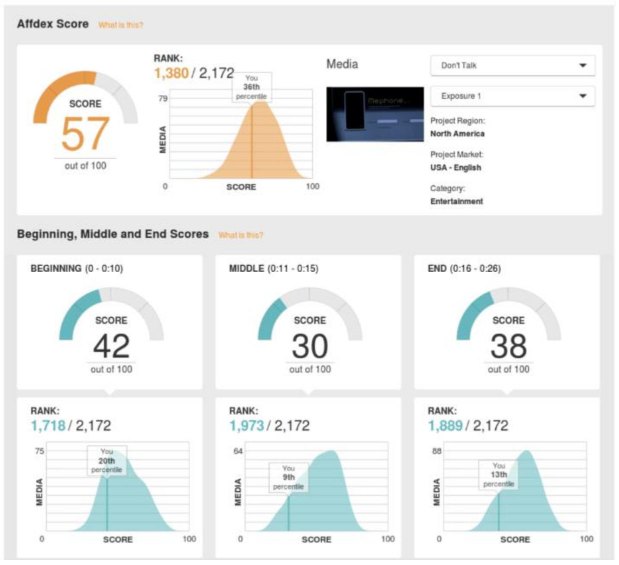
Affetiva media analytics

Brands, advertisers and market researchers know that emotions influence consumer behavior. Understanding consumer emotional engagement with content and experiences, is key to creating the best marketing campaigns and optimizing media spend. Gathering quality emotion data is critical as it predicts key success metrics such as brand recall, sales lift, purchase intent and likelihood to share. Affectiva Media Analytics solution is built using our industry-leading Emotion AI. Commonly known as “facial coding” in the marketing industry, our solution provides deep insight into unfiltered and unbiased consumer emotional responses to brand content. Based on this, brand owners can understand if their campaigns are working as intended, what makes them sing and how to create more effective edits.

Affetiva is a technology provider to the marketing and content industry. It's AI is a critical component of your research methodology and easy to deploy. All your research panelists need is internet connectivity and a standard web camera – it's simple, easy and highly accurate. As viewers watch your stimulus, It measures their moment-by-moment facial expressions of emotion. The results are aggregated and displayed in an easy to use dashboard.



Affetiva's solution also provides norms that tie directly to outcomes such as brand recall, sales lift, purchase intent and virality. These norms help benchmark how your ads perform compared to those of your competitors – by geography, product category, media length and on repeat view. Unparalleled in the industry, these norms are built on the world's largest emotion database.



Affdex for Market Research – Norms

Leading marketing insights firms like Kantar, MediaCom and Unruly, and 1/4 of the Fortune Global 500, including 1,400 brands like Mars, Kellogg's and CBS, by using Affectiva's Media Analytics solution to optimize content and media spend.

- Measures 7 emotions and 20 facial expressions; filter by demo and survey data.
- Seamlessly integrates with your survey and research methodology; works with any panel provider.
- Easy to use SaaS solution; all you need is a standard webcam and internet connectivity.
- Norms to benchmark the effectiveness of your content by geography, product category, media length and repeat view.
- Built using the world's largest emotion database of 40,000 ads and more than 7.7 million faces analyzed in 87 countries.

"Facial coding is a powerful way to understand consumers' unfiltered responses, and Affectiva's market leading solution enables us to understand a wide range of emotional responses to marketing content." – Graham Page, Managing Director of Offer and Innovation, Kantar Millward Brown

Uses

1. Improve story flow
Moment-by-moment emotion data can pinpoint viewer confusion and lack of engagement. This insight helps improve the story arch of an animatic or ad.

2. Create cut-downs
Identify the most emotionally engaging moments in longer TV ads so you can retain the most impactful parts when cutting down to shorter online ads.

3. TV show character analysis
Assess audience engagement with characters on TV shows and the interplay of characters – especially important when introducing new actors to a show.

4. Movie trailer creation
Movie studios use emotion analytics to test different versions of a movie trailer, and use the most engaging moments in each to create the final cut.

5. Determine media spend
Test final ads for emotional engagement to identify potential wear out. Direct your advertising dollars to the ads with the best emotional impact on repeat view.

6. Test voice overs and brand reveal

Use emotion data to see if your audiences are emotionally engaged at the moment of brand reveal in an ad, and test the effectiveness of taglines and voice-overs.

2. Beyond Verbal

Website: Beyond Verbal
Founded: 2012
Headquarters (and other locations): Tel Aviv, Israel
Amount raised (latest round): $10 million ($3 million, September 2016)
Investors: Kuang-Chi Science, Singulariteam, Omninet Capital, Winnovation
Founders: SVP BizDev Yoav Hoshen, Yuval Mor

Beyond Verbal claims to be one of the only companies in the world giving literal emotional feedback in terms of analytics, but unlike other members of this list they aren't reliant on facial recognition. They track voice patterns, keeping track of lonely family members and finding ways to apply the technology to the dating world by matching people according to attitudes and moods. They have pivoted from focusing on marketing to healthcare, pioneering a study with the Mayo Clinic to **link voice patterns to progressive heart disease**.

They have demonstrated their emolytic capabilities by analyzing Donald Trump during last year's Republican primary debates and his combative exchanges with people such as former Fox News host Megyn Kelly.

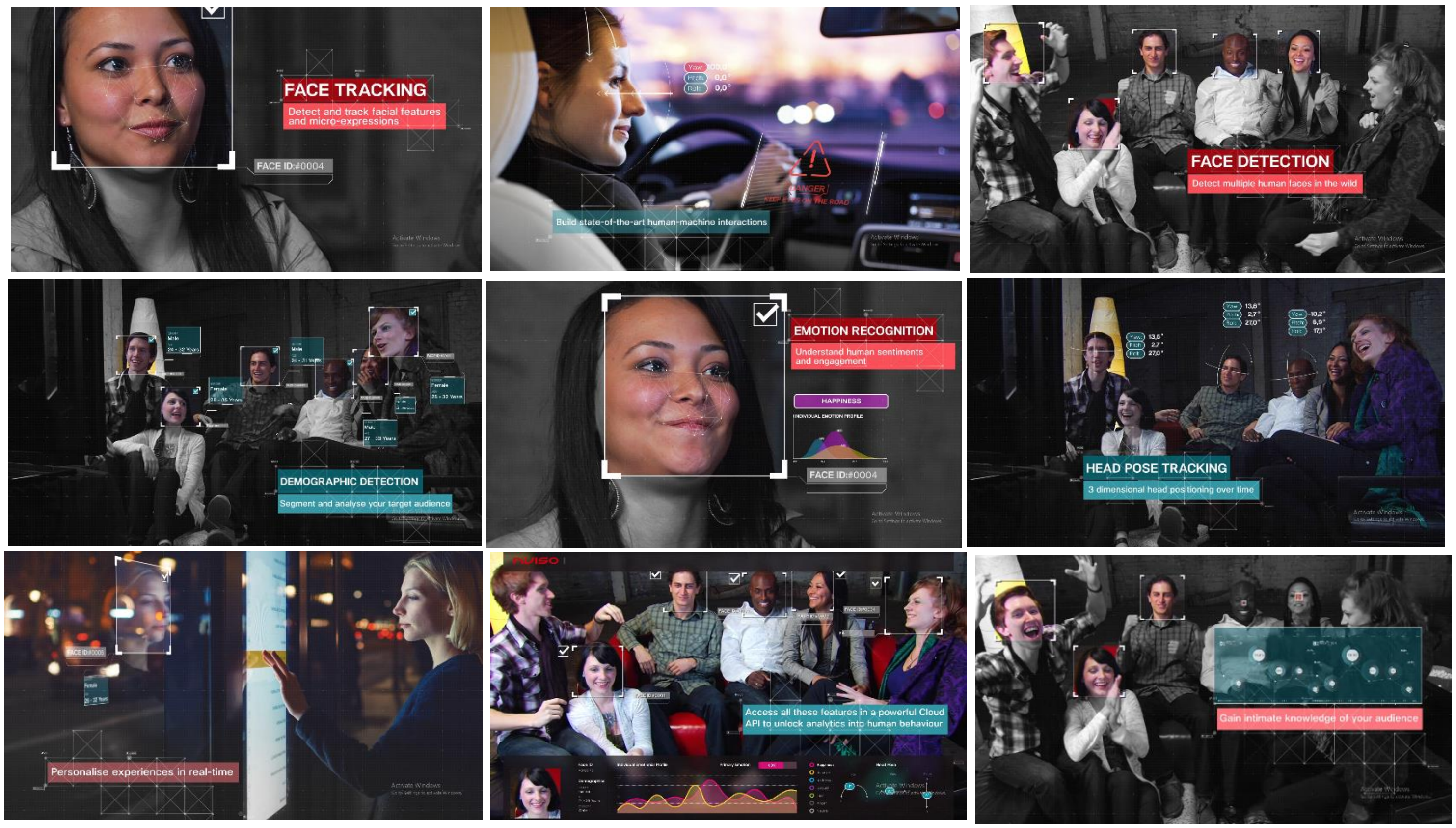
This company, perhaps more than others, is producing technology that might have a massive impact on the way voice assistants and spoken translation technologies work in the future, choosing to respond to statements or questions with specific tones or translations that reflect the nuance of a bad (or good) mood.

FOX NEWS
FOX NEWS DEBATE
Tough and substantive communication, arrogance, stubbornness.
Arrogance and condescention.

3. nViso [23]

Website: nViso
Founded: 2005
Headquarters (and other locations): Lausanne, Switzerland
Founders: CEO Tim Llewellynn

nViso boasts what it calls "the most scalable, robust, and accurate artificial intelligence solutions" for instant emotional analysis of consumers on the market, focusing their business on major brands and advertising. They say they have original 3D facial imaging technology that works with "ordinary webcams" to gauge consumers' responses online.


FACE TRACKING
Detect and track facial features and micro-expressions
FACE ID:#0004
Build state-of-the-art human-machine interactions
DANGER
FACE DETECTION
Detect multiple human faces in the wild
DEMOGRAPHIC DETECTION
Segment and analyse your target audience
EMOTION RECOGNITION
Understand human sentiments and engagement
HAPPINESS
FACE ID:#0004
HEAD POSE TRACKING
3 dimensional head positioning over time
Personalise experiences in real-time
nViso
Access all these features in a powerful Cloud API to unlock analytics into human behaviour
Gain intimate knowledge of your audience

NVISO's visual intelligence technology uses computers to learn from examples opposed to being manually programmed. Using deep Convolutional Neural Networks (CNNs) and state-of-the-art machine learning to understand human behaviors depicted in images and videos, it can achieve accuracy levels that surpass human performance in many narrowly defined tasks.

NVISO uses large training datasets containing millions of samples with proprietary software to detect, analyze and interpret behaviors from different learning scenarios (emotions, face identification, semantics, etc.). By continuously improving their datasets, algorithm training, and benchmarking capabilities they constantly expand visual intelligence applications across industries.

NVSIO's application in Health care:

Artificial intelligence (AI) systems, blending data and advanced algorithms to mimic the cognitive functions of the human mind, have begun to simplify and enhance even the simplest aspects of our everyday experiences – and the healthcare industry is no exception.

Modern AI has the ability to transform the healthcare industry by analyzing vast amounts of data with incredible accuracy. New tools powered by AI, empowers healthcare professionals to focus on their patients instead of data and technology.

Deep learning for pain and patient management:

Pain monitoring

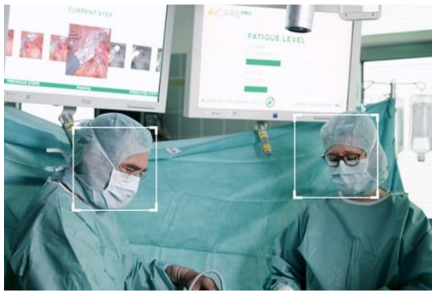
AI has the ability to transform pain management for vulnerable patients with innovative facial muscle movement analytics. Enables accurate pain assessment of patients with communication difficulties such as dementia or with pre-verbal children.

Patient monitoring

Highly accurate biometrics analysis helps in the identification of individuals by precisely recognizing their unique characteristics to ensure patient security. Reliably identify patients and caregivers at the bedside and in the home environment.

Patient profiling

Machine learning enables patient emotional profiling to address various challenges in communication and medical procedures. The applications range from identifying patients reactions to verbal advise, to real-time monitoring in medical operations.



4. CrowdEmotion

Website: CrowdEmotion, @CrowdEmotion
Headquarters (and other locations): London, England
Amount raised (latest round): Unknown
Founders: CEO Matthew Celuszak, Daniel Jabry, CTO Diego Caravana

CrowdEmotion is a British company whose platform for facial expression provides insights on emotional expression. Their other platform, MeMo, can be used for two-way video chat and self-analysis to better help with personal engagement tactics in business or for your own personal growth. They have been working with the BBC for the last three years to measure TV audience engagement.

CEO Matthew Celuszak expressed confidence in his technology’s ability to affect the way that media engages with its audiences when they announced that partnership in 2014.

“With today’s media noisier than ever, we’re here to innovate, bring emotions to life and reshape broadcast media through our findings.”

5. iMotions [24]

Website: iMotions, @iMotionsGlobal
Founded: 2005
Headquarters (and other locations): Copenhagen, Denmark (Boston, Massachusetts)
Amount raised (latest round): $4.3 million ($2.7 million, 2007)
Investors: Inventure Capital, Syddansk Venture, The Way Forward Aps
Founders: CEO Peter Hartzbech

The company focuses on supporting research teams at universities and within larger companies, saying their software supports more than 50 market-available biosensors and eye sensors that will help collect data and produce observations with their machine learning technology. Nielsen, Deloitte, P&G, and Harvard University are listed as customers on the company website. Their team has also worked with Stanford to track eye movements by drivers in what they say is one of the most advanced driving simulators in the world.

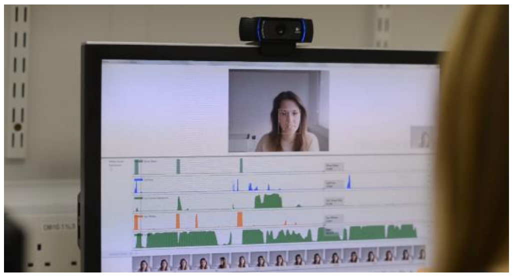
They also announced a partnership back in 2015 with previous list entry Affectiva to better integrate emotional analytics with biometric research with iMotions Founder & CEO Peter Hartzbech saying at the time, “Measuring unfiltered and unbiased emotional responses is key to understanding human behavior in consumer engagement and user experience.”

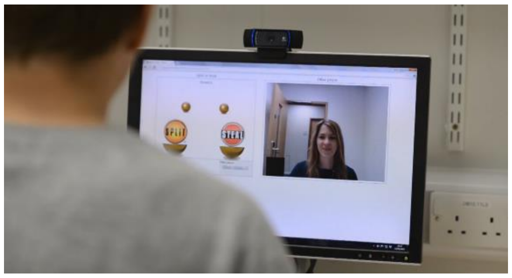
iMotions is high tech software made to execute human behavior research with high validity. iMotions seamlessly integrates multiple biosensors that provide different insight; such as Eye Tracking, EDA/GSR, EEG, ECG and Facial Expression Analysis. The combination of different sensors and data sources allows you to make a clearer and more incisive understanding of human behavior. Through real-time measurements of nonconcious responses, iMotions provides the bigger picture on human actions, thoughts and feelings for you to tap into new innovation – getting you the results faster.

- Hardware agnostic software with +50 biosensor devices
- Seamless integration of even your own research data
- Full scalability when you need it
- Accelerated output for validated and fast results

Some of iMotions applications

- iMotions helps Oxford University address entirely new areas of research into emotions. By utilizing an objective and automated system for measurements of facial expressions, Dr. Danielle Shore is able to rapidly explore emotional reactions in game theory contexts.
- iMotions helps HeyHuman think differently about brands and marketing. By measuring people's behavior during advertising testing, UX studies, concept development and more, Neuroscience Consultant Aoife McGuinness is able to deliver more nuanced insights and create more engaging content.
- Professor Marco Palma and the Human Behavior Lab at Texas A&M University is pioneering research in neuroeconomics with iMotions, scaling experiments and executing research faster than ever.
- Crispin Haywood and the Shopper Science Lab at GSK wanted to measure consumer behavior in realistic physical shopping environments. Now they can – with eye tracking and other biosensors in iMotions – in order to surface insights of what works and what doesn't in retail.





6. Realeyes

Website: Realeyes
Founded: 2007
Headquarters (and other locations): London, England
Amount raised (latest round): $9.2 million ($2.5 million debt financing, 2015)
Investors: Entrepreneurs Fund (EF), SmartCap AS, European Commission, European Regional Development Fund
Founders: Martin Salo, Elnar Hajiyev, CEO Mihkel Jäätma

Realeyes gets people to share access to their personal webcams, from which they can use their own computer vision algorithms and machine learning to track expressions during broadcasts, advertisements, and other media. Taking this visual feedback, they can help content creators to learn from their mistakes and put together videos that audiences will want to engage with. They offer 24-hour report turnarounds for $3,500, according to their website. It works by dragging and dropping a given video into the Realeyes dashboard and defining audience segments and can analyze up to 300 people at a time.

Technology [25]

Using computer vision and machine learning, we measure how people feel as they view your content online through their webcam.

Drag, Drop, Done

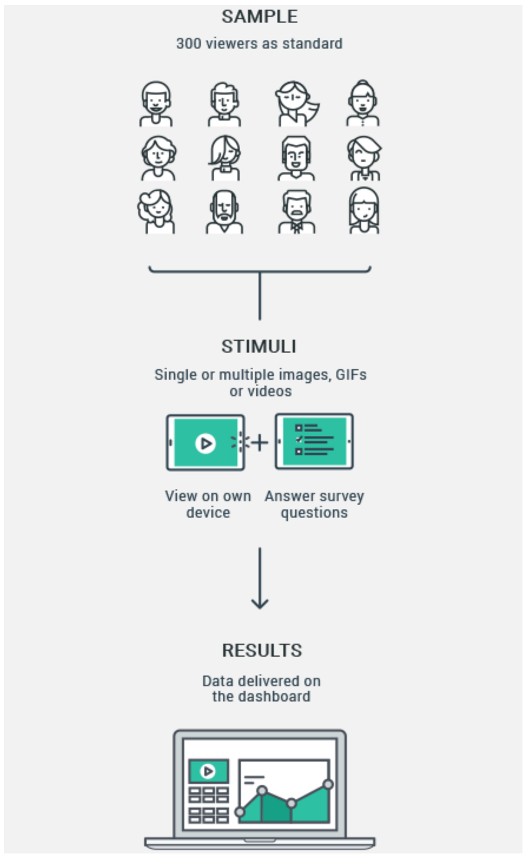
Upload your own or competitor images, GIFs or videos at any stage of its life-cycle. We source a sample audience between 150 to 300 viewers based on your objectives.

Real Responses

We can measure the attention, emotions and sentiment of your sample audience as they watch your content on their own device at home. You can even ask brand lift survey questions too.

Real Results, Fast

Delivered to the dashboard within 3 hours (with Realeyes GO) - inform your campaign strategy at speed by seeing how audiences responded to your creative.



Project development [26]

Time limitation was an impediment for developing my project as I wanted.

The current version shows the possibility of designing an API for emotion recognition. But it needs extra endpoints to be fully useful.

Some possible API that can be added to the project for making it more effective are mentioned bellow.

Microsoft Cognitive services

A comprehensive family of AI services and cognitive APIs to help you build intelligent apps.

Face API - V1.0

This API is currently available in:

- Australia East - australiaeast.api.cognitive.microsoft.com
- Brazil South - brazilsouth.api.cognitive.microsoft.com
- Canada Central - canadacentral.api.cognitive.microsoft.com
- Central India - centralindia.api.cognitive.microsoft.com
- Central US - centralus.api.cognitive.microsoft.com
- East Asia - eastasia.api.cognitive.microsoft.com
- East US - eastus.api.cognitive.microsoft.com
- East US 2 - eastus2.api.cognitive.microsoft.com
- France Central - francecentral.api.cognitive.microsoft.com
- Japan East - japaneast.api.cognitive.microsoft.com
- Japan West - japanwest.api.cognitive.microsoft.com
- Korea Central - koreacentral.api.cognitive.microsoft.com
- North Central US - northcentralus.api.cognitive.microsoft.com
- North Europe - northeurope.api.cognitive.microsoft.com
- South Africa North - southafricanorth.api.cognitive.microsoft.com
- South Central US - southcentralus.api.cognitive.microsoft.com
- Southeast Asia - southeastasia.api.cognitive.microsoft.com
- UK South - uksouth.api.cognitive.microsoft.com
- West Central US - westcentralus.api.cognitive.microsoft.com
- West Europe - westeurope.api.cognitive.microsoft.com
- West US - westus.api.cognitive.microsoft.com
- West US 2 - westus2.api.cognitive.microsoft.com

Face - Detect

Detect human faces in an image, return face rectangles, and optionally with faceIds, landmarks, and attributes.

- No image will be stored. Only the extracted face feature(s) will be stored on server. The faceId is an identifier of the face feature and will be used in **Face - Identify**, **Face - Verify**, and **Face - Find Similar**. The stored face features will expire and be deleted 24 hours after the original detection call.
- Optional parameters include faceId, landmarks, and attributes. Attributes include age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure and noise. Some of the results returned for specific attributes may not be highly accurate.
- JPEG, PNG, GIF (the first frame), and BMP format are supported. The allowed image file size is from 1KB to 6MB.
- The minimum detectable face size is 36x36 pixels in an image no larger than 1920x1080 pixels. Images with dimensions higher than 1920x1080 pixels will need a proportionally larger minimum face size.
- Up to 100 faces can be returned for an image. Faces are ranked by face rectangle size from large to small.
- For optimal results when querying **Face - Identify, Face - Verify** and **Face - Find Similar** ('returnFaceId' is true), use faces that are: frontal, clear, and with a minimum size of 200x200 pixels (100 pixels between eyes).
- Different 'detectionModel' values can be provided. To use and compare different detection models.
    - 'detection_01': The default detection model for Face - Detect. Recommend for near frontal face detection. For scenarios with exceptionally large angle (head-pose) faces, occluded faces or wrong image orientation, the faces in such cases may not be detected.
    - 'detection_02': Detection model released in 2019 May with improved accuracy especially on small, side and blurry faces. Face attributes and landmarks are disabled if you choose this detection model.
- Different 'recognitionModel' values are provided. If follow-up operations like Verify, Identify, Find Similar are needed, specify the recognition model with 'recognitionModel' parameter. The default value for 'recognitionModel' is 'recognition_01', if latest model needed, explicitly specify the model you need in this parameter. Once specified, the detected faceIds will be associated with the specified recognition model. More details, please refer to **(How to specify a recognition model**)
    - 'recognition_01': The default recognition model for **Face - Detect**. All those faceIds created before 2019 March are bonded with this recognition model.
    - 'recognition_02': Recognition model released in 2019 March. 'recognition_02' is recommended since its overall accuracy is improved compared with 'recognition_01'.

Http Method

POST

We should select the testing console in the region where we created our resources. (Notice that the region which have written below are link)

West US
West US 2
East US
East US 2
West Central US
South Central US
West Europe
North Europe
South East Asia
East Asia
Australia East
Brazil South
Canada Central
Central India
UK South
Japan East
Central US
France Central
Korea Central
Japan West
North Central US
South Africa North
UAE North

Request URL

https://{endpoint}/face/v1.0/detect[?returnFaceId][&returnFaceLandmarks][&returnFaceAttributes][&recognitionModel][&returnRecognitionModel][&detectionModel]

Request parameters

returnFaceId (optional)	boolean	Return faceIds of the detected faces or not. The default value is true.
returnFaceLandmarks (optional)	boolean	Return face landmarks of the detected faces or not. The default value is false.
returnFaceAttributes (optional)	string	Analyze and return the one or more specified face attributes in the comma-separated string like "returnFaceAttributes=age,gender". Supported face attributes include age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure and noise. Face attribute analysis has additional computational and time cost.
recognitionModel (optional)	string	The 'recognitionModel' associated with the detected faceIds. Supported 'recognitionModel' values include "recognition_01" or "recognition_02". The default value is "recognition_01". "recognition_02" is recommended since its overall accuracy is improved compared with "recognition_01".
returnRecognitionModel (optional)	boolean	Return 'recognitionModel' or not. The default value is false.
detectionModel (optional)	string	The 'detectionModel' associated with the detected faceIds. Supported 'detectionModel' values include "detection_01" or "detection_02". The default value is "detection_01".

Request headers

Content-Type	string	Media type of the body sent to the API.
Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API.

Request body

To detect in a URL (or binary data) specified image.

JSON fields in the request body:

Fields	Type	Description
url	String	URL if input image

Application/json	Application/octet-stream
{ "url": "http://example.com/1.jpg" }	[binary data]

Response 200

A successful call returns an array of face entries ranked by face rectangle size in descending order. An empty response indicates no faces detected. A face entry may contain the following values depending on input parameters:

Fields	Type	Description
faceId	String	Unique faceId of the detected face, created by detection API and it will expire 24 hours after the detection call. To return this, it requires "returnFaceId" parameter to be true.
recognitionModel	String	The 'recognitionModel' associated with this faceId. This is only returned when 'returnRecognitionModel' is explicitly set as true.
faceRectangle	Object	A rectangle area for the face location on image.
faceLandmarks	Object	An array of 27-point face landmarks pointing to the important positions of face components. To return this, it requires "returnFaceLandmarks" parameter to be true.
faceAttributes	Object	Face Attributes: • age: an estimated "visual age" number in years. It is how old a person looks like rather than the actual biological age. • gender: male or female. • smile: smile intensity, a number between [0,1]. • facialHair: return lengths in three facial hair areas: moustache, beard and sideburns. The length is a number between [0,1]. 0 for no facial hair in this area, 1 for long or very thick facial hairs in this area. • headPose: 3-D roll/yaw/pitch angles for face direction. • glasses: glasses type. Values include 'NoGlasses', 'ReadingGlasses', 'Sunglasses', 'SwimmingGoggles'. • emotion: emotion intensity, including neutral, anger, contempt, disgust, fear, happiness, sadness and surprise. • hair: group of hair values indicating whether the hair is visible, bald, and hair color if hair is visible. • makeup: whether eye, lip areas are made-up or not. • accessories: accessories around face, including 'headwear', 'glasses' and 'mask'. Empty array means no accessories detected. Note this is after a face is detected. Large mask could result in no face to be detected. • blur: face is blurry or not. Level returns 'Low', 'Medium' or 'High'. Value returns a number between [0,1], the larger the blurrier. • exposure: face exposure level. Level returns 'GoodExposure', 'OverExposure' or 'UnderExposure'. • noise: noise level of face pixels. Level returns 'Low', 'Medium' and 'High'. Value returns a number between [0,1], the larger the noisier

Application/json

```
[
    {
        "faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
        "recognitionModel": "recognition_02",
        "faceRectangle": {
            "width": 78,
            "height": 78,
            "left": 394,
            "top": 54
        },
        "faceLandmarks": {
            "pupilLeft": {
                "x": 412.7,
                "y": 78.4
            },
            "pupilRight": {
                "x": 446.8,
                "y": 74.2
            },
            "noseTip": {
                "x": 437.7,
                "y": 92.4
            },
            "mouthLeft": {
                "x": 417.8,
                "y": 114.4
            },
            "mouthRight": {
                "x": 451.3,
                "y": 109.3
            },
            "eyebrowLeftOuter": {
                "x": 397.9,
                "y": 78.5
            },
            "eyebrowLeftInner": {
                "x": 425.4,
                "y": 70.5
            },
            "eyeLeftOuter": {
                "x": 406.7,
                "y": 80.6
            },
            "eyeLeftTop": {
                "x": 412.2,
                "y": 76.2
            },
            "eyeLeftBottom": {
                "x": 413.0,
                "y": 80.1
            },
            "eyeLeftInner": {
                "x": 418.9,
                "y": 78.0
            },
            "eyebrowRightInner": {
                "x": 4.8,
                "y": 69.7
            },
            "eyebrowRightOuter": {
                "x": 5.5,
                "y": 68.5
            },
            "eyeRightInner": {
                "x": 441.5,
                "y": 75.0
            },
            "eyeRightTop": {
                "x": 446.4,
                "y": 71.7
            },
            "eyeRightBottom": {
                "x": 447.0,
                "y": 75.3
            },
            "eyeRightOuter": {
                "x": 451.7,
                "y": 73.4
            },
            "noseRootLeft": {
                "x": 428.0,
                "y": 77.1
            },
            "noseRootRight": {
                "x": 435.8,
                "y": 75.6
            },
            "noseLeftAlarTop": {
                "x": 428.3,
                "y": 89.7
            },
            "noseRightAlarTop": {
                "x": 442.2,
                "y": 87.0
            },
            "noseLeftAlarOutTip": {
                "x": 424.3,
                "y": 96.4
            },
            "noseRightAlarOutTip": {
                "x": 446.6,
                "y": 92.5
            },
            "upperLipTop": {
                "x": 437.6,
                "y": 105.9
            },
            "upperLipBottom": {
                "x": 437.6,
                "y": 108.2
            },
            "underLipTop": {
                "x": 436.8,
                "y": 111.4
            },
            "underLipBottom": {
                "x": 437.3,
                "y": 114.5
            }
        },
        "faceAttributes": {
            "age": 71.0,
            "gender": "male",
            "smile": 0.88,
            "facialHair": {
                "moustache": 0.8,
                "beard": 0.1,
                "sideburns": 0.02
            },
            "glasses": "sunglasses",
            "headPose": {
                "roll": 2.1,
                "yaw": 3,
                "pitch": 1.6
            },
            "emotion": {
                "anger": 0.575,
                "contempt": 0,
                "disgust": 0.006,
                "fear": 0.008,
                "happiness": 0.394,
                "neutral": 0.013,
                "sadness": 0,
                "surprise": 0.004
            },
            "hair": {
                "bald": 0.0,
                "invisible": false,
                "hairColor": [
                    {"color": "brown", "confidence": 1.0},
                    {"color": "blond", "confidence": 0.88},
                    {"color": "black", "confidence": 0.48},
                    {"color": "other", "confidence": 0.11},
                {"color": "gray", "confidence": 0.07},
                {"color": "red", "confidence": 0.03}
            ]
        },
        "makeup": {
            "eyeMakeup": true,
            "lipMakeup": false
        },
        "occlusion": {
            "foreheadOccluded": false,
            "eyeOccluded": false,
            "mouthOccluded": false
        },
        "accessories": [
            {"type": "headWear", "confidence": 0.99},
            {"type": "glasses", "confidence": 1.0},
            {"type": "mask"," confidence": 0.87}
        ],
        "blur": {
            "blurLevel": "Medium",
            "value": 0.51
        },
        "exposure": {
            "exposureLevel": "GoodExposure",
            "value": 0.55
        },
        "noise": {
            "noiseLevel": "Low",
            "value": 0.12
        }
      }
    }
]
```

Response 400

Error code and message returned in JSON:

Response 400

Error code and message returned in JSON:

Error Code	Error Message Description
BadArgument	JSON parsing error. Bad or unrecognizable request JSON body.
BadArgument	Invalid argument returnFaceAttributes. Supported values are: age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure and noise in a comma-separated format.
BadArgument	'recognitionModel' is invalid.
BadArgument	'detectionModel' is invalid.
BadArgument	'returnFaceAttributes' is not supported by detection_02.
BadArgument	'returnLandmarks' is not supported by detection_02.
InvalidURL	Invalid image format or URL. Supported formats include JPEG, PNG, GIF(the first frame) and BMP.
InvalidURL	Failed to download image from the specified URL. Remote server error returned.
InvalidImage	Decoding error, image format unsupported.
InvalidImageSize	Image size is too small. The valid image file size should be larger than or equal to 1KB.
InvalidImageSize	Image size is too big. The valid image file size should be no larger than 6MB.

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Request body is invalid."
    }
}
```

Response 401

Error code and message returned in JSON:

Error code	Error Message Description
Unspecified	Invalid subscription Key or user/plan is blocked.

Application/json
{ "error": { "code": "Unspecified", "message": "Access denied due to invalid subscription key. Make sure you are subscribed to an API you are trying to call and provide the right key." } }

```
{
    "error": {
        "code": "Unspecified",
        "message": "Access denied due to invalid subscription key. Make sure you are subscribed to an
API you are trying to call and provide the right key."
    }
}
```

Response 403

Application/json

```
{
    "error": {
        "statusCode": 403,
        "message": "Out of call volume quota. Quota will be replenished in 2 days."
    }
}
```

Response 408

Operation exceeds maximum execution time.

Application/json

```
{
    "error": {
        "code": "OperationTimeOut",
        "message": "Request Timeout."
    }
}
```

Response 415

Unsupported media type error. Content-Type is not in the allowed types:

1. For an image URL, Content-Type should be application/json
2. For a local image, Content-Type should be application/octet-stream

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Invalid Media Type."
    }
}
```

Response 429

Application/json
{ "error": { "statusCode": 429, "message": "Rate limit is exceeded. Try again in 26 seconds." } }

```
{
    "error": {
       "statusCode": 429,
        "message": "Rate limit is exceeded. Try again in 26 seconds."
    }
}
```

Code samples

Python

```
########### Python 3.2 #############
import http.client, urllib.request, urllib.parse, urllib.error, base64

headers = {
    # Request headers
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': '{subscription key}',
}

params = urllib.parse.urlencode({
    # Request parameters
    'returnFaceId': 'true',
    'returnFaceLandmarks': 'false',
    'returnFaceAttributes': '{string}',
    'recognitionModel': 'recognition_01',
    'returnRecognitionModel': 'false',
    'detectionModel': 'detection_01',
})

try:
    conn =
http.client.HTTPSConnection('westus.api.cognitive.microsoft.co
m')
    conn.request("POST", "/face/v1.0/detect?%s" % params,
"{body}", headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################
```

Curl

```
curl -v -X POST
"https://westus.api.cognitive.microsoft.com/fac
e/v1.0/detect?returnFaceId=true&returnFaceL
andmarks=false&returnFaceAttributes={string}
&recognitionModel=recognition_01&returnRec
ognitionModel=false&detectionModel=detectio
n_01"
-H "Content-Type: application/json"
-H "Ocp-Apim-Subscription-Key: {subscription
key}"

--data-ascii "{body}"
```

Find-Similar

Given query face's faceId, to search the similar-looking faces from a faceId array, a face list or a large face list. faceId array contains the faces created by **Face - Detect,** which ???will expire 24 hours after creation. A "faceListId" is created by **FaceList - Create** containing persistedFaceIds that will not expire. And a "largeFaceListId" is created by **LargeFaceList - Create** containing persistedFaceIds that will also not expire. Depending on the input the returned similar faces list contains faceIds or persistedFaceIds ranked by similarity.

Find similar has two working modes, "matchPerson" and "matchFace". "matchPerson" is the default mode that it tries to find faces of the same person as possible by using internal same-person thresholds. It is useful to find a known person's other photos. Note that an empty list will be returned if no faces pass the internal thresholds. "matchFace" mode ignores same-person thresholds and returns ranked similar faces anyway, even the similarity is low. It can be used in the cases like searching celebrity-looking faces.

The 'recognitionModel' associated with the query face's faceId should be the same as the 'recognitionModel' used by the target faceId array, face list or large face list.

Http Method

POST

We should select the testing console in the region where we created our resources. (Notice that the region which have written below are link)

West US
West US 2
East US
East US 2
West Central US
South Central US
West Europe
North Europe
South East Asia
East Asia
Australia East
Brazil South
Canada Central
Central India
UK South
Japan East
Central US
France Central
Korea Central
Japan West
North Central US
South Africa North
UAE North

Request URL

https://{endpoint}/face/v1.0/findsimilars

Request headers

Content-Type	string	Media type of the body sent to the API.
Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API. Found in your **Cognitive Services accounts.**

Request body

JSON fields in request body:

Fields	**Type**	**Description**
faceId	String	faceId of the query face. User needs to **call Face - Detect** first to get a valid faceId. Note that this faceId is not persisted and will expire 24 hours after the detection call.
faceListId	String	An existing user-specified unique candidate face list, created in **FaceList - Create**. Face list contains a set of persistedFaceIds which are persisted and will never expire. Parameter faceListId, largeFaceListId and faceIds should not be provided at the same time.
largeFaceListId	String	An existing user-specified unique candidate large face list, created in **LargeFaceList - Create**. Large face list contains a set of persistedFaceIds which are persisted and will never expire. Parameter faceListId, largeFaceListId and faceIds should not be provided at the same time.
faceIds	Array	An array of candidate faceIds. All of them are created by **Face - Detect** and the faceIds will expire 24 hours after the detection call. The number of faceIds is limited to 1000. Parameter faceListId, largeFaceListId and faceIds should not be provided at the same time.
maxNumOfCandidatesReturned (optional)	Number	Optional parameter. The number of top similar faces returned. The valid range is [1, 1000].It defaults to 20.
mode (optional)	String	Optional parameter. Similar face searching mode. It can be "matchPerson" or "matchFace". It defaults to "matchPerson".

Application/json

```
{
    "faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
    "largeFaceListId": "sample_list",
    "maxNumOfCandidatesReturned": 10,
    "mode": "matchPerson"
}
```

Response 200

A successful call returns an array of the most similar faces represented in faceId if the input parameter is faceIds or persistedFaceId if the input parameter is faceListId or largeFaceListId.

JSON fields in response body:

Fields	Type	Description
persistedFaceId	String	persistedFaceId of candidate face when find by faceListId or largeFaceListId. persistedFaceId in face list/large face list is persisted and will not expire. As showed in below response.
faceId	String	faceId of candidate face when find by faceIds. faceId is created by **Face - Detect** and will expire 24 hours after the detection call.
confidence	Number	Similarity confidence of the candidate face. The higher confidence, the more similar. Range between [0,1].

Application/json

```
[
   {
      "persistedFaceId" : "015839fb-fbd9-4f79-ace9-7675fc2f1dd9",
      "confidence" : 0.82
   },
   ...
]
```

Response 400

Error code and message returned in JSON:

Error Code	Error Message Description
BadArgument	Invalid request body.
BadArgument	Mode is invalid.
BadArgument	The argument maxNumOfCandidatesReturned is not valid.
BadArgument	The length of faceIds is not in a valid range.
BadArgument	LargeFaceListId, faceListId and faceIds, not exactly one of them is valid.
BadArgument	Face list ID is invalid.
BadArgument	Large face list ID is invalid.
BadArgument	LargeFaceListId, faceListId and faceIds are all null.
BadArgument	2 or more of largeFaceListId, faceListId and faceIds are not null.
BadArgument	'recognitionModel' is incompatible.
FaceNotFound	Query face cannot be found.
FaceListNotFound	Face list is not found.
LargeFaceListNotFound	Large face list is not found.
LargeFaceListNotTrained	Large face list is not trained.
FaceListNotReady	Face list is empty.
LargeFaceListTrainingNotFinished	Large face list training is not finished.

Application/json

```
{
   "error": {
      "code": "BadArgument",
      "message": "Request body is invalid."
   }
}
```

Response 401

Error code and message returned in JSON:

Error Code	Error Message Description
Unspecified	Invalid subscription Key or user/plan is blocked.

Application/json

```
{
   "error": {
      "code": "Unspecified",
      "message": "Access denied due to invalid subscription key. Make sure you are subscribed to an API you are trying to call and provide the right key."
   }
}
```

Response 403

Application/json

```
{
   "error": {
      "statusCode": 403,
      "message": "Out of call volume quota. Quota will be replenished in 2 days."
   }
}
```

Response 415

Unsupported media type error. Only "application/json" is valid for this API.

Application/json

```
{
   "error": {
      "code": "BadArgument",
      "message": "Invalid Media Type."
   }
}
```

Response 429

Application/json
```
{
    "error": {
        "statusCode": 429,
        "message": "Rate limit is exceeded. Try again in 26 seconds."
    }
}
```

Code samples

Python	Curl

```
########### Python 3.2 #############
import http.client, urllib.request, urllib.parse, urllib.error, base64

headers = {
    # Request headers
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': '{subscription key}',
}

params = urllib.parse.urlencode({
})

try:
    conn = http.client.HTTPSConnection('westus.api.cognitive.microsoft.com')
    conn.request("POST", "/face/v1.0/findsimilars?%s" % params, "{body}", headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################
```
	```
	@ECHO OFF

	curl -v -X POST "https://westus.api.cognitive.microsoft.com/face/v1.0/findsimilars"
	-H "Content-Type: application/json"
	-H "Ocp-Apim-Subscription-Key: {subscription key}"

	--data-ascii "{body}"
	```

Face - Group

Divide candidate faces into groups based on face similarity.

- The output is one or more disjointed face groups and a messyGroup. A face group contains faces that have similar looking, often of the same person. Face groups are ranked by group size, i.e. number of faces. Notice that faces belonging to a same person might be split into several groups in the result.
- MessyGroup is a special face group containing faces that cannot find any similar counterpart face from original faces. The messyGroup will not appear in the result if all faces found their counterparts.
- Group API needs at least 2 candidate faces and 1000 at most.
- The 'recognitionModel' associated with the query faces' faceIds should be the same.

Http Method

POST

We should select the testing console in the region where we created our resources. (Notice that the region which have written below are link)

West US
West US 2
East US
East US 2
West Central US
South Central US
West Europe
North Europe
South East Asia
East Asia
Australia East
Brazil South
Canada Central
Central India
UK South
Japan East
Central US
France Central
Korea Central
Japan West
North Central US
South Africa North
UAE North

Request URL

https://{endpoint}/face/v1.0/group

Request headers

Content-Type	string	Media type of the body sent to the API.
Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API. Found in your **Cognitive Services accounts.**

Request body

JSON fields in request body:

Fields	Type	Description
faceIds	Array	Array of candidate faceId created by **Face - Detect**. The maximum is 1000 faces.

Application/json

```
{
    "faceIds": [
        "c5c24a82-6845-4031-9d5d-978df9175426",
        "015839fb-fbd9-4f79-ace9-7675fc2f1dd9",
        "65d083d4-9447-47d1-af30-b626144bf0fb",
        "fce92aed-d578-4d2e-8114-068f8af4492e",
        "30ea1073-cc9e-4652-b1e3-d08fb7b95315",
        "be386ab3-af91-4104-9e6d-4dae4c9fddb7",
        "fbd2a038-dbff-452c-8e79-2ee81b1aa84e",
        "b64d5e15-8257-4af2-b20a-5a750f8940e7"
    ]
}
```

Response 200

A successful call returns one or more groups of similar faces (rank by group size) and a messyGroup.

JSON fields in response body:

Fields	Type	Description
groups	Array	A partition of the original faces based on face similarity. Groups are ranked by number of faces.
messyGroup	Array	Face ids array of faces that cannot find any similar faces from original faces.

Application/json

```
{
    "groups": [
        [
            "c5c24a82-6845-4031-9d5d-978df9175426",
            "015839fb-fbd9-4f79-ace9-7675fc2f1dd9",
            "fce92aed-d578-4d2e-8114-068f8af4492e",
            "b64d5e15-8257-4af2-b20a-5a750f8940e7"
        ],
        [
            "65d083d4-9447-47d1-af30-b626144bf0fb",
            "30ea1073-cc9e-4652-b1e3-d08fb7b95315"
        ]
    ],
    "messyGroup": [
        "be386ab3-af91-4104-9e6d-4dae4c9fddb7"
    ]
}
```

Response 400

Error code and message returned in JSON:

Error Code	Error Message Description
BadArgument	Invalid request body.
BadArgument	The length of faceIds is not in a valid range. Out of range [2, 1000].
BadArgument	'recognitionModel' is incompatible.
FaceNotFound	Current face is not found

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Request body is invalid."
    }
}
```

Response 401

Error code and message returned in JSON:

Error Code	Error Message Description
Unspecified	Invalid subscription Key or user/plan is blocked.

Application/json

```
{
    "error": {
        "code": "Unspecified",
        "message": "Access denied due to invalid subscription key. Make sure you are
subscribed to an API you are trying to call and provide the right key."
    }
}
```

Response 403

Application/json
{ "error": { "statusCode": 403, "message": "Out of call volume quota. Quota will be replenished in 2 days." } }

```
{
    "error": {
        "statusCode": 403,
        "message": "Out of call volume quota. Quota will be replenished in 2 days."
    }
}
```

Response 415

Unsupported media type error. Only "application/json" is valid for this API.

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Invalid Media Type."
    }
}
```

Response 429

Application/json

```
{
    "error": {
        "statusCode": 429,
        "message": "Rate limit is exceeded. Try again in 26 seconds."
    }
}
```

Code samples

Python

```
########### Python 3.2 #############
import http.client, urllib.request, urllib.parse,
urllib.error, base64

headers = {
    # Request headers
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': '{subscription key}',
}

params = urllib.parse.urlencode({
})

try:
    conn =
http.client.HTTPSConnection('westus.api.cognitive.m
icrosoft.com')
    conn.request("POST", "/face/v1.0/group?%s" %
params, "{body}", headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################
```

Curl

```
@ECHO OFF

curl -v -X POST
"https://westus.api.cognitive.microsoft.
com/face/v1.0/group"
-H "Content-Type: application/json"
-H "Ocp-Apim-Subscription-Key:
{subscription key}"

--data-ascii "{body}"
```

Face - Identify

1-to-many identification to find the closest matches of the specific query person face from a person group or large person group.

For each face in the faceIds array, Face Identify will compute similarities between the query face and all the faces in the person group (given by personGroupId) or large person group (given by largePersonGroupId), and return candidate person(s) for that face ranked by similarity confidence. The person group/large person group should be trained to make it ready for identification. You can find more in **PersonGroup - Train** and **LargePersonGroup - Train.**

Remarks:

- The algorithm allows more than one face to be identified independently at the same request, but no more than 10 faces.
- Each person in the person group/large person group could have more than one face, but no more than 248 faces.
- Higher face image quality means better identification precision. Consider high-quality faces: frontal, clear, and face size is 200x200 pixels (100 pixels between eyes) or bigger.
- Number of candidates returned is restricted by maxNumOfCandidatesReturned and confidenceThreshold. If no person is identified, the returned candidates will be an empty array.
- Try **Face - Find Similar** when you need to find similar faces from a face list/large face list instead of a person group/large person group.
- The 'recognitionModel' associated with the query faces' faceIds should be the same as the 'recognitionModel' used by the target person group or large person group.

Http Method

POST

We should select the testing console in the region where we created our resources. (Notice that the region which have written below are link)

West US
West US 2
East US
East US 2
West Central US
South Central US
West Europe
North Europe
South East Asia
East Asia
Australia East
Brazil South
Canada Central
Central India
UK South
Japan East
Central US
France Central
Korea Central
Japan West
North Central US
South Africa North
UAE North

Request URL

https://{endpoint}/face/v1.0/identify

Request headers

Content-Type	string	Media type of the body sent to the API.
Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API.

Request body

JSON fields in request body:

Fields	**Type**	**Description**
faceIds	Array	Array of query faces faceIds, created by the **Face - Detect**. Each of the faces are identified independently. The valid number of faceIds is between [1, 10].
personGroupId	String	personGroupId of the target person group, created by **PersonGroup - Create**. Parameter personGroupId and largePersonGroupId should not be provided at the same time.
largePersonGroupId	String	largePersonGroupId of the target large person group, created by **LargePersonGroup - Create.** Parameter personGroupId and largePersonGroupId should not be provided at the same time.
maxNumOfCandidatesReturned (optional)	Number	The range of maxNumOfCandidatesReturned is between 1 and 100 (default is 10).
confidenceThreshold (optional)	Number	Optional parameter. Customized identification confidence threshold, in the range of [0, 1]. Advanced user can tweak this value to override default internal threshold for better precision on their scenario data. Note there is no guarantee of this threshold value working on other data and after algorithm updates.

Application/json

```
{
    "largePersonGroupId": "sample_group",
    "faceIds": [
        "c5c24a82-6845-4031-9d5d-978df9175426",
        "65d083d4-9447-47d1-af30-b626144bf0fb"
    ],
    "maxNumOfCandidatesReturned": 1,
    "confidenceThreshold": 0.5
}
```

Response 200

A successful call returns the identified candidate person(s) for each query face.

JSON fields in response body:

Fields	Type	Description
faceId	String	faceId of the query face.
candidates	Array	Identified person candidates for that face (ranked by confidence). Array size should be no larger than input maxNumOfCandidatesReturned. If no person is identified, will return an empty array.
personId	String	personId of candidate person.
confidence	Number	A float number between 0.0 and 1.0.

Application/json

```
[
    {
        "faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
        "candidates": [
            {
                "personId": "25985303-c537-4467-b41d-bdb45cd95ca1",
                "confidence": 0.92
            }
        ]
    },
    {
        "faceId": "65d083d4-9447-47d1-af30-b626144bf0fb",
        "candidates": [
            {
                "personId": "2ae4935b-9659-44c3-977f-61fac20d0538",
                "confidence": 0.89
            }
        ]
    }
]
```

Response 400

Error code and message returned in JSON:

Error Code	Error Message Description
BadArgument	Invalid request body.
BadArgument	The argument maxNumOfCandidatesReturned is not valid. Range is [1,5]
BadArgument	The argument confidenceThreshold is not valid. Range is [0, 1]
BadArgument	Face ID is invalid.
BadArgument	Person group ID is invalid. Valid format should be a string composed by numbers, English letters in lower case, '-', '_', and no longer than 64 characters.
BadArgument	Large person group ID is invalid. Valid format should be a string composed by numbers, English letters in lower case, '-', '_', and no longer than 64 characters.
BadArgument	'recognitionModel' is incompatible.
PersonGroupIdAndLargePersonGroupIdBothNotNull	Large person group ID and person group ID are both not null.
PersonGroupIdAndLargePersonGroupIdBothNull	Large person group ID and person group ID are both null.
PersonGroupNotFound	Person group is not found.
LargePersonGroupNotFound	Large person group is not found.
FaceNotFound	Face is not found.
PersonGroupNotTrained	Person group not trained.
LargePersonGroupNotTrained	Large person group not trained.
PersonGroupTrainingNotFinished	Person group is under training.
LargePersonGroupTrainingNotFinished	Large person group is under training.

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Large Person group is invalid."
    }
}
```

Response 401

Error code and message returned in JSON:

Error Code	Error Message Description
Unspecified	Invalid subscription Key or user/plan is blocked.

Application/json

```
{
    "error": {
        "code": "Unspecified",
        "message": "Access denied due to invalid subscription key. Make sure you are
subscribed to an API you are trying to call and provide the right key."
    }
}
```

Response 403

Application/json

```
{
    "error": {
        "statusCode": 403,
        "message": "Out of call volume quota. Quota will be replenished in 2 days."
    }
}
```

Response 409

Training process is conflict with identify. Try identify again when training is finished.

Application/json
{ "error": { "code": "LargePersonGroupTrainingNotFinished", "message": "Large person group is under training." } }

```
{
    "error": {
        "code": "LargePersonGroupTrainingNotFinished",
        "message": "Large person group is under training."
    }
}
```

Response 415

Unsupported media type error. Only "application/json" is valid for this API.

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Invalid Media Type."
    }
}
```

Response 429

Application/json

```
{
    "error": {
        "statusCode": 429,
        "message": "Rate limit is exceeded. Try again in 26 seconds."
    }
}
```

Code samples

Python

```
########### Python 3.2 #############
import http.client, urllib.request, urllib.parse,
urllib.error, base64

headers = {
    # Request headers
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': '{subscription key}',
}

params = urllib.parse.urlencode({
})

try:
    conn =
http.client.HTTPSConnection('westus.api.cognitive.m
icrosoft.com')
    conn.request("POST", "/face/v1.0/identify?%s" %
params, "{body}", headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################
```

Curl

```
@ECHO OFF

curl -v -X POST
"https://westus.api.cognitive.microsoft.
com/face/v1.0/identify"
-H "Content-Type: application/json"
-H "Ocp-Apim-Subscription-Key:
{subscription key}"

--data-ascii "{body}"
```

Face - Verify

Verify whether two faces belong to a same person or whether one face belongs to a person.

Remarks:

- Higher face image quality means better identification precision. Consider high-quality faces: frontal, clear, and face size is 200x200 pixels (100 pixels between eyes) or bigger.
- The 'recognitionModel' associated with the query faces' faceIds should be the same as the 'recognitionModel' used by the target face, person group or large person group.

Http Method

POST

We should select the testing console in the region where we created our resources. (Notice that the region which have written below are link)

West US
West US 2
East US
East US 2
West Central US
South Central US
West Europe
North Europe
South East Asia
East Asia
Australia East
Brazil South
Canada Central
Central India
UK South
Japan East
Central US
France Central
Korea Central
Japan West
North Central US
South Africa North
UAE North

Request URL

https://{endpoint}/face/v1.0/verify

Request headers

Content-Type	string	Media type of the body sent to the API.
Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API.

Request body

JSON fields in face to face verification request body:

Fields	**Type**	**Description**
faceId1	String	faceId of one face, comes from **Face - Detect**.
faceId2	String	faceId of another face, comes from **Face - Detect.**

JSON fields in face to person verification request body:

Fields	**Type**	**Description**
Description	String	faceId of the face, comes from **Face - Detect.**
personGroupId	String	Using existing personGroupId and personId for fast loading a specified person. personGroupId is created in **PersonGroup - Create**. Parameter personGroupId and largePersonGroupId should not be provided at the same time.
largePersonGroupId	String	Using existing largePersonGroupId and personId for fast loading a specified person. largePersonGroupId is created in **LargePersonGroup - Create**. Parameter personGroupId and largePersonGroupId should not be provided at the same time.
personId	String	Specify a certain person in a person group or a large person group. personId is created in **PersonGroup Person - Create** or **LargePersonGroup Person - Create.**

Application/json
{ "faceId": "c5c24a82-6845-4031-9d5d-978df9175426", "personId": "815df99c-598f-4926-930a-a734b3fd651c", "largePersonGroupId": "sample_group" }

```
{
    "faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
    "personId": "815df99c-598f-4926-930a-a734b3fd651c",
    "largePersonGroupId": "sample_group"
}
```

Response 200

A successful call returns the verification result. JSON fields in response body:

JSON fields in response body:

Fields	Type	Description
isIdentical	Boolean	True if the two faces belong to the same person or the face belongs to the person, otherwise false.
confidence	Number	A number indicates the similarity confidence of whether two faces belong to the same person, or whether the face belongs to the person. By default, isIdentical is set to True if similarity confidence is greater than or equal to 0.5. This is useful for advanced users to override "isIdentical" and fine-tune the result on their own data.

Application/json

```
{
    "isIdentical": true,
    "confidence": 0.9
}
```

Response 400

Error code and message returned in JSON:

Error Code	Error Message Description
BadArgument	Bad and unrecognizable JSON body.
BadArgument	Face ID is invalid. faceId1, faceId2 or faceId is invalid and valid faceId comes from **Face - Detect.**
BadArgument	Person ID is invalid. Valid personId is generated from **PersonGroup Person - Create** or **LargePersonGroup Person - Create** for existing person.
BadArgument	Request body is invalid.
BadArgument	Person group ID or Large Perosn Group ID is invalid. Valid format should be a string composed by numbers, English letters in lower case, '-', '_', and no longer than 64 characters.
BadArgument	'recognitionModel' is incompatible.

Application/json
{ "error": { "code": "BadArgument", "message": "Request body is invalid." } }

Response 401

Error code and message returned in JSON:

Error Code	Error Message Description
Unspecified	Invalid subscription Key or user/plan is blocked.

Application/json
{ "error": { "code": "Unspecified", "message": "Access denied due to invalid subscription key. Make sure you are subscribed to an API you are trying to call and provide the right key." } }

Response 403

Application/json

```
{
    "error": {
        "statusCode": 403,
        "message": "Out of call volume quota. Quota will be replenished in 2 days."
    }
}
```

Response 404

Error code and message returned in JSON:

Error Code	Error Message Description
FaceNotFound	Face is not found. The faceId is expired or not exist.
PersonNotFound	Person is not found.
PersonGroupNotFound	Person Group is not found.
LargePersonGroupNotFound	Large Person Group is not found.

‘

Application/json

```
{
    "error": {
        "code": "LargePersonGroupNotFound",
        "message": "Large person group is not found."
    }
}
```

Response 415

Unsupported media type error. Only "application/json" is valid for this API.

Application/json

```
{
    "error": {
        "code": "BadArgument",
        "message": "Invalid Media Type."
    }
}
```

Response 429

Application/json

```
{
    "error": {
        "statusCode": 429,
        "message": "Rate limit is exceeded. Try again in 26 seconds."
    }
}
```

Code samples

Python

```
############ Python 3.2 #############
import http.client, urllib.request, urllib.parse,
urllib.error, base64

headers = {
    # Request headers
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': '{subscription key}',
}

params = urllib.parse.urlencode({
})

try:
    conn =
http.client.HTTPSConnection('westus.api.cognitive.m
icrosoft.com')
    conn.request("POST", "/face/v1.0/verify?%s" %
params, "{body}", headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################
```

Curl

```
curl -v -X POST
"https://westus.api.cognitive.microsoft.
com/face/v1.0/verify"
-H "Content-Type: application/json"
-H "Ocp-Apim-Subscription-Key:
{subscription key}"

--data-ascii "{body}"
```

Microsoft Azure [27]

What is Azure Cognitive Services?

Cognitive Services bring AI within reach of every developer—without requiring machine-learning expertise. All it takes is an API call to embed the ability to see, hear, speak, search, understand, and accelerate decision-making into your apps.

Deliver low-friction, state-of-the-art facial recognition

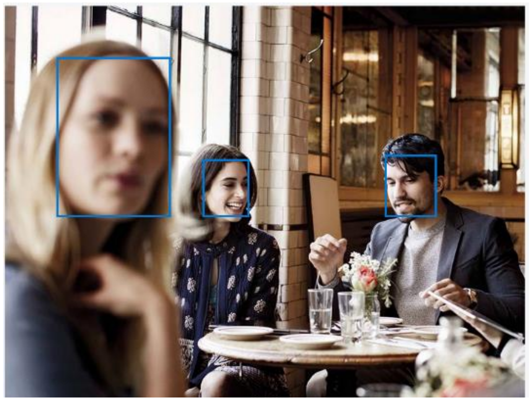
Embed facial recognition into your apps for a seamless and highly secured user experience. No machine learning expertise is required. Features include: face detection that perceives faces and attributes in an image; person identification that matches an individual in your private repository of up to 1 million people; perceived emotion recognition that detects a range of facial expressions like happiness, contempt, neutrality, and fear; and recognition and grouping of similar faces in images.

Apply facial recognition for a range of scenarios

Detect, identify, and analyze faces in images and videos. Build on top of this technology to support various scenarios—for example, authenticate people for access, count people in a space for crowd control, or garner crowd insights for media campaigns.

Face detection

Detect one or more human faces along with attributes such as: age, emotion, gender, pose, smile and facial hair, Including 27 landmarks for each face in the image.



Detection result:
detection_02
JSON:

```
 [
  {
    "faceId": "5f262116-0400-4817-a57c-4d618136f11e",
    "faceRectangle": {
      "top": 101,
      "left": 96,
      "width": 218,
      "height": 307
    },
    "faceAttributes": null,
    "faceLandmarks": null
  },
  {
    "faceId": "22853634-1a46-4010-9a54-1816040df042",
    "faceRectangle": {
      "top": 286,
      "left": 728,
      "width": 96,
      "height": 120
    },
    "faceAttributes": null,
    "faceLandmarks": null
  },
  {
    "faceId": "cdc8110f-ed0f-4956-a900-666fa416f3a5",
    "faceRectangle": {
      "top": 296,
      "left": 385,
      "width": 86,
      "height": 116
    },
    "faceAttributes": null,
    "faceLandmarks": null
  }
```

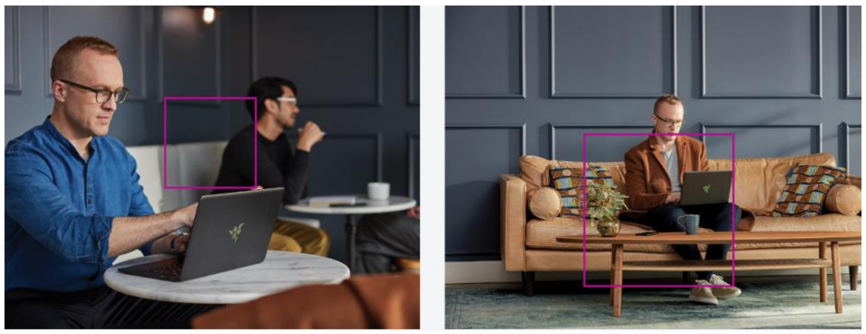
Detection result:
detection_01
JSON:

```
 [
  {
    "faceId": "b493d64a-8e79-4b41-8528-4748357e37e2",
    "faceRectangle": {
      "top": 166,
      "left": 128,
      "width": 218,
      "height": 218
    },
    "faceAttributes": {
      "hair": {
        "bald": 0.07,
        "invisible": false,
        "hairColor": [
        {
          "color": "blond",
          "confidence": 1.0
        },
        {
          "color": "red",
          "confidence": 0.91
        },
        {
          "color": "brown",
          "confidence": 0.69
        },
        {
          "color": "other",
          "confidence": 0.31
        },
        {
          "color": "gray",
          "confidence": 0.04
        },
        {
          "color": "black",
          "confidence": 0.01
        }
      ]
    },
    "smile": 0.003,
    "headPose": {
      "pitch": 0.4,
      "roll": 12.9,
      "yaw": 23.4
    },
    "gender": "female",
    "age": 20.0,
    "facialHair": {
      "moustache": 0.0,
      "beard": 0.0,
      "sideburns": 0.0
    },
    "glasses": "NoGlasses",
    "makeup": {
      "eyeMakeup": true,
      "lipMakeup": true
    },
    "emotion": {
      "anger": 0.0,
      "contempt": 0.0,
      "disgust": 0.0,
      "fear": 0.0,
      "happiness": 0.003,
      "neutral": 0.996,
      "sadness": 0.0,
      "surprise": 0.001
    },
    "occlusion": {
      "foreheadOccluded": false,
      "eyeOccluded": false,
      "mouthOccluded": false
    },
    "accessories": [],
    "blur": {
      "blurLevel": "high",
      "value": 0.9
    },
    "exposure": {
      "exposureLevel": "goodExposure",
      "value": 0.64
    },
    "noise": {
      "noiseLevel": "low",
      "value": 0.0
    }
  },
  "faceLandmarks": {
    "pupilLeft": {
      "x": 197.8,
      "y": 212.6
    },
    "pupilRight": {
      "x": 288.7,
      "y": 232.5
    },
    "noseTip": {
      "x": 245.6,
      "y": 278.8
    },
    "mouthLeft": {
      "x": 193.4,
      "y": 324.9
    },
    "mouthRight": {
      "x": 256.7,
      "y": 340.5
    },
    "eyebrowLeftOuter": {
      "x": 170.0,
      "y": 179.1
    },
    "eyebrowLeftInner": {
      "x": 231.6,
      "y": 189.2
    },
    "eyeLeftOuter": {
      "x": 182.8,
      "y": 210.0
    },
    "eyeLeftTop": {
      "x": 198.7,
      "y": 206.5
    },
    "eyeLeftBottom": {
      "x": 196.8,
    "y": 217.0
  },
  "eyeLeftInner": {
    "x": 211.9,
    "y": 215.9
  },
  "eyebrowRightInner": {
    "x": 275.2,
    "y": 199.0
  },
  "eyebrowRightOuter": {
    "x": 325.2,
    "y": 208.1
  },
  "eyeRightInner": {
    "x": 276.3,
    "y": 229.6
  },
  "eyeRightTop": {
    "x": 292.8,
    "y": 226.2
  },
  "eyeRightBottom": {
    "x": 291.3,
    "y": 237.1
  },
  "eyeRightOuter": {
    "x": 306.0,
    "y": 234.9
  },
  "noseRootLeft": {
    "x": 232.5,
    "y": 220.3
  },
  "noseRootRight": {
    "x": 260.9,
    "y": 227.0
  },
  "noseLeftAlarTop": {
    "x": 220.9,
    "y": 256.0
  },
  "noseRightAlarTop": {
    "x": 265.2,
    "y": 263.3
  },
  "noseLeftAlarOutTip": {
    "x": 205.8,
    "y": 272.6
  },
  "noseRightAlarOutTip": {
    "x": 270.9,
    "y": 286.1
  },
  "upperLipTop": {
    "x": 236.2,
        "y": 316.8
      },
      "upperLipBottom": {
        "x": 233.8,
        "y": 331.6
      },
      "underLipTop": {
        "x": 232.3,
        "y": 332.3
      },
      "underLipBottom": {
        "x": 230.5,
        "y": 348.5
      }
    }
  },
  {
    "faceId": "cd0af163-a995-4b64-86b2-833d700d1ac8",
    "faceRectangle": {
      "top": 314,
      "left": 718,
      "width": 88,
      "height": 88
    },
    "faceAttributes": {
      "hair": {
        "bald": 0.11,
        "invisible": false,
        "hairColor": [
          {
            "color": "black",
            "confidence": 1.0
          },
          {
            "color": "brown",
            "confidence": 0.97
          },
          {
            "color": "other",
            "confidence": 0.25
          },
          {
            "color": "gray",
            "confidence": 0.23
          },
          {
            "color": "red",
            "confidence": 0.03
          },
          {
            "color": "blond",
            "confidence": 0.01
          }
        ]
      },
      "smile": 0.988,
    "headPose": {
      "pitch": -23.5,
      "roll": -12.2,
      "yaw": -32.3
    },
    "gender": "male",
    "age": 31.0,
    "facialHair": {
      "moustache": 0.4,
      "beard": 0.4,
      "sideburns": 0.4
    },
    "glasses": "NoGlasses",
    "makeup": {
      "eyeMakeup": false,
      "lipMakeup": false
    },
    "emotion": {
      "anger": 0.0,
      "contempt": 0.001,
      "disgust": 0.0,
      "fear": 0.0,
      "happiness": 0.988,
      "neutral": 0.01,
      "sadness": 0.0,
      "surprise": 0.0
    },
    "occlusion": {
      "foreheadOccluded": false,
      "eyeOccluded": false,
      "mouthOccluded": false
    },
    "accessories": [],
    "blur": {
      "blurLevel": "low",
      "value": 0.04
    },
    "exposure": {
      "exposureLevel": "goodExposure",
      "value": 0.58
    },
    "noise": {
      "noiseLevel": "low",
      "value": 0.0
    }
  },
  "faceLandmarks": {
    "pupilLeft": {
      "x": 744.2,
      "y": 339.2
    },
    "pupilRight": {
      "x": 778.3,
      "y": 334.1
    },
    "noseTip": {
```

```
    "x": 755.2,
    "y": 361.8
  },
  "mouthLeft": {
    "x": 750.3,
    "y": 382.2
  },
  "mouthRight": {
    "x": 782.4,
    "y": 377.6
  },
  "eyebrowLeftOuter": {
    "x": 728.7,
    "y": 331.7
  },
  "eyebrowLeftInner": {
    "x": 746.4,
    "y": 329.5
  },
  "eyeLeftOuter": {
    "x": 738.6,
    "y": 340.4
  },
  "eyeLeftTop": {
    "x": 743.1,
    "y": 337.7
  },
  "eyeLeftBottom": {
    "x": 743.5,
    "y": 340.8
  },
  "eyeLeftInner": {
    "x": 748.5,
    "y": 339.1
  },
  "eyebrowRightInner": {
    "x": 762.7,
    "y": 326.4
  },
  "eyebrowRightOuter": {
    "x": 795.2,
    "y": 324.3
  },
  "eyeRightInner": {
    "x": 771.9,
    "y": 335.1
  },
  "eyeRightTop": {
    "x": 778.0,
    "y": 332.0
  },
  "eyeRightBottom": {
    "x": 778.2,
    "y": 335.6
  },
  "eyeRightOuter": {
        "x": 783.4,
        "y": 333.3
      },
      "noseRootLeft": {
        "x": 750.7,
        "y": 339.1
      },
      "noseRootRight": {
        "x": 762.2,
        "y": 337.5
      },
      "noseLeftAlarTop": {
        "x": 749.2,
        "y": 354.2
      },
      "noseRightAlarTop": {
        "x": 766.2,
        "y": 353.2
      },
      "noseLeftAlarOutTip": {
        "x": 746.4,
        "y": 362.5
      },
      "noseRightAlarOutTip": {
        "x": 771.0,
        "y": 360.0
      },
      "upperLipTop": {
        "x": 759.5,
        "y": 374.8
      },
      "upperLipBottom": {
        "x": 760.3,
        "y": 378.5
      },
      "underLipTop": {
        "x": 762.9,
        "y": 385.2
      },
      "underLipBottom": {
        "x": 763.6,
        "y": 390.3
      }
    }
  },
  {
    "faceId": "64556a31-b1d6-47a0-beef-507373256b4a",
    "faceRectangle": {
      "top": 320,
      "left": 407,
      "width": 83,
      "height": 83
    },
    "faceAttributes": {
      "hair": {
        "bald": 0.04,
    "invisible": false,
    "hairColor": [
      {
        "color": "brown",
        "confidence": 1.0
      },
      {
        "color": "black",
        "confidence": 0.93
      },
      {
        "color": "red",
        "confidence": 0.2
      },
      {
        "color": "gray",
        "confidence": 0.1
      },
      {
        "color": "other",
        "confidence": 0.09
      },
      {
        "color": "blond",
        "confidence": 0.08
      }
    ]
  },
  "smile": 1.0,
  "headPose": {
    "pitch": -5.4,
    "roll": 2.4,
    "yaw": 31.0
  },
  "gender": "female",
  "age": 24.0,
  "facialHair": {
    "moustache": 0.0,
    "beard": 0.0,
    "sideburns": 0.0
  },
  "glasses": "NoGlasses",
  "makeup": {
    "eyeMakeup": true,
    "lipMakeup": true
  },
  "emotion": {
    "anger": 0.0,
    "contempt": 0.0,
    "disgust": 0.0,
    "fear": 0.0,
    "happiness": 1.0,
    "neutral": 0.0,
    "sadness": 0.0,
    "surprise": 0.0
  },
    "occlusion": {
      "foreheadOccluded": false,
      "eyeOccluded": false,
      "mouthOccluded": false
    },
    "accessories": [],
    "blur": {
      "blurLevel": "low",
      "value": 0.0
    },
    "exposure": {
      "exposureLevel": "goodExposure",
      "value": 0.63
    },
    "noise": {
      "noiseLevel": "low",
      "value": 0.03
    }
  },
  "faceLandmarks": {
    "pupilLeft": {
      "x": 431.0,
      "y": 342.8
    },
    "pupilRight": {
      "x": 463.6,
      "y": 342.1
    },
    "noseTip": {
      "x": 455.7,
      "y": 363.0
    },
    "mouthLeft": {
      "x": 430.0,
      "y": 383.1
    },
    "mouthRight": {
      "x": 463.5,
      "y": 382.5
    },
    "eyebrowLeftOuter": {
      "x": 413.5,
      "y": 335.4
    },
    "eyebrowLeftInner": {
      "x": 444.2,
      "y": 331.4
    },
    "eyeLeftOuter": {
      "x": 424.0,
      "y": 343.4
    },
    "eyeLeftTop": {
      "x": 430.6,
      "y": 341.1
    },
"eyeLeftBottom": {
  "x": 430.4,
  "y": 345.3
},
"eyeLeftInner": {
  "x": 436.5,
  "y": 343.2
},
"eyebrowRightInner": {
  "x": 458.9,
  "y": 331.1
},
"eyebrowRightOuter": {
  "x": 476.2,
  "y": 331.6
},
"eyeRightInner": {
  "x": 458.2,
  "y": 342.4
},
"eyeRightTop": {
  "x": 463.6,
  "y": 340.1
},
"eyeRightBottom": {
  "x": 464.1,
  "y": 344.2
},
"eyeRightOuter": {
  "x": 469.2,
  "y": 341.5
},
"noseRootLeft": {
  "x": 444.3,
  "y": 343.0
},
"noseRootRight": {
  "x": 454.1,
  "y": 342.7
},
"noseLeftAlarTop": {
  "x": 442.8,
  "y": 357.9
},
"noseRightAlarTop": {
  "x": 458.9,
  "y": 356.3
},
"noseLeftAlarOutTip": {
  "x": 439.3,
  "y": 366.4
},
"noseRightAlarOutTip": {
  "x": 463.6,
  "y": 363.6
},
      "upperLipTop": {
        "x": 451.5,
        "y": 377.6
      },
      "upperLipBottom": {
        "x": 451.4,
        "y": 380.8
      },
      "underLipTop": {
        "x": 450.1,
        "y": 388.4
      },
      "underLipBottom": {
        "x": 450.2,
        "y": 391.7
      }
    }
  }
]
```

Face verification

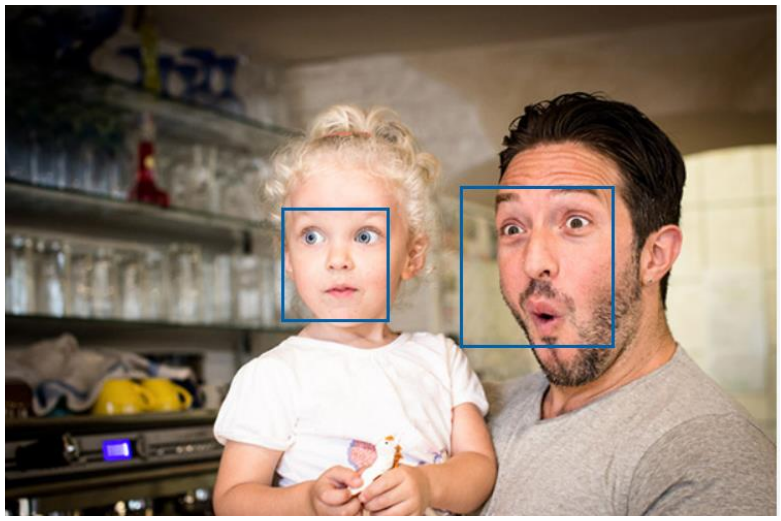
Check the likelihood that two faces belong to the same person and receive a confidence score.



Verification result: The two faces belong to the same person. Confidence is 0.94865

Perceived emotion recognition

Detect perceived facial expressions such as anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise. It is important to note that facial expressions alone do not represent the internal states of people.



Detection result:
2 faces detected

```
 JSON:
 [
  {
    "faceRectangle": {
      "top": 141,
      "left": 356,
      "width": 123,
      "height": 123
    },
    "faceAttributes": {
      "emotion": {
        "anger": 0.0,
        "contempt": 0.0,
        "disgust": 0.0,
        "fear": 0.006,
        "happiness": 0.0,
        "neutral": 0.002,
        "sadness": 0.0,
        "surprise": 0.991
      }
    }
  },
  {
    "faceRectangle": {
      "top": 157,
      "left": 216,
      "width": 87,
      "height": 87
    },
    "faceAttributes": {
      "emotion": {
        "anger": 0.001,
        "contempt": 0.104,
        "disgust": 0.0,
        "fear": 0.001,
        "happiness": 0.001,
        "neutral": 0.093,
        "sadness": 0.0,
        "surprise": 0.8
      }
    }
  }
]
```

KAIROS [28]

KAIROS is building a Face Recognition platform that lets you quickly integrate **human identity features** into your products and services—it's speedy, safe, and secure.

Kairos is a simple concept—you submit images into KAIROS’s API, and its deep learning algorithms analyze the faces found, then the API returns a bunch of useful data about the faces it find. You can use this to search, match and compare faces, or measure characteristics such as age, and gender.

Overview

Detect human faces in photos and images. Kairos returns information on facial features as coordinates on the image.

Base URL

api.kairos.com

Authentication

Requests must be authenticated with your API key. This must be sent as an HTTP header:

app_id: "YOUR APP_ID"
app_key: "YOUR APP_KEY"

*It’s good to notice that you can create and manage your API Keys via your **dashboard**. It's free to sign-up and start testing.

Requests

Requests return a JSON object with the header:

Content-Type: application/json

POST /enroll

Takes a photo, finds the faces within it, and stores the faces into a gallery you create.

To enroll someone into a gallery, all you need to do is submit a JPG or PNG photo. You can submit the photo either as a publicly accessible URL, Base64 encoded photo or as a file upload.

Next you need to choose an identifier for the person being enrolled. The identifier could be their name ("Anjela"), something unique to your app ("ABC128xyz"), or anything meaningful for you. We call that identifier "subject_id".

You also need to pick a name for the gallery API is storing your faces in. Here it is called "gallery_name". If you had used that gallery name before, it will just add your new face to it, otherwise it will create a new gallery for you.

Introducting the "Liveness" feature:

Detects pixel-level patterns to predict spoof-attacks, enabling the best possible combination of user experience, speed and security robustness. To use this feature, simply include the "liveness" selector in your API request.

Notice that this feature is only available for Business & Enterprise Cloud Plans, as well with On-Premises Solutions.

Required Parameters

image: Publicly accessible URL, file upload or Base64 encoded photo.
subject_id: Defined by you. Is used as an identifier for the face.
gallery_name: Defined by you. Is used to identify the gallery.

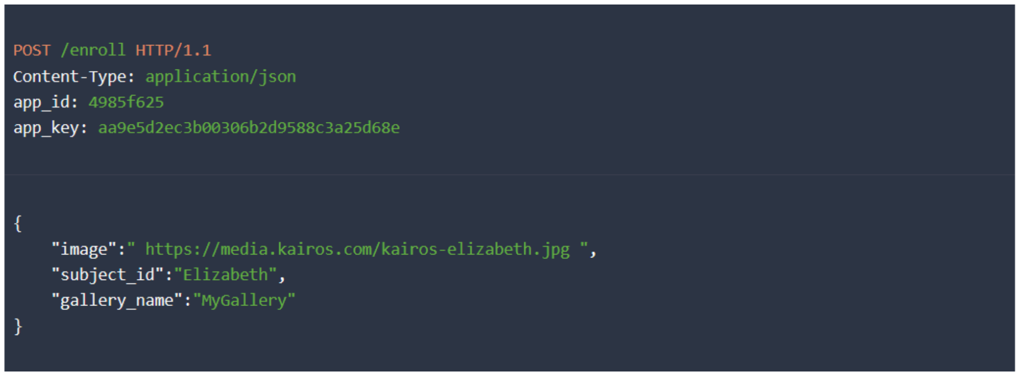
Optional Parameters

minHeadScale: Defined by you. Is used to set the ratio of the smallest face the API should look for in the photo. Accepts a value between .015 (1:64 scale) and .5 (1:2 scale). By default it is set at .015 (1:64 scale) if not specified.
multiple_faces: If set to true lets the API enroll every face found in your photo under the same subject_id.
selector: LIVENESS used to predict spoof-attacks.

Let see it in practice:

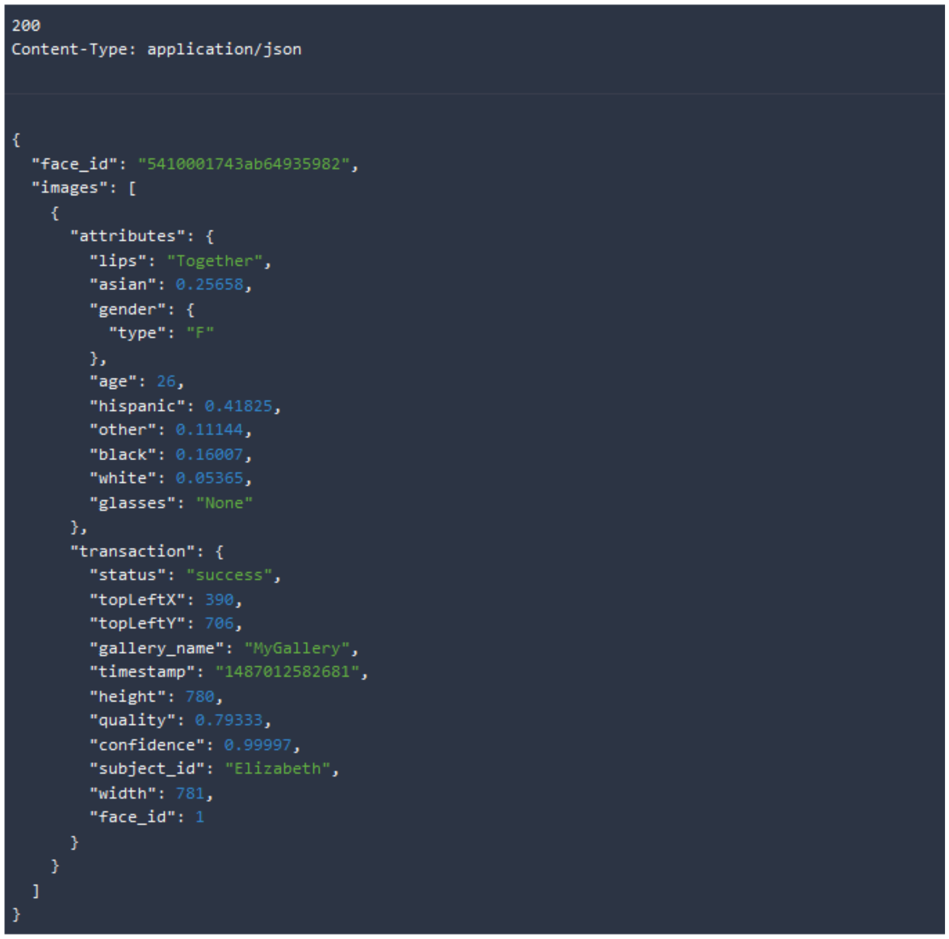
POST https://api.kairos.com/enroll

Request



```
POST /enroll HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e


{
    "image":" https://media.kairos.com/kairos-elizabeth.jpg ",
    "subject_id":"Elizabeth",
    "gallery_name":"MyGallery"
}
```

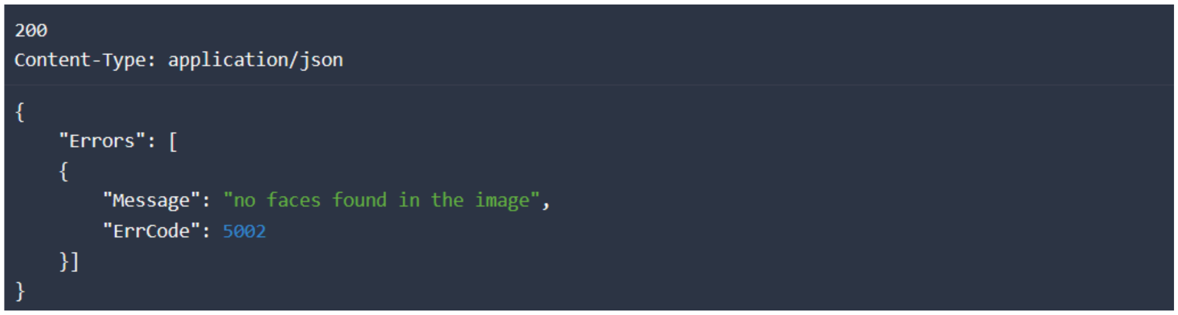
Response



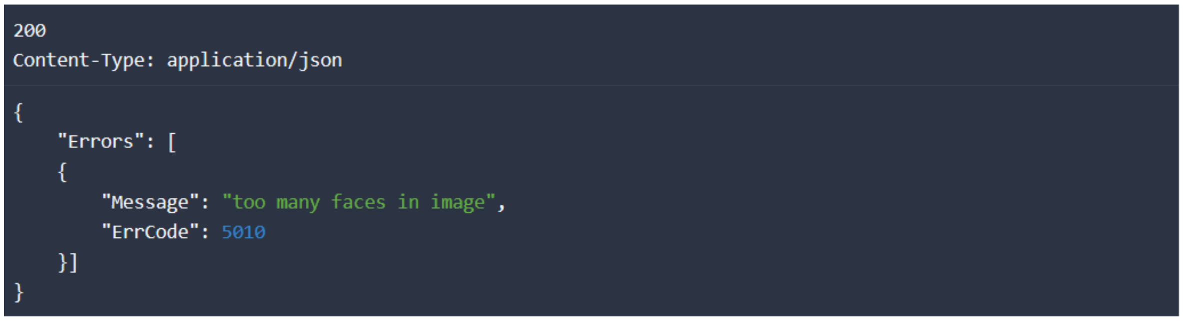
```
200
Content-Type: application/json


{
  "face_id": "5410001743ab64935982",
  "images": [
    {
      "attributes": {
        "lips": "Together",
        "asian": 0.25658,
        "gender": {
          "type": "F"
        },
        "age": 26,
        "hispanic": 0.41825,
        "other": 0.11144,
        "black": 0.16007,
        "white": 0.05365,
        "glasses": "None"
      },
      "transaction": {
        "status": "success",
        "topLeftX": 390,
        "topLeftY": 706,
        "gallery_name": "MyGallery",
        "timestamp": "1487012582681",
        "height": 780,
        "quality": 0.79333,
        "confidence": 0.99997,
        "subject_id": "Elizabeth",
        "width": 781,
        "face_id": 1
      }
    }
  ]
}
```

If too many faces exist in the image or no face found in the image, the response is like below:

Response

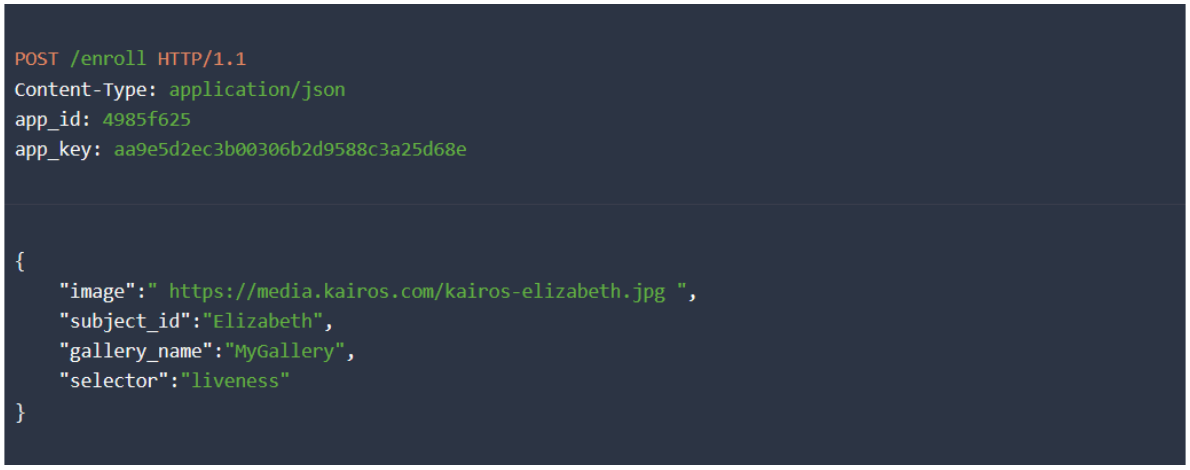


```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "no faces found in the image",
        "ErrCode": 5002
    }]
}
```



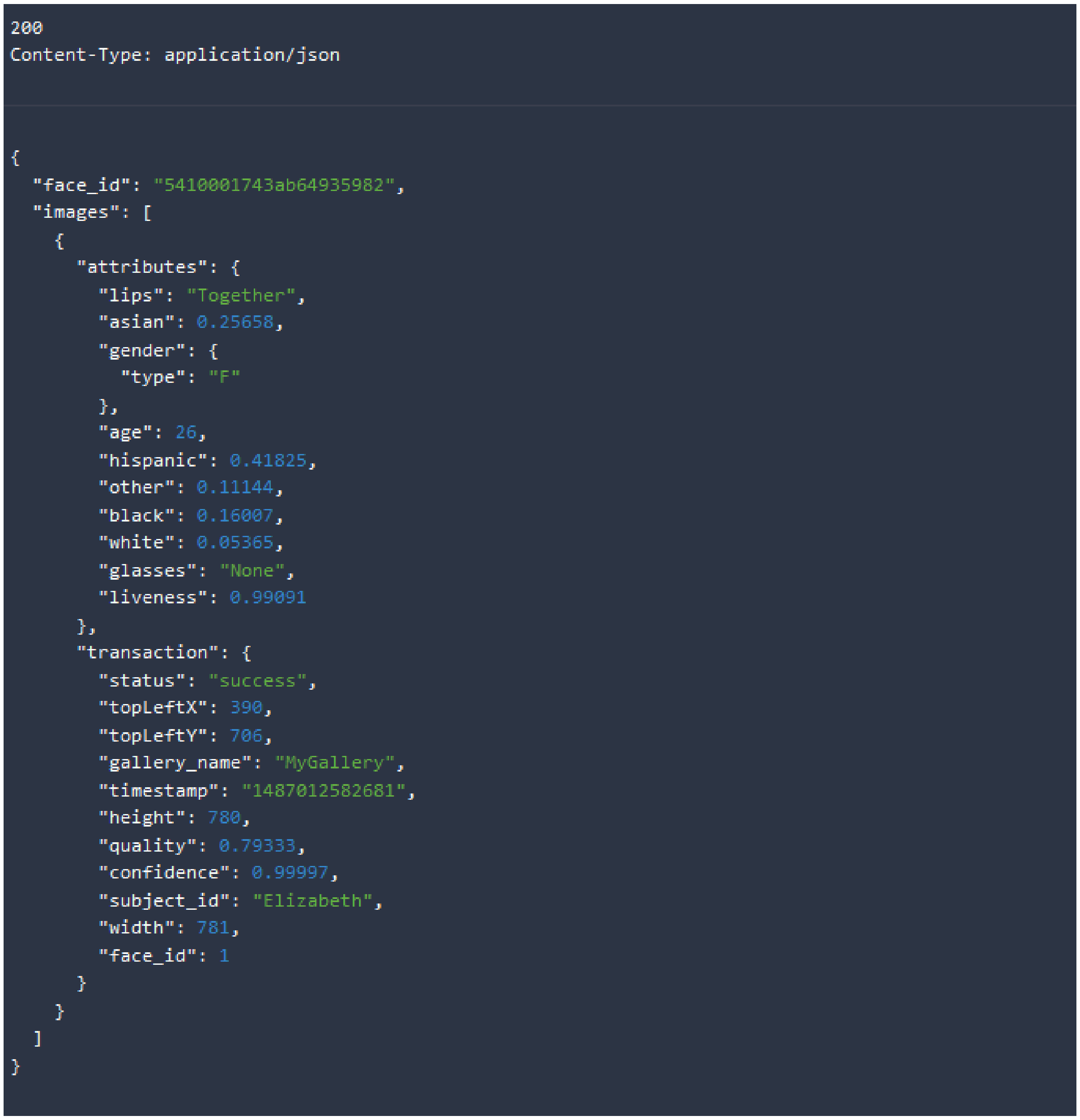
```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "too many faces in image",
        "ErrCode": 5010
    }]
}
```

Request (includes “liveness” selectors)



```
POST /enroll HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e


{
    "image":" https://media.kairos.com/kairos-elizabeth.jpg ",
    "subject_id":"Elizabeth",
    "gallery_name":"MyGallery",
    "selector":"liveness"
}
```

Response (includes "liveness" attribute)



```
200
Content-Type: application/json


{
  "face_id": "5410001743ab64935982",
  "images": [
    {
      "attributes": {
        "lips": "Together",
        "asian": 0.25658,
        "gender": {
          "type": "F"
        },
        "age": 26,
        "hispanic": 0.41825,
        "other": 0.11144,
        "black": 0.16007,
        "white": 0.05365,
        "glasses": "None",
        "liveness": 0.99091
      },
      "transaction": {
        "status": "success",
        "topLeftX": 390,
        "topLeftY": 706,
        "gallery_name": "MyGallery",
        "timestamp": "1487012582681",
        "height": 780,
        "quality": 0.79333,
        "confidence": 0.99997,
        "subject_id": "Elizabeth",
        "width": 781,
        "face_id": 1
      }
    }
  ]
}
```

POST /verify

Takes a photo, finds the face within it, and tries to compare it against a face you have already enrolled into a gallery.

To verify a face that you have enrolled in your gallery, all you need to do is submit a JPG or PNG photo. You can submit the photo either as a publicly accessible URL, Base64 encoded photo or as a file upload.

Next, specify which gallery and subject ID we should search against to compare. These are the same names you used previously during the /enroll calls to create the gallery.

Note that as long as the request is able to perform a match then you will receive a status of "success". You should use the "confidence" value to determine whether the comparison is valid for your application. If the confidence values in excess of 60% are almost always of the same person.

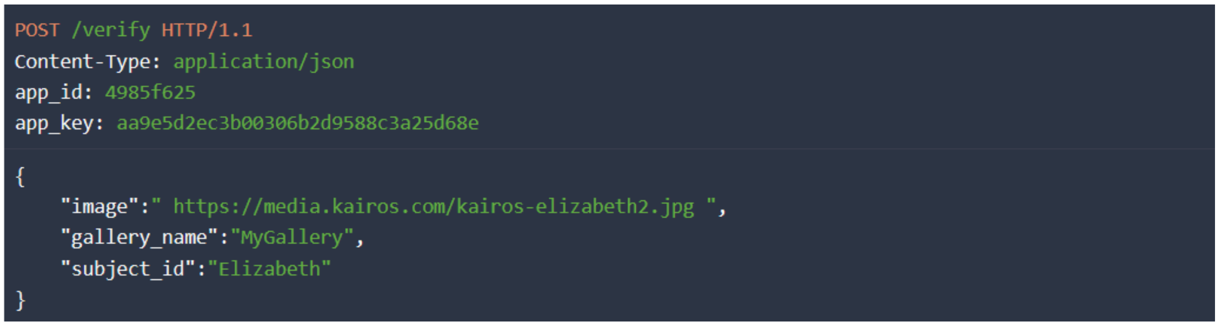
Required Parameters

image: Publicly accessible URL, file upload or Base64 encoded photo.
subject_id: Defined by you. Is used as an identifier for the face.
gallery_name: Defined by you. Is used to identify the gallery.
selector: LIVENESS used to predict spoof-attacks.

Example:

POST https://api.kairos.com/verify

Request



```
POST /verify HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "image":" https://media.kairos.com/kairos-elizabeth2.jpg ",
    "gallery_name":"MyGallery",
    "subject_id":"Elizabeth"
}
```

Response

```
200
Content-Type: application/json

{
  "images": [
    {
      "transaction": {
        "status": "success",
        "subject_id": "Elizabeth",
        "quality": 0.84705,
        "width": 170,
        "height": 287,
        "topLeftX": 108,
        "topLeftY": 55,
        "confidence": 0.88309,
        "gallery_name": "MyGallery"
      }
    }
  ]
}
```

Request (includes "liveness" selector)

```
POST /verify HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "image":"  https://media.kairos.com/kairos-elizabeth2.jpg  ",
    "gallery_name":"MyGallery",
    "subject_id":"Elizabeth",
    "selector":"liveness"
}
```

Response (includes "liveness" attribute)

```
200
Content-Type: application/json


{
    "images": [
    {
        "transaction": {
            "status": "success",
            "subject_id": "Elizabeth",
            "quality": 0.84705,
            "width": 170,
            "height": 287,
            "topLeftX": 108,
            "topLeftY": 55,
            "confidence": 0.88309,
            "gallery_name": "MyGallery",
            "liveness": 0.99091
        }
    }
    ]
}
```

Responses might also be like below:

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "subject ID was not found",
        "ErrCode": 5003
    }]
}
```

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "no faces found in the image",
        "ErrCode": 5002
    }]
}
```

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "too many faces in image",
        "ErrCode": 5010
    }]
}
```

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "no match found",
        "ErrCode": 5012
    }]
}
```

POST /recognize

Takes a photo, finds the faces within it, and tries to match them against the faces you have already enrolled into a gallery.

To match someone to a face enrolled in your gallery, all you need to do is submit a JPG or PNG photo. You can submit the photo either as a publicly accessible URL, Base64 encoded photo or as a file upload.

Next, specify which gallery the API should searchs against for matches. This is the same name you used previously during the /enroll calls to create the gallery.

The API also has the concept of a 'matching threshold', which by default is set at .60 (60%). If the face you submit is .60 similar to one or more faces in your gallery the API will return that as a list of potential candidates and how closely they match. If no one falls in that range the API will return no matches.

Depending on your usage, you may want to adjust the threshold lower or higher to return more or less potential candidates respectively.

*You can find out **more about thresholds**

Required Parameters

image: Publicly accessible URL, file upload or Base64 encoded photo.
gallery_name: Defined by you. Is used to identify the gallery.

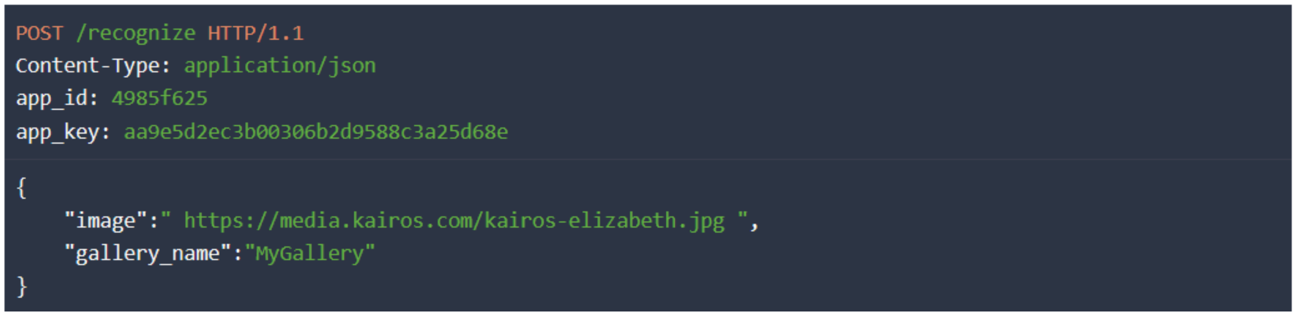
Optional Parameters

minHeadScale: Defined by you. Is used to set the ratio of the smallest face we should look for in the photo. Accepts a value between .015 (1:64 scale) and .5 (1:2 scale). By default it is set at .015 (1:64 scale) if not specified.
threshold: Is used to determine a valid facial match.
max_num_results Is the maximum number of potential matches that are returned. By default it is set to 10 if not supplied.
selector: LIVENESS used to predict spoof-attacks.

Example:

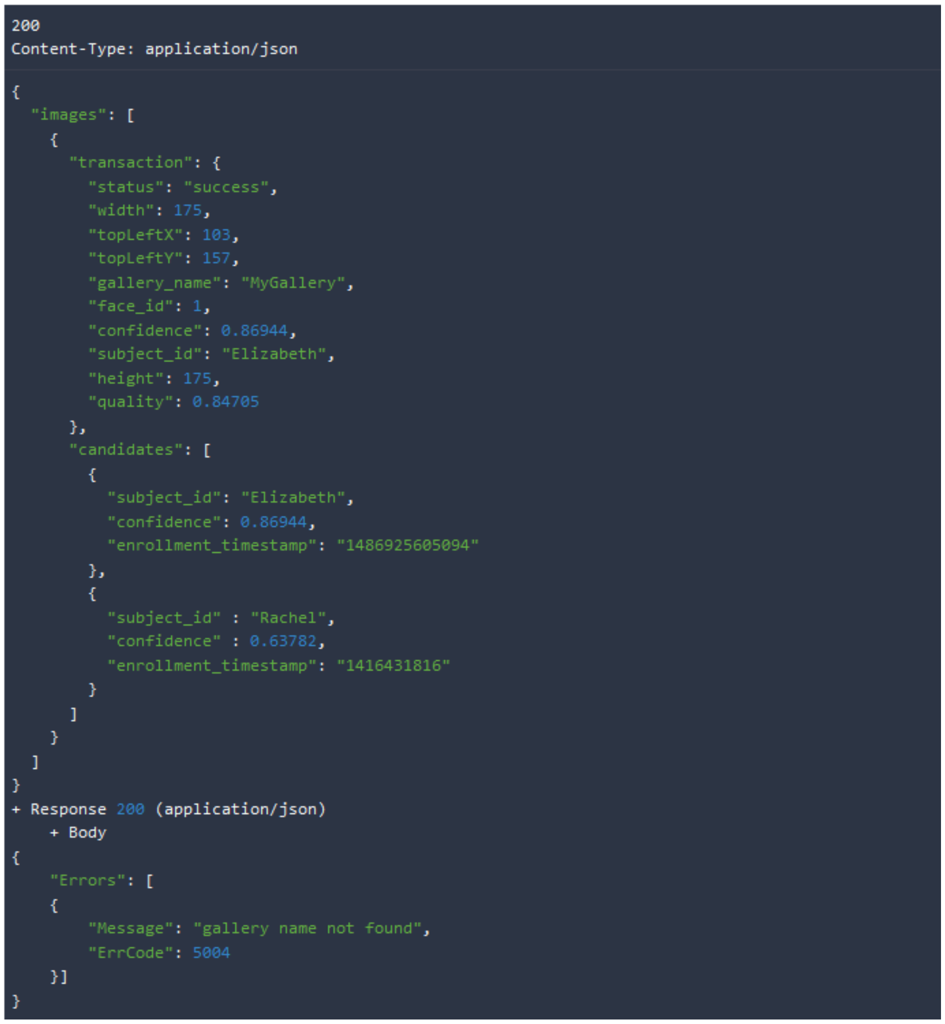
POST https://api.kairos.com/recognize

Request



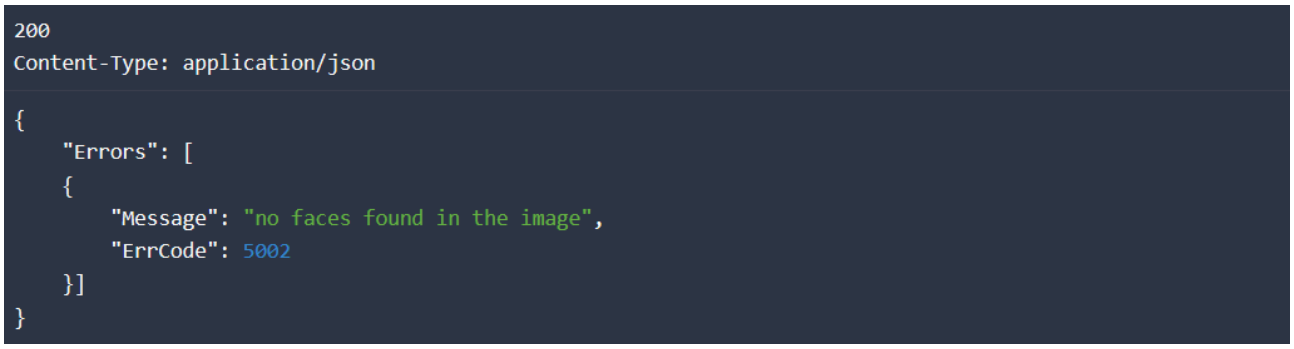
```
POST /recognize HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "image":" https://media.kairos.com/kairos-elizabeth.jpg ",
    "gallery_name":"MyGallery"
}
```

Response



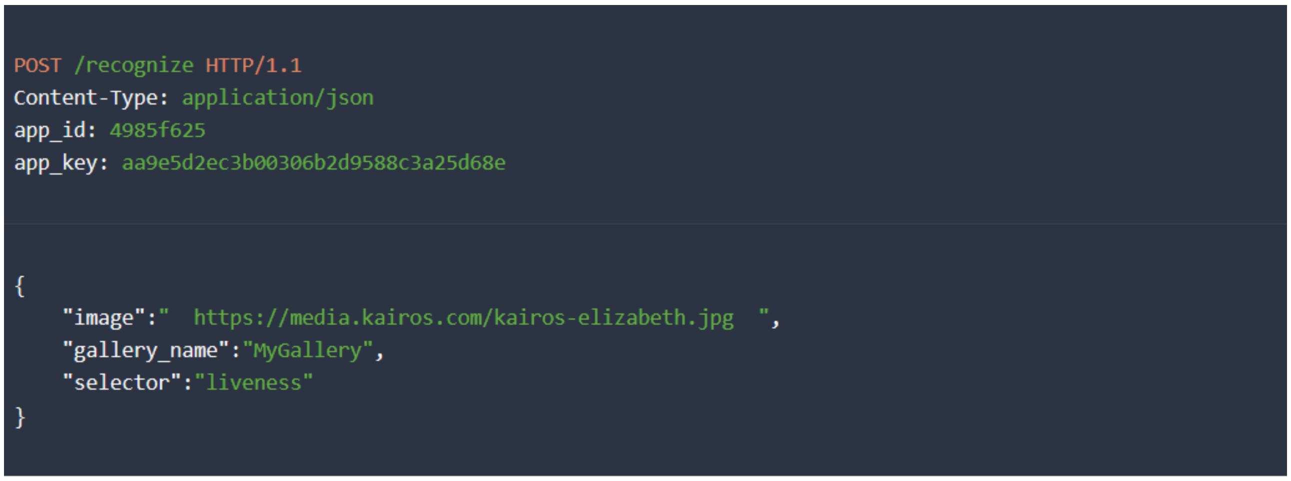
```
200
Content-Type: application/json

{
  "images": [
    {
      "transaction": {
        "status": "success",
        "width": 175,
        "topLeftX": 103,
        "topLeftY": 157,
        "gallery_name": "MyGallery",
        "face_id": 1,
        "confidence": 0.86944,
        "subject_id": "Elizabeth",
        "height": 175,
        "quality": 0.84705
      },
      "candidates": [
        {
          "subject_id": "Elizabeth",
          "confidence": 0.86944,
          "enrollment_timestamp": "1486925605094"
        },
        {
          "subject_id" : "Rachel",
          "confidence" : 0.63782,
          "enrollment_timestamp": "1416431816"
        }
      ]
    }
  ]
}
+ Response 200 (application/json)
    + Body
{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

The response might also be like:



```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "no faces found in the image",
        "ErrCode": 5002
    }]
}
```

Request (includes "liveness" selector)



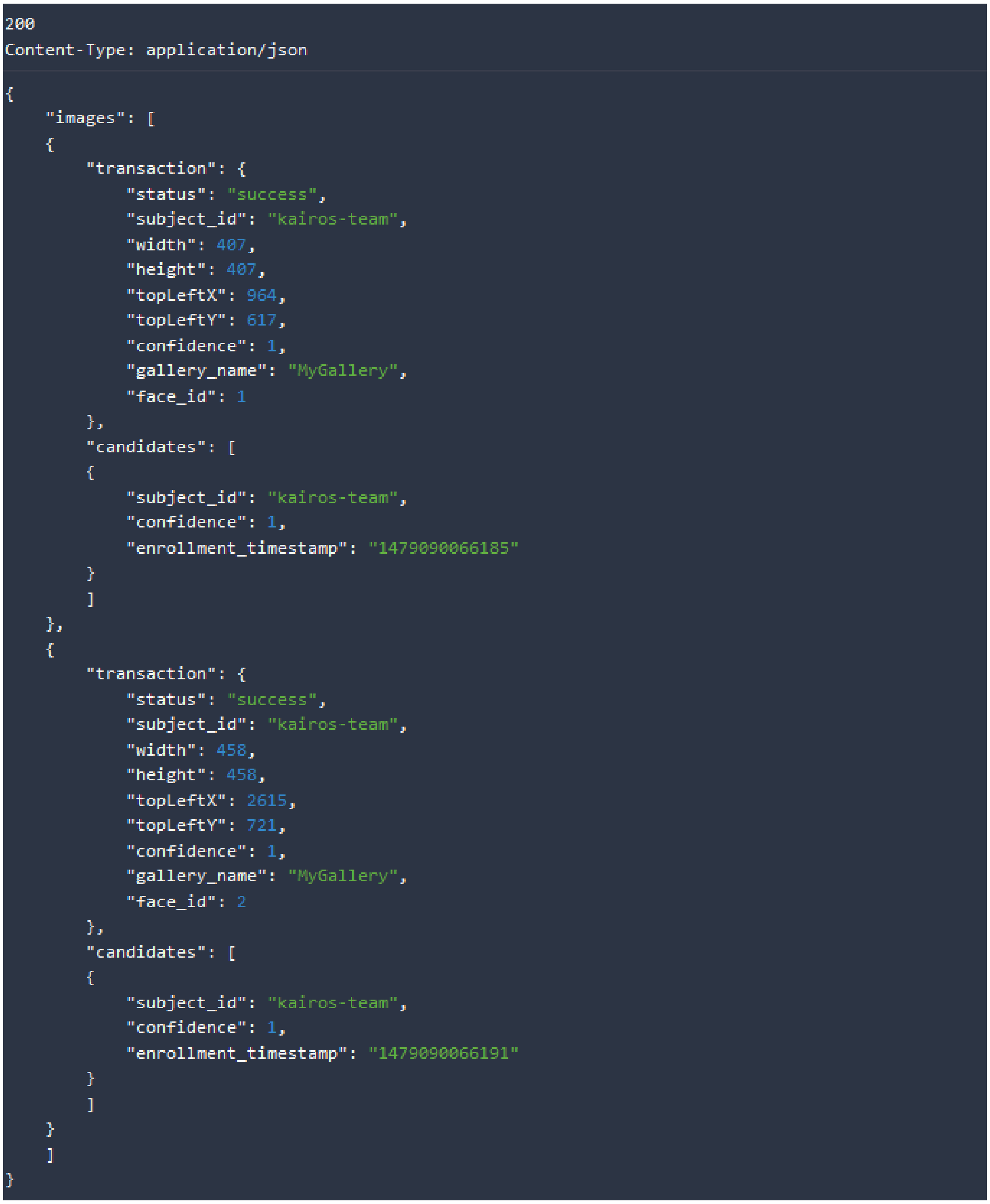
```
POST /recognize HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e


{
    "image":"  https://media.kairos.com/kairos-elizabeth.jpg  ",
    "gallery_name":"MyGallery",
    "selector":"liveness"
}
```

Response (includes "liveness" attribute)

```
200
Content-Type: application/json
```

```
{
    "images": [
      {
        "transaction": {
          "status": "success",
          "width": 175,
          "topLeftX": 103,
          "topLeftY": 157,
          "gallery_name": "MyGallery",
          "face_id": 1,
          "confidence": 0.86944,
          "subject_id": "Elizabeth",
          "height": 175,
          "quality": 0.84705,
          "liveness": 0.99091
        },
        "candidates": [
          {
            "subject_id": "Elizabeth",
            "confidence": 0.86944,
            "enrollment_timestamp": "1486925605094"
          },
          {
            "subject_id" : "Rachel",
            "confidence" : 0.63782,
            "enrollment_timestamp": "1416431816"
          }
        ]
      }
    ]
  }
```

Response:



```
200
Content-Type: application/json

{
    "images": [
    {
        "transaction": {
            "status": "success",
            "subject_id": "kairos-team",
            "width": 407,
            "height": 407,
            "topLeftX": 964,
            "topLeftY": 617,
            "confidence": 1,
            "gallery_name": "MyGallery",
            "face_id": 1
        },
        "candidates": [
        {
            "subject_id": "kairos-team",
            "confidence": 1,
            "enrollment_timestamp": "1479090066185"
        }
        ]
    },
    {
        "transaction": {
            "status": "success",
            "subject_id": "kairos-team",
            "width": 458,
            "height": 458,
            "topLeftX": 2615,
            "topLeftY": 721,
            "confidence": 1,
            "gallery_name": "MyGallery",
            "face_id": 2
        },
        "candidates": [
        {
            "subject_id": "kairos-team",
            "confidence": 1,
            "enrollment_timestamp": "1479090066191"
        }
        ]
    }
    ]
}
```

POST /detect

Takes a photo and returns the facial features it finds.

To detect faces, all you need to do is submit a JPG or PNG photo. You can submit the photo either as a publicly accessible URL or Base64 encoded.

Required Parameters

image Publicly accessible URL or Base64 encoded photo.

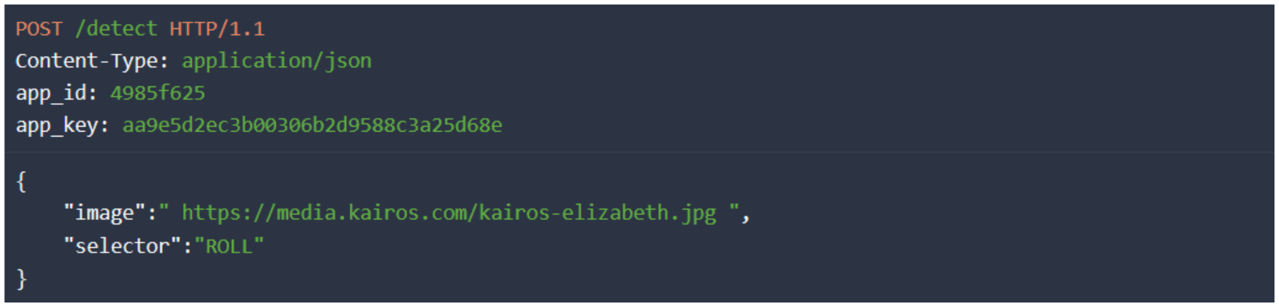
Optional Parameters

minHeadScale: Defined by you. Is used to set the ratio of the smallest face we should look for in the photo. Accepts a value between .015 (1:64 scale) and .5 (1:2 scale). By default it is set at .015 (1:64 scale) if not specified.
selector: Is used to adjust the face detector. If not specified the default of FRONTAL is used. LIVENESS used to predict spoof-attacks. Note that these optional parameters are not reliable for face recognition, but may be useful for face detection uses.

Example

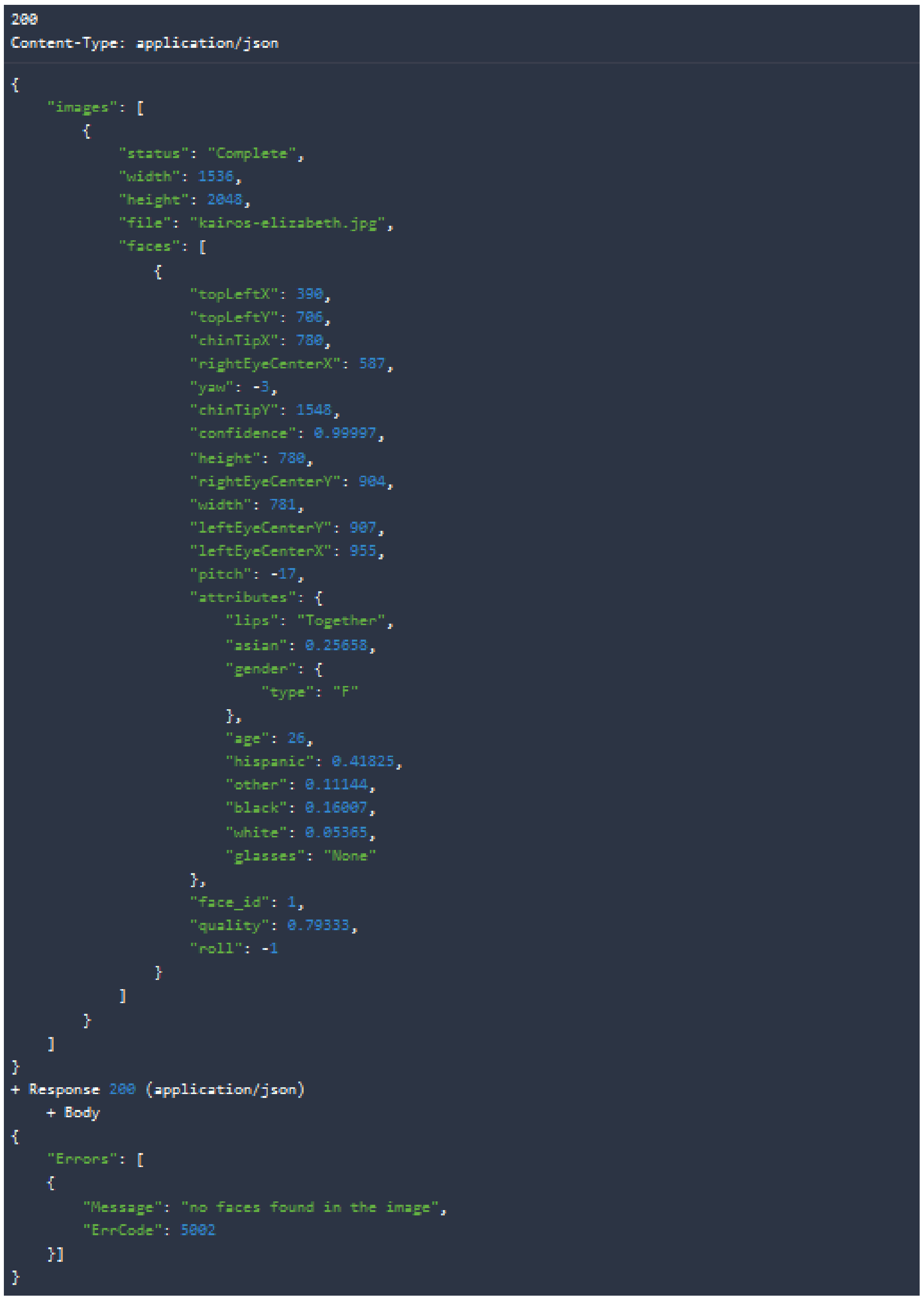
POST https://api.kairos.com/detect

Request



```
POST /detect HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "image":" https://media.kairos.com/kairos-elizabeth.jpg ",
    "selector":"ROLL"
}
```

Response

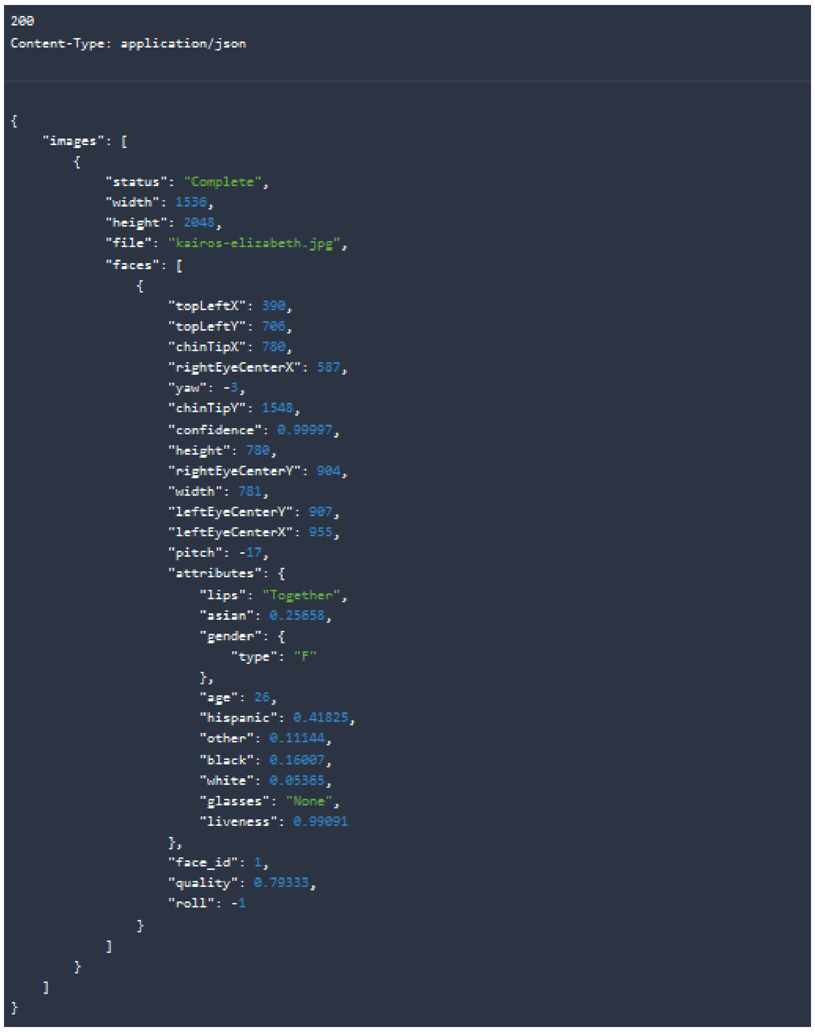


```
200
Content-Type: application/json

{
    "images": [
        {
            "status": "Complete",
            "width": 1536,
            "height": 2048,
            "file": "kairos-elizabeth.jpg",
            "faces": [
                {
                    "topLeftX": 390,
                    "topLeftY": 706,
                    "chinTipX": 780,
                    "rightEyeCenterX": 587,
                    "yaw": -3,
                    "chinTipY": 1548,
                    "confidence": 0.99997,
                    "height": 780,
                    "rightEyeCenterY": 904,
                    "width": 781,
                    "leftEyeCenterY": 907,
                    "leftEyeCenterX": 955,
                    "pitch": -17,
                    "attributes": {
                        "lips": "Together",
                        "asian": 0.25658,
                        "gender": {
                            "type": "F"
                        },
                        "age": 26,
                        "hispanic": 0.41825,
                        "other": 0.11144,
                        "black": 0.16007,
                        "white": 0.05365,
                        "glasses": "None"
                    },
                    "face_id": 1,
                    "quality": 0.79333,
                    "roll": -1
                }
            ]
        }
    ]
}
+ Response 200 (application/json)
    + Body
{
    "Errors": [
    {
        "Message": "no faces found in the image",
        "ErrCode": 5002
    }]
}
```

Request (includes "liveness" selector)

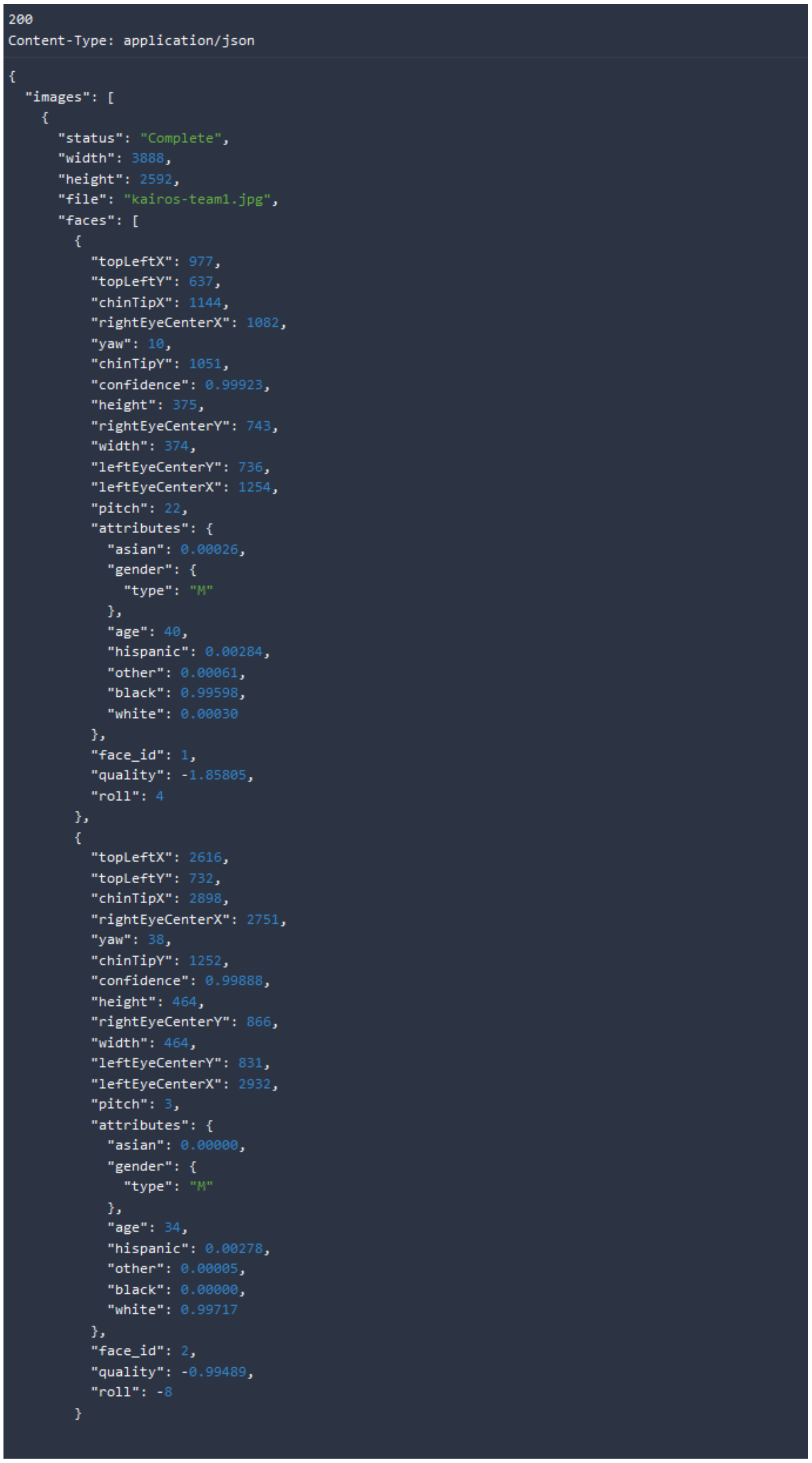
```
POST /detect HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "image":"  https://media.kairos.com/kairos-elizabeth.jpg  ",
    "selector":"ROLL, liveness"
}
```

Response (includes "liveness" attribute)



```
200
Content-Type: application/json

{
    "images": [
        {
            "status": "Complete",
            "width": 1536,
            "height": 2048,
            "file": "kairos-elizabeth.jpg",
            "faces": [
                {
                    "topLeftX": 390,
                    "topLeftY": 706,
                    "chinTipX": 780,
                    "rightEyeCenterX": 587,
                    "yaw": -3,
                    "chinTipY": 1548,
                    "confidence": 0.99997,
                    "height": 780,
                    "rightEyeCenterY": 904,
                    "width": 781,
                    "leftEyeCenterY": 907,
                    "leftEyeCenterX": 955,
                    "pitch": -17,
                    "attributes": {
                        "lips": "Together",
                        "asian": 0.25658,
                        "gender": {
                            "type": "F"
                        },
                        "age": 26,
                        "hispanic": 0.41825,
                        "other": 0.11144,
                        "black": 0.16007,
                        "white": 0.05365,
                        "glasses": "None",
                        "liveness": 0.99091
                    },
                    "face_id": 1,
                    "quality": 0.79333,
                    "roll": -1
                }
            ]
        }
    ]
}
```

Response



```
200
Content-Type: application/json

{
  "images": [
    {
      "status": "Complete",
      "width": 3888,
      "height": 2592,
      "file": "kairos-team1.jpg",
      "faces": [
        {
          "topLeftX": 977,
          "topLeftY": 637,
          "chinTipX": 1144,
          "rightEyeCenterX": 1082,
          "yaw": 10,
          "chinTipY": 1051,
          "confidence": 0.99923,
          "height": 375,
          "rightEyeCenterY": 743,
          "width": 374,
          "leftEyeCenterY": 736,
          "leftEyeCenterX": 1254,
          "pitch": 22,
          "attributes": {
            "asian": 0.00026,
            "gender": {
              "type": "M"
            },
            "age": 40,
            "hispanic": 0.00284,
            "other": 0.00061,
            "black": 0.99598,
            "white": 0.00030
          },
          "face_id": 1,
          "quality": -1.85805,
          "roll": 4
        },
        {
          "topLeftX": 2616,
          "topLeftY": 732,
          "chinTipX": 2898,
          "rightEyeCenterX": 2751,
          "yaw": 38,
          "chinTipY": 1252,
          "confidence": 0.99888,
          "height": 464,
          "rightEyeCenterY": 866,
          "width": 464,
          "leftEyeCenterY": 831,
          "leftEyeCenterX": 2932,
          "pitch": 3,
          "attributes": {
            "asian": 0.00000,
            "gender": {
              "type": "M"
            },
            "age": 34,
            "hispanic": 0.00278,
            "other": 0.00005,
            "black": 0.00000,
            "white": 0.99717
          },
          "face_id": 2,
          "quality": -0.99489,
          "roll": -8
        }
```

POST /gallery/list_all

Lists out all of the galleries you have created.

This method requires no parameters.

Example

POST https://api.kairos.com/gallery/list_all

Request

```
POST /gallery/list_all HTTP/1.1
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e
```

Response

```
200
Content-Type: application/json

{
    "status": "Complete",
    "gallery_ids": [
        "MyGallery",
        "testgallery1",
        "testgallery2",
        "testgallery3",
        "testgallery4"
    ]
}
```

POST /gallery/view

Lists out all of the faces you have enrolled in a gallery.
You just need to pass in the gallery_name and will receive back the list of subjects that you have enrolled within that gallery.

Required Parameters

gallery_name: Defined by you. Is used to identify the gallery.

Example:

POST https://api.kairos.com/gallery/view

Request

```
POST /gallery/view HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "gallery_name":"MyGallery"
}
```

Response

```
200
Content-Type: application/json

{
  "status": "Complete",
  "subject_ids": [
    "Elizabeth",
    "Rachel"
    ]
}
+ Response 200 (application/json)
    + Body
{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

POST /gallery/view_subject

Displays all face_id's and enrollment timestamps for each template you have enrolled from a given gallery_name and subject_id.

Required Parameters

subject_id: Defined by you. Is used as an identifier for the face.
gallery_name: Defined by you. Is used to identify the gallery.

Example

POST https://api.kairos.com/gallery/view_subject

Request

```
POST /gallery/view_subject HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "gallery_name":"MyGallery",
    "subject_id":"test1"
}
```

Response

```
200
Content-Type: application/json

{
    "status": "Complete",
    "face_ids": [
        {
            "face_id": "58f9034731c662745482",
            "enrollment_timestamp": "1492714311586"
        },
        {
            "face_id": "58f903481094d4122021",
            "enrollment_timestamp": "1492714312208"
        },
        {
            "face_id": "58f90347400a95737838",
            "enrollment_timestamp": "1492714312839"
        },
        {
            "face_id": "58f9034819f899628551",
            "enrollment_timestamp": "1492714312957"
        },
        {
            "face_id": "58f90347efcd88306283",
            "enrollment_timestamp": "1492714313011"
        }
    ]
}
+ Response 200 (application/json)
    + Body
{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

Response

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "subject ID was not found",
        "ErrCode": 5003
    }]
}
```

POST /gallery/remove

Removes a gallery and all of its subjects.
Pass your gallery_name and it will be removed along with all of its enrolled subjects.

Required Parameters

gallery_name: Defined by you. Is used to identify the gallery.

Example:

POST https://api.kairos.com/gallery/remove

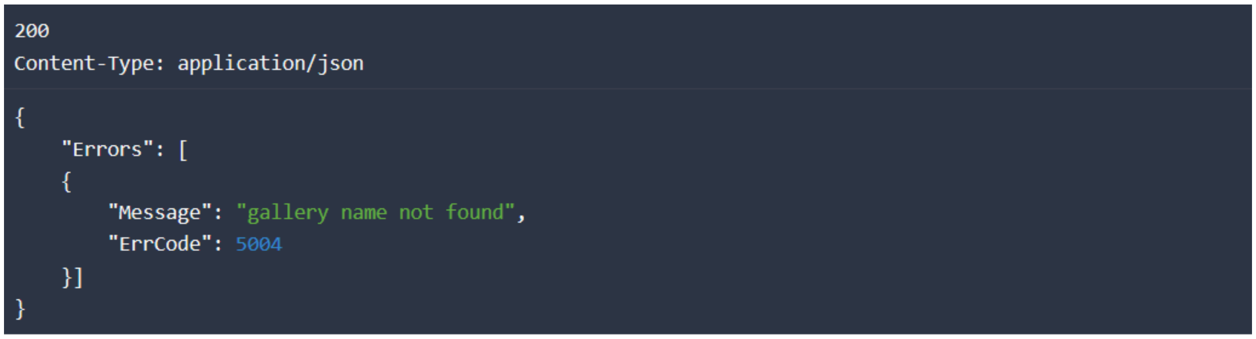
Request

```
POST /gallery/remove HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "gallery_name":"MyGallery"
}
```

Response

```
200
Content-Type: application/json

{
    "status":"Complete",
    "message":"gallery MyGallery was removed"
}
```

Response



```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

POST /gallery/remove_subject

Removes a face you have enrolled within a gallery.

Pass in a gallery_name and a subject_id and we will remove that subject from the gallery. Once the last subject is removed the gallery is removed automatically.

If the 'face_id' parameter is passed in the request, then the API will look for and only remove the individual enrolled template matching the corresponding 'face_id' but not all the matching subject_id's in the gallery.

Required Parameters

subject_id: Defined by you. Is used as an identifier for the face.
gallery_name: Defined by you. Is used to identify the gallery.

Optional Parameters

face_id:: A unique ID from the enroll output when an image has been enrolled.

Example

POST https://api.kairos.com/gallery/remove_subject

Request

```
POST /gallery/remove_subject HTTP/1.1
Content-Type: application/json
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e

{
    "gallery_name":"MyGallery",
    "subject_id":"test1"
}
```

Response

```
200
Content-Type: application/json

{
    "status": "Complete",
    "message" "subject id test1 has been successfully removed"
}
```

Response

```
200
Content-Type: application/json

{
    "status": "Complete",
    "message": "subject id test1 with face id 58f9034743ab64939482 has been successfully removed"
}
```

Response

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "gallery name not found",
        "ErrCode": 5004
    }]
}
```

Response

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "subject ID was not found",
        "ErrCode": 5003
    }]
}
```

Response

```
200
Content-Type: application/json

{
    "Errors": [
    {
        "Message": "subject id with face id was not found",
        "ErrCode": 5003
    }]
}
```

Emotion Analysis

Overview

Analyze and understand emotion, demographics and attention in most videos and images. We look for faces in your footage and pass the facial features and expressions through our face analysis algorithms. We will return values for the 6 universal emotions, age, gender, and other useful meta data about the faces found.

Base URL

```
api.kairos.com
```

Authentication

Requests must be authenticated with your API key. This must be sent as an HTTP header:

```
app_id:  "YOUR APP_ID"
app_key: "YOUR APP_KEY"
```

Requests

Requests return a JSON object with the header:

```
Content-Type: application/json
```

GET /v2/analytics

Returns the overall impressions from a specific uploaded piece of media.

Required Parameters

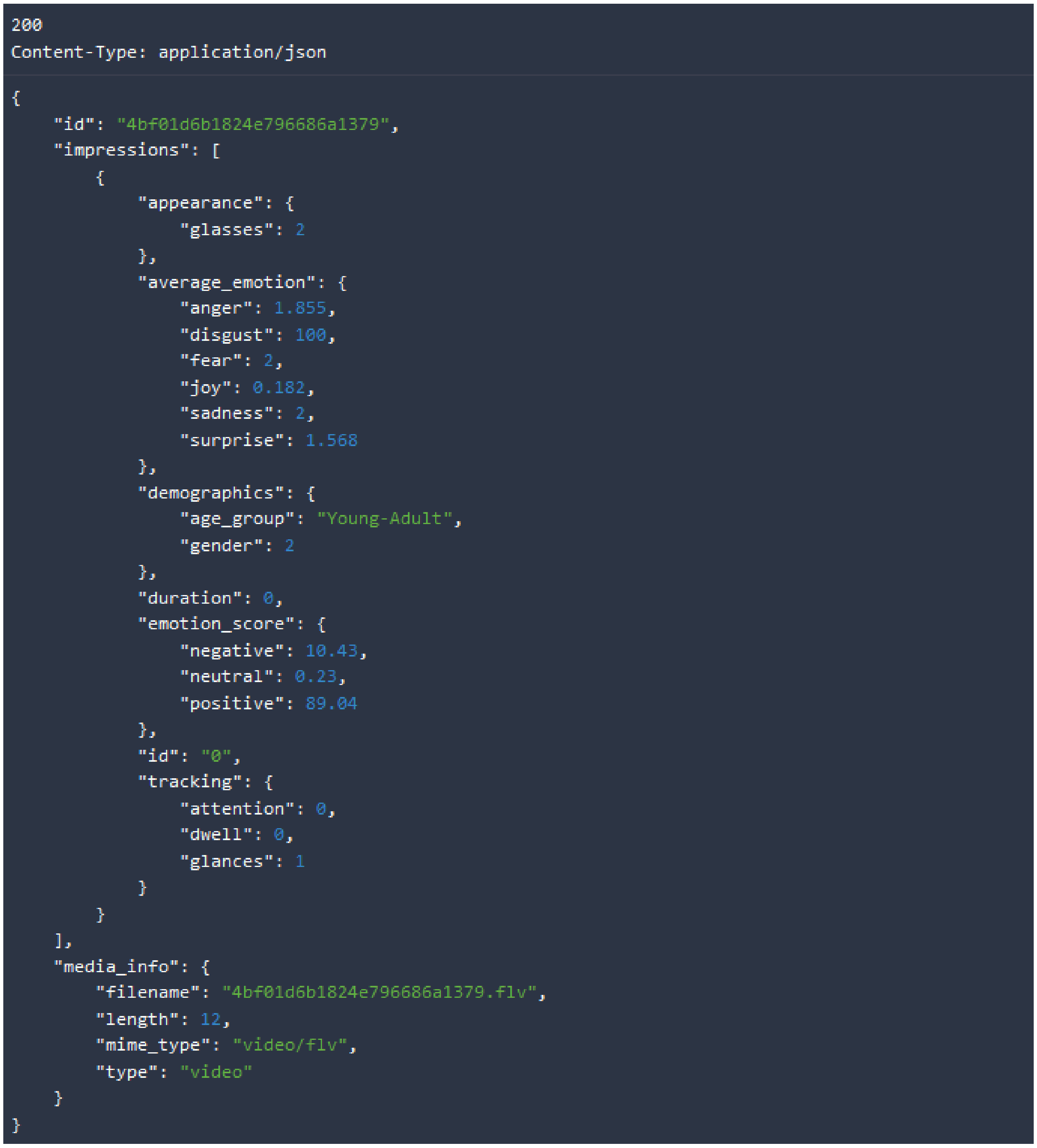
id: The id of the media.

Example

GET https://api.kairos.com/v2/analytics/ {id}
Request

```
GET /v2/analytics/{id} HTTP/1.1
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e
```

Response



```
200
Content-Type: application/json

{
    "id": "4bf01d6b1824e796686a1379",
    "impressions": [
        {
            "appearance": {
                "glasses": 2
            },
            "average_emotion": {
                "anger": 1.855,
                "disgust": 100,
                "fear": 2,
                "joy": 0.182,
                "sadness": 2,
                "surprise": 1.568
            },
            "demographics": {
                "age_group": "Young-Adult",
                "gender": 2
            },
            "duration": 0,
            "emotion_score": {
                "negative": 10.43,
                "neutral": 0.23,
                "positive": 89.04
            },
            "id": "0",
            "tracking": {
                "attention": 0,
                "dwell": 0,
                "glances": 1
            }
        }
    ],
    "media_info": {
        "filename": "4bf01d6b1824e796686a1379.flv",
        "length": 12,
        "mime_type": "video/flv",
        "type": "video"
    }
}
```

Response

```
200
Content-Type: application/json
```

```
{
    "id": "0123Test",
    "status_code": 3,
    "status_message": "Error. Media record not found."
}
```

POST /v2/media

Create a new media object to be processed.

Required Parameters

source: The source of the media. URL or file upload.

Optional Parameters

Landmarks: Set to 1 to receive the feature points, such as eyes, nose, mouth locations, in the JSON response.
timeout: Set timeout in seconds to wait for the media to be processed. Default timeout is 10 seconds. Max timeout allowed is 60 seconds.

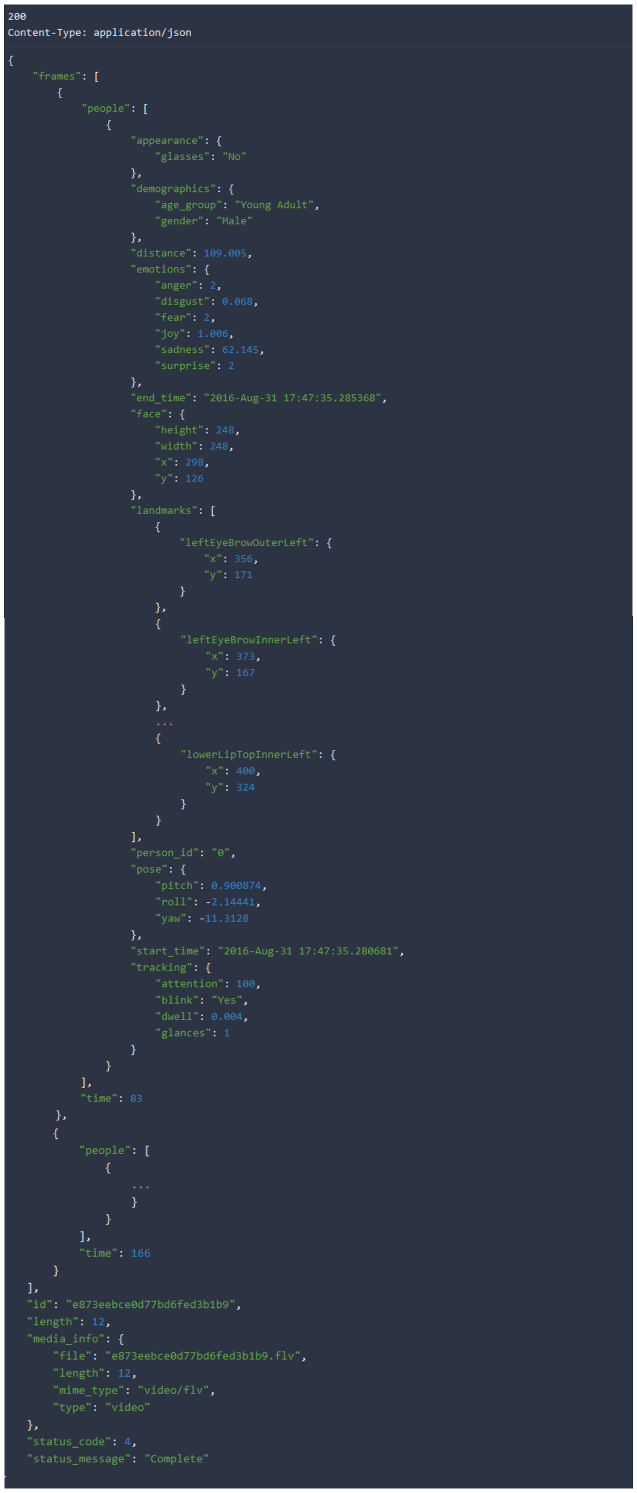
Example

POST https://api.kairos.com/v2/media {?source}

Request

```
POST /v2/media{?source} HTTP/1.1
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e
```

Response



```
200
Content-Type: application/json
```

```
{
    "frames": [
        {
            "people": [
                {
                    "appearance": {
                        "glasses": "No"
                    },
                    "demographics": {
                        "age_group": "Young Adult",
                        "gender": "Male"
                    },
                    "distance": 109.005,
                    "emotions": {
                        "anger": 2,
                        "disgust": 0.068,
                        "fear": 2,
                        "joy": 1.006,
                        "sadness": 62.145,
                        "surprise": 2
                    },
                    "end_time": "2016-Aug-31 17:47:35.285368",
                    "face": {
                        "height": 248,
                        "width": 248,
                        "x": 298,
                        "y": 126
                    },
                    "landmarks": [
                        {
                            "leftEyeBrowOuterLeft": {
                                "x": 356,
                                "y": 171
                            }
                        },
                        {
                            "leftEyeBrowInnerLeft": {
                                "x": 373,
                                "y": 167
                            }
                        },
                        ...
                        {
                            "lowerLipTopInnerLeft": {
                                "x": 400,
                                "y": 324
                            }
                        }
                    ],
                    "person_id": "0",
                    "pose": {
                        "pitch": 0.900874,
                        "roll": -2.14441,
                        "yaw": -11.3128
                    },
                    "start_time": "2016-Aug-31 17:47:35.280681",
                    "tracking": {
                        "attention": 100,
                        "blink": "Yes",
                        "dwell": 0.004,
                        "glances": 1
                    }
                }
            ],
            "time": 83
        },
        {
            "people": [
                {
                    ...
                }
            ],
            "time": 166
        }
    ],
    "id": "e873eebce0d77bd6fed3b1b9",
    "length": 12,
    "media_info": {
        "file": "e873eebce0d77bd6fed3b1b9.flv",
        "length": 12,
        "mime_type": "video/flv",
        "type": "video"
    },
    "status_code": 4,
    "status_message": "Complete"
```

Response

```
202
Content-Type: application/json

{
    "status_code": 2,
    "id": "012345678901234567890123",
    "status_message": "Processing"
}
```

Response

```
200
Content-Type: application/json

{
    "code": 1002,
    "message": "Required source parameter missing."
}
```

GET /v2/media

Returns the results of a specific uploaded piece of media.

Required Parameters

id The id of the media.

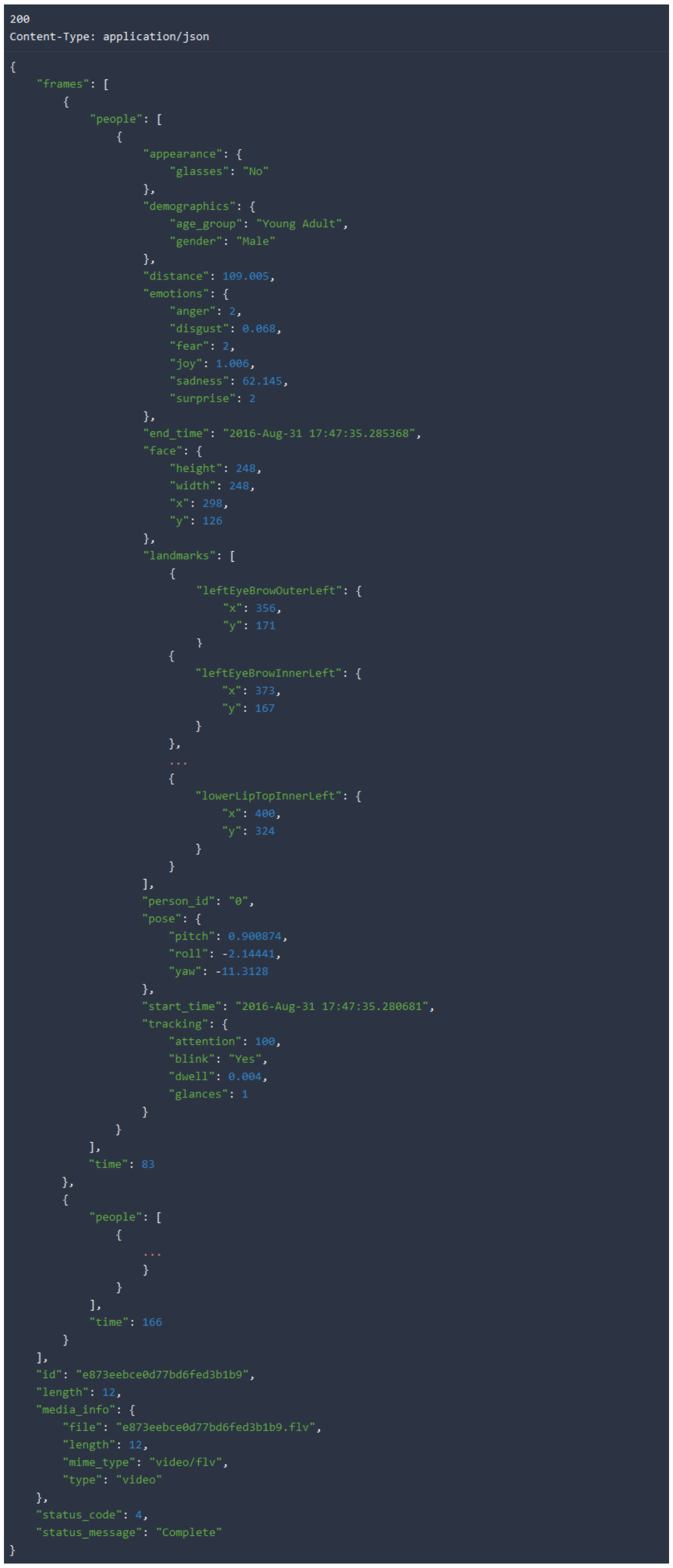
Example

GET https://api.kairos.com/v2/media/ {id}

Request

```
GET /v2/media/{id} HTTP/1.1
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e
```

Response



```
200
Content-Type: application/json
```

```json
{
    "frames": [
        {
            "people": [
                {
                    "appearance": {
                        "glasses": "No"
                    },
                    "demographics": {
                        "age_group": "Young Adult",
                        "gender": "Male"
                    },
                    "distance": 109.005,
                    "emotions": {
                        "anger": 2,
                        "disgust": 0.068,
                        "fear": 2,
                        "joy": 1.006,
                        "sadness": 62.145,
                        "surprise": 2
                    },
                    "end_time": "2016-Aug-31 17:47:35.285368",
                    "face": {
                        "height": 248,
                        "width": 248,
                        "x": 298,
                        "y": 126
                    },
                    "landmarks": [
                        {
                            "leftEyeBrowOuterLeft": {
                                "x": 356,
                                "y": 171
                            }
                        {
                            "leftEyeBrowInnerLeft": {
                                "x": 373,
                                "y": 167
                            }
                        },
                        ...
                        {
                            "lowerLipTopInnerLeft": {
                                "x": 400,
                                "y": 324
                            }
                        }
                    ],
                    "person_id": "0",
                    "pose": {
                        "pitch": 0.900874,
                        "roll": -2.14441,
                        "yaw": -11.3128
                    },
                    "start_time": "2016-Aug-31 17:47:35.280681",
                    "tracking": {
                        "attention": 100,
                        "blink": "Yes",
                        "dwell": 0.004,
                        "glances": 1
                    }
                }
            ],
            "time": 83
        },
        {
            "people": [
                {
                    ...
                    }
                }
            ],
            "time": 166
        }
    ],
    "id": "e873eebce0d77bd6fed3b1b9",
    "length": 12,
    "media_info": {
        "file": "e873eebce0d77bd6fed3b1b9.flv",
        "length": 12,
        "mime_type": "video/flv",
        "type": "video"
    },
    "status_code": 4,
    "status_message": "Complete"
}
```

Response

```
200
Content-Type: application/json

{
    "id": "0123Test",
    "status_code": 3,
    "status_message": "Error. Media record not found."
}
```

DELETE /v2/media

Delete media results. It returns the status of the operation.

Required Parameters

id: The id of the media.

Example

Request

```
DELETE /v2/media/{id} HTTP/1.1
app_id: 4985f625
app_key: aa9e5d2ec3b00306b2d9588c3a25d68e
```

Response

```
200
Content-Type: application/json

{
    "id": "9187207b6f1b41c681a8f529",
    "status_code": "5",
    "status_message": "Deleted"
}
```

Response

```
200
Content-Type: application/json

{
    "code": 5003,
    "message": "Unable to delete media."
}
```

Different methods for detecting emotion on videos and photos [29] [30]

On the table below you can see some articles which are about emotion detection on video and image.

Name	Type	Database	Published
Recognition of Emotion Intensities Using Machine Learning Algorithms: A Comparative Study	Image	Cohn-Kanade (CK), Extended CK (CK+), JAFFE, Bosphorus, BU-3DFE, RU-FACS, NVIE, MMI, DISFA, Belfast Induced Emotion, PAINFUL DATA	2019
MIMAMO Net: Integrating Micro- and Macro-motion for Video Emotion	Video	, OMG emotion dataset,the Aff-Wild dataset.	2019
M3ER: Multiplicative Multimodal Emotion Recognition using Facial, Textual, and Speech Cues	Video ,Text, Speech	IEMOCAP ,CMU-MOSEI	2019
Context-Aware Emotion Recognition Networks	Image and video	ContextAware Emotion Recognition (CAER)(http://caer-dataset.github.io)	2019
Facial Expression Recognition by De-expression Residue Learning	Image	BU-4DFE ,BP4Dspontaneous, CK+, Oulu-CASIA, MMI, BU3DFE, BP4D+.	2018
Deep Multi-Task Learning to Recognize Subtle Facial Expressions of Mental States	Image	e Wild database (LSEMSW), 300-W (landmark), n Oulu-Casia NIR&Vis, CK+	2018
Recognizing Facial Expressions Using Deep Learning	Image	Kaggle (Facial Expression Recognition Challenge) ,Karolinska Directed Emotional Faces	2017
Facial Expression Recognition using Visual Saliency and Deep Learning	Image	ILSVRC2012, CFEE, RaFD	2017
End-to-End Multimodal Emotion Recognition Using Deep Neural Networks	Video, Speech	RECOLA	2017
Image based Static Facial Expression Recognition with Multiple Deep Network Learning	Image	SFEW 2.0, FER	2015

Datasets for emotion detection [31]

Name of some available datasets which are used in emotion recognition projects is listed below

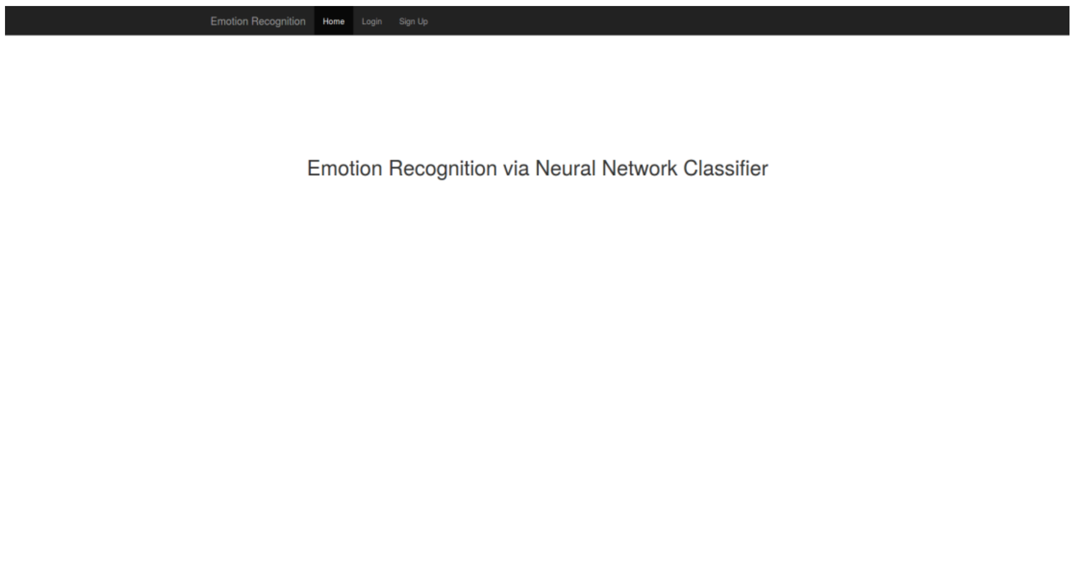
Name	Number of images/videos	Type	Gray/Color	Public available
Extended Cohn-Kanade Dataset (CK+)	593 images	Posed; spontaneous smiles	Mostly gray	Yes
Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)	7356 video and audio files	Posed	Color	Yes
Japanese Female Facial Expressions (JAFFE)	213 images	Posed	Gray	Yes
MMI Database[	1280 videos and over 250 images	Posed and Spontaneous	Color	Yes
Indian Spontaneous Expression Database (ISED)	428 videos	Spontaneous	Color	Yes
Radboud Faces Database (RaFD)	8040 images	Posed	Color	Yes
IMPA-FACE3D	534 images	Posed	Color	Yes
AffectNet	More than one million images	Wild setting	Color	Yes
DISFA	4,845 video frames	Spontaneous	Color	Yes
Aff-Wild	300 videos (over 2,000 minutes of data)	In-the-Wild setting	Color	Yes

* In posed expression databases, the participants are asked to display different basic emotional expressions, while in spontaneous expression database, the expressions are natural.

About the project

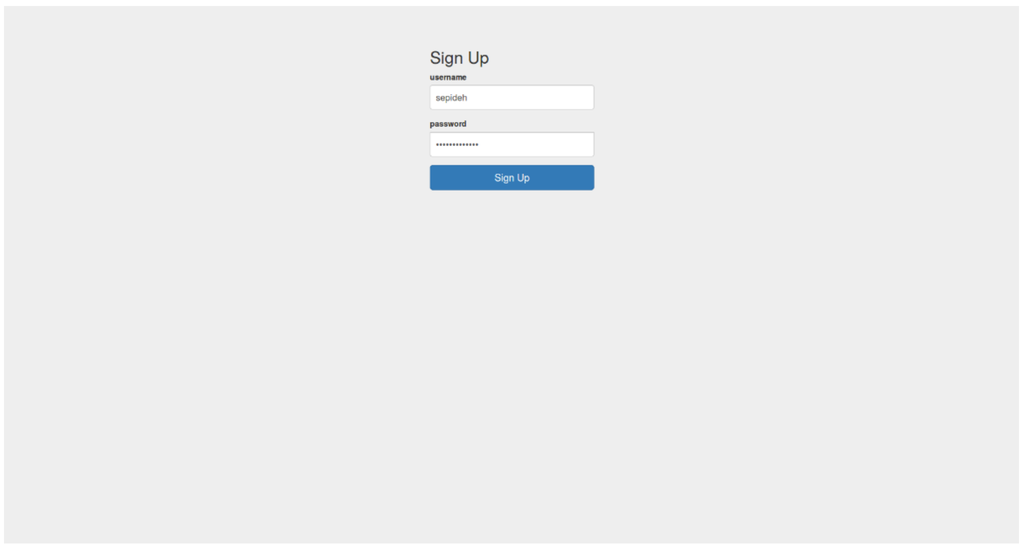
In this project I tried to develop a web server using Flask in which you can create an account for yourself, upload your photos and it will detect faces in the photo and predict the emotion of faces, it is able to detect up to four faces in the image.

I developed the detection part by using Neural Networks Classifiers. **haarcascade_frontalface_deafult.xml** for face detection and **chkPt1.h5** model and a json model for emotion detection in which you can find them in **models** folder which exists in the project folder

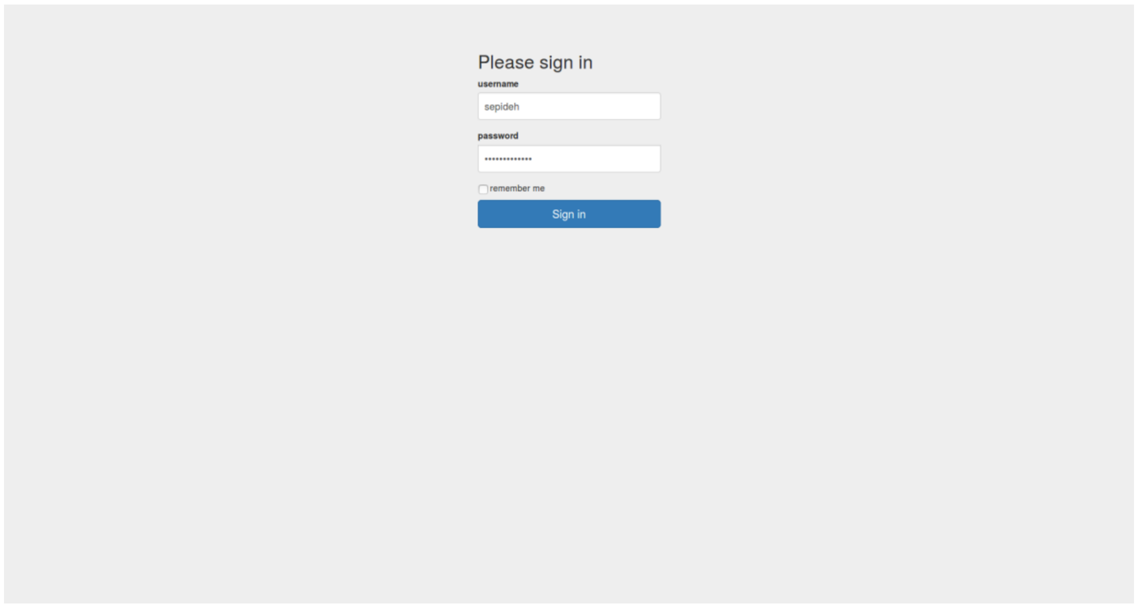


At first new user should create an account for him or herself.

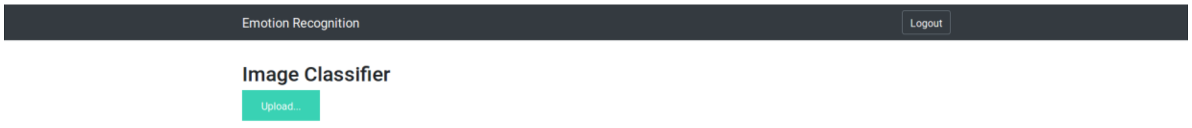
If the user already exists you will see the message **“User already exists”.** After creating an account for yourself you have to go to the login page.



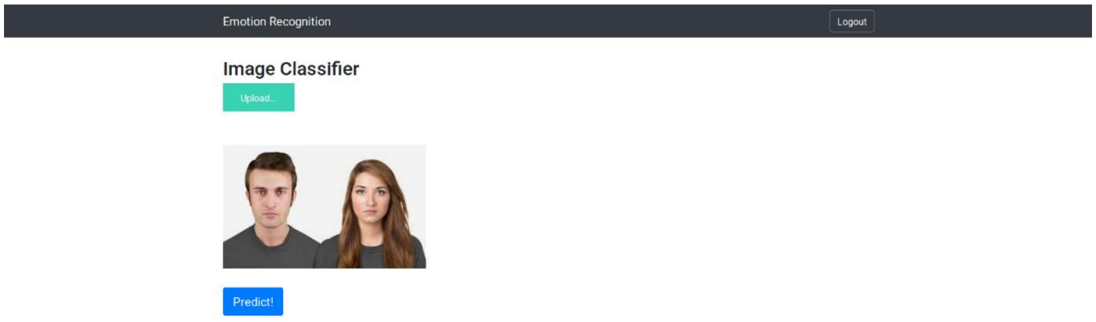
After creating an account you should go to the login page.



Now you are able to upload your photos for predicting emotions in the image.

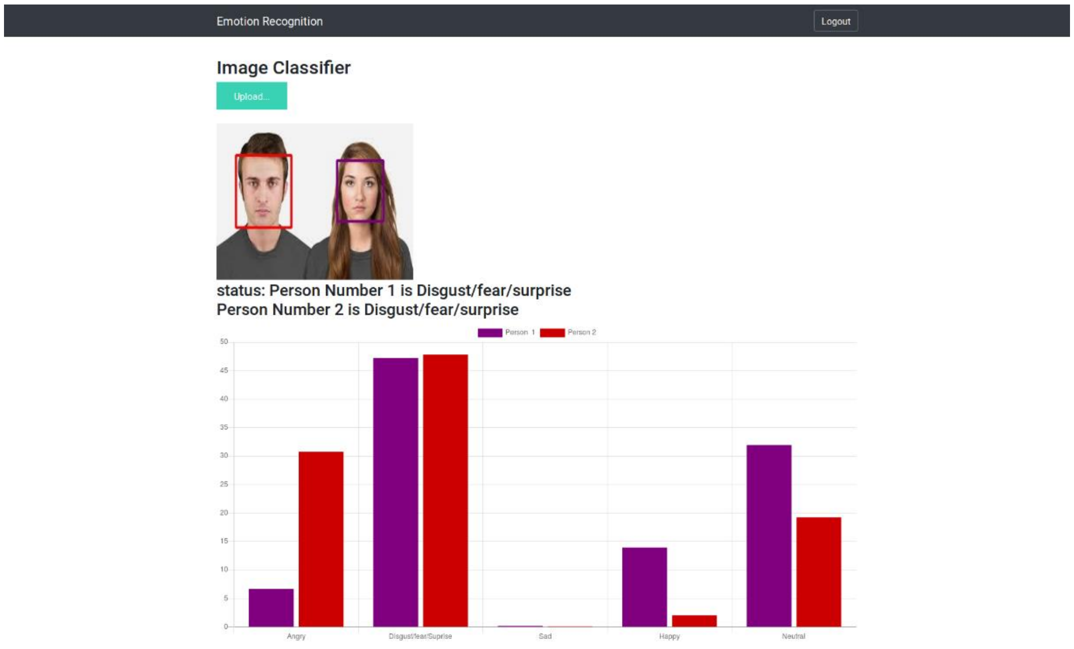


After uploading your photo “**Predict!” button** will be appeared.



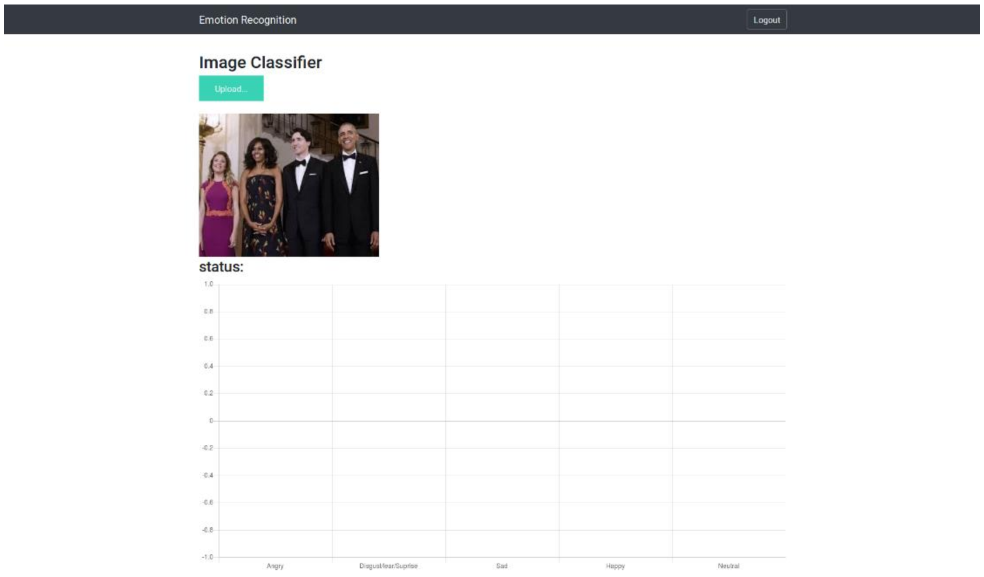
And finally by pushing the **Predict!** button you can see what are the emotion of faces in the image and also see the percentage of each emotion that the model would predict in a chart.

For drawing bar chart I used **Chart.js** which is an open source JavaScript library on **Github** that allows you to draw different types of charts by using the HTML5 canvas element.

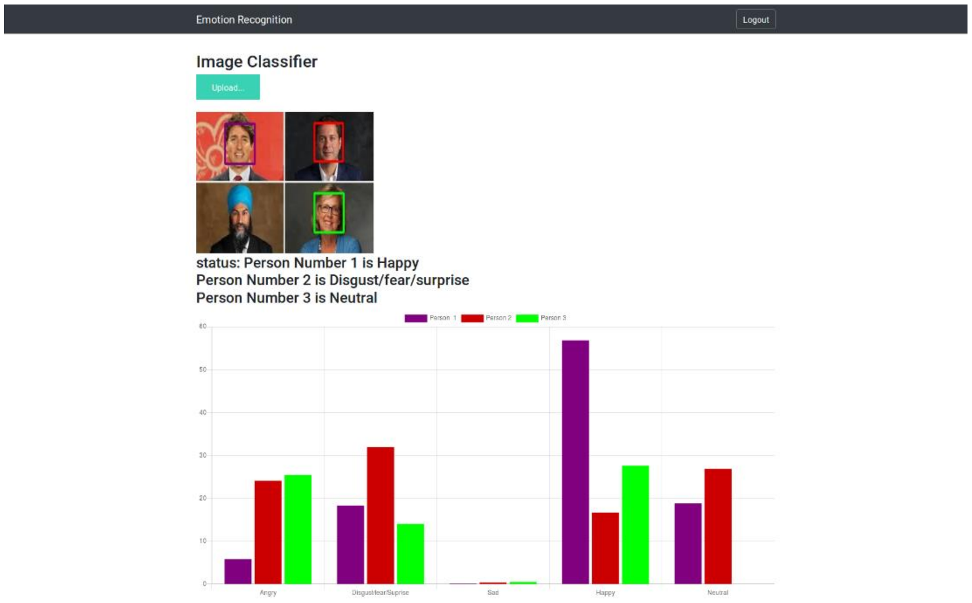


status: Person Number 1 is Disgust/fear/surprise
Person Number 2 is Disgust/fear/surprise

The model which used for emotion detection is not very accurate and sometimes cannot recognize some faces in photos.

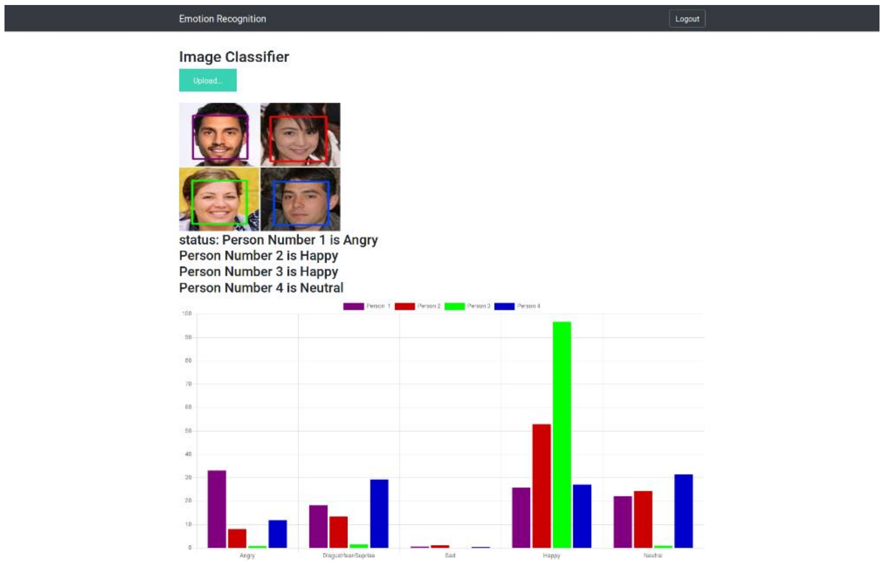


status:



status: Person Number 1 is Happy
Person Number 2 is Disgust/fear/surprise
Person Number 3 is Neutral

The other problem of this model is that sometimes does not predict emotions correctly.



status: Person Number 1 is Angry
Person Number 2 is Happy
Person Number 3 is Happy
Person Number 4 is Neutral

As it can be seen the model predicted that Person 1 emotion is angry. [1][2][3][4][5]

Source code of the project is available at: **https://github.com/sepideh2020/detection**

References:

[1] Kawalkar, S, 2018,Emotion-Recognition---Neural-Networks,https://github.com/shreyashk09/Emotion-Recognition---Neural-Networks

[2] Markcole,R, 2019, keras-flask-deploy-webapp,https://github.com/robmarkcole/keras-flask-deploy-webapp

[3] Grinberg,M, November 28 2013, RESTful Authentication with Flask,https://blog.miguelgrinberg.com/post/restful-authentication-with-flask

[4] Helbert,A, March 2017,Build a User Login System With Flask-Login, Flask-WTForms, Flask-Bootstrap, and Flask-SQLAlchemy, ,https://www.youtube.com/watch?v=8aTnmsDMldY

[5] Simple yet flexible JavaScript charting for designers and developers, Bar, https://www.chartjs.org/docs/latest/charts/bar.html

[6] Emotion recognition, https://en.wikipedia.org/wiki/Emotion_recognition

[7] An Emotion Recognition API for Analyzing Facial Expressions, Kim,S, December15,2016, https://algorithmia.com/blog/emotion-recognition-api-analyzing-facial-expressions

[8] Emotion recognition using facial expressions, June 2017, Introduction, Tarnowski,P, Kołodziej, M, Majkowski,A, J. Rak,R, https://www.sciencedirect.com/science/article/pii/S1877050917305264

[9] 20+ Emotion Recognition APIs That Will Leave You Impressed, and Concerned, Doerrfeld,B, December 31,2015, Updated on September 26th, 2019,

https://nordicapis.com/20-emotion-recognition-apis-that-will-leave-you-impressed-and-concerned/

[10] Emotient API, https://www.programmableweb.com/api/emotient

[11] Affectiva,https://en.wikipedia.org/wiki/Affectiva

[12] EmoVu API,https://www.programmableweb.com/api/emovu

[13]Advancing human potential, https://www.linkedin.com/company/nviso

[14] Kairos, https://www.predictiveanalyticstoday.com/kairos/

[15] Noldus FaceReader REST API, https://www.programmableweb.com/api/noldus-facereader-rest-api

[16] Sightcorphttps://builders.intel.com/ai/membership/sightcorp

[17] Face++ API –SDKs, https://www.programmableweb.com/api/face/sdks

[18] Imotions, https://imotions.com/biosensor/api/

[19] Implement the solution that fits your business, https://emotionresearchlab.com/use-cases/

[20] Driver Emotion Recognition and Real Time Facial Analysis for the Automotive Industry,Mcmanus,A,2017,https://blog.affectiva.com/driver-emotion-recognition-and-real-time-facial-analysis-for-the-automotive-industry

[21] 10 startups pioneering the new field of emotional analytics,Rebak,G,February19,2017,https://www.geektime.com/2017/02/19/10-startups-pioneering-the-new-field-of-emotional-analytics/

[22] Affectiva, https://www.affectiva.com/

[23]NVISIO,https://www.nviso.ai/en/healthcare-ai

[24] IMOTIONS, https://imotions.com/

[25]Realeyes https://www.realeyesit.com/technology/how-it-works/

[26]]Microsoft Cognitive Services

https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395236

[27] Microsoft Azure,https://azure.microsoft.com/en-us/services/cognitive-services/face/

[28] Developing with Kairos, https://www.kairos.com/docs/api/

[29] awesome-affective-computing,Kaiyd,A,https://github.com/AmrMKayid/awesome-affective-computing

[30] AWESOME-FER, Fan,E,https://github.com/EvelynFan/AWESOME-FER

[31]Facial expressiondatabases,

https://en.wikipedia.org/wiki/Facial_expression_databases