Crowd Counting System Using Unmanned Aerial Vehicle (UAV)

Prince Makwana , Vibhor Choudhary , Dr. Gaurav Singal

Abstract—Unmanned Aerial Vehicle (UAV)/ Drones has created much significance in recent times. It is a booming technology which are mostly used in military applications, their use is rapidly expanding to commercial, scientific, agricultural and other applications. Understanding and dealing with safety aspects of crowd dynamics in mass gatherings of people related to sports, religious and cultural activities is very important, specifically with respect to crowd risk analysis and crowd safety. In this regard, efficient monitoring and other safe crowd management techniques have been used to minimize the risks associated with such mass gatherings. An example of this technique is real time monitoring of crowd using a UAV/Drones; this technique is becoming increasingly popular with the objective to save human lives, preserve environment, protect property, keep the peace, and uphold governmental authority. In this paper, a pretrained crowd counting model of ShanghaiTech has been proposed and tested. The system has deployed real time crowd monitoring technique using real-time video frames taken by UAVs. In our research, we proposed the novel end-to-end Single Image Multi-Cascaded CNN model to estimate crowd from aerial view. We also proposed Keras model but due to lack of time testing is not completed using Keras model. The results of MCCNN model is accurate up to 80%.

I. INTRODUCTION

Object Detection from aerial view is challenging task due to vibrations in video frames. Human Detection in crowd video scenes is getting another difficult task. If detecting a particular human who lost in some public gathering is even more challenging task. For all this problems, we can start from estimating crowd from aerial view using. There is a great advantage of estimating dense crowd which aims for security, resource management, traffic monitoring, etc. Recognizing and monitoring in areas where there are chances of heavy constrictions is still a big problem. These events involving large gatherings of people, need to be controlled and monitored. In regard of this, we address the problem: How to estimate number of people in dense crowd scenes?

Fig -1

Fig -2

From Figures 1 & 2, High-density crowd scenes pose major problems in detection and monitoring, due to heavy occlusions and significant variations in people's sizes and appearances. Due to geometric alterations because of perspective issues and the foreshortening in these scenes, density may greatly vary across the field of vision. To counter this issue, we proposed a method in high dense crowd image is divided into 10 different patches and thus evaluating these patches contain single image having low density crowd makes it easier to estimate. Counts based on head detection are not very accurate in high-density crowds, with heads being as small as tens of pixels, but to improve this approach, we proposed a MCCNN model.

II. RELATED WORK

In this section, we viewed 4 different crowd counting methods.

1. Detection-based methods:
   Here, we are using a moving window-like detector to identify people in an image and count how many there are. The methods used for detection require well trained classifiers that can extract low-level features. Although these methods work well for detecting faces,

they do not perform well on crowded images as most of the target objects are not clearly visible.

2. Regression-based method:
   We were unable to extract low-level features using the above approach. Regression-based methods come up trumps here. WE first crop patches from the image and then, for each patch, extract the low level features.

3. Density estimation-based methods:
   We first create a density map for the objects. Then, the algorithm learn linear mapping between the extracted features and their object density maps. We can also use random forest regression to learn non-linear mapping.

4. CNN-based methods:
   Instead of looking at the patches of an image, we build an end-to-end regression method using CNNs. This takes the entire image as input and directly generates the crowd count. CNNs work well with regression or classification tasks, and they have also proved their worth in generating density maps.

In our model, we used Density-based Counting method to estimate crowd count in an image.

Density-based Counting: The concept of an object density map, where the integral (sum) over any sub region equals the number of objects in that region. The density values are estimated from low-level features, thus sharing the advantages of general regression-based methods, while also maintaining location information uses a linear model to predict a pixel's density value from extracted features, and proposed the Maximum Excess over Sub Array (MESA) distance, which is the maximum counting error over all rectangular sub regions, as a learning objective function.

We used two different drone flight controller to test our model:

1. Kk2.1.5 Flight Controller is a multi-rotor flight control board to stabilize the aircraft during the flight. To do this it takes the signal from the 6050MPU gyro/acc (roll, pitch, and yaw) then passes the signal to the Atmega644PA IC. The Atmega644PA IC unit then processes these signals according to the user's selected firmware and passes control signals to the installed Electronic Speed Controllers (ESCs). These signals instruct the ESCs to make fine adjustments to the motors rotational speed which in turn stabilizes your multi-rotor craft. The HobbyKing KK2.1.5 Multi-Rotor control board also uses signals from your radio systems receiver (Rx) and passes these signals to the Atmega644PA IC via the aileron, elevator, throttle and rudder inputs. Once this information has been processed the IC will send varying signals to the ESCs which in turn adjust the rotational speed of each motor to induce controlled flight (up, down, backward, forward, left, right, yaw).

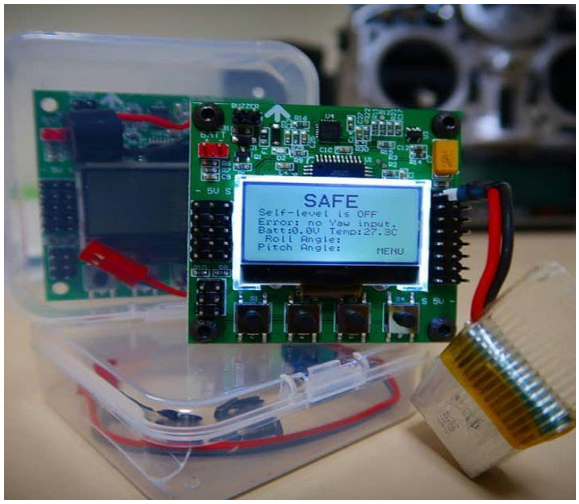


Fig 3

2. Ardupilot is the predecessor of ArduPilot Mega. This is a smaller autopilot that is still based on the Arduino platform. Ardupilot is designed for simpler fixed wing aircraft(3 channels). Supporting autonomous stabilisation, way-point navigation, and one way telemetry with Xbee modules allows for a cost effective UAV autopilot. This board supports 4 RC channels in, and 3 out with one serial port. For stabilisation, Ardupilot supports both thermopiles or an IMU.

Ardupilot Features: 1). Can be used for an autonomous aircraft, car or boat. 2). Built-in hardware failsafe that uses a separate circuit (multiplexer chip and ATTiny processor) to transfer control from the RC system to the autopilot and back again. 3). Includes ability to reboot the main processor in mid-flight. 4). Multiple 3D waypoints (limited only by memory). 5). Altitude controlled with the elevator and throttle. 6). Comes with a 6-pin GPS connector for the 4Hz uBlox5 or 1hz EM406 GPS modules. 7). Has six spare analog inputs (with ADC on each) and six spare digital input/outputs to add additional sensors 8). Supports addition of wireless modules for real-time telemetry. 9). Based on a 16MhZ Atmega328 processor. Total onboard processing power aprox 24 MIPS. 10). Can be powered by either the RC receiver or a separate

battery. 11). Four RC-in channels (plus the autopilot on/off channel) can be processed by the autopilot. Autopilot can also control four channels out. 12). LEDs for power, failsafe (on/off), status and GPS (satellite lock)

Fig 4

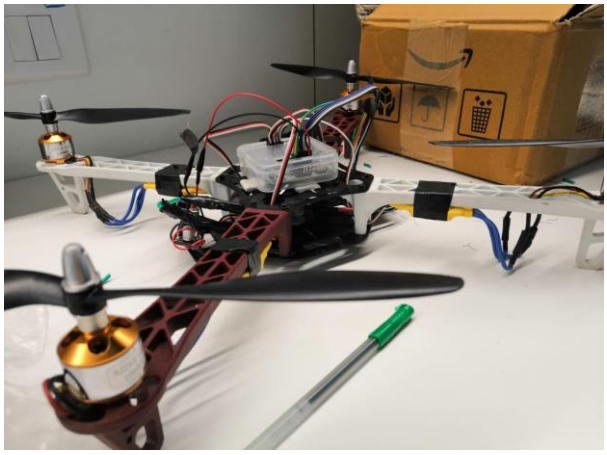


Fig 5: Drone with kk2.1.5 Flight Controller and F450 frame

Fig 6: Autonomous Drone with Ardupilot Flight Controller and F450 frame

III. METHODOLOGY

In this research, inspired by the success of cascaded convolutional networks for related multiple tasks we proposed a method in cascaded fashion as shown in Fig 7. The network takes an image of arbitrary size, and outputs crowd density map. The cascaded network has two stages corresponding to the two sub-tasks, with the first stage learning high-level prior and the second stage preforming density map estimation. Both stages share a set of convolutional features. The first stage consists of a set of convolutional layers and spatial pyramid pooling to handle arbitrarily sized images followed by a set of fully connected layers.

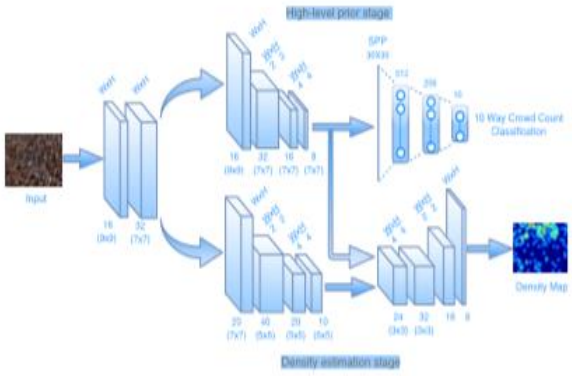


Fig 7 Proposed pre-trained model

The second stage consists of a set of convolutional layers followed by frictionally-strided convolutional layers for up sampling the previous layer's output to account for the loss of details due to earlier pooling layers. Two different set of loss layers are used at the end of the two stages, however, the loss of the second layer is dependent on the output of the earlier stage.

Density estimation

The feature maps obtained from the shared layers are processed by an another CNN network that consists of 4 convolutional layers with a PReLU activation function after every layer. The first two layers are followed by max pooling layers with a stride of 2, due to which the output of CNN layers is down sampled by a factor of 4. The first convolutional layer has 20 feature maps with a filter size of 7 × 7, the second convolutional layer has 40 feature maps with a filter size of 5 × 5, the third layer has 20 feature maps with a filter size of 5 × 5 and the fourth layer has 10 feature maps with a filter size of 5 × 5. The output of this network is combined with that of the last convolutional layer of high-level prior stage using a set of 2 convolutional and 2 fractionally strided convolutional layers. The first two convolutional layers have a filter size of 3 × 3 with 24 and 32 feature maps, respectively. These layers are followed by 2 sets

of fractionally strided convolutional layers with 16 and 18 feature maps, respectively. In addition to integrating high-level prior from an earlier stage, the fractionally strided convolutions learn to upsample the feature maps to the original input size thereby restoring the details lost due to earlier max-pooling layers. The use of these layers results in upsampling of the CNN output by a factor of 4, thus enabling us to regress on full resolution density maps. Standard pixel-wise Euclidean loss is used as the loss layer for this stage. Note that this loss depends on intermediate output of the earlier cascade, thereby enforcing a causal relationship between count classification and density estimation.

Objective function

The cross-entropy loss function for the high-level prior stage is defined as follows:

$$L_c = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}[(y^i = j)F_c(X_i, \Theta)],$$

where N is number of training samples, Θ is a set of network parameters, Xi is the I th training sample, Fc(Xi, Θ) is the classification output, y i is the ground truth class and M is the total number of classes.

The loss function for the density estimation stage is defined as:

$$L_d = \frac{1}{N}\sum_{i=1}^{N} \|F_d(X_i, C_i, \Theta) - D_i\|_2,$$

where Fd(Xi , Ci , Θ) is the estimated density map, Di is the ground truth density map, and Ci are the feature maps obtained from the last convolutional layer of the high-level prior stage. The entire cascaded network is trained using the following unified loss function:

$L = \lambda L_c + L_d$

where λ is a weighting factor.This loss function is unlike traditional multi-task learning, because the loss term of the last stage depends on the output of the earlier one.

Autonomous Flight Plan using Mission Planner

Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project. It is compatible with Windows only. Mission Planner can be used as a configuration utility or as a dynamic control supplement for your autonomous vehicle.

Features of Mission Planner

- Load the firmware (the software) into the autopilot board (i.e. Pixhawk series) that controls your vehicle.
- Setup, configure, and tune your vehicle for optimum performance.
- Plan, save and load autonomous missions into you autopilot with simple point-and-click way-point entry on Google or other maps.
- Download and analyze mission logs created by your autopilot.
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
- With appropriate telemetry hardware we can:
    - Monitor your vehicle's status while in operation.
    - Record telemetry logs which contain much more information the the on-board autopilot logs.
    - View and analyze the telemetry logs.
    - Operate your vehicle in FPV (first person view)

Fig 8 Mission Planner to setup flight plan for automatic flight of drone

V. EXPERIMENTAL RESULTS

In this section, we present the experimental details and evaluation results on two publicly available datasets: ShanghaiTech and UCF CROWD 50. For the purpose of evaluation, the standard metrics used by many existing methods for crowd counting were used. These metrics are defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - y_i'|,$$

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|y_i - y_i'|^2},$$

where MAE is mean absolute error, MSE is mean squared error, N is number of test samples, yi is ground truth count and y 0 I is estimated count corresponding to the I th sample.

Training and implementation details

In this section, details of the training procedure are discussed. To create the training dataset, patches of size ¼, the size of original image are cropped from 100 random locations. Other augmentation techniques like horizontal flipping and noise addition are used to create another 200 patches. The random cropping and augmentation resulted in a total of 300 patches per image in the training dataset. Note that the cropping is used only as a data augmentation technique and the resulting patches are of arbitrary sizes. Several sophisticated methods are proposed in the literature for calculating the ground truth density map. We use a simple method in order to ensure that the improvements achieved are due to the proposed method and are not dependent on the sophisticated methods for calculating the ground truth density maps. Ground truth density map Di corresponding to the i th training patch is calculated by summing a 2D Gaussian kernel centered at every person's location xg as defined below:

$$D_i(x) = \sum_{x_g \in S} \mathcal{N}(x - x_g, \sigma),$$

where σ is the scale parameter of the 2D Gaussian kernel and S is the set of all points at which people are located. The training and evaluation was performed on NVIDIA DGX GPU in Pytorch Docker. λ was set to 0.0001 in (3). Adam optimization with a learning rate of 0.00001 and momentum of 0.9 was used to train the model. Additionally, for the classification (high-level prior) stage, to account for the imbalanced datasets, the losses for each class were weighted based on the number of samples available for that particular class. The training took approximately 2 days for 2000 epochs.

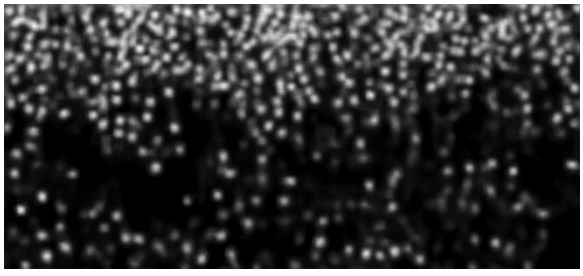
```
EPOCH: 1996, MAE: 9.5, MSE: 17.6
BEST MAE: 9.0, BEST MSE: 16.9, BEST MODEL: cmtl_shtechA_1372.h5
epoch: 1997, step    0, Time: 0.0000s, gt_cnt: 26.4, et_cnt: 35.9
epoch: 1997, step  500, Time: 0.0000s, gt_cnt: 45.8, et_cnt: 48.5
epoch: 1997, step 1000, Time: 0.0000s, gt_cnt:  5.8, et_cnt:  3.0
epoch: 1997, step 1500, Time: 0.0000s, gt_cnt: 201.7, et_cnt: 199.2
epoch: 1997, step 2000, Time: 0.0000s, gt_cnt:  7.6, et_cnt: 12.4
epoch: 1998, step    0, Time: 0.0000s, gt_cnt: 26.4, et_cnt: 31.9
epoch: 1998, step  500, Time: 0.0000s, gt_cnt: 45.8, et_cnt: 46.6
epoch: 1998, step 1000, Time: 0.0000s, gt_cnt:  5.8, et_cnt:  3.8
epoch: 1998, step 1500, Time: 0.0000s, gt_cnt: 201.7, et_cnt: 194.6
epoch: 1998, step 2000, Time: 0.0000s, gt_cnt:  7.6, et_cnt: 14.2
EPOCH: 1998, MAE: 10.1, MSE: 18.7
BEST MAE: 9.0, BEST MSE: 16.9, BEST MODEL: cmtl_shtechA_1372.h5
```

```
epoch: 1999, step    0, Time: 0.0000s, gt_cnt: 26.4, et_cnt: 31.8
epoch: 1999, step  500, Time: 0.0000s, gt_cnt: 45.8, et_cnt: 48.5
epoch: 1999, step 1000, Time: 0.0000s, gt_cnt:  5.8, et_cnt:  3.6
epoch: 1999, step 1500, Time: 0.0000s, gt_cnt: 201.7, et_cnt: 179.3
epoch: 1999, step 2000, Time: 0.0000s, gt_cnt:  7.6, et_cnt: 13.3
epoch: 2000, step    0, Time: 0.0000s, gt_cnt: 26.4, et_cnt: 33.5
epoch: 2000, step  500, Time: 0.0000s, gt_cnt: 45.8, et_cnt: 47.3
epoch: 2000, step 1000, Time: 0.0000s, gt_cnt:  5.8, et_cnt:  2.7
epoch: 2000, step 1500, Time: 0.0000s, gt_cnt: 201.7, et_cnt: 194.1
epoch: 2000, step 2000, Time: 0.0000s, gt_cnt:  7.6, et_cnt: 14.0
EPOCH: 2000, MAE: 9.5, MSE: 17.7
BEST MAE: 9.0, BEST MSE: 16.9, BEST MODEL: cmtl_shtechA_1372.h5
root@15ef8e3b24b5:/home/dgxuser122/data/Crowd Estimation using CCMTL#
[prince te0:docker*
```

UCF CC 50 dataset

The UCF CC 50 is an extremely challenging dataset introduced by Idrees et al. [10]. The dataset contains 50

Ground Truth: 1328



Predicted Count: 1064

Density estimation results using proposed method on S dataset. (a) Input (b) Output Density map

	Part A		Part B	
Method	MAE	MSE	MAE	MSE
MCNN [32]	110.2	173.2	26.4	41.3
Single stage CNN 1	130.4	190.9	29.3	40.5
Proposed method	9.0	16.9	18.9	28.3

Table 1: Comparison results: Estimation errors on the ShanghaiTech dataset. The proposed method achieves lower error compared to existing approaches involving multi column CNNs and sophisticated density maps.

Method	MAE	MSE
Idrees et al	419.5	541.6
Zhang et. al	467.0	498.5
MCNN	377.6	509.1
Proposed method	290.5	360.8

Comparison results: Estimation errors on the UCF_50 dataset.

VI. CONCLUSION

In this paper, we presented a multi-task cascaded CNN network for density map estimation. By learning to classify the crowd count into various groups, we are able to incorporate a high level prior into the network which enables it to learn globally relevant discriminative features thereby accounting for large count variations in the dataset. Additionally, we employed fractionally strided convolutional layers at the end so as to account for the loss of details due to max-pooling layers in the earlier stages there by allowing us to regress on full resolution density maps. The entire cascade was trained in an end-to-end fashion. Extensive experiments performed on challenging datasets and comparison with recent state of-the-art approaches demonstrated the significant improvements achieved by the proposed method.

VII. REFRENCES

1). A. Bansal and K. Venkatesh. People counting in high density crowds from still images. arXiv preprint arXiv:1507.08445, 2015.

2). L. Boominathan, S. S. Kruthiventi, and R. V. Babu. Crowdnet: A deep convolutional network for dense crowd counting. In Proceedings of the 2016 ACM on Multimedia Conference, pages 640–644. ACM, 2016.

3). K. Kang and X. Wang. Fully convolutional neural networks for crowd segmentation. arXiv preprint arXiv:1411.4464, 2014.

4). T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan. Crowded scene analysis: A survey. IEEE Transactions on Circuits and Systems for Video Technology, 25(3):367–386, 2015.

5). Research Paper: https://arxiv.org/abs/1707.09605

6). Wikipedia https://en.wikipedia.org/wiki/Kernel_density_estimation

7). ShanghaiTeech and UCF_50

https://www.crcv.ucf.edu/data/ucf-cc-50/

https://www.dropbox.com/s/fipgjqxl7uj8hd5/ShanghaiTech.zip?dl=0

8). H. Idrees, I. Saleemi, C. Seibert, and M. Shah. Multi-source multi-scale counting in extremely dense crowd . IEEE CVPR, pages 2547–2554, 2013.