Energy Saving System using Image Analysis

Rishavraj Prasad , Nihaal Gupta, Mrs. A Saranya
Department of Software Engineering
SRM Institute of Science and Technology, KTR
Email: rishavrajprasad98@gmail.com

Abstract - Up till now, lights in rooms where automated using IR sensors, which is very costly so, we propose an energy efficient and cost effective method that uses image analysis to predict which part of the room has more people and focus more lights on that part of the room.

People counting in dense groups is a challenging problem due to severe occlusions, few pixels per head, cluttered environments , and skewed camera perspectives. Our approach relies on the fact that head is the most visible part of an individual in a dense crowd. As such, head detector can be used to estimate the spatially varying head size, which is the key feature used in our head counting procedure. We leverage the state-of-the art convolutional neural network for the sparse head detection in dense crowd. Finally, the individual patch counts are summed up to obtain the total count. Our approach relies on the fact that the head is the most visible part of an individual in a dense crowd. This is done by using component is a CNN-based head detector that provides a sparse location of heads and their sizes in the images. And using Arduino UNO to automate lighting mechanism.

Keywords—Artificial Intelligence, Deep Learning, Crowd Estimation, Dense Crowd, People Count ing, Head Detection , Home Automation

I. INTRODUCTION

An important problem of crowd analysis is an accurate estimate of the crowd size. Many algorithms have been proposed in the literature for crowd size estimation. Previously designed algorithms can be broadly categorized into expert-based and learning-based approaches. Expert-based approaches make use of different hand-crafted features extracted from the scene for crowd counting. For low-resolution images or extremely dense crowds. explicit head counting using image features is very challenging as each head has no more than 10-15 pixels. To solve these problems, we divide the image into rectangular patches and use a CNN-based head detector to estimate the head size in each patch. Assuming the perspective distortion is minimal within an image patch, we can then estimate the number of people in the patch. To increase the robustness of the algorithm, we further use a SVM-based binary classifier to classify patches into crowd and not-crowd, and apply weighted averaging to estimate counts for crowd patches that do not have a reliable head size estimate.

To validate the performance of our algorithm, we compare it with seven state-of-the-arts algorithms on three publicly available datasets.

The main contribution of the paper is a crowd size estimation method that does not rely on any labeled training data of crowded scenes. Our method requires only a generic head detector that can be trained by any images with multiple people. And then based on number of people in room we will manipulate the intensity of light of each room, This will provide safe, energy-efficient and cost effective method for home automation.

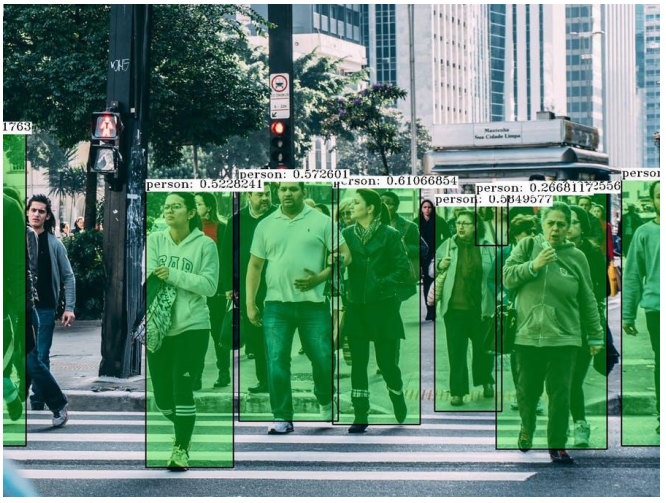

Figure 1: People Counting

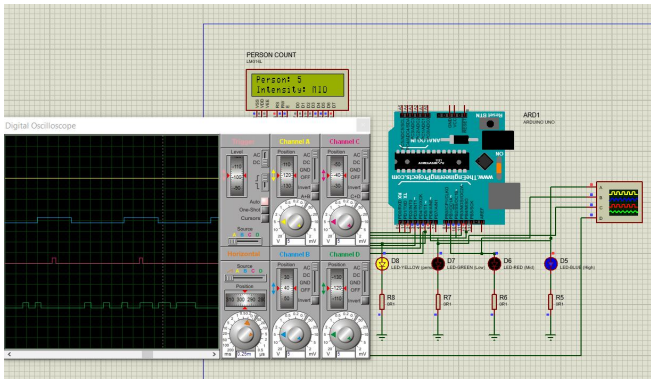

Figure 2: Arduino Uno manipulating Intensity of Led

II. RELATED WORK

Counting by detection is not very accurate when the crowd is dense and the background clutter is high. To overcome these problems, counting by regression is used wherein the features extracted from the local image patches are mapped to the count. Here, neither segmentation nor tracking of individuals is involved. One of the earliest attempts involves extracting the low-level features such as edge details, foreground pixels, and then apply regression modelling to it by mapping the features and the count. Let's discuss two papers to see how regression is used in various scenarios, Cumulative Attribute Space for Age and Crowd Density Estimation. Here, a regression model is learned only when sparse and imbalanced data are available. A cumulative-attribute based regression model is used to map the features extracted from sparse and imbalanced images onto a cumulative attribute space.

This model is used when there's a need to apply regression onto various localised regions in an image. Rather than training a multi-output regression model, a single regression model is used to estimate people in various localised regions, i.e. the model learns the functional mapping between interdependent low-level features and multi-dimensional structured outputs.

III. METHODOLOGY

Dataset:-

Dataset		Num of Images	Min People Count	Max People Count	Avg Count	Total
UCF_CC_50 [2]		50	94	4543	1279.5	63,974
AHU-Crowd [29]		107	58	2201	428.1	45,807
Shanghai Tech [30]	Part A	482	33	3139	501.4	241,677
	Part B	716	9	578	123.6	88,488

The dataset particulars are given in Table I. The UCF CC 50 , AHU-Crowd, and ShanghaiTech Part A contain instances of dense crowds, while the ShanghaiTech Part B contains low to medium density crowds. The recently released AHU-Crowd dataset consists of 107 color images collected from the web, with a total of 45,807 individuals annotated. On average, there are 428 people per image with crowd counts ranging from 58 to 2,201 people.

Architecture diagrams:-

We prepared several diagrams representing the functions and workings of the system. Those diagrams are, Data flow diagram, Sequence diagram, Class diagram, Relational model and ER diagram. It comprises of four major components. The first component is a CNN-based head detector that provides a sparse location of heads and their sizes in the images. The second component is a feature classifier the image is first divided into equal-size rectangular patches, which are categorized as crowd or not crowd by a SVM classifier on SURF features. The third component is a regression module that estimates the head count for each crowd patch based on its spatial coordinates and estimated head sizes. This is resolved by the fourth component in which the counts for these crowd patches are estimated by the spatially dependent weighted average of the counts from the neighboring eight patches. The final step is to sum all the individual patch estimates to get a total count for the entire image. In the following subsections, we detail each component of the algorithm.All the architecture diagrams and the model architecture will be provided below.

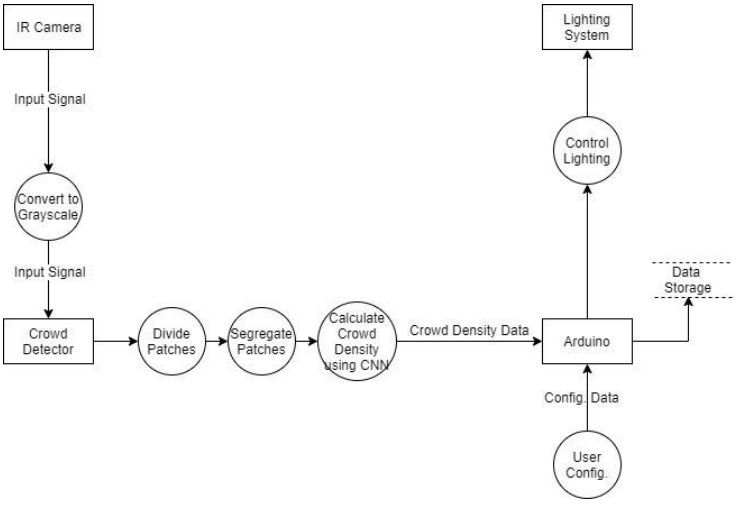

Figure 3. Data Flow Diagram

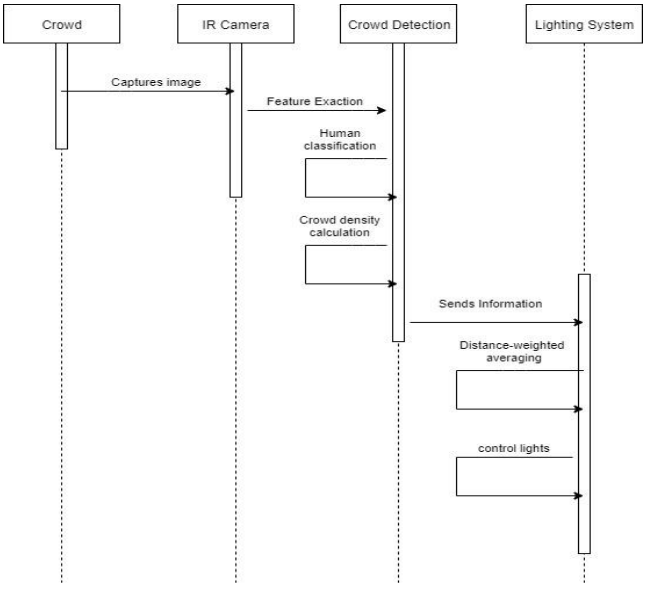

Figure 4. Sequence Diagram

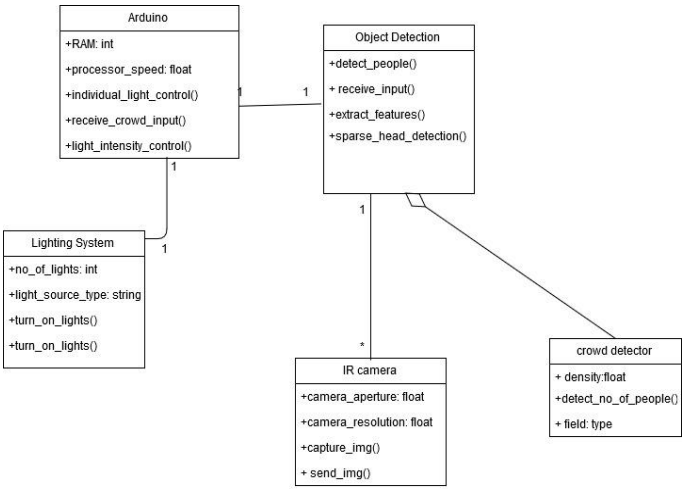

Figure 4. Class Diagram

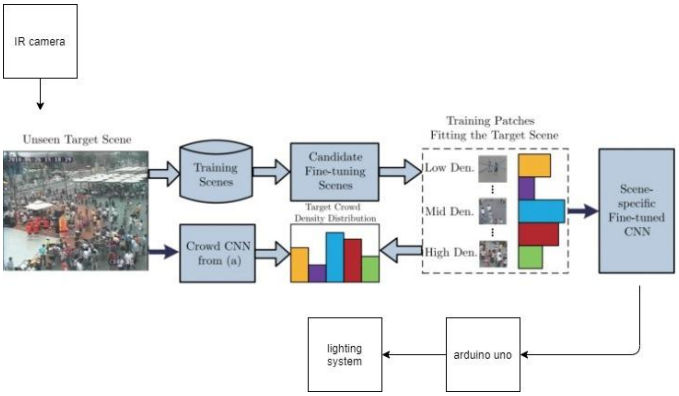

Figure 7. System Architecture

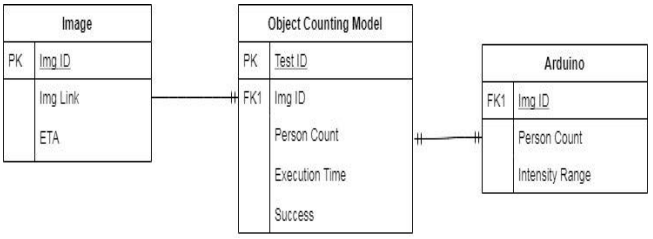

Figure 5. Relational model

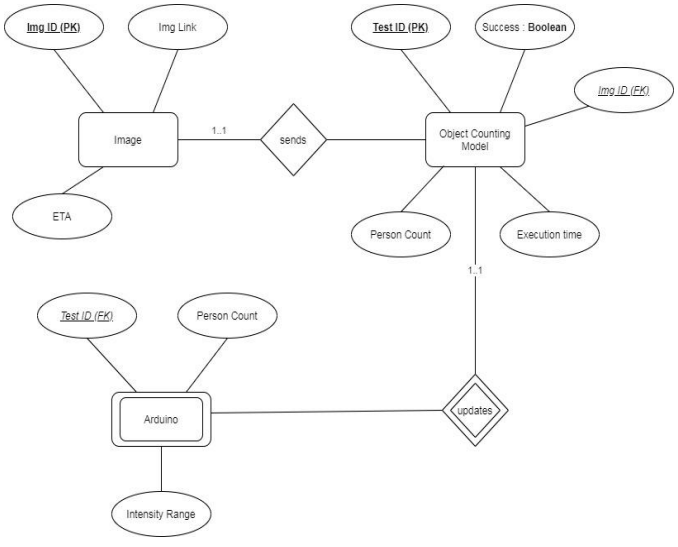

Figure 6. ER Diagram

CNN:Convolutional Neural Networks-

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. An average CNN has three types of layers: Convolution layer which is used to highlight various features in an image such as edges or shapes. ReLu layer, it is a layer with an activation function the most commonly used activation function being the Relu activation function. The activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. Pooling layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

The algorithm uses selective search (fast mode) for generating region proposals. Since we expect our test cases to have smaller head sizes we tune the parameters of the head detector to best suit our scenario. Instead of using the minimum box size of 20 pixels as in , we set it to 5 pixels so that region proposals with height and width less than 5 pixels are discarded. We also set the initial segmentation size parameter k = 50 to ensure that only small segments are used. Values for both these parameters are determined empirically based on the fact that the images in our used datasets contain dense crowds and are often low resolution. Hence, it is desirable to choose these parameters such that we are able to detect the maximum number of heads, especially the smaller ones. We analyzed the box sizes of the detected heads over first 50 images of the ShanghaiTech part. A training set. The minimum box area was found to be 50 pixels2 , and accounting for the presence of even smaller heads in some of the most crowded and low-resolution images, we chose the minimum box width as 5 pixels. Similarly, choosing a small value of k is desired and the choice was made such that our system with 16 GB of RAM could classify all the generated region proposals. Smaller values of k generated more region proposals, especially smaller ones. However, it didnt have any significant effect on head detection performance. Once determined, these parameters were kept the same for all test images.

IV. RESULTS

Once all the information was loaded into the program and the data manipulated appropriately the system could be run. We are using online application nanonets, which can upload your data, annotate it, set the model to train and wait for getting predictions through a browser based UI without writing a single line of code, worrying about GPUs or finding the right architectures for your deep learning models. This provide an easy interface to calculate the number of people and then after successful counting the output is then feed to the arduino where depending on the range of people the led bulb available in room is adjusted at the interval of 1 minute. It can also switch on the bulbs in dark room successfully as soon someone enters the room.

```
import unittest
import project

class TestProject(unittest.TestCase):

    def test_resp_fun(self):
        self.assertEqual(project.resp_fun({'urls' : ['https://media.istockphoto.com/photos/side-view-of-people-walking-in-queue-

    if __name__ == '__main__':
        unittest.main()


In [2]: runfile('C:/Users/madri/OneDrive/Desktop/test_project.py',
wdir='C:/Users/madri/OneDrive/Desktop')
Reloaded modules: project
Persons: 5
.Persons: 5

----------------------------------------------------------------------
Ran 1 test in 9.953s

OK

In [3]:
```

Figure 8. Training and testing positive results

```
import unittest
import project

class TestProject(unittest.TestCase):

    def test_resp_fun(self):
        self.assertEqual(project.resp_fun({'urls' : ['https://media1.s-nbcnews.com/i/newscms/2018_37/1366982/pedestrians-walk-st

    if __name__ == '__main__':
        unittest.main()


In [2]: runfile('C:/Users/madri/OneDrive/Desktop/test_project.py',
wdir='C:/Users/madri/OneDrive/Desktop')
Reloaded modules: project
Persons: 8
FPersons: 8

======================================================================
FAIL: test_resp_fun (__main__.TestProject)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:/Users/madri/OneDrive/Desktop/test_project.py", line 7, in
test_resp_fun
    self.assertEqual(project.resp_fun({'urls' : ['https://media1.s-
nbcnews.com/i/newscms/2018_37/1366982/pedestrians-walk-street-stock-today-
main-180911_a1bec6c0803b9e2dd86e124c9b9fba15.jpg']}),6)
AssertionError: 8 != 6

----------------------------------------------------------------------
Ran 1 test in 12.352s

FAILED (failures=1)

In [3]:
```

Figure 9. Training and testing negative results

V. CONCLUSION

In conclusion, a model was developed that uses the deep learning functions of convolutional neural networks to detect people in the room by using sparse head calculations make an assessment and then control the intensity of each bulb in room. Potential setbacks have been identified in the process that will let us improve upon the process as the research moves forward. This research also provides insight into the testing difficulties associated with large datasets of occlusion images. Most research in this field has focused on small, controlled datasets. With these datasets, irregular features may not have been much of an issue. But as the size of the dataset increases, which causes the model to be a bit biased Although this is a potential difficulty, it is desired because a model utilized by the anyone for home automation.

VI. REFERENCES

1. People Counting in Dense Crowd Images Using Sparse Head Detections
2. Device-Free People Counting in IoT Environments
3. Background-Subtraction Algorithm Optimization for Home Camera-Based Night-Vision Fall Detectors
4. Binary Quadratic Programing for Online Tracking of Hundreds of People in Extremely Crowded Scenes
5. People Counting Based on an IR-UWB Radar Sensor
6. Dense People Counting Using IR-UWB Radar With a Hybrid Feature Extraction Method