INTRO

We are developing a security system that utilizes facial recognition technology for safeguarding high-security environments such as secret research laboratories which have confidential information or machineries. We have implemented a hierarchy system where if a person has enough authorisation, they are able to attend to different tasks. The facial recognition system uses a database to store the data which it uses to identify the scanned face and then grant or reject access depending on the authorisation level and the task.

There are a few restrictions to granting access, such as the face has to be facing the front, and has to be clear and such.

We have also implemented an error system and security alert system that are used when something goes wrong or when certain conditions are not met. The security alert automatically shuts the system and does not let you try again. The error shows a retry message that gives the user three tries(set as flags) to access the system and if the user does not gain access, the system does not allow for more tries.

How it Works:

Our security system takes into account 6 criterias for recognizing faces. The criterias are:

1. Whether or not the person has a beard.
2. Whether or not the person has a mustache.
3. Whether or not the person has spectacles.
4. Skin color
5. Hair color
6. Eye color

Also our system requires the following criterias to be able to grant access without any chances of misconduct:

1. No signs of intoxication
2. No signs of sleepiness
3. Significant amount of light
4. Face facing towards the camera
5. Only one person in the camera frame
6. Clear face

Now all of these things can only be scanned with the help of a camera and supporting software which we do not have. So instead of getting information from the software about the above mentioned criteria, in our code the above criteria will be input from the user.

We have made a hypothetical database as well which would allow you to run the code and test it.

Propositions:

Auth1: returns true if a person has the highest level of authorisation in database1

Auth2:returns true if a person has the middle level of authorisation in database1

Auth3: returns true if a person has the lowest level of authorisation in database1

Clear_face: returns True if the face is clear/ not hidden, otherwise false.

Light: returns True if the level of light in the surrounding is above the threshold, otherwise false.

Face_in_front: returns True if the face is facing in the front, otherwise false.

One_person_frame: returns True if there is only one person in the frame, otherwise false.

Access_granted`: returns True if access is granted.

Flag = 0

Add_flag: adds 1 to Flag

Flag_highest: returns True if flag = 3

Show_error: returns true if error

Check_database: returns true if scanned face is present in database1

Intoxicated: returns True if person is intoxicated

Sleepy: returns True if person is sleepy

New_face: returns True if new face button is pressed

New_face_adding: returns true if new face is being added

Software_error: returns true if there is software error

Face_scanned: returns true if face is scanned

Meltdown: returns true if meltdown button is pressed

Meltdown_initiated: true if meltdown started

Retry: returns retry message

Security_alert: returns true if security is alerted

Database1: database consisting of all the faces of employees along with their authentication levels

System_shut: returns true if system is shut down

Spectacles: returns true if image in database and scanned photo both have spectacles or both don't have spectacles.

Hair_Colour: returns true if image in database and scanned photo both have same hair color.

Skin_Color: returns true if image in database and scanned photo both have same skin color.

Mustache: returns true if image in database and scanned photo both have a mustache or both don't have a mustache.

Beard: returns true if image in database and scanned photo both have a beard or both don't have a beard.

Eye Color: returns true if image in database and scanned photo both have same eye color.

EmployeeID = a number input by the user

Initial_access: Returns true if employee ID exists in the database.

Constraints:

Authorisation constraints -:

The person who has the highest authorisation can add a new face and initiate meltdown.

1. New_face & Auth1 >> New_face_adding
2. Meltdown & Auth1 >> Meltdown_initiated

The person with the second level of authorisation can only add a new face

3. New_face & Auth2 >> New_face_adding
4. Meltdown & Auth2 >> ~Meltdown_initiated
5. ~Meltdown_initiated >> Security_alert

The person with the lowest authorisation can access only the building and rooms

6. New_face & Auth3 >> ~ New_face_adding
7. Meltdown & Auth3 >> ~Meltdown_initiated
8. ~Meltdown_initiated >> Security_alert

Scan Constraints-:

While scanning if the face is clear, there is plenty of light, only one person is in the frame and the person is looking at the camera

1. Clear_face & Light & Face_in_front & One_person_frame & initial_access >> Face_scanned

If any one of the conditions mentioned in the first point is not fulfilled then one is added to the error record and an error message is displayed

2. ~(Clear_ face & Light & Face_in_front & One_person_frame) & ~Flag_highest >> Add_flag & Show_error

If any one of the conditions mentioned in the first point is not fulfilled and then one is added to the error record and an error message is displayed

3. ~(Clear_face & Light & Face_in_front & One_person_frame) & Flag_highest >>

   Show_error & Security_alert

To check for access-:

Check_database would give true only if the face is matched to the criterias.

1. Spectacles & Hair_Colour & Skin_Color & Mustache & Beard & Eye Color >> Check_database

access is only granted if face is scanned checked against the database and no error occurs

2. Face_scanned & ~Show_error & Check_database >> Acess_granted

if there is a software error then everything will be scanned and checked but access won't be granted

3. Face_scanned & ~Show_error & Check_database >> ~Acess_granted

To inform security-:

if an error occurs three times the security is alerted.

1. Flag_highest >> Security_alert & System_shut

if a person looks sleepy or intoxicated, access is not granted and security is alerted.

2. Sleepy | Intoxicated >> ~Acess_granted & Security_alert

If the software has errors:

1. Same scan error
   If a person with his face already in database, tries to put it again, error would show

New_face_adding & Check_database >> show error

Model Exploration:

For our project, we started out with a simple facial recognition system that recognises faces and grants access to the people who are recognised. We thought that this was a very simple idea and needed more complications, hence we added a few more databases. The purpose of more databases was to allow people access even if the face is on the side or if the eyes are closed etc. Then after we got the feedback, we were told to complicate our project as it did not have any interesting features. We brainstormed and gave our project a new outlook. Now we present to you, a high-security system that utilizes facial recognition technology for protecting any secretive location like high end research laboratories. With our security system, an organization can prevent trespassing by unauthorized personnel, give authority to new people joining the organization and even demolish the laboratory in case of an emergency. Our security system comes with high alerting systems which get activated as soon as something suspicious is detected.