A MAJOR PROJECT REPORT

ON

PRECISION OBJECT COUNTING SYSTEM

Submitted to

Sri Indu Collee of Engineering and Technology

in partial fulfillment of the requirements for the award of a degree of

BACHELOR OF TECHNOLOGY

IN

CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by

GAJJI CHAKRAPANI	**20D41A6620**
BETHI DIVYA	**20D41A6609**
VUPPALA LIKITHA	**20D41A6660**
AVULA RISHIKESH SHIVADHAR REDDY	**20D41A6604**

Under the esteemed guidance of

Mrs. SANDHYA BOLLA

(Associate Professor)

DEPARTMENT OF C S E - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution under UGC, accredited by NBA, Affiliated to JNTUH)

Sheriguda, Ibrahimpatnam (2023-2024)

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution under UGC, Accredited by NBA, Affiliated to JNTUH)

DEPARTMENT OF CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

Certified that the Mini Project entitled "**PRECISION OBJECT COUNTING SYSTEM"** is a bonafide work carried out by **GAJJI CHAKRAPANI (20D41A6620), BETHI DIVYA (20D41A6609), VUPPALA LIKITHA (20D41A6660), AVULA RISHIKESH SHIVADHAR REDDY (20D41A6604)** in partial fulfillment for the award of **Bachelor of Technology** in **CSE-Artificial Intelligence and Machine Learning** of SICET, Hyderabad for the academicyear 2023- 2024. The Project has been approved as it satisfies academic requirements concerning the work prescribed for **IV YEAR, II-SEMESTER** of **B. TECH** course.

Mrs. SANDYA BOLLA **Prof. G. UMA MAHESWARI**

(INTERNAL GUIDE) (HEAD OF THE DEPARTMENT)

EXTERNAL EXAMINER

DECLARATION

We declare the project work "**PRECISION OBJECT COUNTING SYSTEM**" was carried out by me and this work is not same as that of any other and has submitted to anywhere else for the award of any other degree.

Signature of the candidate:

1…….

2…….

3…….

4…….

ACKNOWLEDGMENT

With great pleasure, we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a success. We thank the almighty for giving us the courage & perseverance in completing the project.

We are thankful to the Principal Prof. **Dr. G. Suresh**, for permitting us to carry out thisproject and for providing the necessary infrastructure and labs.

We are highly indebted to, **Prof. G. Uma Maheswari**, Head of the Department of CSE-Artificial Intelligence and Machine Learning, for providing valuable guidance at every stage of this project.

We are grateful to our internal project guide, **Mrs. Sandhya Bolla, Associate Professor** for her constant motivation and guidance given by her during the execution of this project work.

We want to thank the Teaching and non-teaching staff of the Department of C S E - Artificial Intelligence and Machine Learning for sharing their knowledge with us.

Last but not least we express our sincere thanks to everyone who helped directly orindirectly with the completion of this project.

GAJJI CHAKRAPANI	20D41A6620
BETHI DIVYA	20D41A6609
VUPPALA LIKITHA	20D41A6660
AVULA RISHIKESH SHIVADHAR REDDY	20D41A6604

ABSTRACT

Object detection (OD) in crowded scenes is a challenging task since objects are densely distributed and partially overlapped. In this paper, we propose a novel OD method by fully exploring the information provided by the image and its estimated density map. Our proposed OD method consists of two main stages. Initial object locations are firstly computed based on object spatial distribution information obtained from the estimated density maps. Inspired by the human visual attention mechanism, a saliency map which offers object boundaries is then employed to accurately estimate the bounding boxes with the support of the estimated initial object locations. We propose the task to estimate the density map of objects from single image with unknown perspective map. We follow the recent progress in object counting through density map estimation. Object density map estimation is usually suffered from scale variance of objects caused by unknown perspective. In addition, the background and irrelevant objects in the image lead to artifacts in the resulting density maps, which build up the error when heat maps are needed by aggregating density maps.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

1. INTRODUCTION

1.1 MOTIVATION:

Dense object counting tasks, including crowd counting, vehicle counting, and general object counting, aim to estimate the number of objects in the image. Counting tasks have practical usage for understanding crowded scenes. For example, crowd counting can be used to prevent accidents caused by overcrowding and estimate the crowd flows in station. And vehicle counting can be used for traffic management on roads or in parking lots. General object counting is useful for the management of goods in the supermarket, farms and factories.

Although counting tasks are important and useful, the real usage is still limited since dense object counting is challenging. One of the main challenges is scale variation. Since the scale of people varies dramatically in images and across different images, it is difficult to extract features for density regression. Another challenge is the occlusions among people since only a small part of each person may be visible in crowd images. Complex backgrounds may also hurt the counting performance, and the domain gap between scenes in datasets and the real world scenes also limits the usage of counting algorithms.

Current state-of-the-art methods use crowd density maps to achieve superior counting performance. Density maps are an intermediate representation, where the sum over any region in the density map indicates the number of people in the region. First, the density maps are generated from the dot annotations, where each dot indicates a person's location. Second, given the input image, algorithms are designed to predict the density map which is then summed to obtain the count.

In practice, the method for generating the density maps is crucial for crowd counting. Improperly generated density maps may dramatically hurt the counting performance – the choice of the kernel bandwidth or kernel shape used to generate the density map is often dataset dependent, and such choices often do not work across different datasets.

In the era of deep learning, we may consider current density maps as a hand- crafted intermediate representation, which is used as a target for training deep networks to count. From the standpoint of end-to-end training, these hand- designed intermediate representations may not be optimal for the particular network architecture and particular dataset.

1.2 OBJECTIVES:

In this, we call these two steps density map generation and density map estimation,

respectively. Most works focus on density map estimation and ignore density map generation. Many different deep networks have been proposed to improve density map estimation, e.g., using different kernel sizes or image pyramids to handle scale variations, or using context or prior information to handle occlusions. Although density map estimation is well-studied, the generation of density maps is often overlooked and uses handcrafted designs without adequate investigation and analysis. The simplest approach to obtain a density map is to convolve the annotation dot map with a Gaussian with fixed width, i.e., place a Gaussian on each dot. Other works scale the Gaussian bandwidth according to the scene perspective, or adaptively use the local congestion level (or distance to nearest neighbors) uses human-shaped kernels, composed of two Gaussians, but is less popular since the body of the person is often occluded in crowd images.

Current state-of-the-art methods use crowd density maps to achieve superior counting performance . Density maps are an intermediate representation, where the sum over any region in the density map indicates the number of people in the region. First, the density maps are generated from the dot annotations, where each dot indicates a person's location. Second, given the input image, algorithms are designed to predict the density map , which is then summed to obtain the count.

In practice, the method for generating the density maps is crucial for crowd counting. Improperly generated density maps may dramatically hurt the counting performance – the choice of the kernel bandwidth or kernel shape used to generate the density map is often dataset dependent, and such choices often do not work across different datasets. In the era of deep learning, we may consider current density maps as a hand-crafted intermediate representation, which is used as a target for training deep networks to count.

2. LITERATURE SURVEY

2.1 LITERATURE SURVEY-1

Title	Benchmark Data and Method for Real-Time People Counting in Cluttered Scenes Using Depth Sensors
Authors	Shijie Sun , Naveed Akhtar , Huansheng Song, Chaoyang Zhang, Jianxin Li , and Ajmal Mian
Published Year	2019
Efficiency	Proved its efficiency in large volumes of extremely imbalanced data. Construct a comprehensive feature vector. Outperforms from previous schemes in terms of recall, precision, and time complexity.
Drawbacks	Poor Application Performance. Not based on real-time datasets. Cannot exploit the new features.
Description	Vision-based automatic counting of people has widespread applications in intelligent transportation systems, security, and logistics. However, there is currently no large-scale public dataset for benchmarking approaches on this problem. This paper fills this gap by introducing the first real-world RGB-D people counting dataset (PCDS) containing over 4500 videos recorded at the entrance doors of buses in normal and cluttered conditions. It also proposes an efficient method for counting people in real-world cluttered scenes related to public transportations using depth videos. The proposed method computes a point cloud from the depth video frame and re-projects it onto the ground plane to normalize the depth information. The resulting depth image is analyzed for identifying potential human heads. The human head proposals are meticulously refined using a 3D human model. The proposals in each frame of the continuous video stream are tracked to trace their trajectories. The trajectories are again refined to ascertain reliable counting. People are eventually counted by accumulating the head trajectories leavingthe scene. To enable effective head and trajectory

	identification, we also propose two different compound features. A thorough evaluation on PCDS demonstrates that our technique is able to count people in cluttered scenes with high accuracy at 45 fps on a 1.7-GHz processor, and hence it can be deployed for effective real-time people counting for intelligent transportation systems. We concatenate the above mentioned four geometric features into a vector in R13. Notice that, although we do consider varied areas of IH in the above mentioned features, the compound feature only accounts for the information that is local to individual rectangles. In the real-world scenarios, the relative locations of the rectangles (that we suspect to contain human heads) can useful information about a bounded object being a human head or not. Therefore, we further define NRDF to account for this additional information. For each rectangle in FH , we compute NRDF as a vector in the 3D- space that is directed towards the center of the rectangle from the center of its nearest rectangle in our current set of head proposals. This feature is further illustrated in Fig. 6. The resulting NRDF R3 is concatenated with the above mentioned feature vector to finally arrive at our compound feature vector in R16. Background removal is a major task in many surveillance related problems. For our approach, reliable background subtraction is necessary for the success of subsequent processing of video frames. Therefore, we separately analyze the performance of our method for this task. We use the popular Gaussian mixture-based background segmentation method (MOG) and the K-nearest neighbors (KNN) based method to benchmark our technique. We note that other approaches for background subtraction also exist, however the selected baseline methods are chosen for their well- established effectiveness for the depth videos. To analyze the performance of our method for human head identification, we first manually labeled 12,148 rectangle proposals in height images for people entering the buses as ‘heads’ and ‘non-heads’. These proposals were generated automatically by the method in Sec. We can argue that the employed classifiers are able to identify human heads in the proposals successfully. We note that the classification performance depicted by is better for the people exiting buses than for the people entering buses. The reason behind this phenomenon is that providing the ground truth we

	only labeled those proposal rectangles as 'heads' that bounded complete human heads. For the case of people entering the buses, many half- heads appeared in the frames due to queuing of people on bus doors. On scrutiny, we found that most of those heads resulted in false positive identifications in our experiment. However,this is not problematic for the overall approach because the final results rely more strongly on tracking of heads on multiple frames, and the half- heads eventually transform into complete heads in the subsequent video frames. We also provide the details of precision, recall and the f1-scores for our head identification experiment.

2.2 LITERATURE SURVEY-2

Title	Scale Driven Convolutional Neural Network Model for People Counting and Localization in Crowd Scenes
Authors	SALEH BASALAMAH, SULTAN DAUD KHAN , AND HABIB ULLAH
Published Year	2019
Efficiency	Outstanding Learning Capabilities Effectively improve prediction accuracy. Reduces the consumption of hardware resources
Drawbacks	Can't learn relation between factors. Do not take account of the influence features Maximizes the complexity of the problem
Description	Counting and localization of people in videos consisting of low density to high density crowds encounter many key challenges including complex backgrounds, scale variations, nonuniform distributions, and occlusions. For this purpose, we propose a scale driven convolutional neural network (SD-CNN) model, which is based on the assumption that heads are the dominant and visible features regardless of the density of crowds. To deal with the problem of different scales of heads in different regions of the videos, we annotate a set of heads in random locations of the videos to develop a scale map representing the mapping

	of head sizes. We then extract scale aware proposals based on the scale map which are fed to the SD-CNN model acting as a head detector. Our model provides a response matrix rendering accurate head positions via nonmaximal suppression. For experimental evaluations, we consider three standard datasets presenting low density to high density crowd scenes. Our proposed SD-CNN model outperforms the state-of-the-art methods in terms of both frame-level and pixel- level analyses. After generating scale-aware proposals, the next step is to classify each proposal into two classes, i.e., head and background. Our detection network follows the classical R- CNN mode [12] and instead of using selective search for proposal generation, we use scale-aware proposals. Before feeding to the network, we extend the bounding box of each proposal by a small margin and then image patch corresponding to each proposal is resized to Fit the input layer of the CNN. For the head detection, we keep the square- like aspect ratios < 2 for all bounding boxes. The classical R-CNN is based on AlexNet architecture which is pretrained on ImageNet [9] dataset. In addition to AlexNet, we used several other alternatives. From the experiment, we noticed that VGGS slightly outperforms AlexNet but was slower in both training and testing. In this section we discuss both qualitative and quantitative analysis of the results obtained from the experiments. We evaluate our SD- CNN framework using three publicly available datasets, UCSD dataset ,WorldExpo'10 and UCF- CC-50 . The summary of the datasets is described in Table 1. Generally, these datasets are annotated in a way that can only be useful for evaluating the performance of regression models. Typically, in these datasets, every individual pedestrian is annotated with a dot in the scene. These dot annotations are not suitable for training our SD- CNN model or other detection based methods. Therefore, we annotated each pedestrian with a bounding box that cover whole body of pedestrian. In the same way, we also annotated the head of each pedestrian using the bounding box. After annotation, we then trained different models discussed in Section IV on Titan Xp with learning rate at 0.01 and decrease it by a factor of 10 after the validation error reaches saturation point. In this section, We evaluate both qualitatively and quantitatively the localization performance of our

	framework. In order to quantify the localization error, we associate the center of estimated bounding box with the ground truth location (single dot) through 1-1 matching strategy. We then compute Precision and Recall at various thresholds and report the overall localization performance in terms of area under the curve. In order to estimate the location, we use the same density maps generated by state-of-the-art methods followed by non-maxima suppression algorithm. The results are reported in Table 5. It is obvious that our proposed model presents higher Precision and Recall rates as compared to the state of- the-art methods. These results attribute to the fact that our model generates scale-aware proposals that capture wide range of head sizes in each image. It can also be observed that all other methods present lower rates for UCF-CC-50 dataset as compared to WorldExpo'10 and UCSD datasets. This is due to the fact the UCF-CC-50 dataset contains more dense images with heavy occlusions as compared to World- Expo'10 and UCSD datasets. We also show some qualitative results of our proposed method in Figure 5. From the Figure 5, it is obvious that the sample images from the UCSD dataset represent low density scene. The sample images taken from two different scenes of World- Expo'10 dataset represent medium densities and the images from UCF-CC-50 represent relatively more complex and extreme high density scenes.

2.3 LITERATURE SURVEY-3

Title	Device-Free People Counting in IoT Environments: New Insights, Results and Open Challenges
Author	Iker Sobron, Manuel Velez
Published Year	2018
Advantages	Low Deployment Cost Quick Calculation Time Very easy to implement for multi-class problem
Drawbacks	A lot of pre-development education Existing system is Opportunistic and uncontrollable Difficulties to obtain better performance
Description	In the last years multiple Internet of Things (IoT) solutions have been developed to detect, track, count and identify human activity from people that do not carry any device nor participate actively in the detection process. When Wi-Fi radio receivers are employed as sensors for device-free human activity recognition, channel quality measurements are preprocessed in order to extract predictive features towards performing the desired activity recognition via machine learning (ML) models. Despite the variety of predictors in the literature, there is no universally outperforming set of features for all scenarios and applications. However, certain feature combinations could achieve a better average detection performance compared to the use of a thorough feature portfolio. Such predictors are often obtained by feature engineering and selection techniques applied before the learning process. This manuscript elaborates on the feature engineering and selection methodology for counting device-free people by solely resorting to the fluctuation and variation of Wi-Fi signals exchanged by IoT devices. We comprehensively review the feature engineering and ML models employed in the literature from a critical perspective, identifying trends, research niches and open challenges. Furthermore, we present and provide the community with a new open database with Wi-Fi measurements in several indoor environments (i.e. rooms, corridors and stairs) where up to 5 people

	can be detected. This dataset is used to exhaustively assess the performance of different ML models with and without feature selection, from which insightful conclusions are drawn regarding the predictive potential of different predictors across scenarios of diverse characteristics. Time statistics have been widely employed for feature construction in many application areas. Common statistical metrics such as mean, variance, moments and the like are combined to yield multiple predictors. Following this engineering approach, the authors in defined a new feature, coined as the coefficient of variation of phase, where the ratio between the standard deviation and mean of the i^{th} CSI subcarrier phase \Hi(n) is computed within a time window. Human motion is detected when the averaged ratio of the coefficient of variation of phase falls within a predefined confidence interval. In the coefficient of variation of the normalized CSI amplitudes is computed following the same criterion. In order to quantify the multi-path propagation conditions due to the presence of an intruder, a new metric is derived. This feature consists of the ratio between the kurtosis and the mean of the coefficient of variation of CSI amplitudes. A pre calibrated detection threshold is necessary for each scenario. Additionally, in the Rician K-factor was postulated as a possible predictor, however, it was neglected due to the lack of time and phase synchronization on commodity Wi-Fi devices in order to extract accurate information for the Rician estimator. It has been mentioned previously, this architectural scheme is usually adopted for decreasing the volume of data transmitted from the IoT nodes to the Cloud. However, due to the fact that nodes can be connected to the Cloud via wired links, the adoption of Edge Computing is not motivated here by data traffic limitations, but rather by the required computational effort and by possible latencies that data processing can yield, which might be of relevance for certain time-critical applications demanding the counting service (e.g. intrusion detection). As a result, raw CSI data can be processed at each end device such that some elaborated features or testing decisions are eventually transmitted to the Cloud for further processing and/or storage. In this regard two computation levels are proposed: a) feature extraction and selection can be performed at each IoT node, so that the selected feature set is transmitted and fed to the

	ML model in the Cloud, whose (re-)training algorithm leverages a higher amount of computational resources; b) both FS and learning/testing are performed at the IoT devices, after which predictions are delivered to the Cloud and served therefrom to applications demanding the people counting service. We delve into the design of a device-free people counting IoT framework, which can be regarded as a cyber-physical system in the IoT context. Apart from the regular application of the Wi-Fi systems, IoT nodes can act as sensors by collecting data about the surrounding radio environment. The sensed CSI, necessary for channel decoding in the wireless communication chain, can be converted into meaningful information about the human activity in a monitored area. Wi-Fi IoT devices are usually connected to a local network where one or several routers grant connectivity to Internet. As a result, sensing data can be shared, gathered and analyzed in a cloud platform in order to provide ubiquitous device-free people counting services over IoT deployments.

3. EXISTING SYSTEM

The existing system is designed to automate the process of counting people within a set of images by leveraging various image processing techniques. It begins by mounting Google Drive to access the image files stored there, ensuring seamless integration with cloud storage solutions. Following this, the system loads image paths from Google Drive, enabling easy retrieval and processing of images from a remote location.

Once the images are loaded, they undergo a series of preprocessing steps aimed at enhancing their quality and facilitating more accurate detection of people. These preprocessing steps include cropping the images to isolate regions of interest, resizing them to standard dimensions for uniform processing, and applying gamma correction to enhance image contrast. Additionally, a Gaussian blur is applied to smooth out any noise in the images, while adaptive histogram equalization adjusts the intensity distribution to further enhance contrast. Otsu's thresholding technique is then utilized to separate foreground objects, such as people, from the background, followed by dilation and erosion operations to refine object boundaries and eliminate any remaining noise.

With the preprocessed images in hand, the system employs contour detection algorithms provided by OpenCV to identify and delineate objects within the images. These contours are subsequently used to label and count the objects, specifically focusing on identifying and tallying the number of people present in each image. Finally, the system displays the original images, along with their corresponding processed versions, and provides a count of the people detected in each image. This comprehensive approach ensures that the system not only accurately counts people but also provides visual feedback to users, facilitating further analysis and decision-making based on the results obtained.

3.1 CHALLENGES IN THE EXISTING SYSTEM

The current system operates within a Bid Generative Adversarial Networks (BiGAN) framework, incorporating an encoder, generator (decoder), and discriminator. This framework encodes selected images into a specific distribution, essential for interpolating between two target images. Notably, the encoder and generator are trained separately. To facilitate training of a comprehensive deep learning model to analyze dynamic density maps, diverse datasets with varying sizes, shapes, and time intervals are required. The training samples should offer smooth and continuous dynamic information. To enhance system visualization, techniques such as blending, field representation, and sampling are employed, aiming to balance simplicity and computational efficiency while minimizing user interactions.

The system's optimal performance is observed with color (RGB) images containing dense crowds, typically exceeding 500 individuals. It comprises four key components: a CNN-based head detector providing sparse head locations and sizes, a feature classifier categorizing rectangular patches into crowd or non-crowd using Support Vector Machine (SVM) on Speeded Up Robust Features (SURF), a regression module estimating head counts for each crowd patch based on spatial coordinates and estimated sizes, and a mechanism to handle patches with undetected heads through spatially dependent weighted averaging of neighboring patches. Finally, the total count for the entire image is obtained by summing individual patch estimates. This approach allows for effective crowd counting without assuming the crowd fills the entire image.

Furthermore, to address patches without crowds, a binary crowd/not-crowd classifier is introduced, aiming to prevent overestimation. The system emphasizes the importance of clear requirements, establishing a formal agreement between clients and providers, thereby reducing financial risks and ensuring project adherence to schedule. Python, as an open-source programming language, offers cross-platform compatibility and facilitates collaboration within the programming community. Its coding style emphasizes readability and includes provisions for multiline statements and comments, particularly through documentation strings (docstrings), which serve as self-explanatory notes accessible at runtime. While docstrings are akin to multiline comments, they differ in their utilization within functions, as comments are completely ignored by the Python Interpreter.

4. PROPOSED SYSTEM

AIM OF PROJECT

The primary aim of this proposed system sets the detection threshold and adjusting the camera. The detection results below the threshold, which is usually set from 0.2 to 0.4, will not be counted. For the sake of simplicity, we use the default value of 0.2 in this work. In the actual scene, the camera should be adjusted to the appropriate height and angle.

SCOPE

The proposed system overcomes the above issues with a novel real-time people counting approach dubbed YOLO-PC (YOLO based People Counting). In the proposed system, after the special pre-treatment, adaptive segmentation and feature extraction for the people counting data, the feature vector is used as the inputs of the trained YOLO to classify and statistics of the total number of the people .

OBJECTIVES

The proposed system is a dynamic background subtraction module is first considered to segment moving objects from each captured video frame. In order to overcome light variations, a dynamic threshold value associated with detecting regions of interest from the differentiated image is iteratively calculated according to the distributions of background and foreground pixels in each frame. After obtaining the foreground regions, four states including new, leaving, merged and split are assigned to the detected moving objects according to their appearances in the current frame. In particular, targets identified as states of merge and split further pass through backward tracking for relieving the occlusion effects by investigating the centroid distances among objects in the previous frame. Finally, targets in four states are tagged to yield the results of people tracking and counting.

ADVANTAGES

- Low computational complexity with high efficiency
- Give more precise on the estimation
- Scale and confidence priors are discovered automatically
- Extremely fast counting strategy with high accuracy
- High performance is achieved

CROWD COUNTING

Traditional crowd counting algorithms are based on individual detection and tracking but these methods do not work well for dense scenes. Thus, global regression based methods are proposed to avoid the detection of individuals by directly regressing the number of people However, global regression ignores the spatial distribution of people, and thus density map based methods are proposed to further predict the spatial density map in images and have achieve the outstanding performance.

DENSITY CROWD COUNTING

Individual detection and tracking based counting algorithms rely on the detection algorithms that do not work well under congested scenes. Most of these algorithms detect human heads and shoulders by detection or tracking algorithms. propose to count the number of people by detecting human heads and shoulders based on foreground segmentation.

REGRESSION BASED COUNTING

To avoid explicit detection of individuals, regression methods are proposed to estimate the number of people directly from low-level features like texture, color, and gradient. Chan et al. Propose to count by directly regressing from global features to crowd number using Gaussian process regression. A prior distribution is proposed in to estimate homogeneous crowds. Multiple features are combined in to improve the performance of crowd counting.

DENSITY MAP BASED COUNTING

Density map based methods are currently the most popular approach to crowd counting since the performance can be dramatically improved by utilizing spatial information. Density maps are typically generated by blurring dot maps in which each dot indicates a person in an image. Since density maps are inter- mediate representations, most algorithms generate density maps beforehand by convolving the dot maps with Gaussian kernels with either fixed or adaptive bandwidths. Then, different network architectures are designed to handle various challenges, such as scale changes, improving the quality of density maps, encoding more contextual information, or adapting to new scenarios.

To handle with scale variation of people proposes a multi-column neural network (MCNN) where each column has different kernel sizes to extract multi-scale features. Similarly, SANet proposes to extract multi-scale features with scale aggregation modules, while Kang and Chan

propose to use image pyramid to deal with scale variation in crowd counting. Instead of fusing multi-scale features, switch-CNN proposes to select a proper column with appropriate receptive field. A tree- structured CNN is proposed to handle the diversity of people in crowds proposes a hierarchical encoder-decoder framework to encode multi-scale features, while proposes an attention based framework to filter background and proposes a novel feature fusion strategy.

Another way to improve the quality of density maps and the performance of crowd counting is to use refinement-based algorithms. Ranjan et al propose a two stage counting framework in which the high-resolution density map is estimated based on the low-resolution density map generated in the initial stage, while Sam and Babu propose a feedback mechanism to refine the predicted density map.

ABLATION STUDIES

DENSITY MAPS

We first compare the effectiveness of different traditional density maps and our generated density maps. For fixed bandwidth kernels, the bandwidths are set to 4 and
16.The learned density maps from ADMG/KDMG generally out- perform manually generated density maps including fixed kernels and adaptive kernels. KDMG achieves superior performance over ADMG, since the former preserves the true count in the learned density map.

KERNEL STUDIES

The effect of kernel size k on the density maps produced by KDMG is investigated on ShTech A and B, and the results are presented in Figure 6 (a). For ShTech A, smaller kernels tend to yield better performance since the crowd is dense. In ShTech B, the best performance is achieved when kernel size is 7 since the crowd is less dense. We thus set kernel size to 5 and 7 for ShTech A and B, respectively. On the other crowd datasets, we use k=5 since they have similar average head size to ShTech A. We also set k=5 for the vehicle and general object counting datasets.

REGULARIZATION

The cosine similarity is used as a regularizer for spatial consistency. The effect of the weight in is investigated on ShanghaiTech A and ShanghaiTech B, and the results are shown in Figure 6 (b). On ShanghaiTech A, larger weights tend to generate better performance since people are close to each other in ShanghaiTech A. Under this circumstance, spatial regularization is effective to learn the spatial density distribution.

SELF-ATTENTION MODULE

To evaluate the effectiveness of the self-attention module in ADMG, we compare using the self-attention module with three variants: “image-att” generates attention from the input image; “direct fusion” directly fuses without attention; “naive- fusion” directly sums all density maps. As in the fusion is more effective with self-attention (“self-att”) than with the input image (“image-att”). The possible reason is that the crowd information is directly obtainable from the density maps, whereas this information needs to be decoded and interpreted from the image, which introduces additional complexity and noise.

LOCAL COUNTING PERFORMANCE

To investigate the local counting performance, we evaluate the frameworks based on GAME metric on UCF-QNRF. The result is shown in Table 12. The local counting performance of ADMG is worse than the traditional method, while KDMG achieves better local counting performance than the baseline CSRNet. KDMG has better local performance because it composites a set of kernels with fixed $k \times k$ size, which keeps the local density regions smooth. In contrast, ADMG uses a series of convolution layers that tends to make the density regions more compact , which cause errors near boundaries of the GAME image patches.

VISUALIZATION

To better understand the generation framework, we compare the learned density maps on different datasets with traditional density maps . For small-bandwidth density maps and ADMG density maps (second column), the density only appear on part of the object, which results in sparse density maps. For density maps generated by adaptive kernels , the density is too smooth for sparse objects. The density maps generated by fixed kernel with bandwidth are similar to KDMG density maps , which can better cover the whole objects with less leakage to the background.

We next visualize the learned individual kernels for KDMG and fixed kernels with bandwidth 16 in Figure 8. Overall, the learned kernels are more flat than fixed kernels especially for very dense images in UCF-QNRF. Since the counters use max-pooling layers which will produce translation invariant features, it will be better to have the same density value for shifted patches.

We also fit the profiles of the learned kernels from KDMG to quadratic functions. In particular, we randomly selected 70 images for each dataset. Then, the images are split into several regions based on the height or width, and the average coefficient is calculated for each region. The correlation statistics and p-value are calculated using data from the 70 images. The magnitude of the quadratic coefficient (the coefficient is always negative) can be used to represent the flatness of a kernel. For each dataset, the magnitude of the quadratic coefficient of a kernel versus its x- coordinate (SKU-110K dataset) or y- coordinate (other datasets) is shown in Figure 10, where the origin (0,0) is the top-left corner of the image. We also plot the best fit line (via linear regression), and test whether there is a significant correlation between the coefficient magnitude and the y-coordinate (or x-coordinate), i.e., whether the line's slope is significantly different from zero.

When the images contain dense crowds under perspective effects (as in ShTech A, UCF-QNRF, PUCPR+, and TRANSCOS), we found that the quadratic coefficient's magnitude is negatively correlated with kernel's y-coordinate. In other words, for these scenes, the curvature of the kernel adapts to placement of people in the scene. For small people far from the camera (small y-coordinates), the quadratic coefficient has larger magnitude, yielding a sharper kernel. In contrast, for large people close to the camera (large y-coordinates), the coefficient has smaller magnitude, yielding a flatter kernel.

A possible explanation for using flatter kernels closer to the camera is that flat structures are easier to predict through the max-pooling layer (as discussed above), and the kernels are less likely to overlap since close people are more spread out in the image space. On the opposite, far people are more likely to have overlapping kernels, and thus a sharper kernel is used so that the densities in overlapping regions does not get too large, compared to the peak value on the person. For the overhead-view images or side-on view , there was no significant correlation between the kernel shape and its location, mainly because the object density and inter-object distance do not change significantly within the image. Finally, for ShTech B, there was a small positive correlation between the coefficient magnitude and the y-coordinate people closer to the camera (large y-coordinates) had slightly sharper kernels (larger coefficient magnitude), compared to people further away. Perhaps this is a side-effect caused by the large variations in camera orientations and heights for images in ShTech B (compared to the other datasets that are more homogenous), which also explains the small effect size (r=0.099). In summary, we found that the kernel-shape changes within the image, baon properties of the scene, and thus KDMG is able to optimize the kernel shape to match the scene and the particular density estimation network

5. SOFTWARE AND HARDWARE SPECIFICATIONS

Hardware	: Minimum Requirement.
Disk Space	: 32 GB or more,10 GB or more for Foundation Edition.
Processor	: 1.4 GHz 64 bit Memory 512 MB.
Display	: (800 × 600) Capable video adapter and monitor.

SOFTWARE SPECIFICATION

Backend Technologies

- Python
- NumPy
- PIL
- Sci-kit Image
- Sci-kit Learn
- Google Collab

6. MODULES

The Image Processing Pipeline for People Counting project consists of several modular components, each fulfilling specific functionalities to achieve the overall objective of detecting and counting people in images. These modules are organized to streamline the image processing workflow and ensure modularity, maintainability, and extensibility of the system.

1. Main Script (`image_processing_pipeline.py`):

The main script serves as the entry point for the image processing pipeline. It orchestrates the sequential execution of image loading, processing, and counting tasks. This module is responsible for importing necessary dependencies, configuring parameters, and invoking appropriate functions from helper modules to process each image. Additionally, it aggregates the results of people counting across multiple images and presents the final count to the user.

2. Helper Functions (`helper_functions.py`):

The helper functions module encapsulates reusable image processing functionalities required throughout the pipeline. It contains implementations for tasks such as cropping, resizing, gamma correction, histogram equalization, thresholding, dilation, and erosion. Each function is designed to perform a specific image processing operation efficiently and can be easily integrated into the main script or other modules as needed. This modular approach promotes code reuse and simplifies maintenance by isolating individual processing tasks.

3. Configuration File (`config.py`):

The configuration file module centralizes the management of configurable parameters used by the image processing pipeline. It defines variables such as file paths, cropping coordinates, and processing parameters, allowing users to customize the behavior of the pipeline without modifying the source code. By separating configuration details from the main logic, this module enhances flexibility and ease of adaptation to different datasets and processing requirements.

4. Image Files (`/frames` Directory):

The image files module comprises the input images used for processing by the pipeline. These images are stored in a designated directory structure, typically organized into subdirectories or named sequentially for easy retrieval and batch processing. The pipeline iterates over these images, loading each one into memory for subsequent processing steps.

5. Dependencies:

The dependencies module encompasses external libraries and packages required for the proper functioning of the image processing pipeline. These dependencies include OpenCV for image processing tasks, Matplotlib for visualization, and NumPy for numerical computing. By managing dependencies separately, the pipeline ensures compatibility with different environments and facilitates installation and version management using package managers such as pip or conda

6. Documentation:

The documentation module contains comprehensive documentation and instructions for using the image processing pipeline. It includes descriptions of modules, functionalities, usage guidelines, and examples to assist users in understanding and utilizing the system effectively. This module serves as a reference resource for developers, users, and maintainers, facilitating collaboration, troubleshooting, and knowledge transfer within the project ecosystem.

7. Tests:

The tests module encompasses unit tests and integration tests designed to validate the correctness and robustness of the image processing pipeline. These tests evaluate individual components, functions, and workflows to identify bugs, errors, or performance issues. By implementing a systematic testing strategy, the pipeline ensures reliability, accuracy, and consistency in its operation across different scenarios and input conditions.

7. SYSTEM DESIGN

7.1 SYSTEM ARCHITECTURE

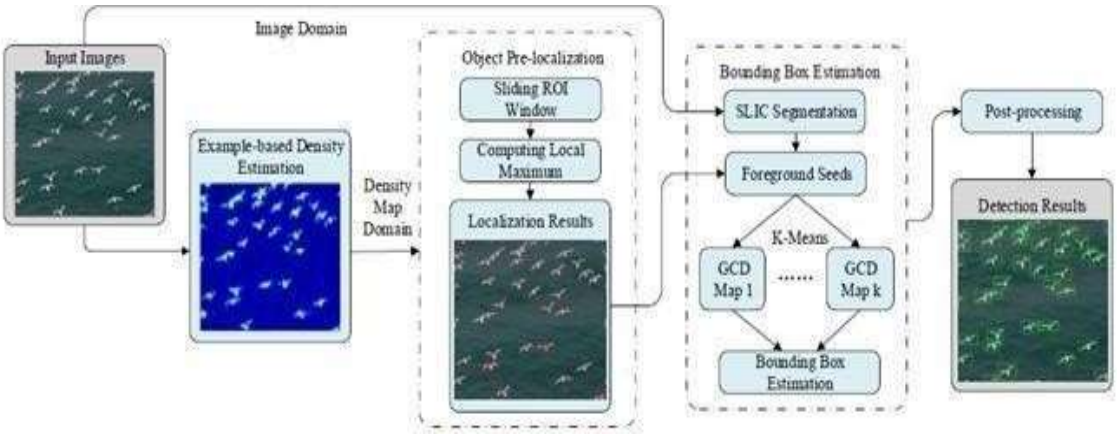


Fig 7.1: Block Diagram

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture. There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc. For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

ARCHITECTURAL DESIGN

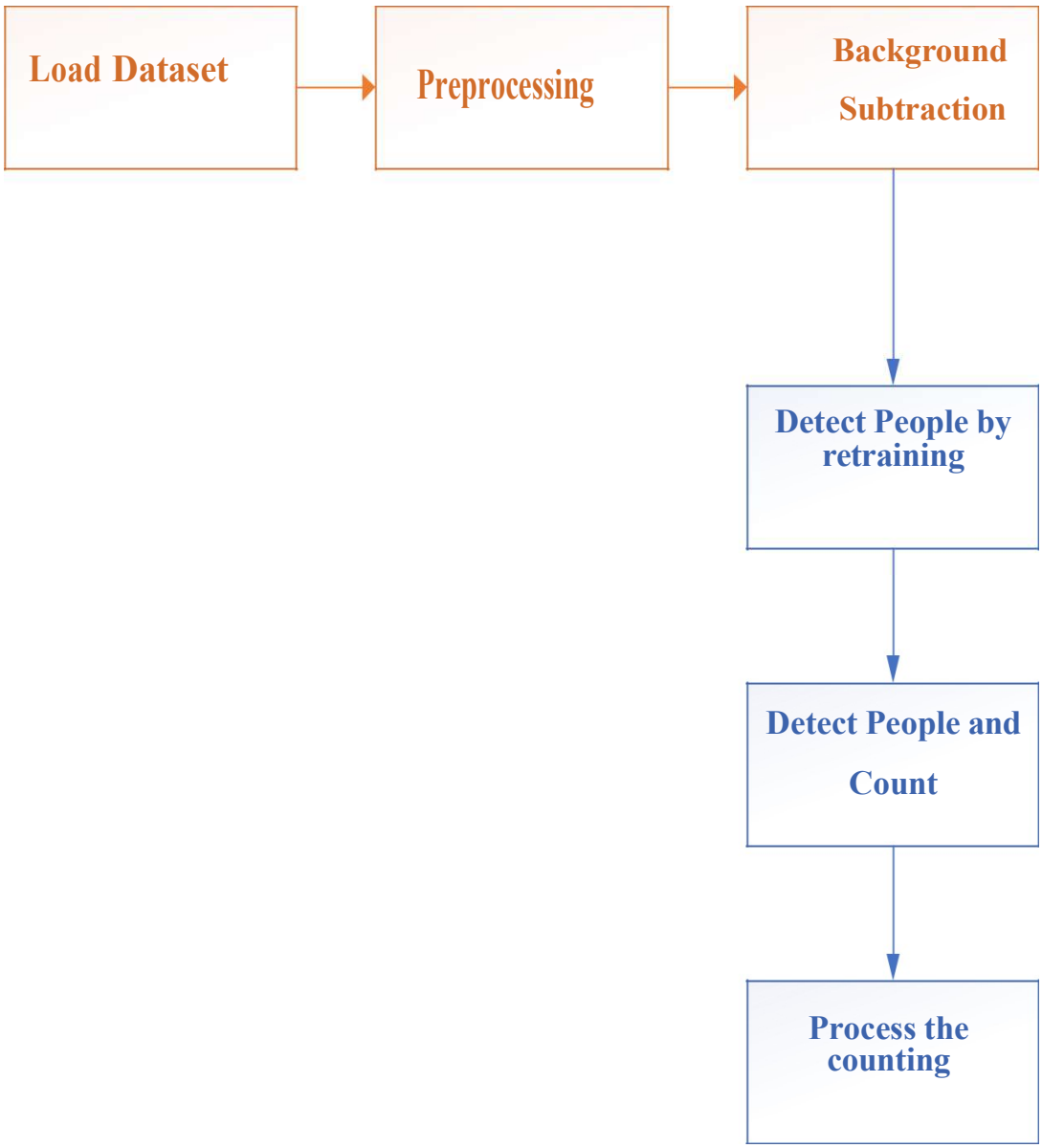


Fig 7.1.1: System Architecture

COUNTING BY DETECTION

MONOLITHIC DETECTION

It trains the classifier using the full-body appearance that's available in the training images using typical features such as Haar wavelets, gradient-based features such as a histogram of oriented gradient (HOG), etc. Learning approaches such as SVMs, random forests have been used that employ a sliding window approach. But these are limited to sparse crowds. To deal with dense crowds, part-based detection is often more useful. CrowdNet is a combination of deep and shallow, fully convolutional neural networks. This feature helps in capturing both the low-level and high-level features. The dataset is augmented to learn scale-invariant representations. The deep network is similar to the well-known VGG-16 network. It captures the high-level semantics needed for crowd counting and returns the density maps . To train a general deep learning model to explore dynamic patterns of density maps, it is important to create available datasets. The property of samples in training data should contain different sizes, shapes, and time intervals, and they should also provide smooth and continuous dynamic information. As a visualization system, we want to balance the conciseness and computational capabilities and to introduce as few interactions as possible. The dynamic change may be difficult to be identified, particularly when there are slight changes and noises. We enhance the visual effect to improve our system by using blending, field representation, and sampling. The next parts introduce how to apply these technologies to implement a general framework. We describe a procedure for initializing the structure of a mixture model and learning all parameters. Parameter learning is done by constructing a latent SVM training problem. We train the latent SVM using the coordinate descent approach described in together with the data-mining and gradient descent algorithms that work with a cache of feature vectors. The desired output of an object detection system is not entirely clear. The goal in the PASCAL challenge is to predict the bounding boxes of objects. In our previous work we reported bounding boxes derived from root filter locations.

Yet detection with one of our models localizes each part filter in addition to the root filter. Furthermore, part filters are localized with greater spatial precision than root filters. It is clear that our original approach discards potentially valuable information gained from using a multiscale deformable part model.

PART BASED DETECTION

Rather than taking the whole human body, this technique considers a part, say head or shoulders and applies a classifier to it. Head solely isn't sufficient in estimating the presence of a person reliably, therefore head + shoulder is the preferred combination in this technique. A new dataset of images is used comprising of 1198 images with 330,000 annotations to train the model. A Multi-Column CNN architecture maps the image to its crowd density map. This model utilizes filters with various receptive fields. The features learned by each column CNN are adaptive to variations in people/head size due to perspective effect or image resolution. Here, the density map is computed accurately based on geometry-adaptive kernels. The crowd density variations are taken into consideration to improve the accuracy and localization of the predicted crowd count. It relays patches from a grid within a crowd scene to independent CNN regressors on a switch classifier. A particular regressor is trained on a crowd scene patch if the performance of the regressor on the patch is the best. A switch classifier is trained alternately with the training of multiple CNN regressors to correctly relay a patch to a particular regressor. Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

SHAPE MATCHING

Ellipses are considered to draw boundaries around humans, and then a stochastic process is used to estimate the number and shape configuration. The Nanonets API allows you to build Object Detection models with ease. You 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. Several techniques have been used to come up with the right solution to the above question. Initially, computer scientists developed basic machine learning and computer vision algorithms like detection, regression, and

density-based approaches to predict crowd density and density maps. Nonetheless, these methods are also bound with various challenges such as variations in scale and perspective, occlusions, non-uniform density, etc. Later, when Convolutional Neural Networks proved its capability in various computer vision tasks by overcoming these failures, researchers shifted their attention to it, in order to exploit its features in deriving the algorithms. When the crowd has loads of people stacked up at one place, then it's termed as the dense crowd, and when the people are sparsely placed, it's a sparse crowd. The methods and techniques which we would be exploring a deal with both dense and sparse crowd. Sparse crowd counting is relatively easy in comparison to Dense crowd counting; hence the algorithms need to work harder for a dense crowd. 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. A majority of the previous approaches ignored the spatial information persisting in the images. However, this approach focuses on the density by learning the mapping between local features and object density maps, thereby incorporating spatial information in the process. This avoids learning each individual separately and therefore tracks a group of individuals at a time. The mapping described could be linear or nonlinear.

7.2 USE CASE DIAGRAM

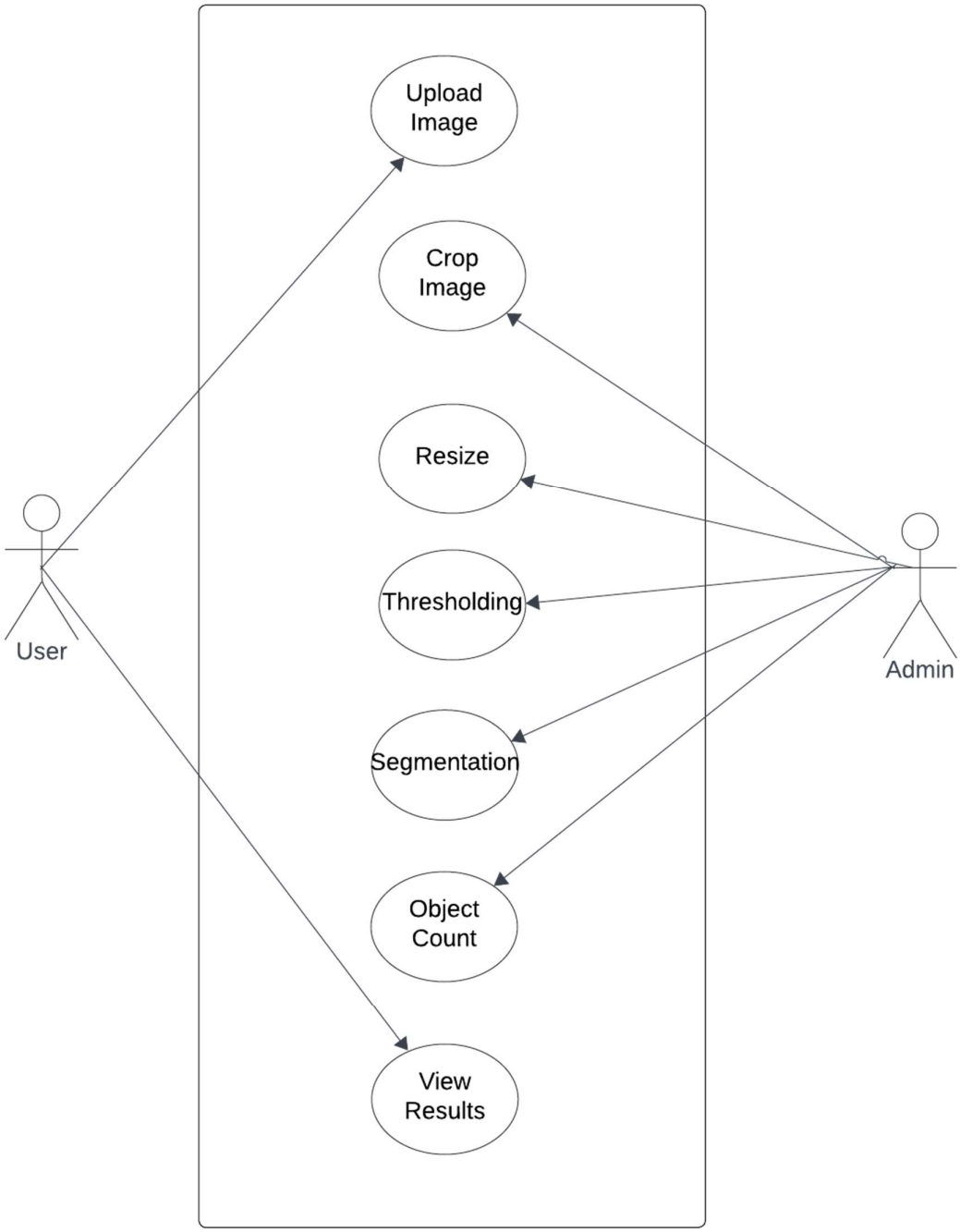


Fig 7.2 : Use Case Diagram

7.3 ACTIVITY DIAGRAM

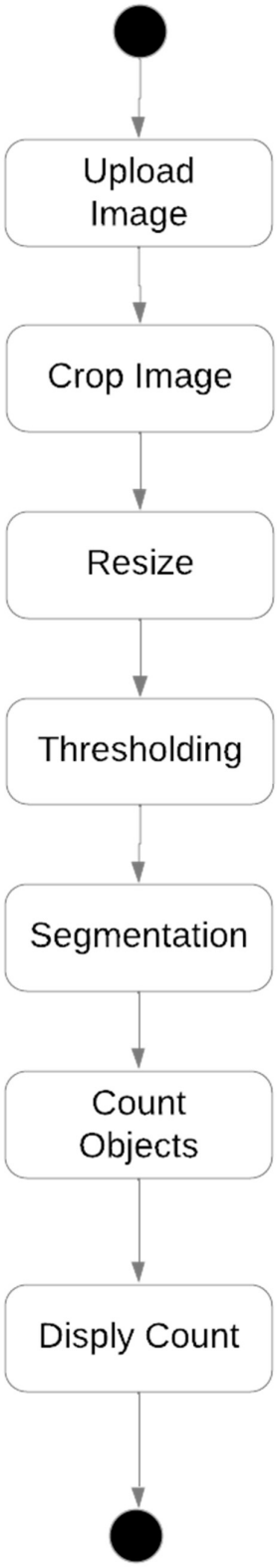


Fig 7.3 : Activity Diagram

7.4 SEQUENCE DIAGRAM

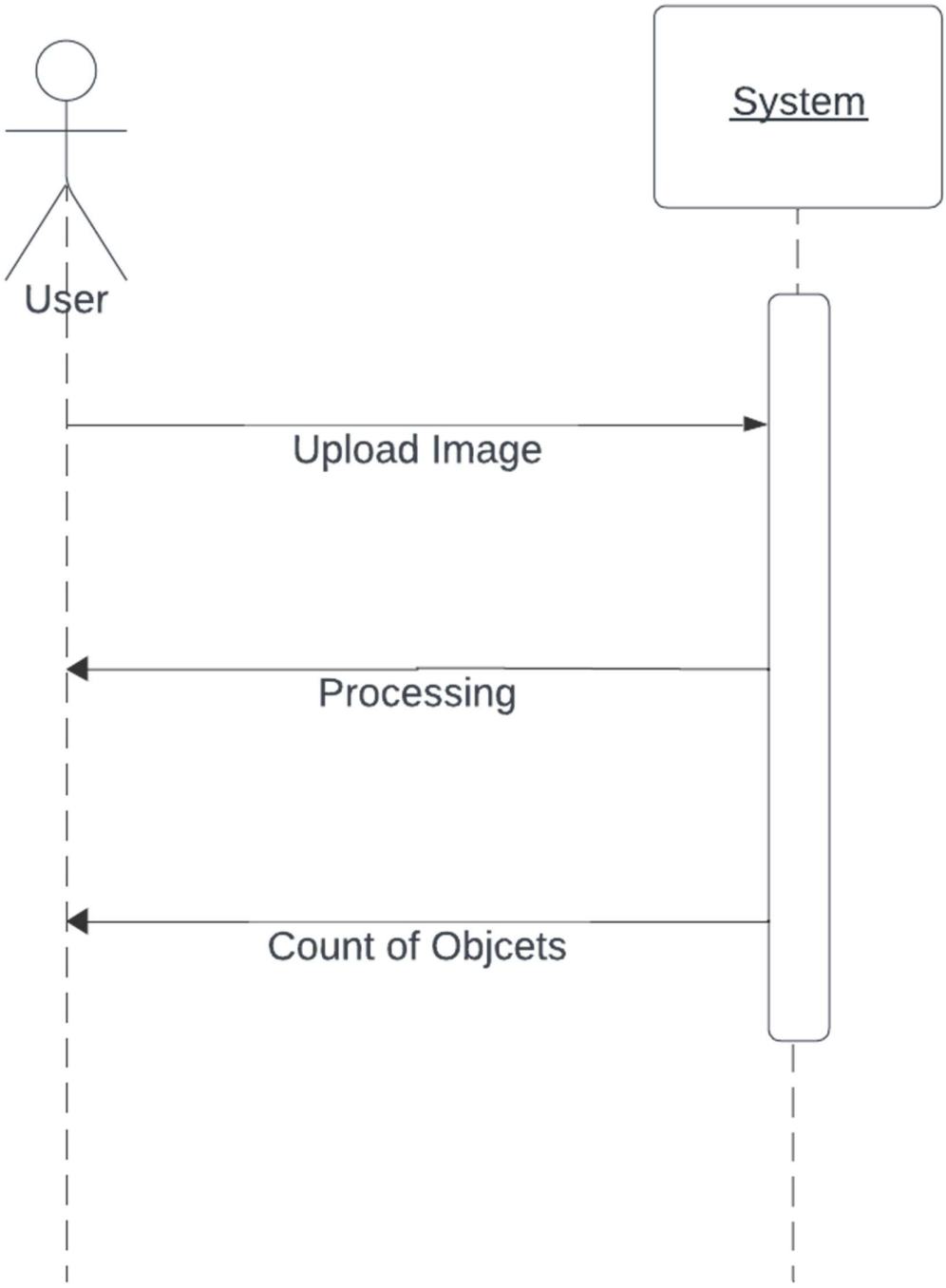


Fig 7.4 : Sequence Diagram

7.5 CLASS DIAGRAM

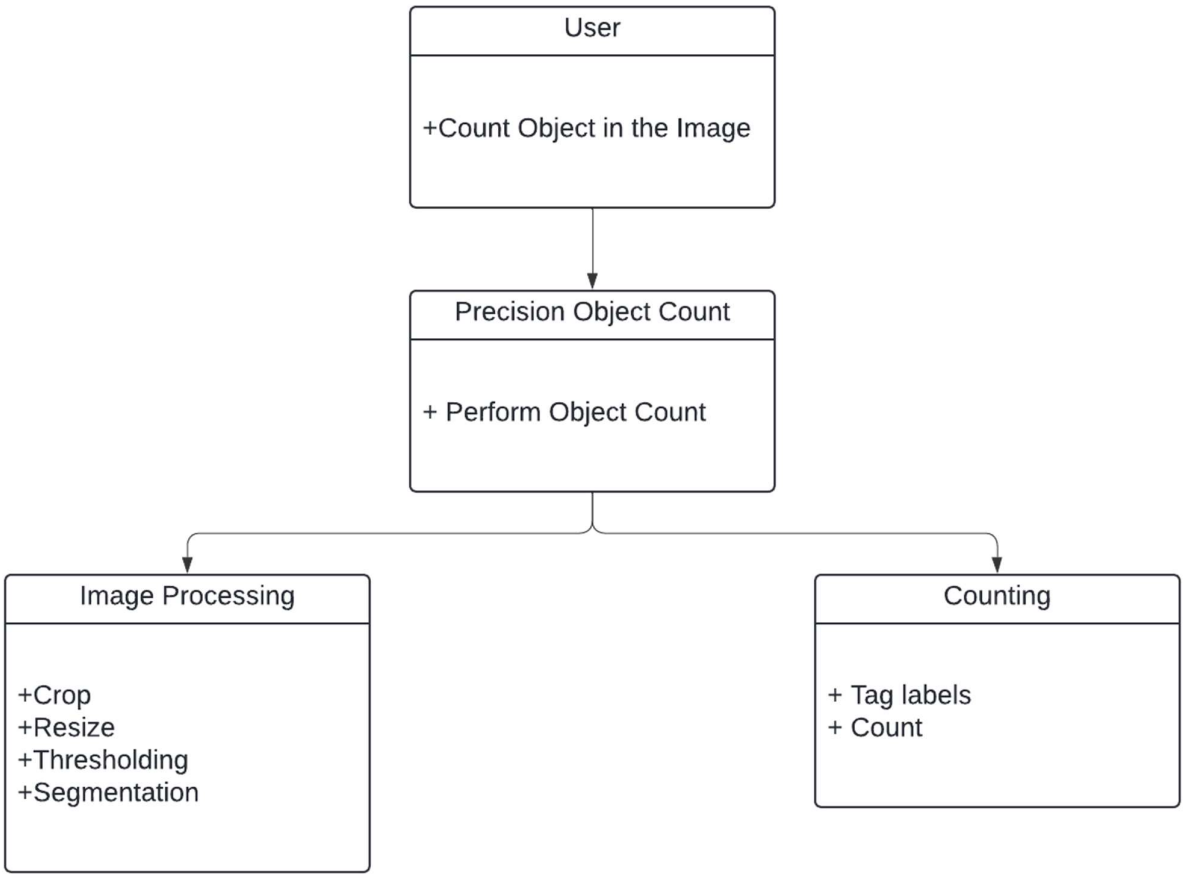


Fig 7.5 : Class Diagram

8. ALGORITHMS

CROPPING ALGORITHM

The cropping algorithm is a fundamental image processing technique used to extract a specific region of interest (ROI) from an image. It involves selecting a subset of pixels within the image based on predefined coordinates or boundaries, effectively removing irrelevant areas and focusing on the desired content. The algorithm typically requires input parameters specifying the starting and ending coordinates of the region to be cropped, along with the dimensions of the resulting cropped image.

During the cropping process, the algorithm iterates over the pixels within the specified region and copies them to a new image buffer, discarding pixels outside the defined boundaries. This operation can be performed efficiently using nested loops or array slicing operations, depending on the implementation. Cropping is commonly used in various image processing tasks, including object detection, image segmentation, and feature extraction. It allows practitioners to isolate objects or regions of interest within an image, facilitating subsequent analysis or processing steps. Additionally, cropping can help improve computational efficiency by reducing the size of images and focusing computational resources on relevant areas, particularly in applications involving large datasets or real-time processing requirements. Overall, the cropping algorithm serves as a foundational tool for extracting meaningful information from images and enhancing the effectiveness of downstream processing tasks.

GAMMA CORRECTION

Gamma correction is a non-linear image processing algorithm used to adjust the brightness and contrast of an image by modifying the relationship between the intensity values of its pixels. It operates by applying a power-law transformation to the pixel values, where each pixel's intensity is raised to a certain exponent called the gamma value. The gamma value typically ranges from 0.1 to 3.0, with 1.0 representing no change. By adjusting the gamma value, gamma correction can either enhance or diminish the contrast of an image.

In practice, gamma correction is often used to compensate for non-linearities in display devices or imaging sensors. Many display technologies, such as cathode ray tube (CRT) monitors, have non-linear response curves, meaning that the relationship between the input signal and the displayed brightness is not linear. Gamma correction helps correct for these non-linearities, ensuring that the displayed image appears visually consistent and accurate. Moreover, gamma correction can

also be used creatively to enhance the visual appearance of images by emphasizing details in darker or lighter regions, improving overall image quality, and making it more aesthetically pleasing. Overall, gamma correction is a versatile algorithm widely used in image processing and computer graphics to manipulate the tonal distribution of images and achieve desired visual effects.

ADAPTIVE HISTOGRAM EQUALIZATION

Adaptive Histogram Equalization (AHE) is an advanced image enhancement technique used to improve the contrast and visibility of details in images by redistributing pixel intensities locally. Unlike traditional histogram equalization, which operates globally on the entire image, AHE divides the image into smaller regions or tiles and applies histogram equalization independently to each region. This adaptive approach allows AHE to enhance local contrast while preserving the overall image structure.

During the AHE process, the histogram of pixel intensities within each tile is computed, and a transformation function is applied to redistribute the pixel values based on the local histogram. Regions with low contrast or uneven illumination are effectively stretched to maximize the dynamic range, while regions with high contrast are compressed to avoid over-enhancement. The result is an image with improved clarity and detail, particularly in areas with varying illumination or texture.

A key advantage of AHE is its ability to adapt to local image characteristics, making it suitable for a wide range of applications, including medical imaging, remote sensing, and surveillance. However, AHE may also amplify noise in regions with low texture or introduce artificial boundaries between adjacent tiles. To address these limitations, variants of AHE, such as Contrast Limited Adaptive Histogram Equalization (CLAHE), impose constraints on the amount of enhancement applied to each tile, ensuring more natural-looking results while still preserving local contrast enhancements. Overall, Adaptive Histogram Equalization is a powerful tool for enhancing image quality and revealing fine details in a variety of applications.

OTSU’S THRESHOLDING

Otsu's Thresholding algorithm is a widely used method for automatically determining the optimal threshold value to separate the foreground and background of an image. It operates by analyzing the histogram of pixel intensities in the input image and finding the threshold value that minimizes the intra-class variance between the foreground and background classes.

The algorithm iterates over all possible threshold values and calculates the variance within each class (foreground and background) based on the pixel intensities. The threshold value that yields the lowest intra-class variance is selected as the optimal threshold for binarizing the image. This threshold effectively separates the pixels into two classes: those with intensities below the threshold (considered background) and those with intensities above the threshold (considered foreground).

Otsu's Thresholding algorithm is particularly effective for images with bimodal intensity distributions, where there are distinct peaks corresponding to the foreground and background. By automatically determining the threshold value, Otsu's method eliminates the need for manual threshold selection and adapts to variations in image content and lighting conditions.
This algorithm finds applications in various image processing tasks such as image segmentation, object detection, and edge detection. Its simplicity, effectiveness, and ability to operate without prior knowledge of the image content make it a valuable tool in many computer vision applications. However, it may not perform optimally in images with non-unimodal intensity distributions or uneven illumination.

DILATION & EROSION

Dilation and erosion are two fundamental morphological operations used in image processing to modify the shape, size, and structure of objects within binary images. Dilation involves expanding the boundaries of foreground regions (pixels with value 1) by adding pixels to their edges. This process is achieved by convolving the binary image with a structuring element, such as a kernel or mask, and setting the value of each pixel in the output image to 1 if any of the corresponding pixels in the input image overlapped with the structuring element. Dilation is often used to fill in gaps, connect disjointed regions, and increase the size of objects in binary images.
On the other hand, erosion shrinks the boundaries of foreground regions by removing pixels from their edges. It operates similarly to dilation but instead sets the value of each pixel in the output image to 1 only if all the corresponding pixels in the input image overlapped with the structuring element. Erosion is useful for removing small protrusions, smoothing object boundaries, and separating touching objects in binary images.

These two operations are often used together in sequence, with dilation followed by erosion (or vice versa), to achieve specific effects such as noise reduction, feature extraction, and morphological transformations. By iteratively applying dilation and erosion with different structuring elements and parameters, various complex operations, such as opening, closing, and morphological gradient, can be performed to manipulate and enhance binary images for different applications in image analysis and computer vision.

CONTOUR-BASED LABELLING

Contour-based labeling is an image processing technique used to identify and count objects within an image based on their contours. Contours are continuous curves that represent the boundaries of objects in an image, and contour-based labeling involves detecting these contours and analyzing their properties to distinguish individual objects from the background.

The algorithm begins by segmenting the image into foreground and background regions using techniques such as thresholding or edge detection. Next, contours are extracted from the binary image using contour tracing algorithms like the Moore-Neighbor tracing algorithm or the Suzuki algorithm. These algorithms traverse the boundary pixels of objects, generating a list of coordinates that define the contours.

Once the contours are extracted, the algorithm analyzes their properties, such as area, perimeter, and shape, to differentiate objects from noise or background artifacts. Objects with specific characteristics, such as size or aspect ratio, can be filtered out or counted based on predefined criteria.

Contour-based labeling is particularly useful for counting objects in scenarios where they are distinct and well-separated from each other. It finds applications in various domains, including object detection, medical imaging, industrial inspection, and biological analysis. However, it may encounter challenges in scenarios with overlapping or touching objects, irregular shapes, or complex backgrounds, which can lead to inaccuracies in object counting and labeling. Despite these limitations, contour-based labeling remains a powerful tool for automating object analysis and quantification in digital images.

9. IMPLEMENTATION

SAMPLE CODE

```
from google.colab import drive
drive.mount('/content/drive')

import cv2
import matplotlib.pyplot as plt
import numpy as np

# Function to display images in a grid layout
def plotting_grid(images, titles, rows, cols, figsize):
    fig, axes = plt.subplots(rows, cols, figsize=(figsize, figsize))
    for i, ax in enumerate(axes.flat):
        ax.imshow(images[i], cmap='gray', vmin=0, vmax=255, interpolation='none')
        ax.set_title(titles[i])
        ax.set_xticks([]), ax.set_yticks([])

# Function to perform cropping
def crop(image, start_y, end_y, start_x, end_x):
    img_cropped = image[start_y:end_y, start_x:end_x]
    return img_cropped

# Function to perform resizing
def resize(image, width, height):
    img_resized = cv2.resize(image, (width, height))
    return img_resized

# Function for gamma correction
def gamma_correction(image, y):
    gamma_correct = np.array(255 * (image / 255) ** y, dtype='uint8')
    return gamma_correct

# Function for adaptive histogram equalization
def adaptive_histogram_equalization(image):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
```

```
    img_equalized = clahe.apply(image)
    return img_equalized

# Function for Otsu's thresholding
def otsu_threshold(image):
    _, img_thresh = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    return img_thresh

# Function for dilation and erosion
def dilation_erosion(image):
    kernel = np.ones((15, 15), np.uint8)
    img_dilation = cv2.dilate(image, kernel, iterations=1)
    img_erode = cv2.erode(img_dilation, kernel, iterations=1)
    img_erode = cv2.medianBlur(img_erode, 7)
    return img_erode

# Function for labeling using contours
def labeling(image):
    contours, _ = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    labeled_img = cv2.drawContours(np.zeros_like(image), contours, -1, (255), thickness=cv2.FILLED)
    count = len(contours)
    plt.title('People counted in this image: ' + str(count))
    plt.imshow(labeled_img, cmap='gray')
    plt.show()
    return count

# Load images
image_paths = [
'/content/drive/MyDrive/object count in  images/code/frames/frames/seq_000001.jpg']

# Process images
total_people_count = 0
for i, img_path in enumerate(image_paths):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    # Check if image is successfully loaded
```

```
    if img is None:
        print(f"Error: Unable to load image at path: {img_path}")
        continue

    # Crop, Resize, Gamma Correction
    cropped_images, cropped_titles = [], []
    for j, (start_y, end_y, start_x, end_x) in enumerate([(100, 380, 380, 600), (20, 220, 100, 300), (200, 500, 0, 300)]):
        # Crop
        cropped_img = crop(img, start_y, end_y, start_x, end_x)
        cropped_images.append(cropped_img)
        cropped_titles.append(f'Image {i + 1}-{chr(ord("A") + j)} (Cropped)')

    # Display original image and cropped images in a grid
    all_images = [img] + cropped_images
    all_titles = [f'Image {i + 1}'] + cropped_titles
    plotting_grid(all_images, all_titles, 1, len(all_images), 18)

    # Resize, Gamma Correction, Adaptive Histogram Equalization, Otsu's Thresholding, Dilation and Erosion
    processed_images, processed_titles = [], []
    for j, cropped_img in enumerate(cropped_images):
        # Resize
        multiplier = 2
        resized_img = resize(cropped_img, cropped_img.shape[1] * multiplier, cropped_img.shape[0] * multiplier)

        # Gamma correction
        gamma_value = [1.2, 0.5, 2.5][j]
        gamma_corrected_img = gamma_correction(resized_img, gamma_value)

        # Apply Gaussian blur for smoothing
        blurred_img = cv2.GaussianBlur(gamma_corrected_img, (5, 5), 0)

        # Adaptive histogram equalization
        equalized_img = adaptive_histogram_equalization(blurred_img)
```

```
        # Adaptive thresholding
        thresholded_img = otsu_threshold(equalized_img)

        # Dilation and Erosion
        dlt_er_img = dilation_erosion(thresholded_img)

        processed_images.extend([resized_img, gamma_corrected_img, blurred_img, equalized_img,
thresholded_img, dlt_er_img])
        processed_titles.extend([
            f'Resized Image {i + 1}-{chr(ord("A") + j)}',
            f'Gamma Image {i + 1}-{chr(ord("A") + j)}',
            f'Blurred Image {i + 1}-{chr(ord("A") + j)}',
            f'Equalized Image {i + 1}-{chr(ord("A") + j)}',
            f'Thresholded Image {i + 1}-{chr(ord("A") + j)}',
            f'Dilatation and Erosion Image {i + 1}-{chr(ord("A") + j)}'
        ])
    # Display processed images in a grid
    plotting_grid(processed_images, processed_titles, len(cropped_images), 6, 18)

    # Labeling using contours
    total_people_count += labeling(dlt_er_img)
# Display total count
print(f'Total people counted across all images: {total_people_count}')
```

10. OUTPUT SCREENSHOTS

As a result of this system we are going to say about the people counting in this project and in this we have used deep learning to run this code and in this we have used convolutional neural network algorithm is used for counting purpose and for remaining modules we have written the code in the code in the program.

Fig 10.1 Input Screenshot 1

The image tells about the input of the project from this we will calculate the density of the persons in the image and it will give most accurate than so many software in this it is the output of our project by this we will get the output density of an image.

Fig 10.2: Input Screenshot 2

Fig 10.1.1 Cropped Gray Scale Image of the Input

This screenshot explains about the grayscale image which it would be converted to count the people in the image of the screenshot 4.1 from which it converted into machine untestable Language and then it will calculate the density of the people and it will give the result by converting the grayscale we use coding to convert the normal image into a grayscale image by this we can convert the image into grayscale then it will count the density of the people and estimate it.

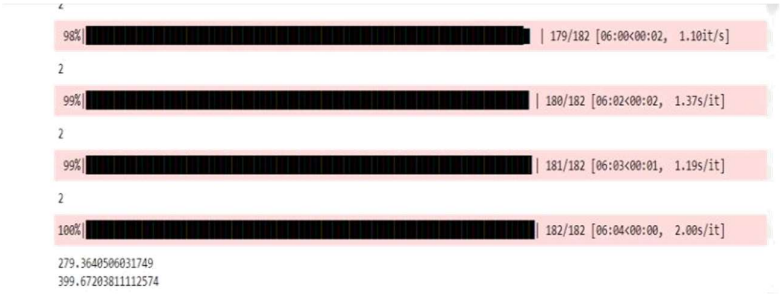
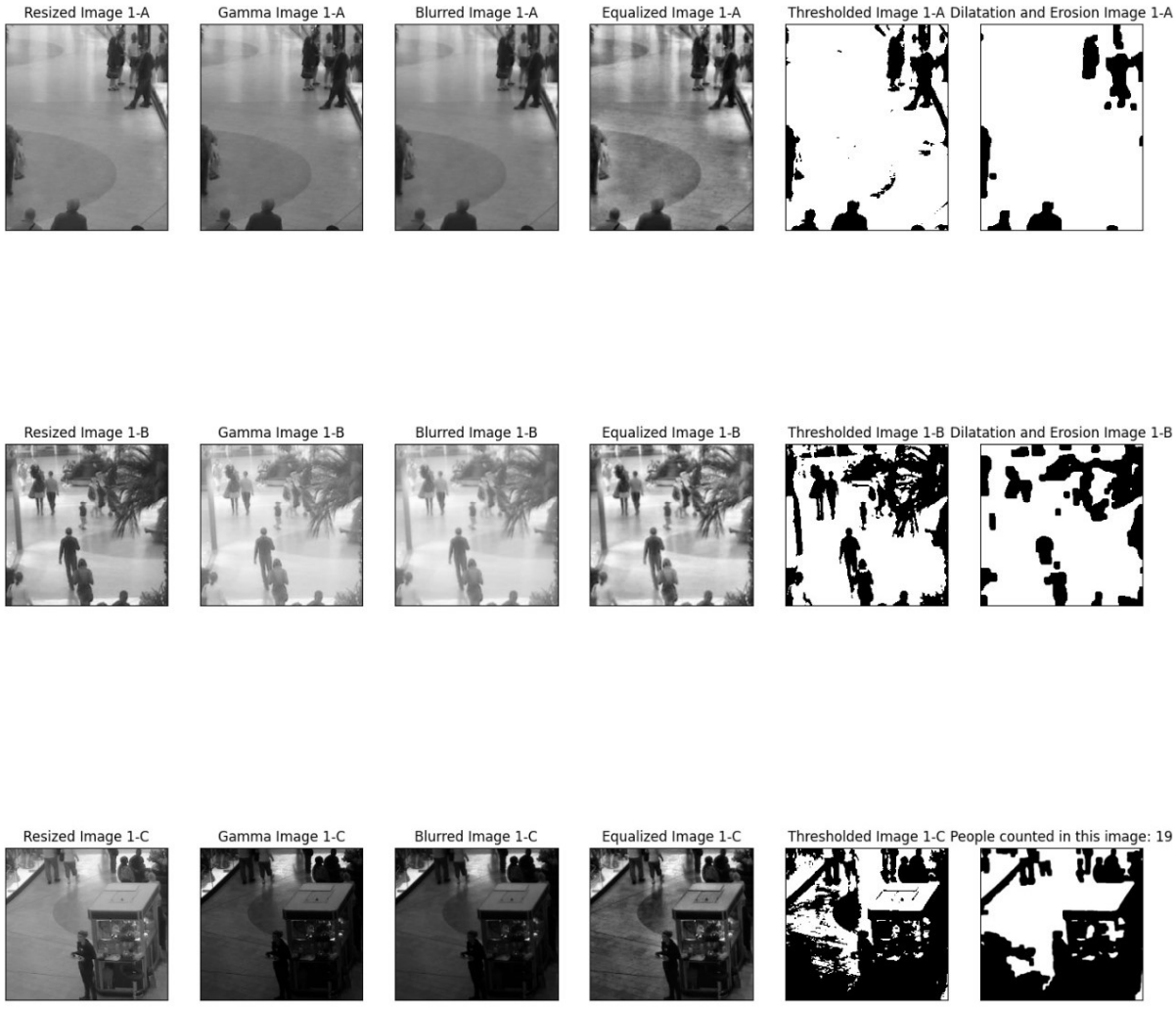


Fig 10.1.2 Density Calculating of the Image

This images shows about the density calculation and the mean squared error and in which how much accurate the result it is giving the output and it will load in percentages and it will calculate. The errors and density of each input it will calculate and it will show the output density as density equals to and it will show the output in this it will give much accurate than the other methods and in this convolutional neural network algorithm is used to calculate the density of every images.

This is the output screen of the project in this it will show the density of an image And then it will calculate the errors.

Fig 10.2.1 Output (Result) 1



Total people counted across all images: 19

Fig 10.2.2 Output (Result) 2

11. CONCLUSION

We propose an accurate small object detection method by exploring from both the image and its estimated density map. Our method takes advantage of the object spatial distribution information in density map but avoids its drawback of obscure object boundary. Although our method is trained by dotted annotation datasets, the estimated bounding box fits the object accurately due to the sufficient boundary information provided by saliency map. The proposed focus prior map can be used as the focus feature for image analysis or used as an effective prior to improve the performance of salient object detection.

12. FUTURE SCOPE

Deep learning models have often achieved increasing success due to the availability of massive datasets and expanding model depth and parameterization. However, in practice factors like memory and computational time during training and testing are important factors to consider when choosing a model from a large bank of models. Training time becomes an important consideration particularly when the performance gain is not commensurate with increased training time as shown in our experiments. Test time memory and computational load are important to deploy models on specialized embedded devices, for example, in AR applications. From an overall efficiency viewpoint, we feel less attention has been paid to smaller and more memory, time efficient models for real-time applications such as road scene understanding and AR. This was the primary motivation behind the proposal of SegNet, which is significantly smaller and faster than other competing architectures, but which we have shown to be efficient for tasks such as road scene understanding.

13. REFERENCES

[1] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun, “Saliency optimization from robust background detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR). IEEE, 2014, pp. 2814–2821.

[2] Na Tong, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang, “Salient object detection via bootstrap learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR). IEEE, 2015, pp. 1884–1892.

[3] Na Tong, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang, “Salient object detection via bootstrap learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR). IEEE, 2015, pp. 1884–1892.

[4] Tobias Franke, Paul Lukowicz, and Ulf Blanke, “Smart crowds in smart cities: real life, city scale deployments of a smartphone based participatory crowd management platform,” Journal of Internet Services and Applications, vol. 6, no. 1, pp. 27,2015.

[5] Y. Wang, Y. Zou, J. Chen, X. Huang, C. Cai, "Example-based visual object counting with a sparsity constraint," in IEEE International Conference on Multimedia and Expo, 2016.

[6] Kang, Z. Ma, and A. B. Chan, “Beyond counting: com- parisons of density maps for crowd analysis tasks-counting, detection, and tracking,” *IEEE Trans. CSVT*, 2018.

[7] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, and H. Wu, “Ad- crowdnet: An attention- injective deformable convolutional network for crowd understanding,” in *CVPR*, June 2019.

[8] C. Liu, X. Weng, and Y. Mu, “Recurrent attentive zooming for joint crowd counting and precise localization,” in *CVPR*, 2019, pp. 1217–1226.

[9] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Advances in Neural Information Processing Systems, vol. 28, 2015.

[10] Zhang, Y. and Tian, Y. "Crowd Counting and Detection: A Survey." IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 6, 2018, pp. 1475-1489.

[11] Goodfellow, Ian, et al. "Generative Adversarial Nets." Advances in Neural Information Processing Systems, vol. 27, 2014.

[12] Zeng, Xinyu, et al. "End-to-End Crowd Counting via Joint Learning Local and Global Counting." IEEE Transactions on Image Processing, vol. 29, 2020, pp. 2076-2087.

[13] Viola, Paul, and Jones, Michael. "Rapid Object Detection using a Boosted Cascade of Simple Features." IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[14] Wang, Yanyong, et al. "Head Detection in Crowd Scenes via Spatially-Constrained Feature Grouping." IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 9, 2020, pp. 3023-3036.

[15] Lipton, Zachary C., et al. "Detecting and Counting People: Learning from Synthetic Training Data." arXiv preprint arXiv:1903.00237, 2019.

[16] Dalal, Navneet, and Triggs, Bill. "Histograms of Oriented Gradients for Human Detection." IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[17] Chollet, François et al. "Keras." GitHub repository, https://github.com/keras-team/keras, 2015.

[18] Abadi, Martín et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv preprint arXiv:1603.04467, 2016.