Appendix I: Data Analytic Application – Crowd Counting

Contents

Abstract 2
1. Introduction 3
1.1. Motivation 3
1.2. Related Works 3
1.3. Contributions 3
2. Assumptions 3
3. Details of the solution 4
4. Results 13
5. Conclusions 19
5.1. Achievements 19
5.2. Future Works 19
References 19

Abstract

This appendix considers the role of automatic estimation of crowd size in very crowded scenarios. A variation of well-established techniques is proposed which is able to estimate crowd size in a distorted video where objects close to the camera appears larger. The technique is based on a machine learning method which can automatically decide the size of people's heads in different parts of the image. Then different strategies are applied depending on the head sizes. Image processing followed by circle finding is used when heads are of medium size. Upper body detection is used when heads are large. If the heads are small, the image is resized and the method is run recursively on the image. An error of around 20% is achieved in very crowded scenarios.

1. Introduction

1.1. Motivation

The management and control of crowds is a crucial problem for human life and safety. When an accident happens at crowded places, many lives can be lost. Many problems involving crowds can be prevented or quickly resolved when all the aspects of crowd management and control are well organized.

1.2. Related Works

The most intuitive method is to detect all the people in the image, but these detection based methods are indispensable in practical crowd scenes because the detection is often very inaccurate due to severe occlusions and scene perspective distortions according to Idrees H., Soomro K. and Shah M.[1].

In the past, there have been a few works on crowd size estimation. The techniques used can be classified into two categories: (I) image processing based like the methods mentioned in [1], [2] and (2) machine learning based like the methods mentioned in [4] [5]. In image processing based methods, multiple image enhancement techniques like segmentation and morphological operations, are used to process the image, then people are detected at locations in the image where the surrounding pixels meet some criteria. The number of people detected is used as the number of people in the image. In machine learning based methods, instead of finding out the heads, crowd density is directly estimated by a machine learning model which needs to be trained on a large number of images.

The machine learning based method requires a large set of images labeled with number of people. In the research by A.N. Marana a, S.A. Velastin, L.F. Costa, and R.A. Lotufo[6], 300 images are used. Counting the number of people in crowded scenarios is very time consuming, and no labeled dataset of very crowded images are available online. Also only a few weeks are scheduled for this research. Thus we didn't focus on the machine learning based methods.

1.3. Contributions

An image processing based method proposed by L. C. Wanjira[2] is available online. It processes the image and finds heads of a fixed range of sizes. Depending on the original method, several modifications are made on it to form our method. The main modification is that instead of using fixed sizes, our method would first divided the image into small patches, then decide the size of heads then use that value to find heads. The image is divided because in each of the small patches, head sizes have a better uniformity. Also background removal is applied which request a video to remove background. Thus the new method works on video. The details of the two methods will be discussed in the methods section. 3 detection based methods are also tested and compared with our proposed method.

2. Assumptions

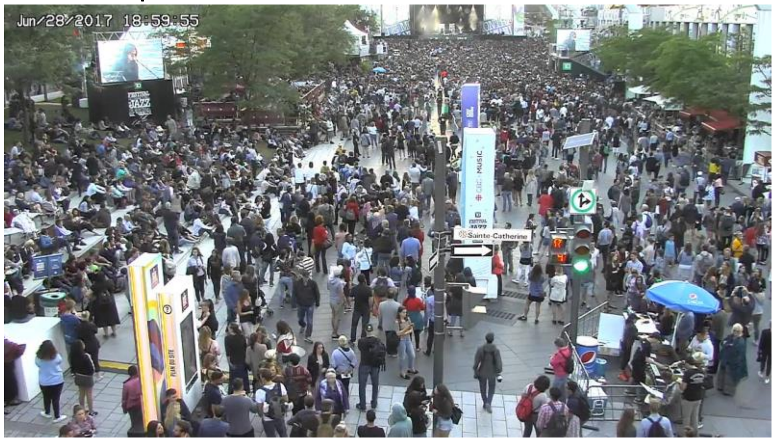

Figure 1: Image taken by camera with expected installation.

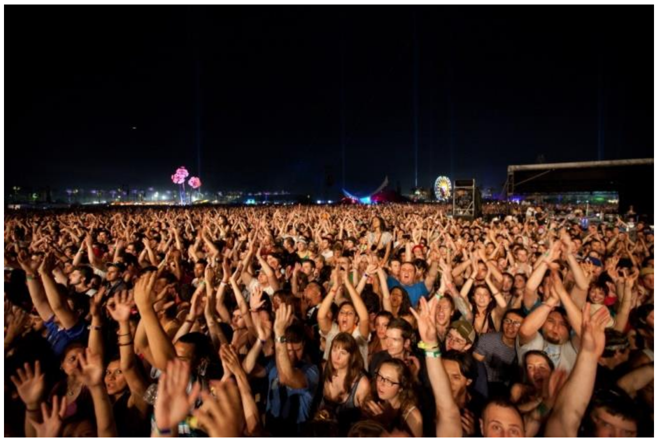
Figure 2: Image taken by camera with too high inclination.

Our proposed method works on a video of resolution around 1280x720p taken by a camera. It can also run with other resolutions, but the accuracy will be affected. The camera should be installed above people's heads and shoot down (not necessarily vertical) on people but too high inclination can cause a very low accuracy. Figure 1 illustrates the image taken by camera of expected installation while in Figure 2, the inclination is too high. Figure 3 shows an example of camera installation. If no video is available, the proposed method still works on a photo, but the accuracy will be reduced. Our method works for various scenarios: crowded/ uncrowded, day/night and different places.

3. Details of the solution

Original Method:

First the method proposed by L. C. Wanjira is tried. It applied various techniques to process the image. Then circles of fixed sizes are detected as people's heads in the image. The development of the program algorithm can be divided into four main steps according to Figure 4. Then each main step is broken into sub steps. The algorithm is implemented in Matlab.

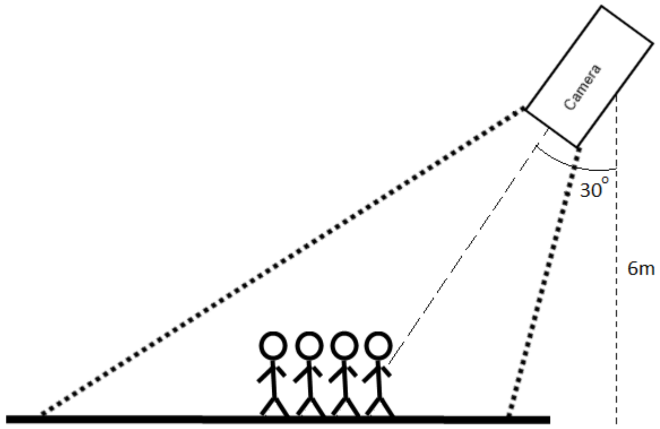

Figure 3: An example of camera installation.

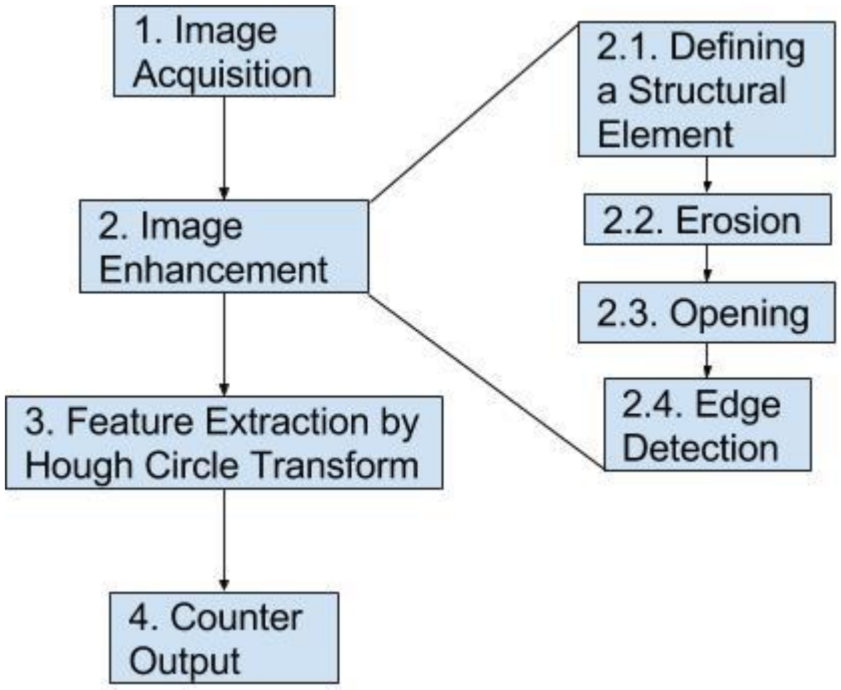

Figure 4: Diagram of the original method.

1) Image Acquisition:

In this step, the image taken from a camera at any angle is read. Because our image enhancement operations perform on grayscale images, the image is then converted into grey scale according to Formula (1), which determines each pixel's luminosity based on the red, green, and blue channels.

$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (1)$$

The number of people in this image will be estimated.

2) **Image Enhancement:** Image processing techniques are carried out in this step. They include:

2.1. Defining a Structuring Element:

A structuring element is a set of pixels with peculiar shape that acts on the pixels of the image to produce an expected result. In many image enhancement techniques, the value of a pixel is modified according to the neighborhood of that pixel. For example, erosion would set the value of that pixel to the minimum value in its neighborhood. The neighborhood is defined by the structural element. We choose a structuring element the same size and shape as the objects we want to process in the input image. In our case a circular structuring element of radius 5 is used to enable us to identify the circular shape of the heads.

2.2. Erosion[7]:

The image is eroded with the structural element defined in 2.1. In a grayscale image, every pixel is eroded and the erosion of that pixel is the minimum of the pixels in its neighborhood, with that neighborhood defined by the structuring element. The basic effect of the operator on an image is to erode away the boundaries of regions of foreground pixels (i.e. white pixels, typically) thus the boundary becomes more smooth. This removes the acute corners and emphasizes the circular form of the heads for easy identification and analyzing.

2.3. Opening[8]:

Opening of the eroded object using the same structural element closes or fills the gaps between objects. One can think of the structural element sweeping around the inside of the boundary of foreground, so that it does not extend beyond the boundary, and shaping the foreground boundary around the boundary of the element. If the foreground object is too small that the element cannot fit in it, that object is removed. In conclusion, it removes small objects from an image while preserving the shape and size of larger objects in the image. Opening is performed on the eroded image generated by Step 2.2 with the structural element described in Step 2.1.

2.4. Edge Detection:

Edge is the boundary between an object and its background. If edges of images can be identified with precision, all the objects can be identified and their area, perimeter, shape etcetera can be calculated. The Canny edge detector [9] is used to detect the edges in the image. The Canny method finds edges by looking for local maxima of the gradient of the image. The gradient is calculated using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges. That is why we choose Canny edge detector. The thresholds used here for strong and weak edges are 0.2 and 0.15.

The details of the Canny Methods are as follows:

a. Gaussian filter: Since all edge detection results are easily affected by image noise, it is essential to filter out the noise to prevent false detection caused by noise. To smooth the image, a Gaussian filter is applied to convolve with the image. This step will slightly smooth the image to reduce the effects of obvious noise on the edge detector. The equation for a Gaussian filter kernel of size (2*k*+1)×(2*k*+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i-(k+1)\right)^2+\left(j-(k+1)\right)^2}{2\sigma^2}\right); 1 \le i,j \le (2k+1) \quad (2)$$

The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases. We use this size in our program. Also, a standard deviation $\sigma = 1.4$ is used.

b. Finding the intensity gradient of the image: An edge in an image may point in a variety of directions, so the Canny Algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient (*G*) and direction(*Θ*) can be determined:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (3)$$

$$\theta = atan2({G_x}^2, {G_y}^2)$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0°, 45°, 90° and 135°). An edge direction falling in each color region will be set to a specific angle values, for instance ϑ in [0°, 22.5°] or [157.5°, 180°] maps to 0°.

c. Non-maximum suppression: Non-Maximum suppression is applied to "thin" the edge. After applying gradient calculation, the edge extracted from the gradient value is still quite blurred. Thus non-maximum suppression can help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:
   i. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
   ii. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

d. Double threshold: After applying non-maximum suppression, the edges extracted provide a more accurate representation of real edges in an image. But some of them are caused by noise. It is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values. If an edge pixel's gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel's gradient value is smaller than the high threshold value and larger than the low threshold value, it is marked as a weak edge pixel. If an edge pixel's value is smaller than the low threshold value, it will be suppressed. For all the weak edges found, if there is a strong edge located at its surrounding pixels, the weak edge will be preserved; else it will also be suppressed. The two threshold values are empirically determined.

3) Feature Extraction by Hough Circle Transform:

Features are inherent properties of data. In this context we are looking to extract heads for counting in estimation of a crowd. Hough Transform is used to identify and mark the heads in form of circles.

Hough transform is a mapping algorithm that processes data from a Cartesian coordinate space into a polar parameter space. It is most useful for finding geometric lines and shapes in binary images. The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough Transform works for this application, as people's heads in the processed images have feature boundaries which can be described by regular circles. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise. Hence it is best suited for our application of identifying heads and counting them.

A circle can be described completely with three pieces of information: the center (a, b) and the radius.

$$\begin{aligned} x &= a + R \cdot \cos\theta \\ y &= b + R \cdot \sin\theta \end{aligned} \tag{4}$$

When the ϑ varies from 0 to 360 degrees, a complete circle of radius *R* is generated. So with the Circle Hough Transform, we expect to find triplets of (*x*, *y*, *R*) that are highly probably circles in the image. We also assume that the radii of head are between 5 and 10 pixels, thus in the searching for the triple, the parameter *R* is restricted to that range. We will discuss the assumption on head size later.

The circular Hough transform method provides a way to find the triplets that are highly probably circles. This method is robust in the presence of noise, occlusion and varying illumination. Its steps are as follows:

a. Accumulator Array Computation
   An accumulator 2-D array with complex values as large as the image is created. Each pixel corresponds to a cell in the array. For all the pixels marked as edges, the pixel will vote for all pixels on the circle centered at that pixel. The radius of the circle can be any value within the range of radius we define. After the vote, the values in the corresponding cells in the accumulator array of the voted pixels increase. In the vote, the radius

information is also encoded in the phase of the array entries. The votes cast by the edge pixels contain information not only about the possible center locations but also about the radius of the circle associated with the center location.

b. Center Estimation
After the votes in the accumulator array are accumulated, the circle centers are estimated by detecting the peaks in the accumulator array.

c. Radius Estimation
The radius is estimated by simply decoding the phase information from the estimated center location in the accumulator array.

4) Counter Output:
The number of circles found in step 3 is outputted as the number of people in the image.

During the whole process, no performance issue is encountered. The program terminates immediately.

Proposed Method:

In the original method, we assume that the heads are of radii between 5 and 10 pixels, which are not true for all scenes. Even in one image the head sizes vary a lot due to distortion. Heads close to the camera can be much larger than those far from the camera. The accuracy can be improved if we know the size of heads in any part of the image. Considering this, the following modification is proposed.

Instead of using a fixed range of radius, the image is first divided into 3 by 3 small patches. For each small patch, a trained machine learning model is used obtain a range of head size in that patch. Then the same image enhancement techniques in the original method are applied to the small patch. After that, circles of the found range of size are detected as heads. The detected heads in all of the patches are counted as output. The detailed workflow is shown in Figure 5. First we prepare a machine learning model trained to find out the heads size in an image. Then a frame is extracted from the video to estimate the number of people in it. After the background and leaves in the video frame are removed, the image is divided into small patches. The machine learning model decides the head size in each of the patches. Then different strategies are applied depending on the head sizes. Image processing followed by circle finding is used when heads are of medium size. Upper body detection is used when heads are large. If the heads are small, the small patch is resized and the method is run recursively on the image.

i. Preprocessing:
Here we will discuss about how the image is acquired and some simple processing operations to reduce error including background removal and leaves removal. After those operations, the image is divided.

1) Image Acquisition:
Instead of using an image taken from camera, we will work on a frame extracted from a video. Then background removal and leaves removal are applied to the video frame. After that, the image is converted to grayscale. The crowd size in this video frame will be estimated. If no video is available, we will work on an image and Step 1.2 will be skipped.

1.1. Frame Extraction:
A frame is extracted from the video. Any frame works and here we use the tenth frame. The video enables us to do background removal in Step 1.2.

1.2. Background Removal:
This step is skipped when no video is available. The use of video enables removing objects that are not people by background removal, which first fits a statistical background model to the frames in the video, then compares the color of individual pixels in a video frame to the background model to determine whether the pixels are part of the background or the foreground. The pixels marked as background are then painted black in the image. Background removal eliminates objects which may cause false detection in the future and the accuracy can be improved significantly, as the original method makes lots of false detection where there are no people. However, some problems may be caused. Sometimes people are marked as background when they do not move for a long time. The background model used in our method is a Gaussian mixture model consisting of 3 Gaussian models and it is trained with the first 40 frames of the video. Figure 6 and Figure 7 show the effect of background removal.

1.3. Leaves Removal:

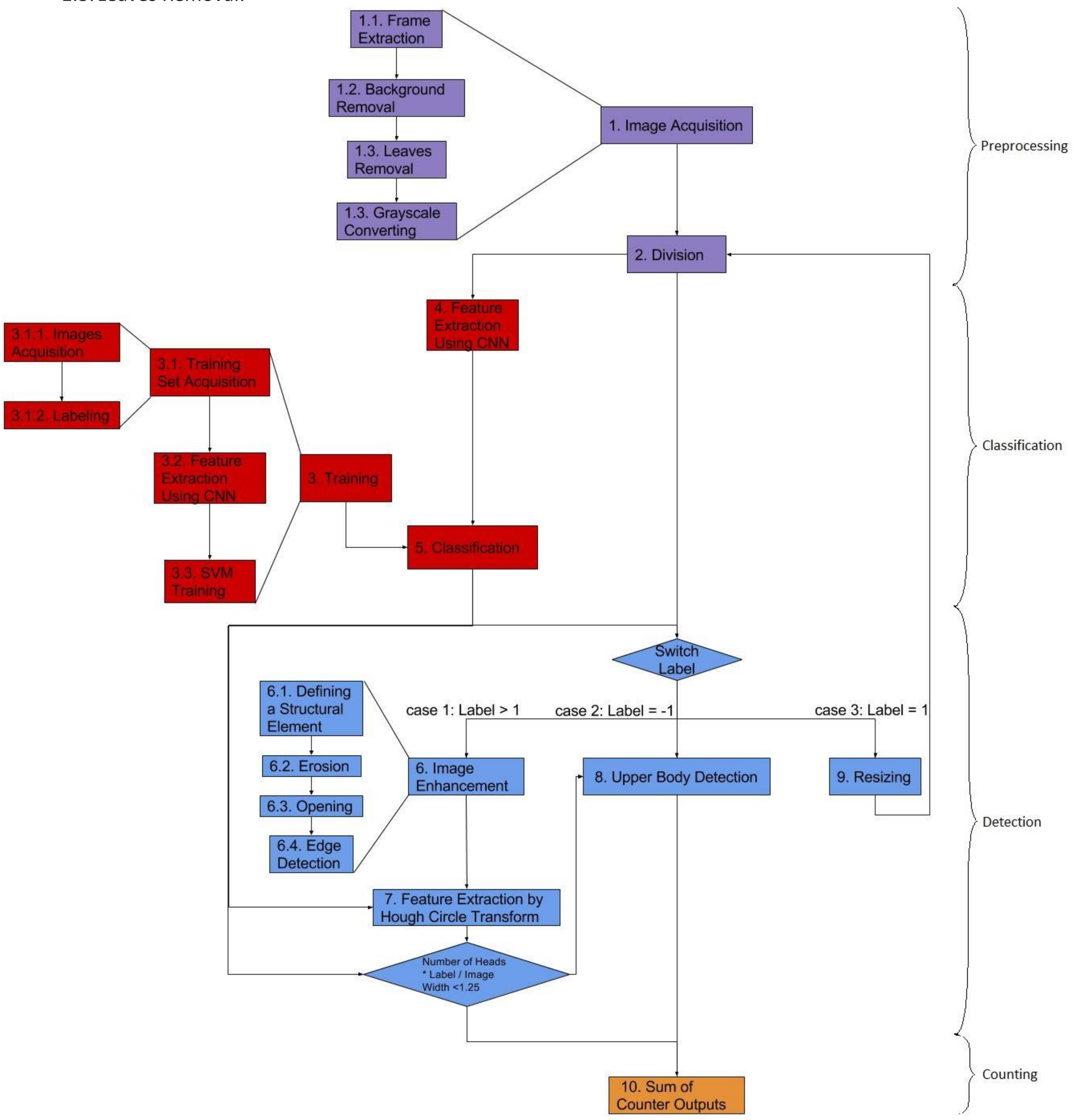

Figure 5: Diagram of the proposed method.

Leaves in the image are removed by finding out all the pixels where green channel has greater value than both of the other two channels and then painting them black.

In the circle finding procedure, objects that are not heads can also be detected as heads. And leaves cause much more trouble than other objects as they can move due to wind, making the background removal unable to mark them as background. Also if we zoom in the grayscale image, the pattern of leaves is indeed similar to crowded heads. Although they can greatly lower the accuracy, it is easy to find out the leaves because their color is green which is very different from the color of people's heads. Painting all green pixels black can remove most of leaves while preserving the pixels of people's heads as we assume their heads (skin and hair) cannot be green. Although they can wear green hats, this case is rare. In Figure 8, leaves are removed from Figure 7.

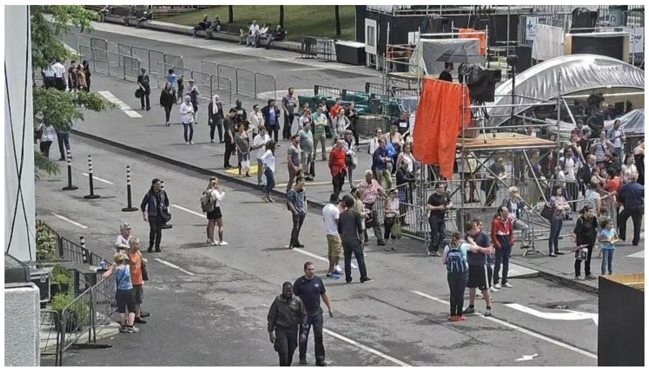
Figure 6: The original image extracted from the video.

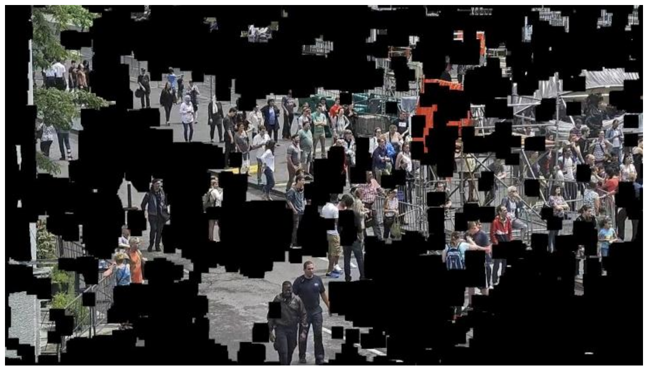
Figure 7: The image with background removed.

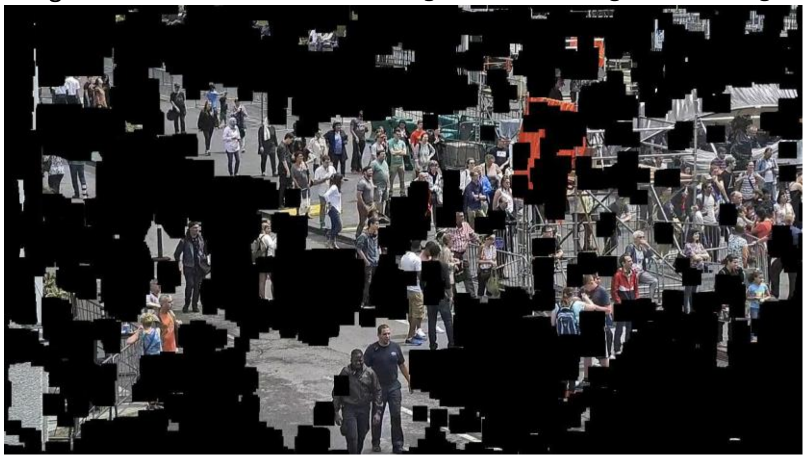
Figure 8: The image after leaves removal.

1.4. Converting to Grayscale

Because our image enhancement operations perform on grayscale images, the image is then converted into grey scale according to Formula (1), which determines each pixel's luminosity based on the red, green, and blue channels.

2) Division:

The image is equally divided into 3 by 3 small patches. The image is divided because in each of the patches, head sizes have a better uniformity. Detecting heads by looking for circles of a particular range of radius is more feasible in each small patch. Dividing the image into 3 by 3 small patches is just one way of doing this. Finding the optimal way to divide the image may be difficult as the optimal way changes depending on the angle of view and we haven't worked on finding it. After division, the following steps will be applied to every small patch.

ii. Classification:

Here a machine learning model is trained. Then it is used to find out the head size in each small patch.

3) Training:

A model was trained to decide the head size in an image. After training, this model will be used to find out the head size in each of the small patches generated in Step 2). Here a support vector machine (SVM)[10] with the features extracted by the 'fc7' deep layer of the AlexNet[11] is trained.

In computer vision, SVM and convolutional neural network (CNN)[11] are two common models. A CNN uses a cascade of layers including an input and an output layer, as well as multiple hidden layers between the input and the output layer. It is in general more complex as it requires more parameters. The complexity enables it to directly take an image as input, use all information in the image and classify the image accurately. However a large training set is required. Otherwise we would encounter overtraining issue which means that the models memorize all the training data instead of learning the classification rules. Since our dataset consists of only 100 images, which is very small, CNN does not work well. A CNN with 30 convolution layers of size 3 by 3, gives 100% accuracy when classifying the training data but the accuracy for the testing data is only 30%. Due to time limit, we did not test CNN with other configuration.

SVM is a simpler model and the raw image with hundreds of thousands of pixels is too complicated for it to learn. Thus features characterizing the image are extracted and used as the inputs of the SVM. Common features are SURF [13], SIFT [14] etc. Apart from the common features, we also consider the features extracted by AlexNet which is a CNN. But different from the CNN we tested. AlexNet is already trained with another dataset which is much larger than ours. The detail of AlexNet and this kind of feature will be described in Step 3.2. The original SVM views data points as high dimensional vectors and tries to find a hyper plane to separate the data points with different labels. To classify the points that cannot be separated by a plane, a non-linear kernel function [15] can be applied to all data points to map them to new locations. Then hopefully those new points can be separated by a plane. The Gaussian kernel [16] is commonly used. With the data set we collected, the combinations of SVM with or without Gaussian kernel and different features including SURF, SIFT, MSER [17], BRISK [18], SURF+SIFT and AlexNet features were trained and tested. SVM with no kernel using AlexNet features gives the highest testing accuracy of 45%, which is higher than CNN. Thus we use SVM with the features extracted by the 'fc7' deep layer of the AlexNet. Although the accuracy for classification is not very high, the final accuracy for crowd estimation is high.

3.1. Training Set Acquisition:

The training set is a set of labeled examples to train our model. In our case, the examples are images, and labels are the head sizes in the corresponding images.

3.1.1. Images Acquisition:

A dataset [19] of black-and-white images of very crowded scenarios in various scenes is downloaded. The dimension of the images ranges from 384 pixels to 1023 pixels. All the images are then divided into 3 by 3 small patches of equal dimension. They are divided because in small images, the head sizes do not vary too much, making it easier to decide the head size in the image. Then 100 small patches are selected randomly to form our data set. The data set is very small as labeling them would be time consuming.

3.1.2. Labeling:

According to the size of heads in the image, each small patch was manually labeled as one of the seven categories in Table 1:

Table 1: Labels for the SVM.

Label Name	Description and Corresponding Range of Head Radius (in pixels)
'-1'	Very few heads are in the image or it is not crowded
'1'	Head radius is smaller than 1.
'2'	Head radius is between 1 and 5.
'5'	Head radius is between 2 and 10.
'10'	Head radius is between 5 and 15.
'15'	Head radius is between 10 and 20.
'20'	Head radius is between 15 and 25.

To determine the head size, the same operations in Step 2) of the original method will be applied to the image. Then algorithm similar to Step 3) of the original method is performed on the resulting edge graph multiple times. Each time, we look for circles of radius in different ranges. Possible ranges are [1, 5], [2, 10], [5, 15], [10, 20] and [15, 25]. Then the range with which the algorithm best finds heads is selected and the image is labeled with the label corresponding to that range. If the heads are too small that none of the ranges works well, the image is labeled as '1'. If the heads size exceeds the maximum range listed (the size is greater than 25 pixels) or very few heads are in the image, the image is labeled as '-1'. Compared to counting the number of people in very crowded scenario, measuring the head size is less time consuming, thus we have enough time to label the 100 examples.

3.2. Feature Extraction Using AlexNet:

A convolutional neural network (CNN) is a class of deep learning [20] neural network that have successfully been applied to analyzing images. In deep learning, a cascade of many layers of nonlinear processing units for feature extraction is used. After training, the output of each layer can be viewed as a feature representation of the input image. Each successive layer uses the output from the previous layer as input and derives a set of higher level features. AlexNet is a CNN consisting of 25 layers. It works well for classification tasks due to its complexity. In this method, the AlexNet pre-trained with ImageNet Challenge dataset is employed to extract features, and the top level features outputted by 'fc7'(the highest deep learning layer) is used.

3.3. SVM Training:

A multiclass SVM was trained based on the settings described above. It is trained with the dataset formed by Step 3.1. It uses features generated by AlexNet and uses no kernel function. The training takes around 10 minutes on a machine with 4-core 3.6GHz Intel Core i7-4796 CPU.

4) Feature Extraction Using CNN:

Features of each small patch are extracted the same way as in Step 3.2. The top level features outputted by AlexNet are extracted.

5) Classification:

The small patch is classified into one of the 7 categories described in Step 3.1.2 by the SVM trained in Step 3.3. The SVM takes the features in Step 4) and outputs a label from one of the 7 labels in Table 1. From the label description in Table 1, the range of the radii of heads in the small patch is known.

iii. Detection:

Depending on the head size and whether it is crowded in the small patch, different strategies are used. If the label is '1', meaning that the heads are very small, the image is resized by 3 in Step 9), then the whole crowd counting program is run on the resized patch recursively; If the label is '-1', heads are large or it is not crowded, people in the small patch are detected by an upper body detector in Step 8); Else go to Step 6) and Step 7) to find circular objects as large as people's heads.

Case 1: Label > 1

6) Image Enhancement:

Same operations in Step 2) of the original method will be applied to each patch. Those operations include defining a structural element, erosion, opening and edge detection.

7) Feature Extraction by Hough Circle Transform:

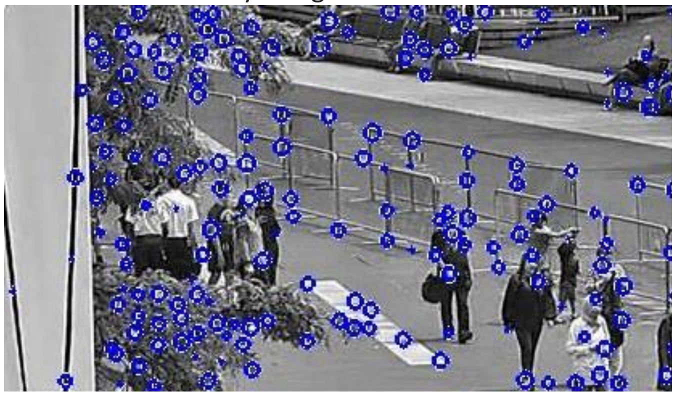
Figure 9: Circles found in uncrowded image.

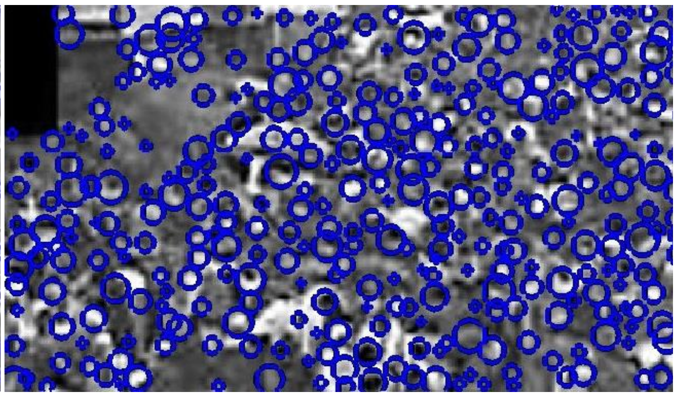
Figure 10: Circles found in crowded image.

This step is similar to Step 3) in the original method. Circles of radius in some range are looked for by Hough Circle Transform. Except that the radius range of the circles is decided by the SVM in Step 5). The SVM outputs a label indicating the head size in the small patch. And the label can only be larger than 1 (otherwise, we would not go to this step). By looking up Table 1, we can find the range of radius corresponding to that label. After this step, the circles found are the heads detected.

The finding circles method works better when it is very crowded because heads are next to each other and no other circular objects as large as heads can fit in the gap between heads. But when it is not crowded, it detects many other objects as heads, which can lead to very high error. However, the detections in crowded and uncrowded images have very different patterns. Figure 9 and Figure 10 show the circles found in an uncrowded image and a crowded one. In Figure 9, the spaces between circles are large. While in Figure 10, circles are very close to each other and circles of the same size cannot fit in the gap between circles at crowded places. The image is almost filled up by the circles. This difference allows us to distinguish crowded and uncrowded images and apply a different strategy to an uncrowded image. We will use a detection based method for uncrowded images because it works well when there are few occlusions, which is the case in uncrowded images.

To check whether it is crowded, expression (5) will be computed and compared to a threshold.

$$Number\ of\ Circles\ *\ Label\ Outputted\ by\ SVM\ /\ Width\ of\ the\ Image \qquad (5)$$

If the value of the expression is not less than 1.25, it is crowded, go to Step 10) to count circles, else go to Step 8) and switch to upper body detection.

In crowded images, we expect the circles found take up a higher proportion of area in the image, which is positively related to the number of circles and the radius of circles, and negatively related to the dimension of the image. Expression (4) roughly reflects this. Note that the label outputted by SVM is usually the middle value of the radius range. The label is larger when larger circles are looked for. The value of the expression is compared with the threshold 1.25 which is derived by calculating the values of 45 small patches and trying to find a value that best separates the crowded and uncrowded images.

This rule to distinguish crowded and uncrowded images is used just because it is simple and we haven't tried other criterion.

8) Upper Body Detection:

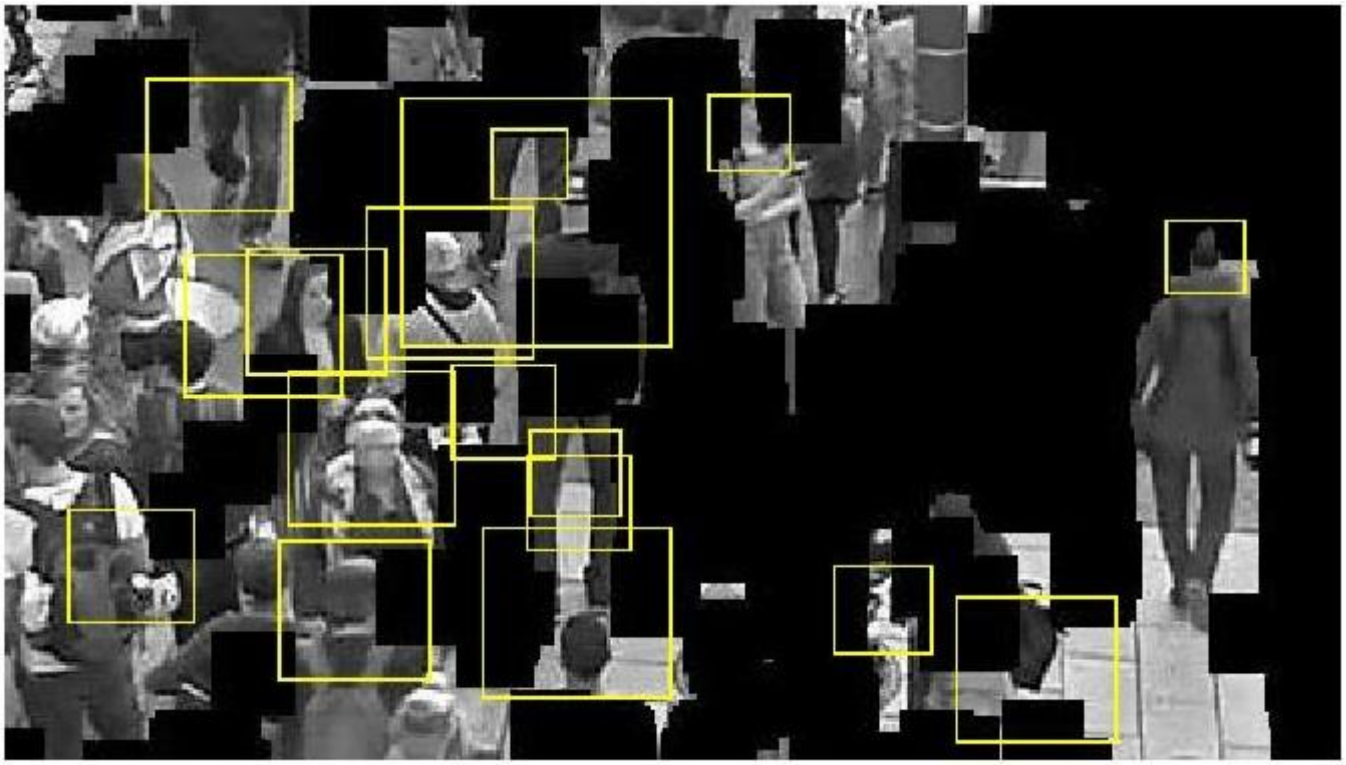

Figure 11: Upper bodies detected by the upper body detector.

A cascade upper body detector [20] using the Viola-Jones algorithm is used to detect people by their upper body. Go to Step 10) after detection. This step is done in two cases: (I) the label outputted by SVM is '-1', meaning that the SVM classifies it as uncrowded and (II) Step 6) and Step 7) are already performed and not many heads are found. Both cases indicate that the image is not very crowded making the finding circles method not accurate. However, since there are fewer occlusions in uncrowded images, detection based methods work well. 3 detectors are considered: people detector, upper body detector and face detector. We will also use those three detection based methods to compare with our method. The detailed configurations of those detectors will be explained in the results section. People detector doesn't work well when people's bodies are occluded which often happens in our images. Face detector cannot detect the back of people's heads. Since our cameras are usually installed above peoples' heads, their upper body (heads and shoulders) are often shown in the video when it is not very crowded. Thus upper body detector is able to detect them.

We also ran the 3 detectors on our images, and upper body detector gives the best result when it is crowded. Although it makes false detection on other objects, they are removed in the background removal step. The result is show in Table 1 in the results section. Figure 1 shows the effect of upper body detection.

9) Resizing:

This step is done when heads are too small in the image. Both finding circles and upper body detection have difficulty in detecting people. Thus the image is resized by scaling up by 3, and the crowd estimation program is run again on the resized image from the division step. After the small patch is resized, it is as large as the original undivided image. Since the background and leaves are already removed in the small patch and it is already in grey scale, the resized image is sent to Step 2).

iv. Counting:

Finally the number of people detected in each small patch is counted and then summed up.

10) Sum of Counter Output:

In this step, the program waits until the people finding in all 9 small patches complete. Then the numbers of people in each small patch are counted and then summed up and that is the number of people estimated in the whole image. The inputs can be either the number of upper bodies detected in Step 8) or the number of heads found in Step 7).

4. Results

We tested our methods with 6 images. Among those images, two of them are a frame extracted from videos which enable us to remove the background. The numbers of people in most of the images are counted manually (the special case will be explained later). The results of images and video frames will be described separately.

Apart from the original and proposed method, detection based methods are also tested to compare with our methods. They are:

1. People Detection

   A people detector [22] using aggregate channel features (ACF) trained with 'caltech-50x21' dataset is used to detect people. The detector is provided by Matlab. The number of people detected is outputted.
2. Upper Body Detection

   A cascade upper body detector using the Viola-Jones algorithm is used to detect upper body. The detector is provided by Matlab. The number of upper bodies detected is outputted.
3. Face Detection

   A cascade face detector [23] using the Viola-Jones algorithm is used to detect upper body. The detector is provided by Matlab. The number of faces detected is outputted.

The original method, the proposed method and the three detection based methods are used to estimate the numbers of people in the 6 testing images. The error is calculated by the absolute difference between the estimation and the manually counted number, together divided by the counted number. We expect lower error.

I. Testing on Images:

We tested our methods with 4 images shown in Figure 12 to Figure 15. Image 1, 3 and 4 are images of crowded scenarios similar to our training set while in Image 2, it is not crowded at all. Because we don't have the video, the background removal step in the proposed method is skipped. The numbers of people in those images are counted manually. The test results are shown in Table 2.

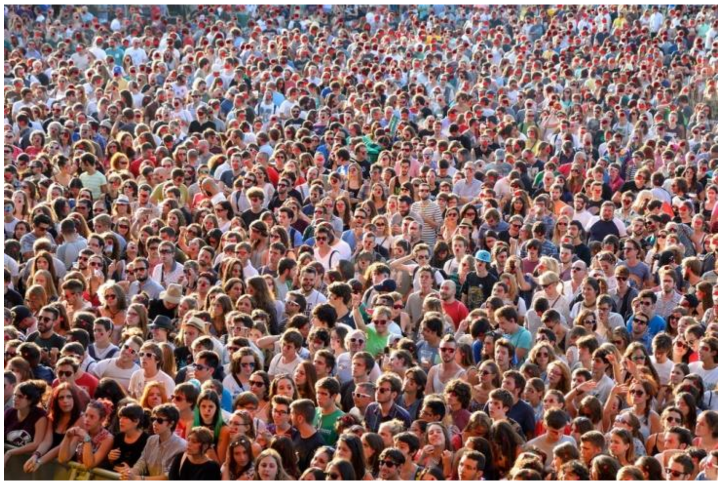
Figure 12: Image 1.

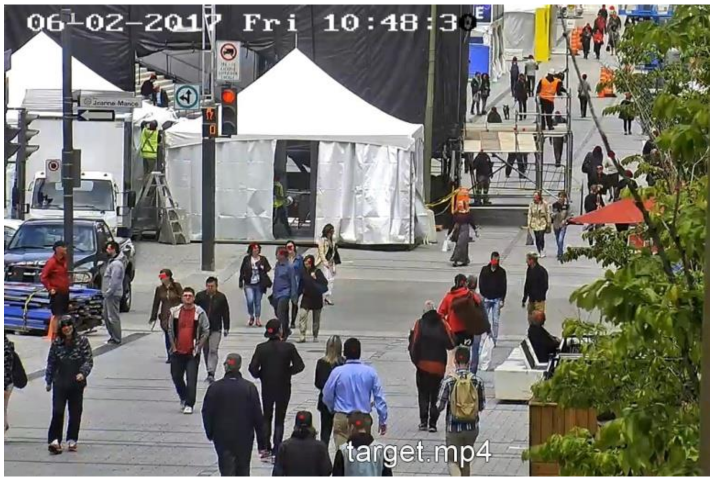

Figure 13: Image 2.

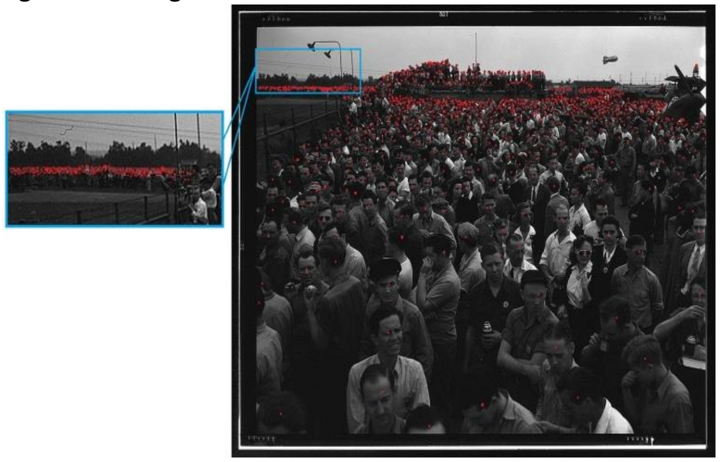
Figure 14: Image 3.

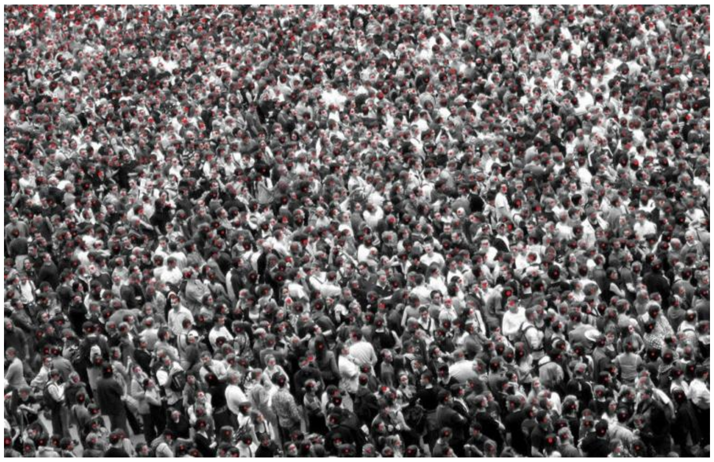
Figure 15: Image 4.

Table 2: The results of all 5 methods on images.

Image Id		Ground Truth	Image Processing Based		Detection Based		
			Proposed	Original	People Detection	Upper Body Detection	Face Detection
1	Count	1237	1531	605	26	598	270
	Error		23%	51%	98%	52%	78%
2	Count	58	98	442	36	242	82
	Error		69%	662%	38%	17%	41%
3	Count	959	1159	401	4	217	103
	Error		21%	58%	100%	81%	89%
4	Count	1526	1775	767	21	541	99
	Error		16%	50%	99%	65%	94%

To see how good the 5 methods detect people, the images of people detected by all our methods in Image 1 are shown below:

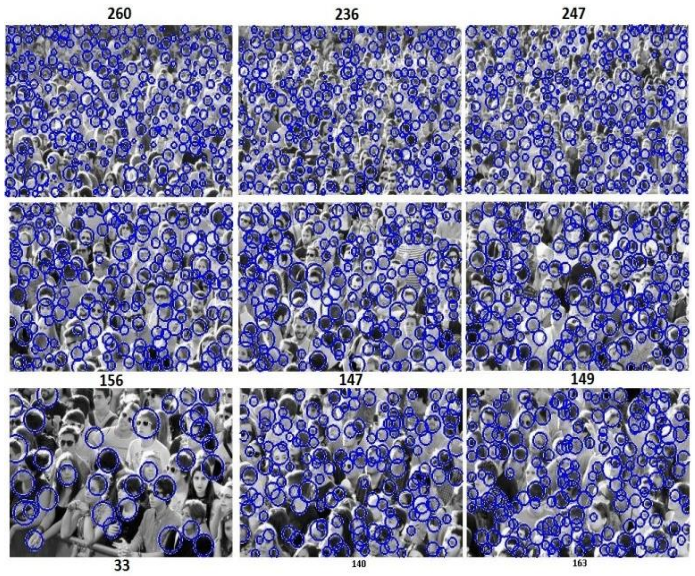

Figure 16: Heads found by the proposed method.

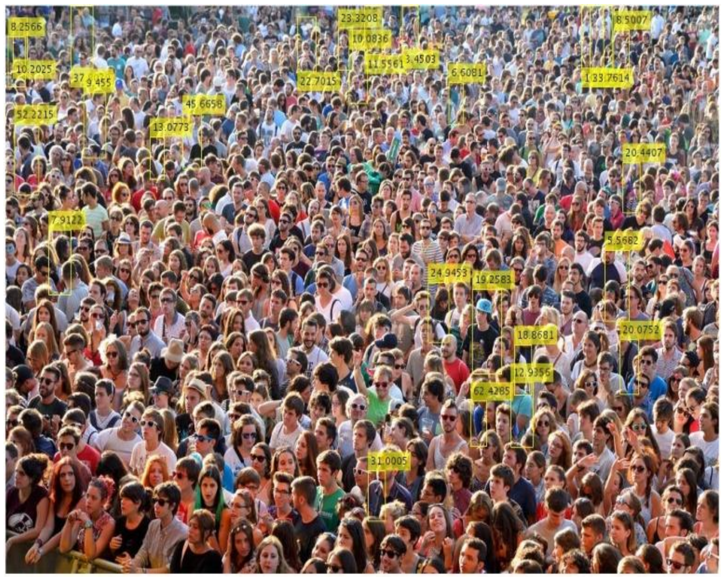

Figure 17: People found by people detection.

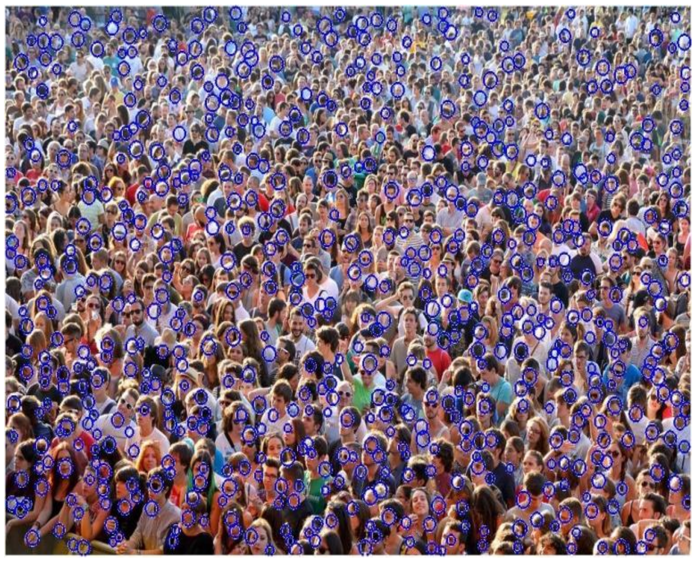

Figure 18: Heads found by the original method.

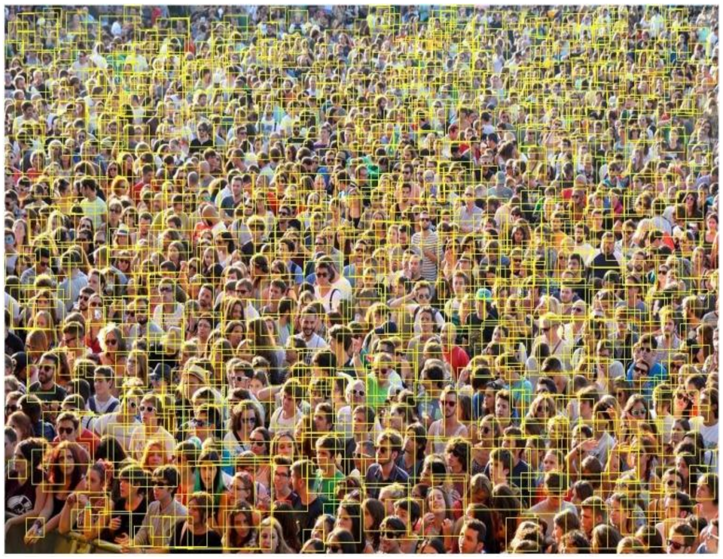

Figure 19: People found by upper body detection.

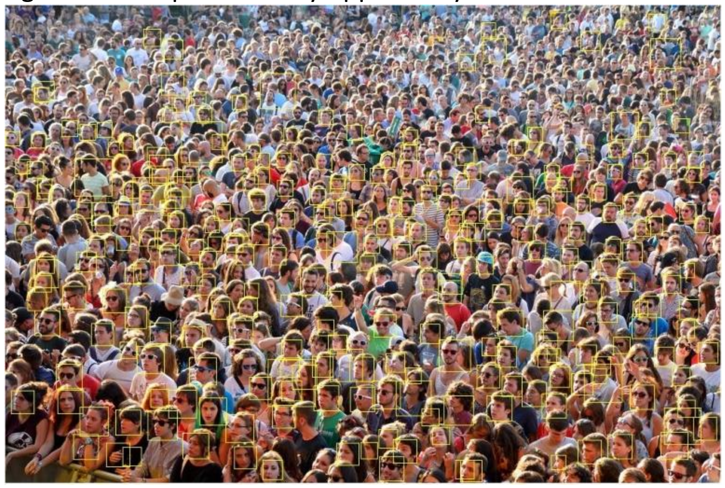

Figure 20: Faces found by face detection.

As is shown in Table 2, detection based methods are in general not good at counting in crowded scenarios. They have error larger than 50% for Image 1, 3 and 4. From Figure 17 we see that very few people are detected by the people detector. This is probably caused by occlusion. In Figure 20, people far from the camera are not detected by the face detector as they are too small in the image. Upper body detection works better than the other two detection based methods but still has an error 0f 52%.

The original method using a fixed range of head size works better than the detection based methods in general. But it also has obvious problems. In Figure 18, we can find lots of misdetection in the front and lots of heads in the back are not detected. Also, it works terribly when it is less crowded images like Image 2.

Our modified method can automatically decide the size of head. The problem of the original method is fixed. The detection is much more accurate as can be seen in Figure 16. Although it still makes some false detection, since the crowd size is large, that error can be neglected and the percent of error is small. For crowded images, its error is between 25% and 15%. For image where it is less crowded like Image 2, the proposed method switches to upper body detection. Also leaves are removed which reduce the number of false detections. Thus the proposed method gives a much lower error (69%) than the original one (662%). Its performance on other crowded images can be visualized in Figure 21 to Figure 22.

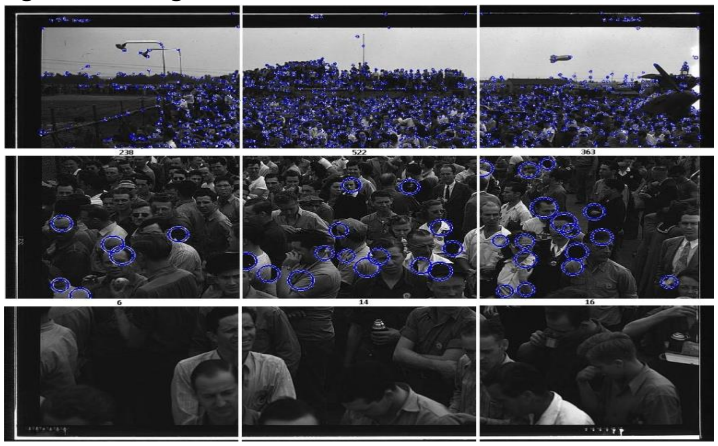

Figure 21: Heads found by the proposed method in Image 3

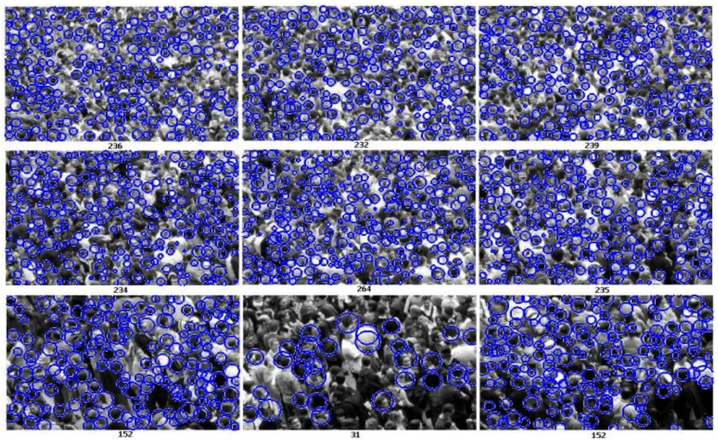

Figure 22: Heads found by the proposed method in Image 4

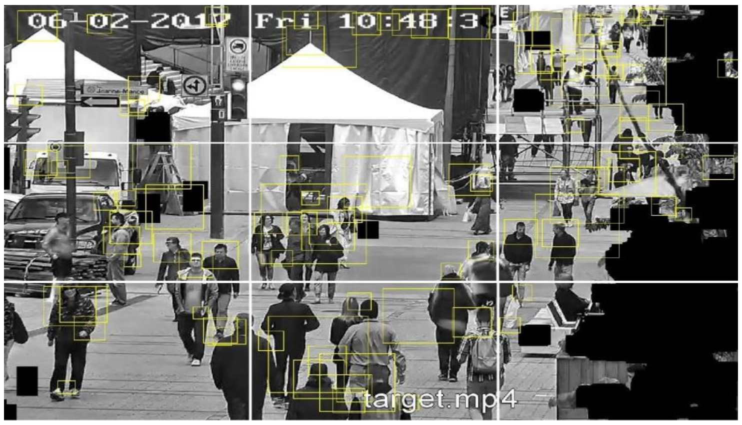

Figure 23: Heads found by the proposed method in Image 2.

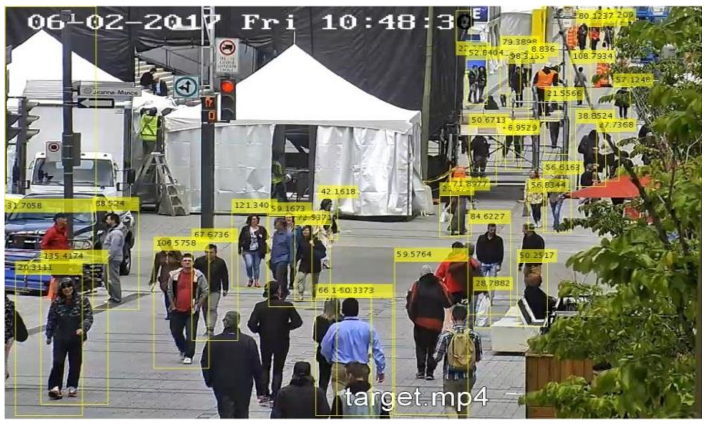

Figure 24: People found by people detection in Image 2.

However in image 2 where it is not crowded at all, the proposed method uses upper body detection which makes lots of false detection. People detection is more accurate than the upper body detector when people' whole bodies are visible. However, given the video, we would be able to remove the background and reduce the number of false detections. The result of people detection on Image 2 is shown in Figure 24.

II. Testing on Video Frames

We tested our methods with 2 frames extracted from different videos shown in Figure 25 and Figure 26. Video Frame 1 is very crowded and Video Frame 2 is less crowded. The numbers of people in Video Frame 2 is counted manually. However, we are not able to count in Video Frame 1 as the heads are too small in the back. To get the crowd size, in the area where heads are large, people are counted manually. In the area where people's heads are too small, the crowd size is estimated by the area multiplied by crowd density. The test results are shown in Table 3.

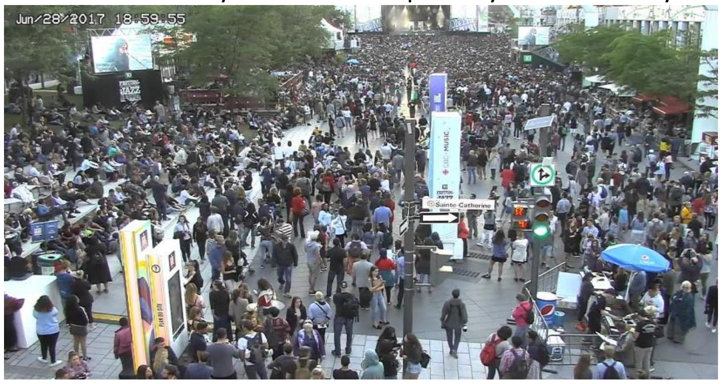

Figure 25: Video Frame 1

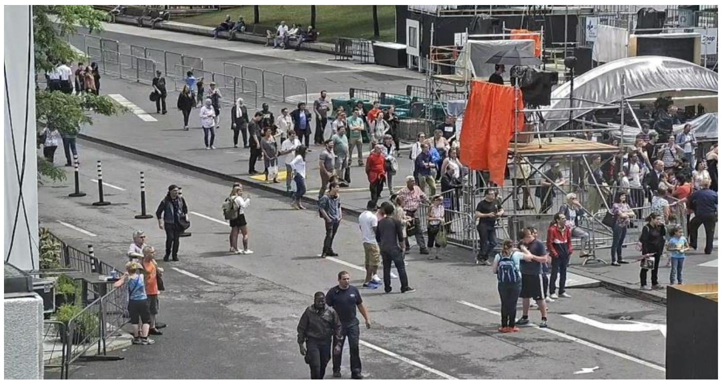

Figure 26: Video Frame 2

Table 3: The results of all 5 methods on video fames

Video Frame Id		Ground Truth	Image Processing Based		Detection Based		
			Proposed	Original	People Detection	Upper Body Detection	Face Detection
1	Count	2497	2658	4031	42	402	144
	Error		6%	61%	98%	84%	94%
2	Count	104	98	3238	50	297	79
	Error		6%	3200%	52%	186%	24%

For both video frames, the error of the proposed method is 6%, which is much lower than the error by the original method. In Video Frame 2, although it is not very crowded, the proposed method gives the best accuracy among all the methods including the detection based. In Figure 27, the background is removed so that no false detection can be made on the background. Without background removal, the proposed method gives an error of 2300%. Using a video to remove the background is indeed helpful.

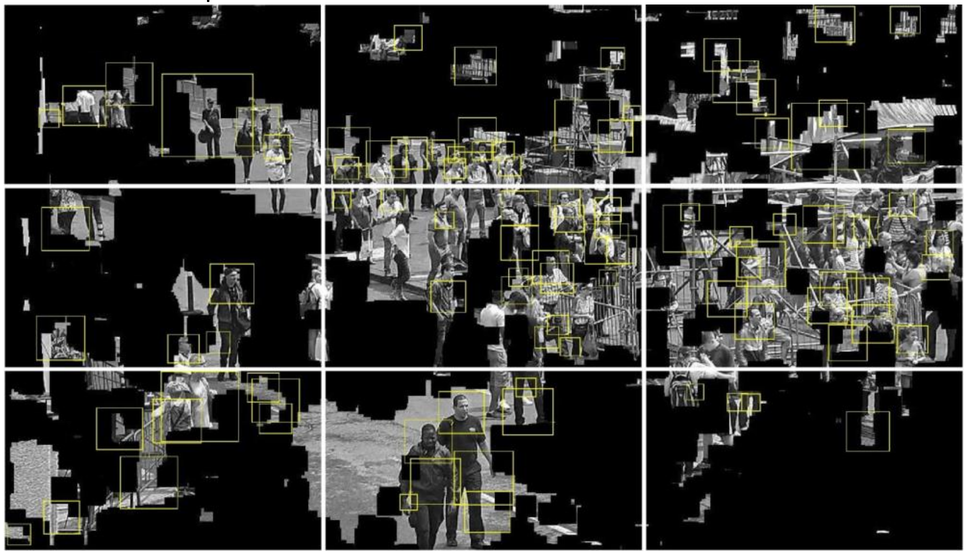

Figure 27: People detected by the proposed method in Video Frame 2.

I. Resolution and Performance

Another thing to mention is the resolution of the image. A lower resolution makes the heads smaller, which makes detection and circle finding more difficult. When the heads are already small in the image, a lower resolution makes the heads disappear, leaving only some texture. This effect can be shown in Figure 28. Thus a lower resolution leads to a lower accuracy. The resolution of Image 5 is reduced to 0.8, 0.6 and 0.4. Then the proposed method was run on the resized images. The change of error is shown in Table 4.

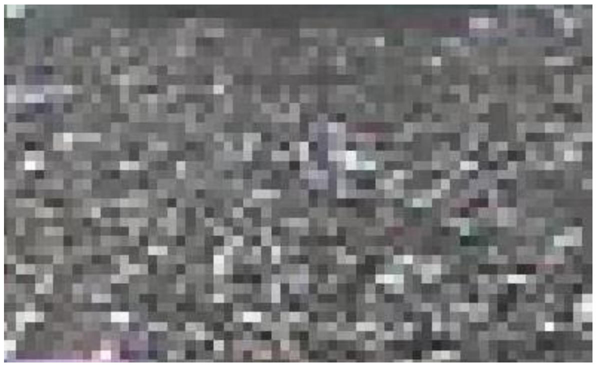
Figure 28: The crowded part in Image 5 after it is resized by half.

Table 4: The error by the 5 methods running on Video Frame 1 with different resolutions.

Resolution/Original Resolution		Proposed Method	Original Method	People Detection	Upper Body Detection	Face Detection
1	Estimation	2658	4031	42	402	144
	Error	6%	61%	98%	84%	94%
0.8	Estimation	1314	2765	32	323	93
	Error	47%	11%	99%	87%	96%
0.6	Estimation	584	1609	21	210	44
	Error	77%	36%	99%	92%	98%
0.4	Estimation	32	736	16	96	10
	Error	99%	71%	99%	96%	100%

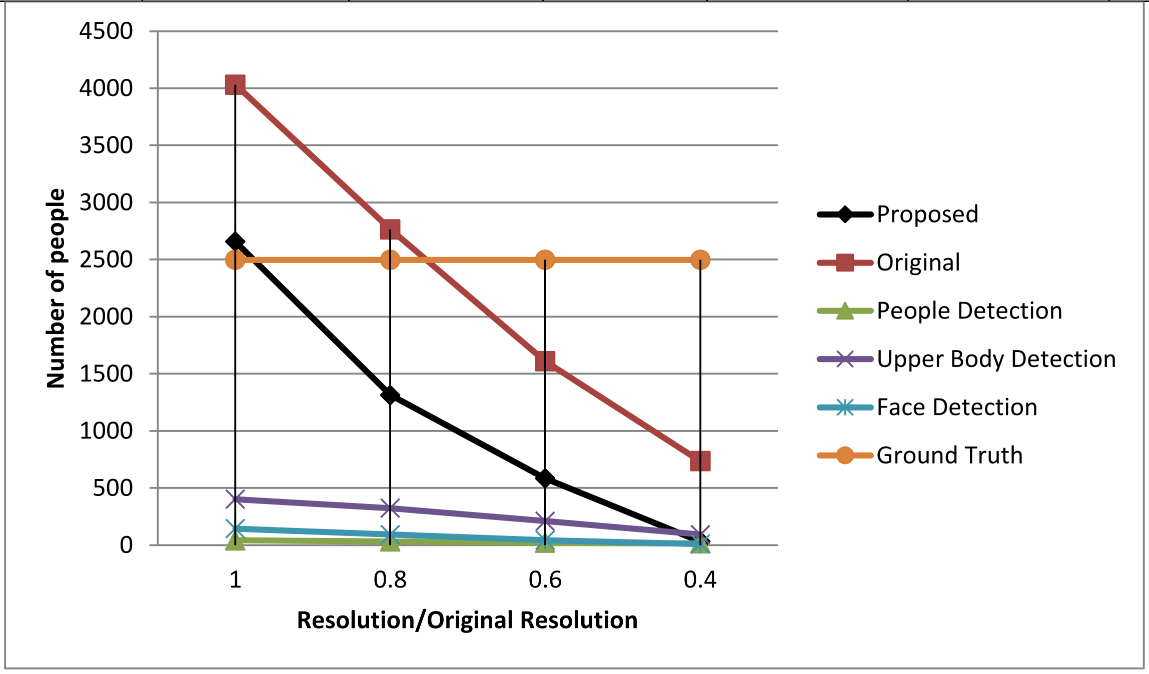

Figure 29: The results by the 5 methods running on Video Frame 1 with different resolutions.

The time it takes for the proposed method to compute depends on how people in each of the 9 patches are detected. Running on a machine with a 4-core 3.6GHz Intel Core i7-4796 CPU, the upper body detection takes 2 seconds on each patch. If circle finding is used, the first two steps: Image Enhancement and Feature Extraction by Hough Circle Transform take 1 second. If after that, not many heads are found, we will switch to upper body detection. Thus another 2 seconds will be needed. If heads are two small, the small patch will be resized by 3. Then we will run the entire process except Image Acquisition again which will double the time to estimate the crowd size. Table 5 shows the time each major step takes.

Table 5: The time major steps take in the proposed method.

Step	Time (sec)
Background Removal	3
Feature Extraction and Classification	2
Upper Body Detection in 1 of the 9 patches	2
Image Enhancement and Feature Extraction by Hough Circle Transform in 1 of the 9 patches	1
Other	2

Table 6 shows the time for each method to estimate the crowd size in video frame 1. In the proposed method, the image is divided into 9 small patches. Upper body detection is used on 4 of them. Two of the small patches are resized by 3 and then further divided. The total time it takes is 35 seconds.

Table 6: The time for each method to estimate crowd size in Video Frame 1.

Methods	Proposed	Original	People Detection	Upper Body Detection	Face Detection
Time (sec)	35	4	0	70	26

5. Conclusions

5.1. Achievements

In the proposed method, the image is first divided into 3 by 3 small patches. A model is trained to find out the head size in each small patch. If the head size is within some range, the small patch is then processed with various image enhancement techniques. After that, circles of sizes found by the model are detected as heads which are then counted. If the small patch is not crowded, upper body detection is used. If the heads are too small, the patch is resized and the program is run on the resized image recursively.

Our proposed approach provides the best results compared to the original image processing method and the 3 detection based methods when dealing with very crowded scenarios. However if we work on an image without background removal and people's entire bodies are visible, the proposed method using upper body detection is less accurate than a people detector.

5.2. Future Works

Regarding future work, the most important improvement would be to have more crowd images labeled. With more labeled examples, we can train a neural network to decide the head size. In the neural network, we can let the loss function be the difference between the numbers of people counted using the real head size and the estimation using predicted head size. If the images are labeled with number of people in the image, we can also try the machine learning based methods. Another trick that may improve the result significantly would be removing the background before Image Enhancement. Also training with colored photos can eliminate the false detection of leaves.

References

[1] Idrees, Haroon, Khurram Soomro, and Mubarak Shah. "Detecting humans in dense crowds using locally-consistent scale prior and global occlusion reasoning." IEEE transactions on pattern analysis and machine intelligence37.10 (2015): 1986-1998, available at https://www.ncbi.nlm.nih.gov/pubmed/26340254

[2] Lwanga Christine Wanjira, "Crowd Size Estimator Using Image Processing Techniques", April 28, 2014, available at

http://eie.uonbi.ac.ke/sites/default/files/cae/engineering/eie/CROWD%20SIZE%20ESTIMATOR%20USING%20IMAGE%20PROCESSING%20TECHNIQUES.pdf

[3] Ma, Ruihua, et al. "On pixel count based crowd density estimation for visual surveillance." Cybernetics and Intelligent Systems, 2004 IEEE Conference on. Vol. 1. IEEE, 2004.

[4] Cho, Siu-Yeung, Tommy WS Chow, and Chi-Tat Leung. "A neural-based crowd estimation by hybrid global learning algorithm." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29.4 (1999): 535-541, available at http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=775269

[5] Cho, Siu-Yeung, and Tommy WS Chow. "A fast neural learning vision system for crowd estimation at underground stations platform." Neural processing letters 10.2 (1999): 111-120, available at https://link.springer.com/content/pdf/10.1023%2FA%3A1018781301409.pdf

[6] Marana, A. NSAV, et al. "Automatic estimation of crowd density using texture." Safety Science 28.3 (1998): 165-175.

[7] van den Boomgard, R, and R. van Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Images," Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing, Vol. 54, No. 3, pp. 254-258, May 1992.

[8] “Morphologically Open Image”, available at https://www.mathworks.com/help/images/ref/imopen.html

[9] Canny, John, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-8, No. 6, 1986, pp. 679-698.

[10] Cortes, C. and Vapnik, V. "Support-vector networks". Machine Learning. Vol. 20, No. 3, pp 273–297, 1995.

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[12] Jianxin Wu, “Introduction to Convolutional Neural Networks”, May 1, 2017, available at https://cs.nju.edu.cn/wujx/paper/CNN.pdf

[13] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." Computer vision–ECCV 2006 (2006): 404-417.

[14] U.S. Patent 6,711,293, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", David Lowe's patent for the SIFT algorithm, March 23, 2004.

[15] Shawe-Taylor, J. and Cristianini, N., “Kernel Methods for Pattern Analysis”, Cambridge University Press, 2004.

[16] Scholkopf, Bernhard, et al. "Comparing support vector machines with Gaussian kernels to radial basis function classifiers." IEEE transactions on Signal Processing 45.11 (1997): 2758-2765, available at http://ieeexplore.ieee.org/abstract/document/650102/

[17] Matas, Jiri, et al. "Robust wide-baseline stereo from maximally stable extremal regions." Image and vision computing 22.10 (2004): 761-767.

[18] Leutenegger, S., M. Chli and R. Siegwart. "BRISK: Binary Robust Invariant Scalable Keypoints", Proceedings of the IEEE International Conference, ICCV, 2011.

[19] The data set is available at http://crcv.ucf.edu/data/crowd_counting.php

[20] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[21] Detect objects using the Viola-Jones algorithm, available at https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html

[22] Detect people using aggregate channel features, available at https://www.mathworks.com/help/vision/ref/peopledetectoracf.html

[23] Detect objects using the Viola-Jones algorithm, available at https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html