ECOLE POLYTECHNIQUE DE BRUXELLES

PROJ-H-402 - COMPUTING PROJECT

Counting People Inside EPB Classrooms

Author: Daniel GOMEZ DE GRACIA
Supervisors: Rudy ERCEK, Sophia AZZAGNUNI

April 2020

Abstract

Having an adequate number of students, relative to the room's number of seats inside the classroom where they follow a lesson, is a challenge that is very desirable to achieve. In a big faculty like the Ecole Polytechnique de Bruxelles, choosing a particular room for a certain course on the base of it's capacity and of the number of students following that course is essential to improve the overall quality of the learning experience. Proper allocation of classrooms would avoid unwanted situations like the ones we see sometimes at present, where a handful of people occupy a 200-seat auditorium for example, or, even worse, where a huge audience is confined in a tiny room, with barely enough seats for everybody. This project aimed to work on a first step to give a solution to this problem. After analyzing many images from several auditoriums, the conslusion is that we can detect close to 100% of the people present in a room, by taking an adequate number of pictures during a period of time of 5 to 10 minutes, with a camera placed between one of the corners of the front wall of the room and the center of that wall.

1 Introduction

In the framework of a project for the academic year 2019-2020, it was decided to develop a practical solution for the "classroom occupation problem" in the Ecole Polytechnique de Bruxelles. This reefers to the situation where auditoriums are overcrowded or nearly empty relative to their capacity, during lectures or practical labs. This project developed some tools and results to help in the resolution of this problem. However, the methods used in this paper for this particular goal are certainly applicable to a large variety of rooms, and even to a variety of goals.

In order to get an accurate idea of the actual occupation rate of a classroom, it was decided to actually count the students present in the room, during the lesson. In order to accomplish this, a camera connected to a Raspberry Pi, mounted on a tripod, and placed strategically inside the room, was used to capture images of the whole audience while the lesson took place. We supposed that the number of students present in the range of vision of the camera was constant (so no students were going in and out of the picture). Later on, the images were processed, and an algorithm was run to try to detect and count the faces as accurately as possible. This project made use of classrooms exclusively belonging to the EPB, focusing particularly on the ones located in the 4th floor on the A-B wing.

This report will first explain how, and under which environmental conditions, the acquisition of the different raw images was made. Then, it will show how several datasets were built for each of the purposes we had. After, it will examine the algorithms' outputs when fed with our images. It will then reveal the best detection ratio achieved for each room, and also the best way of finding the correct number of people present in a room. It will conclude on what the best acquisition techniques and the best adjustments that can be made to unprocessed pictures are, as well as on which of the algorithms and estimations are the most appropriate to apply. Finally, it will discuss possible improvements for this project, regarding acquisition techniques, hardware election, and image processing.

2 Material and Methods

2.1 Materials

To be able to take the necessary pictures for this project, we needed a camera that would be small, easy to carry, and with enough resolution to produce usable images. We wanted it to be able to take several pictures in a row with only one command, and to be able to start/pause the process whenever we chose to. We also needed to get a visual feedback of what was going to be recorded before the capture was launched, to help us correctly adjust the angles and height of the camera . Finally, we also wanted to make sure that the images were being correctly saved during the acquisition, to avoid memory-shortage problems, or

any other bug.

For theses reasons, we decided to use a Raspberry Pi 3.0 with a PiCamera, mounted on a tripod (with the help of a 3D-printed platform) allowing to film from over 2m high, and connected to a portable battery. We equipped the camera with a distortion lens to be able to capture the entire classroom. This camera was able to take images with a pretty good resolution of 3280x2464 pixels.

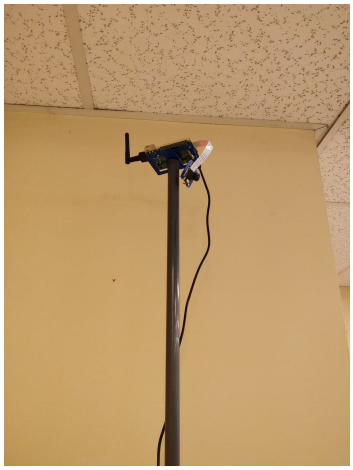

Figure 1: PiCamera mounted on tripod

2.2 Methods

We will first define what the objective of our method was. We wanted to develop an automatic "input/output" process in which pictures of a classroom were the input, and the number of people present in a room was the output. To do this, we would need a basic component: an algorithm that would recognize faces inside a room. We observed however that, depending on the image it was given as input, this kind of face detection algorithms didn't always recognize everyone. We designed a method to help our machine make the most accurate guess about how many people are present. We will discuss in the following sections the process of thought we went through with.

2.2.1 Captured images

To produce our images, we decided first to make 8 different total captures (in several rooms at different angles), numbered from 1 to 8. Each **capture** is defined as the process of constantly taking pictures (one every two seconds approximately) of a constant number of people, keeping parameters like height and angle of the camera unchanged. We took between 10 and 50 pictures per capture. The first 4 captures were done in the same auditorium, but from different positions and angles. These allowed us to study the influence of the camera's placement and orientation on the detection ratio, and reach a conclusion

about the best positioning for that particular room. Next, captures number 4 and 6 were done in the same auditorium in order to compare results for different camera heights, but preserving approximately the same spatial position and angle of the tripod. Finally, we decided to make additional captures 5 and 7 to have a bit more data.

2.2.2 Algorithms

Since we were dealing with pictures containing relatively small faces compared to the size of the entire picture, we decided that we would get the best detection ratios by using algorithms specialized in detecting "tiny faces" (the definition of this term is relatively well established in the world of computer vision). After having a look on recent literature, we found two candidate algorithms: The "Extended Tiny Faces" ([AD18]), and the "Fully Convolutional Head Detector" ([VC19]). These were provided with open source python codes to run them ([Att], [Vor]). After making a few modifications to the original codes to fit their output to our personal preferences, as well as experimenting with the different parameters provided, we were able to run the codes in a regular laptop. Concerning the input that these two algorithms expect, ETF can take an image of any size as input, and it will output an image with the same resolution as the original one. FCHD, on the other hand, can only take 1000x600 pixel images, and will output a relatively low-resolution image of 549x326 pixels. FCHD comes with a configuration file that we can customize, including a "weight decay" parameter to adjust sensibility. To run ETF, we had to search on the internet for an appropriate weight file, and transform it to a pickle format (both provided at [cyd]). To run FCHD, we had to resize all the images to the appropriate format before feeding them to the algorithm.

2.2.3 Dataset

After careful consideration of the possible outcomes of the algorithms, we decided to operate in the following way. First of all, we would identify the possible detection issues we would get when running the algorithms on a single image. Then, we would see if applying certain modifications to that image, like point processing techniques or undistortion, could yield a better result. Finally, if some of the encountered detection issues could not be solved by means of image processing, we would try the algorithms on other images of the same capture to see if some of the problems could now get solved.

The first set of images we created was the **"Basic Image Dataset"**. We brought together 8 images, one from each of the 8 captures we performed, to build the dataset. We will call them "basic images". To make the selection of these pictures, we decided to randomly pick one image among the dozens of pictures that composed each capture set. This way we maximized the diversity of potential issues we could experience during the detec-

tion of faces (like for example, not detecting people who are not looking in front of them, or sitting very far away from the camera, or at the edge of the picture).

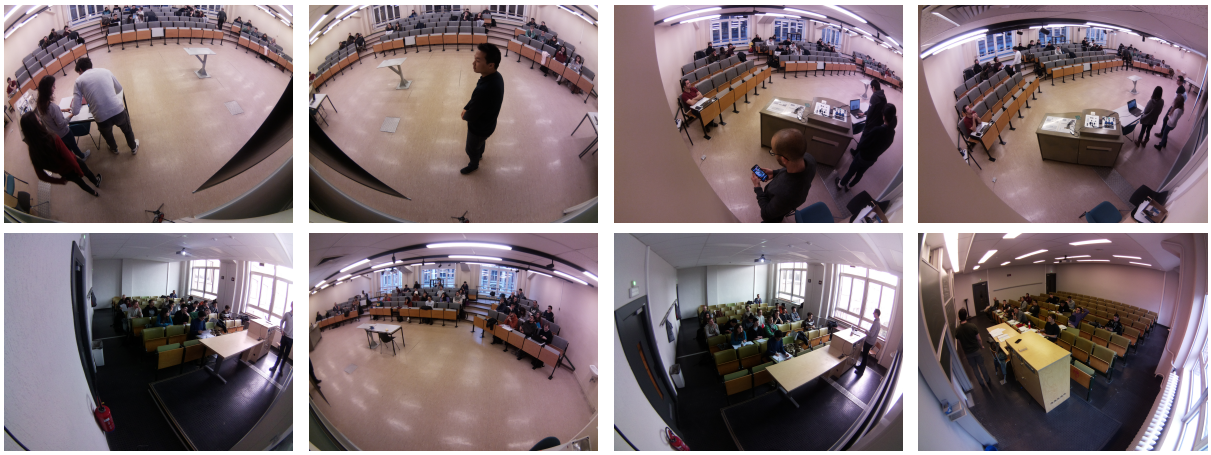

Figure 2: "Basic images" of our dataset, from 1 to 8, left to right, up to down

Additionally, we decided to expand our "Basic Image Dataset" by producing some new pictures by computer treatment, all of them based on those 8 basic images. First, we decided to apply the formula given below [1] to all our base images to manipulate contrast and luminosity, and produce a good variety of new images for each capture, in order to see if any of the algorithms would react differently to them, and hopefully improve our detection ratio. Second, we decided to undistort all of those images (including the original one), using an open source software [Chi18]. We did this because we knew that the camera (and the lens we added to it) produced some distortion in the captured images. We wanted to test if the undistorted images gave better results than the distorted ones. However, obtaining these new pictures wasn't so straightforward. We used an open source code based on an OpenCV example to obtain some calibration parameters from chessboard images (freely given for this purpose by OpenCV). But these parameters were only applicable for low resolution pictures of 640x480 pixels. This forced us to downsize and later enlarge our pictures, which resulted in some inevitable loss of quality. All these newly produced pictures would be called **"Derivated dataset"**.

Here we summarize the parameters of the captured images, and the parameters of the treated images:

Image Number	Contrast (α)	Brightness (β)	Distorsion
1-8	1-3 (steps of 0.25)	0-70 (steps of 10)	yes/no

Figure 3: Processing parameters

Image Number	Camera height	Type of light	Camera Position
1-8	High/Low	Natural/Artificial	Center/Corner

Figure 4: Capture parameters (height is referenced to the height of a door)

$$g(x) = \alpha f(x) + \beta \tag{1}$$

Figure 5: Brightness (β) and contrast (α) transformation. $\alpha = 1$ and $\beta = 0$ represents the unmodified picture

In addition to these basic images, we kept, for each capture, ten to twenty more pictures. In these small **Capture datasets**, we could appreciate "up and down" movement of faces (when looking at the notes or at the board) that were made by students during the time frame of the capture. This behaviour gave us a good variety of images as well, but this time from the same capture, which would be useful for analysis.

2.2.4 External factors

Last thing we had to discuss was the influence of projectors and lights inside the rooms. Many teachers project text slides during lessons, so we had to check if our camera could be dazzled. We confirmed that this wasn't the case. The projector would only appear as a bright small circle in the captured images. We also tested the influence of natural and artificial lights in the room. The captured images had a different look depending on the type of light, so we decided to take this into account as a parameter of the final image.

3 Results

3.1 Definitions

Before showing the results obtained, we will define some intuitive and non-totally objective features that we used to categorize the different spatial head positions we encountered:

- Actual faces: Faces that we were able to count using our human vision
- Detected faces: Faces that were enclosed by a rectangle by the algorithm
- Undetected faces: Faces that were not enclosed by the algorithm, but that we were able to see as humans, no matter how hidden the face was.
- Cut faces: Faces that are not entirely visible because somebody or something in the room hid them (we will say that both eyes are not visible due to the cut). This can be another person, or the person's own hand or arm.
- Edge faces: Faces that are partially cut by the edges of the picture (no matter how much of the face is visible).

- Near faces: Faces that are less than 1 meter away from the camera.
- False positives: Detected entities by the algorithm that are not actual faces.
- Turned away faces: Faces that are turned by at least 90°with respect to the camera angle (either by the side, or by looking up or down), and whose eyes are not visible.

We defined these particular features because they were quite frequently encountered during the lessons we recorded, as people might look down to their desk or to their friend at the side (producing a turned away face), and they might have their face hidden by someone sitting in front of them, or by their hand or arm (cut face). Edge faces also appear when the room is too big to fit the camera's panoramic reach. Regarding the rules used to count people in this first experiment, we decided to take the teacher and/or any students that would be standing up, or back to the camera, into account as another member of the room. This is because the purpose of this first experiment was to get an idea of how algorithms behaved in function of different image modifications, and different classroom scenarios. Getting the exact number of students detected wasn't a priority yet.

3.2 Basic dataset

We will first show the results for the algorithms run on the "basic dataset" and their "derivated" images. Here we present the table with the best detection ratios achieved for each picture, with its parameters shown:

Dataset number	Algorithm	Actual faces	Type of light	Contrast(α)	Brightness(β)	Distorsion (y/n)	Detected Faces	Undetected Faces	Camera angle	Height	Turned away faces	Cut faces	Edge faces	Near faces	False positives	Ratio of detection (%)
1	FCHD	28	Artificial	1,5	70	n	13	15	center	Low	3	1	3	3	0	46
	ETF	28	Artificial	1	0	n	24	3	center		3	1	3	3	0	86
2	FCHD	20	Artificial	2	20	y	18	2	center	Low	0	0	1	1	0	90
	ETF	20	Artificial	1	0	n	19	1	center		0	0	1	1	0	95
3	FCHD	27	Artificial	1,25	30	n	18	9	corner	Low	3	0	0	1	0	67
	ETF	27	Artificial	1	0	n	26	1	corner		3	0	0	1	0	96
4	FCHD	27	Artificial	1,25	10	n	19	8	corner	High	3	0	0	0	0	70
	ETF	27	Artificial	1	0	n	24	3	corner		3	0	0	0	0	89
5	FCHD	19	Natural	1,75	60	n	17	2	corner	Low	0	2	1	0	0	89
	ETF	19	Natural	1	0	n	17	2	corner		0	2	1	0	0	89
6	FCHD	37	Artificial	2,75	20	n	27	10	corner	Medium	1	0	0	0	0	73
	ETF	37	Artificial	2,25	10	n	34	3	corner		1	0	0	0	0	92
7	FCHD	19	Natural	1,25	40	n	18	1	corner	High	0	0	0	0	0	95
	ETF	19	Natural	1	0	n	18	1	corner		0	0	0	0	0	95
8	FCHD	14	Artificial	3	50	n	14	0	corner	High	1	0	0	0	0	100
	ETF	14	Artificial	3	10	n	14	1	corner		1	0	0	0	1	93

Figure 6: Results table

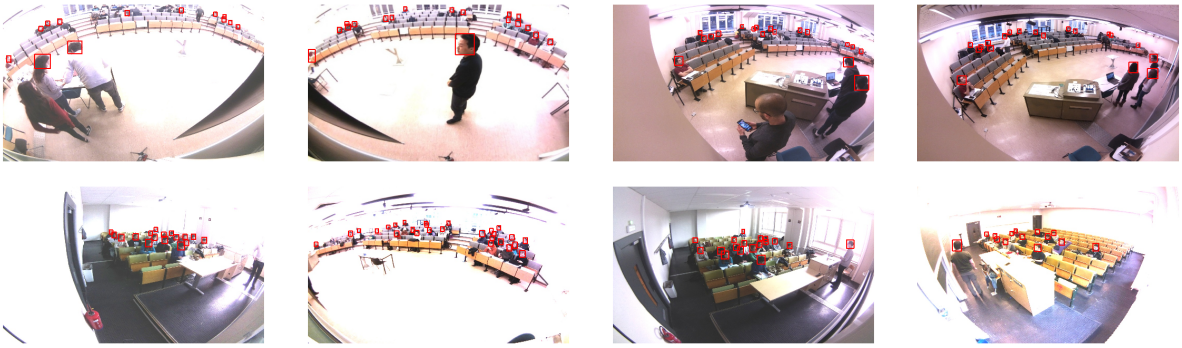

Figure 7: Winners for FCHD

Figure 8: Winners for ETF

As we can see, the algorithm that gave the best detection ratio for most images was ETF. We can also see that we required almost no image processing to produce the best results with ETF, only needing at 2 occasions, to add a little bit more contrast and luminosity to the image to detect some elusive faces. This algorithm did a great job in recognizing partially cut faces, like in image 1. However, we have found that this algorithm has sometimes problems to recognize the people that are the furthest away from the camera, like in image 5. It also struggled with turned aside people (like in images 1, 3, 4 and 8), and with hidden people (like in image 7), regardless if they were close to the camera or not.
On the other hand, FCHD gave relatively disappointing results. The detection ratio was sometimes too low to give an accurate idea of the actual number of people present. To "catch" as many people as possible, we had to heavily tailor the images, by bringing them to uncomfortable levels of brightness, and even sometimes undistorting them. This algorithm also struggled to recognize distant people. The need to work with 1000x600 images hindered even more the task of detecting those distant faces. However, this algorithm excelled in detecting both turned aside people, and hidden people. The fact that this algorithm recognized the very difficult hidden student at the right of image 7, and the turned

away student at the front seat of image 8, is remarkable.
As some general rules on how to optimize the detection ratio, we would suggest to place the camera high enough to minimize cut or hidden faces, but low enough to keep a good resolution of the zone of the picture that we want to analyze. Concerning the positioning of the tripod, the more centered along the front wall, the better (as we would minimise "turned away" and "far-away" faces). It would be a good idea to make sure that the teacher doesn't enter the field of view of the camera during the capture, as he might appear in some images and not in others. If the camera is placed in a very dark room, we would suggest to slightly increase the contrast of the image (placing it around the value 2) and then adjust the brightness (as its the most deviating parameters of all). We didn't find much difference in performance for natural light compared to artificial light.

3.3 Single capture

As we saw in the previous section, "turned aside people" represented a common issue for face detection in the particular environment of this project. We tried to find a solution for this problem by analyzing the group of images we possessed from the same capture. We observed that the natural "up and down" movement of the student's heads in the course of time led to the probable capture of at least one image that would have everybody's head recognizable by the algorithm. We decided to run the algorithm we found was the best in the previous section (ETF) on all the images from some of our captures. We didn't apply any filter to those images, as we found the ETF algorithm to work extremely well under any environmental condition. This is what we obtained for capture 7:

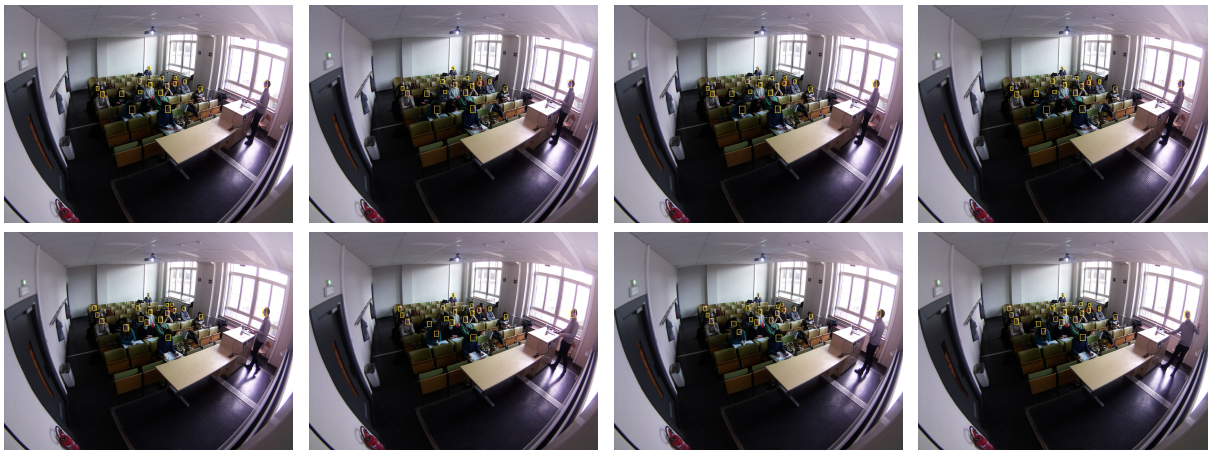

Figure 9: Evolution of detected faces in a 19 people room (capture 7): From 17 (upper row) to 18 and 19 (lower row)

3.4 Complete test

Now that we had a good idea about the factors influencing successful detection, we can describe the methodology that should be used by our automata. The input the automata would take would be a set of images belonging to a capture, and a value of contrast and brightness (that the user should specify in function of the findings that were made in the first experiment). The machine will then transform the provided images if the user wished to do so, and run the detection. It will select the top 30% in terms of number of detected faces, and will output the most prevalent number in that sample as a final guess about how many people are inside the room.
We decided to make a table with the results of our automata for some of the captures (and also an additional one we didn't use in the past experiments), without modifying contrast or brightness. We decided to select a sample where the number of visible people would be constant (including the teacher if he appeared in every picture), so that the algorithm could output a more accurate guess:

Capture Number	Actual faces	Number of pictures	Contrast	Brightness	Top %	Automata output	Camera height	Camera Angle
2	24	44	1	0	20	**24**	Low	Center
6	37	9	1	0	30	**37**	Medium	Corner
7	19	19	1	0	20	**19**	High	Corner
8	13	10	1	0	20	**13**	High	Corner
9	14	12	1	0	20	**14**	Medium	Corner

Figure 10: Counted people table (for top 20% or 30% in this case). We obtained 100% success rate for all captures

Obviously, to extract accurate conclusions about the percentage of correct detections in a capture, we would need to use much bigger samples, and make some statistical studies, briefly discussed in the next section. We didn't have enough data to be able to do it at this stage.

4 Possible improvements

The main improvement that could be made to this project would be to make a statistical study on how often, on average, all the students are recognizable by the ETF algorithm in a single picture, for an arbitrary number of captures, done in an arbitrary period of time. Then we would also need to find the probability of encountering a false positive among the images having the highest counts, to help in adjusting our first parameter. If we could know these parameter, the process of recognizing every single student present in the room could be automatically done, with a very high probability of success.
Other possible improvements for this project would be:

- A better undistortion process, by using intrinsic parameters of our camera.

- Better hardware to run faster the algorithms on bigger datasets.
- Finding a way to run the code on the RaspberryPi or on another hardware that could be used to take the pictures as well.
- Implement localized contrast and brightness modifications, only in "difficult" zones of the image
- Build our own weightfile with a sufficiently big dataset.

5 Conclusion

Accurately counting how many people are present inside a classroom using artificial intelligence and computer vision seemed like a challenging task. After running two experiments with some open source algorithms, we got to the conclusion that this task is very realistically feasible. If we install our camera in an appropriate spot, with a sufficient height, and we run ETF algorithm over a big enough dataset, we can make the program output a number that would approach very closely the real number of students. We could improve this number even more if we carried a larger statistical study on the probability of detecting the exact number of people on single images, and on probabilities of false positives.

References

[AD18] Alexandre Attia and Sharone Dayan. “Detecting and counting tiny faces”. In: *arXiv:1801.06504 [cs, stat]* (Jan. 2018). arXiv: 1801.06504. URL: http://arxiv.org/abs/1801.06504.

[Chi18] ChihaoZhang. *ChihaoZhang/camera-calibration-and-image-undistortion*. original-date: 2018-05-26T08:11:38Z. May 2018. URL: https://github.com/ChihaoZhang/camera-calibration-and-image-undistortion.

[VC19] Aditya Vora and Vinay Chilaka. “FCHD: Fast and accurate head detection in crowded scenes”. In: *arXiv:1809.08766 [cs]* (May 2019). arXiv: 1809.08766. URL: http://arxiv.org/abs/1809.08766.

[Att] Alexandre Attia. *alexattia/ExtendedTinyFaces*. original-date: 2017-12-27T18:24:52Z. URL: https://github.com/alexattia/ExtendedTinyFaces.

[cyd] cydonia. *cydonia999/Tiny_Faces_in_Tensorflow*. original-date: 2017-10-15T10:18:43Z. URL: https://github.com/cydonia999/Tiny_Faces_in_Tensorflow.

[Vor] Aditya Vora. *aditya-vora/FCHD-Fully-Convolutional-Head-Detector*. original-date: 2018-09-23T05:39:57Z. URL: https://github.com/aditya-vora/FCHD-Fully-Convolutional-Head-Detector.