Ingénierie des Systèmes d'Information
Vol. 26, No. 2, April, 2021, pp. 201-209
Journal homepage: http://iieta.org/journals/isi

An Optimised Allotment and Tracking Using Django and Opencv

Mohan Goud Kathi*, Jakeer Hussain Shaik

Vignan's Foundation for Science, Technology and Research, Vadlamudi, Guntur 522213, A.P., India

Corresponding Author Email: kathi.mohangoud@gmail.com

https://doi.org/10.18280/isi.260207

Received: 4 September 2020
Accepted: 23 March 2021

Keywords:
computer vision, Opencv, tracking, face detection, face recognition

ABSTRACT

Developments in tracking of objects is one of the key breakthroughs in computer vision in recent years. A video is a continuous flow of frames. By analysing the difference between a frame to its successive, one can estimate the movement of object. In this paper, the movement of person is detected with the help of two cameras facing opposite to each other. The detected persons face is recognized with the faces in the database, his data about the movement is updated in the excel sheet from time to time and the same is applied in the real-world problem of vigilance on invigilators in the examination hall. Examinations are inescapable to conclude a course. Devising the students and watchdogs into the examination halls are the part of proper conduction of Examinations. One of the key things in the administration of examinations is the issuance of invigilations. The faculty act as invigilators. It is arduous to mull over his/her designation and experience while earmarking the invigilations manually. This paper presents a cybernetic and contemplative method of dole out using Django and tracking the movements of each invigilator using Opencv.

1. INTRODUCTION

Any course in the world origins with admission and ends with an examination. In order to avoid malpractice at the examination, a superintendent system is imperative which is the invigilation system. Virtual invigilation [1] abstains human negligence and assures reliable monitoring but it requisites a HD camera in each and every examination hall and reliability over that system is also under suspicion. Due to this most of the firms still following the invigilation based monitoring system and all the invigilations are allocated manually with manpower. If the apportion is made with software then the allotment errors can eliminate and can reduce the manpower.

The adaptations of computer vision techniques into the invigilation system can make it more advanced. The advancements inface detection and face recognition made it involve in invigilation attendance system. But no much research in tracking the invigilator or student in catching the malpractice.

The face tracking is possible by extracting the face features [2]. There were different methods available for tracking [3]. With the change in the pose, the tracking becomes difficult, an efficient tracking model can able to track at different orientations [4]. Not only pose, but the tracking must also accurate at varying illumination conditions [5]. It is important to address when to start or stop tracking and needs to consider the detector efficiency. Hence a proper management system is essential in long-term tracking [6]. Dornaika and Davoine [7] proposed a framework for tracking face pose and local motion of inner parts. In real-world, the situations arise for robots to follow the human, the detection of human by skin color and implementation of tracking by color [8] can make it achievable. The CNN-based methods also prove their effectiveness in face tracking [9]. Wang et al. [10] propose a method of tracking to use in wireless sensor networks.

This paper concentrates on invigilator allotment and tracking. For tracking invigilator per room, each room needs a processing system along with a camera. Raspberry pi is used as a processing device. In order to ensure the right person is in motion, face recognition is implemented. In order to find the person from both ends two processing system along with cameras are implemented at both the opposite ends per room.

The central server allotted the invigilators as per the Room's strength and the implementation is presented in section 2 and it sends the invigilator list with the image to client devices placed inside the room. The client devices validate the invigilator using Face recognition technique and track and update their movements. The Tracking and Face recognition implementation is presented in sections 3, 4 and 5.

2. IMPLEMENTATION

Django is used for the allotment purposes. It is fast, secure and flexible. It provides database API. Instead of SQL, python can use to create, update and delete the data of the database. Since Guido Van Rossum [11] clasped different features from different languages into python while developing it, it is easy for developers to flourish an application using python.

Django can use to make Raspberry Pi a server [12]. Raspberry Pi is a tiny computer acquainted to the world in 2012. Even though, the purpose of it is to assist in education [13]. Because of its versatility, it is utilizing in sundry applications [14] like domicile security [15], Healthcare system [16], Agriculture [17], Surveillance system [18], Speech to text-based authentication [19], Artificial Intelligence [20], Noise level Monitoring [21], Smart cities [22] etc. Howbeit, practising it in micro applications is a misapply. For example, it is rich in features when juxtaposed with Arduino [13] but is not best suited for hoarding sensor data than Arduino because of its cost and the absence of ADC. The best applications where the raspberry pi can practice are server-based applications because of its underpin of various OSes [23]. Some of the server applications in which raspberry pi is utilizing are the online tests from mobile in a university without the internet [24], using as a server for an IoT application [25].

Django can also, use in other single board computers like Beagle bone black [26]. Hence it is facile to change to any other single board computer if there is a requisite of resources.

Initially, all the faculty details containing Name, Department and Designation are stored into the database by creating a project and app [23] and a class named faculty details on models.py [23]. Django provides an Admin interface [23], using which the faculty details can insert into the database. So, admin interface is enabled and faculty details class is registered into the admin. Also, to allow any others to add the data, a POST method is created using the meta class. This meta class is different from HTML's Meta class and python's Meta class.

The post form is not created using HTML but using Django forms [27]. Because of disadvantages using pure HTML forms. There is a massive security hole because there is no data validation. Django provides CSRF protection [27] for the POST methods. Also, it is hard to make the form using HTML than Django. Widgets are the way of creating HTML without writing it. Django built-in widgets provide all forms of input like text input, URL input, password input, hidden input and so on.

There is a possibility that the duplicates are submitted. The parameters are Name, Department and Designation. Using the unique field option [27], the required field is made unique throughout the table. Since under the same department there can be many faculties, it can't be made unique. Similarly, the designation is also not unique. The Name is unique for the same department and designation but the unique option makes the Name unique throughout the table without considering the department and designation. So unique_together [27] with Name, Department and Designation refrains the replication of data. While allotting invigilations, the user is bestowed with three choices which is illustrated in Figure 1.

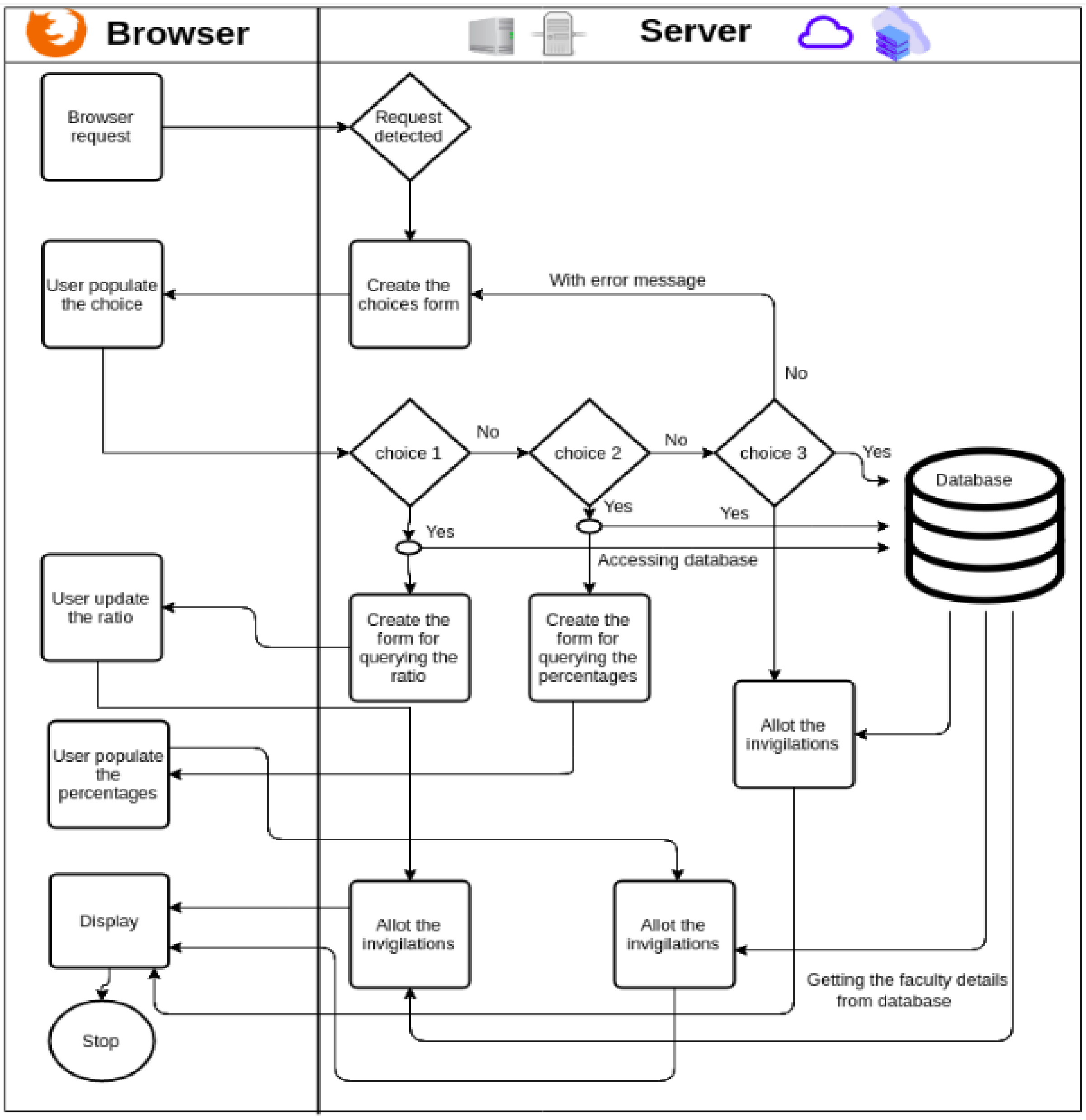


Figure 1. Working of the Invigilation system

The server waits for the request from the browser. When a request detected, the server populates the choices form. The choices form consists of three choices. The server finds one of the choices selected and connects to the database. The database consists of all the rooms, students and faculty data. If choice 1 or 2 are selected, it requests the ratio or percentages by populating the form. Allot invigilators section take input of the populated data and data from the database and produce the allotted invigilations. The algorithm of allotting invigilators is explained below.

Choice 1 – Allotting as per the ratio:

In this choice, the user is importuned for a ratio. This ratio is not used directly to allocate invigilations category wise [Algorithm1-step4]. Instead, the software appraises all the faculty count category wise and convert to a new ratio [Algorithm 1 – Step 3] and allot as per the new ratio. In the end, the invigilations are allotted such that if the ratio of the professor: Associate professor: Assistant professor = x:y:z then if x±1 invigilations are allotted for each professor then y±1 invigilations will be allotted for each of the Associate professor and z±1 invigilations allotted for each of the Assistant professor.

The total invigilations depends on the total sessions and invigilations per session. The days or sessions as key with number of invigilations as values are stored in the dictionary named days. Let days = {'09-4-2020':14,'10-4-2020':9,'11-4-2020':10,'13-4-2020':9, '14-4-2020':10,'15-4-2020':11, '16-4-2020':9,'17-4-2020':6,'19-4-2020':12,'20-4-2020':13}. Now the total invigilations are the sum of all the invigilations per session. Python furnishes 'sum' using which all the values of a dictionary can be added. And the ratio is solicited from the user through a POST form and amass in database. The calculations when ratio input is given can be done in the database itself using F() but it alters all the elements of previously submitted which results in the loss of aboriginal ratio. So the calculations must be for last submitted and at the exterior of the database. The final result will be saved in other fields of the database in order to protect the originals. Let rc and rcf be the model classes that stores ratio and final result respectively. Since there may be many ratios that are antecedently submitted, the views.py must access the latest ratio for attaining the final result. The latest can access by the combination of order_id of primary key (which returns the queryset in ascending order of primary key) and last() (which returns the last object in the queryset). Since Django adds automatically the primary keys in ascending order, the last one is the required element. Let the required element be p. In order to make rc and rcf are mutually connected, one to one relationship is used [28] and if rc is deleted, rcf related to it has no meaning. Hence on_delete is used to delete rcf when rc is deleted.

Algorithm 1 - Allotting as per the ratio:

Step 1: Get the faculty count from the database. Assume, p=a, asstp=b and asscp = c. Where p is the Professor, asstp is Assistant Professor and asscp is the Associate Professor and l,m and n are their respective counts.
Step 2: Requesting for a ratio from user. Assume the ratio is p:asstp: asscp = x:y:z.
Step 3: Converting the previous ratio to new ratio as category value * category ratio/total ratio.
i.e., Now p:asstp:asscp = a * x/(x+y+z): b * y/(x+y+z) : c* z/(x+y+z) = i:j:k (assumption).
Step 4: The invigilations assigned per category can be calculated as,
for category p, Tp = Total invigilations * i/(i+j+k),
for category asstp, Tasstp = Total invigilations * j/(i+j+k),
for category asscp, Tasscp= Total invigilations * k/(i+j+k).
Step 5: Convert step 4 results to integers. The invigilations assigned per individual is,
for category p, pin = Tp/a,
for category asstp, asstpin = Tasstp/b,
for category asscp, asscpin = Tasscp/c.
Step 6: Convert step 5 results to integers. The number of faculty need the maximum invigilations are,
for category p, mp = Tp - pin x a,
for category asstp, masstp = Tasstp - asstpin x b,
for category asscp, masscp = Tasscp - asscpin x c.

Even though if any category is zero but specified some ratio, it does not influence the other ratio.

In order to do the further processing views.py should get (a) the total number of faculty (b) the total number of faculty category wise. (a) can get by selecting all the faculty objects [15] and then counting [15]. (b) can get by using filtering [15] and then counting.

While doing step 5, there is a possibility of divide by zero error because of any of the category count is zero. Hence, a zero condition must be checked.

2.1 Error correction

The calculations in step 4 are floating point values. So, the sum of all not exactly equal to total invigilations. The error is only less than one but since humans count is integers, it must be considered. So, the result of calculations are converted to integers and now the maximum error is n-1 where n is the total number of elements of the ratio. Even though the error is more, all the parameters are integers. The error must add to any of the category or all the categories equally. The best way is to add to the category having the maximum number of invigilators. Python provides max() function using which maximum invigilators category can found and the error can add to it.

2.2 Proper allocation of minima and maxima

The calculations of step 5 are also floating point values. Suppose the value is 4.5 then 4 or 5 invigilations has to apport for each of that catgeory. The assumption is 4 is minimum and 5 is maximum. The integer conversion presents the minimum. The question now is how many are the minimum and how many are the maximum which is calculated in step 6.

Since different categories having different minimum and maximum invigilations, a seperate dictonary having the invigilation count with name is exigent for alloting into the rooms. Categorywise data can procure into views from database by filtering [27]. The queryset now consists of name, department and designation. The faculty name can get as queryset[n].name from queryset. If name only stored as key and invigilation count as value into the dictonary. The problem is name is not unique and dictonary doesn't support having more than one same key. This can be solved by checking the condition queryset[i].name in dict where dict is the dictonary. when the same key is present changing the other key by appending one to it. But this leaves in a dubiety that who is who among them. If suppose department is appended, it disentangles but still there is a feasibility that having name and department is same but having the designation different for a faculty. Hence appending both the department and designation using + operator can unravel this.

Two count values are feasible. At first, the value of the dictionary is filled with the minimum count. The value get from step 6 faculty numbers are now necessitate to alter to the maximum. By adding one to the minimum, can revise to the maximum count. But from where to start. If added from the start, always the first added faculty will obtain the maximum. So different times different techniques need to endeavour. For example one time from first, next time form last, next time based on order etc. Instead, the best way is by effectuating random numbers and then allot to that specific random numbered invigilator. The random number must between the 1 and the length of that specific category and the random number must not engender using random.randint because there is a possibility of a same random number more than once. Hence all the numbers must generate at a time using random.sample which permits the unrepeated random values in a list. Let df be the final dictionary consisting of the key as faculty name and value as the number of invigilations.

Now invigilations has to dispense to days as per the stipulation in the days dictionary. For the efficient allocation of the invigilations the preference must be given for the value having more invigilations in both the days and df dictionaries. So, days and df dictionaries are sorted in descending order as per the values. Python bestows sorted function for sorting the dictionary. Even though because of less faculty or assigning high ratio for specific category, it is feasible that a faculty in df may have more invigilations than the total number of days or sessions. In such instance apportionment is not possible. But if the fount is ratio then the user is supplicated to modify the ratio. In order to percieve the cause is ratio, the worst case ratio 1:1:1 is considered and find the allocation is possible or not.

Python supports list as a value for a key in a dictionary. Now the days dictionary's value (i.e., integer count) is replaced with the list of invigilators selected from df. The care is taken to disallow of repetition of the same invigilator for a session. When an invigilator is allotted from df to days then the df's value of that key is reduced by one. If the value is zero then that invigilator must not allow for any other session.

Choice 2 - Allotting as per the percentages of total:

Algorithm 2:

Step 1: Get the faculty values from database suppose, p=a, asstp=b and asscp = c.
Step 2: Requesting from the user the percentage. Assume the percentage of p is m, asstp is n, automatically the asscp is 100 - m – n.
Step 3: Total invigilations for category p is Tp = (a+b+c)xm/100 , asstp is Ta = (a+b+c)xn/100 and asscp is Tas = (a+b+c)x(100-m-n)/100.
Step 4: Minimum invigilations per p is Pmin = Total invigilations of p / a = (a+b+c)xm/(ax100). Similarly, for each asstp is Amin = (a+b+c)xn/(bx100) and for each asscp is Asmin = (a+b+c)x(100-m-n)/(cx100) and maximum number of invigilations = Minimum invigilations + 1.
Step 5: The number of p having Pmin invigilations = Tp – a x Pmin. Similarly, In the case of asstp is Ta – b x Amin and asscp is Tas – c x Asmin.

Choice 3 - Allotting invigilations considering all are equal:

Algorithm 3:

Step 1: Get the total count of the faculty.
Step 2: Find the number of invigilations for each faculty. The Minimum number of invigilations is min = Total invigilations / Total faculty and the maximum number of invigilations is max = min + 1.
Step 3: The number of faculty for whom the min invigilations are allotted is fmin= Total invigilations – Total faculty x min.
Step 4: The number of faculty for whom the max. invigilations are allotted is fmax = Total faculty – fmin.

After finding the minimum and maximum invigilations per category, the invigilations are allotted for the sessions in case 2 and case 3 is the same as case 1.

After allotting the faculty into rooms, the next step is tracking followed by face recognition.

3. TRACKING

In many applications tracking is useful, for example it is used in counting purposes [29], home surveillance [30] etc. If face recognition is conducted prior to finding the movement, it is necessary to recognise all the faces in room. Instead by recognising only persons in motion, the processing power and time can be reduced. If invigilator is not found in movement, it is predicted as not in motion. The motion is estimated by calculating the difference of the frames as illustrated in Figure 5 and their respective outputs are sketched at Figures 2, 3, 4, 6, 7, 8, 9 and 10.

Figures 2 and 3 are the successive frames. In order to find the movement, the difference is calculated with absdiff method and is illustrated in Figure 4 with background image as Frame2 and foreground image as Frame1. The amount of movement is estimated by drawing the contours. In order to find the contours, the image is thresholded (Figure 8) that highlights the changed portion. In order to stress the moving parts, extra pixels are added by the process of dilation and is illustrated in Figure 9. The dark spots of Figure 9 is taken as contours and drawn over frame 2 to predict the changed portion which is illustrated in Figure 10.

Figure 2. Frame 1

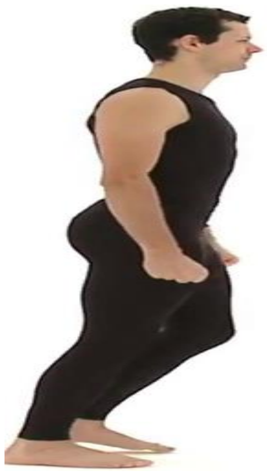


Figure 3. Frame 2

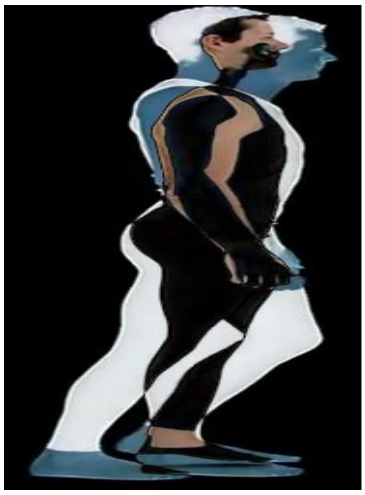


Figure 4. Difference

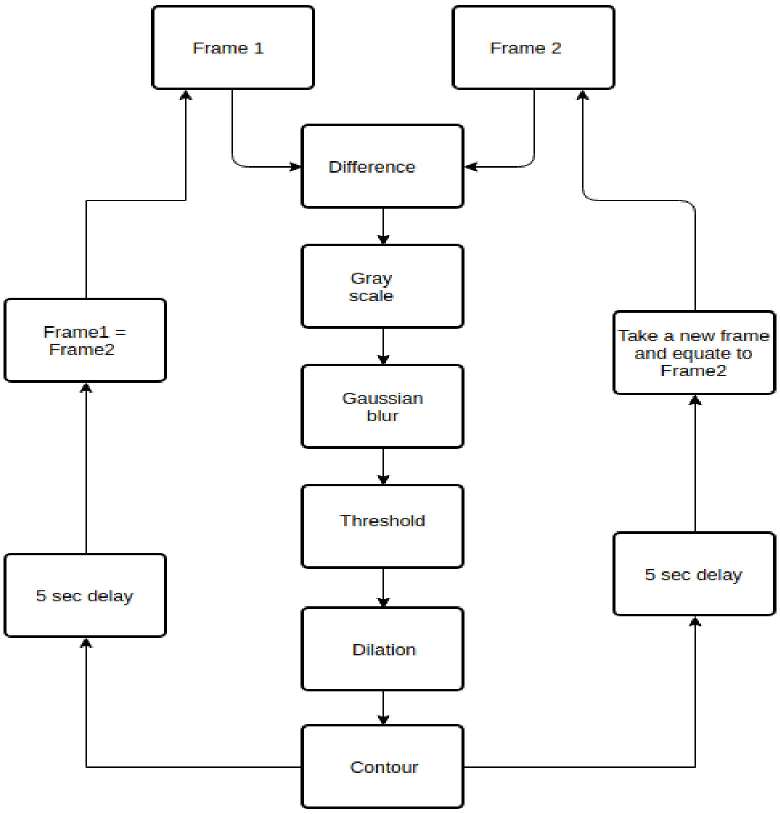
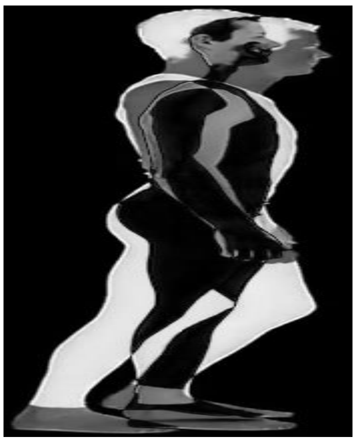
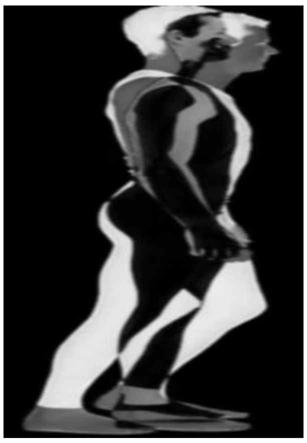


Figure 5. Tracking using the difference of two frames





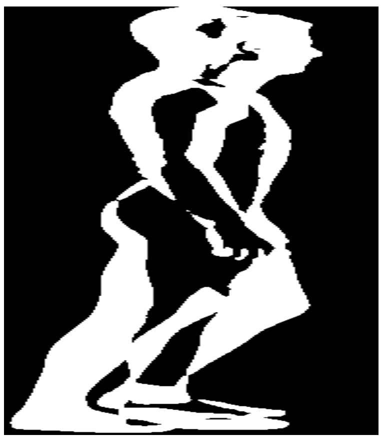


Figure 6. Gray **Figure 7.** Blur **Figure 8.** Threshold

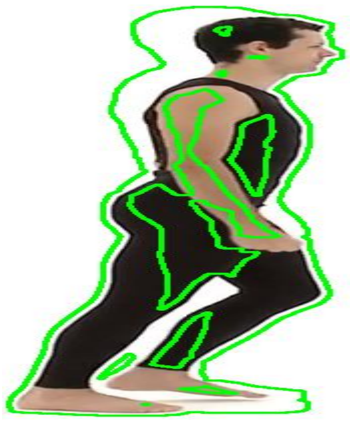


Figure 9. Dilution

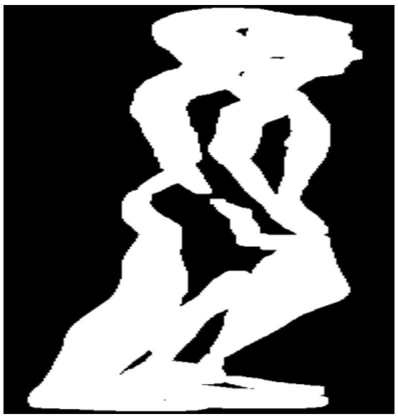


Figure 10. Contours

4. FACE RECOGNITION

The primary task for the face recognition is the face detection [31]. Eventhough haarcascade method [32] is faster, HOG method is used for face detection because of the minimal false detection. Face recognition involves two stages – Feature Extraction and Face Classification. Feature Extraction identifies the face and face clssification provide an identifier for the recognized. The feature extraction stage is attained by training with samples. The trained model can be used to recognize any other face without retraining. The reliability of the trained model can be estimated with triplet loss function.

4.1 Triplet loss function

One way to learn the parameters of the neural network so that it gives a good encoding for faces is by defining an applied gardient descent on the triplet loss function. To apply the triplet loss, the requirement is the comparison of two images.

For example, Figure 11 and Figure 12 are of same person's images, Hence, their encodings must be similar but Figures 11 and 13 are different, hence it is required to get the different encodings. In other words the difference between the encodings of sample image to matched image(d(s,m)) is much lesser than the difference between the encodings of sample image to unmatched image(d(s,u)).

$$d(s,m) \leq d(s,u)$$
$$\|E(s) - E(m)\|^2 - \|E(s) - E(u)\|^2 \leq 0$$

One trivial that it satified is E(s)=E(m)=E(u) = 0 i.e., E (any image) is a vector of all zeros, almost trivially satisfies this equation. So to make sure that the neural network just outputs zero like this for all the encodings. It also shouldn't set all the encodings to equal to each other i.e., encoding of every image is identical to the encoding of the other image. In order to test that it is required to add quite a small element to LHS.

$$\|E(s) - E(m)\|^2 - \|E(s) - E(u)\|^2 + k \leq 0$$

where, k is the another hyper parameter which prevents the neural network in preventing the trivial solution. k is also called margin parameter.

The loss function for a single triplet is L(s,m,u) = max(E(s,m)-E(s,u)+k , 0).

The overall cause function for it can be J = $\sum_{i=1}^{n} L(s^{(i)},m^{(i)},u^{(i)})$.

Let's say if the training set consists of 10,000 pictures. In order to train the learning algorithm, there is a need of generating triplets (sample, matched and unmatched) for each of 10,000 pictures and then train the learning algorithm using this type of cause functions drawn from the training set.

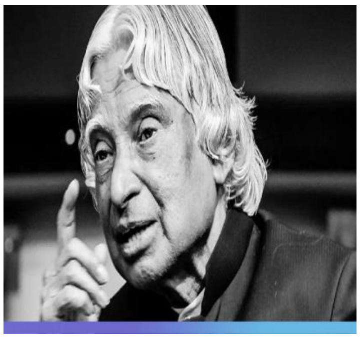


Figure 11. Sample

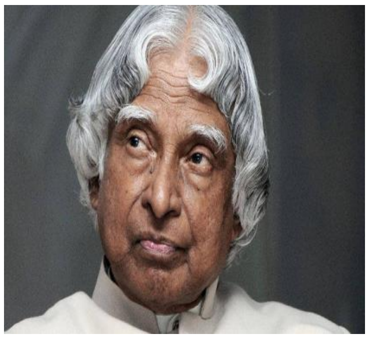


Figure 12. Matched

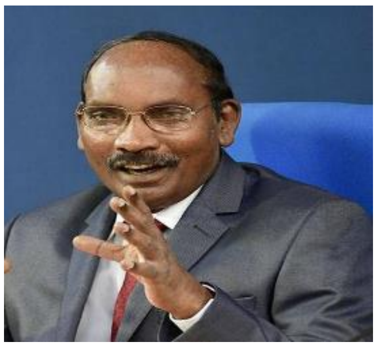


Figure 13. Unmatched

4.2 Face recognizer

Step 1: Detect the faces using HOG method.
Step 2: Isolate the faces and represent those with the 128D facial embeddings using the neural network.
Step 3: Compare the embeddings using the Euclidean distance =

$$\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+----+(a_p-b_p)^2} = \sqrt{\sum_{i=1}^{p}(a_i-b_i)^2}$$

Step 4: if the Euclidean distance < threshold then the faces are said to be matched else un matched.

5. MOVEMENT ESTIMATION

The estimated movement is identified and drawn contours over them and invigilator movement is detected through these countours which is illustrated in Figure 14. Using the contour image from the Tracking algorithm, each contour is separated from the image using a contour separator. Each contour is processed with a face detector. Because of its high accuracy and speed, HOG is selected as the Face detector. If any face is found in the contour, it is processed with Face recognizer, it compares with the faces of invigilators and decides that the invigilator is in motion and his data would be updated.

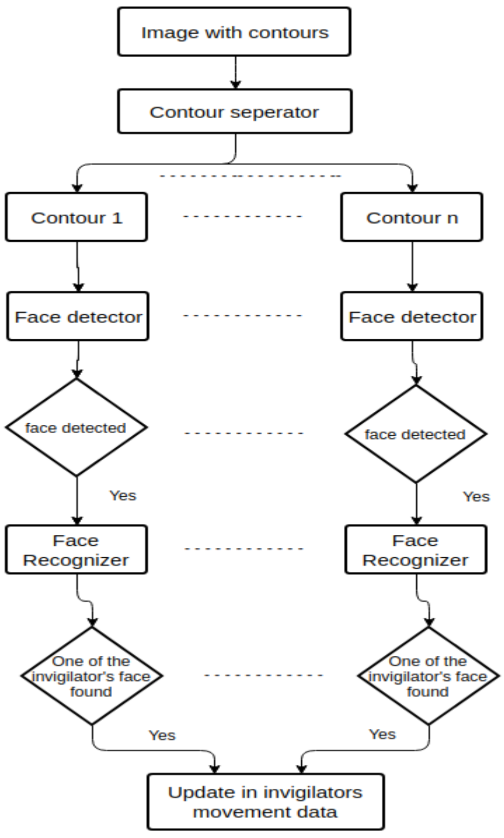


Figure 14. Movement Estimation

6. RESULTS AND DISCUSSION

The software is tested by taking the following faculty of which five are professors, nine are Associate Professors and sixteen are Assistant Professors.

L Roja - ECE - Prof
S Mounika - CSE - Prof
P Sandhya - IT - Prof
A Sujata - EEE - Prof
E Lavanya - CSE – Prof

E Rahul - IT - Assocprof
B Vasu - CSE - Assocprof
C Chndram - ECE - Assocprof
L Ramu - CSE - Assocprof
P Durga - EEE - Assocprof
J Tarak - ECE - Assocprof
K Sushmitha - IT – Assocprof
Z Meena - EEE – Assocprof
M Madan - ECE – Assocprof

K Malavya - CSE - Asstprof
J Karthik - CSE - Asstprof
S Jeevan - ECE - Asstprof
Y Muneeth - EEE - Asstprof
L Munaiah - IT - Asstprof
P Mani - CSE - Asstprof
G Ramesh - ECE - Asstprof
J Jagadesh - EEE - Asstprof
R Manikarnika - IT - Asstprof
K Syed - CSE - Asstprof
M Joseph - ECE - Asstprof
P Sonia - EEE - Asstprof
J Ashraf - IT - Asstprof
L Rajesh - CSE - Asstprof
V Ravi - ECE - Asstprof
L Basha - ECE - Asstprof

The invigilators allocation is initiated once the students allocation is completed. The requirement of invigilators count is estimated based on the rooms into which students are allotted. The required invigilation count is calculated and stored with date as dictionary named days.

days={'09-4-2020':14,'10-4-2020':9,'11-4-2020':10,'13-4-2020':9,'14-4-2020':10,'15-4-2020':11,'16-4-2020':9,'17-4-2020':6,'19-4-2020':12,'20-4-2020':13}

Next is the choice selection, if the choice 1 is selected, it will prompt for ratio. The ratio here entered is Prof:Assoc.Prof:Assistant.Prof = 1:3:5, it will generates the number of invigilations needed per faculty and is stored in dictonary inv.

Inv = {'K Malavya,CSE,Asstprof': 4, 'J Karthik,CSE,Asstprof': 5, 'S Jeevan,ECE,Asstprof': 4, 'Y Muneeth,EEE,Asstprof': 5, 'L Munaiah,IT,Asstprof': 4, 'P Mani,CSE,Asstprof': 4, 'G Ramesh,ECE,Asstprof': 5, 'J Jagadesh,EEE,Asstprof': 5, 'R Manikarnika,IT,Asstprof': 4, 'K Syed,CSE,Asstprof': 5, 'M Joseph,ECE,Asstprof': 5, 'P Sonia,EEE,Asstprof': 5, 'J Ashraf,IT,Asstprof': 5, 'L Rajesh,CSE,Asstprof': 5, 'V Ravi,ECE,Asstprof': 5, 'L Basha,ECE,Asstprof': 5, 'K Sushmitha,IT,Assocprof': 2, 'Z Meena,EEE,Assocprof': 3, 'M Madan,ECE,Assocprof': 2, 'E Rahul,IT,Assocprof': 2, 'B Vasu,CSE,Assocprof': 3, 'C Chndram,ECE,Assocprof': 3, 'L Ramu,CSE,Assocprof': 3, 'P Durga,EEE,Assocprof': 3, 'J Tarak,ECE,Assocprof': 3, 'S Mounika,CSE,Prof': 0, 'P Sandhya,IT,Prof': 1, 'A Sujata,EEE,Prof': 1, 'L Roja,ECE,Prof': 1, 'E Lavanya,CSE,Prof': 1}

Using days and Inv dictionaries, the invigilators for a day is generated using Algorithm1, which is presented below.

Date - Invigilators

09-4-2020 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof', 'J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof', 'M Joseph,ECE,Asstprof', 'P Sonia,EEE, Asstprof', 'J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof', 'V Ravi,ECE,Asstprof', 'L Basha,ECE,Asstprof', 'K Malavya,CSE,Asstprof', 'S Jeevan,ECE,Asstprof', 'L Munaiah,IT,Asstprof']
10-4-2020 - ['S Jeevan,ECE,Asstprof', 'L Munaiah,IT,Asstprof', 'P Mani,CSE,Asstprof', 'R Manikarnika,IT,Asstprof', 'Z Meena,EEE,Assocprof', 'B Vasu,CSE,Assocprof', 'C Chndram,ECE,Assocprof', 'L Ramu,CSE,Assocprof', 'P Durga,EEE,Assocprof']
11-4-2020 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof', 'J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof', 'M Joseph,ECE,Asstprof', 'P Sonia,EEE,Asstprof', 'J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof', 'V Ravi,ECE,Asstprof']
13-4-2020 - ['L Munaiah,IT,Asstprof', 'P Mani,CSE,Asstprof', 'R Manikarnika,IT,Asstprof', 'Z Meena,EEE,Assocprof', 'B Vasu,CSE,Assocprof', 'C Chndram,ECE,Assocprof', 'L Ramu,CSE,Assocprof', 'P Durga,EEE,Assocprof', 'J Tarak,ECE,Assocprof']
14-4-2020 - ['L Basha,ECE,Asstprof', 'K Malavya,CSE,Asstprof', 'S Jeevan,ECE,Asstprof', 'L Munaiah,IT,Asstprof', 'P Mani,CSE,Asstprof', 'R Manikarnika,IT,Asstprof', 'Z Meena,EEE,Assocprof', 'B Vasu,CSE,Assocprof', 'C Chndram,ECE,Assocprof', 'L Ramu,CSE,Assocprof']
15-4-2020 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof', 'J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof', 'M Joseph,ECE,Asstprof', 'P Sonia,EEE,Asstprof', 'J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof', 'V Ravi,ECE,Asstprof', 'L Basha,ECE,Asstprof']
16-4-2020 - ['P Mani,CSE,Asstprof', 'R Manikarnika,IT,Asstprof', 'P Durga,EEE,Assocprof', 'J Tarak,ECE,Assocprof', 'K Sushmitha,IT,Assocprof', 'M Madan,ECE,Assocprof', 'E Rahul,IT,Assocprof', 'P Sandhya,IT,Prof', 'A Sujata,EEE,Prof']
17-4-2020 - ['J Tarak,ECE,Assocprof', 'K Sushmitha,IT,Assocprof', 'M Madan,ECE,Assocprof', 'E Rahul,IT,Assocprof', 'L Roja,ECE,Prof', 'E Lavanya,CSE,Prof']
19-4-2020 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof', 'J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof', 'M Joseph,ECE,Asstprof', 'P Sonia,EEE,Asstprof', 'J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof', 'V Ravi,ECE,Asstprof', 'L Basha,ECE,Asstprof', 'K Malavya,CSE,Asstprof']
20-4-2020 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof', 'J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof', 'M Joseph,ECE,Asstprof', 'P Sonia,EEE,Asstprof', 'J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof', 'V Ravi,ECE,Asstprof', 'L Basha,ECE,Asstprof', 'K Malavya,CSE,Asstprof', 'S Jeevan,ECE,Asstprof']

After generating the invigilator's list per day, they were allotted into rooms such that for each 30 pupil one invigiltor. For the date 09-04-2020, the invigilators allotted for rooms is presented below,

RM1 - ['J Karthik,CSE,Asstprof', 'Y Muneeth,EEE,Asstprof', 'G Ramesh,ECE,Asstprof']
RM2 - ['J Jagadesh,EEE,Asstprof', 'K Syed,CSE,Asstprof']
RM3 - ['M Joseph,ECE,Asstprof']
RM4 - ['P Sonia,EEE, Asstprof','J Ashraf,IT,Asstprof', 'L Rajesh,CSE,Asstprof']
RM5 - ['V Ravi,ECE,Asstprof', 'L Basha,ECE,Asstprof', 'K Malavya,CSE,Asstprof']
RM6 - ['S Jeevan,ECE,Asstprof', 'L Munaiah,IT,Asstprof']

After allotting into the rooms, the two cameras fixed per room send the pics to django server. The Estimation of movement per invigilator per room is accomplished by drawing the contours over the difference of the frames. The tool used for image anlysis is opencv.

7. CONCLUSIONS

The invigilation system starts with invigilator allocation. Django is used for allocation by considering the faculty, room strength and the students from the Postgres database. The central server shares the allotted invigilator data with the client devices placed inside the examination halls. The tracking algorithm tracks the movement by extracting each frame. The present tracking algorithm uses the difference in frames which is efficient than others. KLT tracking method takes more training time. PHD filtering and Practicle filtering tracking methods are inaccurate in detecting small movements. The trace learning tracking method is strongly dependent on weak classifiers. All these disadvantages are overcome by the present frame difference method. The motion found parts are drawn contours and are processed with a Face detector. The face detector used in this paper is HOG. The Haar cascade method is faster than HOG but inaccurate to use in real-time. CNN is accurate than HOG and Haar Cascade but too slow with CPUs. Hence HOG method is found to be best suited for the realtime application. The found face is validated with the face recognizer and is based on 128D embeddings whose accuracy is 99.38% which is the best accuracy to use in real-time.

REFERENCES

[1] Meena, R., Sujeetha Devi, T., Vinodhini, R., Vishwam, R., Yamini Priya, H. (2016). Design and Implementation of Virtual invigilation System and Smart Examscheduler. Advances in Natural and Applied Sciences.
[2] Lin, G., Tsai, T. (2012). A face tracking method using feature point tracking. 2012 International Conference on Information Security and Intelligent Control, Yunlin, Taiwan, pp. 210-213. https://doi.org/10.1109/ISIC.2012.6449743
[3] Alqahtani, F., Banks, J., Chandran, V., Zhang, J. (2020). 3D face tracking using stereo cameras: A review. IEEE Access, 8: 94373-94393. https://doi.org/10.1109/ACCESS.2020.2994283

[4] Wang, P., Ji, Q. (2008). Robust face tracking via collaboration of generic and specific models. IEEE Transactions on Image Processing, 17(7): 1189-1199. https://doi.org/10.1109/TIP.2008.924287

[5] La Cascia, M., Sclaroff, S., Athitsos, V. (2000). Fast, reliable head tracking under varying illumination: an approach based on registration of texture-mapped 3D models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(4): 322-336. https://doi.org/10.1109/34.845375.

[6] Duffner, S., Odobez, J. (2013). Track creation and deletion framework for long-term online multiface tracking. IEEE Transactions on Image Processing, 22(1): 272-285. https://doi.org/10.1109/TIP.2012.2210238.

[7] Dornaika, F., Davoine, F. (2006). On appearance based face and facial action tracking. In IEEE Transactions on Circuits and Systems for Video Technology, 16(9): 1107-1124. https://doi.org/10.1109/TCSVT.2006.881200

[8] Vadakkepat, P., Lim, P., De Silva, L.C., Jing, L., Ling, L.L. (2008). Multimodal approach to human-face detection and tracking. IEEE Transactions on Industrial Electronics, 55(3): 1385-1393. https://doi.org/10.1109/TIE.2007.903993

[9] Qi, Y., Zhang, S., Jiang, F., Zhou, H., Tao, D., Li, X. (2020). Siamese local and global networks for robust face tracking. IEEE Transactions on Image Processing, 29: 9152-9164. https://doi.org/10.1109/TIP.2020.3023621

[10] Wang, G., Bhuiyan, M.Z.A., Cao, J., Wu, J. (2014). Detecting movements of a target using face tracking in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 25(4): 939-949. https://doi.org/10.1109/TPDS.2013.91

[11] Charles Severance, University of Michigan. (2015). Guido van rossum: The early years of python. IEEE Computer Society. https://doi.org/10.1109/MC.2015.45

[12] Varma, R.K., Raju, P.L.N., Priyanka, M. (2017). An IoT application for environmental monitoring and control using Raspberry-Pi. International Journal of Engineering and Technology (IJET), 9(3): 546-552. https://doi.org/10.21817/ijet/2017/v9i3/170903S082

[13] Text book: Adrian McEwen. Designing the Internet of Things. Wiley.

[14] Coughlin, T. (2017). Supersize My Pi. IEEE Consumer Electronics Magazine. https://doi.org/10.1109/MCE.2017.2685038

[15] Jose, A.C., Malekian, R. (2017). Improving smart home security: Integrating logical sensing into smart home. IEEE Sensors Journal, 17(13): 4269-4286. https://doi.org/10.1109/JSEN.2017.2705045

[16] Yeh, K.H. (2017). A secure IoT-based healthcare system with body sensor networks. IEEE Access, 4: 10288-10299. https://doi.org/10.1109/ACCESS.2016.2638038

[17] Kamath, R., Balachandra, M., Prabhu, S. (2019). Raspberry Pi as visual sensor nodes in precision agriculture: A study. IEEE Access, 7: 45110-45122. https://doi.org/10.1109/ACCESS.2019.2908846

[18] Goyal, S., Desai, P., Swaminathan, V. (2017). Multi-level security embedded with surveillance system. IEEE Sensors Journal, 17(22): 7497-7501. https://doi.org/10.1109/JSEN.2017.2756876

[19] D'haro, L.F., De Córdoba, R., Rojo, J.I., Díez, J., Avendaño, D., Bermudo, J.M. (2014). Low-cost speaker and language recognition systems running on a raspberry Pi. IEEE Latin America Transactions, 12(4): 755-763. https://doi.org/10.1109/TLA.2014.6868880

[20] Lee, C.H. (2017). Location-aware speakers for the virtual reality environments. IEEE Access, 5: 2636-2640. https://doi.org/10.1109/ACCESS.2017.2672556

[21] Segura-Garcia, J., Felici-Castell, S., Perez-Solano, J.J., Cobos, M., Navarro, J.M. (2015). Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks. IEEE Sensors Journal, 15(2): 836-844. https://doi.org/10.1109/JSEN.2014.2356342

[22] He, J.H., Wei, J., Chen, K., Tang, Z.Y., Zhou, Y., Zhang, Y. (2018). Multitier fog computing with large-scale IoT data analytics for smart cities. IEEE Internet of Things Journal, 5(2): 677-686. https://doi.org/10.1109/JIOT.2017.2724845

[23] Text book: Arshadeep Bahga, Vijay Madisetti, "Internet of Things", University Press-2018.

[24] Rodríguez, R.A., Cammarano, P., Giulianelli, D.A., Vera, P.M., Trigueros, A., Albornoz, L.J. (2018). Using raspberry Pi to create a solution for accessing educative questionnaires from mobile devices. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 13(4): 144-151. https://doi.org/10.1109/RITA.2018.2879387

[25] Zhao, C.W., Jegatheesan, J., Loon, S.C. (2015). Exploring IOT application using raspberry Pi. International Journal of Computer Networks and Applications, 2(1): 27-34.

[26] Alvarado, A., Bajaña, B., Munoz, J.A., Velásquez, W. (2017). Bicycle protection within an university area using geolocation and perimeter security. IEEE Latin America Transactions, 15(6): 1137.

[27] Text book: Adrian Holovaty, Jacob K. Moss. Django: The Definitive Guide to Django: Web Development Done Right. https://doi.org/10.1007/978-1-4302-1937-8

[28] Text Book: Sanjeev Jaiswal, Ratan Kumar, "Learning Django Web Development", PACT publishers – 2015.

[29] García, J., Gardel, A., Bravo, I., Lázaro, J.L., Martínez, M., Rodríguez, D. (2013). Directional people counter based on head tracking. IEEE Transactions on Industrial Electronics, 60(9): 3991-4000. https://doi.org/10.1109/TIE.2012.2206330

[30] Cucchiara, R., Grana, C., Prati, A., Vezzani, R. (2005). Computer vision system for in-house video surveillance. IEEE Proceedings-Vision, Image and Signal Processing, 152(2): 242-249. https://doi.org/10.1049/ip-vis:20041215

[31] Hsu, R.L., Abdel-Mottaleb, M., Jain, A.K. (2002). Face detection in color images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5): 696-706. https://doi.org/10.1109/34.1000242

[32] Huang, J., Shang, Y.Y., Chen, H. (2019). Improved viola-jones face detection algorithm based on HoloLens. EURASIP Journal on Image and Video Processing. https://doi.org/10.1186/s13640-019-0435-6